McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes

– Full Version* –

Ewan Fleischmann, Christian Forler, Stefan Lucks, and Jakob Wenzel

Bauhaus-University Weimar, Germany
{Ewan.Fleischmann, Christian.Forler, Stefan.Lucks, Jakob.Wenzel}@uni-weimar.de

Abstract. On-Line Authenticated Encryption (OAE) combines privacy with data integrity and is on-line computable. Most block cipher-based schemes for Authenticated Encryption can be run on-line and are provably secure against *nonce-respecting* adversaries. But they fail badly for more general adversaries. This is not a theoretical observation only – in practice, the reuse of nonces is a frequent issue[1].
In recent years, cryptographers developed *misuse resistant* schemes for Authenticated Encryption. These guarantee excellent security even against general adversaries which are allowed to reuse nonces. Their disadvantage is that encryption can be performed in an off-line way, only. This paper introduces a nw family of OAE schemes –called McOE– dealing both with nonce-respecting *and* with general adversaries. Furthermore, we present three family members, *i.e.,* McOE-X, McOE-D, and McOE-G. All of these members are based on a 'simple' block cipher. In contrast to all other OAE schemes known so far, they provably guarantee reasonable security against general adversaries as well as standard security against nonce-respecting adversaries.

Keywords: authenticated encryption, online encryption, provable security, misuse resistant

1 Introduction

On-Line Authenticated Encryption (OAE). Application software often requires a network channel that guarantees the privacy and authenticity of data being communicated between two parties. Cryptographic schemes able to meet both of these goals are commonly referred to as Authenticated Encryption (AE) schemes.

The ISO/IEC 19772:2009 standard for AE [21] defines generic composition (Encrypt-then-MAC [3]) and five dedicated AE schemes: OCB2 [39], SIV [43] (denoted as "Key Wrap" in [21]), CCM [14], EAX [7], and GCM [34]. To integrate an AE-secure channel most seamlessly into a typical software architecture, application developers expect it to encrypt in an *on-line* manner meaning that the i-th ciphertext block can be written before the $(i+1)$-th plaintext block has to be read. A restriction to off-line encryption, where usually the entire plaintext must be known in advance (or read more than once) is an encumbrance to software architects.

Nonces and their reuse. Goldwasser and Micali [18] formalized encryption schemes as stateful or probabilistic, because otherwise important security properties are lost. Rogaway [38, 41, 42] proposed an unified point of view, by always defining a cryptographic scheme as a deterministic algorithm that takes an user supplied nonce (a *number used once*). So the application programmer – and not the encryption scheme – is responsible for flipping coins or maintaining state. This reflects cryptographic practice since the algorithm itself is often implemented by a multi-purpose cryptographic library which is more or less application-agnostic.

* A shortened version of this paper appears in the proceedings of Fast Software Encryption (FSE) 2012.

[1] A prominent example is the PlayStation 3 'jailbreak' [20], where application developers used a constant that was actually supposed to be a nonce for a digital signature scheme.

secure ...	against nonce-respecting adversaries	ag. nonce-reusing adversaries
on-line	CCFB[33] CHM[22] CIP[23] CWC[29] EAX[7] GCM[34] IACBC[26] IAPM[26] **McOE-D** **McOE-G** **McOE-X** OCB1-3 [42, 39, 30] RPC[11] TAE[31] XCBC[17]	**McOE-D** (this paper) **McOE-G** (this paper) **McOE-X** (this paper)
off-line	BTM[24] CCM[14] HBS[25] SIV[43] SSH-CTR[37]	BTM[24] HBS[25] SIV[43]

Table 1. Classification of provably secure block cipher-based AE Schemes. CCM and SSH-CTR are considered off-line because encryption requires prior knowledge of the message length. Note that the family of McOE schemes, because of being on-line, satisfies a slightly weaker security definition against nonce-reusing adversaries than SIV, HBS, and BTM.

In theory, the concept of a nonce is simple. In practice, it is challenging to ensure that a nonce is *never* reused. Flawed implementations of nonces are ubiquitous [10, 20, 28, 46, 47]. Apart from implementation failures, there are fundamental reasons why software developers cannot always prevent nonce-reuse. A persistently stored counter, which is increased and written back each time a new nonce is needed, may be reseted by a backup – usually after some previous data loss. Similarly, the internal and persistent state of an application may be duplicated when a virtual machine is cloned, etc.

Related Work and Our Contribution. We aim to achieve *both simultaneously*: security against nonce-reusing adversaries (sometimes also called nonce-misusing adversaries) *and* support for on-line-encryption in terms of an AE scheme. Apart from generic composition (Encrypt-then-Mac, EtM), none of the ISO/IEC 19772:2009 schemes – in fact, no previously published AE scheme at all – achieves both of these goals, cf. Table 1. In this table, we classify a vast variety of provably secure block cipher-based AE scheme with respect to their on-line-ability and against which adversaries (nonce-respecting versus -reusing) they are proven secure.

Since EtM is not a concrete scheme but merely a generic construction technique, there are some challenges left in order to make it full on-line secure: First, an appropriate on-line cipher has to be chosen. Second, a suitable, on-line computable, secure, and deterministic MAC must be selected. And, third, the EtM scheme requires at least two *independent* keys to be secure. Since two schemes are used in parallel, it is likely to squander resources in terms of run time and – important for hardware designers – in terms of space. Since EtM first has to be turned into an OAE scheme by making the appropriate choices, we do not include it in our analysis.

As it turned out, we actually found nonce-reuse attacks for *all* of those schemes, cf. Table 2 and Appendix A. In this paper we present a new family of on-line authenticated encryption schemes called McOE. The general structure is based on the Tweak Chain Hash (TCH) construction from [31] which is adepted from the Matyas-Meyer-Oseas (MMO) construction [35]. We introduce three members of the McOE family – called McOE-X, McOE-D, McOE-G. Each of them is able to fill the gap in the upper-right of Table 1. We argue that closing this gap is both practically relevant and theoretically interesting.

Initial Value (IV) based AE schemes which provide security against repeated IV's have been addressed by Rogaway and Shrimpton in [43]. Furthermore, they shaped the notion of "misuse resistance" and proposing SIV as a solution. SIV and related schemes (HBS [25] and BTM [24]) actually provide excellent security against nonce-reusing adversaries, though there are other potential misuse cases, cf. Appendix A.3. Their main disadvantage is that they are inherently off-line: For encryption, one must either keep the entire plaintext in memory, or read the plaintext twice.

Ideally, an adversary seeing the encryptions of two (equal-length) plaintexts P_1 and P_2 cannot even decide if $P_1 = P_2$ or not. When using a nonce more than once, deciding about $P_1 = P_2$ is

	privacy attack workload	authenticity attack workload
CCFB [33]	$O(1)$	$O(1)$
CCM [14]	$O(1)$	$\ll 2^{(n/2)}$ [15]
CHM [22]	$O(1)$	$O(1)$
CIP [23]	$O(1)$	$O(1)$
CWC [29]	$O(1)$	$O(1)$
EAX [7]	$O(1)$	$O(1)$
GCM [34]	$O(1)$	$O(1)$
IACBC [26]	$O(1)$	$O(1)$

	privacy attack workload	authenticity attack workload
IAPM [26]	$O(1)$	$O(1)$
OCB1 [42]	$O(1)$	$O(1)$
OCB2 [39]	$O(1)$	$O(1)$
OCB3 [30]	$O(1)$	$O(1)$
RPC [11]	$O(1)$	$O(1)$
TAE [?]	$O(1)$	$O(1)$
XCBC [17]	$O(2^{n/4})$	?

Table 2. Overview of our **nonce-reuse** attacks on published AE schemes, excluding SIV, HBS and BTM, which have been explicitly designed to resist nonce-reuse. Almost all attacks achieve an advantage close to 1. The "attack workload" covers the computational effort, amount of needed memory as well as the time complexity. Details are given in Appendix A.

easy. SIV and its relatives ensure that nothing else is feasible for nonce-reusing adversaries. In the case of on-line encryption, where the first few bits of the encryption of a lengthy message must not depend on the last few bits of that message, there is unavoidably something beyond $P_1 = P_2$. The adversary can compare any two ciphertexts for their longest common prefix, and then conclude about common prefixes of the secret plaintexts. Our notion of *misuse resistance* means that this is all the adversary can gain. Even in the case of a nonce-reuse, the adversary

1. cannot do anything beyond determining the length of common plaintext prefixes and
2. the scheme still provides the usual level of authenticity for AE (INT-CTXT).

The first property is common for on-line ciphers/permutations (OPRP) [1]. Recently, [45] studies the design of on-line ciphers from tweakable block ciphers bearing some similarities to our approach, especially to TC3. In contrast to the McOE family, the constructions from [45] provide no authentication. The McOE schemes are, *e.g.,* based on a normal block cipher *or* a tweakable block cipher.

Design Principles for AE Schemes. The question how to provide authenticated encryption (without stating that name) when given a secure on-line cipher is studied in [2], the revised and full version of [1]. The first idea in [2] only provides security if all messages are of the same length. The second idea repairs that by prepending the message's length to the message, at the cost of being off-line, since the message length must be known at the beginning of the encryption process. The third idea is to prepend and append a random W to a message M and then to perform the on-line encryption of $(W||M||W)$. This looks promising, but the same W is used for two different purposes, putting different constraints on the generation of W. For privacy, it suffices that W behaves like a nonce, not requiring secrecy or unpredictability. Even if W is not a nonce, but the same W is used for the encryption of several messages, all the adversary can determine are the lengths of common plaintexts prefixes, as we required for nonce-reuse. On the other hand, authenticity actually assumes a *secret or unpredictable* W, rather than a nonce. If the adversary can guess W before choosing a message, she asks for the authenticated encryption of $(M||W)$. Then she can predict the authenticated encryption of M without actually asking for it.

The McOE family replaces the "random" W by a proper nonce and the *key-dependent* tag computation value τ, performing a nonce-dependent on-line encryption of $(M||\tau)$. The encryption can also depend on some associated data, which turns McOE into a family of schemes for OAEAD *(On-Line Authenticated Encryption with Associated Data).*

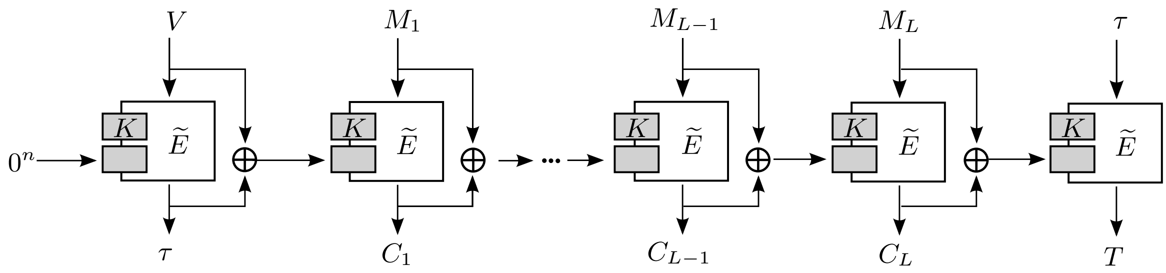

Fig. 1. The generic McOE construction, where $\widetilde{E}$ denotes a tweakable block cipher.

Roadmap. In Section 2 we describe the basic design principle of McOE. Then we present the members of the McOE family. Furthermore, we introduce concret instances of those family, and provide performance data when instantiated with either AES-128 or Threefish-512 as the underlying block cipher. Section 3 deals with general notions and definitions, and Section 4 defines the security of OAE. The main result of the paper, the security proof of the generic McOE scheme and its analysis is presented in Section 5. In Sections 6, 7, and 8 we show the security of McOE-X, McOE-D, and McOE-G, respectively. The discussion in Section 9 concludes the paper. The appendix deals with misuse attacks against published AE schemes, and provides some proof supplements.

2 Practical On-Line Authenticated Encryption using AES and Threefish

2.1 Generic Construction of the McOE Family (without Tag-splitting)

Our design goal was to build a misuse-resistent on-line authenticated encryption scheme, which follows the on-line permutation approach discussed by Bellare *et al.* in [1]. Therefore, our generic McOE structure (see Figure 1) is based on TC3, which is an on-line encryption scheme presented by Rogaway and Zhan in [45]. Like TC3, our scheme is based on a tweakable block cipher – called $\widetilde{E}_K$ – but is stateful regarding to the usage of a nonce. Additionally, we expanded it to a full-fledged authenticated encryption scheme with an additional effort of only two tweakable block cipher calls.

We also introduce the *tag-splitting* (TS) method for processing messages whose length is not a multiple of the block length. Without TS, we would have to pad such messages and then encrypt the padded messages – resulting in an expanded ciphertext. TS is similar to a well-known length preserving method called ciphertext stealing (CTS), *e.g.,* [13]. Note, that CTS requires to process the last block before the last but one, which is not possible for McOE.

The encryption and decryption of the generic construction of McOE without TS can be described by the following two algorithms.

Definition 1 (Generic McOE Scheme without Tag-Splitting). *Let $\Pi = \{\mathcal{K}, \mathcal{E}, \mathcal{D}\}$ be an authenticated encryption scheme, where $\mathcal{K}$ denotes the key derivation function, $\mathcal{E}$ the encryption function, and $\mathcal{D}$ the decryption function. Let $\widetilde{E} \in Block(n, n, n)$ be a tweakable block cipher. Furthermore, let H be the header with $H \in D_n^{L_H}$, M be the message with $M \in D_n^L$ for some integer L, T be the authentication tag with $T \in D_n$, and C be the ciphertext with $C \in D_n^L$. Then $\mathcal{E}$ and $\mathcal{D}$ of the* McOE *family are given by the algorithms* ***EncryptAuthenticate*** *and* ***DecryptAuthenticate****, respectively. Here, the encryption function takes a header H and a message M returning a ciphertext C and a tag T. The decryption function takes a header H, a ciphertext C and a tag T and returns either a plaintext M or the fail symbol $\perp$.*

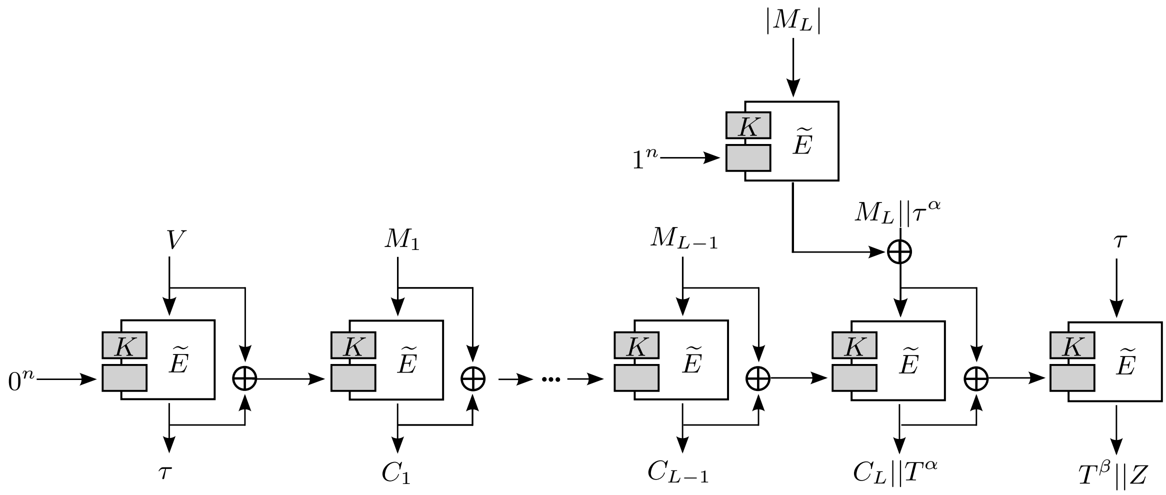

Fig. 2. The generic McOE construction, where $\widetilde{E}$ denotes a tweakable block cipher. For simple reference, we denote T^α as the $(n - |C_L|)$-bit string $T[0...n - |C_L| - 1]$ and T^β as the $|C_L|$-bit string $T[n - |C_L|, ..., n]$. Additionally, we denote the corresponding strings $\tau^\alpha = \tau[0, \ldots, |C_L| - 1]$ and $\tau^\beta = \tau[|C_L|, \ldots, n - 1]$, respectively.

EncryptAuthenticate(H, M)
1. $U \leftarrow 0^n$
2. ***for*** $i = 1, \ldots, L_H - 1$ ***do***
$\quad U \leftarrow \widetilde{E}_K(U, H_i) \oplus H_i$
3. $\tau \leftarrow \widetilde{E}_K(U, H_{L_H})$
4. $U \leftarrow \tau \oplus H_{L_H}$
5. ***for*** $i = 1, \ldots, L$ ***loop***
$\quad C_i \leftarrow \widetilde{E}_K(U, M_i) \oplus M_i$
$\quad U \leftarrow M_i \oplus C_i$
6. $T \leftarrow \widetilde{E}_K(U, \tau)$
7. ***return*** $(C_1, \ldots, C_L, T)$

DecryptAuthenticate(H, C, T)
1. $U \leftarrow 0^n$
2. ***for*** $i = 1, \ldots, L_H - 1$ ***do***
$\quad U \leftarrow \widetilde{E}_K(U, H_i) \oplus H_i$
3. $\tau \leftarrow \widetilde{E}_K(U, H_{L_H})$
4. $U \leftarrow \tau \oplus H_{L_H}$
5. ***for*** $i = 1, \ldots, L$ ***loop***
$\quad M_i \leftarrow \widetilde{E}_K^{-1}(U, C_i)$
$\quad U \leftarrow M_i \oplus C_i$
6. ***if*** $T = \widetilde{E}_K(U, \tau)$ ***then***
$\quad$ ***return*** $(M_1, \ldots, M_L)$
$\quad$ ***else return*** $\perp$

In the case when the header or the message length is not a multiple of the block size n, we recommend to use the secure 10*-padding. Furthermore, the header has to consist of at least one block, since the tag computation value τ depends on it. Hence, the whole header can be seen as a nonce. To fulfill the requirement of length preserving encryption, we introduce the generic McOE scheme using the TS method in the next section.

2.2 Generic Construction of the McOE Family (Tag-splitting)

In Figure 2 you can see the generic McOE scheme when using Tag-splitting to provide length preserving. Both the encryption and decryption process can be seen in the following pseudocode. Note that the additional effort to generate the tag – in comparison to McOE without TS – is given by only one block cipher invocation (cf. line 7 and 9 in **EncryptAuthenticateSplitTag** and **DecryptAuthenticateSplitTag**, respectively).

Definition 2 (Generic McOE Scheme with Tag-Splitting). *Let $\Pi = \{\mathcal{K}, \mathcal{E}, \mathcal{D}\}$ be an authenticated encryption scheme, where $\mathcal{K}$ denotes the key derivation function, $\mathcal{E}$ the encryption function, and $\mathcal{D}$ the decryption function. Let $\widetilde{E} \in Block(n, n, n)$ be a tweakable block cipher.*

*Furthermore, let H be the header with $H \in D_n^{L_H}$, M be the message with $M \in D_n^L||\{0,1\}^{l^*}$ for some integers L and l^*, $0 < l^* < n$, T be the authentication tag with $T \in D_n$, and C be the ciphertext with $C \in D_n^L||\{0,1\}^{l^*}$. Then, the tag-splitting variants of $\mathcal{E}$ and $\mathcal{D}$ are given by the algorithms **EncryptAuthenticateSplitTag** and **DecryptAuthenticateSplitTag**, respectively. Here, the encryption function takes a header H and a message M returning a ciphertext C and a tag T. The decryption function takes a header H, a ciphertext C and a tag T and returns either a plaintext M or the fail symbol $\perp$.*

EncryptAuthenticateSplitTag(H, M)

1. $U \leftarrow 0^n$
2. ***for*** $i = 1, \ldots, L_H - 1$ ***do***
   $U \leftarrow \widetilde{E}_K(U, H_i) \oplus H_i$
3. $\tau \leftarrow \widetilde{E}_K(U, H_{L_H})$
4. $U \leftarrow \tau \oplus H_{L_H}$
5. ***for*** $i = 1, \ldots, L - 1$ ***loop***
   $C_i \leftarrow \widetilde{E}_K(U, M_i) \oplus M_i$
   $U \leftarrow M_i \oplus C_i$
6. $M^* \leftarrow (M_L||\tau[0 \ldots n - l^* - 1])$
7. $M^* \leftarrow M^* \oplus \widetilde{E}_K(1^n, |M_L|)$
8. $C^* \leftarrow \widetilde{E}_K(U, M^*)$
9. *Parse* $C_L||T[0 \ldots n - l^* - 1] \leftarrow C^*$
10. $U \leftarrow M^* \oplus C^*$
11. $C^{**} \leftarrow \widetilde{E}_K(U, \tau)$
12. $T[n - l^* \ldots n - 1] \leftarrow C^{**}[0 \ldots l^* - 1]$
13. ***return*** $(C_1, \ldots, C_{L-1}, C_L^*, T)$

DecryptAuthenticateSplitTag(H, C, T)

1. $U \leftarrow 0^n$
2. ***for*** $i = 1, \ldots, L_H - 1$ ***do***
   $U \leftarrow \widetilde{E}_K(U, H_i) \oplus H_i$
3. $\tau \leftarrow \widetilde{E}_K(U, H_{L_H})$
4. $U \leftarrow \tau \oplus H_{L_H}$
5. ***for*** $i = 1, \ldots, L - 1$ ***loop***
   $M_i \leftarrow \widetilde{E}_K^{-1}(U, C_i)$
   $U \leftarrow M_i \oplus C_i$
6. $C^* \leftarrow C_L||T[0 \ldots n - l^* - 1]$
7. $M^* \leftarrow \widetilde{E}_K^{-1}(U, C^*)$
8. $U \leftarrow M^* \oplus C^*$
9. $M^* \leftarrow M^* \oplus \widetilde{E}_K(1^n, |C_L|)$
10. *Parse* $M_L||\tau'[0 \ldots n - l^* - 1] \leftarrow M^*$
11. $T' = \widetilde{E}_K(U, \tau)$
12. ***if*** $\tau'[0 \ldots n - l^* - 1] = \tau[0 \ldots n - l^* - 1]$
    and $T'[0 \ldots l^* - 1] = T[n - l^* \ldots n - 1]$
    then return $(M_1, \ldots, M_L)$ ***else return*** $\perp$

Both schemes, with and without Tag-splitting, are secure in the common CCA setting assuming a nonce respecting adversary. In addition, they guarantee a certain amount of security in the nonce-misuse scenario, *i.e.,* indistinguishability from an on-line permutation and secure against existential forgery attacks.

2.3 McOE-X

The first instance presented in this paper is called McOE-X, where the 'X' indicates the way of handling the tweak. This scheme uses an ordinary block cipher which is converted to a tweakable block cipher by XORing the tweak (*i.e.,* the chaining value) to the key K. A depiction of this instance is given in Figure 3.

The none tag-splitting and tag-splitting modes of McOE-X can be described by the algorithms introduced in Section 2.1 (see Definition 1 and 2) where the tweakable block cipher $\widetilde{E}_K$ is defined by

$$\widetilde{E}_K(U, M) := E_{K \oplus U}(M),$$

where E_K is a common block cipher, *e.g.,* AES, Serpent, or MARS. For performance testing we have implemented McOE-X using the block ciphers AES-128 and Threefish-512, resulting in the two practical instances McOE-X-AES and McOE-X-Threefish. Both implementations are easily extended to smoothly handle associated data, *i.e.,* data that is not encrypted but only authenticated. The security proofs considering associated data are given in Section 6.

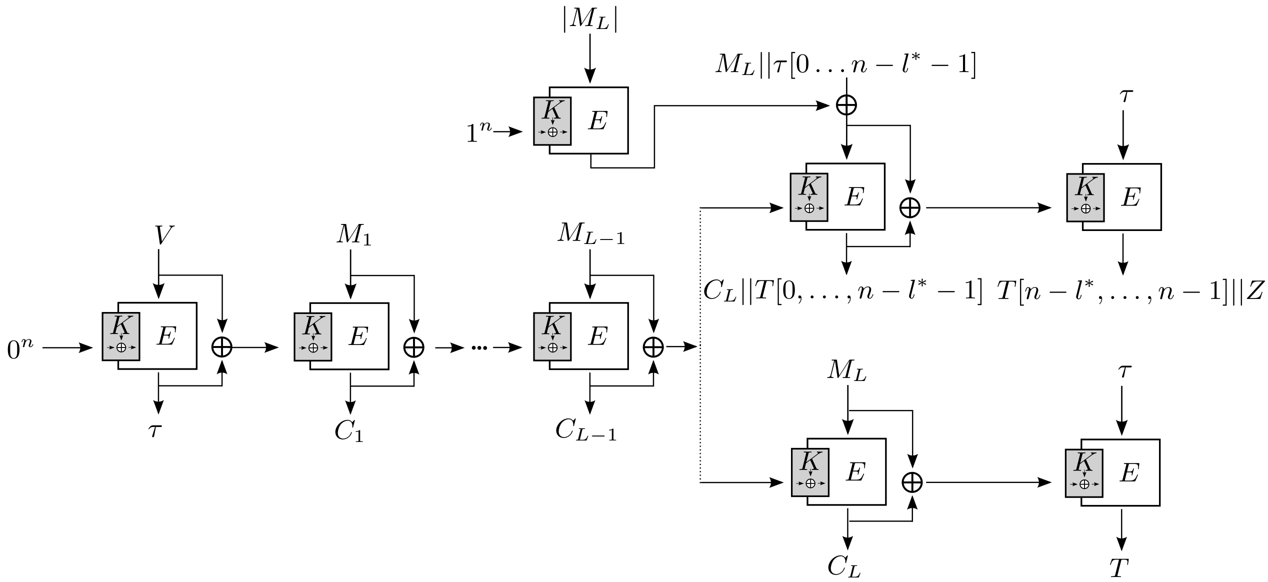

Fig. 3. The McOE-X encryption process. In case that the message length is not a multiple of the block size, McOE-X performs tag-splitting (upper variant). Otherwise, the tag can be computed without splitting (lower variant). The key used for the block cipher E is computed by the injective function $K \oplus U$ which is given the secret key K and the chaining value input U. The tag returned is the n-bit value T. The $n - l$-bit value Z is discarded. The decryption process works in a similar way from 'left to right' only the block cipher component E is replaced by its counterpart E^{-1} apart from one exception: the first call computes τ.

The choice of Threefish-512 is based on two facts. First, it contains a really agile key scheduler, since it is optimized for hashing messages in the MMO (Matyas-Meyer-Oseas) mode. Second, it processes message blocks of size 512 bit, which results in less frequent incovations of the block cipher E_K.

Remark. For this instance of the McOE family we do need related key resistance for the block cipher E since the adversary can 'partially control' some relations among keys used in the computation. We need this requirement only for McOE-X and not for the two instances introduced in the next sections.

2.4 McOE-D

In this section we present another member of the McOE family – called McOE-D. This scheme invokes the block cipher E twice for processing one message block (see Figure 4). The tweakable block cipher $\widetilde{E}_K$ is defined as follows

$$\widetilde{E}_K(U, M) := E_K(E_K(M) \oplus U),$$

where E_K denotes a common block cipher, M the message, and U (chaining value) the tweak. To get rid of the key relation issue we used the double invocation technique (*i.e.,* the block cipher E_K is called twice) introduced by Liskov *et al.* in [31]. This implies that the key scheduler is only applied at the beginning and not for every message block as in McOE-X. So the additional effort compared to McOE-X is only the difference between the computation effort of the key scheduler and a block cipher call.

For this member of the McOE family we also present a version realizing the *tag-splitting* approach, which was introduced before (see Section 6).

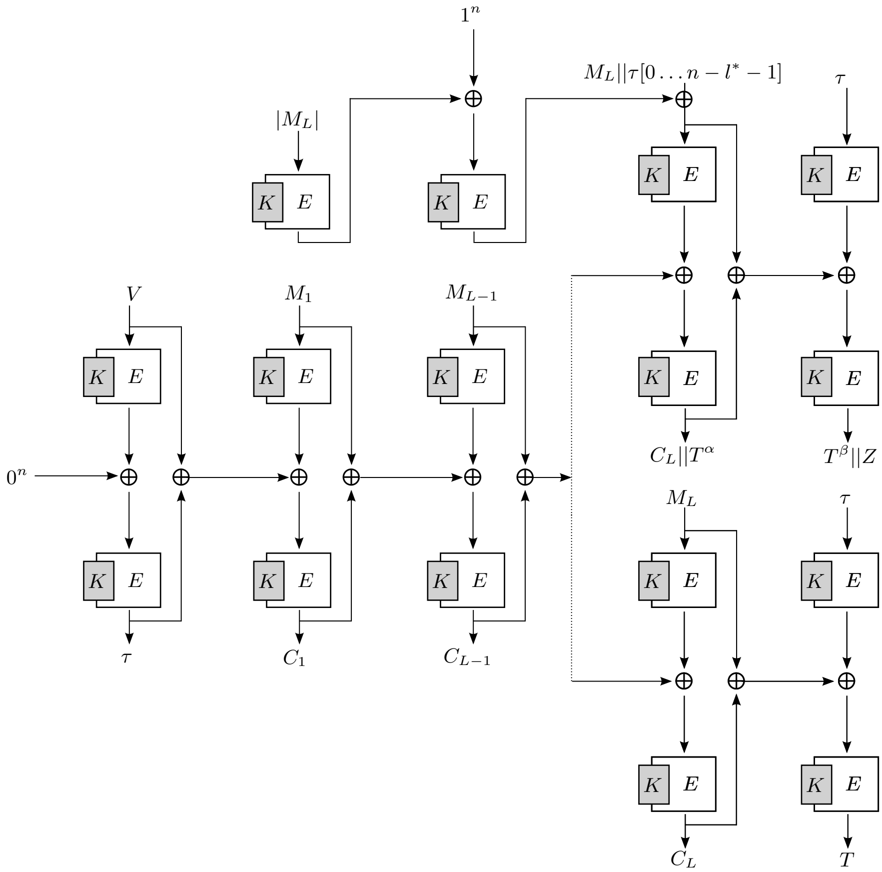

Fig. 4. The McOE-D encryption process. In case that the message length is not a multiple of the block size, McOE-D performs tag-splitting (upper variant), where T^α denotes $T[0, \ldots, n-l^*-1]$ and T^β denotes $T[n-l^*, \ldots, n-1]$. Else, the tag can be computed without splitting (lower variant). The key used for the block cipher E is the same in every encryption. Hence, it is constant and can be precomputed. The tag returned is the n-bit value T. The $n-l$-bit value Z is discarded. The decryption process works in a similar way from 'left to right' only the block cipher component E is replaced by its counterpart E^{-1} apart from one exception: the first call computes τ.

2.5 McOE-G

The third and last member of the McOE family presented in this paper is given by McOE-G. This version updates the chaining value by applying an almost XOR-universal hash function to the XOR result of the previous message block and ciphertext block (see Figure 5). In our practical implementation, we use the Galois-Field multiplication for H, *i.e.,* the key K_2 is multiplied with the chaining value over $GF(2^{128})$ defined by the low weight irreducibel polynomial $g(x) = x^{128} + x^7 + x^2 + x + 1$ as used in OCB [42] and GCM [34].

The tweakable block cipher $\widetilde{E}_K$ is then defined as follows

$$\widetilde{E}_K(U, M) := E_{K_1}(M \oplus H_{K_2}(U)) \oplus H_{K_2}(U),$$

where E_K denotes a common block cipher, M the message, and U (chaining value) the tweak. The key K is denoted by the concatenation of K_1 and K_2, *i.e.,* $K = K_1||K_2$.

For this member of the McOE family we also present a version realizing the *tag-splitting* approach, which was introduced before (see Section 6). Here, an additional key K_2 is requiered for the XOR-universal hash function H_{K_2} as shown in Figure 5, respectively.

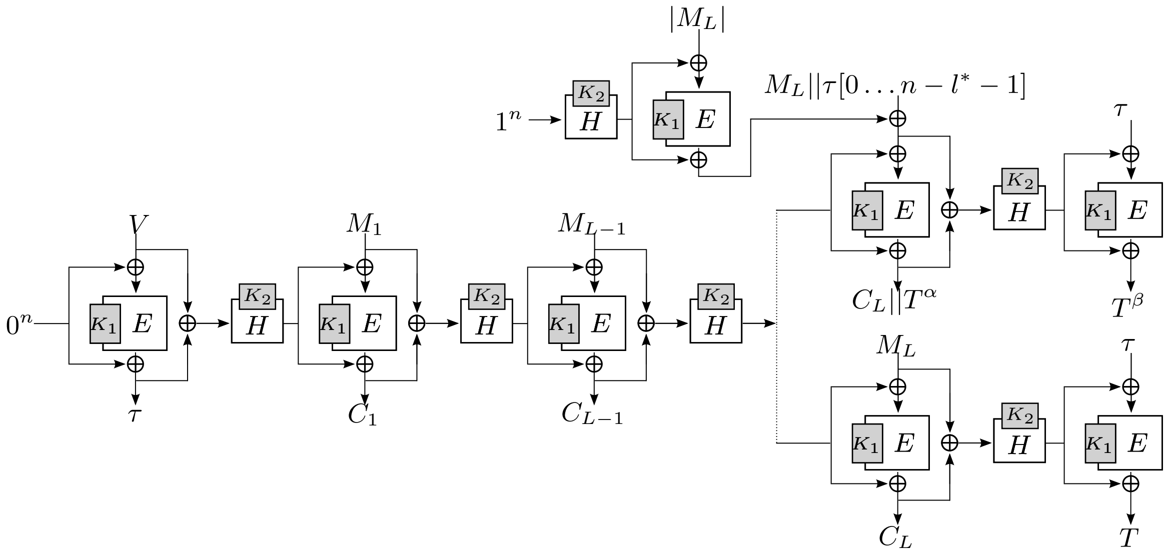

Fig. 5. The McOE-G encryption process. In case that the message length is not a multiple of the block size, McOE-G performs tag-splitting (upper variant), where T^α denotes $T[0, \ldots, n - l^* - 1]$ and T^β denotes $T[n - l^*, \ldots, n - 1]$. Else, the tag can be computed without splitting (lower variant). The key used for the block cipher E is the same for every encryption. Hence, it can be precomputed. The tag returned is the n-bit value T. The $n - l$-bit value Z is discarded. The decryption process works in a similar way from 'left to right' only the block cipher component E is replaced by its counterpart E^{-1} apart from one exception: the first call computes τ.

The McOE-G scheme can be easily extended to smoothly handle associated data, *i.e.,* data that is not encrypted but only authenticated. The security proofs considering associated data are given in Section 8.

2.6 Benchmarking

This section is about measuring the performance of all three presented members of the McOE family. The reference values are given by the CBC encryption scheme. Note, that the implementation of the CBC mode does not contain authentication. The results of our *naive* implementation based on common reference code are illustrated in Table 3.

Block cipher	**Impl.**	**Message length in Bytes**									
		64	128	256	512	1024	2048	4096	8192	16384	32768
McOE-X-AES	software	31.2	26.3	23.9	22.7	22	21.7	21.6	21.5	21.5	21.5
McOE-X-AES	AES-NI	14.2	12.2	11.2	10.7	10.5	10.4	10.4	10.3	10.3	10.3
McOE-X-Threefish	software	19.5	13.1	9.9	8.3	7.5	7.1	6.9	6.8	6.8	6.7
McOE-D-AES	software	40.1	33	29.4	27.6	26.7	26.3	26.1	25.9	25.9	25.9
McOE-D-AES	AES-NI	11.6	9.9	8.3	7.2	6.7	6.4	6.3	6.3	6.2	6.2
McOE-G-AES	software	33	27.9	25.4	24.1	23.5	23.2	23	22.9	22.8	22.8
McOE-G-AES	GF-NI/AES-NI	12.5	10.6	9.7	9.3	9	8.9	8.9	8.8	8.8	8.8
AES-CBC encryption	software	38.3	35.9	13.5	13.3	13.2	13.2	13.1	13.1	13.1	13.1
AES-CBC encryption	AES-NI	4	3.7	3.6	3.5	3.5	3.5	3.5	3.5	3.5	3.5

Table 3. Performance values (cycles-per-byte, single core), measured on a Core i5 540M for AES-128 and Threefish-512. McOE-X is the main contribution in the current paper, McOE-D invokes the underlying block cipher twice and McOE-G uses Galois field arithmetic. For a comparsion, we also provide the performance of unauthenticated AES-CBC. The AES software implementation is based on Gladman [16], whereas the hardware implementation is based on the Intel AES-NI Sample Library[12]. The Threefish implementation is based on the NIST/SHA-3 reference source as provided by the Skein authors [36]. Finally, the implementation of Galois field NI multiplication (GF-NI) is based on the example-code from [19].

3 On-Line Authenticated Encryption and Related Notions

3.1 Definitions

Length of Longest Common Prefix ($\mathbf{LLCP}_n$). The length of a string $x \in \{0,1\}^n$ is denoted by $|x| := n$. For integers $n, \ell, d \geq 1$, set $D_n^d = (\{0,1\}^n)^d$, and $D_n^* := \bigcup_{d\geq 0} D_n^d$, and $D_{\ell,n} = \bigcup_{0\leq d\leq \ell} D_n^d$. Note that D_n^0 only contains the empty string. For $M \in D_n^d$; we write $M = (M_1, \ldots, M_d)$ with $M_1, \ldots, M_d \in D_n$. For $P, R \in D_n^*$, say, $P \in D_n^p$ and $R \in D_n^r$, we define the *length of the longest common n-prefix* of P and R as

$$\mathrm{LLCP}_n(P, R) = \max_i \{P_1 = R_1, \ldots, P_i = R_i\}.$$

For a non-empty set $\mathcal{Q}$ of strings in D_n^* we define $\mathrm{LLCP}_n(\mathcal{Q}, P)$ as $\max_{q\in\mathcal{Q}}\{\mathrm{LLCP}_n(q, P)\}$. For example, if $P \in \mathcal{Q}$, then $\mathrm{LLCP}_n(\mathcal{Q}, P) = |P|/n$.

For convenience, we introduce a notation for a *restriction on a set.* Let $\mathcal{Q} = \{0,1\}^a \times \{0,1\}^b \times \{0,1\}^c$, then we denote $\mathcal{Q}_{|b,c} = \{(B, C) \mid \exists A : (A, B, C) \in \mathcal{Q}\}$ as the restriction of $\mathcal{Q}$ to B and C. This generalizes in the obvious way.

Game Based Proofs. Most of the proofs in this paper use the concept of *game playing proofs.* The presented games in this paper are written in a language heavily based on $\mathcal{L}$, that was introduced by Bellare and Rogaway in [5]. A game has three kinds of functions: An initialization function `Initialize`, a finalization function `Finalize`, and oracle functions. Any adversary A that is playing a game calls at first the `Initialize` function. In the following, A then makes some oracle queries and finally it ends the game by invoking `Finalize`. For adversaries, a function of a game is a black box. They have no access to any local or global variable of any game. An adversary wins the game if and only if `Finalize` returns `true`. We denote $\Pr[A^G \Rightarrow 1]$ as the probability that the adversary wins the game G.

Note, in this paper we usually use a three digit line number which follows the notation of Bellare and Rogaway where the first digit denotes the Game, *e.g.,* 444 denotes the 44-th line of Game G_4.

3.2 Block Ciphers and On-Line Permutations

Block Cipher. A (k, n) block cipher is a keyed family of permutations consisting of two paired algorithms $E : D_k \times D_n \to D_n$ and $E^{-1} : D_k \times D_n \to D_n$, accepting a k-bit key and an input from D_n for some $k, n > 0$. For $n > 0$, $Block(k, n)$ is the set of all (k, n) block ciphers. For any $E \in Block(k, n)$ and a fixed key $K \in D_k$, the decryption $E_K^{-1}(Y) := E^{-1}(K, Y)$ is the inverse function of enryption $E_K(X) := E(K, X)$, so that $E_K^{-1}(E_K(X)) = X$ holds for any $X \in D_n$.

We follow the usual convention to write oracles, that are provided to an algorithm, as superscripts. We define the related key PRP-security of a block cipher E by the success probability of an adversary trying to differentiate between the block cipher and a random permutation.

Definition 3. *Let $E \in Block(k, n)$ and denote by E^{-1} the corresponding inverse. Let $\varphi : D_k \times D_n \to D_k$. A fixed related key adversary A has access to an E oracle with two parameters such that she can query either $E_{\varphi(K,\cdot)}(\cdot)$ or $E^{-1}_{\varphi(K,\cdot)}(\cdot)$.*

Let $\textsc{Perm}(n, n)$ *be the set of n-bit permutations such that the first parameter models the permutation and the second parameter the value that is to be permuted,* i.e., *for $\pi \in \textsc{Perm}(n, n)$ it holds that $\pi(Z, \cdot)$ is a random permutation for any given value of Z.*

```
1 OPerm(V, M)
2     (j, p) ← LLCP*_n(Q_{|V,M}, (V, M));
3     Q ← Q ∪ (V, M, C);
4     for i = 1, ..., p do
5         C_i ← C_i^j;
6     for i = p + 1, ..., |M|/n do
7         C_i ←$ D_n \ D[V||M_1||...||M_{i-1}];
8         D[V||M_1||...||M_{i-1}] ← D[V||M_1||...||M_{i-1}] ∪ C_i;
9     return C;
```

Fig. 6. Lazy sampling implementation of a stateful (n-)on-line permutation. In line 2, the Oracle LLCP_n^* is invoked returning (j,p) where p denotes the length of the prefix n determined via LLCP_n and j denotes the index in $\mathcal{Q}_{|V,M}$. In line 5, we denote by C_i^j the i-th n-bit block of the j-th entry in $Q_{|C}$.

The related-key (RK) advantage [32] of A in breaking E is then defined as

$$\mathbf{Adv}_{E}^{\text{RK-CPA-PRP}}(A) = |\Pr[K \xleftarrow{\$} D_k A^{E_{\varphi(K,\cdot)}(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathit{Perm(n,n)} : A^{\pi(\cdot,\cdot)} \Rightarrow 1]|$$

$$\mathbf{Adv}_{E,E^{-1}}^{\text{RK-CCA-PRP}}(A) = |\Pr[K \xleftarrow{\$} D_k : A^{E_{\varphi(K,\cdot)}(\cdot),E^{-1}_{\varphi(K,\cdot)}(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathit{Perm(n,n)} : A^{\pi(\cdot,\cdot),\pi^{-1}(\cdot,\cdot)} \Rightarrow 1]|.$$

Tweakable Block Cipher The concept of a tweakable block ciphers was introduced by Liskov *et al.* in [31]. The design is based on a common block cipher, which is extended by a so called tweak. A tweakable (k,v,n) block cipher is a family of functions consisting of two paired algorithms $E : D_k \times D_v \times D_n \to D_n$ and $E^{-1} : D_k \times D_v \times D_n \to D_n$, accepting a key $K \in D_k$, a tweak $V \in D_v$, and an input from D_n for some $k,v,n > 0$. $\mathit{Block}(k,v,n)$ is the set of all (k,v,n) tweakable block ciphers. For any $\widetilde{E} \in \mathit{Block}(k,v,n)$ and two fixed values $K \in D_k$ and $V \in D_v$, the decryption $\widetilde{E}_K^{-1}(V,Y) := E^{-1}(K,V,Y)$ is the inverse function of encryption $\widetilde{E}_K(V,X) := \widetilde{E}(K,V,X)$, *i.e.*, $\widetilde{E}_K(K,V,\cdot)$ is a permutation.

Definition 4. *Let $\widetilde{E} \in \mathit{Block}(k,v,n)$ and denote by $\widetilde{E}^{-1}$ the corresponding inverse. A fixed adversary A has access to an $\widetilde{E}$ oracle with three parameters such that she can query either $\widetilde{E}_K(\cdot,\cdot)$ or $\widetilde{E}_K^{-1}(\cdot,\cdot)$.*

Let $\mathrm{TP{\scriptsize ERM}}(v,n)$ be the set of n-bit permutations such that the first parameter models the permutation and the second parameter the value that is to be permuted, i.e., *for $\pi \in \mathrm{TP{\scriptsize ERM}}(v,n)$ it holds that $\pi(Z,\cdot)$ is a random permutation for any given value of Z.*

The advantage for an adversary A to distinguish $\widetilde{E}$ from a randomly chosen permutation from $\mathrm{TP{\scriptsize ERM}}(v,n)$ is defined as

$$\mathbf{Adv}_{\widetilde{E}}^{\text{T-IND-CPA}}(A) = |\Pr[K \xleftarrow{\$} D_k : A^{\widetilde{E}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{TP{\scriptsize ERM}}(v,n) : A^{\pi(\cdot,\cdot)} \Rightarrow 1]|$$

$$\mathbf{Adv}_{\widetilde{E},\widetilde{E}^{-1}}^{\text{T-IND-CCA}}(A) = |\Pr[K \xleftarrow{\$} D_k : A^{\widetilde{E}_K(\cdot,\cdot),\widetilde{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{TP{\scriptsize ERM}}(v,n) : A^{\pi(\cdot,\cdot),\pi^{-1}(\cdot,\cdot)} \Rightarrow 1]|.$$

On-Line Permutation (OPerm). We aim for larger permutations that not only permute single blocks but can handle multiple/variable block messages. Such a permutation, from D_n^* to D_n^*, is (n-)on-line if the i-th block of the output is determined completely by the first i blocks of the input. Let denote $\mathrm{OPerm}_{n,*}$ the set of all on-line permutations from D_n^* to D_n^*. It is easy to extend the definition with a state space D_v. Let $\mathrm{OPerm}_{n,*}^v$ denote the set of all functions from $D_v \times D_n^*$ to D_n^*. Then for each $f \in \mathrm{OPerm}_{n,*}^v$ and $V \in D_v$, the function $f(V,\cdot)$ is an (n-)on-line permutation. Figure 6 illustrates a lazy sampling implementation of $\mathrm{OPerm}_{n,*}^v$.

Next, we introduce the formal definition of a family of (n-)on-line functions which is the basic design principle of the McOE family.

Definition 5. *Let $n, k, v \geq 0$, $K \in D_k, V \in D_v$. A family of functions $F : D_k \times D_v \times D_n^* \to D_n^*$ is (n-)on-line if for any instance of this family determined by K, V, $F(K, V, \cdot)$ is a permutation and there exists for any message $M = (M_1, M_2, \ldots, M_\ell)$ a family of functions $f^i : D_k \times (D_v \times D_n^{i-1}) \times D_n \to D_n$, $i = 1, \ldots, \ell$ such that*

$$\begin{aligned}\Pi(K, V, M) = \;& f_K^1((V, \emptyset), M_1) || f_K^2((V, M_1), M_2) \\ & || \ldots || \\ & f_K^{\ell-1}((V, M_1 || \ldots || M_{\ell-2}), M_{\ell-1}) || f_K^\ell((V, M_1 || \ldots || M_{\ell-1}), M_\ell),\end{aligned}$$

where "$||$" being the concatenation of strings, holds.

An encryption scheme is (n-)on-line if the encryption function is (n-)on-line. A thorough discussion of on-line encryption and its properties can be found in [1].

Proposition 1. *Let F be an (n-)on-line function as defined in Definition 5, then all $f_K^i(V, \cdot)$ are n-bit permutations.*

The proof is similar to Proposition 3.4 of [1].

Let F be a family of (n-)on-line functions. Assume that for each uniform randomly chosen F_K from F, each $f_K^i(V, \cdot)$ is a PRP, then it is easy to see that F_K is indistinguishable from the OPerm oracle as shown in Figure 6. We call such a a family of (n-)on-line functions on-line pseudo random permutations (OPRP).

3.3 Authenticated Encryption (With Associated Data)

An authenticated encryption scheme is a tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Its aim is to provide privacy and data integrity. The key generation function $\mathcal{K}$ takes no input and returns a randomly chosen key K from the key space, *e.g.,* from D_k. The encryption algorithm $\mathcal{E}$ and the decryption algorithm $\mathcal{D}$ are deterministic algorithms that map values from $D_k \times D_n^+ \times D_n^*$ to a string or – if the input is invalid – the value $\perp$. The header H consists either only of the initial value/nonce $V \in D_n$ (if no data is to be authenticated/checked in the encryption/decryption process) or is a combination of V and a value from D_n^*. So $H \subset D_n^+$ in either case. For sake of convenience, we usually write $\mathcal{E}_K^H(M)$ for $\mathcal{E}(K, H, M)$ and $\mathcal{D}_K^H(M)$ for $\mathcal{D}(K, H, M)$, where the message M is chosen from D_n^*, $H \in D_n^+$ and a key from the key space. We require $\mathcal{D}_K^H(\mathcal{E}_K^H(M)) = M$ for any possible K, M, H, and define the tag size for a message $M \in D_n^*$ and header $H \in D_n^+$ as $\textsc{tag}(H, M) := |\mathcal{E}_K^H(M)| - |M|$. We denote an authenticated encryption scheme with the requirement that the initial vector V is only used once in a *nonce based* scheme. Otherwise, we call such a scheme *deterministic*. Similarly, we call an adversary *nonce-respecting* (NR) if no nonce is used twice for any query. Otherwise, the adversary is called *nonce-ignoring* (NI).

4 Security Notions for On-Line Authenticated Encryption

Authenticated (On-Line) Encryption tries to achieve privacy and authenticity at the same time. Therefore we need security notions to handle this twofold goal. For AE, there have been notions and their relations introduced for deterministic [44] and nonce based [3, 4, 27, 38, 42] AE schemes. In order to have one convenient toolset of notions, we adopt the notion of CCA3 security suggested in [44] as a *natural strengthening* of CCA2 security.

We parameterize our definition in order to define different – but closely related – notions by explicitly stating whether we mean an on-line or off-line scheme, $\omega \in \{\textsc{ae}, \textsc{oae}\}$, and stating the adversary behavior as either nonce-respecting or nonce-ignoring, $\nu \in \{\textsc{nr}, \textsc{ni}\}$.

```
Game G_CPA, [G_CCA3]

Initialize(ω, ν)
    b ←$ {0,1};
    if(b=1) then
        K ← 𝒦();

Finalize(d)
    return (b = d);

Encrypt(H, M)
    if (ν = NR and V ∈ B) then
        return ⊥;
    if(b=1) then
        C ← ℰ_K(H, M);
    else
        C ← $^ω(H, M);
    B ← B ∪ {V};
    [Q ← Q ∪ {(H, C)};]
    return C;

[Decrypt(H, C)
    if ((H, C) ∈ Q) then
        return ⊥;
    if(b=1) then
        M ← 𝒟_K(H, C);
    else
        M ← ⊥(H, C);
    return M;]
```

Fig. 7. $G_{\text{CPA}}(\omega,\nu)$ is the $\text{CPA}_{\Pi}^{(\omega,\nu)}$-Game and $G_{\text{CCA3}}(\omega,\nu)$ the $\text{CCA3}_{\Pi}^{(\omega,\nu)}$-Game where $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$. Game G_{CCA3} contains the code in the box while G_{CPA} does not. The oracle $\$^{\text{AE}}(H,M)$ returns a string of length $|M| + \text{TAG}(H,M)$, this string is on-line compatible if $\omega = \text{OAE}$. V denotes the last block of the header representing the nonce/initial value.

Definition 6 (CCA3(ω,ν)). *Let $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$ be an authenticated encryption scheme with header space D_n^+ and message space D_n^*, and fix an adversary A. The advantage of A breaking Π is defined as*

$$\mathbf{Adv}_{\Pi}^{\text{CCA3}(\omega,\nu)}(A) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$^{\omega}(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right]\right|.$$

The adversary's random-bits oracle, $\$^{\text{AE}}(\cdot,\cdot)$ or $\$^{\text{OAE}}(\cdot,\cdot)$, returns on a query with header $H \in D_n^+$ and plaintext $X \in D_n^*$ a random string of length $|\mathcal{E}_K(M)|$ which is either on-line or not, depending on the variable ω. The $\perp(\cdot,\cdot)$ oracle returns $\perp$ on every input. We assume *wlog.* that the adversary A never ask a query which answer is already known. It is easy to see that we can rewrite the term given in Definition 6 as

$$\mathbf{Adv}_{\Pi}^{\text{CCA3}(\omega,\nu)}(A) = \Bigg|\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right] \quad (1)$$

$$+ \Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$^{\omega}(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right]\Bigg|. \quad (2)$$

One can interpret (1) as the advantage that an adversary has on the integrity of the ciphertext and (2) as the advantage that an CPA adversary has on the privacy. Using this decomposition as a motivational starting point, we now define ciphertext integrity and what we mean by a CPA adversary on authenticated encryption schemes. From now on, our definitions are based on the game playing methodology. For example, we can restate Definition 6 using the game G_{CCA3} given in Figure 7 as

$$\mathbf{Adv}_{\Pi}^{\text{CCA3}(\omega,\nu)}(A) = 2\left|\Pr[A^{G_{\text{CCA3}}(\omega,\nu)} \Rightarrow 1] - 0.5\right|.$$

We denote $\mathbf{Adv}_{\Pi}^{\text{CCA3}(\omega,\nu)}(q,\ell,t)$ as the maximum advantage over all CCA3(ω,ν) adversaries run in time at most t, ask a total maximum of q queries to $\mathcal{E}$ and $\mathcal{D}$, and whose total query length is not more than ℓ blocks.

4.1 Privacy and Integrity Notions for Authenticated Encryption Schemes.

Similarly, we define the privacy and integrity of an authenticated (on-line) encryption scheme $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$ with header space D_n^+, message space D_n^* and tag-size function $\text{TAG}(H,M)$ as follows.

Definition 7. *Let $G_{CPA}(\omega,\nu)$ be the $CPA_{\Pi}^{\omega,\nu}$ game given in Figure 7. Fix an adversary A. The advantage of A breaking Π is defined as*

$$\mathbf{Adv}_{\Pi}^{CPA(\omega,\nu)}(A) \le 2\left|\Pr[A^{G_{CPA}(\omega,\nu)} \Rightarrow 1] - 0.5\right|.$$

```
Game G_{INT-CTXT}

Initialize(ν)
    K ← 𝒦();

Finalize()
    return win;

Encrypt(H,M)
    if (ν = NR and V ∈ B) then
        return ⊥;
    C ← ℰ_K(H,M);
    B ← B ∪ {V};
    𝒬 ← 𝒬 ∪ {(H,C)};
    return C;

Verify(H,C)
    M ← 𝒟_K(H,C);
    if ((H,C) ∉ Q and M ≠ ⊥) then
        win ← true;
    return (M ≠ ⊥);
```

Fig. 8. Game $G_{INT-CTXT}(\nu)$ is the INT-CTXT$_{\Pi}^{\omega,\nu}$ game where $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. V denotes the last block of the header representing the nonce/initial value.

.

Definition 8. *Let* $G_{\text{INT-CTXT}}(\nu)$ *be the* INT-CTXT$_{\Pi}^{\nu}$ *game given in Figure 8. Fix an adversary A. The advantage of A breaking Π is defined as*

$$\mathbf{Adv}_{\Pi}^{\text{INT-CTXT}(\nu)}(A) \leq \Pr[A^{G_{\text{INT-CTXT}(\nu)}} \Rightarrow 1].$$

We denote $\mathbf{Adv}_{\Pi}^{CPA(\omega,\nu)}(q,t,\ell)$ and $\mathbf{Adv}_{\Pi}^{\text{INT-CTXT}(\nu)}(q,t,\ell)$ as the maximum advantage over all $CPA(\omega,\nu)$ resp. INT-CTXT(ν) adversaries run in time at most t, ask a total maximum of q queries to $\mathcal{E}$ and $\mathcal{D}$, and whose total query length is not more than ℓ blocks.

4.2 CCA3 is equal to INT-CTXT plus CPA.

We now give a generalization of Theorem 3.2 from Bellare and Namprempre [3]. It simply states the equivalence of a scheme being CCA3 secure and both INT-CTXT and CPA secure (often denoted as IND-CPA secure). These statements hold in the on-line and off-line case.

Theorem 1. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme. Fix $\omega \in \{\text{AE}, \text{OAE}\}$ and $\nu \in \{\text{NR}, \text{NI}\}$. Let A be an* CCA3$(\omega,\nu)_{\Pi}$*-adversary running in time t, making q queries with a total length of at most ℓ blocks. Then there are a* CPA(ω,ν)*-adversary A_p and an* INT-CTXT(ω,ν)*-adversary A_c such that*

$$\mathbf{Adv}_{\Pi}^{\text{CCA3}(\omega,\nu)}(A) \leq \mathbf{Adv}_{\Pi}^{\text{CPA}(\omega,\nu)}(A_p) + \mathbf{Adv}_{\Pi}^{\text{INT-CTXT}(\omega,\nu)}(A_c).$$

Furthermore, A_c and A_p run in time $O(t)$ and both make at most q queries in each case.

The proof is given in Appendix B.

4.3 Relations between PRP and OPRP.

Let us proceed from on-line authenticated encryption schemes in a common case, where we consider an adversary A to be nonce respecting and the regarded scheme to be CCA3$^{\text{AE,NR}}$ secure. For such schemes it is desirable to obtain a certain level of security, even in a misuse scenario. Due to the nature of this scenario an on-line authenticated encryption scheme can *solely* be CCA3$^{\text{OAE,NI}}$ secure. Hence it is of great interest to determine the relation between these two security notions.

Lemma 1. *Let F be an OPRP, introduced in Section 3.2, and A be a nonce respecting adversary. Then*

$$\mathbf{Adv}_{F}^{PRP(nr)}(A) = |\Pr[K \stackrel{\$}{\leftarrow} \mathcal{K} : A^{F_K} \Rightarrow 1] - \Pr[p \stackrel{\$}{\leftarrow} PRP : A^{p} \Rightarrow 1]| = 0.$$

Proof (Sketch). Let denote (V_i, M_i) the i-th encryption query. For each V_i, V_j with $i \neq j$ it holds true that $V_i \neq V_j$, since we assume a nonce respecting adversary. Consequently $F_K(V_i, \cdot)$ and $F_K(V_j, \cdot)$ are two independent PRPs, due to the fact that all $f_k^{\ell}(V||X, \cdot)$ with $X \in D_n^{\ell-1}$ are PRPs. This implies that $F_K(V, .)$ is a PRP. □

This Lemma shows that a CCA3$^{\text{OAE,NI}}$ secure on-line authenticated encryption scheme is also CCA3$^{\text{AE,NR}}$ secure against nonce respecting adversaries.

5 Security of the Generic McOE Scheme

In this section, we analyse the security of the generic McOE scheme. We introduced this scheme by Definition 1 and 2 (cf.Section 2), which also provide the corresponding pseudocode. We show that McOE achieves our two-fold goal, by proofing that it guarantees a certain minimum, well defined security against a nonce-ignoring adversary. More formal, we show that McOE is $\text{CCA3}^{\text{OAE,NI}}$ secure, which implies that McOE is also fully secure against a nonce-respecting adversary, *i.e.,* $\text{CCA3}^{\text{AE,NR}}$ secure (cf. Section 4.3).

5.1 Security Analysis of McOE without Tag-Splitting

We now proceed to show the security of McOE. For this we use the results of Theorem 1 and show the INT-CTXT and RK-CPA-PRP security separately.

Theorem 2. *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be a* McOE *scheme as in Definition 1,* i.e., $\mathcal{K}$ *is the key derivation function,* $\mathcal{E} = \textbf{EncryptAuthenticate}$ *and* $\mathcal{D} = \textbf{DecryptAuthenticate}$. *Then*

$$\mathbf{Adv}_{\Pi}^{\text{CCA3(OAE,NI)}}(q, \ell, t) \leq \frac{3(q+\ell)(q+\ell+1)+4q+3\ell}{2^n-(q+\ell)} + 3\mathbf{Adv}_{\widetilde{E},\widetilde{E}^{-1}}^{\text{T-IND-CCA}}(q+\ell, O(t)).$$

Proof. The proof follows from Theorem 1 together with Lemmas 2 and 3. □

In the follwoing, for the sake of simplification, we provide an upper bound which is much easier to grasp than the original bound, but not as tight as the original bound given in the theorem above.

Corollary 1. *Let assume that* $\ell \geq 7$, $\ell \geq q$ *and the* T-IND-CCA*-advantage is at most* δ *for an adversary which amount of queries is at most* $q+\ell$ *and its running time is* $O(t)$. *Then the following bound holds*

$$\mathbf{Adv}_{\Pi}^{\text{CCA3(OAE,NI)}}(q, \ell, t) \leq \frac{14\ell^2}{2^n-(2\ell)} + \delta.$$

Lemma 2. *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be a* McOE *scheme as in Definition 1. Let* q *be the number of total queries an adversary* A *is allowed to ask and* ℓ *be an integer representing the total length in blocks of the queries to* $\mathcal{E}$ *and* $\mathcal{D}$. *Then,*

$$\mathbf{Adv}_{\Pi}^{\text{INT-CTXT(NI)}}(q, \ell, t) \leq \frac{(q+\ell)(q+\ell+1)}{2^n-(q+\ell)} + \frac{2q+\ell}{2^n-(q+\ell)} + \mathbf{Adv}_{\widetilde{E},\widetilde{E}^{-1}}^{\text{T-IND-CCA}}(q+\ell).$$

Proof. Our bound is derived by game playing arguments. Consider games G_1-G_3 of Figure 9 and a fixed adversary A asking at most q queries with a total length of at most ℓ blocks. The functions **Initialize** and **Finalize** are identical for all games in this proof. Lets denote G_0 as the Game INT-CTXT(NI) as defined in Figure 8. Definition 8 states that

$$\mathbf{Adv}_{\Pi}^{\text{INT-CTXT(NI)}}(A) \leq \Pr[A^{G_0} \Rightarrow 1].$$

In G_1, the encryption and verify placeholders are replaced by their generic McOE counterparts as of Definition 1. Clearly, $\Pr[A^{G_0} \Rightarrow 1] = \Pr[A^{G_1} \Rightarrow 1]$. We now discuss the differences between G_1 and G_2. The set B_1 is initialized with 0^n and then collects all new key-input values U, which are computed during the encryption or verification process (in lines 204, 207, 213, 223, 226, 232 and 237).

Initialize()
  $K \xleftarrow{\$} \mathcal{K}()$;
  $B_1 \leftarrow \{0^n\}$;

Encrypt(H, M) Game G_1
  $L_H \leftarrow |H|/n$; $L \leftarrow |M|/n$;
  $U \leftarrow 0^n$;
  for $i = 1, ..., L_H$ **do**
    $\tau \leftarrow \widetilde{E}_K(U, H_i)$;
    $U \leftarrow H_i \oplus \tau$;
  for $i = 1, ..., L$ **do**
    $C_i \leftarrow \widetilde{E}_K(U, M_i)$;
    $U \leftarrow C_i \oplus M_i$;
  $T \leftarrow \widetilde{E}_K(U, \tau)$;
  $\mathcal{Q} \leftarrow (H, M, C, T)$;
  return $(C_1, \ldots, C_L, T)$;

Encrypt(H, M) Game G_2, $\boxed{G_3}$
  $L_H \leftarrow |H|/n$; $L \leftarrow |M|/n$;
  $B_2 \leftarrow B_2 \cup H$;
  $p \leftarrow \mathrm{LLCP}_n(\mathcal{Q}_{|H,M}, (H, M))$;
  $U \leftarrow 0^n$;
  for $i = 1, \ldots, L_H$ **do**
    $\tau \leftarrow \widetilde{E}_K(U, H_i)$;
    $U \leftarrow H_i \oplus \tau$;
    if $(U \in B_1$ **and** $i > p)$ **then**
      bad $\leftarrow$ **true**; $\boxed{U \xleftarrow{\$} \{0,1\}^n \setminus B_1;}$
    $B_1 \leftarrow B_1 \cup U$;
  for $i = 1, \ldots, L$ **do**
    $C_i \leftarrow \widetilde{E}_K(U, M_i)$;
    $U \leftarrow C_i \oplus M_i$;
    if $(U \in B_1$ **and** $i + L_H > p)$ **then**
      bad $\leftarrow$ **true**; $\boxed{U \xleftarrow{\$} \{0,1\}^n \setminus B_1;}$
    $B_1 \leftarrow B_1 \cup U$;
  $T \leftarrow \widetilde{E}_K(U, \tau)$;
  $\mathcal{Q} \leftarrow (H, M, C, T)$;
  return $(C_1, \ldots, C_L, T)$;

Finalize()
  return win;

Verify(H, C, T) Game G_1
  $L_H \leftarrow |H|/n$; $L \leftarrow |C|/n$;
  $U \leftarrow 0^n$;
  for $i = 1, ..., L_H$ **do**
    $\tau \leftarrow \widetilde{E}_K(U, H_i)$;
    $U \leftarrow H_i \oplus \tau$;
  for $i = 1, ..., L$ **do**
    $M_i \leftarrow \widetilde{E}_K^{-1}(U, C_i)$;
    $U \leftarrow C_i \oplus M_i$;
  if $(T = \widetilde{E}_K(U, \tau)$ **and** $(H, C) \notin \mathcal{Q}_{|H,C})$ **then**
    win $\leftarrow$ **true**;
  $\mathcal{Q} \leftarrow (H, \perp, C, \perp)$;
  return $(T = \widetilde{E}_K(U, \tau))$

Verify(H, C, T) Game G_2, $\boxed{G_3}$
  $L_H \leftarrow |H|/n$; $L \leftarrow |C|/n$;
  $p \leftarrow \mathrm{LLCP}_n(\mathcal{Q}_{|H,M}, (H, M))$;
  $U \leftarrow 0^n$;
  for $i = 1, \ldots, L_H$ **do**
    $\tau \leftarrow \widetilde{E}_K(U, H_i)$;
    $U \leftarrow H_i \oplus \tau$;
    if $(U \in B_1$ **and** $i > p)$ **then**
      bad $\leftarrow$ **true**; $\boxed{U \xleftarrow{\$} \{0,1\}^n \setminus B_1;}$
    $B_1 \leftarrow B_1 \cup U$;
  for $i = 1, \ldots, L - 1$ **do**
    $M_i \leftarrow \widetilde{E}_K^{-1}(U, C_i)$;
    $U \leftarrow C_i \oplus M_i$;
    if $(U \in B_1$ **and** $i + L_H > p)$ **then**
      bad $\leftarrow$ **true**; $\boxed{U \xleftarrow{\$} \{0,1\}^n \setminus B_1;}$
    $B_1 \leftarrow B_1 \cup U$;
  $M_L \leftarrow \widetilde{E}_K^{-1}(U, C_L)$;
  $U \leftarrow C_L \oplus M_L$;
  if $(U \in B_1$ **and** $H \notin B_2)$ **then**
    bad $\leftarrow$ **true**; $\boxed{U \xleftarrow{\$} \{0,1\}^n \setminus B_1;}$
  if $(T = \widetilde{E}_K(U, \tau)$ **and** $(H, C, T) \notin \mathcal{Q}_{|H,C,T})$ **then**
    win $\leftarrow$ **true**;
  $\mathcal{Q} \leftarrow (H, \perp, C, \perp)$;
  $B \leftarrow B \cup U$;
  return $(T = \widetilde{E}_K(U, \tau))$;

Fig. 9. Games G_1-G_3 for the proof of Lemma 2. Game G_3 contains the code in the box while G_2 does not.

In lines 203 and 222, the LLCP$_n$ oracle is inquired. Finally, the variable `bad` is set to `true` if one of the if-conditions in lines 208, 214, 227, 233, or 238 is `true`. *None* of these modifications affect the values returned to the adversary and therefore

$$\Pr[A^{G_1} \Rightarrow 1] = \Pr[A^{G_2} \Rightarrow 1].$$

For our further discussion we require another game G_4 which is explained in more detail later in this proof[2]. It follows that

$$\begin{aligned}
\Pr[A^{G_2} \Rightarrow 1] &= \Pr[A^{G_3} \Rightarrow 1] + |\Pr[A^{G_2} \Rightarrow 1] - \Pr[A^{G_3} \Rightarrow 1| \\
&\leq \Pr[A^{G_3} \Rightarrow 1] + \Pr[A^{G_3} \textit{sets } \texttt{bad}] \\
&\leq \Pr[A^{G_4} \Rightarrow 1] + |\Pr[A^{G_3} \Rightarrow 1] - \Pr[A^{G_4} \Rightarrow 1]| + \Pr[A^{G_3} \textit{sets } \texttt{bad}]. \qquad (3)
\end{aligned}$$

We now proceed to upper bound any of the three terms contained in (3) – in right to left order. The success probability of game G_3 does not differ from the success probability of G_2 unless a chaining value U occurs twice. In this case, the adversary must (1) either have 'found' a collision for $\widetilde{E}_K(X, Y) \oplus Y$, *i.e.,* she stumbles over (X, Y) and (X', Y') such that $\widetilde{E}_K(X, Y) \oplus Y = \widetilde{E}_K(X', Y') \oplus Y'$ or, (2), must have found a preimage of 0^n, which is always the starting point of our chain. Note, the value 0^n is initially stored in the set B_1. In both cases, the variable `bad` would have been set to `true`, and it follows by [9] that

$$\Pr[A^{G_3} \textit{sets } \texttt{bad}] \leq \frac{(q+\ell)(q+\ell+1)}{2^n - (q+\ell)} + \frac{q+\ell}{2^n - (q+\ell)}.$$

We now describe the new game G_4. It is equal to G_3 *except* that the tweakable block cipher $\widetilde{E}$ and its inverse $\widetilde{E}^{-1}$ are replaced by the functions **EncryptBlock** and **DecryptBlock**, which are modeled as a set of pseudo random permutations, where the index is given by the tweak. We assume that they are implemented via lazy sampling. More precisely, the call $\widetilde{E}_K(X, Y)$ is replaced by an invocation of $\textbf{EncryptBlock}_K(X, Y)$ and the call $\widetilde{E}_K^{-1}(X, Y)$ is replaced by an invocation of $\textbf{DecryptBlock}_K(X, Y)$. We now upper bound the difference between G_3 and G_4.

So, by definition of G_4, we have

$$|\Pr[A^{G_3} \Rightarrow 1] - \Pr[A^{G_4} \Rightarrow 1]| \leq \mathbf{Adv}^{\text{T-IND-CCA}}_{\widetilde{E}, \widetilde{E}^{-1}}(q+\ell, O(t)).$$

Finally, we have to upper bound the advantage for the adversary A to win the game G_4. A can only win this game if the condition in line 238 (resp. 438 for game G_4) is `true`. As usual, we assume *wlog.* that A doesn't ask a question if the answer is already known which implies that $(H, C, T) \notin \mathcal{Q}_{|H,C,T}$. For our analysis we distinguish between three cases. So we formally adjust line 240 (*i.e.,* choose as the tag computation operation either $\widetilde{E}$ or $\widetilde{E}^{-1}$) such that we always have enough randomness left for our result.

Case 1: *$H \in B_2$ and $U \in B$.*
Since we already have computed τ in the past, the chance of success is upper bounded by the probability $\Pr[\widetilde{E}_K^{-1}(U, T) = \tau]$ which can be upper bounded by $1/(2^n - (q+\ell))$.

Case 2: *$H \notin B_2$, and $U \notin B_1$.*
Then the tagging operation uses a new tweak and therefore the output of $\widetilde{E}_K(U, \tau)$ is uniformly distributed and the success probability is $\leq 1/2^n$.

Case 3: *$H \in B_2$ and $U \notin B_1$.*
The chance of success is upper bounded by $\Pr[\widetilde{E}_K^{-1}(U, T) = \tau]$ which can be upper bounded by $1/2^n$.

[2] Since the difference is very minor, we do not provide an extra figure.

Note, the 'missing' fourth case has been explicitly excluded by line 240 (resp. 440). Since these three cases are mutually exclusive, we can upper bound the success probability for q queries as

$$\Pr[A^{G_4} \Rightarrow 1] \leq \frac{q}{2^n - (q+\ell)}.$$

Our claim follows by adding up the individual bounds. $\square$

Lemma 3. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a* McOE *scheme as in Definition 1 (i). Let q be the number of total queries an adversary A is allowed to ask and ℓ be an integer representing the total length of the queries to $\mathcal{E}$ and $\mathcal{D}$. Then,*

$$\mathbf{Adv}_{\Pi}^{\text{CPA(AOE,NI)}}(q,\ell,t) \leq\ 2\left(\frac{(q+\ell)(q+\ell+1)}{2^n-(q+\ell)} + \frac{q+\ell}{2^n-(q+\ell)} + \mathbf{Adv}_{\widetilde{E}}^{\text{T-IND-CPA}}(q+\ell)\right).$$

Proof. Our bound is derived by game playing arguments. Consider games G_1 and G_2 of Figure 12. The functions **Initialize** and **Finalize** are identical for any of those games.

At first we investigate the differences between the CPA(AOE, NI) game from Figure 7 and G_1 from Figure 12. In G_1 we have replaced $\mathcal{E}$ by its definition of McOE, and $\$^w$ by an on-line encryption oracle **OnlinePermutation** (line 102) that just models a 'perfect' OPRP, *i.e.,* for two plaintexts with an equal prefix it returns two ciphertexts that also share a prefix of the same length. We again assume this oracle to be implemented by lazy sampling. Then, set B collects all chaining values (lines 113 and 119) in order to intercept the occurrence of two equal chaining values which do lead to two equal tweaks for the encryption of a block.

In line 105, the oracle LLCP_n is invoked returning the length of the longest common prefix of (H, M) and $\mathcal{Q}_{|H,M}$.

Finally, the variable `bad` is set to `true` if (one of) the conditions of lines 111/211 or 117/217 hold. These changes do not affect the success probability of an adversary, because the output of the oracle remains unchanged. More precisely, the distribution of the output does not change. This means that

$$\mathbf{Adv}_{\Pi}^{\text{CPA(AOE, NI)}}(A) = 2 \cdot |\Pr[A^{G_1} \Rightarrow 1] - 0.5|,$$

and therefore, by common game playing arguments – using a new game G_3 described shortly –,

$$\begin{aligned}\Pr[A^{G_1} \Rightarrow 1] &\leq \Pr[A^{G_2} \Rightarrow 1] + |\Pr[A^{G_1} \Rightarrow 1] - \Pr[A^{G_2} \Rightarrow 1]| \\ &\leq \Pr[A^{G_2} \Rightarrow 1] + \Pr[A^{G_2} \textit{sets}\ \texttt{bad}] \\ &\leq \Pr[A^{G_3} \Rightarrow 1] + |\Pr[A^{G_2} \Rightarrow 1] - \Pr[A^{G_3} \Rightarrow 1]| + \Pr[A^{G_2} \textit{sets}\ \texttt{bad}].\end{aligned}$$

The success probability of game G_2 does not differ from the success probability of G_1 unless a chaining value U occurs twice. In this case, the adversary must either have found a collision for $\widetilde{E}_K(X,Y) \oplus Y$, *i.e.,* she has found (X,Y) and (X',Y') such that $\widetilde{E}_K(X,Y) \oplus Y = \widetilde{E}_K(X',Y') \oplus Y'$ or must have found a preimage of 0^n. In both cases, the variable `bad` would have been set to `true`, and it follows again by [9] that

$$\Pr[A^{G_2} \textit{sets}\ \texttt{bad}] \leq \frac{(q+\ell)(q+\ell+1)}{2^n-(q+\ell)} + \frac{q+\ell}{2^n-(q+\ell)}.$$

The aforementioned new game G_3 is equal to the game G_2 *except* that the tweakable block cipher $\widetilde{E}$ and its inverse $\widetilde{E}^{-1}$ are replaced by the functions **EncryptBlock** and **DecryptBlock**, which are modeled as set of pseudo random permutations, where the index is given by the tweak. We assume that they are implemented via lazy sampling. More precisely, the call $\widetilde{E}_K(X,Y)$ is

```
Initialize()
  b ←$ {0,1};  K ←$ K();  B ← 0^n;

Encrypt(H, M)   Game G1, [G2]
  if (b = 0) then
    C ← OnlinePermutation(H, M);
  else
    L_H ← |H|/n;  L ← |M|/n;
    p ← LLCP_n(Q, (H, M));
    Q ← Q ∪ (H, M);
    U ← 0^n;
    for i = 1, ..., L_H do
      τ ← Ẽ_K(U, H_i);
      U ← H_i ⊕ τ;
      if (U ∈ B and i > p) then
        bad ← true;  [U ←$ {0,1}^n \ B;]
      B ← B ∪ U;
    for i = 1, ..., L do
      C_i ← Ẽ_K(U, M_i);
      U ← C_i ⊕ M_i;
      if (U ∈ B and i + L_H > p) then
        bad ← true;  [U ←$ {0,1}^n \ B;]
      B ← B ∪ U;
  return C;

Finalize(d)
  return (b=d);
```

Fig. 10. Games G_1 and G_2 for the proof of Lemma 3. Game G_2 contains the code in the box while G_1 does not.

replaced by an invocation of $\mathbf{EncryptBlock}_K(X, Y)$ and the call $\widetilde{E}_K^{-1}(X, Y)$ is replaced by an invocation of $\mathbf{DecryptBlock}_K(X, Y)$. We now upper bound the difference between G_2 and G_3. So, by definition of G_4, we have

$$|\Pr[A^{G_2} \Rightarrow 1] - \Pr[A^{G_3} \Rightarrow 1]| \leq \mathbf{Adv}_{\widetilde{E}}^{\text{T-IND-CPA}}(q + \ell, O(t)).$$

Finally, we have to upper bound the advantage for an adversary A to win the game G_3. Since the U cannot collide and it is not possible to compute a preimage for any query, the algorithm for $b = 0$ *is* an OPRP, and therefore the success probability to win G_3 for any adversary is 0.5, *i.e.,* she has no advantage in winning this game.

Our claim follows by adding up the individual bounds. □

5.2 Security Analysis of McOE with Tag-Splitting

Theorem 3. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a* McOE *scheme as in Definition 2,* i.e., *$\mathcal{K}$ is the key generation function, $\mathcal{E} =$* **EncryptAuthenticateSplitTag** *and $\mathcal{D} =$* **DecryptAuthenticate-SplitTag**. *For $q \leq 2^{n/2-2}$ we have*

$$\mathbf{Adv}_{\Pi}^{\text{CCA3(OAE,NI)}}(q, \ell, t) \leq \frac{4(q+\ell+2)(q+\ell+3) + 6(2q+\ell)}{2^n - (q+\ell)} + \frac{3q(q+1)}{2^n - q} + \frac{q}{2^{n/2} - q} + 3\mathbf{Adv}_{E,E^{-1}}^{\text{T-IND-CCA}}(2q + \ell, O(t)).$$

Proof. The proof follows from Theorem 1 together with Lemmas 4 and 6. □

For the sake of simplification we provide an upper bound which is much easier to grasp than the original bound, but not as tight as the original bound given in the theorem above.

Corollary 2. *Let assume that $\ell \geq 35$, $\ell \geq q$ and the* T-IND-CCA-*advantage is at most δ for an adversary which amount of queries is at most $q + \ell$ and its running time is $O(t)$. Then the following bound holds*

$$\mathbf{Adv}_{\Pi}^{\text{CCA3(OAE,NI)}}(q, \ell, t) \leq \frac{21\ell^2 + \ell}{2^n - (2\ell)} + \frac{\ell}{2^{n/2} - \ell} + \delta.$$

```
Initialize()
  K <-$ 𝒦(); B <- {0^n, 1^n};
```

```
Finalize()
  return win;
```

```
Encrypt(H, M)  Game G_1
  L_H <- |H|/n;  L <- ⌈|M|/n⌉;
  U <- 0^n;
  for i = 1, ..., L_H do
    τ <- Ẽ_K(U, H_i);
    U <- H_i ⊕ τ;
  for i = 1, ..., L − 1 do
    C_i <- Ẽ_K(U, M_i);
    U <- C_i ⊕ M_i;
  M* <- M_L || τ[0, ... n − |M_L| − 1];
  M* <- M* ⊕ Ẽ_K(1^n, |M_L|);
  C* <- Ẽ_K(U, M*);
  C_L <- C*[0, ..., |M_L| − 1];
  T[0, ..., n − |M_L| − 1] <- C*[|M_L| − 1, ..., n − 1];
  U <- M* ⊕ C*
  T[n − |M_L|, ..., n − 1] <- Ẽ_K(U, τ)[0, ..., |M_L|];
  𝒬 <- (H, M, C, T);
  return (C_1, ..., C_L, T);
```

```
Verify(H, C, T)  Game G_1
  L_H <- |H|/n;  L <- ⌈|C|/n⌉;
  U <- 0^n;
  for i = 1, ..., L_H do
    τ <- Ẽ_K(U, H_i);
    U <- H_i ⊕ τ;
  for i = 1, ..., L − 1 do
    M_i <- Ẽ_K^{-1}(U, C_i);
    U <- C_i ⊕ M_i;
  C* <- C_L || T[0 ... n − |C_L|* − 1];
  M* <- Ẽ_K^{-1}(U, C*);
  U <- M* ⊕ C*;
  M* <- M* ⊕ Ẽ_K(1^n, |C_L|);
  M_L <- M*[0, ..., |C_L| − 1];
  τ'[0 ... n − |C_L| − 1] <- M*[|C_L|, ..., n − 1];
  T' <- Ẽ_K(U, τ);
  if τ'[0 ... n − l* − 1] = τ[0 ... n − l* − 1]
    and T'[0 ... l* − 1] = T[n − l* ... n − 1]
    and (H, C) ∉ 𝒬_{|H,C} then
      win <- true;
  𝒬 <- (H, ⊥, C, ⊥);
  return win;
```

```
Encrypt(H, M)  Game G_2, [G_3]
  L_H <- |H|/n;  L <- ⌈|M|/n⌉;
  p <- LLCP_n(𝒬_{|H,M}, (H, M));
  U <- 0^n;
  for i = 1, ..., L_H do
    τ <- Ẽ_K(U, H_i);
    U <- H_i ⊕ τ;
    if (U ∈ B and i > p) then
      bad <- true; [U <-$ {0,1}^n \ B;]
    B <- B ∪ U;
  for i = 1, ..., L − 1 do
    C_i <- Ẽ_K(U, M_i);
    U <- C_i ⊕ M_i;
    if (U ∈ B and i + L_H > p) then
      bad <- true; [U <-$ {0,1}^n \ B;]
    B <- B ∪ U;
  M* <- M_L || τ[0, ... n − |M_L| − 1];
  M* <- M* ⊕ Ẽ_K(1^n, |M_L|);
  if (M* ∈ A[U] and L + L_H − 1 = p) then
    bad <- true; [M* <-$ {0,1}^n \ A[U];]
  A[U] <- A[U] ∪ M*;
  C* <- Ẽ_K(U, M*);
  C_L <- C*[0, ..., |M_L| − 1];
  T[0, ..., n − |M_L| − 1] <- C*[|M_L| − 1, ..., n − 1];
  U <- C* ⊕ M*;
  if (U ∈ B and L + L_H > p) then
    bad <- true; [U <-$ {0,1}^n \ B;]
  B <- B ∪ U;
  T[n − |M_L|, ..., n − 1] <- Ẽ_K(U, τ)[0, ..., |M_L| − 1];
  𝒬 <- (H, M, C, T);
  return (C_1, ..., C_L, T);
```

```
Verify(H, C, T)  Game G_2, [G_3]
  L_H <- |H|/n;  L <- ⌈|C|/n⌉;
  p <- LLCP_n(𝒬_{|H,M}, (H, M));
  U <- 0^n;
  for i = 1, ..., L_H do
    τ <- Ẽ_K(U, H_i);
    U <- H_i ⊕ τ;
    if (U ∈ B and i > p) then
      bad <- true; [U <-$ {0,1}^n \ B;]
    B <- B ∪ U;
  for i = 1, ..., L − 1 do
    M_i <- Ẽ_K^{-1}(U, C_i);
    U <- C_i ⊕ M_i;
    if (U ∈ B and i + L_H > p) then
      bad <- true; [U <-$ {0,1}^n \ B;]
    B <- B ∪ U;
  C* <- C_L || T[0 ... n − |C_L|* − 1];
  M* <- Ẽ_K^{-1}(U, C*);
  if (M* ∈ A[U] and L + L_H − 1 = p) then
    bad <- true; [M* <-$ {0,1}^n \ A[U];]
  A[U] <- A[U] ∪ M*;
  M* <- M* ⊕ Ẽ_K(1^n, |M_L|);
  M_L <- M*[0, ..., |C_L| − 1];
  τ'[0 ... n − |C_L| − 1] <- M*[|C_L|, ..., n − 1];
  U <- M* ⊕ C*;
  if (U ∈ B and L + L_H + 1 > p) then
    bad <- true; [U <-$ {0,1}^n \ B;]
  B <- B ∪ U;
  T' <- Ẽ_K(U, τ);
  if τ'[0 ... n − l* − 1] = τ[0 ... n − l* − 1]
    and T'[0 ... l* − 1] = T[n − l* ... n − 1]
    and (H, C) ∉ 𝒬_{|H,C} then
      win <- true;
  𝒬 <- (H, ⊥, C, ⊥);
  return win;
```

Fig. 11. Games G_1-G_3 for the proof of Lemma 4. Game G_3 contains the code in the box while G_2 does not.

Lemma 4. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a* McOE *scheme as in Definition 1 (ii). Let $q \leq 2^{n/2-2}$ be the number of total queries an adversary A is allowed to ask and ℓ be an integer representing the total length in blocks of the queries to $\mathcal{E}$ and $\mathcal{D}$. Then,*

$$\mathbf{Adv}_{\Pi}^{\text{INT-CTXT(NI)}}(q, \ell, t) \leq \frac{(2q+\ell+2)(2q+\ell+3)}{2^n - (q+\ell)} + \frac{2(2q+\ell)}{2^n-(q+\ell)} + \frac{q(q+1)}{2^n-q} + \frac{q}{2^{n/2}-q} + \mathbf{Adv}_{\widetilde{E},\widetilde{E}^{-1}}^{\text{T-IND-CCA}}(2q+\ell, O(t)).$$

Proof. Our bound is derived by game playing arguments. Consider games G_1-G_3 of Figure 11 and a fixed adversary A asking at most q queries with a total length of at most ℓ blocks. The functions **Initialize** and **Finalize** are identical for all games in this proof. Lets denote G_0 as the Game INT-CTXT(NI) as defined in Figure 8. Definition 8 states that

$$\mathbf{Adv}_{\Pi}^{\text{INT-CTXT(NI)}}(A) \leq \Pr[A^{G_0} \Rightarrow 1].$$

In G_1, the encryption and verify placeholders are replaced by their generic McOE counterparts as of Definition 1. We now discuss the differences between G_1 and G_2. The set B is initialized with $\{0^n, 1^n\}$ and then collects all new key-input values U which are computed during the encryption or verification process (in lines 209, 215, 227, 240, 246, and 258). Furthermore, the sets $A[U]$ collect the masked values of M^* (cf. lines 220 and 252) for a certain prefix.

In lines 202 and 233, the LLCP$_n$ oracle is inquired. Finally, the variable `bad` is set to `true` if one of the if-conditions in lines 207, 213, 218, 225, 238, 244, 249 or 256 hold. *None* of these modifications affect the values returned to the adversary and therefore

$$\Pr[A^{G_1} \Rightarrow 1] = \Pr[A^{G_2} \Rightarrow 1].$$

For our further discussion we require another game G_4 which is explained in more detail later in this proof[3]. It follows that

$$\begin{aligned} \Pr[A^{G_2} \Rightarrow 1] &\leq \Pr[A^{G_3} \Rightarrow 1] + |\Pr[A^{G_2} \Rightarrow 1] - \Pr[A^{G_3} \Rightarrow 1]| \\ &\leq \Pr[A^{G_3} \Rightarrow 1] + \Pr[A^{G_3} \textit{sets } \texttt{bad}] \\ &\leq \Pr[A^{G_4} \Rightarrow 1] + |\Pr[A^{G_3} \Rightarrow 1] - \Pr[A^{G_4} \Rightarrow 1]| + \Pr[A^{G_3} \textit{sets } \texttt{bad}]. \end{aligned} \tag{4}$$

We now proceed to upper bound any of the three terms contained in (4) – in right to left order. The success probability of game G_3 does not differ from the success probability of G_2 unless a chaining value U occurs twice. In this case, the adversary must (1) either have 'found' a collision for $\widetilde{E}_K(X,Y) \oplus Y$, *i.e.,* she stumbles over (X,Y) and (X',Y') such that $\widetilde{E}_K(X,Y) \oplus Y = \widetilde{E}_K(X',Y') \oplus Y'$ or, (2), must have found a preimage of 0^n or 1^n, which is always the starting point of our chain. Note, the values 0^n and 1^n are initially stored in the set B. In both cases, the variable `bad` would have been set to `true`. From [9] follows as upper bound

$$\frac{(2q+\ell+2)(2q+\ell+3)}{2^n-(q+\ell)} + \frac{2(2q+\ell)}{2^n-(q+\ell)}.$$

Furthermore, we have to consider the case when a collision occurs between the masked value of M^* and the set $A[U]$. As an adversary can ask q queries, it follows that the probability that the flag `bad` is set to `true` in lines 219 and 250 can be upper bounded by

$$\frac{q(q+1)}{2^n-q}.$$

[3] Since the difference is very minor, we do not provide an extra figure.

By adding up both bounds it follows that

$$\Pr[A^{G_3} sets\ \mathtt{bad}] \le \frac{(2q+\ell+2)(2q+\ell+3)}{2^n-(q+\ell)} + \frac{2(2q+\ell)}{2^n-(q+\ell)} + \frac{q(q+1)}{2^n-q}.$$

We now describe the new game G_4. It is equal to G_3 *except* that the block cipher $\widetilde{E}$ and its inverse $\widetilde{E}^{-1}$ are replaced by the functions **EncryptBlock** and **DecryptBlock**, which are modeled as a set of pseudo random permutations, where the index is given by the tweak. We assume that they are implemented via lazy sampling. More precisely, the call $\widetilde{E}_K(X,Y)$ is replaced by an invocation of $\mathbf{EncryptBlock}_K(X,Y)$ and the call $\widetilde{E}_K^{-1}(X,Y)$ is replaced by an invocation of $\mathbf{DecryptBlock}_K(X,Y)$. We now upper bound the difference between G_3 and G_4 by

$$|\Pr[A^{G_3} \Rightarrow 1] - \Pr[A^{G_4} \Rightarrow 1]| \le \mathbf{Adv}^{\text{T-IND-CCA}}_{\widetilde{E},\widetilde{E}^{-1}}(2q+\ell, O(t)).$$

Finally, we have to upper bound the advantage for the adversary A to win the game G_4. A can only win this game if the condition in lines 260-262 (resp. 460-462 for game G_4) holds. As usual, we assume *wlog.* that A does not ask a question if the answer is already known which implies that $(H,C,T) \notin \mathcal{Q}_{|H,C,T}$. We formally adjust lines 260-262 (*i.e.,* choose as the tag computation operation either $\widetilde{E}$ or $\widetilde{E}^{-1}$) such that we always have enough randomness left for our result. For simple reference, we denote the last two chaining values as U_L and U_{L+1}. For our analysis we distinguish between our two main cases. First, the case when $|M_L| = n$, *i.e.,* the size of the last message block is equal to the block size n. Second, the case when the size of the last message block M_L is not equal to n.

Case 1: In this case we first consider that $U_L \in B$. This implies that $(C_1,\dots,C_L)$ must be part of a common prefix of a previous query. The adversary can only win if T is new, *i.e.,* not a part of a previous occured prefix. The upper bound is then given by

$$\Pr\left[\widetilde{E}_K^{-1}(U_{L+1},T) = \tau\right] = 0,$$

since $\widetilde{E}^{-1}$ is a PRP.
If $U_L \notin B$, we can upper bound the success probability for one query by $1/(2^n-q)$. Hence, for q queries we can upper bound the success probability by $q/(2^n-q)$.

Case 2: Now we consider the case $|M_L| \neq n$. It can be upper bounded by Lemma 5. The success probability is at most $q/(2^{n/2}-q)$.

Since both cases are mutually exclusive, we can upper bound the success probability for q queries by

$$\frac{q}{2^{n/2}-q}.$$

Our claim follows by adding up the individual bounds. □

Lemma 5. *Let $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$ be a* McOE *scheme as in Definition 2 and q be the number of total queries with $q \le 2^{n/2-2}$. The probability that an adversary A wins G_4 as defined in the proof of Lemma 4, for any message which is not a multiple of the block size n, can be upper bounded by $q/(2^{n/2}-q)$.*

Proof. For simple reference, we denote the last two chaining values as U_L and U_{L+1}. Furthermore, we denote T^α as the $(n-|C_L|)$-bit string $T[0...n-|C_L|-1]$ and T^β as the $|C_L|$-bit string $T[n-|C_L|,...,n]$. Additionally, we denote the corresponding strings $\tau^\alpha = \tau[0,\dots,|C_L|-1]$ and $\tau^\beta = \tau[|C_L|,\dots,n-1]$, respectively.

Case 1: $U_{L+1} \in B$

This case implies that $(C_1, ..., C_L, T^\alpha)$ must be part of a common prefix from a previous query, otherwise this would imply a collison of the chaining value which is already handled by setting the flag `bad` to `true` in game G_2 (cf. line 248 of Figure 4). Hence, the adversary can only win if T^β is new, *i.e.,* not a part of a previous occured prefix. The upper bound is given by

$$\max_{Z} \left\{ \Pr[\widetilde{E}_{K,}^{-1}(U_{L+1}T^\beta || Z) = \tau] \right\} = 0,$$

since $\widetilde{E}^{-1}$ is a PRP.

Case 2: $U_{L+1} \notin B$

This case implies that $C_L || T^\alpha$ must be new. The probability that the condition from line 260 holds – for q queries – can be upper bounded by

$$\Pr_\alpha = \max_{M_L} \left\{ \Pr\left[\widetilde{E}_K^{-1}(U_L, C_L || T^\alpha) = (M_L || \tau^\alpha) \oplus \widetilde{E}_K(1^n, |M_L|) \right] \right\} \leq \frac{2}{2^{(n-|C_L|)} - 2q}.$$

Hence, the probability for q queries can be upper bound by $\frac{2q}{2^{(n-|C_L|)}-2q}$.

From the assumption $U_{L+1} \notin B$ follows that U_{L+1} is new. Since $\widetilde{E}$ is a PRP, we can upper bound the probability that the condition from line 452 holds by

$$\Pr_\beta = \max_{Z} \left\{ Pr[\widetilde{E}_K(U_{L+1}\tau) = T^\beta || Z] \right\} \leq \frac{1}{2^{|C_L|} - q}.$$

Then, the probability for q queries can be upper bound by $q/(2^{|C_L|} - q)$.

The success probability of this case depends on the length of $|C_L|$. So we can distinguish between the following three subcases.

Subcase 2.1: $|C_L| < n/2$

In this case, we can upper bound $\Pr_\alpha$ by $\frac{1}{2^{n/2}-q}$ and $\Pr_\beta$ by 1. Hence the total success probability for q queries is at most $\frac{q}{2^{n/2}-q}$.

Subcase 2.2: $|C_L| = n/2$

In this case, we can upper bound $\Pr_\alpha$ by $\frac{2}{2^{n/2}-2q}$ and $\Pr_\beta$ by $\frac{1}{2^{n/2}-q}$. Hence the total success probability for q queries is at most $\frac{2q^2}{2^{n-1}-q^2}$.

Subcase 2.3: $|C_L| > n/2$

In this case, we can upper bound $\Pr_\alpha$ by 1 and $\Pr_\beta$ by $\frac{1}{2^{n/2+1}-q}$. Hence the total success probability for q queries is at most $\frac{q}{2^{n/2+1}-q}$.

Since all three subcases are mutually exclusive, we can upper bound the success probability for $q \leq 2^{n/2-2}$ queries by

$$\max \left\{ \frac{q}{2^{n/2} - q}, \frac{2q^2}{2^{n-1} - q^2}, \frac{q}{2^{n/2+1} - q} \right\} \leq \frac{q}{2^{n/2} - q}.$$

Due to the fact that Case 1 and Case 2 are mutually exclusive, we can upper bound the success probability for q queries by

$$\max \left\{ 0, \frac{q}{2^{n/2} - q} \right\} \leq \frac{q}{2^{n/2} - q}.$$

Our claim follows by adding up the individual bounds. □

Lemma 6. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a* McOE *scheme as in Definition 2. Let q be the number of total queries an adversary A is allowed to ask and ℓ be an integer representing the total length of the queries to $\mathcal{E}$ and $\mathcal{D}$. Then,*

$$\begin{aligned}\mathbf{Adv}_{\Pi}^{\mathrm{CPA(\textsc{aoe},\textsc{ni})}}(q,\ell,t) \leq\ & 2\left(\frac{(q+\ell+2)(q+\ell+3)}{2^n-(q+\ell)} + \frac{2(2q+\ell)}{2^n-(q+\ell)} + \frac{q(q+1)}{2^n-q}\right)\\ &+ 2\mathbf{Adv}_{\widetilde{E}}^{\text{T-IND-CPA}}(2q+\ell, O(t)).\end{aligned}$$

Proof. At first we investigate the differences between the $CPA(\textsc{aoe}, \textsc{ni})$ game from Figure 7 and G_1 from Figure 12. In G_1 we have replaced $\mathcal{E}$ by its definition of McOE, and $\$^w$ by an on-line encryption oracle **OPerm** (line 102) that just models a 'perfect' OPRP, *i.e.,* for two plaintexts with an equal prefix it returns two ciphertexts that also share a prefix of the same length. We again assume this oracle to be implemented by lazy sampling. Then, set B collects all chaining values (lines 113 and 119) in order to intercept the occurrence of two equal chaining values which do lead to two equal tweaks for the encryption of a block. The sets $A[U]$ collect all values of masked M^* for specific chaining values U (line 124).

In line 105, the oracle LLCP_n is invoked returning the length of the longest common prefix of (H, M) and $\mathcal{Q}_{|H,M}$.

Finally, the variable `bad` is set to `true` if (one of) the conditions of lines 111/211, 117/217 or 122/222 holds. These changes do not affect the success probability of an adversary, because the output of the oracle remains unchanged. More precisely, the distribution of the output does not change. This means that

$$\mathbf{Adv}_{\Pi}^{\mathrm{CPA(\textsc{aoe},\ \textsc{ni})}}(A) = 2 \cdot |\Pr[A^{G_1} \Rightarrow 1] - 0.5|,$$

and therefore, by common game playing arguments – using a new game G_3 described shortly –

$$\begin{aligned}\Pr[A^{G_1} \Rightarrow 1] &\leq \Pr[A^{G_2} \Rightarrow 1] + |\Pr[A^{G_1} \Rightarrow 1] - \Pr[A^{G_2} \Rightarrow 1]|\\ &\leq \Pr[A^{G_2} \Rightarrow 1] + \Pr[A^{G_2} \mathit{sets}\ \texttt{bad}]\\ &\leq \Pr[A^{G_3} \Rightarrow 1] + |\Pr[A^{G_2} \Rightarrow 1] - \Pr[A^{G_3} \Rightarrow 1]| + \Pr[A^{G_2} \mathit{sets}\ \texttt{bad}].\end{aligned}$$

The success probability of game G_2 does not differ from the success probability of G_1 unless (1) a chaining value U occurs twice or (2) a collision between two masked values M^* – sharing the same chaining value U – occurs.

In the first case, the adversary must either have found a collision for $\widetilde{E}_K(X,Y) \oplus Y$, *i.e.,* she has found (X, Y) and (X', Y') such that $\widetilde{E}_K(X,Y) \oplus Y = \widetilde{E}_K(X',Y') \oplus Y'$ or must have found a preimage of 0^n or 1^n. In these cases, the variable `bad` would have been set to `true`, and it follows again by [9] that

$$\frac{(q+\ell+2)(q+\ell+3)}{2^n-(q+\ell)} + \frac{2(2q+\ell)}{2^n-(q+\ell)}$$

In the second case, an adversary can ask q queries, and it follows that the probability for the flag `bad` is set to `true` in line 123 can be upper bounded by

$$\frac{q(q+1)}{2^n-q}.$$

By adding up both bounds follows

$$\Pr[A^{G_2} \mathit{sets}\ \texttt{bad}] \leq \frac{(q+\ell+2)(q+\ell+3)}{2^n-(q+\ell)} + \frac{2(2q+\ell)}{2^n-(q+\ell)} + \frac{q(q+1)}{2^n-q}.$$

Initialize()
$b \xleftarrow{\$} \{0,1\};\ K \xleftarrow{\$} \mathcal{K}();\ B \leftarrow \{0^n, 1^n\};$

Finalize(d)
  return (b=d);

Encrypt(H, M) Game G_1, $\boxed{G_2}$
  if $(b = 0)$ **then**
    $C \leftarrow \mathbf{OPerm}(H, M)$;
  else
    $L_H \leftarrow |H|/n;\ L \leftarrow |M|/n;$
    $p \leftarrow \mathrm{LLCP}_n(\mathcal{Q}, (H, M));$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup (H, M);$
    $U \leftarrow 0^n;$
    for $i = 1, \ldots, L_H$ **do**
      $\tau \leftarrow \widetilde{E}_K(U, H_i);$
      $U \leftarrow H_i \oplus \tau;$
      if $(U \in B$ **and** $i > p)$ **then**
        bad $\leftarrow$ **true**; $\boxed{U \xleftarrow{\$} \{0,1\}^n \setminus B;}$
      $B \leftarrow B \cup U;$
    for $i = 1, \ldots, L - 1$ **do**
      $C_i \leftarrow \widetilde{E}_K(U, M_i);$
      $U \leftarrow C_i \oplus M_i;$
      if $(U \in B$ **and** $i + L_H > p)$ **then**
        bad $\leftarrow$ **true**; $\boxed{U \xleftarrow{\$} \{0,1\}^n \setminus B;}$
      $B \leftarrow B \cup U;$
    $M^* \leftarrow M_L || \tau[0, \ldots n - |M_L| - 1];$
    $M^* \leftarrow M^* \oplus \widetilde{E}_K(1^n, |M_L|);$
    if $(M^* \in A[U]$ **and** $L + L_H - 1 = p)$ **then**
      bad $\leftarrow$ **true**; $\boxed{M^* \xleftarrow{\$} \{0,1\}^n \setminus A[U];}$
    $A \leftarrow A[U] \cup M^*;$
    $C^* \leftarrow \widetilde{E}_K(U, M^*);$
    $C_L \leftarrow C^*[0, ..., |M_L| - 1];$
  return $(C_1, \ldots, C_L)$;

Fig. 12. Games G_1 and G_2 for the proof of Lemma 3. Game G_2 contains the code in the box while G_1 does not.

The aforementioned new game G_3 is equal to the game G_2 *except* that the block cipher $\widetilde{E}$ and its inverse $\widetilde{E}^{-1}$ are replaced by randomly chosen functions **EncryptBlock** and **DecryptBlock**, which are modeled as pseudo random permutations . We assume that they are implemented via lazy sampling. More precisely, the call $\widetilde{E}_K(X)$ is replaced by an invocation of $\mathbf{EncryptBlock}_K(X)$ and the call $\widetilde{E}_K^{-1}(X)$ is replaced by an invocation of $\mathbf{DecryptBlock}_K(X)$. We now upper bound the difference between G_2 and G_3. So, by definition of G_4, we have

$$|\Pr[A^{G_2} \Rightarrow 1] - \Pr[A^{G_3} \Rightarrow 1]| \leq \mathbf{Adv}_{\widetilde{E},\widetilde{E}^{-1}}^{\text{T-IND-CPA}}(2q + \ell, O(t)).$$

Finally, we have to upper bound the advantage for an adversary A to win the game G_3. Since U cannot collide and it is not possible to compute a preimage for any query, the algorithm for $b = 0$ *is* an OPRP, and therefore the success probability to win G_3 for any adversary is 0.5, *i.e.,* she has no advantage in winning this game.

Our claim follows by adding up the individual bounds. □

6 The On-Line Authenticated Encryption Scheme McOE-X

This section shows the security of McOE-X for any given block cipher. The here presented upper bounds are based on the generic proof of the McOE family as showed in Section 5. Note, that we have generalized the xor-operation between the key and the tweak by a function $\varphi : D_k \times D_n \to D_n$. For any fixed key K, $\varphi(K, \cdot)$ and the xor-operation are injectiv. Therefore, we can replace the xor-operation by the function φ.

Theorem 4. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a* McOE-X *scheme as in Definition 1 and 2, where the tweakable block cipher $\widetilde{E}$ is given by*

$$\widetilde{E}_K(U, M) := E_{\varphi(K,U)}(M).$$

with $E \in Block(n, n)$ and φ is an injective function. Furthermore, the amount of queries an adversary is allowed to ask is at most $2^{n/2-2}$. Then, the upper bounds for the variants with and without using the Tag-Splitting method are as follows.

(i) Security without Tag-Splitting.

$$\mathbf{Adv}_{\Pi}^{\text{CCA3(OAE,NI)}}(q,\ell,t) \le \frac{3(q+\ell)(q+\ell+1)+4q+3\ell}{2^n-(q+\ell)} + 3\mathbf{Adv}_{E}^{\text{RK-CCA-PRP}}(2q+\ell, O(t)).$$

(ii) Security with Tag-Splitting.

$$\mathbf{Adv}_{\Pi}^{\text{CCA3(OAE,NI)}}(q,\ell,t) \le \frac{4(q+\ell+2)(q+\ell+3)+6(2q+\ell)}{2^n-(q+\ell)} + \frac{3q(q+1)}{2^n-q} + \frac{q}{2^{n/2}-q} + 3\mathbf{Adv}_{E}^{\text{RK-CCA-PRP}}(2q+\ell, O(t)).$$

Proof. *The proofs of* (*i*) *and* (*ii*) *follow from Theorem 2 and Theorem 3. Since φ is an injective function, a collision for the chaining values U implies a collision for the values of $\varphi(K,U)$. Furthermore, it is easy to see that the advantage for the tweakable block cipher can be upper bounded by the* RK-CCA-PRP *advantage of an adversary A, even though she has only limited control over the tweak,* i.e., *the chaining value U.* □

7 The On-Line Authenticated Encryption Scheme McOE-D

Theorem 5. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a* McOE-X *scheme as in Definition 1 and 2, where the tweakable block cipher $\widetilde{E}$ is given by*

$$\widetilde{E}_K(U,M) := E_K(E_K(M) \oplus U).$$

with $E \in Block(n,n)$ and φ is an injective function. Furthermore, the amount of queries an adversary is allowed to ask is at most $2^{n/2-6}$. Then, the upper bounds for the variants with and without using the Tag-Splitting method are as follows.

(i) Security without Tag-Splitting.

$$\mathbf{Adv}_{\Pi}^{\text{CCA3(OAE,NI)}}(q,\ell,t) \le \frac{3(q+\ell)(q+\ell+1)+4q+3\ell}{2^n-(q+\ell)} + 3\left(2\mathbf{Adv}_{E,E^{-1}}^{\text{CCA-PRP}}(2q+\ell, O(t)) + \frac{8(2q+\ell)^2+2(2q+\ell)}{2^n-(2q+\ell)}\right)$$

(ii) Security with Tag-Splitting.

$$\mathbf{Adv}_{\Pi}^{\text{CCA3(OAE,NI)}}(q,\ell,t) \le \frac{4(q+\ell+2)(q+\ell+3)+6(2q+\ell)}{2^n-(q+\ell)} + \frac{3q(q+1)}{2^n-q} + \frac{q}{2^{n/2}-q} + 3\left(2\mathbf{Adv}_{E,E^{-1}}^{\text{CCA-PRP}}(2q+\ell, O(t)) + \frac{8(2q+\ell)^2+2(2q+\ell)}{2^n-(2q+\ell)}\right)$$

Proof. *The proofs of* (*i*) *and* (*ii*) *follow from Theorem 2, Theorem 3, and Lemma 7.* □

Lemma 7. *Let $E \in Block(n,n)$. Lets define the tweakable block cipher $\widetilde{E}_K$ as*

$$\widetilde{E}_K(U,M) := E_K(E_K(M) \oplus U),$$

where E_K denotes a common block cipher, M the message, and U (chaining value) the tweak. Furthermore, the inverse of $\widetilde{E}_K$ is defined by

$$\widetilde{E}_K^{-1}(U,C) := E_K^{-1}(E_K^{-1}(C) \oplus U),$$

where E_K denotes a common block cipher, C the ciphertext, and U (chaining value) the tweak. Then, the T-IND-CCA *advantage of an adversary A is given by*

$$\mathbf{Adv}_{\widetilde{E},\widetilde{E}^{-1}}^{\text{T-IND-CCA}}(q,t) \le \left(2\mathbf{Adv}_{E,E^{-1}}^{\text{CCA-PRP}}(q, O(t)) + \frac{8q^2+2q}{2^n-q}\right).$$

```
Initialize()
  b ←$ {0,1};  B1 ← ∅; B2 ← ∅;

Finalize(d)
  return (d=b);
```

```
Encrypt(U,M)          Game G1
  if (∃C' : (U,M,C') ∈ Q) then
    return C';
  if (b = 1) then
    C ← π(π(M) ⊕ U);
  else
    C ←$ {0,1}^n;
  Q ← Q ∪ (U,M,C);
  return C;

Decrypt(U,V,C)
  if (∃M' : (U,M',C) ∈ Q) then
    return M';
  if (b = 1) then
    M ← π^{-1}(π^{-1}(M) ⊕ U);
  else
    M ←$ {0,1}^n;
  Q ← Q ∪ (U,M,C);
  return M;
```

```
Encrypt(U,M)          Game G2, [G3]
  if ∃C' : (U,M,C') ∈ Q then
    return C';
  if (b = 1) then
    X ← π(M) ⊕ U;
    if (X ∈ B1) or (X ∈ Q|M) then
      bad ← true; [X ←$ {0,1}^n \ B1 ∪ Q|M];
    B1 ← B1 ∪ {X};
    C ← π(X);
    B2 ← B2 ∪ {X ⊕ U};
    if (C ∈ B2) then
      bad ← true; [C ←$ {0,1}^n \ B2];
  else
    C ←$ {0,1}^n;
  Q ← (U,C,M);
  return C;

Decrypt(U,C)
  if (∃M' : (U,M',C) ∈ Q) then
    return M';
  if (b = 1) then
    Y ← π^{-1}(C) ⊕ U;
    if (Y ∈ B2) or (Y ∈ Q|C) then
      bad ← true; [Y ←$ {0,1}^n \ B2 ∪ Q|C];
    B2 ← B2 ∪ {Y};
    M ← π^{-1}(Y);
    B1 ← B1 ∪ {Y ⊕ U};
    if (M ∈ B1) then
      bad ← true; [M ←$ {0,1}^n \ B1];
  else
    M ←$ {0,1}^n;
  Q ← (U,C,M);
  return M;
```

Fig. 13. Games G_1, G_2, and G_3 for the proof of Lemma 7. Game G_3 contains the code in the box while G_2 does not. The functions **Initialize** and **Finalize** are the same for all three games.

Proof. The proof borrows ideas from the XEX proof presented by Rogaway in [40]. Let A be an adversary that runs in time t and *wlog.* makes exactly q queries. At first we define

(1) $p_1 = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{\widetilde{E}_K(\cdot,\cdot),\widetilde{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1]$,

(2) $p_2 = \Pr[\pi \xleftarrow{\$} \text{Perm}(n) : A^{\widetilde{\pi}_\pi(\cdot,\cdot),\widetilde{\pi}^{-1}_{\pi^{-1}}(\cdot,\cdot)} \Rightarrow 1]$,

(3) $p_3 = \Pr[A^{\$(\cdot,\cdot)\$(\cdot,\cdot)} \Rightarrow 1]$,

(4) $p_4 = \Pr[\pi \xleftarrow{\$} \text{TPerm}(v,n) : A^{\pi(\cdot,\cdot),\pi^{-1}(\cdot,\cdot)} \Rightarrow 1]$.

The experiment denoted by (1) models the adversary A with access to the enciphering and deciphering function of the tweakable block cipher $\widetilde{E}/\widetilde{E}^{-1}$, respectively. Experiment (2) denotes the replacement of the block cipher E_K by the PRP π. Experiment (4) however models the adversary A having access to an ideal tweakable block cipher. So the CCA-PRP advantage of an adversary on the tweakable block cipher $\widetilde{E}$ is upper bounded by

$$p_1 - p_4 = (p_1 - p_2) + (p_2 - p_3) + (p_3 - p_4),$$

which we proceed to do in exact these three steps. It is easy to upper bound the first and the third addend as follows.

$(p_1 - p_2)$**.** Here we consider the difference between McOE-D using a block cipher E, and McOE-D, where the block cipher is replaced by the PRP π. The success probability is therefore upper bounded by $\mathbf{Adv}^{CCA-PRP}_{E,E^{-1}}(q, O(t))$.

$(p_3 - p_4)$. This is the well-known replacement of a random permutation – and its inverse – by a pair of random functions. Since the adversary is allowed to ask up to q queries, the probability is upper bounded by $(q^2 - q)/2^{n+1}$.

We now upper bound the second addend, $p_2 - p_3$ by a game playing argument. Consider games G_1, G_2 and G_3 of Figure 13. Game G_1 is defined in a way such that $|p_2 - p_3| = \Pr[A^{G_1} \Rightarrow 1]$. In G_2 we modified the case $b = 1$ as follows. In lines 207 and 225 (lines 209 and 223) the xor-output of the encrypted plaintext X (decrypted ciphertext Y) and the tweak U of each query is added to the set B_1 (B_2).

Additionally, in line 205 (221), we test, whether the xor-output of a new query is already element of the set B_1 (B_2) or if the value of X (Y) is equal to a plaintext (ciphertext) which already exists in the query history queue. Furthermore, we test in line 210 (226) if a ciphertext (plaintext) is already an element of the set B_2 (B_1). If one of these tests succeed, we set a flag `bad` to `true`. These cases imply that an adversary gains knowledge collected from previous queries. If no bad event occurs, the set of remaining available outcome of the encryption (decryption) function is uniformly distributed. Since these changes do not effect the success probability for any adversary it follows that

$$\Pr[A^{G_1} \Rightarrow 1] = \Pr[A^{G_2} \Rightarrow 1].$$

By common game playing arguments, it holds that

$$|\Pr[A^{G_2} \Rightarrow 1] - \Pr[A^{G_3} \Rightarrow 1]| \leq \Pr[A^{G_3} \mathit{sets}\ \texttt{bad}].$$

Then, clearly,

$$\begin{aligned} |p_2 - p_3| &= \Pr[A^{G_2} \Rightarrow 1] \\ &\leq \Pr[A^{G_3} \Rightarrow 1] + |\Pr[A^{G_2} \Rightarrow 1] - \Pr[A^{G_3} \Rightarrow 1]| \\ &\leq \Pr[A^{G_3} \Rightarrow 1] + \Pr[A^{G_3} \mathit{sets}\ \texttt{bad}]. \end{aligned} \tag{5}$$

We are left with upper bounding the two addends of (5). We first bound $\Pr[A^{G_3} \mathit{sets}\ \texttt{bad}]$. In line 206 `bad` is set if the xor difference $X = \pi(M) \oplus U$ is already an element of the set B_1 and the value X consists already in the query history queue. We can upper bound this probabilities by $2(2q^2 + q)/(2^n - q)$, since π is PRP. By reusing this argument, we have the same bound for line 222. Additionally, the probability that `bad` is set to `true` in line 211 is given by $(q^2 + q)/(2^n - q)$. The same argument holds for line 227 and it follows that

$$\Pr[A^{G_3} \mathit{sets}\ \texttt{bad}] \leq \frac{6q^2 + 4q}{2^n - q}.$$

The success probability for winning game three,*i.e.,* the event $A^{G_3} \Rightarrow 1$, can be upper bounded by the common PRP-PRF game playing argument given in [6]. Hence, the success probability is given by

$$\Pr[A^{G_3} \Rightarrow 1] \ \leq \ \frac{q^2 - q}{2^{n+1}}.$$

Our claim follows by adding up the individual bounds. □

Remarks. If an adversary has access to inner block cipher the double construction has some intense security issues as shown in [8]. Hence, the adversary is only allowed to query the tweakable block cipher $\widetilde{E}$ and not the block cipher inside.

8 The On-Line Authenticated Encryption Scheme McOE-G

This section shows the security of McOE-G for any given block cipher and when using an ϵ-AXU secure hash function. The here presented upper bounds are based on (1) the generic proof of the McOE family as showed in Section 5 and (2) the paper of Liskov *et al.* (see Theorem 2 of [31]).

Liskov *et al.* showed that the T-IND-CCA advantage of an adversary A is at most

$$\mathbf{Adv}^{\text{T-IND-CCA}}_{\widetilde{E},\widetilde{E}^{-1}}(q,t) \leq \mathbf{Adv}^{\text{CCA-PRP}}_{E,E^{-1}}(q,O(t)) + 3\epsilon q^2 \tag{6}$$

We use this result for the following security proof.

Theorem 6.
Let $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$ be a McOE-G *scheme as in Definition 1 and 2, where the tweakable block cipher $\widetilde{E}$ is given by*

$$\widetilde{E}_K(U,M) := E_{K_1}(M \oplus H_{K_2}(U)) \oplus H_{K_2}(U).$$

with $E \in Block(n,n)$ and H is a family of ϵ-AXU hash functions. Furthermore, the amount of queries an adversary is allowed to ask is at most $2^{n/2-2}$. Then, the upper bounds for the variants with and without using the Tag-Splitting method are as follows.

(i) Security without Tag-Splitting.

$$\mathbf{Adv}^{\text{CCA3(OAE,NI)}}_{\Pi}(q,\ell,t) \leq \frac{3(q+\ell)(q+\ell+1)+4q+3\ell}{2^n-(q+\ell)} + 3\left(\mathbf{Adv}^{\text{CCA-PRP}}_{E,E^{-1}}(2q+\ell,O(t)) + 3\epsilon(2q+\ell)^2\right)$$

(ii) Security with Tag-Splitting.

$$\mathbf{Adv}^{\text{CCA3(OAE,NI)}}_{\Pi}(q,\ell,t) \leq \frac{4(q+\ell+2)(q+\ell+3)+6(2q+\ell)}{2^n-(q+\ell)} + \frac{3q(q+1)}{2^n-q} + \frac{q}{2^{n/2}-q} + 3\left(\mathbf{Adv}^{\text{CCA-PRP}}_{E,E^{-1}}(2q+\ell,O(t)) + 3\epsilon(2q+\ell)^2\right)$$

Proof. *The proofs of* (*i*) *and* (*ii*) *follow from Theorem 2, Theorem 3 and Equation 6.* □

Remark. McOE-G is not secure, if an adversary has oracle access to the internal building blocks, *i.e.,* the block cipher E and the ϵ-AXU hash function H. This is shown by Black *et al.* in [8]. Hence, it is crucial that the adversary is only allowed to query the tweakable block cipher $\widetilde{E}$ and not one of its parts.

9 Discussion

New Challenges for Research. At this point of time, cryptographic research has developed an inpressive number of good schemes for encryption, authentication, and authenticated encryption. Many of these schemes have been proven secure under standard assumptions on the underlying primitives. In practice, however, such schemes are often used in a way that undermines security. Trying to design cryptosystems as "misuse resistant" as possible still stands as a challenge for cryptographers.

Furthermore, our research seems to pose new challenges for the design of symmetric primitives. Ideally, we would like to implement McOE using a tweakable n-bit block cipher with

n-bit tweaks, supporting fast random tweak changes. Due to the current lack of such a primitive, we designed McOE-X, which requires an ordindary n-bit block cipher being secure against XOR-related key attacks, and supporting fast random key changes. Much beyond McOE, cryptosystem designers could benefit from new tweak-agile tweakable block ciphers and new key-agile ordinary block ciphers.

It is mentionable that McOE-X, when using Threefish-512 in software, performs considerably better as when using software or even hardware AES-128. Note, Threefish-512 actually is a tweakable block cipher, but the 128-bit tweak is too short for McOE. As an alternative, we developed further variants of McOE using double encryption and Galois field arithmetic. These two variants also do not expose the underlying block cipher to related-key attacks.

Conclusion. Originally, this research has been inspired by the search for a default authenticated encryption mode of operation for a general-purpose cryptographic library. It should offer, by default, a huge failure tolerance for practical software developers and still allow being used in an on-line manner.

Since the well-known schemes, such as OCB and SIV, did not fit our requirements, we searched for other ways to achieve the security and functionality we were looking for. Apart from McOE, generic composition (Encrypt-then-Mac) of a secure on-line cipher for encryption and a secure deterministic MAC for authentication, using two independent keys might be another solution. As it turned out, using McOE, one can save the additional key and the time to generate the MAC by using a slightly tweaked on-line cipher for both encryption and authentication.

References

1. Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. Online Ciphers and the Hash-CBC Construction. In *CRYPTO*, pages 292–309, 2001.
2. Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. Online Ciphers and the Hash-CBC Construction. Cryptology ePrint Archive, Report 2007/197; full version of [1], 2007. `http://eprint.iacr.org/`.
3. Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *J. Cryptology*, 21(4):469–491, 2008.
4. Mihir Bellare and Phillip Rogaway. Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In *ASIACRYPT*, pages 317–330, 2000.
5. Mihir Bellare and Phillip Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. Cryptology ePrint Archive, Report 2004/331; full version of [6], 2004. `http://eprint.iacr.org/`.
6. Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *EUROCRYPT*, pages 409–426, 2006.
7. Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX Mode of Operation. In *FSE*, pages 389–407, 2004.
8. John Black, Martin Cochran, and Thomas Shrimpton. On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. In *EUROCRYPT*, pages 526–541, 2005.
9. John Black, Phillip Rogaway, and Thomas Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In *CRYPTO*, pages 320–335, 2002.
10. Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. In *MOBICOM*, pages 180–189, 2001.
11. Enrico Buonanno, Jonathan Katz, and Moti Yung. Incremental Unforgeable Encryption. In *FSE*, pages 109–124, 2001.
12. Intel Corporation. AES-NI Sample Library v1.2. `http://software.intel.com/en-us/articles/download-the-intel-aesni-sample-library/`, 2010.
13. Joan Daemen. *Hash Function and Cipher Design: Strategies Based on Linear and Differential Cryptanalysis.* Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium, March 1995.
14. Morris Dworkin. *Special Publication 800-38C: Recommendation for block cipher modes of operation: the CCM mode for authentication and confidentiality.* National Institute of Standards and Technology, U.S. Department of Commerce, May 2005.
15. Pierre-Alain Fouque, Gwenaëlle Martinet, Frédéric Valette, and Sébastien Zimmer. On the Security of the CCM Encryption Mode and of a Slight Variant. In *ACNS*, pages 411–428, 2008.

16. Brian Gladman. Brian Gladman's AES Implementation, 19th June 2006. `http://gladman.plushost.co.uk/oldsite/AES/index.php`.
17. Virgil D. Gligor and Pompiliu Donescu. Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes. In *FSE*, pages 92–108, 2001.
18. Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
19. Shay Gueron and Michael E. Kounavis. Efficient implementation of the Galois Counter Mode using a carry-less multiplier and a fast reduction algorithm, journal = Inf. Process. Lett. 110(14-15):549–553, 2010.
20. George Hotz. Console Hacking 2010 - PS3 Epic Fail. 27th Chaos Communications Congress, 2010. `http://events.ccc.de/congress/2010/Fahrplan/attachments/1780_27c3_console_hacking_2010.pdf`.
21. ISO/IEC. *19772:2009, Information technology – Security techniques – Authenticated Encryption*, 2009.
22. Tetsu Iwata. New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In *FSE*, pages 310–327, 2006.
23. Tetsu Iwata. Authenticated Encryption Mode for Beyond the Birthday Bound Security. In *AFRICACRYPT*, pages 125–142, 2008.
24. Tetsu Iwata and Kan Yasuda. BTM: A Single-Key, Inverse-Cipher-Free Mode for Deterministic Authenticated Encryption. In *Selected Areas in Cryptography*, pages 313–330, 2009.
25. Tetsu Iwata and Kan Yasuda. HBS: A Single-Key Mode of Operation for Deterministic Authenticated Encryption. In *FSE*, pages 394–415, 2009.
26. Charanjit S. Jutla. Encryption Modes with Almost Free Message Integrity. *J. Cryptology*, 21(4):547–578, 2008.
27. Jonathan Katz and Moti Yung. Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In *FSE*, pages 284–299, 2000.
28. Tadayoshi Kohno. Attacking and Repairing the WinZip Encryption Scheme. In *ACM Conference on Computer and Communications Security*, pages 72–81, 2004.
29. Tadayoshi Kohno, John Viega, and Doug Whiting. CWC: A High-Performance Conventional Authenticated Encryption Mode. In *FSE*, pages 408–426, 2004.
30. Ted Krovetz and Phillip Rogaway. New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In *FSE*, pages 310–327, 2006.
31. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. *J. Cryptology*, 24(3):588–613, 2011.
32. Stefan Lucks. Ciphers secure against related-key attacks. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 359–370. Springer, 2004.
33. Stefan Lucks. Two-Pass Authenticated Encryption Faster Than Generic Composition. In *FSE*, pages 284–298, 2005.
34. David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In *In INDOCRYPT, volume 3348 of LNCS*, pages 343–355. Springer, 2004.
35. Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*, chapter 9.40. CRC Press, 1996.
36. Niels Ferguson and Stefan Lucks and Bruce Schneier and Doug Whiting and Mihir Bellare and Tadayoshi Kohno and Jon Callas and Jesse Walker. Skein source code and test vectors. `http://www.skein-hash.info/downloads`.
37. Kenneth G. Paterson and Gaven J. Watson. Plaintext-dependent decryption: A formal security treatment of ssh-ctr. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 345–361. Springer, 2010.
38. Phillip Rogaway. Authenticated-Encryption with Associated-Data. In *ACM Conference on Computer and Communications Security*, pages 98–107, 2002.
39. Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In *ASIACRYPT*, pages 16–31, 2004.
40. Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *In Advances in Cryptology ASIACRYPT 2004*, pages 16–31. Springer, 2004.
41. Phillip Rogaway. Nonce-Based Symmetric Encryption. In *FSE*, pages 348–359, 2004.
42. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In *ACM Conference on Computer and Communications Security*, pages 196–205, 2001.
43. Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In *EUROCRYPT*, pages 373–390, 2006.
44. Phillip Rogaway and Thomas Shrimpton. Deterministic Authenticated-Encryption: A Provable-Security Treatment of the Key-Wrap Problem. Cryptology ePrint Archive, Report 2006/221; full version of [43], 2006. `http://eprint.iacr.org/`.
45. Phillip Rogaway and Haibin Zhang. Online Ciphers from Tweakable Blockciphers. In *CT-RSA*, pages 237–249, 2011.

46. Todd Sabin. Vulnerability in Windows NT's SYSKEY encryption. *BindView Security Advisory*, 1999. Available at `http://marc.info/?l=ntbugtraq&m=94537191024690&w=4`.
47. Hongjun Wu. The Misuse of RC4 in Microsoft Word and Excel. Cryptology ePrint Archive, Report 2005/007, 2005. `http://eprint.iacr.org/`.

A Misuse-Attacks: The Weak Point of Current Authenticated Encryption (AE) Schemes.

A.1 Attacking Schemes without Claimed Resistance Against Nonce-Reuse

Cipher-block-chaining (CBC) is an unauthenticated encryption mode which is sometimes used as the encryption component of an AE scheme. But one can easily distinguish CBC encryption from a good on-line cipher, if the nonce (or the IV) is constant. The attack from [1] only needs three chosen plaintexts. Counter mode, which has been very popular among the designers of AE schemes, fails terribly in nonce-reuse settings, since it generates exactly the same keystream twice when a nonce is reused. It was to be expected that a scheme using counter mode or CBC inherits the nonce-reuse issue from that mode. But, as it turned out, common AE schemes also fail at the authenticity frontier (see Table 2 in Section 1 for an overview). This is an unpleasant surprise, since the cryptographic community has known well deterministic MACs for a long time – so why is the authenticity provided by most authenticated encryption schemes so much more fragile than the authenticity provided by well-known MACs?

The following two attack patterns will be used in most of our attacks.

Repeated Keystream. Many AE schemes generate a keystream $S = F_K(V)$ of length $|M|$, depending on the secret key K and the nonce V. They encrypt a message M by computing the corresponding ciphertext $C = S \oplus M$, typically by applying a block cipher in counter mode. If the same nonce is used more than once, the following attack straightforwardly breaks the privacy:

1. Encrypt a plaintext M under the nonce V to a ciphertext C with tag T.
2. Encrypt a plaintext $M' \neq M$ under the same V to a ciphertext C' and a tag T'.
3. It turns out that $C' = C \oplus M \oplus M'$ holds.

Linear Tag. Many AE schemes, which generate a keystream $S = F_K(V)$ as above, apply the Encrypt-then-Mac (EtM) paradigm and allow to rewrite the authentication tag T as

$$T = f(V) \oplus g(C),$$

where V is the nonce, C is the ciphertext, and f and g are some key-dependent functions. This enables the adversary to mount the linear attack introduced by the following four steps.

1. Encrypt the plaintext M under the nonce V to (C, T) with $T = f(V) \oplus g(C)$.
2. Encrypt the plaintext $M' \neq M$ with $|M'| = |M|$ under the nonce $V' \neq V$ to (C', T') with the tag $T' = f(V') \oplus g(C')$.
3. Set $M'' := M' \oplus C' \oplus C$. Encrypt M'' under the nonce V' to (C'', T''). Observe $C'' = C$, thus $T'' = f(V') \oplus g(C)$.
4. Set $T^* = T \oplus T' \oplus T'' = f(V) \oplus g(C')$, The adversary accepts (C', T^*) under V.

Two-Pass AE(AD) Modes: CWC [29], GCM [34], CCM [14], EAX [7], CHM [22] . All the common two-pass AE(AD) modes, CHM,CWC, GCM, CCM and EAX, use the counter mode as the underlying encryption operation and are thus vulnerable to the repeated keystream attack pattern. Four of them, CHM, CWC, GCM, and EAX, are designed according to the EtM paradigm, and are thus vulnerable to the linear tag attack pattern. The designers of CCM followed Mac-then-Encrypt (MtE), which seems to defend against the linear tag pattern. Forgery attacks against CCM have been presented in [15], though.

Mixed AE(AD) Modes: RPC [11] and CCFB [33]. RPC combines counter mode and electronic codebook mode. Given an n-bit block cipher E under a key K and a c-bit counter cnt, RPC takes an $(n-c)$-bit plaintext block M_i and computes the ciphertext block $C_i := E_K(M_i||(\text{cnt}+i) \bmod 2^c)$. Authentication is performed locally for each ciphertext block: During decryption, RPC computes $(M_i||X_i) = E_K^{-1}(C_i)$ and accepts M_i as authentic if and only if $X_i = (\text{cnt}+i) \bmod 2^c$. The nonce defines cnt.

Under nonce-reuse, the same sequence $(\text{cnt}+i) \bmod 2^c$ of counter values is used for different messages. This makes it easy to attack the privacy – essentially, when encrypting messages of m $(n-c)$-bit blocks, RPC degrades into m independent electronic codebooks. Also, given two authentic ciphertexts, $(C_1^0, \dots, C_L^0)$ and $(C_1^1, \dots, C_L^1)$, any ciphertext $(C_1^{\sigma(1)}, \dots, C_L^{\sigma(L)})$ with $\sigma(i) \in \{0,1\}$ is valid, since authenticity is verified locally for each $C_i^{\sigma(i)}$.

Similarly to RPC, CCFB is a combination of Counter and CFB mode. Given an $(n-c)$-bit nonce and $(n-c)$-bit plaintext blocks M_1, …, M_m CCFB generates $(n-c)$-bit temporary values D_i, c-bit temporary tags T_i and $(n-c)$-bit ciphertext blocks C_i as follows:

1. $C_0 := N$;
2. for $i \in \{2, \dots, m\}$ do $(D_i||T_i) := E_K(C_{i-1}||\langle i\rangle)$; $C_i := M_i \oplus D_i$
3. $(*, T_{m+1}) := E_K(C_m||\langle i+2\rangle)$;

Unlike RPC, CCFB only uses the local tags T_i temporarily; the final authentication tag is $T = T_1 \oplus T_2 \oplus \cdots \oplus T_{m+1}$.

Note that the first ciphertext block C_1 is essentially the encryption of M_1 in counter mode. Thus, a variant of the repeated keystream pattern is applicable to CCFB. And the following variant of the linear tag pattern applies to CCFB (for simplicity, we assume single-block messages only):

1. Encrypt the plaintext M_1 under V to (C_1, T).
2. Encrypt the plaintext $M_1' \neq M_1$ under $V' \neq V$ to (C_1', T').
3. Set $M_1'' := M_1' \oplus C_1' \oplus C_1$. Encrypt M_1'' under V' to (C_1'', T''). Observe $C_1'' = C_1$.
4. The adversary accepts $(C_1', T \oplus T' \oplus T'')$ under V.

One-Pass AE(AD) Modes: IAPM [26], OCB1[42], OCB2[39], OCB3[30], TAE [31]. Given a nonce V and a secret key K, IAPM [26] encrypts a plaintext $(M_1, \dots, M_m)$ to a ciphertext $(C_1, \dots C_m)$ and an authentication tag T as follows.

Initial step: Generate $m+2$ values $s_0, s_1, \dots s_{m+1}$ depending on V and K, but not on the plaintext $(M_1, \dots, M_m)$.
Encryption: For $i \in \{1, \dots, m\}$: $C_i := E_K(M_i \oplus s_i) \oplus s_i$.
Authentication tag: $T := E_K(s_{m+1} \oplus \sum_{1 \le i \le m} M_i) \oplus s_0$.

When encrypting messages of m n-bit blocks, IAPM behaves like a set of m independent electronic codebook and, like RPC, is vulnerable to distinguishing attacks based on this. Similarly to RPC, IAPM behaves like a set of m independent electronic codebooks and is vulnerable to the same distinguishing attack. A forgery can exploit the fact that two different same-length messages $(M_1, \dots, M_m)$ and $(M_1', \dots, M_m')$, encrypted under the same nonce, have the same authentication tag $T = E_K(s_{m+1} \oplus \sum_{1 \le i \le m} M_i) \oplus s_0 = E_K(s_{m+1} \oplus \sum_{1 \le i \le m} M_i') \oplus s_0$ if $\sum_{1 \le i \le m} M_i = \sum_{1 \le i \le m} M_i'$.

As much as our attacks are concerned, OCB1–3 and TAE are quite similar to IAPM, and the attacks are the same.

More One-Pass Modes: IACBC [26] and XCBC [17]. Given a nonce V and a secret key K, IACBC [26] encrypts $(M_1, \ldots, M_m)$ to $(C_1, \ldots C_m)$ and an authentication tag T as follows.

Initial step: Generate $m+1$ values $s_0, s_1, \ldots s_m$ depending on V and K, but not on the plaintext $(M_1, \ldots, M_m)$.
Encryption: $x_0 := V$; For $i \in \{1, \ldots, m\}$: $x_i := E_K(M_i \oplus x_{i-1})$, $C_i := x_i \oplus s_i$.
Authentication tag: $T := E_K(x_m \oplus \sum_{1 \leq i \leq m} M_i) \oplus s_0$.

Note that under nonce-reuse the authentication tag leaks information about the message. Namely, if we encrypt two messages $(M_1, \ldots, M_m)$ and $(M'_1, \ldots, M'_m)$ of the same length m under the same nonce, the two authentication tags are the same if and only if $\sum_{1 \leq i \leq m} M_i = \sum_{1 \leq i \leq m} M'_i$.

The following nonce-reuse attack distinguishes IACBC encryption from an online permutation and also provides an existential forgery. For simplicity, we only consider 1-block messages $V \neq W$, which we also use as nonces:

1. Encrypt W under V to (C_1, T).
2. Encrypt V under W to (C'_1, T').
3. Encrypt V under V to (C''_1, T'').
4. Set $C'''_1 := C_1 \oplus C'_1 \oplus C''_1$ and $T''' := T \oplus T' \oplus T''$.
   (C'''_1, T) is a valid encryption of W under W.

Given a nonce V and secret keys K and K', XCBC encrypts a plaintext $(M_1, \ldots, M_m)$ to a ciphertext $(C_1, \ldots C_m)$ and an authentication tag T as follows.

Initial step: Generate $m+1$ values $s_1, \ldots s_{m+1}$ depending on V and K, but not on the plaintext $(M_1, \ldots, M_m)$.
Encryption:
   1. $C_0 := E_K(V)$; $x_0 := E_{K'}(V)$;
   2. Generate an additional message word $M_{m+1} := x_0 \oplus M_1 \oplus \cdots \oplus M_m$ for authentication.
   3. For $i \in \{1, \ldots, m+1\}$: $x_i := E_K(M_i \oplus x_{i-1})$, $C_i := (x_i + s_i) \bmod 2^n$.

The best attack we have found for XCBC is not quite as baneful as the attacks on the other schemes, as the attack workload is at $O(2^{n/4})$, and the attack only provides a distinguisher, not a forger. For this reuse-nonce chosen-plaintext attack, we ignore the authentication tag:

1. Generate $2^{n/4}$ encryptions of messages M^i_1 under a nonce V to C^i_1.
   Statistically, expect one pair $i \neq j$ such that the least significant $n/2$ bits of C^i_1 are identical to the least significant $n/2$ bits of C^j_1.
2. Generate $2^{n/4}$ encryptions of messages (M^i_1, M^k_2) and (M^j_1, M^ℓ_2) under V to (C^i_1, C^k_2) and (C^j_1, C^ℓ_2), where the least significant $n/2$ bits of M^k_2 and M^ℓ_2 are the same.
   Statistically, expect one pair $k \neq \ell$ such that $C^k_2 = C^\ell_2$ holds.
3. Choose an arbitrary M_3. Encrypt (M^i_1, M^k_2, M_3) and (M^j_1, M^ℓ_2, M_3) under V to $(C^i_1, C^k_2, C^{i,k}_3)$ and $(C^j_1, C^\ell_2, C^{j,\ell}_3)$.
   Observe $C^{i,k}_3 = C^{j,\ell}_3$.

A.2 Dedicated Online Ciphers and Authenticated Encryption

The current paper refers to online ciphers to define the privacy of AOE against general adversaries. Online ciphers have first been studied in [1]. The eprint version of the same paper describes three AE modes for online cipher, using a random nonce V:

1. Prepend V to the plaintext, append redundancy (e.g., a fixed number of 0-bits).
2. Prepend V and then the length of the plaintext, append redundancy.

3. Prepend a random value V as the IV, and append the same V.

The first is secure if and only if the message length is fixed. The second is an obvious repair of the first, but the exact plaintext length must be known at the beginning of the encryption process. This, though using an underlying online cipher, the scheme itself isn't online. The third is vulnerable to a chosen plaintext attack using a message $M||V$, if the adversary can guess V.

A.3 Offline Schemes, Defeating Nonce-Reuse (SIV [43], HBS [25], BTM [24])

Given a nonce N, a message M and associated Data H, these schemes perform two steps:

1. Generate the authentication tag T from H, M, and N.
2. Encrypt M in counter mode, using T as the nonce.

This is inherently offline, because one must finish step 1 before one can start step 2. All of SIV, HBS, and BTM perform counter mode encryption, but employ different MAC schemes to generate the tag T.

This usage of the counter mode is vulnerable in an *on-line decryption misuse* case, where, during decryption, a would-be plaintext is compromised before the tag has been verified. A chosen-ciphertext adversary can exploit that to determine an unknown keystream and then to decrypt an unknown message.

Another misuse case may apply when nonce-reuse is possible and the sender reads the message twice, once for each of the two steps – if there is any chance that *the message has been modified* between the two read operations.

Note that both misuse cases become quite harmless if one replaces the counter mode encryption by the application of an on-line permutation.

B Proof of Theorem 1

Consider games G_0, G_1, G_2 of Figure 14. For a fixed CCA3(ω,ν) adversary A on the scheme Π it holds that

$$\begin{aligned}\Pr[A_{\Pi}^{\mathrm{CCA3}(\omega,\nu)} \Rightarrow 1] &= \Pr[A^{G_0} \Rightarrow 1] \\ &= \Pr[A^{G_1} \Rightarrow 1] + (\Pr[A^{G_0} \Rightarrow 1] - \Pr[A^{G_1} \Rightarrow \mathtt{true}]) \\ &\leq \Pr[A^{G_1} \Rightarrow 1] + \Pr[A^{G_1} \mathit{sets}\ \mathtt{bad}].\end{aligned}$$

Since the `Decrypt` oracles of G_1 and G_2 always return $\perp$,

$$\Pr[A^{G_1} \Rightarrow 1] = \Pr[A^{G_2} \Rightarrow 1].$$

Now, we design two adversaries A_c and A_p so that

$$\begin{aligned}&\Pr[A^{G_1} \mathit{sets}\ \mathtt{bad}] \leq \Pr[A_c{}_{\Pi}^{\text{INT-CTXT}(\omega,\nu)} \Rightarrow 1] \text{ and} \\ &\Pr[A^{G_2} \Rightarrow 1] \leq \Pr[A_p{}_{\Pi}^{\mathrm{CPA}(\omega,\nu)} \Rightarrow 1].\end{aligned}$$

A_p: Adversary A_p simply runs A answering A's **Encrypt** queries using its own **Encrypt** oracle, and answers **Decrypt** queries with $\perp$. A_p outputs whatever A outputs.

A_c: Adversary A_c runs A answering A's **Encrypt** queries using its own **Encrypt** oracle. It submits A's **Decrypt** queries to it's **Verify** oracle (*cf.* Figure 8) and, regardless of the response, returns $\perp$. Note that the **Verify** oracle sets `win` to **true** if and only if a fresh **Decrypt** query is valid. Just such a query would set the variable `bad` to **true**. □

```
Initialize(ω, ν)
    b ←$ {0,1};
    if(b=1) then
        K ← 𝒦();

Finalize(d)
    return (d=b);
```

```
Encrypt(H,M)
    if (ν = NR and V ∈ B) then
        return ⊥;
    else
        B ← B ∪ {V};
    if(b=1) then
        C ← ℰ_K(H,M);
    else
        C ← $^ω(H,M);
    𝒬 ← 𝒬 ∪ {(H, C)};
    return C;
```

```
Decrypt(H,C)            Game G_0, [G_1]
    M ← ⊥;
    if ((H, C) ∉ 𝒬 and b=1) then
        M ← 𝒟_K(H,C);
    if(M≠ ⊥) then
        bad ← true; [M ← ⊥;]
    return M;

Decrypt(H,C)            Game G_2
    return ⊥;
```

Fig. 14. Games G_0, G_1 and G_2 for the proof of Theorem 1. Game G_1 contains the code in the box while G_0 does not. H_0 denotes the first block of the header representing the nonce/initial value.