Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions*

Kazuhiko Minematsu

NEC Corporation, Japan
k-minematsu@ah.jp.nec.com

Abstract. This paper proposes a new scheme for authenticated encryption (AE) which is typically realized as a blockcipher mode of operation. The proposed scheme has attractive features for fast and compact operation. When it is realized with a blockcipher, it requires one blockcipher call to process one input block (i.e. rate-1), and uses the encryption function of the blockcipher for both encryption and decryption. Moreover, the scheme enables one-pass, parallel operation under two-block partition. The proposed scheme thus attains similar characteristics as the seminal OCB mode, without using the inverse blockcipher. The key idea of our proposal is a novel usage of two-round Feistel permutation, where the round functions are derived from the theory of tweakable blockcipher. We also provide basic software results, and describe some ideas on using a non-invertible primitive, such as a keyed hash function.

Keywords: Authenticated Encryption, Blockcipher Mode, Pseudorandom Function, OCB.

1 Introduction

Authenticated encryption. Authenticated encryption, AE for short, is a method to simultaneously provide message confidentiality and integrity (authentication) using a symmetric-key cryptographic function. Although a secure AE function can be basically obtained by an adequate composition of secure encryption and message authentication [10, 23], this requires at least two independent keys, and the composition methods in practice (say, AES + HMAC in TLS) frequently deviate from what proved to be secure [31]. Considering this situation, there have been numerous efforts devoted to efficient, one-key constructions. Among many approaches to AE, blockcipher mode of operation is one of the most popular ones. We have CCM [2], GCM [3], EAX [11], OCB [24, 33, 35] and the predecessors [18, 22], and CCFB [27], to name a few. We have some standards, such as NIST SP 800-38C (CCM) and 38D (GCM), and ISO/IEC 19772 [4].

This paper presents a new AE mode using a blockcipher, or more generally, a pseudorandom function (PRF). Our proposal has a number of desirable features for fast and compact operations. Specifically, when the underlying n-bit

* A corresponding ePrint report is available at http://eprint.iacr.org/2013/628.

blockcipher is E_K (where K denotes the key), the properties of our proposal can be summarized as follows.

- The key is one blockcipher key, K.
- Encryption and decryption can be done by the encryption function of E_K.
- For s-bit input, the number of E_K calls is $\lceil s/n \rceil + 2$, i.e., rate-1 processing, for both encryption and decryption.
- On-line, one-pass, and parallel encryption and decryption, under two-block partition.
- Provable security up to about $2^{n/2}$ input blocks, based on the assumption that E_K is a pseudorandom function (PRF) or a pseudorandom permutation (PRP).

These features are realized with a novel usage of two-round Feistel permutation, where internal round functions are PRFs with input masking. From this we call our proposal OTR, for Offset Two-round. Table 1 provides a summary of properties of popular AE modes and ours, which shows that OTR attains similar characteristics as the seminal OCB mode, without using the inverse blockcipher. The proposed scheme generates input masks to E_K using GF(2^n) constant multiplications. This technique is called GF doubling [33], which is a quite popular tool for mode design. However, our core idea is rather generic and thus allows other masking methods. We also remark that Liting et al.'s iFeed mode [39] has similar properties to ours, without introducing 2-block partition. However, its decryption is inherently serial, and it seems that a formal security proof has not been presented so far. In return for these attractive features, one potential drawback of OTR is that it inherently needs two-block partition (though the message itself can be of any length in bits), which implies more state memories required than that of OCB. 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.

We also warn that the security is proved for the standard nonce-respecting adversary [34], i.e. the encryption never processes duplicate nonces (or initial vectors), see Section 2.2. Some recent proposals have a provable security under nonce-reusing adversary, or even security without nonce (called on-line encryption) [5,17]. However we do not claim any security guarantee for such adversaries.

Benefits of inverse-freeness. The use of blockcipher inversion, as in OCB, has mainly two drawbacks, as discussed by Iwata and Yasuda [21]. The first is efficiency. The integration of encryption and decryption functions increases size, e.g. footprint of hardware, or memory of software (See Section 6). Moreover, some ciphers have unequal speed for enc/dec. For AES, decryption is slower than encryption on some, typically constrained, platforms. For example, an AES implementation on Atmel AVR by Osvik et al. [30] has about 45% slower decryption than encryption. This property is the initial design choice [15], in preference of encryption-only mode, e.g., CTR, OFB, and CFB. IDEA is another example, where decryption is exceptionally slower than encryption on microcon-

Table 1. A comparison of AE modes. Calls denotes the number of calls for m-block message and a-block header and one-block nonce, without constants.

Mode	Calls	On-line	Parallel	Primitive
CCM [2]	$a + 2m$	no	no	E
GCM [3]	m [E] and $a + m$ [Mul]	yes	yes	$E, \mathrm{Mul}^{\dagger}$
EAX [11]	$a + 2m$	yes	no	E
OCB [24, 33, 35]	$a + m$	yes	yes	E, E^{-1}
CCFB [27]	$a + cm$ for some $1 < c^{\ddagger}$	yes	no	E
OTR	$a + m$	yes¶	yes	E

† $\mathrm{GF}(2^n)$ multiplication
‡ Security degrades as c approaches 1
¶ two-block partition

trollers [32]. The uneven performance figures of blockcipher enc/dec functions is undesirable in practice, when the mode uses both functions.

The second is 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 [24, 33, 35], though a recent work of Aoki and Yasuda [7] showed a relaxation on the security condition for OCB without tag truncation. In contrast, when the mode uses only E, the security assumption is relaxed to PRP or PRF.

In addition, the inverse-freeness allows instantiations using non-blockcipher primitives, such as a hash function. Some basic ideas on this direction are explained in Section 7.4.

Hardware assistance. We remark that some software platforms have hardware-assisted blockcipher, most notably AES instructions called AESNI in Intel and AMD CPUs. AESNI enables the same performance for AES encryption and decryption. Therefore, when our proposal uses AESNI, the performance would be roughly similar to that of OCB-AES with AESNI, though the increased number of states may degrade the result. We have other SW platforms where hardware AES is available but decryption is slower (e.g., [19]). Basically, the value of our proposal is *not* to provide the fastest operation on modern CPUs, instead, to increase the availability of OCB-like performance for various platforms, using single algorithm.

2 Preliminaries

2.1 Basic Notations

Let $\mathbb{N} = \{1, 2, \ldots, \}$, and let $\{0,1\}^*$ be the set of all finite-length binary strings, including the empty string ε. The bit length of a binary string X is denoted by $|X|$, and let $|X|_a \stackrel{\mathrm{def}}{=} \max\{\lceil |X|/a \rceil, 1\}$. Here, if $X = \varepsilon$ we have $|X|_a = 1$ for

any $a \geq 1$ and $|X| = 0$. A concatenation of $X, Y \in \{0,1\}^*$ is written as $X\|Y$ or simply XY. A sequence of a zeros is denoted by 0^a. For $k \geq 1$, we denote $\bigcup_{i=1}^{k}\{0,1\}^i$ by $\{0,1\}^{\leq k}$. For $X \in \{0,1\}^*$, let $(X[1], \ldots, X[x]) \xleftarrow{n} X$ denote the n-bit block partitioning of X, i.e., $X[1]\|X[2]\|\ldots\|X[x] = X$ where $x = |X|_n$, and $|X[i]| = n$ for $i < x$ and $|X[x]| \leq n$. If $X = \varepsilon$ the parsing with any $n \geq 1$ makes $x = 1$, $X[1] = \varepsilon$. 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 a finite set $\mathcal{X}$, if X is uniformly chosen from $\mathcal{X}$ we write $X \xleftarrow{\$} \mathcal{X}$. We assume $X \oplus Y$ is ε if X or Y is ε. For a binary string X with $0 \leq |X| \leq n$, $\underline{X}$ denotes the padding written as $X\|1\|0^{n-|X|-1}$. When $|X| = n$, $\underline{X}$ denotes X.

For keyed function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with key $K \in \mathcal{K}$, we may simply write $F_K : \mathcal{X} \to \mathcal{Y}$ if key space is obvious, or even write as F if being keyed with K is obvious. If $E_K : \mathcal{X} \to \mathcal{X}$ is a keyed permutation, or a blockcipher, E_K is a permutation over $\mathcal{X}$ for every $K \in \mathcal{K}$. Its inverse is denoted by E_K^{-1}. A keyed function may have an additional parameter called tweak, in the sense of Liskov, Rivest and Wagner [25]. It is called a tweakable keyed function and written as $\widetilde{F} : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{Y}$ or $\widetilde{F}_K : \mathcal{T} \times \mathcal{X} \to \mathcal{Y}$, where $\mathcal{T}$ denotes the space of tweaks. Instead of writing $\widetilde{F}_K(T, X)$, we may write as $\widetilde{F}_K^{\langle T \rangle}(X)$. A tweakable keyed permutation, or a tweakable blockcipher (TBC), is defined analogously by requiring that every combination of (T, K) produces a permutation over $\mathcal{X}$.

Galois Field. An n-bit string X may be viewed as an element of $\mathrm{GF}(2^n)$ by taking X as a coefficient vector of a polynomial in $\mathrm{GF}(2^n)$. We write $2X$ to denote the multiplication of 2 and X over $\mathrm{GF}(2^n)$, where 2 denotes the generator of the field $\mathrm{GF}(2^n)$. This operation is called *doubling*. We also write $3X$ and $4X$ to denote $2X \oplus X$ and $2(2X)$. The doubling is efficiently implemented by one-bit shift with conditional XOR of a constant, and frequently used as a tool to build efficient blockcipher modes, e.g. [11, 20, 33].

2.2 Random Function and Pseudorandom Function

Let $\mathrm{Func}(n, m)$ be the set of all functions $\{0,1\}^n \to \{0,1\}^m$. In addition, let $\mathrm{Perm}(n)$ be the set of all permutations over $\{0,1\}^n$. A uniform random function (URF) having n-bit input and m-bit output is uniformly distributed over $\mathrm{Func}(n, m)$. It is denoted by $\mathsf{R} \xleftarrow{\$} \mathrm{Func}(n, m)$. An n-bit uniform random permutation (URP), denoted by P, is similarly defined as $\mathsf{P} \xleftarrow{\$} \mathrm{Perm}(n)$.

We also define tweakable URF and URP. Let $\mathcal{T}$ be a set of tweak and $\mathrm{Func}^{\mathcal{T}}(n, m)$ be a set of functions $\mathcal{T} \times \{0,1\}^n \to \{0,1\}^m$. A tweakable URF with tweak $T \in \mathcal{T}$, and n-bit input, m-bit output is written as $\widetilde{\mathsf{R}} \xleftarrow{\$} \mathrm{Func}^{\mathcal{T}}(n, m)$. Note that if $\mathcal{T} = \{0,1\}^t$, $\mathrm{Func}^{\mathcal{T}}(n, m)$ has the same cardinality as $\mathrm{Func}(n+t, m)$, hence $\widetilde{\mathsf{R}}$ is simply realized with URF of $(n+t)$-bit input. In addition, let $\mathrm{Perm}^{\mathcal{T}}(n)$ be a set of functions $\mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ such that, for any $f \in \mathrm{Perm}^{\mathcal{T}}(n)$ and $t \in \mathcal{T}$, $f(t, *)$ is a permutation. A tweakable n-bit URP with tweak $T \in \mathcal{T}$ is defined as $\widetilde{\mathsf{P}} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(n)$. We also define a URF having variable input length (VIL), denoted by $\mathsf{R}^{\infty} : \{0,1\}^* \to \{0,1\}^n$. This can be realized by stateful lazy sampling.

PRF. For c oracles, $O_1, O_2, \ldots, O_c$, we write $\mathcal{A}^{O_1,O_2,\ldots,O_c}$ to represent the adversary $\mathcal{A}$ accessing these c oracles in an arbitrarily order. If O and O' are oracles having the same input and output domains, we say they are compatible. Let $F_K : \{0,1\}^n \to \{0,1\}^m$ and $G_{K'} : \{0,1\}^n \to \{0,1\}^m$ be two compatible keyed functions, with $K \in \mathcal{K}$ and $K' \in \mathcal{K}'$ (key spaces are not necessarily the same). Let $\mathcal{A}$ be an adversary trying distinguish them using chosen-plaintext queries. Then the advantage of $\mathcal{A}$ is defined as

$$\mathtt{Adv}^{\mathtt{cpa}}_{F_K,G_{K'}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{F_K} \Rightarrow 1] - \Pr[K' \xleftarrow{\$} \mathcal{K}' : \mathcal{A}^{G_{K'}} \Rightarrow 1].$$

The above definition can be naturally extended to the case when $G_{K'}$ is a URF, $\mathsf{R} \xleftarrow{\$} \mathrm{Func}(n,m)$. We have

$$\mathtt{Adv}^{\mathtt{prf}}_{F_K}(\mathcal{A}) \stackrel{\text{def}}{=} \mathtt{Adv}^{\mathtt{cpa}}_{F_K,\mathsf{R}}(\mathcal{A}).$$

If F_K is a VIL function we define $\mathtt{Adv}^{\mathtt{prf}}_{F_K}(\mathcal{A})$ as $\mathtt{Adv}^{\mathtt{cpa}}_{F_K,\mathsf{R}^\infty}(\mathcal{A})$. Similarly, for tweakable keyed function $\widetilde{F}_K : \mathcal{T} \times \{0,1\}^n \to \{0,1\}^m$ and $\widetilde{\mathsf{R}} \xleftarrow{\$} \mathrm{Func}^{\mathcal{T}}(n,m)$, we have

$$\mathtt{Adv}^{\mathtt{prf}}_{\widetilde{F}_K}(\mathcal{A}) \stackrel{\text{def}}{=} \mathtt{Adv}^{\mathtt{cpa}}_{\widetilde{F}_K,\widetilde{\mathsf{R}}}(\mathcal{A}).$$

We stress that $\mathcal{A}$ in the above is allowed to choose tweaks, arbitrarily and adaptively. By convention we say F_K is a pseudorandom function (PRF) if $\mathtt{Adv}^{\mathtt{prf}}_{F_K}(\mathcal{A})$ is small (though the formal definition requires F_K to be a function family). Similarly we say F_K is a pseudorandom permutation (PRP) if $\mathtt{Adv}^{\mathtt{prp}}_{F_K}(\mathcal{A}) = \mathtt{Adv}^{\mathtt{cpa}}_{F_K,\mathsf{P}}(\mathcal{A})$ is small and F_K is invertible. A VIL-PRF is defined in a similar way.

2.3 Definition of Authenticated Encryption

Following [11, 34], we define nonce-based AE, or more formally, AE with assosiated data, called AEAD. We then introduce two security notions, privacy and authenticity, to model AE security.

Definition. Let $\mathsf{AE}[\tau]$ be an AE having τ-bit tag, where the encryption and decryption algorithms are $\mathsf{AE}\text{-}\mathcal{E}_\tau$ and $\mathsf{AE}\text{-}\mathcal{D}_\tau$. They are keyed functions. Besides the key, the input to $\mathsf{AE}\text{-}\mathcal{E}_\tau$ consists of a nonce $N \in \mathcal{N}_{ae}$, a header (or associated data) $A \in \mathcal{A}_{ae}$, and a plaintext $M \in \mathcal{M}_{ae}$. The output consists of $C \in \mathcal{M}_{ae}$ and $T \in \{0,1\}^\tau$, where $|C| = |M|$. The tuple (N, A, C, T) will be sent to the receiver. The decryption function is denoted by $\mathsf{AE}\text{-}\mathcal{D}_\tau$. It takes $(N, A, C, T) \in \mathcal{N}_{ae} \times \mathcal{A}_{ae} \times \mathcal{M}_{ae} \times \{0,1\}^\tau$, and outputs a plaintext M with $|M| = |C|$ if input is determined as valid, or error symbol $\perp$ if determined as invalid.

Security. A PRIV-adversary $\mathcal{A}$ against $\mathsf{AE}[\tau]$ accesses $\mathsf{AE}\text{-}\mathcal{E}_\tau$, where the i-th query consists of nonce N_i, header A_i, and plaintext M_i. 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 $(N, A, M) \in$

$\mathcal{N}_{ae} \times \mathcal{A}_{ae} \times \mathcal{M}_{ae}$ and returns $(C, T) \xleftarrow{\$} \{0,1\}^{|M|} \times \{0,1\}^{\tau}$. The privacy notion for $\mathcal{A}$ is defined as

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

An AUTH-adversary $\mathcal{A}$ against $\mathsf{AE}[\tau]$ accesses $\mathsf{AE}\text{-}\mathcal{E}_\tau$ and $\mathsf{AE}\text{-}\mathcal{D}_\tau$, 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

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathsf{AE}[\tau]}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{AE}\text{-}\mathcal{E}_\tau, \mathsf{AE}\text{-}\mathcal{D}_\tau} \text{ forges }], \tag{2}$$

where $\mathcal{A}$ forges if $\mathsf{AE}\text{-}\mathcal{D}_\tau$ returns a bit string (other than $\perp$) for a decryption query (N'_i, A'_i, C'_i, T'_i) for some $1 \le i \le q_v$ such that $(N'_i, A'_i, C'_i, T'_i) \neq (N_j, A_j, C_j, T_j)$ for all $1 \le j \le q$. 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 $i' \neq i$.

Moreover, when F_K and $G_{K'}$ are compatible with $\mathsf{AE}\text{-}\mathcal{E}_\tau$, let $\mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{F,G}(\mathcal{A})$ be the same function as $\mathtt{Adv}^{\mathtt{cpa}}_{F,G}(\mathcal{A})$ but $\mathcal{A}$ is restricted to be nonce-respecting. Note that $\mathtt{Adv}^{\mathtt{priv}}_{\mathsf{AE}[\tau]}(\mathcal{A}) = \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{\mathsf{AE}\text{-}\mathcal{E}_\tau,\$}(\mathcal{A})$ holds for any nonce-respecting $\mathcal{A}$. Let $\mathbf{F} = (F^e_K, F^d_K)$ and $\mathbf{G} = (G^e_{K'}, G^d_{K'})$ be the pairs of encryption and decryption functions that are compatible with $(\mathsf{AE}\text{-}\mathcal{E}_\tau, \mathsf{AE}\text{-}\mathcal{D}_\tau)$. We define

$$\mathtt{Adv}^{\mathtt{cca\text{-}nr}}_{\mathbf{F},\mathbf{G}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{F^e_K, F^d_K} \Rightarrow 1] - \Pr[K' \xleftarrow{\$} \mathcal{K}' : \mathcal{A}^{G^e_{K'}, G^d_{K'}} \Rightarrow 1], \tag{3}$$

where $\mathcal{A}$ is assumed to be nonce-respecting for encryption queries. Then we have

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathsf{AE}[\tau]}(\mathcal{A}) \le \mathtt{Adv}^{\mathtt{cca\text{-}nr}}_{\mathsf{AE}[\tau],\mathsf{AE}'[\tau]}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{auth}}_{\mathsf{AE}'[\tau]}(\mathcal{A}) \tag{4}$$

for any AE scheme $\mathsf{AE}'[\tau]$ and any AUTH-adversary $\mathcal{A}$.

3 Specification of OTR

We present an AE scheme based on an $E_K : \{0,1\}^n \to \{0,1\}^n$, which is denoted by OTR$[E, \tau]$, where $\tau \in \{1, \ldots, n\}$ denotes the length of tag. The encryption function and decryption function of OTR$[E, \tau]$ are denoted by OTR-$\mathcal{E}_{E,\tau}$ and OTR-$\mathcal{D}_{E,\tau}$. Here OTR-$\mathcal{E}_{E,\tau}$ (OTR-$\mathcal{D}_{E,\tau}$) has the same interface as $\mathsf{AE}\text{-}\mathcal{E}_\tau$ ($\mathsf{AE}\text{-}\mathcal{D}_\tau$) of Section 2.3, with nonce space $\mathcal{N}_{ae} = \{0,1\}^{\le n-1} \setminus \{\varepsilon\}$, header space $\mathcal{A}_{ae} = \{0,1\}^*$, message space $\mathcal{M}_{ae} = \{0,1\}^*$, and tag space $\{0,1\}^\tau$. The functions OTR-$\mathcal{E}_{E,\tau}$ and OTR-$\mathcal{D}_{E,\tau}$ are further decomposed into the encryption and decryption cores, EF_E, DF_E, and the authentication core, AF_E. Figs. 1 and

2 depict the scheme. As shown by Fig. 2, OTR consists of two-round Feistel permutations using a blockcipher taking a distinct input mask in each round. To authenticate the plaintext a check sum is computed for the right part of two-round Feistel (namely the even plaintext blocks), and the tag is derived from encrypting the check sum with an input mask. The overall structure has a similarity to OCB, and the function AF_E is a variant of PMAC [33].

4 Security Bounds

We provide the security bounds of OTR. Here we assume the underlying blockcipher is an n-bit URP, P. The bounds when the underlying blockcipher is a PRP are easily derived from our bounds, using a standard technique, thus omitted.

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

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{OTR}[\mathsf{P},\tau]}(\mathcal{A}) \leq \frac{6\sigma^2_{\mathtt{priv}}}{2^n}$$

holds for $\sigma_{\mathtt{priv}} = q + \sigma_A + \sigma_M$.

Theorem 2. *Fix $\tau \in \{1, \ldots, n\}$. For any AUTH-adversary $\mathcal{A}$ with parameter $(q, q_v, \sigma_A, \sigma_M, \sigma_{A'}, \sigma_{C'})$,*

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{OTR}[\mathsf{P},\tau]}(\mathcal{A}) \leq \frac{6\sigma^2_{\mathtt{auth}}}{2^n} + \frac{q_v}{2^\tau}$$

holds for $\sigma_{\mathtt{auth}} = q + q_v + \sigma_A + \sigma_M + \sigma_{A'} + \sigma_{C'}$.

5 Proofs of Theorems 1 and 2

Overview. For the limited space we here explain the basic proof steps of Theorems 1 and 2, with some intuitions. Full proofs will appear at the full version of this paper. The proofs consist of two steps, where in the first step we interpret OTR as a mode of TBC and in the second step we prove the indistinguishability between the tweakable URF and the TBC used in OTR. This structure is essentially the same as OCB proofs, as well as many other schemes based on TBC.

First Step: TBC-based Design. In the first step, we define an AE scheme denoted by $\mathbb{OTR}'[\tau]$. It is compatible with $\mathrm{OTR}[E, \tau]$ and uses a tweakable n-bit URF, $\widetilde{\mathsf{R}} : \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$, and an independent VIL-URF, $\mathsf{R}^\infty : \{0,1\}^* \to \{0,1\}^n$, Here, tweak $T \in \mathcal{T}$ is written as $T = (x, i, \omega) \in \mathcal{N}_{ae} \times \mathbb{N} \times \Omega$, where $\Omega \stackrel{\mathrm{def}}{=} \{\mathtt{f}, \mathtt{s}, \mathtt{a}_1, \mathtt{a}_2, \mathtt{b}_1, \mathtt{b}_2, \mathtt{h}, \mathtt{g}_1, \mathtt{g}_2\}$. The values $\mathtt{h}, \mathtt{g}_1, \mathtt{g}_2$ will not be used until the next step. Here $\mathbb{OTR}'[\tau]$ consists of encryption core $\mathbb{OTR}'$-$\mathcal{E}_\tau$ and decryption core $\mathbb{OTR}'$-$\mathcal{D}_\tau$. The definition of $\mathbb{OTR}'$ is in Fig. 3. Counterparts to EF and DF are denoted by $\mathbb{EF}$ and $\mathbb{DF}$, also shown in Fig. 3. The bounds of $\mathbb{OTR}'$ are in the following theorem. The proof of Theorem 3 will be given in the full version.

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

1. $(C, TE) \leftarrow \mathrm{EF}_E(N, M)$
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-$\mathcal{D}_{E,\tau}(N, A, C, T)$

1. $(M, TE) \leftarrow \mathrm{DF}_E(N, C)$
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 $\mathrm{EF}_E(N, M)$

1. $\Sigma \leftarrow 0^n$
2. $\delta \leftarrow E(\underline{N}), L \leftarrow 4\delta$
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. $\quad C[2i] \leftarrow E(L \oplus \delta \oplus C[2i-1]) \oplus M[2i-1]$
7. $\quad \Sigma \leftarrow \Sigma \oplus M[2i]$
8. $\quad L \leftarrow 2L$
9. **if** m **is even**
10. $\quad L^* \leftarrow L \oplus \delta$
11. $\quad Z \leftarrow E(L \oplus M[m-1])$
12. $\quad C[m] \leftarrow \mathtt{msb}_{|M[m]|}(Z) \oplus M[m]$
13. $\quad C[m-1] \leftarrow E(L^* \oplus \underline{C[m]}) \oplus M[m-1]$
14. $\quad \Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$
15. **if** m **is odd**
16. $\quad L^* \leftarrow L$
17. $\quad C[m] \leftarrow \mathtt{msb}_{|M[m]|}(E(L^*)) \oplus M[m]$
18. $\quad \Sigma \leftarrow \Sigma \oplus \underline{M[m]}$
19. **if** $|M[m]| \neq n$ **then** $TE \leftarrow E(3L^* \oplus \Sigma)$
20. **else** $TE \leftarrow E(3L^* \oplus \delta \oplus \Sigma)$
21. $C \leftarrow (C[1], \ldots, C[m])$
22. **return** (C, TE)

Algorithm $\mathrm{DF}_E(N, C)$

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

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

1. $\Xi \leftarrow 0^n$
2. $\gamma \leftarrow E(0^n), Q \leftarrow 4\gamma$
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** $TA \leftarrow E(Q \oplus \gamma \oplus \Xi)$
9. **else** $TA \leftarrow E(Q \oplus 2\gamma \oplus \Xi)$
10. **return** TA

Fig. 1. The encryption and decryption algorithms of OTR with n-bit blockcipher E. Tag size is $0 < \tau \leq n$, and $\underline{X}$ denotes the 10^* padding of X (See Section 2.1).

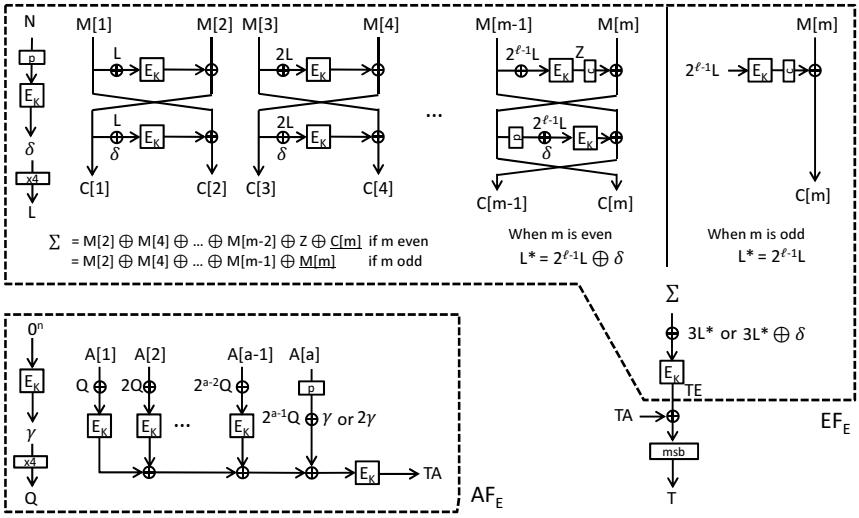

Fig. 2. Encryption of OTR. The p box denotes the 10^* padding of input X ($\underline{X}$), and the c box denotes the $\mathtt{msb}_i$ function.

Theorem 3. *Fix $\tau \in \{1, \ldots, n\}$. For any PRIV-adversary $\mathcal{A}$,*

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathbb{OTR}'[\tau]}(\mathcal{A}) = 0.$$

Moreover, for any AUTH-adversary $\mathcal{A}$ using q encryption queries and q_v decryption queries,

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathbb{OTR}'[\tau]}(\mathcal{A}) \leq \frac{2q_v}{2^n} + \frac{q_v}{2^\tau}.$$

Proof Intuition. To understand Theorem 3, there are two important properties of a two-round Feistel permutation, denoted by $\phi_{f_1,f_2} : \{0,1\}^{2n} \to \{0,1\}^{2n}$. Here $\phi_{f_1,f_2}(X[1], X[2]) = (Y[1], Y[2])$ where $Y[1] = f_1(X[1]) \oplus X[2]$ and $Y[2] = f_2(Y[1]) \oplus X[1]$ and f_1 and f_2 are independent n-bit URFs. Then we have the followings.

Property 1. For any $(X[1], X[2]) \in \{0,1\}^{2n}$, $\phi_{f_1,f_2}(X[1], X[2])$ is uniformly random.

Property 2. Let $(Y[1], Y[2]) = \phi_{f_1,f_2}(X[1], X[2])$, and let $(Y'[1], Y'[2])$ be a function of $(X[1], X[2], Y[1], Y[2])$ satisfying $(Y'[1], Y'[2]) \neq (Y[1], Y[2])$. Then $X'[2]$, where $(X'[1], X'[2]) = \phi^{-1}_{f_1,f_2}(Y'[1], Y'[2])$, is uniform unless the event $\mathtt{Bad}_1 : X[1] = X'[1]$ occurs, which has the probability at most $1/2^n$.

Property 1 is simple because f_1 and f_2 are independent and the output of ϕ consists of those of f_1 and f_2. Property 2 needs some cares. It holds because if $X[1] \neq X'[1] = f_2(Y'[1]) \oplus Y'[2]$, $f_1(X'[1])$ is distributed uniformly random,

independent of all other variables, and this makes $X'[2] = f_1(X'[1]) \oplus Y'[1]$ completely random. The $\mathtt{Bad}_1$ event has probability $1/2^n$ when $Y'[1] \neq Y'[1]$, and otherwise 0. Note that $(X[1], X[2], Y[1], Y[2])$ reveals corresponding I/O pairs of f_1 and f_2, however this does not help gain the probability of $\mathtt{Bad}_1$.

Intuitively, the privacy bound of Theorem 3 is simply obtained by the fact that all TBC calls in the game has distinct tweaks and all output blocks contain at least one TBC output with unique tweak. Combined with Property 1, this makes all output blocks perfectly random, hence the privacy bound is 0. For the authenticity bound, suppose adversary $\mathcal{A}$ performs an encryption query (N, A, M) and obtains (C, T), and then performs a decryption query (N', A', C', T') for some $C \neq C'$ with $|C| = |C'|$, with $(N', A') = (N, A)$. This implies that there exists at least one chunk ($2n$-bit block) of C' different from the corresponding chunk in C, and from Property 2, the right half of the corresponding decrypted plaintext chunk is completely random, unless $\mathtt{Bad}_1$ occurs. There is another chance for the adversary to win, i.e. the checksum collision $\mathtt{Bad}_2 : \Sigma' = \Sigma$, which has probability $1/2^n$ provided $\mathtt{Bad}_1$ did not happen. Hence we have $\Pr[\mathtt{Bad}_1 \cup \mathtt{Bad}_2] \leq \Pr[\mathtt{Bad}_1] + \Pr[\mathtt{Bad}_2 | \overline{\mathtt{Bad}_1}] \leq 2/2^n$. When both events did not happen (i.e. given $\overline{\mathtt{Bad}_1 \cup \mathtt{Bad}_2}$), the final chance is to successfully guess the tag, where the probability is clearly bounded by $1/2^\tau$ because different checksums yield independent tags. Hence the authenticity bound is $2/2^n + 1/2^\tau$ for any $\mathcal{A}$ using $q_v = 1$ decryption query (of course we need to consider the existence of other encryption queries and many other cases for (N', A', C', T') as well, however the above bound holds for all cases). Finally we use a well-known result of Bellare, Goldreich and Mityagin [9] to obtain $2q_v/2^n + q_v/2^\tau$ for any $q_v \geq 1$.

Second Step: Analysis of TBC. In the bottom of Fig. 3 we define a TBC, $\widetilde{G}[\mathsf{P}]^{\langle N,i,\omega\rangle}(X)$, where (N, i, ω) is a tweak. It uses an n-bit URP P. We remark that $\widetilde{G}[\mathsf{P}]$ slightly abuse N as it allows $N = 0^n$. Hence the tweak space is $\mathcal{T}' = \{\mathcal{N}_{ae} \cup \{0^n\}\} \times \mathbb{N} \times \Omega$. For tweaks that do not appear in Fig. 3, we let them as undefined. Let $\widetilde{\mathsf{R}}$ be a tweakable URF compatible with $\widetilde{G}[\mathsf{P}]$. Then we have the following proposition and lemma.

Proposition 1. *If $\mathbb{EF}_{\widetilde{\mathsf{R}}}$ ($\mathbb{DF}_{\widetilde{\mathsf{R}}}$) uses $\widetilde{G}[\mathsf{P}]$ instead of $\widetilde{\mathsf{R}}$, we obtain EF_{P} (DF_{P}).*

Lemma 1. *For any $\mathcal{A}$ with q queries, $\mathtt{Adv}^{\mathtt{cpa}}_{\widetilde{G}[\mathsf{P}],\widetilde{\mathsf{R}}}(\mathcal{A}) \leq 5q^2/2^n$.*

Fig. 3 shows a function $\mathbb{AF}_{\widetilde{\mathsf{R}}} : \{0,1\}^* \to \{0,1\}^n$. The internal $\widetilde{\mathsf{R}}$ is a tweakable URF compatible with $\widetilde{G}[\mathsf{P}]$. It is again easy to observe that if $\mathbb{AF}_{\widetilde{\mathsf{R}}}$ uses $\widetilde{G}[\mathsf{P}]$ instead of $\widetilde{\mathsf{R}}$, we obtain AF_{P}. We provide the security bound for $\mathbb{AF}_{\widetilde{\mathsf{R}}}$, which is as follows.

Lemma 2. *For any $\mathcal{A}$ with σ input blocks, we have $\mathtt{Adv}^{\mathtt{prf}}_{\mathbb{AF}_{\widetilde{\mathsf{R}}}}(\mathcal{A}) \leq \sigma^2/2^{n+1}$.*

The proofs of Lemmas 1 and 2 are almost the same as XE mode and (a part of) PMAC proofs [33] and will be given in the full version.

Deriving Bounds. Let $\mathbb{OTR}[\tau]$ be an AE consisting of $\mathbb{EF}_{\widetilde{\mathsf{R}}}$, $\mathbb{DF}_{\widetilde{\mathsf{R}}}$, and $\mathbb{AF}_{\widetilde{\mathsf{R}}}$ shown in Fig. 3. For privacy notion, there exist adversaries $\mathcal{B}$ against $\mathbb{AF}_{\widetilde{\mathsf{R}}}$ with σ_A input blocks, and $\mathcal{C}$ against $\widetilde{G}[\mathsf{P}]$ with $\sigma_{\mathtt{priv}}$ queries, satisfying

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{OTR}[\mathsf{P},\tau]}(\mathcal{A}) \le \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{\mathrm{OTR}[\mathsf{P},\tau],\mathbb{OTR}[\tau]}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{\mathbb{OTR}[\tau],\mathbb{OTR}'[\tau]}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{\mathbb{OTR}'[\tau],\$}(\mathcal{A}) \tag{5}$$

$$\le \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{\mathrm{OTR}[\mathsf{P},\tau],\mathbb{OTR}[\tau]}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{AF}_{\widetilde{\mathsf{R}}},\mathsf{R}^\infty}(\mathcal{B}) + \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{\mathbb{OTR}'[\tau],\$}(\mathcal{A}) \tag{6}$$

$$\le \mathtt{Adv}^{\mathtt{cpa}}_{\widetilde{G}[\mathsf{P}],\widetilde{\mathsf{R}}}(\mathcal{C}) + \frac{\sigma_A^2}{2^{n+1}} \tag{7}$$

$$\le \frac{5\sigma^2_{\mathtt{priv}}}{2^n} + \frac{\sigma_A^2}{2^{n+1}} \tag{8}$$

$$\le \frac{6\sigma^2_{\mathtt{priv}}}{2^n}. \tag{9}$$

where the third inequality follows from Proposition 1, Lemma 2, and Theorem 3, and the fourth inequality follows from Lemma 1. Similarly, for authenticity notion, there exist $\mathcal{B}$ against $\mathbb{AF}_{\widetilde{\mathsf{R}}}$ with $\sigma_A + \sigma_{A'}$ input blocks, and $\mathcal{C}$ against $\widetilde{G}[\mathsf{P}]$ with $\sigma_{\mathtt{auth}}$ queries, satisfying

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{OTR}[\mathsf{P},\tau]}(\mathcal{A}) \le \mathtt{Adv}^{\mathtt{cca\text{-}nr}}_{\mathrm{OTR}[\mathsf{P},\tau],\mathbb{OTR}'[\tau]}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{auth}}_{\mathbb{OTR}'[\tau]}(\mathcal{A}) \tag{10}$$

$$\le \mathtt{Adv}^{\mathtt{cca\text{-}nr}}_{\mathrm{OTR}[\mathsf{P},\tau],\mathbb{OTR}[\tau]}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{cca\text{-}nr}}_{\mathbb{OTR}[\tau],\mathbb{OTR}'[\tau]}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{auth}}_{\mathbb{OTR}'[\tau]}(\mathcal{A}) \tag{11}$$

$$\le \mathtt{Adv}^{\mathtt{cca\text{-}nr}}_{\mathrm{OTR}[\mathsf{P},\tau],\mathbb{OTR}[\tau]}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{AF}_{\widetilde{\mathsf{R}}},\mathsf{R}^\infty}(\mathcal{B}) + \mathtt{Adv}^{\mathtt{auth}}_{\mathbb{OTR}'[\tau]}(\mathcal{A}) \tag{12}$$

$$\le \mathtt{Adv}^{\mathtt{cpa}}_{\widetilde{G}[\mathsf{P}],\widetilde{\mathsf{R}}}(\mathcal{C}) + \frac{(\sigma_A + \sigma_{A'})^2}{2^{n+1}} + \frac{2q_v}{2^n} + \frac{q_v}{2^\tau} \tag{13}$$

$$\le \frac{5\sigma^2_{\mathtt{auth}}}{2^n} + \frac{(\sigma_A + \sigma_{A'})^2}{2^{n+1}} + \frac{2\sigma_{A'}}{2^n} + \frac{q_v}{2^\tau} \tag{14}$$

$$\le \frac{6\sigma^2_{\mathtt{auth}}}{2^n} + \frac{q_v}{2^\tau}, \tag{15}$$

where the fourth inequality follows from Proposition 1, Lemma 2, and Theorem 3, and the fifth inequality follows from Lemma 1. This concludes the proof.

6 Experimental Results on Software

We implemented OTR on software. The purpose of this implementation is not to provide a fast code, but to see the effect of inverse-freeness in an experimental environment. We wrote a reference-like AES C code that takes byte arrays and uses 4Kbyte tables for combined S-box and Mixcolumn lookup, so-called T-tables. AES decryption of our code is slightly slower than encryption (see Table 2). We then wrote pure C code of OTR using the above AES code. All components, e.g. XOR of blocks and GF doubling, are byte-wise codes. For comparison we also wrote a C code of OCB2 [33] in the same manner, which is similar to a reference code by Krovetz [1].

We ran both codes on an x86 PC (Core i7 3770, Ivy bridge, 3.4GHz) with 64-bit Windows 7. We used Visual C++ 2012 (VC12) to obtain 32-bit and 64-bit executables and used GCC 4.7.1 for 32-bit executables, with option `-O2`. We measured speed for 4Kbyte messages and one-block header. We also tested the same code on an ARM board (Cortex-A8 1GHz) using GCC 4.7.3 with `-O2` option. Their speed figures in cycles per byte[1] are shown in the upper part of Table 2. For both OTR and OCB2, we can observe a noticeable slowdown from raw AES, however, OTR still receives the benefit of faster AES encryption. Another metric is the size, which is shown in the lower part of Table 2. For OTR we can remove the inverse T-tables and inverse S-box from AES code, as they are not needed for AES encryption, resulting in smaller AES objects.

We also measured the performance of these codes when AES is implemented using AESNI (on the Core i7 machine, using VC12). We simply substituted T-table AES with single-block AES routine using AESNI. In addition, two common functions to OCB2 and OTR, namely XOR of two 16-byte blocks and GF doubling, are substituted with SIMD intrinsic codes. Other byte-wise functions are unchanged. On our machine single-block AES ran at around 4.5 to 5.5 cycles per byte, for both encryption and decryption. Table 3 shows the results. It looks interesting, in that, although we did not write a parallel AESNI routine, we could observe the obvious effect of AESNI parallelism via compiler. Notably, both OTR and OCB2 achieved about 2 cycles per byte for 4K data, and OCB2 is slight faster as expected. We think further optimization of OTR as well as OCB2 would be possible if we use parallel AES routine with a careful register handling.

These experiments, though quite naive, imply OTR's good performance under multiple platforms with a simple code. Of course, optimized implementations for various platforms are interesting future topics.

7 Remarks

7.1 Remove Inverse from OCB

The abstract structure of OTR has a similarity to OCB, however, removing inverse is not a trivial task. Roughly, in OCB, each plaintext block is given to the ECB mode of an n-bit TBC $\widetilde{E}_K$ [25], namely $C[i] = \widetilde{E}_K^{\langle T \rangle}(M[i])$, where tweak T consists of nonce N and other parameters, based on a blockcipher E_K. The OCB decryption uses the inversion of TBC, $\widetilde{E}_K^{-1}$, and the security proof requires that $\widetilde{E}_K$ is a tweakable SPRP, i.e. $(\widetilde{E}_K, \widetilde{E}_K^{-1})$ and $(\widetilde{\mathsf{P}}, \widetilde{\mathsf{P}}^{-1})$ are hard to distinguish when $\widetilde{\mathsf{P}} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(n)$. Since $\widetilde{E}_K^{-1}$ needs a computation of E_K^{-1}, a natural way to remove E_K^{-1} from OCB is to compose $\widetilde{E}_K$ from a PRP or a PRF. For example we can do this by using a $2n$-bit 4-round Feistel cipher as $\widetilde{E}_K$, based on an n-bit PRF, F_K. Then, the resulting mode (of F_K) is inverse-free and provably secure,

[1] As we were unable to use cycle counter in the ARM device, the measurement of ARM was based on a timer.

Table 2. Reference implementation results of OTR and OCB2. (Upper) Speed in cycles per byte. (Lower) Object size in Kbyte.

	x86			ARM
Algorithm	VC12(32-bit)	VC12(64-bit)	gcc 4.7.1(32-bit)	gcc 4.7.3
OTR Enc	27.59	18.94	22.02	69.88
OTR Dec	27.56	18.99	22.2	69.78
OCB2 Enc	27.38	19.93	22.69	71.22
OCB2 Dec	30.86	25.43	34.29	76.16
AES Enc	18.29	12.98	15.9	54.38
AES Dec	22.28	18.36	26.64	58.14

	x86			ARM
Object	VC12(32-bit)	VC12(64-bit)	gcc 4.7.1(32-bit)	gcc 4.7.3
OTR.o	19.9	21.3	5.4	5.9
OCB2.o	20.5	21.7	4.6	5.3
AES_Enc.o	20.2	20.7	6.7	7.1
AES_EncDec.o	45.4	46.2	17.3	17.9
OTR Total	40.1	42.0	12.1	13.0
OCB2 Total	65.9	67.9	21.9	23.2

Table 3. Performance of codes with single-block AES routine using AES-NI. Data `x` denotes the plaintext length in bytes, and `a`/`b` denotes `a` (`b`) cycles per byte in 32-bit (64-bit) VC12 compilation.

Data (byte)	128	512	1024	2048	4096
OTR Enc	6.01/5.43	3.32/3.16	2.85/2.74	2.66/2.51	2.49/2.40
OTR Dec	7.22/5.60	3.81/3.15	3.06/2.72	2.79/2.51	2.59/2.39
OCB2 Enc	6.39/5.60	3.26/2.76	2.81/2.26	2.53/2.02	2.37/1.90
OCB2 Dec	6.36/5.86	3.04/2.80	2.59/2.26	2.28/2.03	2.11/1.91

since 4-round Feistel cipher is an SPRP, as shown by Luby and Rackoff [26] (it is easy to turn a SPRP into a tweakable SPRP). However, we then need four F_K calls per two blocks, i.e. the rate is degraded to two. Considering this, the two-round Feistel is seemingly a bad choice, since it even fails to provide a (tweakable) PRP. As explained in Section 5, the crucial observation is that, the encryption of two-round Feistel in OTR is invoked only once for each tweak, and that the authenticity needs only an n-bit unpredictable value in the decryption, rather than $2n$ bits. Two-round Feistel fulfills these requirements, which makes OTR provably secure.

7.2 Design Rationale for Masking

We remark that using the same mask for the two round functions, i.e. using $2^i L$ for the first and second rounds of a two-round Feistel, does not work. This is because Property 2 of Section 5 does not hold anymore since the two-round Feistel becomes an involution. Once you query $(X[1], X[2])$ and receive

$(Y[1], Y[2]) = \phi_{f_1,f_2}(X[1], X[2])$, you know $X'[2] = Y[2]$ always holds (where $(X'[1], X'[2]) = \phi^{-1}_{f_1,f_2}(Y'[1], Y'[2])$), when $(Y'[1], Y'[2]) = (X[1], X[2])$. This implies that the adversary can control the checksum value in the decryption, hence breaks authenticity.

We also remark that the masks for EF_E depend on N, hence do not allow precomputation. In contrast the latest OCB3 allows mask precomputation by using $E_K(0^n)$ [24]. The reason is that we want our scheme not to generate $E_K(0^n)$ for header-less usage (i.e. when A is always empty). As a result our scheme has a rather similar structure as OCB2 and an AEAD mode based on OCB2, called AEM [33]. Recent studies reported that the doubling is not too slow [6], hence we employ on-the-fly doubling as a practical masking option.

7.3 Comparison with Other Inverse-free Modes

Section 6 only considers a comparison with OCB. Here we provide a basic comparison with other modes, in particular those not using the blockcipher inverse. Table 1 shows examples of such inverse-free modes. Among them, CCM, GCM, and EAX are rate-2, assuming the speed of field multiplication in GCM is comparable with blockcipher encryption. At least in theory, OTR is faster for sufficiently long messages for its rate-1 computation. For CCFB, the rate c is a variable satisfying $1 < c$ and $c \approx 1$ is impractical for weak security guarantee[2]. 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 usually needs large memories. At the same time, a potential disadvantage of OTR is the complexity introduced by the two-round Feistel, such as a limited on-line/parallel capability, and a slight complex design compared with simple designs reusing existing modes like CTR, CFB, and CMAC.

7.4 Other Instantiations

As the core idea of our proposal is general, it allows various instantiations, by seeing $\mathbb{OTR}$ or $\mathbb{OTR}'$ as a prototype. What we need is just to instantiate $\widetilde{\mathsf{R}}$ accepting n-bit input and tweak (N, i, ω), and producing n-bit output. While we employ GF doubling, one can use a different masking scheme, such as Gray code [24,35], or word-oriented LFSR [14,24,38], or bit-rotation of a special prime length [28]. Moreover, we can use non-invertible cryptographic primitives, typically a Hash-based PRF such as HMAC, or a permutation of Keccak [12] with Even-Mansour conversion [16] for implementing a keyed permutation. In the latter case the resulting scheme does not need an inversion of the permutation, which is different from the permutation-based OCB described at [29], and there is no output loss like "capacity" bits of SpongeWrap [13]. In these settings, it is possible that the underlying primitive accepts longer input than output. Then

[2] More formally, the security bound is roughly $\sigma^2/2^{n/c}$ for privacy and $(\sigma^2/2^{n/c} + 1/2^{n(1-(1/c))})$ for authenticity, with single decryption query and σ total blocks.

a simple tweaking method by tweak prepending can be an option. For example we take SipHash [8], which is a VIL-PRF with 64-bit output. A SipHash-based scheme would be obtained by replacing $\widetilde{\mathsf{R}}^{\langle N,i,\omega\rangle}(*)$ of $\mathbb{OTR}'$ (Fig. 3) with $\mathrm{SipHash}_K(N\|i\|\omega\|*)$, and replacing $\mathsf{R}^{\infty}(*)$ with $\mathrm{SipHash}_K(0^n\|0\|\mathtt{h}\|*)$, accompanied with an appropriate input encoding. As SipHash has an iterative structure, a caching of an internal value allows efficient computation of $\mathrm{SipHash}_K(N\|i\|\omega\|X)$ from $\mathrm{SipHash}_K(N\|i'\|\omega'\|X')$. We remark that this scheme has roughly 64-bit security. The proof is trivial from Theorem 3, combined with the assumption that SipHash is a VIL-PRF.

8 Concluding Remarks

We have presented an authenticated encryption scheme using a PRF. This scheme enables rate-1, on-line, and parallel processing for both encryption and decryption. The core idea of our proposal is to use two-round Feistel with input masking, combined with a message check sum. As a concrete instantiation we provide a blockcipher mode, called OTR, entirely based on a blockcipher encryption function, which may be seen as an "inverse-free" version of OCB. Our proposal has a higher complexity than OCB outside the blockcipher, hence it will not outperform OCB when the blockcipher enc/dec functions are natively supported and equally fast (say CPU with AESNI), despite the relaxed security assumption. Still, our proposal would be useful for various other environments where the use of blockcipher inverse imposes a non-negligible cost, or simply when the available crypto function is not invertible.

Acknowledgments. The author would like to thank anonymous reviewers for careful reading and invaluable suggestions, which greatly improved the presentation of the paper. The author also would like to thank Tetsu Iwata for fruitful discussions, and Sumio Morioka and Tomoyasu Suzaki for useful comments on implementation aspects.

References

1. Reference C code of OCB2, `http://www.cs.ucdavis.edu/~rogaway/ocb/code-2.0.htm/`
2. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality . NIST Special Publication 800-38C (2004), national Institute of Standards and Technology.
3. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D (2007), national Institute of Standards and Technology.
4. Information Technology - Security techniques - Authenticated encryption, ISO/IEC 19772:2009. International Standard ISO/IEC 19772 (2009)

5. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and Authenticated Online Ciphers. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT (1). Lecture Notes in Computer Science, vol. 8269, pp. 424–443. Springer (2013)
6. 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/`
7. Aoki, K., Yasuda, K.: The Security of the OCB Mode of Operation without the SPRP Assumption. In: Susilo and Reyhanitabar [37], pp. 202–220
8. Aumasson, J.P., Bernstein, D.J.: SipHash: A Fast Short-Input PRF. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 7668, pp. 489–508. Springer (2012)
9. Bellare, M., Goldreich, O., Mityagin, A.: The Power of Verification Queries in Message Authentication and Authenticated Encryption. Cryptology ePrint Archive, Report 2004/309 (2004), `http://eprint.iacr.org/`
10. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 1976, pp. 531–545. Springer (2000)
11. Bellare, M., Rogaway, P., Wagner, D.: The EAX Mode of Operation. In: Roy and Meier [36], pp. 389–407
12. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak SHA-3 submission (January 2011), `http://keccak.noekeon.org/`
13. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7118, pp. 320–337. Springer (2011)
14. Chakraborty, D., Sarkar, P.: A general construction of tweakable block ciphers and different modes of operations. IEEE Transactions on Information Theory 54(5), 1991–2006 (2008)
15. Daemen, J., Rijmen, V.: AES Proposal: Rijndael (1999)
16. Even, S., Mansour, Y.: A Construction of a Cipher From a Single Pseudorandom Permutation. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.) ASIACRYPT. Lecture Notes in Computer Science, vol. 739, pp. 210–224. Springer (1991)
17. Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In: Canteaut, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 7549, pp. 196–215. Springer (2012)
18. Gligor, V.D., Donescu, P.: Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes. In: Matsui, M. (ed.) FSE. Lecture Notes in Computer Science, vol. 2355, pp. 92–108. Springer (2001)
19. 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)
20. 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)
21. 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)

22. Jutla, C.S.: Encryption Modes with Almost Free Message Integrity. In: Pfitzmann, B. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 2045, pp. 529–544. Springer (2001)
23. Krawczyk, H.: The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In: Kilian, J. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 2139, pp. 310–331. Springer (2001)
24. 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)
25. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable Block Ciphers. In: Yung, M. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 2442, pp. 31–46. Springer (2002)
26. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM J. Comput. 17(2), 373–386 (1988)
27. 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)
28. Minematsu, K.: A Short Universal Hash Function from Bit Rotation, and Applications to Blockcipher Modes. In: Susilo and Reyhanitabar [37], pp. 221–238
29. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering Generic Composition. DIAC 2013: Directions in Authenticated Ciphers (2013), available from `http://2013.diac.cr.yp.to/`
30. 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)
31. Paterson, K.: Authenticated Encryption in TLS. DIAC 2013: Directions in Authenticated Ciphers (2013), available from `http://2013.diac.cr.yp.to/`
32. Rinne, S.: Performance Analysis of Contemporary Light-Weight Cryptographic Algorithms on a Smart Card Microcontroller. SPEED – Software Performance Enhancement for Encryption and Decryption (2007), available from `http://www.hyperelliptic.org/SPEED/start07.html`
33. 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)
34. Rogaway, P.: Nonce-Based Symmetric Encryption. In: Roy and Meier [36], pp. 348–359
35. 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)
36. 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)
37. Susilo, W., Reyhanitabar, R. (eds.): Provable Security - 7th International Conference, ProvSec 2013, Melaka, Malaysia, October 23-25, 2013. Proceedings, Lecture Notes in Computer Science, vol. 8209. Springer (2013)
38. Zeng, G., Han, W., He, K.: High Efficiency Feedback Shift Register: σ-LFSR. Cryptology ePrint Archive, Report 2007/114 (2007), `http://eprint.iacr.org/`
39. Zhang, L., Han, S., Wu, W., Wang, P.: iFeed: the Input-Feed AE Modes. Rump Session of FSE 2013 (2013), slides from `http://fse.2013.rump.cr.yp.to/`

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

$(C, TE) \leftarrow \mathbb{EF}_{\widetilde{\mathsf{R}}}(N, M)$
if $A \neq \varepsilon$ **then** $TA \leftarrow \mathsf{R}^\infty(A)$
else $TA \leftarrow 0^n$
$T \leftarrow \mathtt{msb}_\tau(TE \oplus TA)$
return (C, T)

Algorithm $\mathbb{OTR}'\text{-}\mathcal{D}_\tau(N, A, C, T)$

$(M, TE) \leftarrow \mathbb{DF}_{\widetilde{\mathsf{R}}}(N, C)$
if $A \neq \varepsilon$ **then** $TA \leftarrow \mathsf{R}^\infty(A)$
else $TA \leftarrow 0^n$
$\widehat{T} \leftarrow \mathtt{msb}_\tau(TE \oplus TA)$
if $\widehat{T} = T$ **return** M
else return $\perp$

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

$(C, TE) \leftarrow \mathbb{EF}_{\widetilde{\mathsf{R}}}(N, M)$
if $A \neq \varepsilon$ **then** $TA \leftarrow \mathbb{AF}_{\widetilde{\mathsf{R}}}(A)$
else $TA \leftarrow 0^n$
$T \leftarrow \mathtt{msb}_\tau(TE \oplus TA)$
return (C, T)

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

$(M, TE) \leftarrow \mathbb{DF}_{\widetilde{\mathsf{R}}}(N, C)$
if $A \neq \varepsilon$ **then** $TA \leftarrow \mathbb{AF}_{\widetilde{\mathsf{R}}}(A)$
else $TA \leftarrow 0^n$
$\widehat{T} \leftarrow \mathtt{msb}_\tau(TE \oplus TA)$
if $\widehat{T} = T$ **return** M
else return $\perp$

Algorithm $\mathbb{EF}_{\widetilde{\mathsf{R}}}(N, M)$

$\Sigma \leftarrow 0^n$
$(M[1], \ldots, M[m]) \xleftarrow{n} M$
$\ell \leftarrow \lceil m/2 \rceil$
for $i = 1$ **to** $\ell - 1$ **do**
  $C[2i-1] \leftarrow \widetilde{\mathsf{R}}^{\langle N, i, \mathtt{f}\rangle}(M[2i-1]) \oplus M[2i]$
  $C[2i] \leftarrow \widetilde{\mathsf{R}}^{\langle N, i, \mathtt{s}\rangle}(C[2i-1]) \oplus M[2i-1]$
  $\Sigma \leftarrow \Sigma \oplus M[2i]$
if m **is even**
  $Z \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{f}\rangle}(M[m-1])$
  $C[m] \leftarrow \mathtt{msb}_{|M[m]|}(Z) \oplus M[m]$
  $C[m-1] \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{s}\rangle}(\underline{C[m]}) \oplus M[m-1]$
  $\Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$
  if $|M[m]| \neq n$
    then $TE \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{a}_1\rangle}(\Sigma)$
    else $TE \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{a}_2\rangle}(\Sigma)$
if m **is odd**
  $C[m] \leftarrow \mathtt{msb}_{|M[m]|}(\widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{f}\rangle}(0^n)) \oplus M[m]$
  $\Sigma \leftarrow \Sigma \oplus \underline{M[m]}$
  if $|M[m]| \neq n$
    then $TE \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{b}_1\rangle}(\Sigma)$
    else $TE \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{b}_2\rangle}(\Sigma)$
$C \leftarrow (C[1], \ldots, C[m])$
return (C, TE)

Algorithm $\mathbb{DF}_{\widetilde{\mathsf{R}}}(N, C)$

$\Sigma \leftarrow 0^n$
$(C[1], \ldots, C[m]) \xleftarrow{n} C$
$\ell \leftarrow \lceil m/2 \rceil$
for $i = 1$ **to** $\ell - 1$ **do**
  $M[2i-1] \leftarrow \widetilde{\mathsf{R}}^{\langle N, i, \mathtt{s}\rangle}(C[2i-1]) \oplus C[2i]$
  $M[2i] \leftarrow \widetilde{\mathsf{R}}^{\langle N, i, \mathtt{f}\rangle}(M[2i-1]) \oplus C[2i-1]$
  $\Sigma \leftarrow \Sigma \oplus M[2i]$
if m **is even**
  $M[m-1] \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{s}\rangle}(\underline{C[m]}) \oplus C[m-1]$
  $Z \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{f}\rangle}(M[m-1])$
  $M[m] \leftarrow \mathtt{msb}_{|C[m]|}(Z) \oplus C[m]$
  $\Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$
  if $|M[m]| \neq n$
    then $TE \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{a}_1\rangle}(\Sigma)$
    else $TE \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{a}_2\rangle}(\Sigma)$
if m **is odd**
  $M[m] \leftarrow \mathtt{msb}_{|C[m]|}(\widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{f}\rangle}(0^n)) \oplus C[m]$
  $\Sigma \leftarrow \Sigma \oplus \underline{M[m]}$
  if $|C[m]| \neq n$
    then $TE \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{b}_1\rangle}(\Sigma)$
    else $TE \leftarrow \widetilde{\mathsf{R}}^{\langle N, \ell, \mathtt{b}_2\rangle}(\Sigma)$
$M \leftarrow (M[1], \ldots, M[m])$
return (M, TE)

Algorithm $\mathbb{AF}_{\widetilde{\mathsf{R}}}(A)$

$\Xi \leftarrow 0^n$
$(A[1], \ldots, A[a]) \xleftarrow{n} A$
for $i = 1$ **to** $a - 1$ **do**
  $\Xi \leftarrow \Xi \oplus \widetilde{\mathsf{R}}^{\langle 0^n, i, \mathtt{h}\rangle}(A[i])$
$\Xi \leftarrow \Xi \oplus \underline{A[a]}$
if $|A[a]| \neq n$ **then** $TA \leftarrow \widetilde{\mathsf{R}}^{\langle 0^n, a, \mathtt{g}_1\rangle}(\Xi)$
else $TA \leftarrow \widetilde{\mathsf{R}}^{\langle 0^n, a, \mathtt{g}_2\rangle}(\Xi)$
return TA

Algorithm $\widetilde{G}[\mathsf{P}]^{\langle N, i, \omega\rangle}(X)$

Preprocessing: $\gamma \leftarrow \mathsf{P}(0^n)$, $Q \leftarrow 4\gamma$
if $N \neq 0^n$ **then** $\delta \leftarrow \mathsf{P}(\underline{N})$, $L \leftarrow 4\delta$
  switch ω
  Case $\mathtt{f}$ **:** $\Delta \leftarrow 2^{i-1}L$
  Case $\mathtt{s}$ **:** $\Delta \leftarrow 2^{i-1}L \oplus \delta$
  Case $\mathtt{a}_1$ **:** $\Delta \leftarrow 3(2^{i-1}L \oplus \delta)$
  Case $\mathtt{a}_2$ **:** $\Delta \leftarrow 3(2^{i-1}L \oplus \delta) \oplus \delta$
  Case $\mathtt{b}_1$ **:** $\Delta \leftarrow 2^{i-1}3L$
  Case $\mathtt{b}_2$ **:** $\Delta \leftarrow 2^{i-1}3L \oplus \delta$
else switch ω
Case $\mathtt{h}$ **:** $\Delta \leftarrow 2^{i-1}Q$
Case $\mathtt{g}_1$ **:** $\Delta \leftarrow 2^{i-1}Q \oplus \gamma$
Case $\mathtt{g}_2$ **:** $\Delta \leftarrow 2^{i-1}Q \oplus 2\gamma$
$Y \leftarrow \mathsf{P}(\Delta \oplus X)$
return Y

Fig. 3. The components of $\mathbb{OTR}'[\tau]$ and $\mathbb{OTR}[\tau]$. An exception is $\widetilde{G}[\mathsf{P}]$, which is a tweakable PRP implicitly used by $\mathrm{OTR}[\mathsf{P}, \tau]$.