Improved Masking for Tweakable Blockciphers with Applications to Authenticated Encryption

Robert Granger[1⋆], Philipp Jovanovic[2], Bart Mennink[3⋆⋆], and Samuel Neves[4]

[1] École polytechnique fédérale de Lausanne, Switzerland
robert.granger@epfl.ch
[2] Faculty of Computer Science and Mathematics, University of Passau, Germany
jovanovic@fim.uni-passau.de
[3] Dept. Electrical Engineering, ESAT/COSIC, KU Leuven, and iMinds, Belgium
bart.mennink@esat.kuleuven.be
[4] CISUC, Dept. of Informatics Engineering, University of Coimbra, Portugal
sneves@dei.uc.pt

Abstract. A popular approach to tweakable blockcipher design is via masking, where a certain primitive (a blockcipher or a permutation) is preceded and followed by an easy-to-compute tweak-dependent mask. In this work, we revisit the principle of masking. We do so alongside the introduction of the tweakable Even-Mansour construction MEM. Its masking function combines the advantages of word-oriented LFSR- and powering-up-based methods. We show in particular how recent advancements in computing discrete logarithms over finite fields of characteristic 2 can be exploited in a constructive way to realize highly efficient, constant-time masking functions. If the masking satisfies a set of simple conditions, then MEM is a secure tweakable blockcipher up to the birthday bound. The strengths of MEM are exhibited by the design of fully parallelizable authenticated encryption schemes OPP (nonce-respecting) and MRO (misuse-resistant). If instantiated with a reduced-round BLAKE2b permutation, OPP and MRO achieve speeds up to 0.55 and 1.06 cycles per byte on the Intel Haswell microarchitecture, and are able to significantly outperform their closest competitors.

Keywords. Tweakable Even-Mansour, masking, optimization, discrete logarithms, authenticated encryption, BLAKE2.

1 Introduction

Authenticated encryption (AE) has faced significant attention in light of the ongoing CAESAR competition [17]. An AE scheme aims to provide both confidentiality and integrity of processed data. While the classical approach is predominantly

⋆ Robert Granger is supported by the Swiss National Science Foundation via grant number 200021-156420.

⋆⋆ Bart Mennink is a Postdoctoral Fellow of the Research Foundation – Flanders (FWO). He is supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007).

blockcipher-based, where an underlying blockcipher is used to encrypt, novel approaches start from a permutation and either rely on Sponge-based principles or on the fact that the Even-Mansour construction $E(K, M) = P(K \oplus M) \oplus K$ is a blockcipher.

Characteristic for the majority of blockcipher-based AE schemes is that they rely on a tweakable blockcipher where changes in the tweak can be realized efficiently. The most prominent example of this is the OCB2 mode which internally uses the XEX tweakable blockcipher [71]:

$$\mathrm{XEX}(K, (X, i_0, i_1, i_2), M) = E(K, \delta \oplus M) \oplus \delta \ ,$$

where $\delta = \mathbf{2}^{i_0}\mathbf{3}^{i_1}\mathbf{7}^{i_2}E(K, X)$. The idea is that every associated data or message block is transformed using a different tweak, where increasing i_0, i_1, or i_2 can be done efficiently. This approach is furthermore used in second-round CAESAR candidates AEZ, COPA, ELmD, OTR, POET, and SHELL. Other approaches to masking include Gray code ordering (used in OCB1 and OCB3 [72,58] and OMD) and the word-oriented LFSR-based approach where $\delta = \varphi^i(E(K, X))$ for some LFSR φ (suggested by Chakraborty and Sarkar [20]).

The same masking techniques can also be used for permutation-based tweakable blockciphers. For instance, Minalpher uses the Tweakable Even-Mansour (TEM) construction [75] with XEX-like masking, and similar for Prøst. This TEM construction has faced generalizations by Cogliati et al. [26,27] and Mennink [65], but none of them considers efficiency improvements of the masking.

1.1 Masked Even-Mansour (MEM) Tweakable Cipher

As a first contribution, we revisit the state of the art in masking with the introduction of the "Masked Even-Mansour" tweakable blockcipher in Section 3. At a high level, MEM is a Tweakable Even-Mansour construction, where the masking combines ideas from both word-oriented LFSR- and powering-up-based masking. As such, MEM combines "the best of both" masking approaches, leading to significant improvements in simplicity, error-proneness, and efficiency.

In more detail, let P be a b-bit permutation. MEM's encryption functions is defined as

$$\widetilde{E}(K, X, \bar{i}, M) = P(\delta(K, X, \bar{i}) \oplus M) \oplus \delta(K, X, \bar{i}) \ ,$$

where $\bar{i} = (i_0, \ldots, i_{u-1})$ and where the masking function is of the form

$$\delta(K, X, \bar{i}) = \varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}(P(X \parallel K)) \ ,$$

for a certain set of LFSRs $(\varphi_0, \ldots, \varphi_{u-1})$. MEM's decryption function $\widetilde{D}$ is specified analogously but using P^{-1} instead of P.

The tweak space and the list of LFSRs are clearly required to satisfy some randomness condition. Indeed, if a distinguisher can choose a list of tweaks $\bar{i}$ such that $\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}(L)$ for a uniformly random L offers no or limited

entropy, it can easily distinguish MEM from a random primitive. A similar case applies if the distinguisher can make two different maskings collide with high probability. Denote by ϵ the minimal amount of entropy offered by the functions $\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}$ and $\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0} \oplus \varphi_{u-1}^{i'_{u-1}} \circ \cdots \circ \varphi_0^{i'_0}$ for any two maskings $\bar{i}, \bar{i}'$ (see Definition 1 for the formal definition). Then, we prove that MEM is a secure tweakable blockcipher in the ideal permutation model up to $\frac{4.5q^2+3qp}{2^\epsilon} + \frac{p}{2^k}$, where q is the number of construction queries, p the number of primitive queries, and k the key length. The security proof follows Patarin's H-coefficient technique, which has shown its use to Even-Mansour security proofs before in, among others, [24,23,4,28,26,66].

To guarantee that the maskings offer enough randomness, it is of pivotal importance to define a proper domain of the masking. At the least, the functions $\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}$ should be different for all possible choices of $\bar{i}$, or more formally, such that there do not exist $\bar{i}, \bar{i}'$ such that

$$\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0} = \varphi_{u-1}^{i'_{u-1}} \circ \cdots \circ \varphi_0^{i'_0} \ .$$

Guaranteeing this requires the computation of discrete logarithms. For small cases, such as $b = 64$ and $b = 128$, we can inherit the computations from Rogaway for XEX [71]. For instance, for $b = 128$, it is known that $u = 3$, $(\varphi_0, \varphi_1, \varphi_2) = (\mathbf{2}, \mathbf{3}, \mathbf{7})$, and $(i_0, i_1, i_2) \in \{-2^{108}, \ldots, 2^{108}\} \times \{-2^7, \ldots, 2^7\} \times \{-2^7, \ldots, 2^7\}$ does the job.

We extend the XEX approach to much larger block sizes by taking advantage of the recent breakthroughs in the computation of discrete logs in small characteristic fields, beginning with [33], followed by [48]. Computation of individual discrete logarithms for the 1024-bit block used in our MEM instantiation takes about 8 hrs on a single core of a standard desktop computer, after an initial precomputation, applicable to all logarithms, of 33.3 hrs. Larger blocks are also attainable, rendering workarounds such as subgroups [77] or different modes [74] largely unnecessary.

Peculiarly, there have been uses of XEX for state sizes larger than $b = 128$ bits, even though it has been unclear what restrictions on the indices are due. For instance, Prøst [53] defines a COPA and OTR instance for a 256- and 512-bit blockcipher; both use maskings of the form $\mathbf{2}^{i_0}\mathbf{3}^{i_1}\mathbf{7}^{i_2}$ for i_0 ranging between 0 and the maximal message length. For COPA, it has $(i_1, i_2) \in \{0, \ldots, 5\} \times \{0, 1\}$ and for OTR it has $(i_1, i_2) \in \{0, 1\} \times \{0\}$. The security proof of Prøst never formally computes conditions on the indices, and simply inherits the conditions for $b = 128$. By computing the discrete logarithms in the respective fields—a computationally easy task, demonstrated in Section 3.6—we can confirm that the tweaks are unique for $i_0 \in \{0, \ldots, 2^{246} - 1\}$ in the 256-bit block case, and $i_0 \in \{0, \ldots, 2^{505} - 1\}$ in the 512-bit block case.

1.2 Application to Nonce-Based AE

As first application, we present the Offset Public Permutation (OPP) mode in Section 4, a parallelizable nonce-based AE based on MEM. It can be considered

as a permutation-based generalization of OCB3 [58] to arbitrary block sizes using permutations and using the improved masking from MEM. Particularly, assuming security of MEM, the proof of [58] mostly carries over, and we obtain that OPP behaves like a random AE up to attack complexity dominated by $\min\{2^{b/2}, 2^k\}$. OPP also shows similarities with Kurosawa's adaption of IAPM and OCB to the permutation-based setting [59].

Using the masking techniques described later in this paper, OPP has excellent performance when compared to contemporary permutation-based schemes, such as first-round CAESAR [17] submissions Artemia, Ascon, CBEAM, ICEPOLE, Keyak, NORX, π-Cipher, PRIMATEs, and STRIBOB, SpongeWrap schemes in general [11,66], and sp-AELM [1]. OPP improves upon these by being inherently parallel and rate-1; the total overhead of the mode reduces to 2 extra permutation calls and the aforementioned efficient masking.

In particular, when instantiated with a reduced-round BLAKE2b permutation [6], OPP achieves a peak speed of 0.55 cycles per byte on an Intel Haswell processor (see Section 8). This is faster than any other permutation-based CAESAR submission. In fact, there are only a few CAESAR ciphers, such as Tiaoxin (0.28 cpb) or AEGIS (0.35 cpb), which are faster than the above instantiation of OPP. However, both require AES-NI to reach their best performance and neither of them is arbitrarily parallelizable.

1.3 Application to Nonce-Misuse Resistant AE

We also consider permutation-based authenticated encryption schemes that are resistant against nonce-reuse. We consider "full" nonce-misuse resistance, where the output is completely random for different inputs, but we remark that similarly schemes can be designed to achieve "online" nonce-misuse resistance [29,44], for instance starting from COPA [3]. It is a well-known result that nonce-misuse resistant schemes are inherently offline, meaning that two passes over the data must be made in order to perform the authenticated encryption.

The first misuse-resistant AE we consider is the parallelizable Misuse-Resistant Offset (MRO) mode (Section 5). It starts from OPP, but with the absorption performed on the entire data and with encryption done in counter mode instead.[5] As the underlying MEM is used by the absorption and encryption parts for different maskings, we can view the absorption and encryption as two independent functions and a classical MAC-then-Encrypt security proof shows that MRO is secure up to complexity dominated by $\min\{2^{b/2}, 2^k, 2^{t/2}\}$.

Next, we consider Misuse-Resistant Sponge (MRS) in Section 6. It is not directly based on MEM; it can merely be seen as a cascaded evaluation of the Full-state Keyed Duplex of Mennink et al. [66], a generalization of the Duplex of Bertoni et al. [11]: a first evaluation computes the tag on input of all data, the second evaluation encrypts the message with the tag functioning as the nonce.

[5] MRO's structure is comparable with the independently introduced Synthetic Counter in Tweak [46,47].

MRS is mostly presented to suit the introduction of the Misuse-Resistant Sponge-Offset hybrid (MRSO) in Section 7, which absorbs like MRS and encrypts like MRO. (It is also possible to consider the complementary Offset-Sponge hybrid, but we see no potential applications of this construction.) The schemes MRS and MRSO are proven secure up to complexity approximately $\min\{2^{c/2}, 2^{k/2}, 2^{t/2}\}$ and $\min\{2^{(b-\tau)/2}, 2^k, 2^{t/2}\}$, respectively.

While various blockcipher-based fully misuse-resistant AE schemes exist (such as SIV [73], GCM-SIV [40], HS1-SIV [57], AEZ [43], Deoxys$^=$ and Joltik$^=$ [46,47] (using Synthetic Counter in Tweak mode), and DAEAD [21]), the state of the art for permutation-based schemes is rather scarce. In particular, the only misuse-resistant AE schemes known in literature are Haddoc and Mr. Monster Burrito by Bertoni et al. [13]. Haddoc lacks a proper formalization but it appears to be similar to MRSO, and the security and efficiency bounds mostly carry over. Mr. Monster Burrito is a proof of concept to construct a permutation-based robust AE comparable with AEZ [43], but it is four-pass and thus not very practical.[6]

When instantiated with a reduced-round BLAKE2b permutation, MRO achieves a peak speed of 1.06 cycles per byte on the Intel Haswell platform (see Section 8). This puts MRO on the same level as AES-GCM-SIV [40] (1.17 cpb), which, however, requires AES-NI to reach its best performance. We further remark that MRO is also more efficient than MRSO, and thus the Haddoc mode.

2 Notation

We set $\mathbb{N} = \{0, 1, \ldots\}$ and $\mathbb{N}^+ = \{1, 2, \ldots\}$, and denote by $\mathbb{F}_{2^n}$ the finite field of order 2^n with $n \geq 1$. A b-bit string X is an element of $\{0,1\}^b$ or equivalently of the $\mathbb{F}_2$-vector space $\mathbb{F}_2^b$. The length of a bit string X in bits is denoted by $|X|$ $(= b)$ and in r-bit blocks by $|X|_r$. For example, the size of X in bytes is $|X|_8$. The bit string of length 0 is identified with ε. The *concatenation* of two bit strings X and Y is denoted by $X \parallel Y$. The encoding of an integer x as an n-bit string is denoted by $\langle x \rangle_n$. The symbols $\neg$, $\vee$, $\wedge$, $\oplus$, $\ll$, $\gg$, $\lll$, and $\ggg$, denote bit-wise *NOT*, *OR*, *AND*, *XOR*, *left-shift*, *right-shift*, *left-rotation*, and *right-rotation*, respectively.

Given a b-bit string $X = x_0 \parallel \cdots \parallel x_{b-1}$ we define $\mathsf{left}_l(X) = x_0 \parallel \cdots \parallel x_{l-1}$ to be the l left-most and $\mathsf{right}_r(X) = x_{b-r} \parallel \cdots \parallel x_{b-1}$ to be the r right-most bits of X, respectively, where $1 \leq l, r \leq b$. In particular, note that $X = \mathsf{left}_l(X) \parallel \mathsf{right}_{b-l}(X) = \mathsf{left}_{b-r}(X) \parallel \mathsf{right}_r(X)$. We define the following mapping functions which extend a given input string X to a multiple of the block size b and cut it into chunks of b bits:

$$\mathsf{pad}_b^0 : \{0,1\}^* \to (\{0,1\}^b)^+, X \mapsto X \parallel 0^{(b-|X|) \bmod b} ,$$

[6] We remark that the state of the art on online misuse-resistant permutation-based AE is a bit more advanced. For instance, APE [2] is online misuse-resistant, and achieves security against the release of unverified plaintext, but satisfies the undesirable property of backwards decryption. Also Minalpher and Prøst-COPA are online misuse-resistant.

$$\mathsf{pad}_b^{10} : \{0,1\}^* \to (\{0,1\}^b)^+, X \mapsto X \parallel 1 \parallel 0^{(b-|X|-1) \bmod b} .$$

The set of all permutations of *width* $b \in \mathbb{N}$ bits is denoted by $\mathsf{Perm}(b)$. The parameters $k, n, \tau \in \mathbb{N}$ conventionally define the size of the key, nonce, and tag, respectively, for which we require that $n \leq b - k - 1$. In the context of Sponge functions $r \in \mathbb{N}$ and $c \in \mathbb{N}$ denote *rate* and *capacity* such that $b = r + c$, and we require $k \leq c$.

When writing $X \xleftarrow{\$} \mathcal{X}$ for some finite set $\mathcal{X}$, we mean that X gets sampled uniformly at random from $\mathcal{X}$.

2.1 Distinguishers

A distinguisher $\mathbf{D}$ is a computationally unbounded probabilistic algorithm. By $\mathbf{D}^{\mathcal{O}}$ we denote the setting that $\mathbf{D}$ is given query access to an oracle $\mathcal{O}$: it can make queries to $\mathcal{O}$ adaptively, and after this, the distinguisher outputs 0 or 1. If we consider two different oracles $\mathcal{O}$ and $\mathcal{P}$ with the same interface, we define the distinguishing advantage of $\mathbf{D}$ by

$$\Delta_{\mathbf{D}}(\mathcal{O}\,;\,\mathcal{P}) = \left|\mathbf{Pr}\left(\mathbf{D}^{\mathcal{O}} = 1\right) - \mathbf{Pr}\left(\mathbf{D}^{\mathcal{P}} = 1\right)\right| . \tag{1}$$

Here, the probabilities are taken over the randomness from $\mathcal{O}$ and $\mathcal{P}$. The distinguisher is usually bounded by a limited set of resources, e.g., it is allowed to make at most q queries to its oracle. We will use the definition of Δ for our formalization of the security (tweakable) blockciphers and authenticated encryption. Later in the paper, Δ will furthermore be used to measure the security of PRFs, and others.

2.2 Tweakable Blockciphers

Let $\mathcal{T}$ be a set of "tweaks." A tweakable blockcipher $\widetilde{E} : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^b \to \{0,1\}^b$ is a function such that for every key $K \in \{0,1\}^k$ and tweak $T \in \mathcal{T}$, $\widetilde{E}(K, T, \cdot)$ is a permutation in $\mathsf{Perm}(b)$. We denote its inverse by $\widetilde{E}^{-1}(K, T, \cdot)$. Denote by $\widetilde{\mathsf{Perm}}(\mathcal{T}, b)$ the set of families of tweakable permutations $\widetilde{\pi}$ such that $\widetilde{\pi}(T, \cdot) \in \mathsf{Perm}(b)$ for every $T \in \mathcal{T}$.

The conventional security definitions for tweakable blockciphers are tweakable pseudorandom permutation (TPRP) security and *strong* TPRP (STPRP) security: in the former, the distinguisher can only make forward construction queries, while in the latter it is additionally allowed to make inverse construction queries. We will consider a mixed security notion, where the distinguisher may only make forward queries for a subset of tweaks. It is inspired by earlier definitions from Rogaway [71] and Andreeva et al. [3].

Let $P \xleftarrow{\$} \mathsf{Perm}(b)$ be a b-bit permutation, and consider a tweakable blockcipher $\widetilde{E}$ based on permutation P. Consider a partition $\mathcal{T}_0 \cup \mathcal{T}_1 = \mathcal{T}$ of the tweak space into *forward-only* tweaks $\mathcal{T}_0$ and *forward-and-inverse* tweaks $\mathcal{T}_1$. We define the

mixed tweakable pseudorandom permutation (MTPRP) security of $\widetilde{E}$ against a distinguisher $\mathbf{D}$ as

$$\mathbf{Adv}^{\widetilde{\text{mprp}}}_{\widetilde{E},P}(\mathbf{D}) = \Delta_{\mathbf{D}}(\widetilde{E}^{\pm}_K, P^{\pm}\,;\, \widetilde{\pi}^{\pm}, P^{\pm})\ , \tag{2}$$

where the probabilities are taken over the random choices of K, $\widetilde{\pi}$, and P. The distinguisher is not allowed to query $\widetilde{E}^{-1}_K$ for tweaks from $\mathcal{T}_0$. By $\mathbf{Adv}^{\widetilde{\text{mprp}}}_{\widetilde{E},P}(q,p)$ we denote the maximum advantage over all distinguishers that make at most q construction queries and at most p queries to $P^{\pm}$.

Note that the definition of MTPRP matches TPRP if $(\mathcal{T}_0, \mathcal{T}_1) = (\mathcal{T}, \emptyset)$ and STPRP if $(\mathcal{T}_0, \mathcal{T}_1) = (\emptyset, \mathcal{T})$. It is a straightforward observation that if a tweakable cipher $\widetilde{E}$ is MTPRP for two sets $(\mathcal{T}_0, \mathcal{T}_1)$, then it is MTPRP for $(\mathcal{T}_0 \cup \{T\}, \mathcal{T}_1 \setminus \{T\})$ for any $T \in \mathcal{T}_1$. Ultimately, this observation implies that an STPRP is a TPRP.

2.3 Authenticated Encryption

Let $\Pi = (\mathcal{E}, \mathcal{D})$ be an authenticated encryption (AE) scheme which is keyed via a secret key $K \in \{0,1\}^k$ and operates as follows:

$$\mathcal{E}_K(N, H, M) = (C, T)\ ,$$
$$\mathcal{D}_K(N, H, C, T) = M/\perp\ .$$

Here, N is the nonce, H the associated data, M the message, C the ciphertext, and T the tag. In our analysis, we always have $|M| = |C|$, and we require that

$$\mathcal{D}_K(N, H, \mathcal{E}_K(N, H, M)) = M$$

for all N, H, M. By $\$_{\mathcal{E}}$ we define the idealized version of $\mathcal{E}_K$, which returns $(C, T) \xleftarrow{\$} \{0,1\}^{|M|+\tau}$ for every input. Finally, we denote by $\perp$ a function that returns $\perp$ upon every query.

Our AE schemes are based on a b-bit permutation P, and we will analyze the security of them in the setting where P is a random permutation: $P \xleftarrow{\$} \mathsf{Perm}(b)$. Following, among others, Rogaway and Shrimpton [73], Namprempre et al. [68], and Gueron and Lindell [40], we define the AE security of Π against a distinguisher $\mathbf{D}$ as

$$\mathbf{Adv}^{\text{ae}}_{\Pi,P}(\mathbf{D}) = \Delta_{\mathbf{D}}(\mathcal{E}_K, \mathcal{D}_K, P^{\pm}\,;\, \$_{\mathcal{E}}, \perp, P^{\pm})\ , \tag{3}$$

where the probabilities are taken over the random choices of K, $\$_{\mathcal{E}}$, and P. The distinguisher is not allowed (i) to repeat any query and (ii) to relay the output of $\mathcal{E}_K$ to the input of $\mathcal{D}_K$. Note that we do *not* a priori require the distinguisher to be nonce-respecting: depending on the setting, it may repeat nonces at its own discretion. We will always explicitly mention whether we consider nonce-respecting or nonce-reusing distinguishers. By $\mathbf{Adv}^{\text{ae}}_{\Pi,P}(q_{\mathcal{E}}, q_{\mathcal{D}}, \sigma, p)$ we denote the maximum advantage over all (nonce-respecting/reusing) distinguishers that make at most $q_{\mathcal{E}}$ queries to the encryption oracle and at most $q_{\mathcal{D}}$ to the decryption oracle, of total length at most σ padded blocks, and that make at most p queries to $P^{\pm}$.

3 Tweakable Even-Mansour with General Masking

We present the tweakable Even-Mansour construction MEM. Earlier appearances of tweakable Even-Mansour constructions include Sasaki et al. [75], Cogliati et al. [26], and Mennink [65], but these constructions target different settings, do not easily capture the improved maskings as introduced below, and are therefore not applicable in this work.

Our specification can be seen as a generalization of both the XE(X) construction of Rogaway [71] and the tweakable blockcipher from Chakraborty and Sarkar [20] to the permutation-based setting. While Rogaway limited himself to 128-bit fields, we realize our approach to fields well beyond the reach of Pohlig-Hellman: historically the large block size would have been a severe obstruction, as observed in works by Yasuda and Sarkar [77,74], and some schemes simply ignored the issue [53]. The breakthroughs in computing discrete logarithms in small characteristic fields [33,48,8,37] allow to easily pass the 128-bit barrier. In particular, for blocks of 2^n bits, it is eminently practical to compute discrete logarithms for $n \leq 13$. Further details of our solution of discrete logarithms over $\mathbb{F}_{2^{512}}$ and $\mathbb{F}_{2^{1024}}$ are described in Section 3.6.

3.1 Definition

Let $b \in \mathbb{N}^+$ and $P \in \mathsf{Perm}(b)$. In the following we specify MEM, a *tweakable Even-Mansour block cipher with general masking* $(\widetilde{E}, \widetilde{D})$ where $\widetilde{E}$ and $\widetilde{D}$ denote encryption and decryption functions, respectively. Let $u \geq 1$, and let $\Phi = \{\varphi_0, \ldots, \varphi_{u-1}\}$ be a set of functions $\varphi_j : \{0,1\}^b \to \{0,1\}^b$. Consider a *tweak space* $\mathcal{T}$ of the form

$$\mathcal{T} \subseteq \{0,1\}^{b-k} \times \mathbb{N}^u \tag{4}$$

and specify the *general masking function* $\delta : \{0,1\}^k \times \mathcal{T} \to \{0,1\}^b$ as

$$\delta : (K, X, i_0, \ldots, i_{u-1}) \mapsto \varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}(P(X \parallel K)) \ .$$

By convention, we set $\varphi_j^{i_j} = id$ for $i_j = 0$, for each $0 \leq j \leq u-1$. For brevity of notation we write $\bar{i} = (i_0, \ldots, i_{u-1})$, and set

$$\mathcal{T}_{\bar{i}} = \left\{ \bar{i} \mid \exists X \text{ such that } (X, \bar{i}) \in \mathcal{T} \right\} \ .$$

The encryption function $\widetilde{E} : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^b \to \{0,1\}^b$ is now defined as

$$\widetilde{E} : (K, X, \bar{i}, M) \mapsto P(\delta(K, X, \bar{i}) \oplus M) \oplus \delta(K, X, \bar{i}) \ ,$$

where M denotes the to be encrypted message. The decryption function $\widetilde{D} : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^b \to \{0,1\}^b$ is defined analogously as

$$\widetilde{D} : (K, X, \bar{i}, C) \mapsto P^{-1}(\delta(K, X, \bar{i}) \oplus C) \oplus \delta(K, X, \bar{i}) \ ,$$

where C denotes the to be decrypted ciphertext. Note that the usual block cipher property $\widetilde{D}(K, X, \bar{i}, \widetilde{E}(K, X, \bar{i}, M)) = M$ is obviously satisfied. Throughout the document, we will often use the following shorthand notation for $\widetilde{E}^{\bar{i}}_{K,X}(M) = \widetilde{E}(K, X, \bar{i}, M)$, $\widetilde{D}^{\bar{i}}_{K,X}(C) = \widetilde{D}(K, X, \bar{i}, C)$, and $\delta^{\bar{i}}_{K,X} = \delta(K, X, \bar{i})$.

3.2 Security

Eq. (4) already reveals that we require some kind of restriction on $\mathcal{T}$. Informally, we require the masking functions $\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}$ to generate pairwise independent values for different tweaks. More formally, we define proper tweak spaces in Definition 1. This definition is related to earlier observations in Rogaway [71] and Sarkar [74].

Definition 1. *Let $u \geq 1$ and $\Phi = \{\varphi_0, \ldots, \varphi_{u-1}\}$ be a set of functions. The tweak space $\mathcal{T}$ is* ϵ-proper *relative to the function set Φ if the following two properties are satisfied.*

1. *For any $y \in \{0,1\}^b$, $(i_0, \ldots, i_{u-1}) \in \mathcal{T}_{\bar{i}}$, and uniformly random $L \xleftarrow{\$} \{0,1\}^b$:*

$$\Pr\left[\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}(L) = y\right] = 2^{-\epsilon} \ .$$

2. *For any $y \in \{0,1\}^b$, distinct $(i_0, \ldots, i_{u-1}), (i'_0, \ldots, i'_{u-1}) \in \mathcal{T}_{\bar{i}}$, and uniformly random $L \xleftarrow{\$} \{0,1\}^b$:*

$$\Pr\left[\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}(L) \oplus \varphi_{u-1}^{i'_{u-1}} \circ \cdots \circ \varphi_0^{i'_0}(L) = y\right] = 2^{-\epsilon} \ .$$

We are now ready to prove the security of MEM.

Theorem 2. *Let $u \geq 1$ and $\Phi = \{\varphi_0, \ldots, \varphi_{u-1}\}$ be a set of functions. Let $P \xleftarrow{\$} \mathsf{Perm}(b)$. Assume that the tweak space $\mathcal{T}$ is ϵ-proper relative to Φ. Let $\mathcal{T}_0 \cup \mathcal{T}_1 = \mathcal{T}$ be a partition such that $(0, \ldots, 0) \notin \mathcal{T}_{1_{\bar{i}}}$. Then,*

$$\mathbf{Adv}^{\widetilde{\text{mprp}}}_{\widetilde{E},P}(q,p) \leq \frac{4.5q^2}{2^\epsilon} + \frac{3qp}{2^\epsilon} + \frac{p}{2^k} \ .$$

The proof is given in Supporting Material A. It is based on Patarin's H-coefficient technique [70,24], and borrows ideas from [71,20,74,65].

3.3 History of Masking

Many masking functions have been suggested since such schemes were invented. IAPM [52] proposed the masking to be a subset sum of c encrypted blocks derived from the nonce, where 2^c is the maximum number of blocks a message can have. In the same document Jutla also suggested masking the jth block with $(j+1)K + IV \bmod p$, for some prime p near the block size. XCBC [30] used a similar masking function, but replaced arithmetic modulo p by arithmetic modulo 2^b, at the cost of some tightness in security reductions.

OCB [72,71,58] and PMAC [14] used the field $\mathbb{F}_{2^b}$ for their masking. There are two different masking functions used in variants of OCB:

– The powering-up method of OCB2 [71] computes $\varphi^i(L) = x^i \cdot L$, where $\cdot$ is multiplication in $\mathbb{F}_{2^b}$, and x is a generator of the field.

– The Gray code masking of OCB1 [72] and OCB3 [58] computes $\varphi^i(L) = \gamma_i \cdot L$, where $\gamma_i = i \oplus (i \gg 1)$. This method requires a single XOR to compute $\varphi^{i+1}(L)$ given $\varphi^i(L)$, provided a precomputation of $\log_2 |M|$ multiples of L is carried out in advance. Otherwise, up to $\log_2 i$ field doublings are required to obtain $\gamma_i \cdot L$. This Gray code trick was also applicable to IAPM's subset-sum masking.

Another family of masking functions, word-oriented LFSRs, was suggested by Chakraborty and Sarkar [20]. Instead of working directly with the polynomial representation $\mathbb{F}_2[x]/f$ for some primitive polynomial f, word-oriented LFSRs treat the block as the field $\mathbb{F}_{2^{wn}}$, where w is the native word size. Thus, the block can be represented as a polynomial of degree n over F_{2^w}, which makes the arithmetic more software-friendly. A further generalized variant of this family of generators is described (and rejected) in [58, Appendix B], who also attribute the same technique to [78]. Instead of working with explicitly-constructed field representations, one starts by trying to find a $b \times b$ matrix $M \in \mathrm{GL}(b, \mathbb{F}_2)$ that is very efficient to compute. Then, if this matrix has a primitive minimal polynomial of degree b, this transformation is in fact isomorphic to $\mathbb{F}_{2^b}$ and has desirable masking properties. The masking function is then $\varphi^i(L) = M^i \cdot L$.

Although the above maximal-period matrix recursions have only recently been suggested for use in efficient masking, the approach has been long studied by designers of non-cryptographic pseudorandom generators. For example, Niederreiter [69, Section 4] proposed a pseudorandom generator design based on a matrix recursion. Later methods, like the Mersenne Twister family [63] and the Xorshift [62] generator, improved the efficiency significantly by cleverly choosing the matrix shape to be CPU-friendly.

More recently, Minematsu [67] suggested a different approach to masking based on data-dependent rotation. In particular,

$$\varphi^i(L) = \bigoplus_{0 \le j < b} \begin{cases} (L \lll j) & \text{if } \lfloor i/2^j \rfloor \bmod 2 = 1 \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

where the block size b is prime. With Gray code ordering, one only needs one rotation and XOR per sequential mask *without* storing previous masks. That being said, the prime block size is inconvenient, and data-dependent rotation is a relatively expensive operation compared to some of the previous techniques.

3.4 Proposed Masking for $u = 1$

We loosely follow the Xorshift [62] design approach for our masking procedure. Let $b = nw$ be the block size, interpreted as n words of w bits. We begin with fast linear operations available in most current CPUs and encode them as $w \times w$ matrices. More precisely, we denote by 0 the all-zero matrix, by I the identity matrix, by SHL_c and SHR_c matrices corresponding to left- and right-shift by c bits, by ROT_c the matrix realizing left-rotation by c bits, and by AND_c the matrix corresponding to bit-wise AND with a constant c. Then, we construct

block matrices using those operations in a way that minimizes computational effort. To maximize efficiency we consider $b \times b$ matrices over $\mathbb{F}_2$ of the form

$$M = \begin{pmatrix} 0 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I \\ X_0 & X_1 & \cdots & X_{n-1} \end{pmatrix} \tag{5}$$

with $X_i \in \{0, I, \mathsf{SHL}_c, \mathsf{SHR}_c, \mathsf{ROT}_c, \mathsf{AND}_c\}$ where $\dim(X_i) = w$ for $0 \leq i \leq n-1$. We favor matrices where only a minimal amount of X_i are nonzero. For a concrete selection of $X_0, \ldots, X_{n-1}$ we check if the matrix order is maximal; if so, this matrix is suitable for a masking function that respects the conditions listed above.

Testing candidate masks for maximal order may be efficiently performed without any explicit matrix operations. Given a candidate linear map corresponding to a matrix M of the form Eq. (5),

$$(x_0, \ldots, x_{n-1}) \mapsto (x_1, \ldots, x_{n-1}, f(x_0, \ldots, x_{n-1})) \ ,$$

one can simply select $x_0, \ldots, x_{n-1}$ at random, define $x_{i+n} = f(x_i, \ldots, x_{i+n-1})$, and obtain the connection polynomial $p(x)$ from the sequence of least significant bits of $x_0, \ldots, x_{2b}$ using, e.g., Berlekamp-Massey. If $p(x)$ is a primitive polynomial of degree b, $p(x)$ is also the minimal polynomial of the associated matrix M.

This approach yields a number of simple and efficient masking functions. In particular, the 3-operation primitives $(x_0 \lll r_0) \oplus (x_i \gg r_1)$ and $(x_0 \lll r_0) \oplus (x_i \ll r_1)$ are found for several useful block and word sizes, as Table 1 illustrates. Some block sizes do not yield such small generators with any primitive we tried; in particular, 128-bit blocks require at least 4 operations, which is consistent—albeit somewhat better—with the results of [58, Appendix B]. Lemma 3 shows that this approach yields proper masking functions according to Definition 1.

Lemma 3. *Let M be an $b \times b$ matrix over $\mathbb{F}_2$ of the form shown in Eq. (5). Furthermore, let M's minimal polynomial be primitive and of degree b. Then given the function $\varphi_0^i(L) = M^i \cdot L$, any tweak set with $\mathcal{T}_i \subseteq \{0, \ldots, 2^b - 2\}$ is a b-proper tweak space by Definition 1.*

Proof. [20, Proposition 1] directly applies.

One may wonder whether there is any significant advantage of the above technique over, say, the Gray code sequence with the standard polynomial representation. We argue that our approach improves on it in several ways:

Simplicity OCB (especially OCB2) requires implementers to be aware of Galois field arithmetic. Our approach requires no user knowledge—even implicitly—of field or polynomial arithmetic, but only *unconditional* shifts and XOR operations. Even Sarkar's word-based LFSRs [74] do not hide the finite field structure from implementers, thus making it easier to make mistakes.

Table 1: Sample masking functions for various state sizes b and respective decompositions into n words of w bits

b	w	n	φ
128	8	16	$(x_1, \ldots, x_{15}, (x_0 \lll 1) \oplus (x_9 \gg 1) \oplus (x_{10} \ll 1))$
128	32	4	$(x_1, \ldots, x_3, (x_0 \lll 1) \oplus (x_1 \wedge 31) \oplus (x_2 \wedge 127))$
128	32	4	$(x_1, \ldots, x_3, (x_0 \lll 5) \oplus x_1 \oplus (x_1 \ll 13))$
128	64	2	$(x_1, (x_0 \lll 11) \oplus x_1 \oplus (x_1 \ll 13))$
256	32	8	$(x_1, \ldots, x_7, (x_0 \lll 17) \oplus x_5 \oplus (x_5 \gg 13))$
256	64	4	$(x_1, \ldots, x_3, (x_0 \lll 3) \oplus (x_3 \gg 5))$
512	32	16	$(x_1, \ldots, x_{15}, (x_0 \lll 5) \oplus (x_3 \gg 7))$
512	64	8	$(x_1, \ldots, x_7, (x_0 \lll 29) \oplus (x_1 \ll 9))$
800	32	25	$(x_1, \ldots, x_{15}, (x_0 \lll 25) \oplus x_{21} \oplus (x_{21} \gg 13))$
1024	8	128	$(x_1, \ldots, x_{127}, (x_0 \lll 1) \oplus x_{125} \oplus (x_{125} \gg 5))$
1024	64	16	$(x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$
1600	32	50	$(x_1, \ldots, x_{49}, (x_0 \lll 3) \oplus (x_{23} \gg 3))$
1600	64	25	$(x_1, \ldots, x_{24}, (x_0 \lll 55) \oplus x_{21} \oplus (x_{21} \ll 21))$
1600	64	25	$(x_1, \ldots, x_{24}, (x_0 \lll 15) \oplus x_{23} \oplus (x_{23} \ll 23))$

Constant-time Both OCB masking schemes require potentially variable-time operations to compute each mask—be it conditional XOR, number of trailing zeroes, or memory accesses indexed by $\mathsf{ntz}(i+1)$. This is easily avoidable by clever implementers, but it is also a pitfall avoidable by our design choice. Even in specifications aimed at developers [56], $\mathsf{double}(S)$ is defined as a variable-time operation.

Efficiency Word-based masking has the best space-time efficiency tradeoff of all considered masking schemes. It requires only minimal space usage—one block—while also involving a very small number of operations beyond the XOR with the block (as low as 3, cf. Table 1). It is also SIMD-friendly, allowing the generation of several consecutive masks with a single short SIMD instruction sequence.

In particular, for permutations that can take advantage of a CPU's vector units via "word-slicing"—which is the case for Salsa20, ChaCha, Threefish, and many other ARX designs—it is possible to compute a few consecutive masks at virtually the same cost as computing a single mask transition. It is also efficient, with our masking, to add the mask to the plaintext both in transposed order (word-sliced) and regular order.

For concreteness, consider the mask sequence $(x_1, \ldots, x_{15}, (x_0 \lll 5) \oplus (x_3 \gg 7))$ and a permutation using 512-bit blocks of 32-bit words. Suppose further that we are working with a CPU with 8-wide vectors, e.g., AVX2. Given 8 additional words of storage, it is possible to compute $L = (x_1, \ldots, x_{15}, (x_0 \lll 5) \oplus (x_3 \gg 7), \ldots, (x_7 \lll 5) \oplus (x_{10} \gg 7))$ entirely in parallel. Consider now the transposed set of 8 blocks $m_0, \ldots, m_7$; adding the mask consists of $m_0 \oplus L_{0\text{–}15}, m_1 \oplus L_{1\text{–}16}, \ldots$. On the other hand, when the blocks are word-sliced—with m'_0 being the first 32-bit word of m_i, m'_1 being the second, and so on—adding the mask is still efficient, as $m'_0 \oplus L_{0\text{–}7}, m'_1 \oplus L_{1\text{–}8}, \ldots$. This would be impossible with the standard masking schemes used in, e.g., OCB.

There is also an advantage at the low-end—φ can easily be implemented as a circular array, which implies that only an index and the logical operations must be executed for each mask update. This is superior to both the typical Gray code and powering-up approach, in that shifting by one requires moving *every word* of the mask, instead of only one of them. Additionally, storage is often a precious resource in low-end systems, and the Gray code method requires significantly more than one block to achieve its best performance.

3.5 Proposed Masking for $u = 2$ and $u = 3$

Modes often require the tweak space to have multiple dimensions. In particular, the modes of Sections 4 and 5 require the tweak space to have 2 and 3 "coordinates." To extend the masking function from Section 3.4 to a tweak space divided into disjunct sets, we have several options. We can simply split the range $[0, 2^b - 1]$ into equivalence classes, e.g., $i_0 = 4k + 0, i_1 = 4k + 1, \ldots$ for at most 4 different tweak indexes. Some constructions instead store a few fixed tweak values that are used later as "extra" finalization tweaks.

The approach we follow takes a cue from XEX [71]. Before introducing the scheme itself, we need a deeper understanding of the masking function φ introduced in Section 3.4. At its core, φ is a linear map representable by a matrix M with primitive minimal polynomial $p(x)$. In fact, φ can be interpreted as the matrix representation [61, §2.52] of $\mathbb{F}_{2^b}$, where M is, up to a change of basis, the companion matrix of $p(x)$. This property may be exploited to quickly jump ahead to an arbitrary state $\varphi^i(L)$: since $\varphi^i(L) = M^i \cdot L$ and additionally $p(M) = 0$, then $(x^i \bmod p(x))(M) = (x^i)(M) + (p(x)q(x))(M) = (x^i)(M) = M^i$. Therefore we can implement arbitrarily large "jumps" in the tweak space by evaluating the right polynomials over M. This property—like fast word-oriented shift registers—has had its first uses in the pseudorandom number generation literature [42].

Since we may control the polynomials here, we choose the very same polynomials as Rogaway for the best performance: $x + 1$, and $x^2 + x + 1$, denoted in [71] as **3** and **7**. Putting everything together, our masking for $u = 3$ becomes

$$\begin{aligned}\delta(K, X, i_0, i_1, i_2) &= ((x)(M))^{i_0}((x+1)(M))^{i_1}((x^2+x+1)(M))^{i_2} \cdot P(K \parallel X) \\ &= M^{i_0}(M+I)^{i_1}(M^2+M+I)^{i_2} \cdot P(K \parallel X)\ .\end{aligned}$$

To ensure that the tweak space is b-proper we need one extra detail: we need to ensure that the logarithms $\log_x(x+1)$ and $\log_x(x^2+x+1)$ are sufficiently apart. While for 128, Rogaway has already computed the corresponding discrete logarithms [71] using generic methods, larger blocks make it nontrivial to show b-properness. The following lemma shows that one particular function satisfies Definition 1. The lemma uses the discrete logarithms whose computation is described in Section 3.6.

Lemma 4. *Let* $\varphi(x) : \{0,1\}^{1024} \mapsto \{0,1\}^{1024}$ *be the linear map* $(x_0, \ldots, x_{15}) \mapsto (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$. *Further, let* M *be the* 1024×1024 *matrix*

associated with φ such that $\varphi(L) = M \cdot L$. Let $\Phi = \{\varphi_0^{i_0}, \varphi_1^{i_1}, \varphi_2^{i_2}\}$ be the set of functions used in the masking, with $\varphi_0^{i_0}(L) = M^{i_0} \cdot L$, $\varphi_1^{i_1}(L) = (M+I)^{i_1} \cdot L$, and $\varphi_2^{i_2}(L) = (M^2+M+I)^{i_2} \cdot L$. The tweak space

$$\mathcal{T} = \mathcal{T}_0 \times \mathcal{T}_1 \times \mathcal{T}_2 = \{0, 1, \ldots, 2^{1020}-1\} \times \{0,1,2,3\} \times \{0,1\}$$

is b-proper relative to the function set Φ.

Proof. The proof closely follows [71, Proposition 5]. Let $i_0 \in \mathcal{T}_0$, $i_1 \in \mathcal{T}_1$, and $i_2 \in \mathcal{T}_2$. We first show that $\varphi_0^{i_0} \circ \varphi_1^{i_1} \circ \varphi_2^{i_2}$ is unique for any distinct set of tweaks.

An easy computation shows that $p(x) = x^{1024} + x^{901} + x^{695} + x^{572} + x^{409} + x^{366} + x^{203} + x^{163} + 1$ is the minimal polynomial of M. This polynomial is both irreducible and primitive, which implies that the order of M is $2^{1024}-1$. We begin by determining the logarithms of $M+I$ and M^2+M+I relatively to M. This may be accomplished by computing $l_1 = \log_x(x+1)$ and $l_2 = \log_x(x^2+x+1)$ in the field $\mathbb{F}_2[x]/p(x)$, see Section 3.6.

The values l_1 and l_2 let us represent $M^{i_0}M^{i_1}M^{i_2}$ as $M^{i_0}M^{l_1 i_1}M^{l_2 i_2}$. Given a second distinct pair (i_0', i_1', i_2'), we have that $M^{i_0}M^{l_1 i_1}M^{l_2 i_2} = M^{i_0'}M^{l_1 i_1'}M^{l_2 i_2'}$ iff $i_0 + l_1 i_1 + l_2 i_2 = i_0' + l_1 i_1' + l_2 i_2' \pmod{2^{1024}-1}$. Equivalently, $i_0 - i_0' = (i_1 - i_1')l_1 + (i_2 - i_2')l_2 \pmod{2^{1024}-1}$. By a simple exhaustive search through the valid ranges of i_1 and i_2 we are able to see that the smallest absolute difference $(i_1 - i_1')l_1 + (i_2 - i_2')l_2$ occurs when $i_1 - i_1' = -1$ and $i_2 - i_2' = -1$, and is $\approx 2^{1020.58}$. Since $i_0 - i_0'$ is at most $\pm(2^{1020}-1)$, collisions cannot happen. Since each mask is unique, the fact that $\mathcal{T}$ is b-proper follows from Lemma 3. □

3.6 Computing Discrete Logarithms in $\mathbb{F}_{2^{512}}$ and $\mathbb{F}_{2^{1024}}$

While the classical incarnation of the Function Field Sieve (FFS) with $\mathbb{F}_2$ as the base field could no doubt solve logarithms in $\mathbb{F}_{2^{512}}$ with relatively modest computational resources—see for example [49,76]—the larger field would require a significant amount of work [7]. One could instead use subfields other than $\mathbb{F}_2$ and apply the medium-base-field method of Joux and Lercier [50], which would be relatively quick for $\mathbb{F}_{2^{512}}$, but not so easy for $\mathbb{F}_{2^{1024}}$.

However, with the advent of the more sophisticated modern incarnation of the FFS, development of which began in early 2013 [33,48,34,8,35,36,51,37], the target fields are now regarded as small, even tiny, at least relative to the largest such example computation where a DLP in $\mathbb{F}_{2^{9234}}$ was solved [38]. Since these developments have effectively rendered small characteristic DLPs useless for public key cryptography, (despite perhaps some potential doubters [19, Section 5]) it is edifying that there is a constructive application in cryptography[7] for what is generally regarded as a purely cryptanalytic pursuit.

Due to the many subfields present in the fields in question, there is a large parameter space to explore with regard to the application of the modern techniques,

[7] Beyond cryptography, examples abound in computational mathematics: in finite geometry; representation theory; matrix problems; group theory; and Lie algebras in the modular case; to name but a few.

and it becomes an interesting optimization exercise to find the most efficient approach. Moreover, such is the size of these fields that coding time rather than computing time is the dominant term in the overall cost. We therefore solved the relevant DLPs using MAGMA V2.19-1 [16], which allowed us to develop rapidly. All computations were executed on a standard desktop computer with a 2.0 GHz AMD Opteron processor.

3.6.1 Fields Setup. For reasons of both efficiency and convenience we use $\mathbb{F}_{2^{16}}$ as base field for both target fields, given by the following extensions:

$$\mathbb{F}_{2^4} = \mathbb{F}_2[U]/(U^4 + U + 1) = \mathbb{F}_2(u) \ ,$$
$$\mathbb{F}_{2^{16}} = \mathbb{F}_{2^4}[V]/(V^4 + V^3 + V + u) = \mathbb{F}_{2^4}(v) \ .$$

We represent $\mathbb{F}_{2^{512}}$ as $\mathbb{F}_{2^{16}}[X]/(I_{32}(X)) = \mathbb{F}_{2^{16}}(x)$, where I_{32} is the degree 32 irreducible factor of $H_{32}(X) = h_1(X^{16})X + h_0(X^{16})$, where $h_1 = (X + u^9 + u^5v + u^{13}v^2 + u^3v^3)^3$ and $h_0 = X^3 + u^2 + u^9v^2 + u^{13}v^3$. The other irreducible factors of $H_{32}(X)$ have degrees 6 and 11.

We represent $\mathbb{F}_{2^{1024}}$ as $\mathbb{F}_{2^{16}}[X]/(I_{64}(X)) = \mathbb{F}_{2^{16}}(x)$, where I_{64} is the degree 64 irreducible factor of $H_{64}(X) = h_1(X^{16})X + h_0(X^{16})$, where $h_1 = (X + u + u^7v + u^4v^2 + u^7c^3)^5$ and $h_0 = X^5 + u^9 + u^4v + u^6v^2 + v^3$. The other irreducible factors of $H_{64}(X)$ have degrees 7 and 10. Transforming from the original representations of Section 3.6.3 to these is a simple matter [60].

Note that ideally one would only have to use h_i's of degree 2 and 4 to obtain degree 32 and 64 irreducibles, respectively. However, no such h_i's exist and so we are forced to use h_i's of degree 3 and 5. The penalty for doing so incurs during the relation generation, see Section 3.6.2, and during the descent, in particular for degree 2 elimination, see Section 3.6.3.

Remark. The degrees of the irreducible cofactors of I_{32} in H_{32} and of I_{64} in H_{64} is an essential consideration in the set up of the two fields. In particular, if the degree d_f of a cofactor f has a non-trivial GCD with the degree of the main irreducible, then it should be considered as a 'trap' for the computation of the logarithms of the factor base elements, modulo all primes dividing $2^{16 \cdot \gcd(d_f, 32i)} - 1$ for $i = 1, 2$, for $\mathbb{F}_{2^{512}}$ and $\mathbb{F}_{2^{1024}}$, respectively [45,25]. This is because $\mathbb{F}_{2^{16}}[X]/(H_{32i}(X))$ will contain another copy of $\mathbb{F}_{2^{16 \cdot \gcd(d_f, 32i)}}$ which arises from f, and hence the solution space modulo primes dividing $2^{16 \cdot \gcd(d_f, 32i)} - 1$ has rank > 1. Our choice of h_0 and h_1 in each case limits the effect of this problem to prime factors of $2^{32} - 1$, namely subgroups of tiny order within which we solve the DLPs using a linear search. The irreducible cofactors are also traps for the descent phase [35], but are easily avoided.

3.6.2 Relation Generation and Logarithms of Linear Elements. The factor base is defined to be $\mathcal{F} = \{x + d \mid d \in \mathbb{F}_{2^{16}}\}$. To generate relations over $\mathcal{F}$, we use the technique from [33], described most simply in [35]. In particular, for both target fields let $y = x^{16}$; by the definitions of I_{32} and I_{64} it follows in both

cases that $x = h_0(y)/h_1(y)$. Using these field isomorphisms, for any $a, b, c \in \mathbb{F}_{2^{16}}$ we have the field equality

$$x^{17} + ax^{16} + bx + c = \frac{1}{h_1(y)} \left(yh_0(y) + ayh_1(y) + bh_0(y) + ch_1(y) \right). \quad (6)$$

One can easily generate (a, b, c) triples such that the left hand side of Eq. (6) always splits completely over $\mathcal{F}$. Indeed, one first computes the set $\mathcal{B}$ of 16 values $B \in \mathbb{F}_{2^{16}}$ such that the polynomial $f_B(X) = X^{17} + BX + B$ splits completely over $\mathbb{F}_{2^{16}}$ [15]. Assuming $c \neq ab$ and $b \neq a^{16}$, the left hand side of Eq. (6) can be transformed (up to a scalar factor) into f_B, where $B = \frac{(b+a^{16})^{17}}{(c+ab)^{16}}$. Hence if this B is in $\mathcal{B}$ then the left hand side also splits. In order to generate relations, one repeatedly chooses random $B \in \mathcal{B}$ and random $a, b \neq a^{16} \in \mathbb{F}_{2^{16}}$, computes $c = ((b + a^{16})^{17})^{1/16} + ab$, and tests whether the right hand side of Eq. (6) also splits over $\mathbb{F}_{2^{16}}$. If it does then one has a relation, since $(y + d) = (x + d^{1/16})^{16}$, and each h_1 is a power of a factor base element.

The probability that the right hand side of Eq. (6) splits completely is heuristically 1/4! and 1/6! for $\mathbb{F}_{2^{512}}$ and $\mathbb{F}_{2^{1024}}$ respectively. In both cases we obtain $2^{16} + 200$ relations, which took about 0.3 hrs and 8.8 hrs, respectively. To compute the logarithms of the factor base elements, we used MAGMA's `ModularSolution` function, with its `Lanczos` option set, modulo the 9th to 13th largest prime factors of $2^{512} - 1$ for the smaller field and modulo the 10th to 16th largest prime factors of $2^{1024} - 1$ for the larger field. These took about 13.5 hrs and 24.5 hrs, respectively.

3.6.3 Individual Logarithms. The original representations of the target fields are:

$$\begin{aligned} \mathbb{F}_{2^{512}} &= \mathbb{F}_2[T]/(T^{512} + T^{335} + T^{201} + T^{67} + 1) = \mathbb{F}_2(t) \,, \\ \mathbb{F}_{2^{1024}} &= \mathbb{F}_2[T]/(T^{1024} + T^{901} + T^{695} + T^{572} + T^{409} + T^{366} + T^{203} + T^{163} + 1) \\ &= \mathbb{F}_2(t) \,. \end{aligned}$$

In order to solve the two relevant DLPs in each original field, we need to compute three logarithms in each of our preferred field representations, namely the logarithms of the images of t, $t + 1$ and $t^2 + t + 1$—which we denote by t_0, t_1 and t_2—relative to some generator. We use the generator x in both cases.

For $\mathbb{F}_{2^{512}}$, we multiply the targets t_i by random powers of x and apply a continued fraction initial split so that $x^k t_i \equiv n/d \pmod{I_{32}}$, with n of degree 16 and d of degree 15, until both n and d are 4-smooth. One then just needs to eliminate irreducible elements of degree 2, 3, 4 into elements of smaller degree. For degree 4 elements, we apply the building block for the quasi-polynomial algorithm due to Granger, Kleinjung, and Zumbrägel [36,37], which is just degree 2 elimination but over a degree 2 extended base field. This results in each degree 4 element being expressed as a product of powers of at most 19 degree 2 elements, and possibly some linear elements. For degree 3 elimination we use Joux's bilinear quadratic system approach [48], which expresses each degree 3 element as a

product of powers of again at most 19 degree 2 elements and at least one linear element. For degree 2 elimination, we use the on-the-fly technique from [33], but with the quadratic system approach from [34], which works for an expected proportion $1-(1-1/2!)^{16} = 255/256$ of degree 2's, since the cofactor in each case has degree 2. On average each descent takes about 10 s, and if it fails due to a degree 2 being ineliminable, we simply rerun it with a different random seed. Computing logarithms modulo the remaining primes only takes a few seconds with a linear search, which completes the following results:

$$\begin{aligned}\log_t(t+1) = {} & 501632302866570670563660970955028961903690197966887 3\\ & 487264378851651440588241161115592058268630926672385 4\\ & 512235779287054265328022610551493984901818209298 02 ,\\ \log_t(t^2+t+1) = {} & 778979505459703512296093350265308220986572478078438 1\\ & 216662651301933387803414250047794195008130367563340 1\\ & 118596646581200776656548532019025482993657737894 62 .\end{aligned}$$

The total computation time for these logarithms is less than 14 hrs.

For $\mathbb{F}_{2^{1024}}$, we use the same continued fraction initial split, but now with n and d of degree 32 and 31, until each is 4-smooth, but also allowing a number of degree 8 elements. Finding such an expression takes on average 7 hrs, which, while not optimal, means that the classical special-Q elimination method could be obviated, i.e., not coded. For degree 8 elimination, we again use the building block for the quasi-polynomial algorithm of Granger et al., which expresses such a degree 8 element as a product of powers of at most 21 degree 4 elements, and possibly some degree 2 and 1 elements. Degree 4 and 3 elimination proceed as before, but with a larger cofactor of the element to be eliminated on the r.h.s. due to the larger degrees of h_0 and h_1. Degree 2 elimination is significantly harder in this case, since the larger degrees of the h_i's mean that the elimination probability for a random degree 2 element was only $1-(1-1/4!)^{16} \approx 0.494$. However, using the recursive method from the DLP computation in $\mathbb{F}_{2^{4404}}$ [35] allows this to be performed with near certainty. If any of the eliminations fails, then as before we simply rerun the eliminations with a different random seed. In total, after the initial rewrite of the target elements into a product of degree 1, 2, 3, 4, and 8 elements, each descent takes just under an hour. Again, computing logarithms modulo the remaining primes takes less than a minute with a linear search, completing the following results:

$$\begin{aligned}\log_t(t+1) = {} & 356031381070238016894189506806176884676852879916524\\ & 279675345656550984270765575541375310062097902188572 0\\ & 196678535148030769731170945683137201859849917444119 6\\ & 147033260221616158337836258365757075663102493592798 4\\ & 249827223869952857623068524280576393895115544816738 7\\ & 149681903876406067502645471152193264955124750148 ,\\ \log_t(t^2+t+1) = {} & 161005643918902879345214446131555844702011737643264 2\end{aligned}$$

$$
\begin{aligned}
&5524859486238161374654279717800300706136749607630601\\
&4967362673777547140089938700144112424081388711871290\\
&7973319251629628361398267351880948069161459793052257\\
&1907117948291164323355528169854354396482029507781947\\
&2534171313076937775797909159788879361876099888834\ .
\end{aligned}
$$

The total computation time for these logarithms is about 57 hrs.

Note that it is possible to avoid the computations in $\mathbb{F}_{2^{512}}$ altogether by embedding the relevant DLPs into $\mathbb{F}_{2^{1024}}$. However, the descent time would take longer than the total time, at least with the non-optimal descent that we used. We considered the possibility of using "jokers" [35], which permit one to halve the degree of even degree irreducibles when they are elements of a subfield of index 2. However, it seems to only be possible when one uses compositums, which is not possible in the context of the fields $\mathbb{F}_{2^{2^n}}$. In any case, such optimizations are academic when the total computation time is as modest as those recorded here, and our approach has the bonus of demonstrating the easiness of computing logarithms in $\mathbb{F}_{2^{512}}$, as well as in $\mathbb{F}_{2^{1024}}$.

With regard to larger n, it would certainly be possible to extend the approach of Kleinjung [55] to solve logarithms in the fields $\mathbb{F}_{2^{2^n}}$ for $n = 11, 12$ and 13, should this be needed for applications, without too much additional computational effort.

4 Offset Public Permutation Mode (OPP)

We present the *Offset Public Permutation Mode* (OPP), a nonce-respecting authenticated encryption mode with support for associated data which uses the techniques presented in Section 3. It can be seen as a generalization of OCB3 [58] to arbitrary block sizes using permutations and using improved masking techniques from Section 3.

4.1 Specification of OPP

OPP uses MEM of Section 3.1 for $u = 3$ and $\Phi = \{\alpha, \beta, \gamma\}$ with $\alpha(x) = \varphi(x)$, $\beta(x) = \varphi(x) \oplus x$ and $\gamma(x) = \varphi(x)^2 \oplus \varphi(x) \oplus x$, employing φ as introduced in Section 3.4. Furthermore, the general masking function is specified as

$$\delta : (K, X, i_0, i_1, i_2) \mapsto \gamma^{i_2} \circ \beta^{i_1} \circ \alpha^{i_0}(P(X \parallel K)) \ .$$

We require that the tweak space of MEM used in OPP is b-proper with respect to Φ as introduced in Definition 1 and proven in Lemma 4.

The formal specification of OPP is given in Fig. 1 and an overview of the scheme is depicted in Fig. 2. We refer to the authentication part of OPP as OPPAbs and to the encryption part as OPPEnc. The OPPAbs mode requires only the encryption function $\widetilde{E}$, while the OPPEnc mode uses both $\widetilde{E}$ and $\widetilde{D}$ of MEM.

Let H_i and M_j denote b-bit header and message blocks with $0 \le i \le h-1$ and $0 \le j \le m-1$ where $h = |H|_b$ and $m = |M|_b$. Note that the size of the last blocks H_{h-1} and M_{m-1} is potentially smaller than b bits. To realize proper domain separation between full and partial data blocks, and different data types, OPP uses the following setup:

OPPAbs			OPPEnc		
data block	condition	(i_0, i_1, i_2)	data block	condition	(i_0, i_1, i_2)
H_i	$0 \le i < h-1$	$(i, 0, 0)$	M_j	$0 \le j < m-1$	$(j, 0, 1)$
H_{h-1}	$\lvert H\rvert \bmod b = 0$	$(h-1, 0, 0)$	M_{m-1}	$\lvert M\rvert \bmod b = 0$	$(m-1, 0, 1)$
H_{h-1}	$\lvert H\rvert \bmod b \neq 0$	$(h-1, 1, 0)$	M_{m-1}	$\lvert M\rvert \bmod b \neq 0$	$(m-1, 1, 1)$
$\bigoplus_{j=0}^{m-1} M_j$	$\lvert M\rvert \bmod b = 0$	$(h-1, 2, 0)$			
$\bigoplus_{j=0}^{m-1} M_j$	$\lvert M\rvert \bmod b \neq 0$	$(h-1, 3, 0)$			

Algorithm: $\mathsf{OPPEnc}(K, X, M)$
1. $M_0 \parallel \cdots \parallel M_{m-1} \leftarrow M$, s.t. $|M_i| = b, 0 \le |M_{m-1}| < b$
2. $C \leftarrow \varepsilon$
3. $S \leftarrow 0^b$
4. **for** $i \in \{0, \ldots, m-2\}$ **do**
5. $\quad C_i \leftarrow \widetilde{E}^{i,0,1}_{K,X}(M_i)$
6. $\quad C \leftarrow C \parallel C_i$
7. $\quad S \leftarrow S \oplus M_i$
8. **end**
9. **if** $|M_{m-1}| > 0$ **then**
10. $\quad Z \leftarrow \widetilde{E}^{m-1,1,1}_{K,X}(0)$
11. $\quad C_{m-1} \leftarrow \mathsf{left}_{|M_{m-1}|}(\mathsf{pad}^0_b(M_{m-1}) \oplus Z)$
12. $\quad C \leftarrow C \parallel C_{m-1}$
13. $\quad S \leftarrow S \oplus \mathsf{pad}^{10}_b(M_{m-1})$
14. **end**
15. **return** C, S

Algorithm: $\mathsf{OPPDec}(K, X, C)$
1. $C_0 \parallel \cdots \parallel C_{m-1} \leftarrow C$, s.t. $|C_i| = b, 0 \le |C_{m-1}| < b$
2. $M \leftarrow \varepsilon$
3. $S \leftarrow 0^b$
4. **for** $i \in \{0, \ldots, m-2\}$ **do**
5. $\quad M_i \leftarrow \widetilde{D}^{i,0,1}_{K,X}(C_i)$
6. $\quad M \leftarrow M \parallel M_i$
7. $\quad S \leftarrow S \oplus M_i$
8. **end**
9. **if** $|C_{m-1}| > 0$ **then**
10. $\quad Z \leftarrow \widetilde{E}^{m-1,1,1}_{K,X}(0)$
11. $\quad M_{m-1} \leftarrow \mathsf{left}_{|C_{m-1}|}(\mathsf{pad}^0_b(C_{m-1}) \oplus Z)$
12. $\quad M \leftarrow M \parallel M_{m-1}$
13. $\quad S \leftarrow S \oplus \mathsf{pad}^{10}_b(M_{m-1})$
14. **end**
15. **return** M, S

Algorithm: $\mathsf{OPPAbs}(K, X, H, S, l)$
1. $H_0 \parallel \cdots \parallel H_{h-1} \leftarrow H$, s.t. $|H_i| = b, 0 \le |H_{h-1}| < b$
2. $S' \leftarrow 0^b$
3. **for** $i \in \{0, \ldots, h-2\}$ **do**
4. $\quad S' \leftarrow S' \oplus \widetilde{E}^{i,0,0}_{K,X}(H_i)$
5. **end**
6. **if** $|H_{h-1}| > 0$ **then**
7. $\quad S' \leftarrow S' \oplus \widetilde{E}^{h-1,1,0}_{K,X}(\mathsf{pad}^{10}_b(H_{h-1}))$
8. **end**
9. $j \leftarrow \lceil l/b \rceil + 2$
10. **return** $\mathsf{left}_\tau(S' \oplus \widetilde{E}^{h-1,j,0}_{K,X}(S))$

Algorithm: $\mathsf{OPP}\mathcal{E}(K, N, H, M)$
1. $X \leftarrow \mathsf{pad}^0_{b-n-k}(N)$
2. $C, S \leftarrow \mathsf{OPPEnc}(K, X, M)$
3. $T \leftarrow \mathsf{OPPAbs}(K, X, H, S, |M| \bmod b)$
4. **return** C, T

Algorithm: $\mathsf{OPP}\mathcal{D}(K, N, H, C, T)$
1. $X \leftarrow \mathsf{pad}^0_{b-n-k}(N)$
2. $M, S \leftarrow \mathsf{OPPDec}(K, X, C)$
3. $T' \leftarrow \mathsf{OPPAbs}(K, X, H, S, |M| \bmod b)$
4. **if** $T = T'$ **then return** M **else return** $\perp$ **end**

Fig. 1: Offset Public Permutation Mode (OPP)

4.2 Security of OPP

Theorem 5. *Let* $P \xleftarrow{\$} \mathsf{Perm}(b)$. *Then, in the* nonce-respecting *setting,*

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{OPP},P}(q_\mathcal{E}, q_\mathcal{D}, \sigma, p) \le \frac{4.5\sigma^2}{2^b} + \frac{3\sigma p}{2^b} + \frac{p}{2^k} + \frac{2^{n-\tau}}{2^n - 1} \ .$$

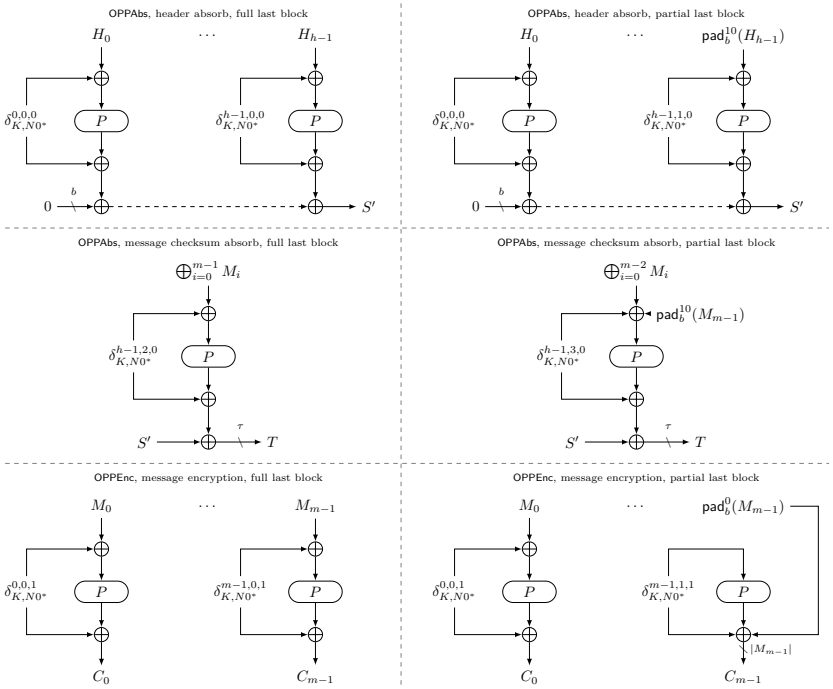

Fig. 2: Offset Public Permutation Mode (OPP)

The proof is given in Supporting Material B. Note that OPP shares its structure with OCB3 of Krovetz and Rogaway [58]. In more detail, we will show that once MEM gets replaced by a random tweakable permutation $\widetilde{\pi}$, OPP becomes exactly the ΘCB3 construction [58]. The proof then follows by combining the security of MEM and the security of ΘCB3.

5 Misuse-Resistant Offset Mode (MRO)

We present the *Misuse-Resistant Offset Mode* (MRO), a MAC-then-Encrypt AE mode with support for associated data which fully tolerates nonce re-usage. In some sense, MRO is the misuse-resistant variant of OPP and also uses the techniques presented in Section 3. It can be seen as a permutation-based variation of PMAC [14] followed by a permutation-based variation of CTR mode, and shares ideas with the Synthetic Counter in Tweak (SCT) as introduced in Deoxys v1.3 and Joltik v1.3 [46,47], though MRO is permutation-based and employs the improved masking schedule of Section 3.

5.1 Specification of MRO

The formal specification of MRO is given in Fig. 3 and an overview of the scheme is depicted in Fig. 4. Similar to OPP, we refer to the authentication part of MRO

as MROAbs and to the encryption part as MROEnc. In contrast to OPP, MRO only requires the encryption function $\widetilde{E}$ of MEM. Using notation as in the OPP mode, MRO uses the following setup for masking:

MROAbs			MROEnc		
data block	condition	(i_0, i_1, i_2)	data block	condition	(i_0, i_1, i_2)
H_i	$0 \le i \le h-1$	$(i, 0, 0)$	M_j	$0 \le j \le m-1$	$(0, 0, 1)$
M_j	$0 \le j \le m-1$	$(j, 1, 0)$			
$\lvert H\rvert \,\|\, \lvert M\rvert$	n.a.	$(0, 2, 0)$			

Algorithm: $\mathsf{Absorb}(K, X, S, A, j)$
1. **if** $|A| > 0$ **then**
2. $\quad A_0 \,\|\, \cdots \,\|\, A_{a-1} \leftarrow \mathsf{pad}_b^0(A)$
3. $\quad$ **for** $i \in \{0, \ldots, a-1\}$ **do**
4. $\quad\quad S \leftarrow S \oplus \widetilde{E}_{K,X}^{i,j,0}(A_i)$
5. $\quad$ **end**
6. **end**
7. **return** S

Algorithm: $\mathsf{MROAbs}(K, X, H, M)$
1. $S \leftarrow 0^b$
2. $S \leftarrow \mathsf{Absorb}(K, X, S, H, 0)$
3. $S \leftarrow \mathsf{Absorb}(K, X, S, M, 1)$
4. $S \leftarrow \widetilde{E}_{K,X}^{0,2,0}(S \oplus |H| \,\|\, |M|)$
5. **return** $\mathsf{left}_\tau(S)$

Algorithm: $\mathsf{MROEnc}(K, X, T, M)$
1. $C \leftarrow \varepsilon$
2. **if** $|M| > 0$ **then**
3. $\quad M_0 \,\|\, \cdots \,\|\, M_{m-1} \leftarrow \mathsf{pad}_b^0(M)$
4. $\quad$ **for** $i \in \{0, \ldots, m-1\}$ **do**
5. $\quad\quad C_i \leftarrow M_i \oplus \widetilde{E}_{K,X}^{0,0,1}(T \,\|\, i)$
6. $\quad\quad C \leftarrow C \,\|\, C_i$
7. $\quad$ **end**
8. **end**
9. **return** $\mathsf{left}_{|M|}(C)$

Algorithm: $\mathsf{MRODec}(K, X, T, C)$
1. $M \leftarrow \varepsilon$
2. **if** $|C| > 0$ **then**
3. $\quad C_0 \,\|\, \cdots \,\|\, C_{m-1} \leftarrow \mathsf{pad}_b^0(C)$
4. $\quad$ **for** $i \in \{0, \ldots, m-1\}$ **do**
5. $\quad\quad M_i \leftarrow C_i \oplus \widetilde{E}_{K,X}^{0,0,1}(T \,\|\, i)$
6. $\quad\quad M \leftarrow M \,\|\, M_i$
7. $\quad$ **end**
8. **end**
9. **return** $\mathsf{left}_{|C|}(M)$

Algorithm: $\mathsf{MRO}\mathcal{E}(K, N, H, M)$
1. $X \leftarrow \mathsf{pad}_{b-n-k}^0(N)$
2. $T \leftarrow \mathsf{MROAbs}(K, X, H, M)$
3. $C \leftarrow \mathsf{MROEnc}(K, X, T, M)$
4. **return** C, T

Algorithm: $\mathsf{MRO}\mathcal{D}(K, N, H, C, T)$
1. $X \leftarrow \mathsf{pad}_{b-n-k}^0(N)$
2. $M \leftarrow \mathsf{MRODec}(K, X, T, C)$
3. $T' \leftarrow \mathsf{MROAbs}(K, X, H, M)$
4. **if** $T = T'$ **then return** M **else return** $\perp$ **end**

Fig. 3: Misuse-Resistant Offset Mode (MRO)

5.2 Security of MRO

Theorem 6. *Let* $P \xleftarrow{\$} \mathsf{Perm}(b)$. *Then, in the* nonce-reuse *setting,*

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MRO},P}(q_\mathcal{E}, q_\mathcal{D}, \sigma, p) \le \frac{6.5\sigma^2}{2^b} + \frac{3\sigma p}{2^b} + \frac{p}{2^k} + \frac{q_\mathcal{E}^2/2 + q_\mathcal{D}}{2^\tau} \ .$$

The proof is given in Supporting Material C. The proof is in fact a standard-model proof where the scheme is considered to be based on MEM. It is a modular proof that, at a high level, consists of the following steps:

(i) The first step in the analysis is to replace MEM with a random secret tweakable permutation, at the cost of the MTPRP security of MEM.

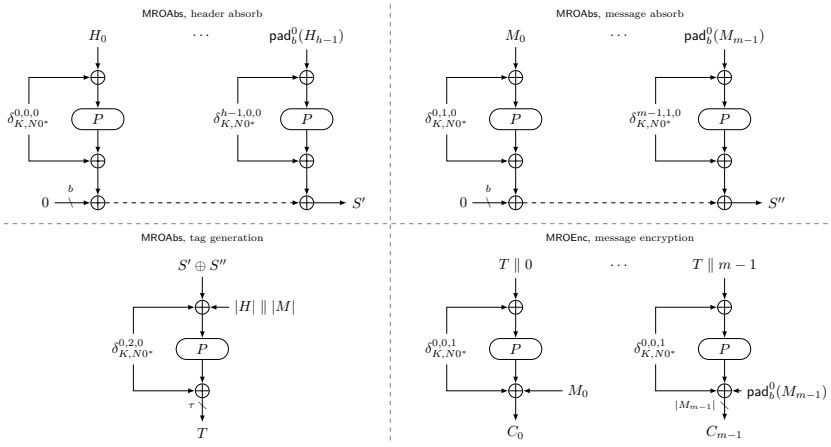

Fig. 4: Misuse-Resistant Offset Mode (MRO)

(ii) The absorption function and encryption function call the tweakable cipher for *distinct* tweaks. Hence, using an adaption of the MAC-then-Encrypt paradigm to misuse resistance [68,40] allows us to analyze the MAC parts and the encryption parts separately.

6 Misuse-Resistant Sponge (MRS)

We introduce the *Misuse-Resistant Sponge Mode* (MRS), a MAC-then-Encrypt Sponge-based AE mode with support for associated data which fully tolerates nonce re-usage. The absorption function is a full-state keyed Sponge MAC [12,4,66]. The encryption function follows the SpongeWrap approach [11,66].

6.1 Specification of MRS

The formal specification of MRS is given in Fig. 5 and an overview of the scheme is depicted in Fig. 6. It consists of an absorption function MRSAbs and an encryption function MRSEnc, in a MAC-then-Encrypt mode, but using the same primitive and same key in both functions. We remark that MRS as given in Fig. 5 only does one round of squeezing in order to obtain the tag. This can be easily generalized to multiple rounds, without affecting the security proofs.

We briefly discuss the differences of MRS with Haddoc, the misuse-resistant AE scheme presented by Bertoni et al. [13] at the 2014 SHA-3 workshop. Haddoc follows the MAC-then-Encrypt paradigm as well, where the MAC function is identical to MRSAbs. For encryption, however, Haddoc uses the Sponge in CTR mode. At a high level, and in our terminology, this boils down to

$$C_i \leftarrow M_i \oplus \mathsf{left}_r(P(T \parallel \langle i \rangle \parallel 1 \parallel K)) \ ,$$

for $0 \leq i \leq m-1$. In other words, MRS and Haddoc structurally differ in the way encryption is performed, and in fact, Haddoc more closely matches the ideas of the MRSO hybrid of Section 7.

Algorithm: $\mathsf{Absorb}(S, A)$

1. **if** $|A| > 0$ **then**
2. $\quad A_0 \parallel \cdots \parallel A_{a-1} \leftarrow \mathsf{pad}^0_b(A)$
3. $\quad$ **for** $i \in \{0, \ldots, a-1\}$ **do**
4. $\quad\quad S \leftarrow P(S)$
5. $\quad\quad S \leftarrow S \oplus A_i$
6. $\quad$ **end**
7. **end**
8. **return** S

Algorithm: $\mathsf{MRSEnc}(K, T, M)$

1. $C \leftarrow \varepsilon$
2. **if** $|M| > 0$ **then**
3. $\quad S \leftarrow T \parallel 0^* \parallel 1 \parallel K$
4. $\quad M_0 \parallel \cdots \parallel M_{m-1} \leftarrow \mathsf{pad}^0_r(M)$
5. $\quad$ **for** $i \in \{0, \ldots, m-1\}$ **do**
6. $\quad\quad S \leftarrow P(S)$
7. $\quad\quad S \leftarrow S \oplus (M_i \parallel 0^c)$
8. $\quad\quad C \leftarrow C \parallel \mathsf{left}_r(S)$
9. $\quad$ **end**
10. **end**
11. **return** $\mathsf{left}_{|M|}(C)$

Algorithm: $\mathsf{MRS}\mathcal{E}(K, N, H, M)$

1. $T \leftarrow \mathsf{MRSAbs}(K, N, H, M)$
2. $C \leftarrow \mathsf{MRSEnc}(K, T, M)$
3. **return** C, T

Algorithm: $\mathsf{MRSAbs}(K, N, H, M)$

1. $S \leftarrow N \parallel 0^* \parallel 0 \parallel K$
2. $S \leftarrow \mathsf{Absorb}(S, H)$
3. $S \leftarrow \mathsf{Absorb}(S, M)$
4. $S \leftarrow P(S)$
5. $S \leftarrow S \oplus |H| \parallel |M|$
6. $S \leftarrow P(S)$
7. $T \leftarrow \mathsf{left}_\tau(S)$
8. **return** T

Algorithm: $\mathsf{MRSDec}(K, T, C)$

1. $M \leftarrow \varepsilon$
2. **if** $|C| > 0$ **then**
3. $\quad S \leftarrow T \parallel 0^* \parallel 1 \parallel K$
4. $\quad C_0 \parallel \cdots \parallel C_{m-1} \leftarrow \mathsf{pad}^0_r(C)$
5. $\quad$ **for** $i \in \{0, \ldots, m-1\}$ **do**
6. $\quad\quad S \leftarrow P(S)$
7. $\quad\quad M \leftarrow M \parallel \mathsf{left}_r(S \oplus (C_i \parallel 0^c))$
8. $\quad\quad S \leftarrow C_i \parallel \mathsf{right}_c(S)$
9. $\quad$ **end**
10. **end**
11. **return** $\mathsf{left}_{|C|}(M)$

Algorithm: $\mathsf{MRS}\mathcal{D}(K, N, H, C, T)$

1. $M \leftarrow \mathsf{MRSDec}(K, T, C)$
2. $T' \leftarrow \mathsf{MRSAbs}(K, N, H, M)$
3. **if** $T = T'$ **then return** M **else return** $\perp$ **end**

Fig. 5: Misuse-Resistant Sponge (MRS)

6.2 Security of MRS

Theorem 7. *Let* $P \xleftarrow{\$} \mathsf{Perm}(b)$. *Then, in the* nonce-reuse *setting,*

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MRS},P}(q_\mathcal{E}, q_\mathcal{D}, \sigma, p) \leq \frac{4\sigma^2}{2^b} + \frac{4\sigma^2}{2^c} + \frac{2\sigma p}{2^k} + \frac{q_\mathcal{E}^2/2 + q_\mathcal{D} q_\mathcal{E} + q_\mathcal{D}}{2^\tau} \ .$$

The proof is given in Supporting Material D. It is different from the proofs for OPP and MRO, although it is also effectively a standard-model proof. It relies on the observation that both the absorption and the encryption phase are in fact evaluations of the Full-state Keyed Duplex [11,66]. This construction has been proven to behave like a random functionality, with the property that it always outputs uniformly random data, up to common prefix in the input. Assuming that the distinguisher never makes duplicate queries, MRSAbs never has common prefixes; assuming tags never collide, MRSEnc never has common prefixes; and finally, the initial inputs to MRSAbs versus MRSEnc are always different due to the 0/1 domain separation. The proof then easily follows.

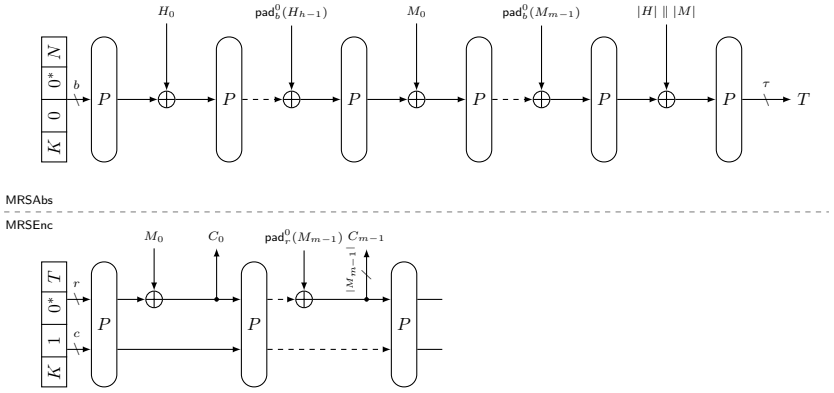

Fig. 6: Misuse-Resistant Sponge (MRS)

7 Misuse-Resistant Sponge-Offset (MRSO)

The constructions of Sections 5 and 6 can be combined in a straightforward way to obtain two hybrid constructions: the *Misuse-Resistant Sponge-then-Offset Mode* (MRSO) and the *Misuse-Resistant Offset-then-Sponge Mode* (MROS). While we cannot think of any practical use-case for MROS, we do think the MRSO hybrid is useful. As suggested in Section 6, MRSO is comparable with—and in fact improves over—Haddoc.

7.1 Specification of MRSO

The formal specification of the MRSO AE scheme is formalized in Fig. 7. It MACs the data using MRSAbs and encrypts using MROEnc. MRSO uses MEM as specified for OPP but requires only a very limited selection of tweaks and has $i_1 = i_2 = 0$ fixed. Thus, the general masking function can be simplified to

$$\delta : (K, X, i_0) \mapsto \alpha^{i_0}(P(X \parallel K)) \ .$$

For the encryption part MROEnc this is clear (cf. Section 5). For the absorption part MRSAbs, this is less clear: informally, it is based on the idea of setting $L = P(N \parallel 0^* \parallel K)$, and of XORing this value everywhere in-between two consecutive evaluations of P. Because at the end of MRSAbs, a part of the rate is extracted, this "trick" only works if performed with the rightmost $b - \tau$ bits of L. Therefore, MRSO is based on a slight adjustment of MEM with $b - \tau$-bit maskings only. The details follow in the security proof (Supporting Material E). Let $h = |H|_b$ and $m = |M|_b$ denote the number of b-bit header and message blocks, respectively. We use the following setup for masking:

MRSAbs			MROEnc		
data block	condition	i_0	data block	condition	i_0
H_i	$0 \le i \le h-1$	0	M_j	$0 \le j \le m-1$	1
M_j	$0 \le j \le m-1$	0			
$\lvert H\rvert \parallel \lvert M\rvert$	n.a.	0			

Algorithm: $\mathsf{MRSO}\mathcal{E}(K, N, H, M)$

1. $X \leftarrow \mathsf{pad}^0_{b-n-k}(N)$
2. $T \leftarrow \mathsf{MRSAbs}(K, N, H, M)$
3. $C \leftarrow \mathsf{MROEnc}(K, X, T, M)$
4. **return** C, T

Algorithm: $\mathsf{MRSO}\mathcal{D}(K, N, H, C, T)$

1. $X \leftarrow \mathsf{pad}^0_{b-n-k}(N)$
2. $M \leftarrow \mathsf{MRODec}(K, X, T, C)$
3. $T' \leftarrow \mathsf{MRSAbs}(K, N, H, M)$
4. **if** $T = T'$ **then return** M **else return** $\perp$ **end**

Fig. 7: Hybrid Sponge-Offset mode MRSO. Refer to Figs. 3 and 5 for the sub-algorithms

7.2 Security of MRSO

Theorem 8. *Let* $P \xleftarrow{\$} \mathsf{Perm}(b)$. *Then, in the* nonce-reuse *setting,*

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MRSO},P}(q_{\mathcal{E}}, q_{\mathcal{D}}, \sigma, p) \leq \frac{2\sigma^2}{2^b} + \frac{5.5\sigma^2}{2^{b-\tau}} + \frac{3\sigma p}{2^{b-\tau}} + \frac{p}{2^k} + \frac{q_{\mathcal{E}}^2/2 + q_{\mathcal{D}}}{2^\tau} \ .$$

The proof is similar to the proof of MRO, and is given in Supporting Material E.

8 Implementation

In this section we discuss our results on the implementations of concrete instantiations of OPP, MRO, and MRS. For all three schemes we use state, key, tag, and nonce sizes of $b = 1024$, $k = \tau = 256$, and $n = 128$ bits. For P we employ the BLAKE2b [6] permutation with $l \in \{4, 6\}$ rounds. For OPP and MRO we use

$$\varphi(x_0, \ldots, x_{15}) = (x_1, ..., x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

and for MRSEnc we set rate and capacity to $r = 768$ and $c = 256$ bits.

To remain self-contained, we now define the BLAKE2b permutation. The permutation operates on a state $S = (s_0, \ldots, s_{15})$ with 64-bit words s_i. A single round $F(S)$ consists of the sequence of operations

$$\begin{array}{llll} G(s_0, s_4, s_8, s_{12}); & G(s_1, s_5, s_9, s_{13}); & G(s_2, s_6, s_{10}, s_{14}); & G(s_3, s_7, s_{11}, s_{15}); \\ G(s_0, s_5, s_{10}, s_{15}); & G(s_1, s_6, s_{11}, s_{12}); & G(s_2, s_7, s_8, s_{13}); & G(s_3, s_4, s_9, s_{14}); \end{array}$$

where

$$G(a, b, c, d) = \begin{cases} a = a + b; \ d = (d \oplus a) \ggg 32; \ c = c + d; \ b = (b \oplus c) \ggg 24; \\ a = a + b; \ d = (d \oplus a) \ggg 16; \ c = c + d; \ b = (b \oplus c) \ggg 63; \end{cases}$$

BLAKE2 and its predecessors have been heavily analyzed, e.g., [54,41]. These results are mostly of theoretical interest though since the complexity of the attacks vastly outweigh our targeted security level. Nevertheless, the BLAKE2 permutation family has some evident and well-known non-random characteristics [5]: for any $l > 0$, it holds that $F^l(0) = 0$ and

$$F^l(a, a, a, a, b, b, b, b, c, c, c, c, d, d, d, d) = (w, w, w, w, x, x, x, x, y, y, y, y, z, z, z, z)$$

Table 2: Performance of the OPP, MRO, and MRS schemes instantiated with the BLAKE2b permutation

		$l = 4$			$l = 6$		
Platform	Impl.	OPP	MRO	MRS	OPP	MRO	MRS
Cortex-A8	NEON	4.26	8.07	8.50	5.91	11.32	12.21
Sandy Bridge	AVX	1.24	2.41	2.55	1.91	3.58	3.87
Haswell	AVX2	0.55	1.06	2.40	0.75	1.39	3.58

for arbitrary values a, b, c, and d. With some care in the design phase, these symmetric states can be easily avoided though, so that an adversary can not exploit them as a distinguisher. Thus, we use slightly modified variants of the schemes introduced in Sections 4 to 7. Instead of initializing the masks with $P(N \parallel 0^{640} \parallel K)$ in OPP and MRO, we encode the round number l and tag size τ as 64-bit strings and use $P(N \parallel 0^{512} \parallel \langle l \rangle_{64} \parallel \langle \tau \rangle_{64} \parallel K)$. Analogously, the Sponge states in MRSAbs and MRSEnc are initialized with $N \parallel 0^{448} \parallel \langle l \rangle_{64} \parallel \langle \tau \rangle_{64} \parallel \langle 0 \rangle_{64} \parallel K$ and $T \parallel 0^{320} \parallel \langle l \rangle_{64} \parallel \langle \tau \rangle_{64} \parallel \langle 1 \rangle_{64} \parallel K$, respectively.

We wrote reference implementations of all schemes in plain C and optimized variants using the AVX, AVX2, and NEON instruction sets.[8] Performance was measured on the Intel Sandy Bridge and Haswell microarchitectures, and on the ARM Cortex-A8. Tables 2 and 3 list timings of our optimized implementations and of some reference AEAD schemes, respectively. All values are given for "long messages" (≥ 4 KiB) with cycles per byte (cpb) as unit.

In the nonce-respecting scenario our fastest proposal is OPP with 4 BLAKE2b rounds. Our 4-fold word-sliced AVX2-implementation achieves 0.55 cpb on Haswell, which amounts to a throughput of about 6.36 GiBps assuming a processor frequency of 3.5 GHz. Compared to its competitors AES-GCM, OCB3, ChaCha20-Poly1305 and Deoxys$^{\neq}$ (v1.3), this instantiation of OPP is faster by factors of about 1.87, 1.25, 3.80, and 3.07 respectively. Even the more conservative 6-round variant of our scheme is able to maintain high speeds at 0.75 cpb (4.67 GiBps) reducing the distance to the above competitors to factors of 1.37, 0.92, 2.78, and 2.25. On ARM platforms, without AES-NI instructions, the performance advantage of OPP is even more significant. There our NEON-implementation outperforms the AES-based ciphers OCB3 and AES-GCM by factors of about 6.78 and 9.06. The highly hand-optimized Salsa20-Poly1305 implementation of [10] is somewhat able to keep pace and is only slower by a factor of around 1.92.

In the misuse-resistant scenario our fastest proposal is MRO with 4 BLAKE2b rounds. Our 4-fold word-sliced AVX2-implementation achieves 1.06 cpb on Haswell which is equivalent to a throughput of 3.30 GiBps at a frequency of 3.5 GHz. In comparison to schemes such as AES-GCM-SIV and Deoxys$^{=}$ (v.1.3), the above instantiation of MRO is faster by factors of about 1.10 and 3.09. For the 6-round version with 1.39 cpb these factors are reduced to 0.79 and 2.35, respectively. Unfortunately, there is not enough published data on performance of misuse-resistant AE schemes on ARM. As for OPP in the nonce-respecting scenario, one

[8] The source code of our schemes is freely available at [64] under a CC0 license.

Table 3: Performance of some reference AEAD modes

	nonce-respecting					misuse-resistant	
Platform	AES-GCM	OCB3	Chacha20-Poly1305	Salsa20-Poly1305	Deoxys$^{\neq}$	GCM-SIV	Deoxys$^{=}$
Cortex-A8	38.6	28.9	-	5.60+2.60	-	-	-
Sandy Bridge	2.55	0.98	-	-	1.32	-	2.59
Haswell	1.03	0.69	1.43+0.66	-	1.69	1.17	3.28
References	[18,39]	[58,39]	[31,32]	[10]	[46]	[40]	[46]

can expect similar performance gaps between the misuse-resistant AES-based schemes and MRO. When comparing MRO to Salsa20-Poly1305, which assumes a nonce-respecting adversary, the two schemes are basically on par.

Due to the inherently sequential Sponge-construction used in MRS, advanced implementation techniques like 4-fold word-slicing are not possible. It is therefore no big surprise that MRS performs worse than MRO. Nevertheless, on Haswell MRS achieves 2.40 cpb ($l = 4$) and 3.58 cpb ($l = 6$) which translate to throughputs of 1.45 GiBps and 0.97 GiBps, respectively. Thus, MRS is still competitive to other misuse-resistant AE schemes on Intel platforms. But also on ARM, it demonstrates good performance almost on the level of MRO. Currently, we have not written any implementations for MRSO but it is to be expected that its performance lies somewhere between MRO and MRS.

Acknowledgements

We kindly thank Thorsten Kleinjung for the observations on irreducible cofactors presented in the remark at the end of Section 3.6.1 and for further helpful discussions during our work.

References

1. Agrawal, M., Chang, D., Sanadhya, S.: sp-AELM: Sponge based authenticated encryption scheme for memory constrained devices. In: Foo, E., Stebila, D. (eds.) ACISP 15. LNCS, vol. 9144, pp. 451–468. Springer (2015)
2. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated permutation-based encryption for lightweight cryptography. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 168–186. Springer (2015)
3. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and authenticated online ciphers. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 424–443. Springer (2013)
4. Andreeva, E., Daemen, J., Mennink, B., Van Assche, G.: Security of keyed sponge constructions using a modular proof approach. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 364–384. Springer (2015)
5. Aumasson, J.P., Jovanovic, P., Neves, S.: Analysis of NORX: Investigating differential and rotational properties. In: Aranha, D.F., Menezes, A. (eds.) LATINCRYPT 2014. LNCS, vol. 8895, pp. 306–324. Springer (2015)

6. Aumasson, J.P., Neves, S., Wilcox-O'Hearn, Z., Winnerlein, C.: BLAKE2: Simpler, smaller, fast as MD5. In: Jacobson Jr., M.J., Locasto, M.E., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 13. LNCS, vol. 7954, pp. 119–135. Springer (2013)
7. Barbulescu, R., Bouvier, C., Detrey, J., Gaudry, P., Jeljeli, H., Thomé, E., Videau, M., Zimmermann, P.: Discrete logarithm in GF(2^{809}) with FFS. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 221–238. Springer (2014)
8. Barbulescu, R., Gaudry, P., Joux, A., Thomé, E.: A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 1–16. Springer (2014)
9. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS. pp. 394–403. IEEE Computer Society Press (1997)
10. Bernstein, D.J., Schwabe, P.: NEON crypto. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 320–339. Springer (2012)
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the Sponge: Single-pass authenticated encryption and other applications. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer (2011)
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Security of the Keyed Sponge Construction. In: SKEW 2011 (2011)
13. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: Using Keccak technology for AE: Ketje, Keyak and more. In: SHA-3 2014 Workshop (2014)
14. Black, J., Rogaway, P.: A block-cipher mode of operation for parallelizable message authentication. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 384–397. Springer (2002)
15. Bluher, A.W.: On $x^{q+1}+ax+b$. Finite Fields and Their Applications 10(3), 285–305 (2004)
16. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbolic Comput. 24(3-4), 235–265 (1997), computational algebra and number theory (London, 1993)
17. CAESAR — Competition for Authenticated Encryption: Security, Applicability, and Robustness (2014)
18. Câmara, D.F., Gouvêa, C.P.L., López, J., Dahab, R.: Fast software polynomial multiplication on ARM processors using the NEON engine. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E.R., Xu, L., Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E.R., Xu, L. (eds.) CD-ARES 2013 Workshops: MoCrySEn and SeCIHD. LNCS, vol. 8128, pp. 137–154. Springer, Heidelberg (2013)
19. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: How to compress homomorphic ciphertexts. Cryptology ePrint Archive, Report 2015/113 (2015)
20. 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)
21. Chakraborty, D., Sarkar, P.: On modes of operations of a block cipher for authentication and authenticated encryption. Cryptology ePrint Archive, Report 2014/627 (2014)
22. Chang, D.H., Dworkin, M.J., Hong, S., Kelsey, J.M., Nandi, M.: A keyed sponge construction with pseudorandomness in the standard model. In: The Third SHA-3 Candidate Conference (2012)

23. Chen, S., Lampe, R., Lee, J., Seurin, Y., Steinberger, J.P.: Minimizing the two-round Even-Mansour cipher. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 39–56. Springer (2014)
24. Chen, S., Steinberger, J.P.: Tight security bounds for key-alternating ciphers. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 327–350. Springer (2014)
25. Cheng, Q., Wan, D., Zhuang, J.: Traps to the BGJT-algorithm for discrete logarithms. LMS Journal of Computation and Mathematics 17, 218–229 (2014)
26. Cogliati, B., Lampe, R., Seurin, Y.: Tweaking Even-Mansour ciphers. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 189–208. Springer (2015)
27. Cogliati, B., Seurin, Y.: Beyond-birthday-bound security for tweakable Even-Mansour ciphers with linear tweak and key mixing. In: Cheon, J.H., Iwata, T. (eds.) ASIACRYPT 2015. LNCS, Springer (2015), to appear
28. Cogliati, B., Seurin, Y.: On the provable security of the iterated Even-Mansour cipher against related-key and chosen-key attacks. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 584–613. Springer (2015)
29. 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)
30. Gligor, V.D., Donescu, P.: Fast encryption and authentication: XCBC encryption and XECB authentication modes. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 92–108. Springer (2002)
31. Goll, M., Gueron, S.: Vectorization on ChaCha stream cipher. In: Latifi, S. (ed.) ITNG 2014. pp. 612–615. IEEE Computer Society (2014)
32. Goll, M., Gueron, S.: Vectorization of Poly1305 message authentication code. In: Latifi, S. (ed.) ITNG 2015. pp. 612–615. IEEE Computer Society (2015)
33. Göloglu, F., Granger, R., McGuire, G., Zumbrägel, J.: On the function field sieve and the impact of higher splitting probabilities — application to discrete logarithms in $\mathbb{F}_{2^{1971}}$ and $\mathbb{F}_{2^{3164}}$. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 109–128. Springer (2013)
34. Göloglu, F., Granger, R., McGuire, G., Zumbrägel, J.: Solving a 6120-bit DLP on a desktop computer. In: Lange, T., Lauter, K., Lisonek, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 136–152. Springer (2014)
35. Granger, R., Kleinjung, T., Zumbrägel, J.: Breaking '128-bit secure' supersingular binary curves — (or how to solve discrete logarithms in $F_{2^{4\cdot 1223}}$ and $F_{2^{12\cdot 367}}$). In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 126–145. Springer (2014)
36. Granger, R., Kleinjung, T., Zumbrägel, J.: On the powers of 2. Cryptology ePrint Archive, Report 2014/300 (2014)
37. Granger, R., Kleinjung, T., Zumbrägel, J.: On the discrete logarithm problem in finite fields of fixed characteristic. Cryptology ePrint Archive, Report 2015/685 (2015)
38. Granger, R., Kleinjung, T., Zumbrägel, J.: Discrete Logarithms in $GF(2^{9234})$. NMBRTHRY list (31/1/2014)
39. Gueron, S.: AES-GCM software performance on the current high end CPUs as a performance baseline for CAESAR competition. In: DIAC 2013 (2013)
40. Gueron, S., Lindell, Y.: GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte. Cryptology ePrint Archive, Report 2015/102 (2015)

41. Guo, J., Karpman, P., Nikolic, I., Wang, L., Wu, S.: Analysis of BLAKE2. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 402–423. Springer (2014)
42. Haramoto, H., Matsumoto, M., Nishimura, T., Panneton, F., L'Ecuyer, P.: Efficient jump ahead for $\mathbb{F}_2$-linear random number generators. INFORMS Journal on Computing 20(3), 385–390 (2008)
43. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 15–44. Springer (2015)
44. Hoang, V.T., Reyhanitabar, R., Rogaway, P., Vizár, D.: Online authenticated-encryption and its nonce-reuse misuse-resistance. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 493–517. Springer (2015)
45. Huang, M., Narayanan, A.K.: On the relation generation method of Joux for computing discrete logarithms. CoRR abs/1312.1674 (2013)
46. Jean, J., Nikolić, I., Peyrin, T.: Deoxys v1.3. CAESAR Round 2 submission (2015)
47. Jean, J., Nikolić, I., Peyrin, T.: Joltik v1.3. CAESAR Round 2 submission (2015)
48. Joux, A.: A new index calculus algorithm with complexity $L(1/4 + o(1))$ in small characteristic. In: Lange, T., Lauter, K., Lisonek, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 355–379. Springer (2014)
49. Joux, A., Lercier, R.: The function field sieve is quite special. In: Fieker, C., Kohel, D.R. (eds.) ANTS-V. LNCS, vol. 2369, pp. 431–445. Springer (2002)
50. Joux, A., Lercier, R.: The function field sieve in the medium prime case. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 254–270. Springer (2006)
51. Joux, A., Pierrot, C.: Improving the polynomial time precomputation of Frobenius representation discrete logarithm algorithms — simplified setting for small characteristic finite fields. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 378–397. Springer (2014)
52. Jutla, C.S.: Encryption modes with almost free message integrity. Journal of Cryptology 21(4), 547–578 (Oct 2008)
53. Kavun, E.B., Lauridsen, M.M., Leander, G., Rechberger, C., Schwabe, P., Yalçın, T.: Prøst v1. CAESAR Round 1 submission (2014)
54. Khovratovich, D., Nikolic, I., Pieprzyk, J., Sokolowski, P., Steinfeld, R.: Rotational cryptanalysis of ARX revisited. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 519–536. Springer (2015)
55. Kleinjung, T.: Discrete logarithms in GF(2^{1279}). NMBRTHRY list (17/10/2014)
56. Krovetz, T., Rogaway, P.: The OCB authenticated-encryption algorithm. RFC 7253 (Informational) (2014)
57. Krovetz, T.: HS1-SIV v1. CAESAR Round 1 submission (2014)
58. 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)
59. Kurosawa, K.: Power of a public random permutation and its application to authenticated encryption. IEEE Transactions on Information Theory 56(10), 5366–5374 (2010)
60. Lenstra, Jr., H.W.: Finding isomorphisms between finite fields. Mathematics of Computation 56(193), 329–347 (1991)
61. Lidl, R., Niederreiter, H.: Finite fields, Encyclopedia of Mathematics and its Applications, vol. 20. Cambridge University Press, Cambridge, United Kingdom, 2nd edn. (1997)
62. Marsaglia, G.: Xorshift RNGs. Journal of Statistical Software 8(14) (2003)

63. Matsumoto, M., Nishimura, T.: Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation 8(1), 3–30 (1998)
64. MEM Family of AEAD Schemes (2015), https://github.com/MEM-AEAD
65. Mennink, B.: XPX: Generalized Tweakable Even-Mansour with Improved Security Guarantees. Cryptology ePrint Archive, Report 2015/476 (2015)
66. Mennink, B., Reyhanitabar, R., Vizár, D.: Security of full-state keyed Sponge and Duplex: Applications to authenticated encryption. In: Cheon, J.H., Iwata, T. (eds.) ASIACRYPT 2015. LNCS, Springer (2015), to appear
67. Minematsu, K.: A short universal hash function from bit rotation, and applications to blockcipher modes. In: Susilo, W., Reyhanitabar, R. (eds.) ProvSec 2013. LNCS, vol. 8209, pp. 221–238. Springer (2013)
68. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering generic composition. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 257–274. Springer (2014)
69. Niederreiter, H.: Factorization of polynomials and some linear-algebra problems over finite fields. Linear Algebra and its Applications 192, 301–328 (1993)
70. Patarin, J.: The "coefficients H" technique (invited talk). In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 328–345. Springer (2009)
71. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer (2004)
72. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A block-cipher mode of operation for efficient authenticated encryption. In: ACM CCS 01. pp. 196–205. ACM Press (2001)
73. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer (2006)
74. Sarkar, P.: Pseudo-random functions and parallelizable modes of operations of a block cipher. IEEE Transactions on Information Theory 56(8), 4025–4037 (2010)
75. Sasaki, Y., Todo, Y., Aoki, K., Naito, Y., Sugawara, T., Murakami, Y., Matsui, M., Hirose, S.: Minalpher v1. CAESAR Round 1 submission (2014)
76. Thomé, E.: Computation of discrete logarithms in $F_{2^{607}}$. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 107–124. Springer (2001)
77. Yasuda, K.: A one-pass mode of operation for deterministic message authentication-security beyond the birthday barrier. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 316–333. Springer (2008)
78. Zeng, G., Han, W., He, K.: High efficiency feedback shift register: $\sigma-$LFSR. Cryptology ePrint Archive, Report 2007/114 (2007)

Supporting Material

A Proof of Theorem 2 (Security of MEM)

Let $K \xleftarrow{\$} \{0,1\}^k$, $P \xleftarrow{\$} \mathsf{Perm}(b)$, and $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$. Consider a deterministic MTPRP distinguisher $\mathbf{D}$ for the tweakable blockcipher $\widetilde{E}$ of Section 3.1. By the security definition of Section 2.2, in the real world it has access to $\mathcal{O} = (\widetilde{E}^{\pm}, P^{\pm})$, and in the ideal world to $\mathcal{P} = (\widetilde{\pi}^{\pm}, P^{\pm})$. It makes q construction queries, and p primitive queries to $P^{\pm}$.

By hypothesis, the tweak space $\mathcal{T}$ is ϵ-proper relative to the set of functions Φ. The distinguisher is expected to obey the masking partition $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1$ in such a way that tweaks from $\mathcal{T}_0$ are only used *in forward direction*. We have $(0, \ldots, 0) \notin \mathcal{T}_{1\bar{i}}$ by assumption, meaning that this masking can be used in forward direction only.

Views. The information that is gathered by $\mathbf{D}$ is summarized in a view ν. This view will consist of a summary of all information $\mathbf{D}$ learns during the interaction with its oracles, as well as some additional information that will be revealed after the proof.

The construction queries are summarized in a directionless view

$$\nu_1 = \{(X_1, \bar{i}_1, M_1, C_1), \ldots, (X_q, \bar{i}_q, M_q, C_q)\} \ ,$$

meaning that the jth construction query was made on input of tweak $(X_j, \bar{i}_j)$ and message M_j, and responded with cipher C_j, or vice versa. Note that we do not make a distinction between queries with tweak from $\mathcal{T}_0$ and $\mathcal{T}_1$; the analysis is identical, with the only exception being $\bar{i}_j = (0, \ldots, 0)$, in which case we have to keep in mind that the query is made in forward direction. Let s denote the number of distinct values X_j, and let $\{Y_1, \ldots, Y_s\}$ be the minimal set such that it includes $X_1, \ldots, X_q$.

The primitive queries are summarized in a view

$$\nu_2 = \{(x_1, y_1), \ldots, (x_p, y_p)\} \ ,$$

which means that the jth primitive query was either a forward query on input of x_j (with response y_j) or vice versa.

As the distinguisher is deterministic, there is a one-to-one mapping between these directionless views (ν_1, ν_2) and the interaction of $\mathbf{D}$ with its oracles. Therefore, (ν_1, ν_2) properly summarizes the conversation.

To simplify our security analysis, we will reveal the keying information to $\mathbf{D}$ at the end of the interaction. This "trick" was employed in earlier H-coefficient technique based security proofs of Even-Mansour constructions (see, e.g., [26,23,24]), but because for $\widetilde{E}$ the key is first transformed to obtain the masks, the situation is slightly more technical, and we follow ideas from [65]. In more detail, we reveal

$$\nu_K = \{(K_{Y_1}, K^{\star}_{Y_1}), \ldots, (K_{Y_s}, K^{\star}_{Y_s})\} \ .$$

In the real world, we have $K_{Y_j} = Y_j \parallel K$ and $K^\star_{Y_j} = P(Y_j \parallel K)$, where $K \xleftarrow{\$} \{0,1\}^k$ is the key used for the entire game. In the ideal world, we set $K \xleftarrow{\$} \{0,1\}^k$, $K_{Y_j} = Y_j \parallel K$, and $K^\star_{Y_j} \xleftarrow{\$} \{0,1\}^b$ for $j = 1, \ldots, s$.

Note that the disclosure of ν_K is without loss of generality: it will only *increase* the success probability of $\mathbf{D}$. The total view is denoted

$$\nu = (\nu_1, \nu_2, \nu_K) \ .$$

We assume that $\mathbf{D}$ never makes duplicate queries, and hence ν_1 contains no duplicate elements, and the same for ν_2.

Attainable Views and View-Compliant Oracles. Denote by $X_{\mathcal{O}}$ the probability distribution of views when interacting with $\mathcal{O}$, and similarly $X_{\mathcal{P}}$ the probability distribution of views when interacting with $\mathcal{P}$. A view ν is called "attainable" if it may appear during an interaction with $\mathcal{P}$, or formally if $\mathbf{Pr}\left(X_{\mathcal{P}} = \nu\right) > 0$. Throughout, we will only focus on attainable views. Concretely, for ν_1 this implies that for any $j \neq j'$ such that $(X_j, \bar{i}_j) = (X_{j'}, \bar{i}_{j'})$, we have $M_j \neq M_{j'}$ and $C_j \neq C_{j'}$. For ν_2, attainability implies that for all $j \neq j'$, we have $x_j \neq x_{j'}$ and $y_j \neq y_{j'}$.

We say that a tweakable permutation $\widetilde{\pi} \in \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$ is *compliant* with construction view $\nu_!$, denoted $\widetilde{\pi} \vdash \nu_1$, if $\widetilde{\pi}(X, \bar{i}, M) = C$ for each $(X, \bar{i}, M, C) \in \nu_1$. Similarly, if $\nu_P = \{(x_1, y_1), \ldots, (x_p, y_p)\}$ is a primitive view (for instance, $\nu_P = \nu_2$ or $\nu_P = \nu_K$), we say that a permutation $P \in \mathsf{Perm}(b)$ is *compliant* with ν_P, denoted $P \vdash \nu_P$, if $P(x) = y$ for each $(x, y) \in \nu_P$.

Patarin's H-Coefficient Technique. Our proof is based on Patarin's H-coefficient technique [70]. We will follow the formalism of Chen and Steinberger [24].

Lemma 9 (Patarin's Technique). *Let $\mathbf{D}$ be a deterministic distinguisher. Denote by $\mathcal{V}$ the set of attainable views, and consider a partition $\mathcal{V} = \mathcal{V}_{\text{good}} \cup \mathcal{V}_{\text{bad}}$ of $\mathcal{V}$ into good and bad views. Let $0 \leq \varepsilon$ be such that for all $\nu \in \mathcal{V}_{\text{good}}$,*

$$\frac{\mathbf{Pr}\left(X_{\mathcal{O}} = \nu\right)}{\mathbf{Pr}\left(X_{\mathcal{P}} = \nu\right)} \geq 1 - \varepsilon \ . \tag{7}$$

Then, the distinguishing advantage satisfies $\Delta_{\mathbf{D}}(\mathcal{O}; \mathcal{P}) \leq \varepsilon + \mathbf{Pr}\left(X_{\mathcal{P}} \in \mathcal{V}_{\text{bad}}\right)$.

A proof of this lemma can be found, among others, in [24,23]. At a high level, the idea of the technique is to identify views for which the fraction of (7) is large, and to isolate them as being "bad" views. Then, for all "good" views the distributions $X_{\mathcal{O}}$ and $X_{\mathcal{P}}$ are relatively close to each other and ε will be small. Note that if the definition of bad views is too loose, $\mathbf{Pr}\left(X_{\mathcal{P}} \in \mathcal{V}_{\text{bad}}\right)$ will be large, while if it is too tight, ε will be larger.

Definition of Bad Views. Note that in the real world, all tuples in ν define an input/output-tuple for P: for ν_2 and ν_K this is clear, for ν_1 this follows from the definition of $\widetilde{E}$. Provided no collision within these tuples occurs, ν defines exactly $q+p+s$ tuples for P. Inspired by this, we will call a transcript *bad* if any collision occurs. Formally, ν is called *bad* if one of the following conditions holds:

$$
\begin{aligned}
\mathrm{bad}_{1,1}: &\text{ for distinct some } (X,\bar{i},M,C),(X',\bar{i}',M',C') \in \nu_1\text{:}\\
&\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}(K_X^\star) \oplus \varphi_{u-1}^{i'_{u-1}} \circ \cdots \circ \varphi_0^{i'_0}(K_{X'}^\star) \in \{M \oplus M', C \oplus C'\}\ ,\\
\mathrm{bad}_{1,2}: &\text{ for some } (X,\bar{i},M,C) \in \nu_1 \text{ and } (x,y) \in \nu_2\text{:}\\
&\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}(K_X^\star) \in \{M \oplus x, C \oplus y\}\ ,\\
\mathrm{bad}_{1,K}: &\text{ for some } (X,\bar{i},M,C) \in \nu_1 \text{ and } (K_Y,K_Y^\star) \in \nu_K\text{:}\\
&\varphi_{u-1}^{i_{u-1}} \circ \cdots \circ \varphi_0^{i_0}(K_X^\star) \in \{M \oplus K_Y, C \oplus K_Y^\star\}\ ,\\
\mathrm{bad}_{2,K}: &\text{ for some } (x,y) \in \nu_2 \text{ and } (K_Y,K_Y^\star) \in \nu_K\text{:}\\
&x = K_Y \text{ or } y = K_Y^\star\ ,\\
\mathrm{bad}_{K,K}: &\text{ for distinct some } (K_Y,K_Y^\star),(K_{Y'},K_{Y'}^\star) \in \nu_K\text{:}\\
&K_Y^\star = K_{Y'}^\star\ .
\end{aligned}
$$

We write $\mathrm{bad} = \mathrm{bad}_{1,1} \vee \mathrm{bad}_{1,2} \vee \mathrm{bad}_{1,K} \vee \mathrm{bad}_{2,K} \vee \mathrm{bad}_{K,K}$. Above bad events are structured as follows: $\mathrm{bad}_{1,1}$ covers both possible collisions (input- and output-) of tuples within ν_1; $\mathrm{bad}_{1,2}$ and $\mathrm{bad}_{1,K}$ cover collisions between ν_1 on the one hand and ν_2 resp. ν_K on the other hand. Similarly for $\mathrm{bad}_{2,K}$ and $\mathrm{bad}_{K,K}$. Note that, by attainability of views, there are no collisions within ν_2, hence there is no such bad event as $\mathrm{bad}_{2,2}$. Also, note that for $\mathrm{bad}_{K,K}$ the input values in $\mathrm{bad}_{K,K}$ never collide, as they are of the form $Y \parallel K$ and $Y' \parallel K$ for $Y \neq Y'$.

In the remainder of the proof, we will derive an upper bound on the probability that a *bad* view appears in the ideal world, $\mathbf{Pr}\left(X_{\mathcal{P}} \in \mathcal{V}_{\mathrm{bad}}\right)$, and we will show that for any *good* transcript we have $\mathbf{Pr}\left(X_{\mathcal{O}} = \nu\right) \geq \mathbf{Pr}\left(X_{\mathcal{P}} = \nu\right)$. This will immediately complete the proof via Lemma 9.

Probability of Bad View in Ideal World. Our goal is to bound $\mathbf{Pr}\left(X_{\mathcal{P}} \in \mathcal{V}_{\mathrm{bad}}\right)$. Let ν be any view in the ideal world $\mathcal{P} = (\widetilde{\pi}^{\pm}, P^{\pm})$, and denote by $\mathbf{Pr}\left(\mathrm{bad}\right)$ the probability that this view satisfies bad. By basic probability theory,

$$
\begin{aligned}
\mathbf{Pr}\left(\mathrm{bad}\right) &= \mathbf{Pr}\left(\mathrm{bad}_{1,1} \vee \mathrm{bad}_{1,2} \vee \mathrm{bad}_{1,K} \vee \mathrm{bad}_{2,K} \vee \mathrm{bad}_{K,K}\right)\\
&\leq \mathbf{Pr}\left(\mathrm{bad}_{1,1}\right) + \mathbf{Pr}\left(\mathrm{bad}_{1,2}\right) + \mathbf{Pr}\left(\mathrm{bad}_{1,K}\right) + \mathbf{Pr}\left(\mathrm{bad}_{2,K}\right) + \mathbf{Pr}\left(\mathrm{bad}_{K,K}\right)\ .
\end{aligned} \tag{8}
$$

Recall that by definition of the ideal world, we have $K \xleftarrow{\$} \{0,1\}^k$, $K_{Y_j} = Y_j \parallel K$, and $K_{Y_j}^\star \xleftarrow{\$} \{0,1\}^b$ for $j = 1,\ldots,s$. For queries with $\bar{i}_j = (0,\ldots,0)$, we will additionally use that this query is necessarily made in forward direction, which implies that the value C_j is randomly drawn from a set of size at least $2^b - q$.

Regarding $\mathrm{bad}_{1,1}$, let $(X, \bar{i}, M, C), (X', \bar{i}', M', C') \in \nu_1$ be any two distinct queries. We make a distinction depending on the values $(X, \bar{i})$ and $(X', \bar{i}')$:

- If $(X, \bar{i}) = (X', \bar{i}')$, we necessarily have $M \neq M'$ and $C \neq C'$ by attainability of transcripts and $\mathrm{bad}_{1,1}$ is set by these two queries with probability 0;
- If $X \neq X'$, then $K^\star_X$ and $K^\star_{X'}$ are two independently generated random values and we argue based on the former. The condition of $\mathrm{bad}_{1,1}$ reads:

$$\varphi^{i_{u-1}}_{u-1} \circ \cdots \circ \varphi^{i_0}_0(K^\star_X) \in \{M \oplus M' \oplus \varphi^{i'_{u-1}}_{u-1} \circ \cdots \circ \varphi^{i'_0}_0(K^\star_{X'}), \\ C \oplus C' \oplus \varphi^{i'_{u-1}}_{u-1} \circ \cdots \circ \varphi^{i'_0}_0(K^\star_{X'})\} \ .$$

  As $K^\star_X \xleftarrow{\$} \{0,1\}^b$, by Definition 1 part 1 this happens with probability at most $2/2^\epsilon$;
- If $X = X'$ but $\bar{i} \neq \bar{i}'$, then $K^\star_X = K^\star_{X'} \xleftarrow{\$} \{0,1\}^b$ and Definition 1 part 2 directly shows that the condition is satisfied with probability at most $2/2^\epsilon$.

Summing over all possible queries, we obtain $\mathbf{Pr}\left(\mathrm{bad}_{1,1}\right) \leq \frac{2\binom{q}{2}}{2^\epsilon} \leq \frac{q^2}{2^\epsilon}$.

Regarding $\mathrm{bad}_{1,2}$, let $(X, \bar{i}, M, C) \in \nu_1$ and $(x, y) \in \nu_2$ by any two queries. Let $(K_X, K^\star_X) \in \nu_K$ be the unique tuple in ν_K corresponding to the tweak X in the tuple from ν_1. As $K^\star_X \xleftarrow{\$} \{0,1\}^b$, by Definition 1 part 1 the two queries satisfy the condition of $\mathrm{bad}_{1,2}$ with probability at most $2/2^\epsilon$. Summing over all possible queries, we obtain $\mathbf{Pr}\left(\mathrm{bad}_{1,2}\right) \leq \frac{2qp}{2^\epsilon}$.

Regarding $\mathrm{bad}_{1,K}$, we divide it into two subcases:

$$\begin{aligned} \mathrm{bad}_{1,K}: \ & \text{for some } (X, \bar{i}, M, C) \in \nu_1 \text{ and } (K_Y, K^\star_Y) \in \nu_K\text{:} \\ & \text{(a)} \ \ \varphi^{i_{u-1}}_{u-1} \circ \cdots \circ \varphi^{i_0}_0(K^\star_X) = M \oplus K_Y \ , \text{ or} \\ & \text{(b)} \ \ \varphi^{i_{u-1}}_{u-1} \circ \cdots \circ \varphi^{i_0}_0(K^\star_X) = C \oplus K^\star_Y \ . \end{aligned}$$

Let $(X, \bar{i}, M, C) \in \nu_1$ and $(K_Y, K^\star_Y) \in \nu_K$ be any two queries. Let $(K_X, K^\star_X) \in \nu_K$ be the unique tuple in ν_K corresponding to the tweak X in the tuple from ν_1.

For $\mathrm{bad}_{1,K}$(a), because $K^\star_X \xleftarrow{\$} \{0,1\}^b$ is generated independently from $K_Y = Y \parallel K$, by Definition 1 part 1 the condition is satisfied with probability $1/2^\epsilon$. Summing over all possible queries, we obtain $\mathbf{Pr}\left(\mathrm{bad}_{1,K}(\mathrm{a})\right) \leq \frac{qs}{2^\epsilon}$.

For $\mathrm{bad}_{1,K}$(b), a technicality is involved if $X = Y$, and above reasoning does not directly carry over. Instead, we make a case distinction:

- If $X \neq Y$, then $K^\star_X$ and $K^\star_Y$ are two independently generated random values and we argue based on the former. As $K^\star_X \xleftarrow{\$} \{0,1\}^b$, by Definition 1 part 1 the event happens with probability $1/2^\epsilon$;
- If $X = Y$ but $\bar{i} \neq (0, \ldots, 0)$, then $K^\star_X = K^\star_Y \xleftarrow{\$} \{0,1\}^b$ and by Definition 1 part 2 the event happens with probability $1/2^\epsilon$;

– If $X = Y$ and $\bar{i} = (0, \ldots, 0)$, then the condition reads $C = 0$. Because masking $\bar{i} = (0, \ldots, 0)$ can only appear in forward queries, the C-values is uniformly randomly drawn from a set of size at least $2^b - q$. The condition is satisfied with probability at most $1/(2^b - q)$.

Summing over all possible queries, we obtain $\mathbf{Pr}\left(\mathrm{bad}_{1,K}(\mathrm{b})\right) \leq \frac{qs}{\min\{2^\epsilon, 2^b - q\}}$. Concluding the case of $\mathrm{bad}_{1,K}$, we obtain $\mathbf{Pr}\left(\mathrm{bad}_{1,K}\right) \leq \frac{qs}{2^\epsilon} + \frac{qs}{\min\{2^\epsilon, 2^b - q\}}$.

Regarding $\mathrm{bad}_{2,K}$, we divide it into two subcases:

$$\begin{aligned} \mathrm{bad}_{2,K}: \ & \text{for some } (x, y) \in \nu_2 \text{ and } (K_Y, K_Y^\star) \in \nu_K\text{:} \\ & \text{(a)} \ \ x = K_Y \ , \text{ or} \\ & \text{(b)} \ \ y = K_Y^\star \ . \end{aligned}$$

For $\mathrm{bad}_{2,K}(\mathrm{a})$, recall that the input values in ν_K are of the form $K_Y = Y \parallel K$. Therefore, the condition is satisfied only if there is a tuple $(x, y) \in \nu_2$ such that $\mathsf{right}_k(x) = K$. Because $K \xleftarrow{\$} \{0,1\}^k$ this happens with probability at most $\mathbf{Pr}\left(\mathrm{bad}_{2,K}(\mathrm{a})\right) \leq \frac{p}{2^k}$.

For $\mathrm{bad}_{2,K}(\mathrm{b})$, let $(x, y) \in \nu_2$ and $(K_Y, K_Y^\star) \in \nu_K$ be any two queries. Because $K_Y^\star \xleftarrow{\$} \{0,1\}^b$, the two queries satisfy $\mathrm{bad}_{2,K}(\mathrm{b})$ with probability $1/2^b$. Summing over all possible queries, we obtain $\mathbf{Pr}\left(\mathrm{bad}_{2,K}(\mathrm{b})\right) \leq \frac{ps}{2^b}$.

Concluding the case of $\mathrm{bad}_{2,K}$, we obtain $\mathbf{Pr}\left(\mathrm{bad}_{2,K}\right) \leq \frac{p}{2^k} + \frac{ps}{2^b}$.

Regarding $\mathrm{bad}_{K,K}$, let $(K_Y, K_Y^\star), (K_{Y'}, K_{Y'}^\star) \in \nu_K$ by any two distinct queries. $K_Y^\star$ and $K_{Y'}^\star$ are two independently generated random values and the condition is satisfied with probability $1/2^b$. Summing over all possible queries, we obtain $\mathbf{Pr}\left(\mathrm{bad}_{K,K}\right) \leq \frac{\binom{s}{2}}{2^b}$.

To conclude, from (8) we obtain

$$\begin{aligned} \mathbf{Pr}\left(X_{\mathcal{P}} \in \mathcal{V}_{\mathrm{bad}}\right) &\leq \frac{q^2}{2^\epsilon} + \frac{2qp}{2^\epsilon} + \frac{qs}{2^\epsilon} + \frac{qs}{\min\{2^\epsilon, 2^b - q\}} + \frac{p}{2^k} + \frac{ps}{2^b} + \frac{\binom{s}{2}}{2^b} \ , \\ &\leq \frac{4.5q^2}{2^\epsilon} + \frac{3qp}{2^\epsilon} + \frac{p}{2^k} \ , \end{aligned}$$

where the simplification is done using $s \leq q$ and $2^b - q \geq 2^{b-1} \geq 2^{\epsilon-1}$.

Probability Ratio for Good Views. We will prove that for any *good* view ν, we have $\mathbf{Pr}\left(X_{\mathcal{O}} = \nu\right) \geq \mathbf{Pr}\left(X_{\mathcal{P}} = \nu\right)$. It suffices to compute the fraction of oracles that could result in view ν for both worlds $\mathcal{O}$ and $\mathcal{P}$. Recall that $\mathbf{D}$ never makes redundant queries, and that the views are assumed to be attainable and *good*. The proof follows [65] almost verbatim, but a few technicalities occur and we include the proof for completeness.

Regarding the real world, goodness of the view guarantees that (a) every tuple in ν_1 defines exactly one input/output-tuple of P, (b) every tuple in ν_2

and ν_K corresponds to exactly one input/output-tuple of P, and (c) that none of these tuples collide in the input or the output. Recall that the input values in ν_K are of the form $Y \parallel K$. Therefore we derive:

$$\begin{aligned}\mathbf{Pr}\left(X_{\mathcal{O}} = \nu\right) &= \mathbf{Pr}\left(K' \xleftarrow{\$} \{0,1\}^k \ : \ K' = K\right) \cdot \\ &\quad \mathbf{Pr}\left(P \xleftarrow{\$} \mathsf{Perm}(b) \ : \ \widetilde{E} \vdash \nu_1 \wedge P \vdash \nu_2 \wedge P \vdash \nu_K\right) \\ &= \frac{1}{2^k} \cdot \frac{(2^b - (q+p+s))!}{2^b!} \ .\end{aligned}$$

Regarding the ideal world, we group the elements of ν_1 according to the tweak values. For $T \in \mathcal{T}$, define

$$q_T = |\{(T, M, C) \in \nu_1 \mid M, C \in \{0,1\}^b\}| \ ,$$

where $\sum_T q_T = q$. Now,

$$\begin{aligned}\mathbf{Pr}\left(X_{\mathcal{P}} = \nu\right) &= \mathbf{Pr}\left(K' \xleftarrow{\$} \{0,1\}^k \ : \ K' = K\right) \cdot \\ &\quad \prod_{j=1}^{s} \mathbf{Pr}\left(K^\star_{Y_j}{}' \xleftarrow{\$} \{0,1\}^b \ : \ K^\star_{Y_j}{}' = K^\star_{Y_j}\right) \cdot \\ &\quad \mathbf{Pr}\left(\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b) \ : \ \widetilde{\pi} \vdash \nu_1\right) \cdot \\ &\quad \mathbf{Pr}\left(P \xleftarrow{\$} \mathsf{Perm}(b) \ : \ P \vdash \nu_2\right) \\ &= \frac{1}{2^k} \cdot \frac{1}{(2^b)^s} \cdot \prod_T \frac{(2^b - q_T)!}{2^b!} \cdot \frac{(2^b - p)!}{2^b!} \ .\end{aligned}$$

Note that $\frac{1}{(2^b)^s} \leq \frac{(2^b-s)!}{2^b!}$. Note furthermore that for any $\alpha, \beta \leq 2^b$ we have $\frac{(2^b-\alpha)!}{2^b!} \cdot \frac{(2^b-\beta)!}{2^b!} \leq \frac{(2^b-(\alpha+\beta))!}{2^b!}$. We thus get

$$\mathbf{Pr}\left(X_{\mathcal{P}} = \nu\right) \leq \frac{1}{2^k} \cdot \frac{(2^b - (q+p+s))!}{2^b!} = \mathbf{Pr}\left(X_{\mathcal{O}} = \nu\right) \ .$$

This completes the proof.

B Proof of Theorem 5 (Security of OPP)

Let $K \xleftarrow{\$} \{0,1\}^k$ and $P \xleftarrow{\$} \mathsf{Perm}(b)$. Let $\mathbf{D}$ be a *nonce-respecting* AE distinguisher against OPP, which means that it never makes an encryption query for a nonce that was used before. For brevity and rigority, write $\mathcal{E}_K^P := \mathsf{OPP}\mathcal{E}_K$ and $\mathcal{D}_K^P := \mathsf{OPP}\mathcal{D}_K$, including an explicit mentioning of the underlying primitive P.

As explained in Section 4, we can identify the tweakable blockcipher $\widetilde{E}$ of Section 3 in OPP. It is used for tweak space $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1$, where

$$\mathcal{T}_0 = \{0,1\}^{b-k} \times \{0, 1, \ldots, 2^{1020} - 1\} \times \{0,1,2,3\} \times \{0\} \ ,$$

$$\mathcal{T}_1 = \{0,1\}^{b-k} \times \{0,1,\ldots,2^{1020}-1\} \times \{0,1,2,3\} \times \{1\} \ .$$

We replace $\widetilde{E}$ with a random secret tweakable permutation $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T},b)$, and find by a hybrid argument:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{OPP},P}(\mathbf{D}) &= \Delta_{\mathbf{D}}(\mathcal{E}_K^P, \mathcal{D}_K^P, P^{\pm} \ ; \ \$_{\mathcal{E}}, \bot, P^{\pm}) \\
&= \Delta_{\mathbf{D}}(\mathcal{E}_{0^k}^{\widetilde{E}_K^P}, \mathcal{D}_{0^k}^{\widetilde{E}_K^P}, P^{\pm} \ ; \ \$_{\mathcal{E}}, \bot, P^{\pm}) \\
&\leq \Delta_{\mathbf{D}}(\mathcal{E}_{0^k}^{\widetilde{\pi}}, \mathcal{D}_{0^k}^{\widetilde{\pi}}, P^{\pm} \ ; \ \$_{\mathcal{E}}, \bot, P^{\pm}) + \Delta_{\mathbf{D}_1}((\widetilde{E}_K^P)^{\pm}, P^{\pm} \ ; \ \widetilde{\pi}^{\pm}, P^{\pm}) \\
&= \Delta_{\mathbf{D}}(\mathcal{E}_{0^k}^{\widetilde{\pi}}, \mathcal{D}_{0^k}^{\widetilde{\pi}}, P^{\pm} \ ; \ \$_{\mathcal{E}}, \bot, P^{\pm}) + \mathbf{Adv}^{\widetilde{\mathrm{mprp}}}_{\widetilde{E},P}(\mathbf{D}_1) \ ,
\end{aligned}$$

where $\mathbf{D}_1$ is some MTPRP distinguisher making at most σ construction queries and at most p queries to $P^{\pm}$ (in the $q_{\mathcal{E}}+q_{\mathcal{D}}$ evaluations of $\mathcal{E}$ and $\mathcal{D}$, the underlying $\widetilde{E}$ is evaluated at most σ times). By Lemma 4, the masking space $\mathcal{T}$ is b-proper, and by Theorem 2 we obtain

$$\mathbf{Adv}^{\widetilde{\mathrm{mprp}}}_{\widetilde{E},P}(\mathbf{D}_1) \leq \mathbf{Adv}^{\widetilde{\mathrm{mprp}}}_{\widetilde{E},P}(\sigma,p) \leq \frac{4.5\sigma^2}{2^b} + \frac{3\sigma p}{2^b} + \frac{p}{2^k} \ .$$

We proceed with the remaining Δ-distance. First note that all construction oracles are independent of $P^{\pm}$ and hence we can drop it without loss of generality. Now, this Δ-distance is in fact the AE security of ΘCB3 of Krovetz and Rogaway [58]. Even though they analyze the privacy and authenticity of ΘCB3 separately, their bounds directly apply and we obtain

$$\Delta_{\mathbf{D}}(\mathcal{E}_{0^k}^{\widetilde{\pi}}, \mathcal{D}_{0^k}^{\widetilde{\pi}}, P^{\pm} \ ; \ \$_{\mathcal{E}}, \bot, P^{\pm}) = \Delta_{\mathbf{D}}(\mathcal{E}^{\widetilde{\pi}}, \mathcal{D}^{\widetilde{\pi}} \ ; \ \$_{\mathcal{E}}, \bot) \leq \frac{2^{n-\tau}}{2^{n-1}} \ .$$

Concluding, we find that

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{OPP},P}(\mathbf{D}) \leq \frac{4.5\sigma^2}{2^b} + \frac{3\sigma p}{2^b} + \frac{p}{2^k} + \frac{2^{n-\tau}}{2^n-1} \ .$$

C Proof of Theorem 6 (Security of MRO)

The section is divided as follows. In Supporting Material C.1, we consider pseudorandom functions based on secret tweakable permutations, and derive a bound on the security of MROAbs. Similarly, we consider the IV-CPA security of MROEnc in Supporting Material C.2. The proof of Theorem 6 is then given in Supporting Material C.3.

C.1 Secret-Primitive Pseudorandom Functions

Let $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T},b)$ be a tweakable permutation with $\mathcal{T}$ of Section 5. Let $\mathcal{F}^{\widetilde{\pi}} : \{0,1\}^* \to \{0,1\}^{\tau}$ be a keyless MAC function based on $\widetilde{\pi}$ that processes a message M of arbitrary length to derive a tag $T \in \{0,1\}^{\tau}$. (Theoretically,

$\widetilde{\pi}$ functions as the key to $\mathcal{F}$, but for the sake of presentation it is better to view $\mathcal{F}$ as a keyless function.) Define by $\$_{\mathcal{F}}$ an idealized version of $\mathcal{F}$, which returns $T \xleftarrow{\$} \{0,1\}^\tau$ for every input. We define the pseudorandom function (PRF) security of $\mathcal{F}$ based on secret $\widetilde{\pi}$ as

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathcal{F}}(\mathbf{D}) = \Delta_{\mathbf{D}}(\mathcal{F}^{\widetilde{\pi}}\,;\,\$_{\mathcal{F}})\ , \tag{9}$$

where the probabilities are taken over the random choices of $\widetilde{\pi}$ and $\$_{\mathcal{F}}$. By $\mathbf{Adv}^{\mathrm{prf}}_{\mathcal{F}}(q,\sigma)$ we denote the maximum advantage over all distinguishers that make at most q queries to the construction encryption oracle, of total length at most σ padded blocks.

Secret-Primitive MROAbs. We will analyze the PRF security of a keyless variant $\mathsf{MROAbs} : \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^\tau$ based on ideal tweakable permutation $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$, cf. Fig. 8.

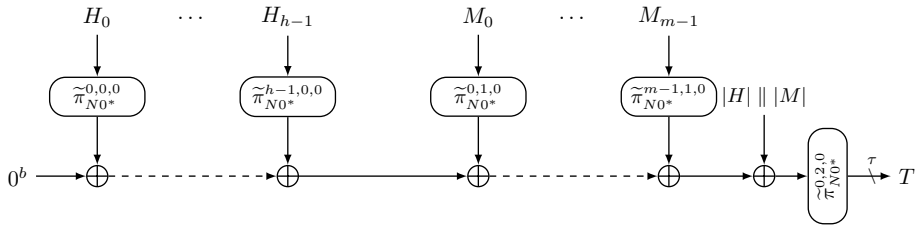

Fig. 8: Keyless $\mathsf{MROAbs}^{\widetilde{\pi}}$ based on secret tweakable permutation $\widetilde{\pi}$

Lemma 10. *Let* $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$ *and consider* $\mathsf{MROAbs}^{\widetilde{\pi}}$ *of Fig. 8. Then,* $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{MROAbs}}(q,\sigma) \le \frac{\sigma^2}{2^b}$.

Proof. Note that every different nonce sets an independent instance of $\mathsf{MROAbs}^{\widetilde{\pi}}$, as $\widetilde{\pi}$ is a random tweakable permutation. For $N \in \{0,1\}^n$, denote by σ_N the total complexity made for nonce N.

Consider $\mathsf{MROAbs}^{\widetilde{\pi}}(N,\cdot,\cdot)$ for any fixed nonce N. Theorem 15 of the full version of [71] shows that it is indistinguishable from $\$_{\mathcal{F}}(N,\cdot,\cdot)$ up to $\frac{\sigma_N^2}{2^b}$. Summation over all nonces gives $\sum_{N\in\{0,1\}^n} \frac{\sigma_N^2}{2^b} \le \frac{\sigma^2}{2^b}$. □

C.2 Secret-Primitive Encryption Schemes

Let $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$ be a tweakable permutation with $\mathcal{T}$ of Section 5. Let $\mathcal{E}^{\widetilde{\pi}} : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^\tau \times \{0,1\}^*$ be a keyless initial value based encryption scheme based on $\widetilde{\pi}$ that processes a message M of arbitrary length as follows:

it selects a random initial value $IV \xleftarrow{\$} \{0,1\}^\tau$ and uses it to derive a ciphertext $C \in \{0,1\}^{|M|}$. It is required to be invertible for fixed IV and nonce N, and its inverse is denoted by $(\mathcal{E}^{\widetilde{\pi}})^{-1}(IV, N, \cdot)$. Define by $\$_{\mathcal{E}}$ an idealized version of $\mathcal{E}$, which returns $IV \parallel C \xleftarrow{\$} \{0,1\}^{\tau+|M|}$ for every input. Following, Namprempre et al. [68], we define the IV-based chosen plaintext attack (IV-CPA) security of $\mathcal{E}$ based on P as

$$\mathbf{Adv}^{\text{iv-cpa}}_{\mathcal{E}}(\mathbf{D}) = \Delta_{\mathbf{D}}(\mathcal{E}^{\widetilde{\pi}}\,;\, \$_{\mathcal{E}})\ , \tag{10}$$

where the probabilities are taken over the random choices of $\mathcal{E}$, $\widetilde{\pi}$ and $\$_{\mathcal{E}}$. By $\mathbf{Adv}^{\text{iv-cpa}}_{\mathcal{E}}(q, \sigma)$ we denote the maximum advantage over all distinguishers that make at most q queries to the construction encryption oracle, of total length at most σ padded blocks.

Secret-Primitive MROEnc. We will analyze the IV-CPA security of a keyless variant $\mathsf{MROEnc} : \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^\tau \to \{0,1\}^*$ based on ideal tweakable permutation $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$, cf. Fig. 9. It is required to be given a random $T \xleftarrow{\$} \{0,1\}^\tau$ (included as explicit parameter to MROEnc for simplicity), which operates as the IV, and produces a ciphertext on input of a nonce and a message. Note that MROEnc only uses $\widetilde{\pi}$ for tweaks of the form $(N, 0, 0, 1)$, but generality is maintained to suit the analysis.

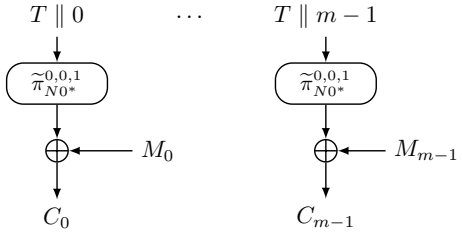

Fig. 9: Keyless $\mathsf{MROEnc}^{\widetilde{\pi}}$ based on secret tweakable permutation $\widetilde{\pi}$

Lemma 11. *Let* $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$ *and consider* $\mathsf{MROEnc}^{\widetilde{\pi}}$ *of Fig. 9. Then,* $\mathbf{Adv}^{\text{iv-cpa}}_{\mathsf{MROEnc}}(q, \sigma) \le \frac{\sigma^2}{2^b} + \frac{q^2}{2^{\tau+1}}$.

Proof. Note that every different nonce sets an independent instance of $\mathsf{MROEnc}^{\widetilde{\pi}}$, as $\widetilde{\pi}$ is a random tweakable permutation. For $N \in \{0,1\}^n$, denote by q_N the total number of queries and by σ_N the total complexity made for nonce N.

Consider $\mathsf{MROEnc}^{\widetilde{\pi}}(N, \cdot, \cdot)$ for any fixed nonce N. It reminds of XOR\$ from Bellare et al. [9] with two differences: (i) XOR\$ uses a random function while MROEnc uses a permutation $\widetilde{\pi}^{0,0,1}_{N0^*}$, and (ii) XOR\$ inputs $T + \text{ctr}$ to the primitive

while MROEnc inputs $T \,\|\, \mathrm{ctr}$. The proof nevertheless mostly carries over. As a first step, we replace $\widetilde{\pi}^{0,0,1}_{N0^*}$ by a random function $\$_{\mathcal{F}} : \{0,1\}^b \to \{0,1\}^b$. This step costs us at most $\frac{\sigma_N^2}{2^b}$ by the RP/RF-switch. Note that every *new* initial value sets an independent instance of $\mathsf{MROEnc}^{\$_{\mathcal{F}}}$, as $\$_{\mathcal{F}}$ is a random function. Denote the q initial values by $T_1, \ldots, T_q$. Clearly, $\mathsf{MROEnc}^{\$_{\mathcal{F}}}$ behaves like a random $\$_{\mathcal{E}}$ as long as $T_i \neq T_j$. A collision in the T's happens with probability at most $\frac{q^2}{2^{\tau+1}}$, due to our condition that the T's are uniformly randomly generated from $\{0,1\}^\tau$. Concluding, $\widetilde{\mathsf{MROEnc}}^{\widetilde{\pi}}(N,\cdot,\cdot)$ is indistinguishable from $\$_{\mathcal{E}}(N,\cdot,\cdot)$ up to $\frac{\sigma_N^2}{2^b} + \frac{q_N^2}{2^{\tau+1}}$. Summation over all nonces gives $\sum_{N\in\{0,1\}^n} \frac{\sigma_N^2}{2^b} + \frac{q_N^2}{2^{\tau+1}} \leq \frac{\sigma^2}{2^b} + \frac{q^2}{2^{\tau+1}}$. □

C.3 Proof of Theorem 6

Let $K \xleftarrow{\$} \{0,1\}^k$ and $P \xleftarrow{\$} \mathsf{Perm}(b)$. Let $\mathbf{D}$ be a *nonce-misusing* AE distinguisher against MRO. For rigority, write $\mathcal{E}[\mathsf{Abs}^P_K, \mathsf{Enc}^P_K] := \mathsf{MRO}\mathcal{E}_K$ and $\mathcal{D}[\mathsf{Abs}^P_K, \mathsf{Dec}^P_K] := \mathsf{MRO}\mathcal{D}_K$, including an explicit mentioning of the underlying primitive P.

As explained in Section 5, we can identify the tweakable blockcipher $\widetilde{E}$ of Section 3 in MRO. It is used for tweak space

$$\mathcal{T} = \mathcal{T}_0 = \{0,1\}^{b-k} \times \{0,1,\ldots,2^{1020}-1\} \times \{0,1,2\} \times \{0,1\} \ .$$

We replace $\widetilde{E}$ with a random secret tweakable permutation $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$, and find by a hybrid argument:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MRO},P}(\mathbf{D}) &= \Delta_{\mathbf{D}}(\mathcal{E}[\mathsf{Abs}^P_K, \mathsf{Enc}^P_K], \mathcal{D}[\mathsf{Abs}^P_K, \mathsf{Dec}^P_K], P^{\pm}\ ;\ \$_{\mathcal{E}}, \bot, P^{\pm}) \\
&\leq \Delta_{\mathbf{D}}(\mathcal{E}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Enc}^{\widetilde{\pi}}], \mathcal{D}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Dec}^{\widetilde{\pi}}], P^{\pm}\ ;\ \$_{\mathcal{E}}, \bot, P^{\pm}) + \\
&\quad \Delta_{\mathbf{D}_1}(\widetilde{E}^P_K, P^{\pm}\ ;\ \widetilde{\pi}, P^{\pm}) \\
&= \Delta_{\mathbf{D}}(\mathcal{E}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Enc}^{\widetilde{\pi}}], \mathcal{D}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Dec}^{\widetilde{\pi}}], P^{\pm}\ ;\ \$_{\mathcal{E}}, \bot, P^{\pm}) + \\
&\quad \mathbf{Adv}^{\widetilde{\mathrm{mprp}}}_{\widetilde{E},P}(\mathbf{D}_1) \ ,
\end{aligned}$$

where $\mathsf{Abs}^{\widetilde{\pi}}$ is the keyless PRF of Fig. 8 and $\mathsf{Enc}^{\widetilde{\pi}}$ the encryption scheme of Fig. 9 (with $\mathsf{Dec}^{\widetilde{\pi}}$ its corresponding decryption function), and where $\mathbf{D}_1$ is some MTPRP distinguisher making at most σ construction queries and at most p queries to $P^{\pm}$ (in the $q_{\mathcal{E}} + q_{\mathcal{D}}$ evaluations of $\mathcal{E}$ and $\mathcal{D}$, the underlying $\widetilde{E}$ is evaluated at most σ times). By Lemma 4, the masking space $\mathcal{T}$ is b-proper, and Theorem 2 applies.

We proceed with the remaining Δ-distance. As before, the construction oracles are independent of $P^{\pm}$ and we can drop it without loss of generality. Let $\widetilde{\pi}' \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$. Then,

$$\begin{aligned}
&\Delta_{\mathbf{D}}(\mathcal{E}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Enc}^{\widetilde{\pi}}], \mathcal{D}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Dec}^{\widetilde{\pi}}], P^{\pm}\ ;\ \$_{\mathcal{E}}, \bot, P^{\pm}) \\
&= \Delta_{\mathbf{D}}(\mathcal{E}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Enc}^{\widetilde{\pi}}], \mathcal{D}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Dec}^{\widetilde{\pi}}]\ ;\ \$_{\mathcal{E}}, \bot)
\end{aligned}$$

$$= \Delta_{\mathbf{D}}(\mathcal{E}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Enc}^{\widetilde{\pi}'}], \mathcal{D}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Dec}^{\widetilde{\pi}'}]\ ;\ \$_{\mathcal{E}}, \bot)\ ,$$

as Abs and $\mathsf{Enc}/\mathsf{Dec}$ evaluate $\widetilde{\pi}$ on different tweaks.

Above reduction allows to view the absorption and encryption to be independently keyed (via $\widetilde{\pi}$ and $\widetilde{\pi}'$). This paves the path for the use of a separation of AE security into PRF security of Abs and IV-CPA security of Enc, as inspired by the MAC-then-Encrypt approach of Namprempre et al. [68] and its adaption to misuse resistance as presented by Gueron and Lindell [40]. For completeness, we re-derive it for our current setting. We have

$$\begin{aligned}
&\Delta_{\mathbf{D}}(\mathcal{E}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Enc}^{\widetilde{\pi}'}], \mathcal{D}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Dec}^{\widetilde{\pi}'}]\ ;\ \$_{\mathcal{E}}, \bot) \\
&\leq \Delta_{\mathbf{D}}(\mathcal{E}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Enc}^{\widetilde{\pi}'}], \mathcal{D}[\mathsf{Abs}^{\widetilde{\pi}}, \mathsf{Dec}^{\widetilde{\pi}'}]\ ;\ \mathcal{E}[\$_{\mathcal{F}}, \mathsf{Enc}^{\widetilde{\pi}'}], \mathcal{D}[\$_{\mathcal{F}}, \mathsf{Dec}^{\widetilde{\pi}'}]) + \\
&\quad \Delta_{\mathbf{D}}(\mathcal{E}[\$_{\mathcal{F}}, \mathsf{Enc}^{\widetilde{\pi}'}], \mathcal{D}[\$_{\mathcal{F}}, \mathsf{Dec}^{\widetilde{\pi}'}]\ ;\ \$_{\mathcal{E}}, \bot) \\
&\leq \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{Abs}}(\mathbf{D}_2) + \Delta_{\mathbf{D}}(\mathcal{E}[\$_{\mathcal{F}}, \mathsf{Enc}^{\widetilde{\pi}'}], \mathcal{D}[\$_{\mathcal{F}}, \mathsf{Dec}^{\widetilde{\pi}'}]\ ;\ \$_{\mathcal{E}}, \bot)\ ,
\end{aligned}$$

where $\mathbf{D}_2$ is some PRF distinguisher making at most $q_{\mathcal{E}} + q_{\mathcal{D}}$ queries to the construction encryption oracle, of total length at most σ blocks. Regarding the remaining distance:

$$\begin{aligned}
&\Delta_{\mathbf{D}}(\mathcal{E}[\$_{\mathcal{F}}, \mathsf{Enc}^{\widetilde{\pi}'}], \mathcal{D}[\$_{\mathcal{F}}, \mathsf{Dec}^{\widetilde{\pi}'}]\ ;\ \$_{\mathcal{E}}, \bot) \\
&\leq \Delta_{\mathbf{D}}(\mathcal{E}[\$_{\mathcal{F}}, \mathsf{Enc}^{\widetilde{\pi}'}], \mathcal{D}[\$_{\mathcal{F}}, \mathsf{Dec}^{\widetilde{\pi}'}]\ ;\ \mathcal{E}[\$_{\mathcal{F}}, \mathsf{Enc}^{\widetilde{\pi}'}], \bot) + \Delta_{\mathbf{D}}(\mathcal{E}[\$_{\mathcal{F}}, \mathsf{Enc}^{\widetilde{\pi}'}], \bot\ ;\ \$_{\mathcal{E}}, \bot) \\
&\leq \Delta_{\mathbf{D}}(\mathcal{E}[\$_{\mathcal{F}}, \mathsf{Enc}^{\widetilde{\pi}'}], \mathcal{D}[\$_{\mathcal{F}}, \mathsf{Dec}^{\widetilde{\pi}'}]\ ;\ \mathcal{E}[\$_{\mathcal{F}}, \mathsf{Enc}^{\widetilde{\pi}'}], \bot) + \mathbf{Adv}^{\text{iv-cpa}}_{\mathsf{Enc}}(\mathbf{D}_3)\ ,
\end{aligned}$$

where $\mathbf{D}_3$ is some IV-CPA distinguisher making at most $q_{\mathcal{E}}$ queries to the construction encryption oracle, of total length at most σ blocks. The remaining distance boils down to forging a tag for a random $\$_{\mathcal{F}}$, in which $\mathbf{D}$ succeeds with probability at most $\frac{q_{\mathcal{D}}}{2^{\tau}}$. Concluding, we find that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MRO},P}(\mathbf{D}) &\leq \mathbf{Adv}^{\widetilde{\mathrm{mprp}}}_{\widetilde{E},P}(\mathbf{D}_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{Abs}}(\mathbf{D}_2) + \mathbf{Adv}^{\text{iv-cpa}}_{\mathsf{Enc}}(\mathbf{D}_3) + \frac{q_{\mathcal{D}}}{2^{\tau}}\ , \\
&\leq \mathbf{Adv}^{\widetilde{\mathrm{mprp}}}_{\widetilde{E},P}(\sigma, p) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{Abs}}(q_{\mathcal{E}} + q_{\mathcal{D}}, \sigma) + \mathbf{Adv}^{\text{iv-cpa}}_{\mathsf{Enc}}(q_{\mathcal{E}}, \sigma) + \frac{q_{\mathcal{D}}}{2^{\tau}}\ .
\end{aligned}$$

A bound on the first term follows from Theorem 2 and the b-properness of the masking. The second two advantages are bounded using Lemmas 10 and 11.

D Proof of Theorem 7 (Security of MRS)

The proof is structurally different from the one of OPP and MRO, and particular does not rely on MEM. The remainder of this section is as follows. In Supporting Material D.1, we present a (normal) blockcipher underlying MRS. Then, in Supporting Material D.2 we discuss the Duplex construction. The proof is given in Supporting Material D.3.

D.1 Blockcipher Construction

Let $P \xleftarrow{\$} \mathsf{Perm}(b)$ and write $b = c + r$. Define the following blockcipher $\widetilde{E}_{\mathsf{MRS}} : \{0,1\}^k \times \{0,1\}^b \to \{0,1\}^b$:

$$\widetilde{E}_{\mathsf{MRS}}(K, M) = P((0^{b-k} \,\|\, K) \oplus M) \oplus (0^{b-k} \,\|\, K) \ .$$

Note that this is a tweakable blockcipher with empty tweak space, $\mathcal{T} = \emptyset$, and the model of Section 2 carries over. We will explain how this blockcipher appears in MRS. Note that in MRSAbs, the state is initialized with $0^{b-k} \,\|\, K$. The trick will be to XOR this value everywhere in-between two consecutive evaluations of P, in a similar fashion as done in [22,4,66]. As $k \leq c$, this adjustment is artificial and it does not change the scheme. In this case, MRSAbs is effectively just compressing its blocks one by one, interleaved with an evaluation of $\widetilde{E}_{\mathsf{MRS}}(K, \cdot)$. The situation for MRSEnc is identical.

Lemma 12. *Let* $P \xleftarrow{\$} \mathsf{Perm}(b)$. *Then,* $\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}_{\mathsf{MRS}},P}(q,p) \leq \frac{2qp}{2^k}$.

Proof. The proof is a straightforward adaption of Theorem 2 of [4] to k-bit keys. See also [66]. □

D.2 Secret-Primitive Full-State Duplex

Bertoni et al. [11] introduced the Duplex construction, and Mennink et al. [66] generalized it to the full-state Duplex. At a high level, the full-state Duplex consists of two interfaces. Duplex.init gets as input a key $K \in \{0,1\}^k$, initializes the state as $0^{b-k} \,\|\, K$, and outputs nothing. Duplex.dup gets as input a message block $M \in \{0,1\}^b$ and a natural number $z \leq r$, transforms the state using M, and outputs a string $Z \in \{0,1\}^z$.[9] We will consider a keyless variant of Duplex based on *secret* permutation $\pi \xleftarrow{\$} \{0,1\}^b$ of Fig. 10. Theoretically, π functions as the key to Duplex.

Mennink et al. proved that this secret-primitive Duplex behaves like a random functionality up to common prefix. Define by $\$_\infty : \{0,1\}^* \to \{0,1\}^\infty$ a random function that takes inputs of arbitrary length and returns random infinite strings, where each bit is randomly drawn. Define by $\$_{\mathrm{dup}}$ a stateful random function with two interfaces: $\$_{\mathrm{dup}}$.init gets no input and initializes its state St to the empty string; $\$_{\mathrm{dup}}$.dup gets as input a message $M \in \{0,1\}^b$ and a natural number $z \leq r$, updates the state as $S \| M$, and outputs $\mathsf{left}_z(\$_\infty(S))$. Note that if $\$_{\mathrm{dup}}$ is never queried in such a way that it calls $\$_\infty$ twice on the same S, it always responds with random strings. For brevity and applicability later on, define the Duplex (DUP) security of Duplex based on secret π as

$$\mathbf{Adv}^{\mathrm{dup}}_{\mathsf{Duplex}}(\mathbf{D}) = \Delta_{\mathbf{D}}(\mathsf{Duplex}^\pi \ ; \ \$_{\mathrm{dup}}) \ , \tag{11}$$

[9] In fact, the classical Duplex is slightly different as it takes message blocks $M \in \{0,1\}^{<b}$ and performs 10^*-padding on these blocks. We will use the Duplex as primitive in another mode, and the padding requirements are handled by the mode that is placed on top of Duplex. The results of [66] carry over.

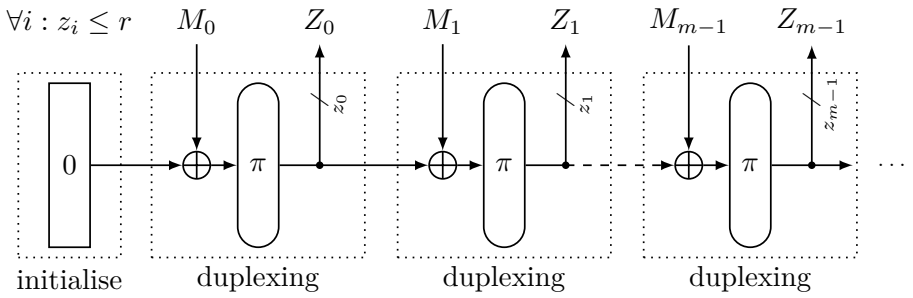

Fig. 10: Keyless Duplex^{π} based on secret permutation π

where the probabilities are taken over the random choices of π and $\$_{\mathrm{dup}}$. By $\mathbf{Adv}^{\mathrm{dup}}_{\mathsf{Duplex}}(q, \sigma)$ we denote the maximum advantage over all distinguishers that make at most q queries to the construction encryption oracle, of total length at most σ padded blocks. Translated to counting over total complexity σ (instead of over λ blocks per query), Mennink et al. [66] proved that Duplex behaves like $\$_{\mathrm{dup}}$.

Lemma 13 (Mennink et al. [66]). *Let* $\pi \xleftarrow{\$} \mathsf{Perm}(b)$ *and consider* Duplex^{π} *of Fig. 10. Then,* $\mathbf{Adv}^{\mathrm{dup}}_{\mathsf{Duplex}}(q, \sigma) \leq \frac{\sigma^2}{2^b} + \frac{\sigma^2}{2^c}$.

We remark that the main result of [66] is the security of Duplex in the public permutation setting, which effectively combines Lemmas 12 and 13. We have separated the results to suit later analysis.

D.3 Proof of Theorem 7

Let $K \xleftarrow{\$} \{0,1\}^k$ and $P \xleftarrow{\$} \mathsf{Perm}(b)$. Let $\mathbf{D}$ be an AE distinguisher against MRS. For brevity and rigority, write $\mathcal{E}^P_K := \mathsf{MRS}\mathcal{E}_K$ and $\mathcal{D}^P_K := \mathsf{MRS}\mathcal{D}_K$, including an explicit mentioning of the underlying primitive P. By a hybrid argument, we can identify the blockcipher $\widetilde{E}_{\mathsf{MRS}}$ in MRS and replace it with a random secret permutation $\pi \xleftarrow{\$} \mathsf{Perm}(b)$:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MRS},P}(\mathbf{D}) &= \Delta_{\mathbf{D}}(\mathcal{E}^P_K, \mathcal{D}^P_K, P^{\pm}\,;\, \$_{\mathcal{E}}, \perp, P^{\pm}) \\
&= \Delta_{\mathbf{D}}(\mathcal{E}^{\widetilde{E}^P_{\mathsf{MRS},K}}_{0^k}, \mathcal{D}^{\widetilde{E}^P_{\mathsf{MRS},K}}_{0^k}, P^{\pm}\,;\, \$_{\mathcal{E}}, \perp, P^{\pm}) \\
&\leq \Delta_{\mathbf{D}}(\mathcal{E}^{\pi}_{0^k}, \mathcal{D}^{\pi}_{0^k}, P^{\pm}\,;\, \$_{\mathcal{E}}, \perp, P^{\pm}) + \Delta_{\mathbf{D}_1}(\widetilde{E}^P_{\mathsf{MRS},K}, P^{\pm}\,;\, \pi, P^{\pm}) \\
&= \Delta_{\mathbf{D}}(\mathcal{E}^{\pi}_{0^k}, \mathcal{D}^{\pi}_{0^k}, P^{\pm}\,;\, \$_{\mathcal{E}}, \perp, P^{\pm}) + \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}_{\mathsf{MRS}},P}(\mathbf{D}_1)\ ,
\end{aligned}$$

where $\mathbf{D}_1$ is some TPRP distinguisher making at most σ construction queries and at most p queries to $P^{\pm}$ (in the $q_{\mathcal{E}} + q_{\mathcal{D}}$ evaluations of $\mathcal{E}$ and $\mathcal{D}$, the underlying $\widetilde{E}$ is evaluated at most σ times).

We proceed with the remaining Δ-distance. First, for simplicity, we drop the explicit keying of 0^k for convenience. Second, note that all construction oracles are independent of $P^{\pm}$ and hence we can drop it without loss of generality. Third, note that $\mathcal{E}^{\pi}$ and $\mathcal{D}^{\pi}$ are in fact equivalent to $\mathcal{E}'[\mathsf{Duplex}^{\pi}]$ and $\mathcal{D}'[\mathsf{Duplex}^{\pi}]$ defined as follows:

Algorithm: $\mathsf{Absorb}(A)$

1. **if** $|A| > 0$ **then**
2. $\quad A_0 \parallel \cdots \parallel A_{a-1} \leftarrow \mathsf{pad}_b^0(A)$
3. $\quad$ **for** $i \in \{0, \ldots, a-1\}$ **do**
4. $\quad\quad \mathsf{Duplex}^{\pi}.\mathrm{dup}(A_i, 0)$
5. $\quad$ **end**
6. **end**
7. **return** S

Algorithm: $\mathsf{Abs}[\mathsf{Duplex}^{\pi}](N, H, M)$

1. $\mathsf{Duplex}^{\pi}.\mathrm{init}()$
2. $\mathsf{Duplex}^{\pi}.\mathrm{dup}(N \parallel 0^* \parallel 0 \parallel 0^k, 0)$
3. $\mathsf{Absorb}(H)$
4. $\mathsf{Absorb}(M)$
5. $S \leftarrow \mathsf{Duplex}^{\pi}.\mathrm{dup}(|H| \parallel |M|, \tau)$
6. **return** S

Algorithm: $\mathsf{Enc}[\mathsf{Duplex}^{\pi}](T, M)$

1. $C \leftarrow \varepsilon$
2. **if** $|M| > 0$ **then**
3. $\quad \mathsf{Duplex}^{\pi}.\mathrm{init}()$
4. $\quad M_{-1} \leftarrow T \parallel 0^* \parallel 1 \parallel 0^k$
5. $\quad M_0 \parallel \cdots \parallel M_{m-1} \leftarrow \mathsf{pad}_r^0(M)$
6. $\quad$ **for** $i \in \{0, \ldots, m-1\}$ **do**
7. $\quad\quad S \leftarrow \mathsf{Duplex}^{\pi}.\mathrm{dup}(M_{i-1}, r)$
8. $\quad\quad C_i \leftarrow S \oplus M_i$
9. $\quad\quad C \leftarrow C \parallel C_i$
10. $\quad$ **end**
11. **end**
12. **return** $\mathsf{left}_{|M|}(C)$

Algorithm: $\mathsf{Dec}[\mathsf{Duplex}^{\pi}](T, C)$

1. $M \leftarrow \varepsilon$
2. **if** $|C| > 0$ **then**
3. $\quad \mathsf{Duplex}^{\pi}.\mathrm{init}()$
4. $\quad M_{-1} \leftarrow T \parallel 0^* \parallel 1 \parallel 0^k$
5. $\quad C_0 \parallel \cdots \parallel C_{m-1} \leftarrow \mathsf{pad}_r^0(C)$
6. $\quad$ **for** $i \in \{0, \ldots, m-1\}$ **do**
7. $\quad\quad S \leftarrow \mathsf{Duplex}^{\pi}.\mathrm{dup}(M_{i-1}, r)$
8. $\quad\quad M_i \leftarrow S \oplus C_i$
9. $\quad\quad M \leftarrow M \parallel M_i$
10. $\quad$ **end**
11. **end**
12. **return** $\mathsf{left}_{|C|}(M)$

Algorithm: $\mathcal{E}'[\mathsf{Duplex}^{\pi}](N, H, M)$

1. $T \leftarrow \mathsf{Abs}[\mathsf{Duplex}^{\pi}](N, H, M)$
2. $C \leftarrow \mathsf{Enc}[\mathsf{Duplex}^{\pi}](T, M)$
3. **return** C, T

Algorithm: $\mathcal{D}'[\mathsf{Duplex}^{\pi}](N, H, C, T)$

1. $M \leftarrow \mathsf{Dec}[\mathsf{Duplex}^{\pi}](T, C)$
2. $T' \leftarrow \mathsf{Abs}[\mathsf{Duplex}^{\pi}](N, H, M)$
3. **if** $T = T'$ **then return** M **else return** $\perp$ **end**

We find,

$$
\begin{aligned}
&\Delta_{\mathbf{D}}(\mathcal{E}^{\pi}_{0^k}, \mathcal{D}^{\pi}_{0^k}, P^{\pm}\,;\, \$_{\mathcal{E}}, \perp, P^{\pm}) \\
&= \Delta_{\mathbf{D}}(\mathcal{E}^{\pi}, \mathcal{D}^{\pi}\,;\, \$_{\mathcal{E}}, \perp) \\
&= \Delta_{\mathbf{D}}(\mathcal{E}'[\mathsf{Duplex}^{\pi}], \mathcal{D}'[\mathsf{Duplex}^{\pi}]\,;\, \$_{\mathcal{E}}, \perp) \\
&\leq \Delta_{\mathbf{D}}(\mathcal{E}'[\$_{\mathrm{dup}}], \mathcal{D}'[\$_{\mathrm{dup}}]\,;\, \$_{\mathcal{E}}, \perp) + \Delta_{\mathbf{D}_2}(\mathsf{Duplex}^{\pi}\,;\, \$_{\mathrm{dup}}) \\
&= \Delta_{\mathbf{D}}(\mathcal{E}'[\$_{\mathrm{dup}}], \mathcal{D}'[\$_{\mathrm{dup}}]\,;\, \$_{\mathcal{E}}, \perp) + \mathbf{Adv}^{\mathrm{dup}}_{\mathsf{Duplex}}(\mathbf{D}_2)\ ,
\end{aligned}
$$

where $\mathbf{D}_2$ is some DUP distinguisher making at most $2(q_{\mathcal{E}} + q_{\mathcal{D}})$ queries of total length at most 2σ blocks.

The remaining Δ-distance is bounded by looking at the calls to $\mathcal{E}'[\$_{\mathrm{dup}}]$ and $\mathcal{D}'[\$_{\mathrm{dup}}]$ separately, and relies on the fact that $\$_{\mathrm{dup}}$ never calls $\$_{\infty}$ on the same state (cf. Supporting Material D.2). An important observation here is that *all* calls to $\$_{\infty}$ via $\mathsf{Abs}[\$_{\mathrm{dup}}]$ start with first block $N \parallel 0^* \parallel 0 \parallel 0^k$ while *all* calls to $\$_{\infty}$ via $\mathsf{Enc}[\$_{\mathrm{dup}}]$ and $\mathsf{Dec}[\$_{\mathrm{dup}}]$ start with first block $T \parallel 0^* \parallel 1 \parallel 0^k$. Hence, the absorption and encryption/decryption can be seen independently.

– Consider an encryption query $\mathcal{E}'[\$_{\mathrm{dup}}](N,H,M)$. The call $\mathsf{Abs}[\$_{\mathrm{dup}}]$ triggers a call to the underlying $\$_\infty$ on input

$$\big(N \,\|\, 0^* \,\|\, 0 \,\|\, 0^k\big) \,\|\, \mathsf{pad}_b(H) \,\|\, \mathsf{pad}_b(M) \,\|\, \big(|H| \,\|\, |M|\big) \ .$$

As $\mathbf{D}$ never repeats queries, this means that $\$_\infty$ has never been queried on this input and the response is $T \xleftarrow{\$} \{0,1\}^\tau$. The call $\mathsf{Enc}[\$_{\mathrm{dup}}]$ triggers calls to the underlying $\$_\infty$ on inputs

$$\begin{aligned}
&\big(T \,\|\, 0^* \,\|\, 1 \,\|\, 0^k\big) \ , \\
&\big(T \,\|\, 0^* \,\|\, 1 \,\|\, 0^k\big) \,\|\, M_0 \ , \\
&\qquad \vdots \\
&\big(T \,\|\, 0^* \,\|\, 1 \,\|\, 0^k\big) \,\|\, M_0 \,\|\, \cdots \,\|\, M_{m-1} \ ,
\end{aligned}$$

where $M_0 \,\|\, \cdots \,\|\, M_{m-1} \leftarrow \mathsf{pad}_r(M)$. *Unless if* T is an old tag, these calls to $\$_\infty$ are new and responded randomly, and we have $C \xleftarrow{\$} \{0,1\}^{|M|}$. If T is an old tag (either from an old encryption or an old decryption query), we cannot guarantee randomness of C, and this will be viewed as a bad event;
– Consider a decryption query $\mathcal{D}'[\$_{\mathrm{dup}}](N,H,C,T)$. Denote $M = \mathsf{Dec}[\$_{\mathrm{dup}}](T,C)$. If there is an older query of the form (N,H,M,T^*), then necessarily $T \neq T^*$ (because $\mathbf{D}$ never repeats queries) and the forgery cannot be successful. On the other hand, if there is no older query of the form (N,H,M,T^*) for any T^*, this means that the call to $\$_\infty$ made via $\mathsf{Abs}[\$_{\mathrm{dup}}]$ is new (the argument is as for the case of $\mathcal{E}'[\$_{\mathrm{dup}}]$) and responded with a random tag T^*. It equals T with probability $1/2^\tau$.

If we sum over all queries, we obtain that $(\mathcal{E}'[\$_{\mathrm{dup}}], \mathcal{D}'[\$_{\mathrm{dup}}])$ is perfectly indistinguishable from $(\$_{\mathcal{E}}, \perp)$, except with probability $\frac{q_{\mathcal{E}}^2/2 + q_{\mathcal{D}} q_{\mathcal{E}}}{2^\tau} + \frac{q_{\mathcal{D}}}{2^\tau}$. Concluding, we find that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{MRS},P}(\mathbf{D}) &\leq \mathbf{Adv}^{\mathrm{prp}}_{\widetilde{E}_{\mathsf{MRS},P}}(\mathbf{D}_1) + \mathbf{Adv}^{\mathrm{dup}}_{\mathsf{Duplex}}(\mathbf{D}_2) + \frac{q_{\mathcal{E}}^2/2 + q_{\mathcal{D}} q_{\mathcal{E}} + q_{\mathcal{D}}}{2^\tau} \ , \\
&\leq \mathbf{Adv}^{\mathrm{prp}}_{\widetilde{E}_{\mathsf{MRS},P}}(\sigma, p) + \mathbf{Adv}^{\mathrm{dup}}_{\mathsf{Duplex}}(2(q_{\mathcal{E}} + q_{\mathcal{D}}), 2\sigma) + \frac{q_{\mathcal{E}}^2/2 + q_{\mathcal{D}} q_{\mathcal{E}} + q_{\mathcal{D}}}{2^\tau} \ .
\end{aligned}$$

The proof is completed using Lemmas 12 and 13.

E Proof of Theorem 8 (Security of MRSO)

The proof is fairly similar to the one of MRO (Supporting Material C), the only major difference is in the use of a different tweakable blockcipher. In Supporting Material E.1 we identify the tweakable blockcipher used in MRSO. Then, in Section we consider the PRF security of MRSAbs. The proof of Theorem 8 is then given in Supporting Material E.3

E.1 Tweakable Blockcipher Constructions

It is already mentioned in Section 7 that a slightly different masking is employed. More detailed, the tweak space is

$$\mathcal{T} = \mathcal{T}_0 = \{0,1\}^{b-k} \times \{0,1\} \ ,$$

and the masking is

$$\delta : (K, X, i) \mapsto 0^\tau \parallel \mathsf{right}_{b-\tau}\big(\varphi^i(P(X \parallel K))\big) \ .$$

We briefly elaborate on the appearance of $\widetilde{E}_{\mathsf{MRSO}}$ in MRSO. At a high level, the trick for MRSAbs is to start considering appearances of $\widetilde{E}_{\mathsf{MRSO}}$ from the *second* permutation (unlike the *first* as done in Supporting Material D). This would work well if the value $P(N \parallel 0^* \parallel 0 \parallel K)$ is XORed twice everywhere in-between two permutation calls, but because at the end of MRSAbs the leftmost τ bits of the state are extracted, this is impossible. However, it is clear that by XORing only the rightmost $b - \tau$ bits into the state everywhere, this will only *decrease* the security of MRSAbs, and the artificial XORings do not alter the scheme. For MROEnc on the other hand, the change to only XORing the rightmost $b - \tau$ bits is for simplicity of analysis.

It is trivial to see that the masking is $(b - \tau)$-proper in accordance with Definition 1, and Theorem 2 applies.

E.2 Secret-Primitive Pseudorandom Functions MRSAbs

We will analyze the PRF security (cf. Supporting Material C.1) of a keyless variant $\mathsf{MRSAbs} : \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^\tau$ based on ideal tweakable permutation $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$. In more detail, we consider $\mathsf{MRSAbs}^{\widetilde{\pi}}$ of Fig. 11. Note that MRSAbs in MRSO only uses $\widetilde{\pi}$ for tweaks of the form $(N0^*, 0)$, but we maintain generality to suit the remainder of the proof.

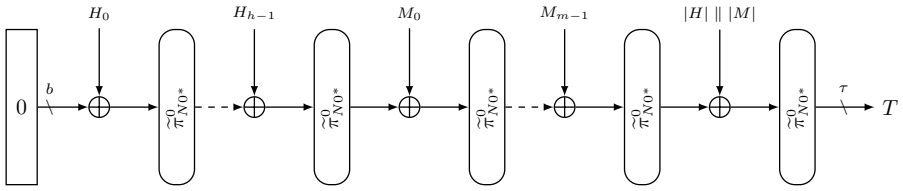

Fig. 11: Keyless $\mathsf{MRSAbs}^{\widetilde{\pi}}$ based on secret tweakable permutation $\widetilde{\pi}$

Lemma 14. *Let* $\widetilde{\pi} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{T}, b)$ *and consider* $\mathsf{MRSAbs}^{\widetilde{\pi}}$ *of Fig. 11. Then,* $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{MRSAbs}}(q, \sigma) \leq \frac{\sigma^2}{2^b} + \frac{\sigma^2}{2^{b-\tau}}$.

Proof. Note that every different nonce sets an independent instance of $\mathsf{MRSAbs}^{\widetilde{\pi}}$, as $\widetilde{\pi}$ is a random tweakable permutation. For $N \in \{0,1\}^n$, denote by σ_N the total complexity made for nonce N.

Consider $\mathsf{MRSAbs}^{\widetilde{\pi}}(N,\cdot,\cdot)$ for any fixed nonce N. This function is in fact equivalent to Duplex^{π} of Fig. 10 in the way as discussed in Supporting Material D.3, with $c = b - \tau$. As a first step, we replace Duplex^{π} with $\$_{\mathrm{dup}}$. By Lemma 13, this step costs at most $\frac{\sigma_N^2}{2^b} + \frac{\sigma_N^2}{2^{b-\tau}}$. Because the distinguisher never makes repeat queries every call to $\$_\infty$ triggered by $\$_{\mathrm{dup}}$ is made with a different state S, and the output is indistinguishable from $\$_{\mathcal{F}}$. Summation over all nonces gives $\sum_{N\in\{0,1\}^n} \frac{\sigma_N^2}{2^b} + \frac{\sigma_N^2}{2^{b-\tau}} \le \frac{\sigma^2}{2^b} + \frac{\sigma^2}{2^{b-\tau}}$. □

E.3 Proof of Theorem 8

The proof is equivalent to the one of Theorem 6 of Supporting Material C.3, with the difference that now we use Lemma 14 instead of Lemma 10, and we use the tweakable cipher of Supporting Material E.1, whose masking is only $(b-\tau)$-proper.