Boosting OMD for Almost Free Authentication of Associated Data

Reza Reyhanitabar, Serge Vaudenay, and Damian Vizár

EPFL, Switzerland

Abstract. We propose *pure* OMD (p-OMD) as a new variant of the Offset Merkle-Damgård (OMD) authenticated encryption scheme. Our new scheme inherits all desirable security features of OMD while having a more compact structure and providing higher efficiency. The original OMD scheme, as submitted to the CAESAR competition, couples a single pass of a variant of the Merkle-Damgård (MD) iteration with the counter-based XOR MAC algorithm to provide privacy and authenticity. Our improved p-OMD scheme dispenses with the XOR MAC algorithm and is *purely* based on the MD iteration; hence, the name "pure" OMD. To process a message of ℓ blocks and associated data of a blocks, OMD needs $\ell + a + 2$ calls to the compression function while p-OMD only requires $\max\{\ell, a\} + 2$ calls. Therefore, for a typical case where $\ell \geq a$, p-OMD makes just $\ell + 2$ calls to the compression function; that is, associated data is processed almost freely compared to OMD. We prove the security of p-OMD under the same standard assumption (pseudorandomness of the compression function) as made in OMD; moreover, the security bound for p-OMD is the same as that of OMD, showing that the modifications made to boost the performance are without any loss of security.

Keywords: Authenticated encryption, OMD, associated data, performance, CAESAR competition.

1 Introduction

An authenticated encryption (AE) scheme provides two complementary data security goals: confidentiality (privacy) and integrity (authenticity). Traditionally, these goals were achieved by combining two cryptographic primitives, a privacy-only encryption scheme and a message authentication code (MAC)—a paradigm known as generic composition (GC) [8, 9, 21]. The notion of AE, as a desirable symmetric-key primitive in its own right, was introduced in 2000 [8, 10, 19]. Since then, security notions for AE schemes have been defined and refined [15, 23, 25–27], together with many dedicated AE designs seeking some advantages over the GC-based schemes.

AE schemes have been studied for over a decade, yet the topic remains a highly active and interesting area of research as evidenced by the currently running CAESAR competition [11]. OMD [13, 14] is one of 57 first-round CAESAR submissions, among which, at the time of writing this paper, 8 submissions are withdrawn due to major security flaws.

Among the features that OMD possesses, the following two are notably interesting and distinctive: OMD is the only CAESAR submission that is designed (as a mode of operation) based on a compression function [3], and it provides (provably) high security levels (about twice that of the AES-based submissions) when implemented with an off-the-shelf compression function such as those of the standard SHA family [2].

Instantiations of OMD using the compression functions of SHA-256 and SHA-512, called OMD-sha256 and OMD-sha512 respectively, can freely benefit from the widely-deployed optimized implementations of these primitives, e.g. [16, 17]; in particular, OMD-sha256 can take advantage of the new Intel SHA Extensions [18].

Motivated by the aforementioned appealing features of OMD, we further investigate the possibility of making algorithmic improvements to the original OMD scheme towards boosting its efficiency, while preserving or even improving its security properties. We show that there is a natural way (inspired from the work of [28]) to modify OMD to make it more compact and efficient with respect to processing associated data (AD). Our new variant of OMD—called *pure* OMD (p-OMD)—has the following features:

– **It inherits all desirable security features of OMD**. We prove the security of p-OMD under the same standard assumption (namely, pseudo-randomness of the compression function) as made in OMD. Furthermore, the proven security bounds for p-OMD are the same as those of OMD. This shows that the modifications we made to OMD, to obtain the performance-boosted variant p-OMD, are without sacrificing any security.
– **It has a more compact structure and processing AD is almost free**. The original OMD scheme couples a single pass of the MD iteration—in which the chaining values are xored with specially crafted offsets—with the counter-based XOR MAC algorithm [7] to process a message and its associated data. The p-OMD scheme dispenses with the XOR MAC algorithm and is solely based on the (masked) MD iteration. This is achieved by absorbing the associated data blocks during the core MD path rather than processing them separately by an additional XOR MAC algorithm. To encrypt a message of ℓ blocks having associated data of a blocks, OMD needs $\ell + a + 2$ calls to the compression function while p-OMD only requires $\max\{\ell, a\} + 2$ calls. That is, for a typical case where $\ell \geq a$, p-OMD makes just $\ell + 2$ calls independently of the length of AD.

We note that neither OMD nor p-OMD satisfy the nonce-reuse misuse-resistance notions defined in [15, 27]. Misuse-resistant variants of OMD are recently proposed in [22], but in these variants the encryption process is not online and they are less efficient than OMD.

A CORRECTION. In the preproceedings version of this paper at FSE 2015, we mistakenly claimed that (in addition to the efficiency advantage of pOMD compared to OMD which is the main contribution of this work) one also gets a partial level of robustness to nonce misuse with respect to the authenticity property. Tomer Ashur and Bart Mennink pointed out [4] that this claim was incorrect; that is, pOMD similar to OMD requires nonce respecting for providing security.

ORGANIZATION OF THE PAPER. Notations and prelimiary concepts are presented in Section 2. Definitions of security notions for AE schemes are reviewed in Section 3. Section 4 provides the specification of the p-OMD mode of operation. In Section 5, we provide the security analysis of p-OMD. Section 6 provides an experimental performance comparison between p-OMD and OMD.

2 Preliminaries

NOTATIONS. Let $x \xleftarrow{\$} S$ denote choosing an element x from a finite set S uniformly at random. $X \leftarrow Y$ is used for denoting the assignment statement where the value of Y is assigned to X. All strings are binary strings. The empty string is denoted by ε. The set of all strings of length n bits (for some positive integer n) is denoted as $\{0,1\}^n$, the set of all strings whose lengths are upper-bounded by L is denoted by $\{0,1\}^{\leq L}$ and the set of all strings of finite length is denoted by $\{0,1\}^*$. The notations $X||Y$ and XY both stand for the string obtained by concatenating a string Y to a string X. For an m-bit string $X = X[m-1]\cdots X[0]$ we denote the first (leftmost) bit by $\mathsf{firstbit}(X) = X[m-1]$ and the last (rightmost) bit by $\mathsf{lastbit}(X) = X[0]$. Let $X[i\cdots j] = X[i]\cdots X[j]$ denote a substring of X, for $m-1 \geq i \geq j \geq 0$; by convention we let $X[i\cdots j] = \varepsilon$ if $i < 0$ and $X[i\cdots j] = X[i\cdots 0]$ if $j < 0$.

For a non-negative integer i let $\langle i \rangle_m$ denote the binary representation of i by an m-bit string. For a bit string $X = X[m-1]\cdots X[0]$, let $\mathsf{str2num}(X) = \sum_{i=0}^{m-1} X[i]2^i$ denote the non-negative integer represented by X. Let $\mathtt{ntz}(i)$ denote the number of trailing zeros (i.e. the number of rightmost bits that are zero) in the binary representation of a positive integer i. Let 1^n0^m denote concatenation of n ones by m zeros.

We let $\mathsf{firstbits}_i(X) = X[m-1\cdots m-i]$ denote the i leftmost bits and $\mathsf{lastbits}_i(X) = X[i-1\cdots 0]$ denote the i rightmost bits of X. For two strings $X = X[m-1]\cdots X[0]$ and $Y = Y[n-1]\cdots Y[0]$ of possibly different lengths, let the notation $X \oplus Y$ denote the bitwise xor of $\mathsf{firstbits}_i(X)$ and $\mathsf{firstbits}_i(Y)$ where $i = \min\{m, n\}$. Clearly, if X and Y have the same length then $X \oplus Y$ matches the usual bitwise xor. For any string X, define $X \oplus \varepsilon = \varepsilon \oplus X = \varepsilon$.

The special symbol $\perp$ signifies both that the value of a variable or a function at some input is undefined, and an error. Let $|Z|$ denote the number of elements of Z if Z is a set, and the length of Z in bits if Z is a string. We let $|\varepsilon| = 0$. For $X \in \{0,1\}^*$ let $X_1||X_2 \cdots ||X_m \xleftarrow{b} X$ denote partitioning X into blocks X_i such that $|X_i| = b$ for $1 \le i \le m-1$ and $|X_m| \le b$; let $m = |X|_b$ denote length of X in b-bit blocks.

For a string $X = X[m-1] \cdots X[0]$, let $X \ll n$ denote the left-shift operation, where the n leftmost bits are discarded and the n vacated right bits are set to 0. We let $X \gg n$ denote the (unsigned) right-shift operation where the n rightmost bits are discarded and the n vacated left bits are set to 0. We let $X \gg_s n$ denote the *signed* right-shift operation where the n rightmost bits are discarded and the n vacated left bits are filled with the original leftmost bit (which is considered as the sign bit); for example, $1001100 \gg_s 3 = 1111001$. If the leftmost bit of X is 0 then we have $X \gg_s n = X \gg n$.

The Finite Field with 2^n Elements. Let $(GF(2^n), \oplus, .)$ denote the Galois Field with 2^n elements. An element α in $GF(2^n)$ is represented as a formal polynomial $\alpha(X) = \alpha_{n-1}X^{n-1} + \cdots + \alpha_1 X + \alpha_0$ with binary coefficients. We can assign an element $\alpha_i \in GF(2^n)$ to an integer $i \in \{0, \ldots, 2^n - 1\}$ in a natural way, similar applies for α_s and a string $s \in \{0,1\}^n$. We sometimes refer to the elements of $GF(2^n)$ directly by strings or integers, if the context does not allow ambiguity. The addition "$\oplus$" and multiplication "." of two field elements in $GF(2^n)$ are defined as usual [14]. For $GF(2^{256})$ we use $P_{256}(X) = X^{256} + X^{10} + X^5 + X^2 + 1$, and for $GF(2^{512})$ we use $P_{512}(X) = X^{512} + X^8 + X^5 + X^2 + 1$ as the irreducible polynomials used in the field multiplications. It is easy to multiply an arbitrary field element α by the element 2 (i.e. X). For example, in $GF(2^{256})$ using $P_{256}(X)$ the doubling operation can be described as follows:

$$2.\alpha = \begin{cases} \alpha \ll 1 & \text{if } \mathsf{firstbit}(\alpha) = 0 \\ (\alpha \ll 1) \oplus 0^{245}10000100101 & \text{if } \mathsf{firstbit}(\alpha) = 1 \end{cases} \quad (1)$$

$$= (\alpha \ll 1) \oplus ((\alpha \gg_s 255) \wedge 0^{245}10000100101) \quad (2)$$

We note that the results computed in (1) and (2) are the same but an implementation using (2) will not be susceptible to the timing attacks unlike one which uses (1).

Advantage Function. The insecurity of a scheme Π in regard to a security property xxx is measured using the resource parametrized function $\mathbf{Adv}^{\mathsf{xxx}}_{\Pi}(\mathbf{r}) = max_{\boldsymbol{A}}\{\mathbf{Adv}^{\mathsf{xxx}}_{\Pi}(\boldsymbol{A})\}$, where the maximum is taken over all adversaries $\boldsymbol{A}$ which use resources bounded by $\mathbf{r}$.

Let $\boldsymbol{A}$ be an adversary that returns a binary value; by $\boldsymbol{A}^{f(.)}(X) \Rightarrow 1$ we refer to the event that $\boldsymbol{A}$ on input X and access to an oracle function $f(.)$ returns 1.

Pseudorandom Functions (PRFs) and Tweakable PRFs. Let $\mathrm{Func}(m,n) = \{f : \{0,1\}^m \to \{0,1\}^n\}$ be the set of all functions from m-bit strings to n-bit strings. A random function (RF) R with m-bit input and n-bit output is a function selected uniformly at random from $\mathrm{Func}(m,n)$. We denote this by $R \xleftarrow{\$} \mathrm{Func}(m,n)$.

Let $\mathrm{Func}^{\mathcal{T}}(m,n)$ be the set of all functions $\left\{\widetilde{f} : \mathcal{T} \times \{0,1\}^m \to \{0,1\}^n\right\}$, where $\mathcal{T}$ is a set of tweaks. A tweakable RF with the tweak space $\mathcal{T}$, m-bit input and n-bit output is a map $\widetilde{R} : \mathcal{T} \times \{0,1\}^m \to \{0,1\}^n$ selected uniformly at random from $\mathrm{Func}^{\mathcal{T}}(m,n)$; i.e. $\widetilde{R} \xleftarrow{\$} \mathrm{Func}^{\mathcal{T}}(m,n)$. Clearly, if $\mathcal{T} = \{0,1\}^t$ then $|\mathrm{Func}^{\mathcal{T}}(m,n)| = |\mathrm{Func}(m+t,n)|$, and hence, $\widetilde{R}$ can be instantiated using a random function R with $(m+t)$-bit input and n-bit output. We use $\widetilde{R}^{\langle T \rangle}(.)$ and $\widetilde{R}(T,.)$ interchangeably, for every $T \in \mathcal{T}$. Notice that each tweak T names a random function $\widetilde{R}^{\langle T \rangle} : \{0,1\}^m \to \{0,1\}^n$ and distinct tweaks name distinct (independent) random functions.

Let $F : \mathcal{K} \times \{0,1\}^m \to \{0,1\}^n$ be a keyed function and let $\widetilde{F} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^m \to \{0,1\}^n$ be a keyed and tweakable function, where the key space $\mathcal{K}$ is some nonempty set. Let $F_K(.) = F(K,.)$ and $\widetilde{F}_K^{\langle T \rangle}(.) = \widetilde{F}(K,T,.)$. Let $\boldsymbol{A}$ be an adversary. Then:

$$\mathbf{Adv}^{\mathrm{prf}}_{F}(\boldsymbol{A}) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \boldsymbol{A}^{F_K(.)} \Rightarrow 1\right] - \Pr\left[R \xleftarrow{\$} \mathrm{Func}(m,n) : \boldsymbol{A}^{R(.)} \Rightarrow 1\right]$$
$$\mathbf{Adv}^{\widetilde{\mathrm{prf}}}_{\widetilde{F}}(\boldsymbol{A}) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \boldsymbol{A}^{\widetilde{F}^{(.)}_K(.)} \Rightarrow 1\right] - \Pr\left[\widetilde{R} \xleftarrow{\$} \mathrm{Func}^{\mathcal{T}}(m,n) : \boldsymbol{A}^{\widetilde{R}^{(.)}(.)} \Rightarrow 1\right]$$

The resource parametrized advantage functions are defined accordingly, considering that the adversarial resources of interest here are the time complexity (t) of the adversary and the total number of queries (q) asked by the adversary (note that we just consider fixed-input-length functions, so the lengths of queries are fixed and known). We say that F is $(t,q;\epsilon)$-PRF if $\mathbf{Adv}^{\mathrm{prf}}_{F}(t,q) \leq \epsilon$. We say that $\widetilde{F}$ is $(t,q;\epsilon)$-tweakable PRF if $\mathbf{Adv}^{\widetilde{\mathrm{prf}}}_{\widetilde{F}}(t,q) \leq \epsilon$.

3 Security Notions for AEAD

Syntax of an AEAD Scheme. A nonce-based authenticated encryption with associated data, AEAD for short, is a symmetric key scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key space $\mathcal{K}$ is some non-empty finite set. The encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \cup \{\perp\}$ takes four arguments, a secret key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N}$, an associated data (a.k.a. header data) $A \in \mathcal{A}$ and a message $M \in \mathcal{M}$, and returns either a ciphertext $\mathbb{C} \in \mathcal{C}$ or a special symbol $\perp$ indicating an error. The decryption algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ takes four arguments $(K, N, A, \mathbb{C})$ and either outputs a message $M \in \mathcal{M}$ or an error indicator $\perp$.

For correctness of the scheme, it is required that $\mathcal{D}(K, N, A, \mathbb{C}) = M$ for any $\mathbb{C}$ such that $\mathbb{C} = \mathcal{E}(K, N, A, M)$. It is also assumed that if algorithms $\mathcal{E}$ and $\mathcal{D}$ receive parameter not belonging to their specified domain of arguments they will output $\perp$. We write $\mathcal{E}_K(N, A, M) = \mathcal{E}(K, N, A, M)$ and similarly $\mathcal{D}_K(N, A, \mathbb{C}) = \mathcal{D}(K, N, A, \mathbb{C})$.

We assume that the message and associated data can be any binary string of arbitrary but finite length; i.e. $\mathcal{M} = \{0,1\}^*$ and $\mathcal{A} = \{0,1\}^*$, but the key and nonce are some fixed-length binary strings, i.e. $\mathcal{N} = \{0,1\}^{|N|}$ and $\mathcal{K} = \{0,1\}^k$, where the positive integers $|N|$ and k are respectively the nonce length and the key length of the scheme in bits. We assume that $|\mathcal{E}_K(N, A, M)| = |M| + \tau$ for some positive fixed constant τ; that is, we will have $\mathbb{C} = C||\mathsf{Tag}$ where $|C| = |M|$ and $|\mathsf{Tag}| = \tau$. We call C the core ciphertext and Tag the tag.

Nonce Respecting Adversaries. Let $\boldsymbol{A}$ be an adversary. We say that $\boldsymbol{A}$ is nonce-respecting if it never repeats a nonce in its *encryption* queries. That is, if $\boldsymbol{A}$ queries the encryption oracle $\mathcal{E}_K(\cdot,\cdot,\cdot)$ on $(N_1, A_1, M_1) \cdots (N_q, A_q, M_q)$ then $N_1, \cdots, N_q$ must be distinct.

Privacy Notion. We adopt the privacy notion called indistinguishability of ciphertext from random bits under CPA (IND$-CPA), which is defined in [26] as a stronger variant of the classical IND-CPA notion [5, 8].

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a nonce-based AEAD scheme. Let $\boldsymbol{A}$ be a nonce-respecting adversary. $\boldsymbol{A}$ is provided with an oracle which can be either a real encryption oracle $\mathcal{E}_K(\cdot,\cdot,\cdot)$ such that on input (N, A, M) returns $\mathbb{C} = \mathcal{E}_K(N, A, M)$, or a fake encryption oracle $\$(\cdot,\cdot,\cdot)$ which on any input (N, A, M) returns $|\mathbb{C}|$ fresh random bits. The advantage of $\boldsymbol{A}$ in mounting a chosen plaintext attack (CPA) against the privacy property of Π is measured as follows:

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(\boldsymbol{A}) = \Pr[K \xleftarrow{\$} \mathcal{K} : \boldsymbol{A}^{\mathcal{E}_K(\cdot,\cdot,\cdot)} \Rightarrow 1] - \Pr[\boldsymbol{A}^{\$(\cdot,\cdot,\cdot)} \Rightarrow 1].$$

Authenticity Notion. We adopt the established notion of authenticity, called integrity of ciphertext (INT-CTXT) under CCA attacks. The notion was originally defined in [8] for AE schemes and later revisited to include (authentication of AD in) AEAD schemes in [23].

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a nonce-based AEAD scheme. Let $\boldsymbol{A}$ be a nonce-respecting adversary. We stress that nonce-respecting is only regarded for the encryption queries; that is, $\boldsymbol{A}$ can repeat nonces during its decryption queries and it can also ask an encryption query with a nonce that was already used in a decryption query. Let $\mathcal{A}$ be provided with the encryption oracle $\mathcal{E}_K(\cdot,\cdot,\cdot)$ and the decryption oracle $\mathcal{D}_K(\cdot,\cdot,\cdot)$; that is, we consider adversaries that can mount chosen ciphertext attacks (CCA). We say that $\boldsymbol{A}$ forges if it makes a decryption query $(N, A, \mathbb{C})$ such that $\mathcal{D}_K(N, A, \mathbb{C}) \neq \perp$ and no previous encryption query $\mathcal{E}_K(N, A, M)$ returned $\mathbb{C}$.

$$\mathbf{Adv}^{\text{auth}}_{\Pi}(\boldsymbol{A}) = \Pr[K \xleftarrow{\$} \mathcal{K} : \boldsymbol{A}^{\mathcal{E}_K(\cdot,\cdot,\cdot),\, \mathcal{D}_K(\cdot,\cdot,\cdot)} \text{ forges}].$$

Resource parameters. Let an adversary $\boldsymbol{A}$ make encryption queries $(N^1, A^1, M^1) \cdots (N^{q_e}, A^{q_e}, M^{q_e})$ and decryption queries $(N'^1, A'^1, \mathbb{C}'^1) \cdots (N'^{q_v}, A'^{q_v}, \mathbb{C}'^{q_v})$. We define the resource parameters of $\boldsymbol{A}$ as $(t, q_e, q_v, \sigma_A, \sigma_M, \sigma_{A'}, \sigma_{\mathbb{C}'}, L_{max})$, where t is the time complexity, q_e and q_v are respectively the total number of encryption queries and decryption queries, L_{max} is the maximum length of each query in bits, $\sigma_A = \sum_{i=1}^{q_e} |A^i|$, $\sigma_M = \sum_{i=1}^{q_e} |M^i|$, $\sigma_{A'} = \sum_{i=1}^{q_v} |A'^i|$ and $\sigma_{\mathbb{C}'} = \sum_{i=1}^{q_v} (|\mathbb{C}'^i| - \tau)$.

The absence of a resource parameter will mean that the parameter is irrelevant in the context and hence omitted.

4 The p-OMD Mode of Operation

p-OMD is a mode of operation that converts a keyed compression function to an AEAD scheme. To instantiate p-OMD, one must first choose and fix a keyed compression function $F : \mathcal{K} \times (\{0,1\}^n \times \{0,1\}^m) \to \{0,1\}^n$ and a tag length $\tau \leq n$; with the key space $\mathcal{K} = \{0,1\}^k$ and $m \leq n$. Let p-OMD$[F, \tau]$ denote the p-OMD instantiated by fixing F and τ.

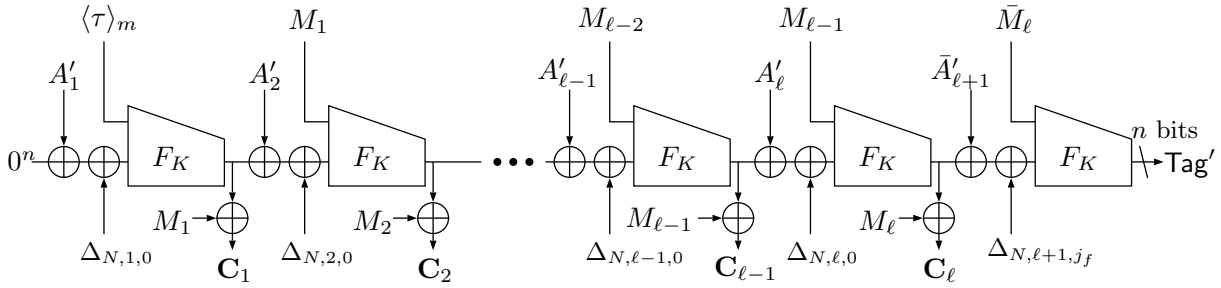
If the compression function at hand does not have a dedicated key input per se, as it is the case for standard hash functions, then a keyed compression function with $n + m$ input bits can be obtained from the keyless compression function with $n + b$ input bits by allocating k input bits for the key, such that $b = m + k$. For example, if we use the compression function of SHA-256, we have $n = 256, b = 512$ and setting $k = 256$ will give us a keyed compression function with $m = n = 256$.

Description of the Mode. The main design rationale behind p-OMD is the integration of AD processing into the same MD path that processes the message. Figure 1 shows a schematic representation of the encryption algorithm of p-OMD$[F, \tau]$. The decryption algorithm can be straightforwardly derived from the encryption algorithm with the additional verification of the authentication tag at the end of the decryption process. While the overall structure of such design is rather simple, the combined processing of the message and associated data blocks in p-OMD creates several additional possible cases, to be treated and analyzed carefully, compared to the analysis of OMD. Figure 2 provides an algorithmic description.

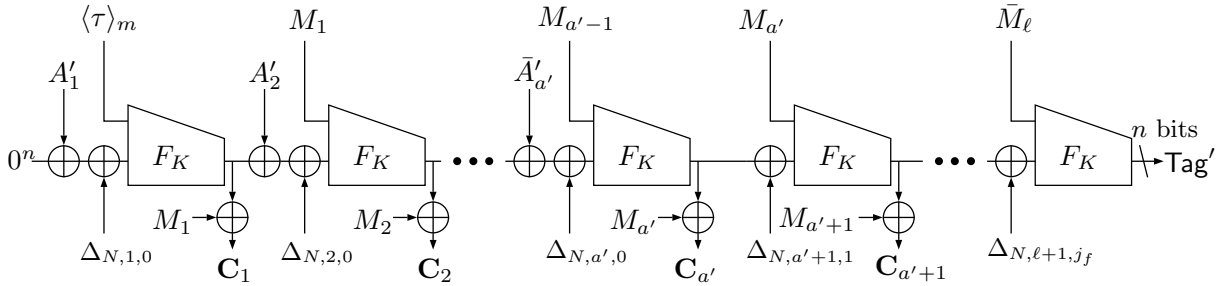
In the following we briefly explain the components that may need further clarification.

(1) **Computing $\Delta_{N,i,j}$.** As shown in Figure 1, before each call to the underlying compression function F, we xor a (key-dependent) masking value $\Delta_{N,i,j}$ to the chaining variable, where N is the nonce, the i component of the index is incremented at each call to the compression function and the j component is changed when needed (according to a pattern that will be detailed shortly). This method is known as the XE method [24] and is used for converting F to a *tweakable* function. There are different plausible ways to compute such masking values (under efficiency and security constraints) [12, 20, 24]. We adopt the Gray code based method following [20]. In the following, all multiplications (denoted by "$\cdot$") are in $GF(2^n)$.

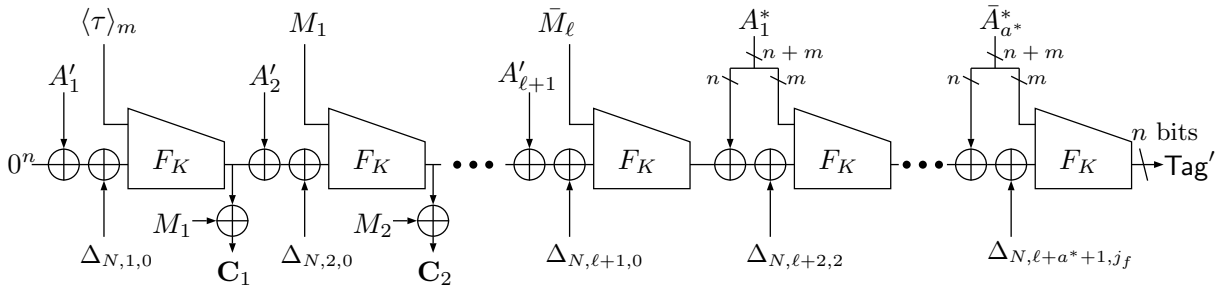
(a) Precomputation. Let $L_*(0) = 0^n$, $L_*(1) = F_K(0^n, 0^m)$ and $L_*(i) = i \cdot L_*(1)$ for $2 \leq i \leq 15$. Let $L(0) = 16 \cdot L_*(1)$ and $L(j) = 2 \cdot L(j-1)$ for $j \geq 1$. For a fast implementation the values $L_*(i)$



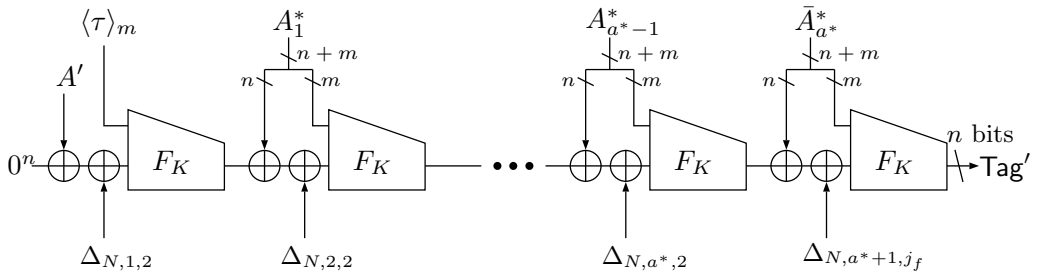
Case A: $\ell > 0$ **and** $|\mathbf{A}|_{\mathbf{n}} = \ell + 1$. Let $\bar{M}_\ell = M_\ell||10^{m-|M_\ell|-1}$ if $|M_\ell| < m$ and $\bar{M}_\ell = M_\ell$ otherwise. Let $\bar{A}'_{a'} = A'_{a'}||10^{n-|A'_{a'}|-1}$ if $|A'_{a'}| < n$ and $\bar{A}'_{a'} = A'_{a'}$ otherwise.



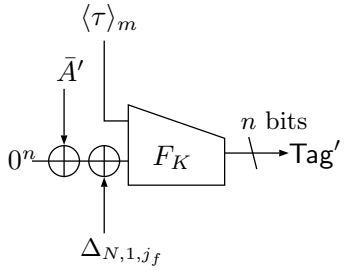
Case B: $\ell > 0$ **and** $|\mathbf{A}|_{\mathbf{n}} < \ell + 1$. Let $\bar{M}_\ell = M_\ell||10^{m-|M_\ell|-1}$ if $|M_\ell| < m$ and $\bar{M}_\ell = M_\ell$ otherwise. Let $\bar{A}'_{a'} = A'_{a'}||10^{n-|A'_{a'}|-1}$ if $|A'_{a'}| < n$ and $\bar{A}'_{a'} = A'_{a'}$ otherwise.



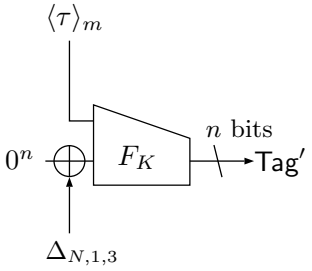
Case C: $\ell > 0$ **and** $|\mathbf{A}|_{\mathbf{n}} > \ell + 1$. Let $\bar{M}_\ell = M_\ell||10^{m-|M_\ell|-1}$ if $|M_\ell| < m$ and $\bar{M}_\ell = M_\ell$ otherwise. Let $\bar{A}^*_{a^*} = A^*_{a^*}||10^{n+m-|A^*_{a^*}|-1}$ if $|A^*_{a^*}| < n + m$ and $\bar{A}^*_{a^*} = A^*_{a^*}$ otherwise.



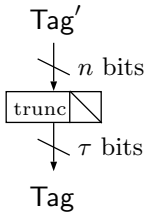
Case D: $\mathbf{M} = \varepsilon$ **and** $|\mathbf{A}| > \mathbf{n}$. Let $\bar{A}^*_{a^*} = A^*_{a^*}||10^{n+m-|A^*_{a^*}|-1}$ if $|A^*_{a^*}| < n + m$ and $\bar{A}^*_{a^*} = A^*_{a^*}$ otherwise.



Case E: $\mathbf{M} = \varepsilon$ **and** $\mathbf{0} < |\mathbf{A}| \leq \mathbf{n}$. Let $\bar{A}' = A'||10^{n-|A'|-1}$ if $|A'| < n$ and $\bar{A}' = A'$ otherwise.



Case F
$\mathbf{M} = \mathbf{A} = \varepsilon$



Obtaining the final tag.

Fig. 1. The encryption process of p-OMD$[F, \tau]$. For the details on how the parameters and masking offsets are computed consult the description in Section 4.

and $L(j)$ can be precomputed and stored in a table for $1 \le i \le 15$ and $0 \le j \le \lceil \log_2(\ell_{max}) \rceil$, where ℓ_{max} is the the bound on the maximum number of blocks in M or A. Alternatively, (if there is a memory restriction) they can be computed on-the-fly. Note that all $L_*(i)$ are linear.

(b) COMPUTATION OF THE MASKING SEQUENCE. The masking values $\Delta_{N,i,j}$ are computed sequentially as follows. Let $\Delta_{N,0,0} = F_K(N||10^{n-1-|N|}, 0^m)$. For $i \ge 1$ and $j, j' \in \{0, 1, \ldots, 15\}$: $\Delta_{N,i,j} = \Delta_{N,i-1,j'} \oplus L(\mathtt{ntz}(i)) \oplus L_*(\mathsf{str2num}(\langle j \rangle_4 \oplus \langle j' \rangle_4))$. For details on how we get this compact relation adopting the Gray code based sequence partition method, we refer to Appendix A.

```
1: Algorithm PRECOMPUTE(K)
2:     L_*(0) = 0^n
3:     L_*(1) ← F_K(0^n, 0^m)
4:     for i ← 2 to 15 do
5:         L_*(i) = i · L_*(1)
6:     L(0) ← 16 · L_*(1)
7:     for i ← 1 to ⌈log_2(ℓ_max)⌉ do
8:         L(i) = 2 · L(i − 1)
9:     return

1: Algorithm E_K(N, A, M)
2:     if |N| > n − 1 then
3:         return ⊥
4:     PARTITION(A, M)
5:     PAD(A', A*, M)
6:     Δ ← F_K(N||10^{n−1−|N|}, 0^m)
7:     Δ ← Δ ⊕ L(0)
8:     H ← 0^n; j ← 0
9:     if a' = 0 and ℓ = 0 then
10:        SWITCH(Δ, j, 3)
11:    else if a' = 0 then
12:        SWITCH(Δ, j, 1)
13:    else
14:        H ← H ⊕ A'_1
15:        if a' = 1 and a* > 0 then
16:            SWITCH(Δ, j, 2)
17:        else if a' = 1 and ℓ = 0 then
18:            SWITCH(Δ, j, 12 + j_A + j_M)
19:    H ← F_K(H ⊕ Δ, ⟨τ⟩_m)
20:    i ← 2
21:    if a' > 1 then                                  ▷ STAGE 1
22:        PROC1(M, A', H, Δ, i)
23:        if i = ℓ + 1 then
24:            C_{i−1} ← H ⊕ M_{i−1}
25:            H ← H ⊕ A'_i
26:            Δ ← Δ ⊕ L(ntz(i))
27:            if a* = 0 then
28:                SWITCH(Δ, j, 4 + j_A + j_M)
29:            H ← F_K(H ⊕ Δ, M̄_{i−1})
30:            i ← i + 1
31:    if ℓ ≥ a' then                                  ▷ STAGE 2
32:        SWITCH(Δ, j, 1)
33:        PROC2(M, H, Δ, i)
34:        C_{i−1} ← H ⊕ M_{i−1}
35:        Δ ← Δ ⊕ L(ntz(i))
36:        SWITCH(Δ, j, 8 + j_A + j_M)
37:        H ← F_K(H ⊕ Δ, M̄_{i−1})
38:        i ← i + 1
39:    else if a* > 0 then                             ▷ STAGE 3
40:        SWITCH(Δ, j, 2)
41:        PROC3(A*, H, Δ, i)
42:        Left ← A*_{a*}[n + m − 1 ··· m]
43:        Right ← A*_{a*}[m − 1 ··· 0]
44:        H ← H ⊕ Left
45:        Δ ← Δ ⊕ L(ntz(i))
46:        SWITCH(Δ, j, 12 + j_A + j_M)
47:        H ← F_K(H ⊕ Δ, Right)
48:    Tag ← H[n − 1 ··· n − τ]
49:    C ← C_1||C_2|| ··· ||C_ℓ||Tag
50:    return C
```

Fig. 2. Description of the encryption algorithm of p-OMD$[F, \tau]$. STAGE 1 processes blocks of message and AD simultaneously (**Cases A,B** and **C** in Figure 1). STAGE 2 processes only message blocks (**Case B** in Figure 1 and the case when we only have a message and no AD that is not in the Figure). STAGE 3 processes only double blocks of AD (**Cases C** and **D** in Figure 1). Note that the **Cases E** and **F** are handled outside of the three stages. Subroutines PARTITION, PAD, SWITCH and PROC1-3 are described in Figure 3.

(2) **Encryption Algorithm**: To encrypt a message $M \in \{0,1\}^*$ with associated data $A \in \{0,1\}^*$ using nonce $N \in \{0,1\}^{|N|}$ and key $K \in \{0,1\}^k$, obtaining a ciphertext $\mathbb{C} = C||\mathsf{Tag} \in \{0,1\}^{|M|+\tau}$, do the following.

(a) PARTITIONING THE MESSAGE AND ASSOCIATED DATA. The partitioning is done by the PARTITION subroutine in Figure 3. Let $M_1||M_2 \cdots M_{\ell-1}||M_\ell \xleftarrow{n} M$. Let $A'||A^* \leftarrow A$ where $A' \leftarrow A[|A| - 1 \cdots |A| - (\ell+1)n]$ and $A^* \leftarrow A[|A| - |A'| - 1 \cdots 0]$ (refer to the notations in Section 2). Let $A'_1||A'_2 \cdots A'_{a'-1}||A'_{a'} \xleftarrow{n} A'$ and $A^*_1||A^*_2 \cdots A^*_{a^*-1}||A^*_{a^*} \xleftarrow{n+m} A^*$. The string A' consists of $a' \le \ell + 1$ n-bit blocks and these blocks will be simply absorbed into the chaining variable during the message encryption. In a typical use case where the associated data is (a header) shorter than the message, we will have $A' = A$ i.e. $A^* = \varepsilon$ (**Case A** and **Case B** in Figure 1). The string A^* will be non-empty only if $|A| > (\ell+1)n$, in which case, while A^* is

being processed, there are no more message blocks to encrypt. To maximize the efficiency, we partition the string A^* into $n+m$-bit blocks so that we can make use of both of the inputs to F (see **Case C** and **Case D** in Figure 1).

(b) Processing the message and associated data. The message and associated data blocks are processed by the modified MD iteration of the keyed compression functions F as shown in Figure 1. For every call to F, the n-bit input (chaining variable) is masked by the value $\Delta_{N,i,j}$; where, the N component in the index denotes the nonce; i starts with the value $i = 1$ at the first call to F and is incremented (by one) for every call; the j component is used to separate logical parts in the encryption process as well as different types of input arguments. Appropriate use of the j component is essential for security and facilitates the analysis, as will be described in the following.

Subroutine PARTITION(A, M)
$b \leftarrow n + m$
$M_1||M_2 \cdots M_{\ell-1}||M_\ell \xleftarrow{m} M$ ▷ $(\ell = |M|_m)$
$A' \leftarrow A[|A| - 1 \cdots |A| - (\ell + 1)n]$
$A^* \leftarrow A[|A| - |A'| - 1 \cdots 0]$
$A'_1||A'_2 \cdots A'_{a'-1}||A'_{a'} \xleftarrow{n} A'$ ▷ $(a' = |A'|_n)$
$A^*_1||A^*_2 \cdots A^*_{a^*-1}||A^*_{a^*} \xleftarrow{b} A^*$ ▷ $(a^* = |A^*|_{n+m})$

Subroutine PAD(A', A^*, M)
if $|M| \bmod m \neq 0$ **then**
  $\bar{M}_\ell \leftarrow M_\ell||10^{m-|M_\ell|-1}$
  $j_M \leftarrow 1$
else
  $\bar{M}_\ell \leftarrow M_\ell$
  $j_M \leftarrow 0$
if $|A'| \bmod n \neq 0$ **then**
  $A'_{a'} \leftarrow A'_{a'}||10^{n-|A'_{a'}|-1}$
  $j_A \leftarrow 2$
else if $|A^*| \bmod n + m \neq 0$ **then**
  $A^*_{a^*} \leftarrow A^*_{a^*}||10^{n+m-|A^*_{a^*}|-1}$
  $j_A \leftarrow 2$
else
  $j_A \leftarrow 0$

Subroutine SWITCH$(\Delta, j, j_{\text{new}})$
$\Delta \leftarrow \Delta \oplus L_*(\mathsf{str2num}(\langle j \rangle_4 \oplus \langle j_{\text{new}} \rangle_4))$
$j \leftarrow j_{\text{new}}$

Subroutine PROC1(M, A', H, Δ, i)
$r_{\text{stop}} \leftarrow \min\{\ell, a'\}$
for $r \leftarrow i$ **to** r_{stop} **do**
  $C_{r-1} \leftarrow H \oplus M_{r-1}$
  $H \leftarrow H \oplus A'_r$
  $\Delta \leftarrow \Delta \oplus L(\mathtt{ntz}(r))$
  $H \leftarrow F_K(H \oplus \Delta, M_{r-1})$
$i \leftarrow r_{\text{stop}} + 1$

Subroutine PROC2(M, H, Δ, i)
for $r \leftarrow i$ **to** ℓ **do**
  $C_{r-1} \leftarrow H \oplus M_{r-1}$
  $\Delta \leftarrow \Delta \oplus L(\mathtt{ntz}(r))$
  $H \leftarrow F_K(H \oplus \Delta, M_{r-1})$
$i \leftarrow \ell + 1$

Subroutine PROC3(A^*, H, Δ, i)
for $r \leftarrow 1$ **to** a^* **do**
  $\mathsf{Left} \leftarrow A^*_r[n + m - 1 \cdots m]$
  $\mathsf{Right} \leftarrow A^*_r[m - 1 \cdots 0]$
  $H \leftarrow H \oplus \mathsf{Left}$
  $\Delta \leftarrow \Delta \oplus L(\mathtt{ntz}(i + r - 1))$
  $H \leftarrow F_K(H \oplus \Delta, \mathsf{Right})$
$i \leftarrow i + a^* - 1$

Fig. 3. The subroutines used in the encryption algorithm of p-OMD$[F, \tau]$ (Figure 2)

(3) **Selection of the j component in the index of the masks $\Delta_{N,i,j}$.** We use different values of j to separate the calls to the masked F in different contexts. Let's classify the calls to the masked F to two types: (1) the final call to F which returns the tag, and (2) the internal calls. We note that in the special case that $M = \varepsilon$ and $|A| \leq n$ there will be only one call to F which returns the tag; hence, it is considered as the final call.

Internal Calls. We use $j \in \{0, 1, 2\}$ for the internal calls made to the masked F as follows.

For $i = 1$, i.e. the first call to F, the value of j is determined as follows:
- if $\ell > 0$ and $a' > 0$ then let $j = 0$,
- if $\ell > 0$ and $a' = 0$ then let $j = 1$,
- if $\ell = 0$ and $a^* > 0$ then let $j = 2$.

For $1 < i < \ell + 1 + a^*$, depending on the presence of message blocks and AD blocks to be processed at the i^{th} call to the masked F, we have:

* if both an n-bit AD block and an m-bit message block are present then $j = 0$,
* if only an m-bit message block is present (no AD block is processed) then $j = 1$,
* if only an $(n+m)$-bit AD block is present (no message block is processed) then $j = 2$.

Final Call. The final call to F which produces the authentication tag uses $j_f \in \{3, 4, 5, \ldots, 14, 15\}$. If the tag is produced by a call to F with $i \neq 1$, we have three main cases depending on the inputs to the final masked F.

* If both an AD block and a message block are present in the final call (see **Case A** in Figure 1) then $j_f \in \{4, 5, 6, 7\}$; where, we let $j_f = 4$ if $|M_\ell| = m$ and $|A'_{a'}| = n$; let $j_f = 5$ if $|M_\ell| < m$ and $|A'_{a'}| = n$; let $j_f = 6$ if $|M_\ell| = m$ and $|A'_{a'}| < n$, and otherwise ($|M_\ell| < m$ and $|A'_{a'}| < n$) let $j_f = 7$.
* If only a message block is present but no AD block is processed in the final call (see **Case B** in Figure 1) then $j_f \in \{8, 9, 10, 11\}$; where, we let $j_f = 8$ if $|M_\ell| = m$ and $|A'_{a'}| = n$; let $j_f = 9$ if $|M_\ell| < m$ and $|A'_{a'}| = n$; let $j_f = 10$ if $|M_\ell| = m$ and $|A'_{a'}| < n$, and otherwise ($|M_\ell| < m$ and $|A'_{a'}| < n$) let $j_f = 11$. For the special case where there is no associate data at all, i.e. $A = \varepsilon$, we let $j_f = 8$ if $|M_\ell| = m$ and let $j_f = 9$ if $|M_\ell| < m$.
* If only an AD block is present but no message block is processed in the final call (see **Case C** and **Case D** in Figure 1) then $j_f \in \{12, 13, 14, 15\}$; where, we let $j_f = 12$ if $|M_\ell| = m$ and $|A^*_{a^*}| = n + m$; let $j_f = 13$ if $|M_\ell| < m$ and $|A^*_{a^*}| = n + m$; let $j_f = 14$ if $|M_\ell| = m$ and $|A^*_{a^*}| < n + m$, and otherwise ($|M_\ell| < m$ and $|A^*_{a^*}| < n + m$) let $j_f = 15$. For the special case where there is no message at all, i.e. $M = \varepsilon$, let $j_f = 12$ if $|A^*_{a^*}| = n + m$ and let $j_f = 14$ if $|A^*_{a^*}| < n + m$.

For $i = 1$ (meaning that the final call is the same as the first call, which happens if $M = \varepsilon$ AND $|A| \leq n$) we need to apply a special treatment:

* if both $M = A = \varepsilon$ then $j_f = 3$ (**Case F** in Figure 1),
* if $M = \varepsilon$ and $0 < |A| \leq n$ then we let $j_f = 12$ if $|A| = n$, otherwise, let $j_f = 14$ (**Case E** box in Figure 1).

Note that there is no variable j_f in Figure 2 as j_f corresponds to a special use of variable j in the last call to F. Specifically, j_f corresponds to the calls to the SWITCH subroutine that use the value of new j of the form $\mathsf{const} + j_A + j_M$ or the value 3.

(4) **Decryption Algorithm**: The decryption algorithm accepts a ciphertext $\mathbb{C} \in \{0,1\}^*$ together with associated data $A \in \{0,1\}^*$ and nonce $N \in \{0,1\}^{|N|}$, and using key $K \in \{0,1\}^k$ obtains a plaintext $M \in \{0,1\}^*$ or returns an invalid indication $\perp$. If $|\mathbb{C}| < \tau$ then return $\perp$. Otherwise let C be the first $|\mathbb{C}| - \tau$ bits of $\mathbb{C}$ and Tag be the remaining τ bits. Now, considering that the encryption process of p-OMD is actually an additive stream cipher with an integrated authentication mechanism, the decryption process proceeds the same as the encryption process up until the verification of the tag, which happens at the end of the decryption process where the newly computed tag Tag' is compared with the provided tag Tag. If $\mathsf{Tag}' = \mathsf{Tag}$ then output M, otherwise output $\perp$.

5 Security Analysis

The security analysis for p-OMD is modular and easy to follow. The high-level structure of the analysis is similar to that of OMD, as expected from the similarities of the algorithms, though the details differ and are more involved. The proof is divided into three main steps as follows:

Step 1: *Idealization of the p-OMD scheme using a tweakable random function.* We first analyse the security of a generalized variant of p-OMD$[F, \tau]$ where the "masked F" (aimed to instantiate a tweakable function) is replaced by an ideal primitive; namely, a tweakable random function $\widetilde{R}$. This generalized scheme is called p-$\mathbb{OMD}[\widetilde{R}, \tau]$ and illustrated in Figure 4. This is the major proof step which differs from and is more involved than that of OMD.

Step 2: *Realization of the tweakable random function by a tweakble PRF.* This is a well-known classical method where the (ideal) random function is replaced by a PRF. This proof step is therefore the same as that of OMD.

Step 3: *Instantiation of the tweakable PRF via a PRF.* To make a tweakable PRF out of a PRF, we use the XE method of [24] with the masking sequence generated based on an appropriate adjustment of a canonical Gray code sequence [20, 26]. This step is similar to that of OMD; only the details of the mask generation function differ.

The security bound for p-OMD is given by Theorem 1. It is interesting to note that the security bound is the same as that of OMD, showing that the natural modifications we made to OMD to obtain p-OMD are without any loss of security.

Theorem 1. *Fix $n \geq 1$, $0 \leq \tau \leq n$. Let $F : \mathcal{K} \times (\{0,1\}^n \times \{0,1\}^m) \rightarrow \{0,1\}^n$ be a PRF, where the key space $\mathcal{K} = \{0,1\}^k$ for $k \geq 1$ and $1 \leq m \leq n$. We have*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{p-OMD}[F,\tau]}(t, q_e, \sigma_e, \ell_{max}) \leq \mathbf{Adv}^{\mathrm{prf}}_{F}(t', 2\sigma_e) + \frac{3\sigma_e^2}{2^n}$$
$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{p-OMD}[F,\tau]}(t, q_e, q_v, \sigma, \ell_{max}) \leq \mathbf{Adv}^{\mathrm{prf}}_{F}(t', 2\sigma) + \frac{3\sigma^2}{2^n} + \frac{q_v \ell_{max}}{2^n} + \frac{q_v}{2^\tau}$$

where q_e and q_v are, respectively, the number of encryption and decryption queries, ℓ_{max} denotes the maximum number of the internal calls to F in an encryption or decryption query, $t' = t + cn\sigma$ for some constant c, and σ_e and σ are the total number of calls to the underlying compression function F in all queries asked by the CPA and CCA adversaries against the privacy and authenticity of the scheme, respectively.

The proof is obtained by combining Lemma 1 in Section 5.1 with Lemma 2 in Section 5.2 and Lemma 3 in Section 5.3.

5.1 Idealization of p-OMD

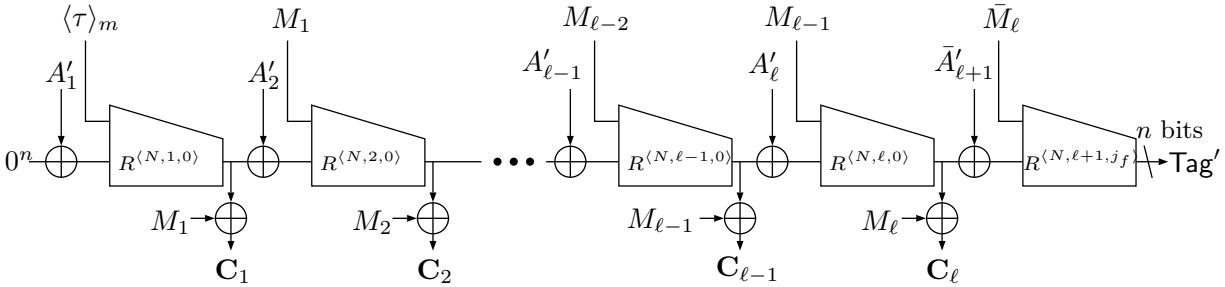
The scheme p-$\mathbb{OMD}[\widetilde{R}, \tau]$ is a generalization (idealization) of p-OMD$[F, \tau]$ that uses a tweakable random function $\widetilde{R} : \mathcal{T} \times (\{0,1\}^n \times \{0,1\}^m) \rightarrow \{0,1\}^n$ instead of the masked F (used for instantiating such an ideal primitive). The p-$\mathbb{OMD}[\widetilde{R}, \tau]$ is depicted in Figure 4. The tweak space $\mathcal{T}$ consists of sixteen mutually exclusive sets of tweaks $\mathcal{T} = \bigcup_{i=0}^{15} \mathcal{N} \times \mathbb{N} \times \{i\}$, where $\mathcal{N} = \{0,1\}^{|N|}$ is the set of nonces and $\mathbb{N}$ is the set of positive integers.

Lemma 1. *Let p-$\mathbb{OMD}[\widetilde{R}, \tau]$ be the scheme shown in Figure 4. Then*

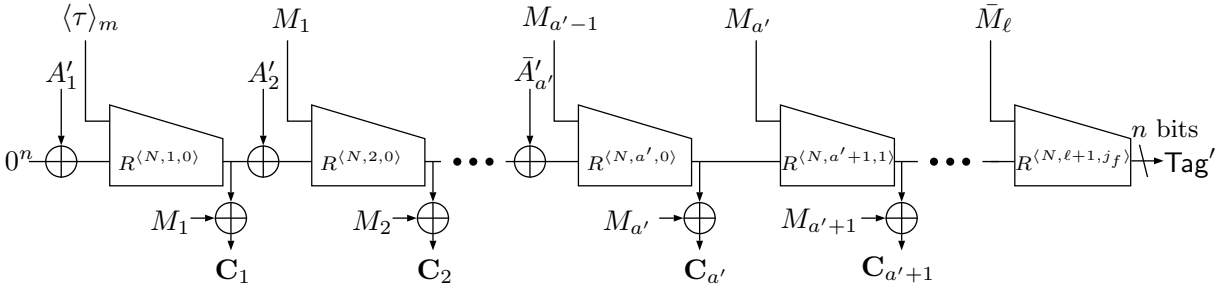
$$\mathbf{Adv}^{\mathrm{priv}}_{p\text{-}\mathbb{OMD}[\widetilde{R},\tau]}(q_e, \sigma_e, \ell_{max}) = 0$$
$$\mathbf{Adv}^{\mathrm{auth}}_{p\text{-}\mathbb{OMD}[\widetilde{R},\tau]}(q_e, q_v, \sigma, \ell_{max}) \leq \frac{q_v \ell_{max}}{2^n} + \frac{q_v}{2^\tau}$$

where q_e and q_v are, respectively, the number of encryption and decryption queries, ℓ_{max} denotes the maximum number of the internal calls to the underlying tweakable random function $\widetilde{R}$ in an encryption or decryption query, and σ_e and σ are the total number of calls to $\widetilde{R}$ in all queries asked by the CPA and CCA adversaries against the privacy and authenticity of the scheme, respectively.

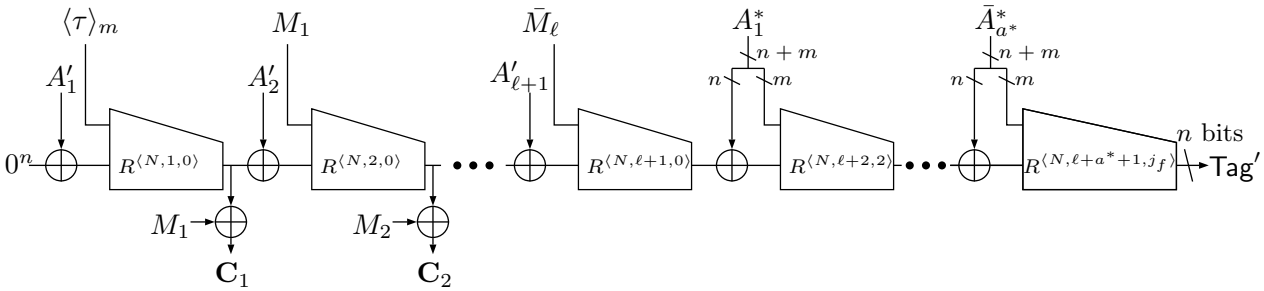
Proof. The proof of the privacy bound is straightforward. Let $\boldsymbol{A}$ be a CPA adversary that asks (encryption) queries $(N^1, A^1, M^1) \cdots (N^{q_e}, A^{q_e}, M^{q_e})$ where all N^r values (for $1 \leq r \leq q_e$) are distinct due to the nonce-respecting assumption on the adversary $\boldsymbol{A}$. Referring to Figure 4, this means that we are



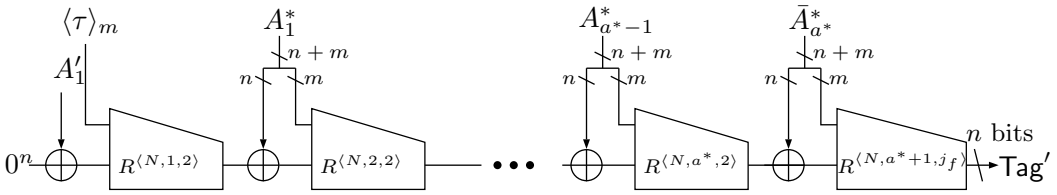
Case A: $\ell > 0$ and $|\mathbf{A}|_{\mathbf{n}} = \ell + 1$. Let $\bar{M}_\ell = M_\ell||10^{m-|M_\ell|-1}$ if $|M_\ell| < m$ and $\bar{M}_\ell = M_\ell$ otherwise. Let $\bar{A}'_{a'} = A'_{a'}||10^{n-|A'_{a'}|-1}$ if $|A'_{a'}| < n$ and $\bar{A}'_{a'} = A'_{a'}$ otherwise.



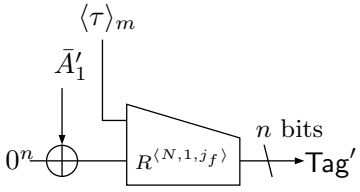
Case B: $\ell > 0$ and $|\mathbf{A}|_{\mathbf{n}} < \ell + 1$. Let $\bar{M}_\ell = M_\ell||10^{m-|M_\ell|-1}$ if $|M_\ell| < m$ and $\bar{M}_\ell = M_\ell$ otherwise. Let $\bar{A}'_{a'} = A'_{a'}||10^{n-|A'_{a'}|-1}$ if $|A'_{a'}| < n$ and $\bar{A}'_{a'} = A'_{a'}$ otherwise.



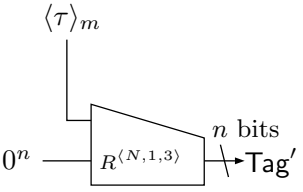
Case C: $\ell > 0$ and $|\mathbf{A}|_{\mathbf{n}} > \ell + 1$. Let $\bar{M}_\ell = M_\ell||10^{m-|M_\ell|-1}$ if $|M_\ell| < m$ and $\bar{M}_\ell = M_\ell$ otherwise. Let $\bar{A}^*_{a^*} = A^*_{a^*}||10^{n+m-|A^*_{a^*}|-1}$ if $|A^*_{a^*}| < n + m$ and $\bar{A}^*_{a^*} = A^*_{a^*}$ otherwise.



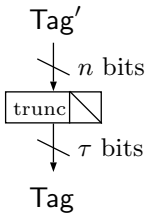
Case D: $\mathbf{M} = \varepsilon$ and $|\mathbf{A}| > \mathbf{n}$. Let $1\bar{A}^*_{a^*} = A^*_{a^*}||10^{n+m-|A^*_{a^*}|-1}$ if $|A^*_{a^*}| < n + m$ and $\bar{A}^*_{a^*} = A^*_{a^*}$ otherwise.



Case E: $\mathbf{M} = \varepsilon$ and $\mathbf{0} < |\mathbf{A}| \leq \mathbf{n}$. Let $\bar{A}'_1 = A'_1||10^{n-|A'_1|-1}$ if $|A'_1| < n$ and $\bar{A}'_1 = A'_1$ otherwise.



Case F
$\mathbf{M} = \mathbf{A} = \varepsilon$



Obtaining the final tag.

Fig. 4. The p-$\mathbb{OMD}[\widetilde{R}, \tau]$ scheme using a tweakable random function $\widetilde{R} \xleftarrow{\$} \mathrm{Func}^{\mathcal{T}}(n+m, n)$.

applying independent random functions $\widetilde{R}^{N_x,i,j}$ each to a single input value, hence the images that the adversary sees (i.e. $\mathbb{C}^r$ for $1 \leq r \leq q_e$) are fresh uniformly random values.

The proof of the authenticity bound is a rather involved case analysis. A visualisation of the hierarchy of the cases as a tree is presented in Figure 5 to improve clarity of the proof. We first analyse the case where the adversary makes a single verification query and then we will use the generic result of Bellare et al. [6] (i.e. use hybrids to get rid of decryption queries one by one) to get a bound against adversaries that make multiple verification queries.

Let $\boldsymbol{A}$ be a single-verification-query adversary that is making q_e encryption queries $(N^1, A^1, M^1) \cdots (N^{q_e}, A^{q_e}, M^{q_e})$. Let $M^i = M^i_1 \cdots M^i_{\ell_i}$ be the message and $A^i = A'^i_1 \cdots A'^i_{a'_i} || A^{*i}_1 \cdots A^{*\,i}_{a^*_i}$ be the associated data in the i^{th} encryption query. Let $\mathbb{C}^i = C^i || \mathsf{Tag}^i$ be the ciphertext received for query (N^i, A^i, M^i). That is, we use superscripts to indicate query numbers and subscripts to denote the block indices in each query. We further let $x^i = 1 + \ell + a^*{}_i$. Note that x_i is the number of calls to the compression function made while processing the i^{th} query and also the value of the second tweak component used in the final call to the compression function which produces Tag^i.

Let $(N, A, \mathbb{C})$ be the forgery attempt by the adversary, where $N \in \{0,1\}^{|N|}$ is the nonce, $A = A'_1 \cdots A'_{a'} || A^*_1 \cdots A^*_{a^*}$ is the associated data, $\mathbb{C} = C || \mathsf{Tag}$ is the ciphertext where $C = C_1 \cdots C_\ell$ and $\mathsf{Tag} \in \{0,1\}^\tau$ is the tag. Let $M = M_1 \cdots M_\ell$ denote the corresponding decrypted message. We let $x = 1 + \ell + a^*$ be the number of calls to F made while processing the forgery attempt (which is the same as the value of the second tweak component in the final call to the compression function that is supposed to produce the Tag). Note that no superscripts are used for the strings in the alleged forgery by the adversary.

In the proof, we refer to the intermediate chaining variables that occur in the query processing, namely let H^i_r denote the *output* of the r^{th} call to the compression function in the processing of the i^{th} encryption query, so we have $H^i_1 = \widetilde{R}^{\langle N,1,0\rangle}(A'^i_1, \langle\tau\rangle_m)$ and $\mathsf{Tag}^i = H^i_{x_i}[n-1 \cdots n-\tau]$. Similarly, we let H_r stand for r^{th} intermediate chaining value in the processing of the forgery attempt.

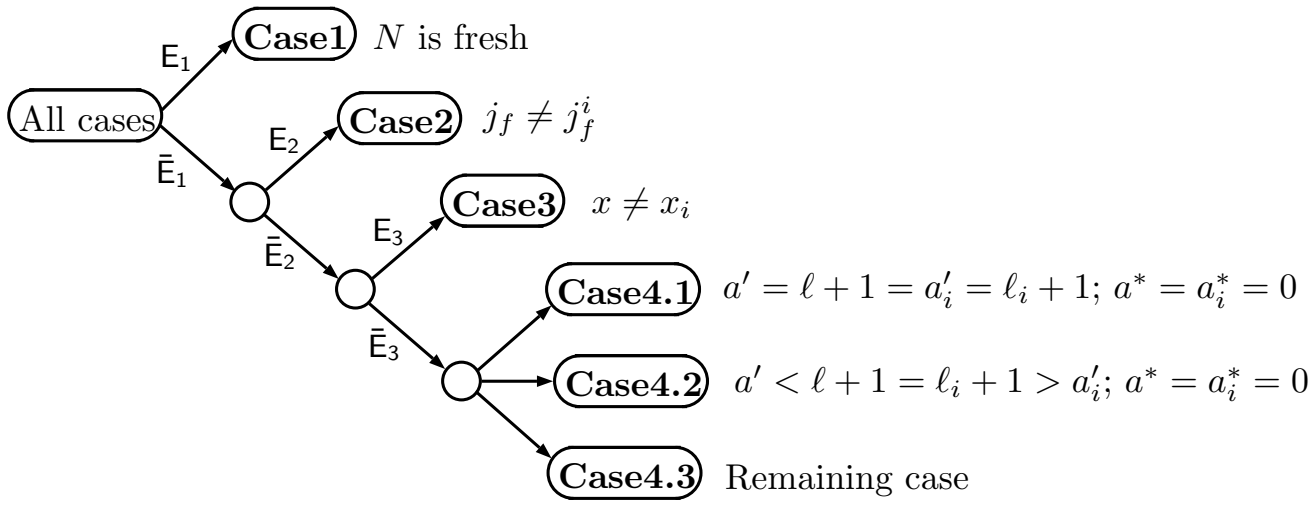


Fig. 5. The structure of the proof of the authenticity bound. A condition on an edge applies to the whole subtree.

We have the following disjoint cases.

Case 1: $N \notin \{N^1, \cdots N^{q_e}\}$. We let E_1 denote the event $N \notin \{N^1, \cdots N^{q_e}\}$ in the following. The adversary has to find a correct Tag that is the first τ bits of a value produced by $\widetilde{R}^{\langle N,x,j_f\rangle}(.)$.

Because the nonce-component N of the tweak $\langle N, x, j_f \rangle$ has not been used in any encryption query, $\boldsymbol{A}$ has not seen any image under $\widetilde{R}^{\langle N,x,j_f \rangle}(.)$ (for any values of x and j_f). Thus the probability that the adversary can succeed in finding correct value of Tag is $2^{-\tau}$.

In all the following cases we have $\bar{\mathsf{E}}_1$, i.e. $N = N^i$ for a *single* $i \in \{1, \ldots, q_e\}$ (noticing that no nonce is reused during encryption queries). We can ignore all queries other than the i^{th} query since the responses to such queries are random and independent (because of using different nonces) to the adversary's task to make the forgery attempt $N, A, \mathbb{C}$ with $N = N^i$.

Case 2: $\bar{\mathsf{E}}_1 \wedge \mathsf{E}_2$, where E_2 is the event that $j_f \neq j_f^i$. Recall that a successfully forged Tag must be the first τ bits of a value produced by $\widetilde{R}^{\langle N,x,j_f \rangle}(.)$. The inequality $j_f \neq j_f^i$ occurring in this case implies that $\widetilde{R}^{\langle N,x,j_f \rangle}(.)$ is a fresh RF and the adversary has not seen any image under it (no matter what are the values of x and x_i) and the adversary has to guess the correct Tag. The probability of a successful forgery is therefore $2^{-\tau}$.

We need to introduce some auxiliary notation for the analysis of the following cases. Consider the i^{th} encryption query. Depending on the length of the message $|M^i|$ and the length of AD $|A^i|$, we can have three situations. In the *first situation*, we have $|M^i|_m + 1 = |A^i|_n$ and $|M^i| > 0$ (**Case A** in Figure 4). This means that the compression function call used to produce Tag^i has a block of message at its m-bit input and an AD block xored to the chaining variable at its n-bit input. We denote this event as $\texttt{type-1}^i$ and we note that $j_f^i \in \{4, 5, 6, 7\}$. The *second possible situation* arises if $|M^i|_m + 1 > |A^i|_n$ with $|M^i| > 0$ (**Case B** in Figure 4). There is no block of the AD xored to the n-bit input of the final compression function call. We denote this event as $\texttt{type-2}^i$ and we note that $j_f^i \in \{8, 9, 10, 11\}$. The *last possible situation* is when either $|M^i|_m + 1 < |A^i|_n$ and $|A^i| > n$ so there is a block of AD xored to the n-bit input as well as another block fed directly to the m-bit input in the final call to the compression function (**Cases C, D** in Figure 4) or $0 < |A^i| \leq n$ and $M^i = \varepsilon$ (**Case E** in Figure 4). We denote this by $\texttt{type-3}^i$ and we note that $j_f^i \in \{12, 13, 14, 15\}$.

We define $\texttt{type-1}$, $\texttt{type-2}$ and $\texttt{type-3}$ for the forgery attempt in a similar way (note that $|C| = |M|$).

In the following, we need to address the event $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2$ i.e. $N = N^i$, $j_f = j_f^i$. We remark that the condition $\bar{\mathsf{E}}_2$ is met for a valid forgery (i.e. $(A, M) \neq (A^i, M^i)$) if and only if both the i^{th} encryption query and the alleged forgery are

- non-empty, i.e. $(A, M) \neq (\varepsilon, \varepsilon) \wedge (A^i, M^i) \neq (\varepsilon, \varepsilon)$,
- padded in the same way, i.e. $(m \mid |C| \Leftrightarrow m \mid |C^i|) \vee (n \mid |A'| \Leftrightarrow n \mid |A'^i|) \vee (n{+}m \mid |A^*| \Leftrightarrow n{+}m \mid |A^{*i}|)$ (we pad the last block of M iff we pad the last block of M^i and the same applies for associated data),
- of the same "type", i.e. $(\texttt{type-1} \wedge \texttt{type-1}^i) \vee (\texttt{type-2} \wedge \texttt{type-2}^i) \vee (\texttt{type-3} \wedge \texttt{type-3}^i)$.

Case 3: $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2 \wedge \mathsf{E}_3$ where E_3 stands for the event that $x \neq x_i$. Recall that Tag^i is produced as the τ most significant bits of an image under $\widetilde{R}^{\langle N,x_i,j_f^i \rangle}(.)$ for some j_f^i and Tag is produced similarly, as the τ most significant bits of an image under $\widetilde{R}^{\langle N,x,j_f \rangle}(.)$ for some j_f. We have two sub-cases.

Case 3a: If $x > x_i$ then $\boldsymbol{A}$ has seen no image under $\widetilde{R}^{\langle N,x,j_f \rangle}(.)$ regardless of the value of j_f and the probability of a successful forgery (equivalent to guessing τ random bits) is $2^{-\tau}$.

Case3b: If $x < x_i$ then a single image under $\widetilde{R}^{\langle N,x,j^i \rangle}$ was used in processing of the i^{th} encryption query. However Tag is produced by a fresh RF $\widetilde{R}^{\langle N,x,j_f \rangle}(.)$, because we always have $j^i \neq j_f$ (because of the rules of selecting the j tweak component). The probability of a successful forgery (equivalent to guessing τ random bits) is $2^{-\tau}$.

Case 4: It remains to address the cases, where we have $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2 \wedge \bar{\mathsf{E}}_3$, i.e. the case, when the i^{th} encryption query and the alleged forgery (1) share the same nonce, (2) are padded in the same way, are both non-empty, are of the same "type" so $j_f = j_f^i$ and (3) they are both processed with the same number of calls to the compression function. We investigate each of the three possible "types" separately.

Case 4.1: $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2 \wedge \bar{\mathsf{E}}_3 \wedge (\texttt{type-1} \wedge \texttt{type-1}^i)$. This means that $a' = \ell + 1 = x = x_i = a'_i = \ell_i + 1$, $a^* = 0$ and $a^*{}_i = 0$. Both Tag^i and Tag are produced by the same RF $\widetilde{R}^{\langle N,x,4\rangle}(.)$. W.l.o.g. assume that both $|A'|$ and $|A'^i|$ are a multiple of n and both $|M|$ and $|M^i|$ are a multiple of m. We can make this assumption because in the other cases the incomplete blocks are *injectively* padded to full length (and because of the condition $\bar{\mathsf{E}}_2$). This means that if two strings are unequal before being padded will also be unequal after the padding, e.g. $A'_{a'} \neq A'^i_{a'_i} \Rightarrow \bar{A}'_{a'} \neq \bar{A}'^i_{a'_i}$ (refer to Figure 4). After that point the security analysis is almost identical with what follows.

The adversary can succeed in producing a valid forgery in two ways. Either the inputs into the last RF in the processing of the i^{th} encryption query and the inputs into the last RF in the processing of the forgery attempt are distinct, i.e. $(H^i_{x-1} \oplus A'^i_{a'}, M^i_\ell) \neq (H_{x-1} \oplus A'_{a'}, M_\ell)$, or they are equal, i.e. $(H^i_{x-1} \oplus A'^i_{a'}, M^i_\ell) = (H_{x-1} \oplus A'_{a'}, M_\ell)$. In the former case, the adversary is left with the task of guessing the output value of a RF on an input, that has not been evaluated before which is bounded with the probability $p_{fn} = 2^{-\tau}$.

In the latter case, the equality $(H^i_{x-1} \oplus A'^i_{a'}, M^i_\ell) = (H_{x-1} \oplus A'_{a'}, M_\ell)$ permits the adversary to set $\mathsf{Tag} = \mathsf{Tag}^i$. We must have $(N, A, M) \neq (N^i, A^i, M^i)$, so there is a position r in which the two queries differ, i.e. we must have an $1 \leq r < x$, such that $(A'_r, M_r) \neq (A'^i_r, M^i_r)$ and after which the queries are identical. So $\boldsymbol{A}$ has not seen the image RF $H_r = \widetilde{R}^{\langle N,r,j_r\rangle}(H_{r-1} \oplus A'_r, M_r)$ but he must ensure that $H_r = H^i_r$. This happens with a probability of 2^{-n}. We bound the total probability of achieving the final collision by $p_{fe} = (x-1)2^{-n}$ obtained by summing the probability of collision $H_r = H^i_r$ for every possible value of r.

The bound of Case 4.1 is finally obtained as the sum $p_{fn} + p_{fe} = (x-1)2^{-n} + 2^{-\tau}$

Case 4.2: $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2 \wedge \bar{\mathsf{E}}_3 \wedge (\texttt{type-2} \wedge \texttt{type-2}^i)$. This implies $\ell + 1 > |A|_n$, $\ell_i + 1 > |A^i|_n$ and $a^* = a^*{}_i = 0$, so $x = x' = \ell + 1 = \ell_i + 1$. W.l.o.g. assume that both $|A'|$ and $|A'^i|$ are a multiple of n and both $|M|$ and $|M^i|$ are a multiple of m by similar argument as in Case 4.1. We have two subcases:

Case 4.2a: $|A|_n = |A^i|_n$, i.e. $a' = a'_i$. Analysis of this case is very similar to Case 4. Again we observe, that the adversary's chance to produce a forgery is bounded by $2^{-\tau}$ if the inputs to the final RF are distinct. The adversary can reuse Tag^i if he manages to force the collision on the inputs to the final RF. The probability that $\boldsymbol{A}$ can succeed in forcing this collision is bounded in the same way as in Case 4.1 by summing 2^{-n} for $1 \leq r < x$. For $r > a'$ we have no more blocks of A' to consider, which in fact gives the adversary even less power. We conclude that the probability of forgery in this case is bounded by $2^{-\tau} + (x-1)2^{-n}$.

Case 4.2b: $|A|_n \neq |A^i|_n$, i.e. $a' \neq a'_i$. The analysis of this case is very similar to the previous one. The difference lies in the fact, that we need to consider, that if $a' > a'_i$ then there is at least one r, such that $1 \leq r < x$ and there is a block A'_r but there is no block A'^i_r (or the other way around if $a' < a'_i$).This implies that two independent RFs $\widetilde{R}^{\langle N,r,0\rangle}(.)$ and $\widetilde{R}^{\langle N,r,1\rangle}(.)$ are applied in the r^{th} call (this is ensured by the rules of selecting the value of the last tweak component j in the internal calls). Keeping this in mind, the analysis of this case follows the same structure as the previous case and we conclude that the probability of forgery is bounded by $2^{-\tau} + (x-1)2^{-n}$

Case 4.3: $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2 \wedge \bar{\mathsf{E}}_3 \wedge (\texttt{type-3} \wedge \texttt{type-3}^i)$ so $\ell + 1 < |A|_n$ and $\ell_i + 1 < |A^i|_n$, so $x = x' = \ell + 1 + a^* = \ell_i + 1 + a^*{}_i$ with both a^* and $a^*{}_i$ non-zero. W.l.o.g. assume that both $|A^*|$ and $|A^{*i}|$ are a multiple of $n + m$ and both $|M|$ and $|M^i|$ are a multiple of m by similar argument as in Case 4.1. We have three subcases.

Case 4.3a: $0 < |A| \leq n$, $0 < |A^i| \leq n$ and $M = M^i = \varepsilon$. W.l.o.g. assume $|A| = |A^i| = n$ (by the argument that uses the injective property of used padding). Since the alleged forgery must be different from all encryption queries, we have $A \neq A^i$ and the adversary must guess the output

of a RF on a new input. The probability of forgery is thus $2^{-\tau}$. In following two subcases we have $|A| > n$ and $|A^i| > n$.

Case 4.3b: $|M|_m = |M^i|_m$, i.e. $\ell = \ell_i$ and $a^* = a^*{}_i$. The analysis is almost identical as in case 4.2a, with the difference that for $r > \ell+1$ we have no more blocks of M to consider but we have $n+m$ blocks of AD instead. The probability of inner collisions $H_r = H_r^i$ for $r > \ell + 1$ is thus also 2^{-n} and we conclude that the probability of forgery is bounded by $2^{-\tau} + (x-1)2^{-n}$.

Case 4.3c: $|M|_m \neq |M^i|_m$, i.e. $\ell \neq \ell_i$ and $a^* \neq a^*{}_i$. Similarly as in Case 4.2b we need to take into account that the adversary can change the length of message and AD, so that there must be r such that $1 \leq r < x$ and such that there is M_r but no M_r^i (or the other way around). The separation of the tweaks again ensures that the probability of internal collision is 2^{-n} for such r. We conclude that the probability of forgery is bounded by $2^{-\tau} + (x-1)2^{-n}$

Finally, using the results of Bellare et al. [6] we get the bound against adversaries that make q_v decryption (verification) queries with length limited by ℓ_{max} as $\frac{q_v}{2^\tau} + \frac{q_v \ell_{max}}{2^n}$.

5.2 Realization of Tweakable RFs with Tweakable PRFs

This is a classical step in which the ideal primitive—tweakable random function $\widetilde{R}$—is replaced with a standard primitive—tweakable PRF $\widetilde{F}$. The security loss induced by this step is stated in the following lemma.

Lemma 2. *Let $\widetilde{R} : \mathcal{T} \times (\{0,1\}^n \times \{0,1\}^m) \to \{0,1\}^n$ be a tweakable RF and $\widetilde{F} : \mathcal{K} \times \mathcal{T} \times (\{0,1\}^n \times \{0,1\}^m) \to \{0,1\}^n$ be a tweakable PRF. Then*

$$\mathbf{Adv}^{\text{priv}}_{p\text{-}\mathbb{OMD}[\widetilde{F},\tau]}(t, q_e, \sigma_e, \ell_{max}) \leq \mathbf{Adv}^{\text{priv}}_{p\text{-}\mathbb{OMD}[\widetilde{R},\tau]}(q_e, \sigma_e, \ell_{max}) + \mathbf{Adv}^{\widetilde{\text{prf}}}_{\widetilde{F}}(t', \sigma_e)$$

$$\mathbf{Adv}^{\text{auth}}_{p\text{-}\mathbb{OMD}[\widetilde{F},\tau]}(t, q_e, q_v, \sigma, \ell_{max}) \leq \mathbf{Adv}^{\text{auth}}_{p\text{-}\mathbb{OMD}[\widetilde{R},\tau]}(q_e, q_v, \sigma, \ell_{max}) + \mathbf{Adv}^{\widetilde{\text{prf}}}_{\widetilde{F}}(t'', \sigma)$$

where q_e and q_v are, respectively, the number of encryption and decryption queries, $q = q_e + q_v$, ℓ_{max} denotes the maximum number of the internal calls to F in an encryption or decryption query, $t' = t + cn\sigma_e$ and $t'' = t + c'n\sigma$ for some constants c, c', and σ_e and σ are the total number of calls to the underlying compression function F in all queries asked by the CPA and CCA adversaries against the privacy and authenticity of the scheme, respectively.

5.3 Instantiation of Tweakable PRFs with PRFs

The last step is to instantiate the tweakable PRFs by means of a (keyed) compression function which is assumed to be PRF. Similar to OMD, we use the XE method of [24] as shown in Fig. 6.

The proof and bound for this step follows from that of OMD, which in turn is a straightforward adaptation of the proof of the XE construction in [20]. Lemma 3 states the bound for this transformation. Here, the only aspect which is different between OMD and p-OMD is the way that the masking sequence $\Delta_{N,i,j}$ is computed. We need to show that the specific mask function $\Delta_K(N,i,j)$ for p-OMD satisfies the same set of security criteria as required from that of OMD. Note that here, we use the notation $\Delta_K(N,i,j)$ to refer to the function that computes the masking sequence $\Delta_{N,i,j}$.

In p-OMD the tweaks are of the form $T = (N,i,j)$ where $N \in \mathcal{N} \cup \{\varepsilon\}$, $1 \leq i \leq 2^{n-6}$ and $j \in \{0,1,\cdots,15\}$. We have to show that the mask function $\Delta_K(T) = \Delta_K(N,i,j)$ (outputting an n-bit mask) satisfies the following two properties for any fixed string $H \in \{0,1\}^n$:

1. $\Pr[\Delta_K(N,i,j) = H] \leq 2^{-n}$ for any (N,i,j)
2. $\Pr[\Delta_K(N,i,j) \oplus \Delta_K(N',i',j') = H] \leq 2^{-n}$ for $(N,i,j) \neq (N',i',j')$

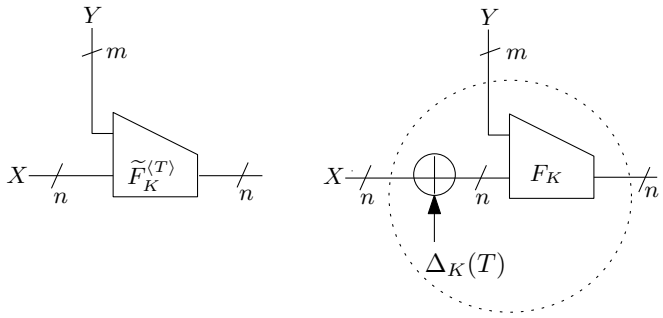


Fig. 6. Building a tweakable PRF $\widetilde{F}_K^{\langle T\rangle} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ using a PRF $F_K : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$.

where the probabilities are taken over random selection of the key.

As shown in Appendix A, it can be easily verified that these two properties are satisfied by the specific mask generation scheme of p-OMD, as described in Section 4.

Lemma 3. *Let $F : \mathcal{K} \times (\{0,1\}^n \times \{0,1\}^m) \to \{0,1\}^n$ be a function family with key space $\mathcal{K}$. Let $\widetilde{F} : \mathcal{K} \times \mathcal{T} \times (\{0,1\}^n \times \{0,1\}^m) \to \{0,1\}^n$ be defined by $\widetilde{F}_K^{\langle T\rangle}(X,Y) = F_K((X \oplus \Delta_K(T)), Y)$ for every $T \in \mathcal{T}, K \in \mathcal{K}$, $X \in \{0,1\}^n, Y \in \{0,1\}^m$ and $\Delta_K(T)$ is the masking function of p-OMD as defined in Section 4. If F is PRF then $\widetilde{F}$ is tweakable PRF; more precisely*

$$\mathbf{Adv}^{\widetilde{\text{prf}}}_{\widetilde{F}}(t,q) \leq \mathbf{Adv}^{\text{prf}}_{F}(t',2q) + \frac{3q^2}{2^n}$$

.

6 Performance Comparison with OMD

To verify the performance advantage of p-OMD over OMD, with respect to processing associated data, we implemented the two algorithms in software and made some measurements to determine and compare their performance.

The comparison is performed on the x86-64 architecture (Intel Core i7-3632QM, with all measurements carried out on a single core). For OMD, we used the OMD-sha512 instantiation optimised for the AVX1 instruction extension, which achieves the best result according to the CAESAR benchmarking measurements [1]. We made the necessary modifications (as in description of p-OMD) to the same code to obtain our implementation of p-OMD. Both OMD and p-OMD were instantiated with the same parameters: key length=512, nonce length=256, tag length=256. Both implementations have been built using the gcc compiler and setting the `-Ofast` optimization flag.

We measure the time complexity of the encryption process for varying lengths of message and associated data. For the sake of this section, let m denote the message length and a the AD length in bytes. We measure the encryption time for $m \in \{64, 128, 192, \ldots, 4096\}$ and $a \in \{64, 128, \ldots m\}$ for every value of m. That is, we consider the typical case when AD is at most as long as the message.

For both OMD and p-OMD and for every pair of values m, a, we measure the time of one encryption using the `rdtsc` instruction 200 times to compute the mean time. This is repeated 91 times and the value we take as the result is the median of these 91 mean encryption times. We additionally apply the same procedure to measure time complexity of the encryption of OMD with $m \in \{64, 128, \ldots, 4096\}$ and $a = 0$. The results are shown in Figure 7.

The top left graph in Figure 7 shows that the relative complexity of encryption of both OMD and p-OMD decreases as the length of AD increases; however, p-OMD performs better than OMD. The top right graph demonstrates that *if the length of AD is close to the message length* then p-OMD has a

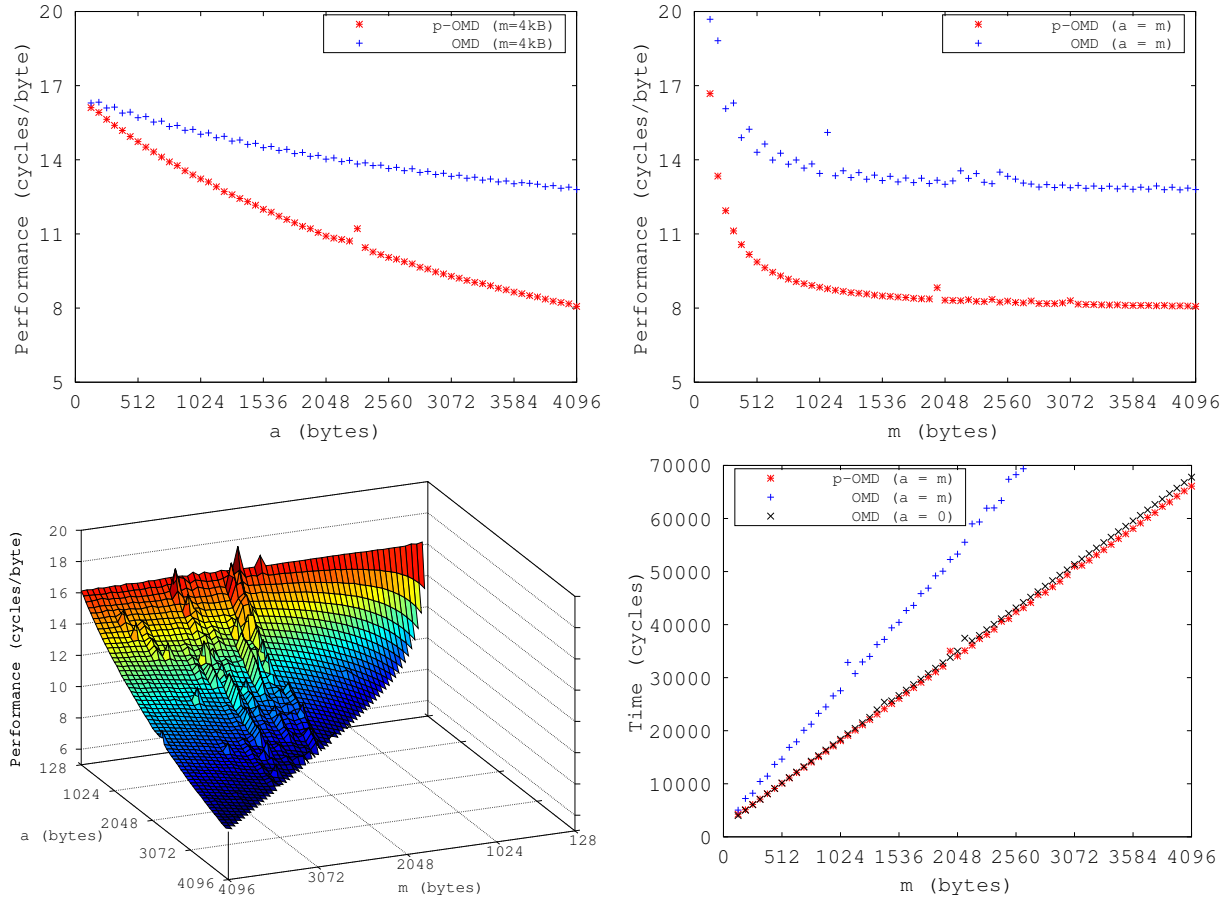


Fig. 7. Performance comparisons between OMD and p-OMD. **Top left**: encryption complexity with fixed message length. **Top right**: encryption complexity with equal message length and AD length. **Bottom right**: comparison of OMD without AD to OMD and p-OMD with AD. **Bottom left**: encryption complexity of p-OMD for varying message and AD lengths.

clear advantage over OMD. The bottom right graph confirms that the p-OMD provides an almost free authentication of associated data compared to OMD.

For both OMD and p-OMD, these measurements exclude the complexity of the precomputation step in computing $\Delta_{N,i,j}$ (see Section 4) which is done only once during the whole lifetime of a key. As an upper bound, we measure the complexity of the precomputation step that is sufficient to encrypt messages with length up to 2^{63} blocks. For OMD the precomputation step takes 5818 cycles while in p-OMD it requires 6863 cycles on average.

Acknowledgments. We would like to thank the anonymous reviewers of FSE 2015 for their constructive comments. We thank Tomer Ashur and Bart Mennink for pointing out a mistaken claim about authenticity under nonce misuse in the preproceedings of this paper. This work was partially supported by Microsoft Research under MRL Contract No. 2014-006 (DP1061305).

References

1. Implementation notes: amd64, titan0, crypto aead. http://bench.cr.yp.to/web-impl/amd64-titan0-crypto_aead.html
2. Secure Hash Standard (SHS) . NIST FIPS PUB 180-4 (Mar 2012)
3. Abed, F., Forler, C., Lucks, S.: Classification of the CAESAR Candidates. IACR Cryptology ePrint Archive 2014 (2014), http://eprint.iacr.org/2014/792
4. Ashur, T., Mennink, B.: Trivial Nonce-Misusing Attack on Pure OMD. Posted to CAESAR Mailing List (February 27, 2015)
5. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A Concrete Security Treatment of Symmetric Encryption. In: FOCS '97. pp. 394–403. IEEE Computer Society (1997)
6. Bellare, M., Goldreich, O., Mityagin, A.: The Power of Verification Queries in Message Authentication and Authenticated Encryption. IACR Cryptology ePrint Archive 2004, 309 (2004)
7. Bellare, M., Guérin, R., Rogaway, P.: XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. In: Coppersmith, D. (ed.) CRYPTO '95. LNCS, vol. 963, pp. 15–28. Springer (1995)
8. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer (2000)
9. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. J. Cryptology 21(4), 469–491 (2008)
10. Bellare, M., Rogaway, P.: Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 317–330. Springer (2000)
11. Bernstein, D.J.: Cryptographic competitions: CAESAR. http://competitions.cr.yp.to
12. 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)
13. Cogliani, S., Maimut, D., Naccache, D., do Canto, R.P., Reyhanitabar, R., Vaudenay, S., Vizár, D.: Offset Merkle-Damgård (OMD) version 1.0: A CAESAR Proposal (Mar 2014), http://competitions.cr.yp.to/round1/omdv10.pdf
14. Cogliani, S., Maimut, D., Naccache, D., do Canto, R.P., Reyhanitabar, R., Vaudenay, S., Vizár, D.: OMD: A Compression Function Mode of Operation for Authenticated Encryption. In: Joux, A., Youssef, A. (eds.) SAC 2014. Lecture Notes in Computer Science, vol. 8781. Springer (2014)
15. Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 196–215. Springer (2012)
16. Guilford, J., Cote, D., Gopal, V.: Fast SHA512 Implementations on Intel® Architecture Processors (Nov 2012), http://www.intel.com/content/www/us/en/intelligent-systems/intel-technology/fast-sha512-implementations-ia-processors-paper.html
17. Guilford, J., Yap, K., Gopal, V.: Fast SHA-256 Implementations on Intel® Architecture Processors (May 2012), http://www.intel.com/content/www/us/en/intelligent-systems/intel-technology/sha-256-implementations-paper.html
18. Gulley, S., Gopal, V., Yap, K., Feghali, W., Guilford, J., Wolrich, G.: Intel® SHA Extensions: New Instructions Supporting the Secure Hash Algorithm on Inte® Architecture Processors (Jul 2013), https://software.intel.com/sites/default/files/article/402097/intel-sha-extensions-white-paper.pdf
19. Katz, J., Yung, M.: Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 284–299. Springer (2001)
20. Krovetz, T., Rogaway, P.: The Software Performance of Authenticated-Encryption Modes. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 306–327. Springer (2011)

21. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering Generic Composition. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 8441, pp. 257–274. Springer (2014)
22. Reyhanitabar, R., Vaudenay, S., Vizár, D.: Misuse-Resistant Variants of the OMD Authenticated Encryption Mode. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S. (eds.) ProvSec 2014. Lecture Notes in Computer Science, vol. 8782, pp. 55–70. Springer (2014)
23. Rogaway, P.: Authenticated-Encryption with Associated-Data. In: ACM Conference on Computer and Communications Security. pp. 98–107 (2002)
24. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: ASIACRYPT. pp. 16–31 (2004)
25. Rogaway, P.: Nonce-Based Symmetric Encryption. In: Roy, B.K., Meier, W. (eds.) FSE. LNCS, vol. 3017, pp. 348–359. Springer (2004)
26. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In: ACM Conference on Computer and Communications Security. pp. 196–205 (2001)
27. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: EUROCRYPT. pp. 373–390 (2006)
28. Yasuda, K.: Boosting Merkle-Damgård Hashing for Message Authentication. In: Kurosawa, K. (ed.) ASIACRYPT 2007. Lecture Notes in Computer Science, vol. 4833, pp. 216–231. Springer (2007)

A The rational behind the masking sequence $\Delta_{N,i,j}$

In this section, we explain the design of the masking function $\Delta_K(N,i,j)$ in p-OMD and show that it fulfils the required security properties.

p-OMD uses the XE construction [24] to instantiate a tweakable PRF $\widetilde{F} : \mathcal{K} \times \mathcal{T} \times (\{0,1\}^n \times \{0,1\}^m) \to \{0,1\}^n$ using a regular PRF $F : \mathcal{K} \times (\{0,1\}^n \times \{0,1\}^m) \to \{0,1\}^n$ by defining $\widetilde{F}_K^{\langle T \rangle}(X,Y) = F_K((X \oplus \Delta_K(T)), Y)$ for every $T \in \mathcal{T}, K \in \mathcal{K}$, $X \in \{0,1\}^n, Y \in \{0,1\}^m$ where $\mathcal{T} = \{0,1\}^{|N|} \times \mathbb{N} \times \mathbb{N}$ and $\Delta_K(T)$ is the masking function of p-OMD.

The purpose of the masking function is to compute *masking offsets* $\Delta_{N,i,j}$ that are tweak-dependent and key-dependent in such a way that meets certain security and efficiency criteria.

Firstly, we note that the security proof of the XE construction still holds if $\Delta_K(N,i,j)$ satisfies the following two security conditions:

1. $\Pr[\Delta_K(N,i,j) = H] \leq 2^{-n}$ for any (N,i,j)
2. $\Pr[\Delta_K(N,i,j) \oplus \Delta_K(N',i',j') = H] \leq 2^{-n}$ for $(N,i,j) \neq (N',i',j')$.

Secondly, we note that the masking offsets should be computable *efficiently* in the order they appear in the masking sequence. We need to efficiently compute $\Delta_{N,i+1,j}$ if we have previously computed $\Delta_{N,i,j}$ for $0 \leq i$ (i.e. "increment" the i component), and also we need to efficiently compute $\Delta_{N,i,j'}$ if we have previously computed $\Delta_{N,i,j}$ for any $j, j' \in \{0,\ldots,15\}$ (i.e. "switch" the j component). To achieve this, we adapt the approach based on standard Gray Code sequence from [20].

Gray Code sequence. For a fixed positive $r > 0$ the Gray Code sequence is a special ordering $a : \{0,1,\cdots,2^r-1\} \to \{0,1\}^r$ of the set or r bit strings. It can be defined recursively as $a(0) = 0^r$ and $a(i) = a(i-1) \oplus 2^{\mathsf{ntz}(i)}$ if $i \geq 1$, where $\mathsf{ntz}(i)$ denotes the number of trailing zeros in the binary representation of i. The basic facts are that a is a bijection and $0 \leq a(i) \leq 2i$ (if represented as integer) for all i. We stress that we therefore have that (1) $a(i) \leq 2^{r+1}$ for all i, and (2) $a(i) \neq a(j)$ for all $i \neq j$.

Construction of the masking function. First recall that we only need to use 16 different values of the j component in p-OMD, i.e., all of its values are representable with 4 bits. Keeping this in mind we first define the sequence of Galois field elements $\gamma(i,j) \in GF(2^n)$ as $\gamma(i,j) = 2^4 \cdot a(i) \oplus j$ for $0 \leq j \leq 15$ (we represent j by as an $(n-6)$ bit string) and $0 \leq i < 2^{n-6}$ where the multiplication is done in the Galois field. Referring to the properties of the Gray Code sequence, we can verify that for all i, we have $\langle a(i) \rangle_n [n-1 \ldots n-5] = 0^5$ and $\langle (2^4 \cdot a(i)) \rangle_n [3 \ldots 0] = 0^4$. This implies that for every allowed pair i,j the value $\gamma(i,j)$ will be a unique element of $GF(2^n)$.

Finally, we define the masking function as

$$\Delta_K(N,i,j) = F_K\left(N || 10^n - |N| - 1, 0^m\right) \oplus \gamma(i,j) \cdot F_K\left(0^n, 0^m\right)$$

where F is the PRF used in p-OMD.

We can now easily verify that the two required security properties are met under the assumption that F is a good PRF. $\Delta_K(N,i,j)$ being a bitwise xor of two independent random n bit strings, we trivially have $\Pr[\Delta_K(N,i,j) = H] \leq 2^{-n}$ for any (N,i,j). To see if $\Pr[\Delta_K(N,i,j) \oplus \Delta_K(N',i',j') = H] \leq 2^{-n}$ for $(N,i,j) \neq (N',i',j')$, we consider two cases, either $N = N'$ or not. In the latter case, $N \neq N'$ implies that $\Delta_K(N,i,j) \oplus \Delta_K(N',i',j') = H$ is equivalent to the event that a bitwise xor of two independent random n-bit strings is equal to some specific value and we conclude that the required property is verified in this case. In the former case, $F_K(N||10^n - |N| - 1, 0^m) = F_K(N'||10^n - |N'| - 1, 0^m)$ so $\Delta_K(N,i,j) \oplus \Delta_K(N',i',j') = H$ occurs iff $(\gamma(i,j) \oplus \gamma(i',j')) \cdot F_K(0^n, 0^m) = H$. Note that we must have $(i,j) \neq (i',j')$ which together with properties listed above imply that the multiplier $(\gamma(i,j) \oplus \gamma(i',j')$ is non-zero. Thus, we conclude that the second condition is met in this case as well.

Compact representation. Let $L_* = F_K(0^n, 0^m)$. Then we have

$$\begin{aligned}\Delta_K(N,i,j) &= F_K\left(N||10^n - |N| - 1, 0^m\right) \oplus \gamma(i,j)L_* \\ &= F_K\left(N||10^n - |N| - 1, 0^m\right) \oplus a(i) \cdot 2^4 \cdot L_* \oplus j \cdot L_*.\end{aligned}$$

We further define $L_*(j) = j \cdot L_*$ for $0 \leq j \leq 15$ and $L(\ell) = 2^{4+\ell} \cdot L_*$ for $0 \leq \ell < n - 6$. Note that $L_*(1) = L_*$, $L_*(0) = 0^n$ and $L(0) = 2^4 \cdot L_*$. Thus we can write $\Delta_K(N,i,j) = F_K(N||10^n - |N| - 1, 0^m) \oplus a(i) \cdot L(0) \oplus L_*(j)$. We can derive two rules. First, keeping in mind the way we have defined the Gray Code sequence we can see that

$$\begin{aligned}\Delta_K(N,i,j) &= F_K\left(N||10^n - |N| - 1, 0^m\right) \oplus a(i) \cdot L(0) \oplus L_*(j) \\ &= F_K\left(N||10^n - |N| - 1, 0^m\right) \oplus \left(a(i-1) \oplus 2^{\mathsf{ntz}(i)}\right) \cdot L(0) \oplus L_*(j) \\ &= F_K\left(N||10^n - |N| - 1, 0^m\right) \oplus a(i-1) \cdot L(0) \oplus 2^{\mathsf{ntz}(i)} \cdot L(0) \oplus L_*(j) \\ &= \Delta_K(N, i-1, j) \oplus 2^{\mathsf{ntz}(i)} \cdot L(0) \\ &= \Delta_K(N, i-1, j) \oplus L(\mathsf{ntz}(i)).\end{aligned}$$

Secondly, for any j, j' from the acceptable range we have

$$\begin{aligned}\Delta_K(N,i,j') &= F_K\left(N||10^n - |N| - 1, 0^m\right) \oplus a(i) \cdot L(0) \oplus L_*(j') \\ &= F_K\left(N||10^n - |N| - 1, 0^m\right) \oplus a(i) \cdot L(0) \oplus L_*(j) \oplus L_*(j) \oplus L_*(j') \\ &= \Delta_K(N,i,j) \oplus j \cdot L_* \oplus j' \cdot L_* \\ &= \Delta_K(N,i,j) \oplus \left(\langle j \rangle_n \oplus \langle j' \rangle_n\right) \cdot L_* \\ &= \Delta_K(N,i,j) \oplus L_*\left(\mathsf{str2num}\left(\langle j \rangle_n \oplus \langle j' \rangle_n\right)\right).\end{aligned}$$

By combining these two rules we obtain the compact definition used in the specification of p-OMD in Section 4.