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

Ewan Fleischmann, Christian Forler, and Stefan Lucks

Bauhaus-University Weimar, Germany
{ewan.fleischmann,christian.forler,stefan.lucks}@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 considers OAE schemes dealing both with nonce-respecting *and* with general adversaries. It introduces McOE, an efficient design for OAE schemes. For this we present in detail one of the family members, McOE-X, which is a design solely based on a standard block cipher. As all the other member of the McOE family, it provably guarantees reasonable security against general adversaries as well as standard security against nonce-respecting adversaries.

Keywords: authenticated encryption, on-line 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 [4]) and five dedicated AE schemes: OCB2 [38], SIV [41] (denoted as "Key Wrap" in [21]), CCM [13], EAX [6], 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

[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.

A. Canteaut (Ed.): FSE 2012, LNCS 7549, pp. 196–215, 2012.

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 [37,39,40] 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.

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 [9,20,28,44,45]. Apart from implementation failures, there are fundamental reasons why software developers can't 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 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, 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 don't include it in our analysis.

As it turned out, we actually found nonce-reuse attacks for *all* of those schemes, cf. Table 2, Appendix A, and, especially, Appendix 1 in the full version of this paper [14]. We present a new construction method for efficient AE schemes, called McOE-X, that is actually able to fill the apparent gap in the upper-right. It belongs to the family of McOE schemes [14]. We argue that closing this gap is both practically relevant and theoretically interesting.

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.

secure ...	**against nonce-respecting adversaries**	**ag. nonce-reusing adversaries**
on-line	CCFB[33] CHM[22] CIP[23] CWC[29] EAX[6] GCM[34] IACBC[26] IAPM[26] **McOE** OCB1-3[40,38,30] RPC[10] TAE[31] XCBC[17]	**McOE** (this paper)
off-line	BTM[24] CCM[13] HBS[25] SIV[41] SSH-CTR[36]	BTM[24] HBS[25] SIV[41]

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. An "attack workload" of X means that the adversary is restricted to at most X units of time and at most X chosen texts. Details are given in Appendix A and in the full version of this paper [14].

	privacy attack workload	authenticity attack workload
CCFB [33]	$O(1)$	$O(1)$
CCM [13]	$O(1)$	$\ll 2^{(n/2)}$ [15]
CHM [22]	$O(1)$	$O(1)$
CIP [23]	$O(1)$	$O(1)$
CWC [29]	$O(1)$	$O(1)$
EAX [6]	$O(1)$	$O(1)$
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 [40]	$O(1)$	$O(1)$
OCB2 [38]	$O(1)$	$O(1)$
OCB3 [30]	$O(1)$	$O(1)$
RPC [10]	$O(1)$	$O(1)$
TAE [31]	$O(1)$	$O(1)$
XCBC [17]	$O(2^{n/4})$	?

Initial Value (IV) based AE schemes maximally forgiving of repeated IV's have been addressed in [41], coining 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. the Appendix of the full version of this paper [14]. 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 can't even decide if $P_1 = P_2$ or not. When using a nonce more than once, deciding about $P_1 = P_2$ is 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. can't 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, [43] studied 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 [43] 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 [3], the revised and full version of [1]. The first idea in [3] 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 a value τ which is *key-dependent*, 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)*.

Roadmap. In this paper we focus on one member of the McOE [14] family of schemes called McOE-X. In Section 2 we describe a concrete block cipher based OAE scheme – called McOE-X– and provide performance data when McOE-X is 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 full McOE-X scheme and its analysis, is presented in Section 5. The discussion in Section 6 concludes the paper. The appendix deals with misuse attacks against published AE schemes.

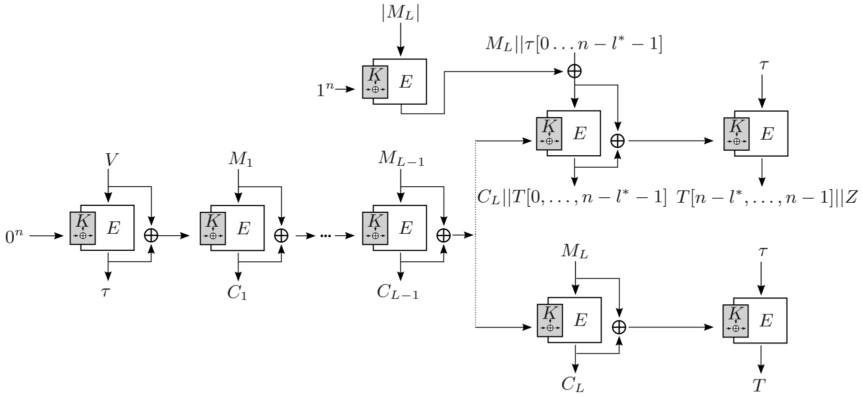

Fig. 1. The McOE-X-AES/McOE-X-Threefish encryption process. If, after the last complete message block has been encrypted, there is some incomplete block left, McOE-X performs tag-splitting (upper variant), Else, the tag can be computed without splitting (lower variant). The key used for the block cipher E is computed by the injective function $K \oplus W$ which is given the secret key K and the chaining value input W. 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 computing τ.

2 Practical On-Line Authenticated Encryption Using AES and Threefish

We start with the fruits of our analysis by giving two concrete instances of OAE schemes (illustrated in Figure 1) including performance data and reference source code[2]. One instance, McOE-X-AES uses AES-128 as the core component while McOE-X-Threefish uses the block cipher Threefish-512, a cipher with 512-bit block size and key size, which is the core working component inside the SHA-3 finalist Skein[35]. 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. The effect of TS is similar to the well-known length preserving method called *ciphertext stealing* (CTS), *e.g.* [12]. But the technique itself is quite different since CTS requires to process the last block before the last but one, which is not possible for McOE-X.

Let E_K be a block cipher taking a k-bit key K and a plaintext/ciphertext of size n-bit. Note that for our chosen instances, AES-128 and Threefish-512, we have $n = k$. The pseudo code for these two McOE-X instances is given in Table 4 – on the upper side without TS, on the lower side with TS.

[2] The reference source code is available on request; it will be published as open source.

Table 3. Performance values (cycles-per-byte, single core), measured on an 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[11]. The Threefish implementation is based on the NIST/SHA-3 reference source as provided by the Skein authors [35]. Finally, the implementation of Galois field NI multiplication (GF-NI) is based on the example-code from [19].

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

The algorithms without TS, **EncryptAuthenticate** and **DecryptAuthenticate**, are simplified algorithms for messages that are aligned on n-bit boundaries, *i.e.* $M = (M_1, \ldots, M_L) \in (\{0,1\}^n)^L$ for some integer L. The TS algorithms are **EncryptAuthenticateSplitTag** and **DecryptAuthenticate-SplitTag**. they can handle arbitrarily sized messages, *i.e.*, $M = (M_1, \ldots, M_L) \in (\{0,1\}^n)^{L-1} || \{0,1\}^{l^*}$ where L and l^* are integers with $0 < l^* < n$ and $'||'$ denotes the string concatenation operator. See Figure 1 and Table 4.

In addition to McOE-X, we introduce two further authenticated encryption schemes following the McOE design principles. The first one is called McOE-D and is based on the THC-CBC construction [7]. The ratio of this scheme is 2-1, *i.e.* the block cipher is invoked twice to encipher resp. decipher one message block. The second one is called McOE-G and is based on the HCBC-2 construction [2]. This scheme updates the chaining value by invoking a universal hash function, *i.e.*, a n-bit Galois-Field multiplication.

Remarks. For McOE-X we actually do need related key resistance for the block cipher E since the adversary can 'partially control' some relations among keys used in the computation. This is not true for the other mentioned constructions.

All McOE schemes are easily extended to smoothly handle associated data, *i.e.* data that is not encrypted but only authenticated. This is discussed in more detail in Section 5.

Table 4. Instances of McOE-X: upper side is for messages whose size is evenly divisible by the block size n; Lower side is for arbitrarily sized messages (TS-variant); see text for details

```
EncryptAuthenticate$(V, M)$
$\tau \leftarrow E_K(V)$
$U \leftarrow V \oplus \tau \oplus K$
for $i = 1, \ldots, L$ loop
    $C_i \leftarrow E_U(M_i)$
    $U \leftarrow M_i \oplus C_i \oplus K$
$T \leftarrow E_U(\tau)$
return $(C_1, \ldots, C_L, T)$
```

```
DecryptAuthenticate$(V, C, T)$
$\tau \leftarrow E_K(V)$
$U \leftarrow V \oplus \tau \oplus K$
for $i = 1, \ldots, L$ loop
    $M_i \leftarrow E_U^{-1}(C_i)$
    $U \leftarrow M_i \oplus C_i \oplus K$
if $T = E_U(\tau)$ then
    return $(M_1, \ldots, M_L)$
    else return $\perp$
```

```
EncryptAuthenticateSplitTag$(V, M)$
$\tau \leftarrow E_K(V)$
$U \leftarrow V \oplus \tau \oplus K$
for $i = 1, \ldots, L - 1$ loop
    $C_i \leftarrow E_U(M_i)$
    $U \leftarrow M_i \oplus C_i \oplus K$
$M^* \leftarrow (M_L || \tau[0 \ldots n - l^* - 1])$
$M^* \leftarrow M^* \oplus E_{K \oplus 1^n}(|M_L|)$
$C^* \leftarrow E_U(M^*)$
Parse $C_L || T[0 \ldots n - l^* - 1] \leftarrow C^*$
$U \leftarrow M^* \oplus C^* \oplus K$
$C^{**} \leftarrow E_U(\tau)$
$T[n - l^* \ldots n - 1] \leftarrow C^{**}[0 \ldots l^* - 1]$
return $(C_1, \ldots, C_{L-1}, C_L^*, T)$
```

```
DecryptAuthenticateSplitTag$(V, C, T)$
$\tau \leftarrow E_K(V)$
$U \leftarrow V \oplus \tau \oplus K$
for $i = 1, \ldots, L - 1$ loop
    $M_i \leftarrow E_U^{-1}(C_i)$
    $U \leftarrow M_i \oplus C_i \oplus K$
$C^* \leftarrow C_L || T[0 \ldots n - l^* - 1]$
$M^* \leftarrow E_U^{-1}(C^*)$
$U \leftarrow M^* \oplus C^* \oplus K$
$M^* \leftarrow M^* \oplus E_{K \oplus 1^n}(|C_L|)$
Parse $M_L || \tau'[0 \ldots n - l^* - 1] \leftarrow M^*$
$T' \leftarrow E_U(\tau)$
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 $\perp$
```

3 On-Line Authenticated Encryption and Related Notions

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

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

Let $\mathcal{Q}$ a non-empty set of strings in D_n^*,. Then we define $\mathrm{LLCP}_n(\mathcal{Q}, P)$ as $\max_{q \in \mathcal{Q}}\{\mathrm{LLCP}_n(q, P)\}$, *e.g.*, 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.* If $\mathcal{Q} = \{0,1\}^a \times \{0,1\}^b \times \{0,1\}^c$, we write $\mathcal{Q}_{|b,c} = \{(B, C) \mid \exists A : (A, B, C) \in \mathcal{Q}\}$. This generalizes in the obvious way.

3.1 Block Ciphers and On-Line Permutations

Block Ciphers. An (k, n) block cipher is a keyed family of permutations consisting of two paired algorithms $E : \{0,1\}^k \times D_n \to D_n$ and $E^{-1} : \{0,1\}^k \times D_n \to D_n$, 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 $K \in \{0,1\}^k$, the decryption $E_K^{-1}(Y) := E^{-1}(K, Y)$ is the inverse function of encryption $E_K(X) := E(K, X)$, 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 1. *Let $E \in Block(k, n)$ and denote by E^{-1} the corresponding inverse. Let $\varphi : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^k$. 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 its inverse. 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. The related-key (RK) advantage [32] of A in breaking E is then defined as*

$$\begin{aligned}\mathbf{Adv}_E^{\text{RK-CPA-PRP}}(A) &= |\Pr[K \xleftarrow{\$} \{0,1\}^k : A^{E_{\varphi(K,\cdot)}(\cdot)} \Rightarrow 1] \\ &\quad - \Pr[\pi \xleftarrow{\$} \mathit{Perm(n,n)} : A^{\pi(\cdot,\cdot)} \Rightarrow 1]| \\ \mathbf{Adv}_{E,E^{-1}}^{\text{RK-CCA-PRP}}(A) &= |\Pr[K \xleftarrow{\$} \{0,1\}^k : A^{E_{\varphi(K,\cdot)}(\cdot), E^{-1}_{\varphi(K,\cdot)}(\cdot)} \Rightarrow 1] \\ &\quad - \Pr[\pi \xleftarrow{\$} \mathit{Perm(n,n)} : A^{\pi(\cdot,\cdot), \pi^{-1}(\cdot,\cdot)} \Rightarrow 1]|.\end{aligned}$$

On-Line Permutations. 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.

Definition 2. *Let $n, k \geq 0$, $K \in \{0,1\}^k, V \in D_n$. A function $\Pi : \{0,1\}^k \times D_n^* \to D_n^*$ is an (n-)on-line permutation if for any fixed K, V the function $\Pi(K, V, \cdot)$ is a permutation and there exists for any message $M = (M_1, \ldots, M_m)$ a family of functions $\widetilde{\pi}^i : \{0,1\}^k \times \{0,1\}^n \times D_n^i \to D_n$, $i = 1, \ldots, m$ such that*

$$\begin{aligned}\Pi(K, V, M) = \;& \widetilde{\pi}_K^1(V, M_1) || \widetilde{\pi}_K^2(V, M[1..2]) \\ & || \ldots || \\ & \widetilde{\pi}_K^{m-1}(V, M[1..m-1]) || \widetilde{\pi}_K^m(V, M[1..m]),\end{aligned}$$

where $M[a \ldots b] := M_a || M_{a+1} || \ldots || M_b$ with "$||$" 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].

3.2 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 $\{0,1\}^k$. The encryption algorithm $\mathcal{E}$ and the decryption algorithm $\mathcal{D}$ are deterministic algorithms that map values from $\{0,1\}^k \times \mathcal{H} \times D_n^*$ to a string or – if the input is invalid – the value $\perp$. The header $\mathcal{H}$ 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 $\mathcal{H} \subset D_n^+$ 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^*, $H \in \mathcal{H}$ and a key from the key space. We require $\mathcal{D}_K^{\mathcal{H}}(\mathcal{E}_K^{\mathcal{H}}(M)) = M$ for any possible K, M, H, and define the tag size for a message $M \in D_n^*$ and header $H \in \mathcal{H}$ 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 [42] and nonce based [4,5,27,37,40] AE schemes. In order to have one convenient toolset of notions, we adopt the notion of CCA3 security suggested in [42] 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, $\nu \in \{\textsc{nr}, \textsc{ni}\}$.

Definition 3 (CCA3(ω, ν)). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme with header space $\mathcal{H}$ and message space D_n^*, and fix an adversary A. The advantage of A breaking Π is defined as*

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

The adversary's random-bits oracle, $\$^{\textsc{ae}}(\cdot,\cdot)$ or $\$^{\textsc{oae}}(\cdot,\cdot)$, returns on a query with header $H \in \mathcal{H}$ 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

```
Game $G_{\mathrm{CPA}}$, $\boxed{G_{\mathrm{CCA3}}}$

Initialize($\omega, \nu$)
    b $\xleftarrow{\$}$ {0,1};
    if(b=1) then
        K $\leftarrow \mathcal{K}()$;

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

Encrypt($H, M$)
    if ($\nu = \mathrm{NR}$ and $V \in B$) then
        return $\perp$;
    if(b=1) then
        $C \leftarrow \mathcal{E}_K$(H,M);
    else
        $C \leftarrow \$^{\omega}(H, M)$;
    $B \leftarrow B \cup \{V\}$;
    $\boxed{\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(H, C)\};}$
    return C;

Decrypt(H,C)
    if $((H, C) \in Q)$ then
        return $\perp$;
    if(b=1) then
        M $\leftarrow \mathcal{D}_K$(H,C);
    else
        M $\leftarrow \perp$(H,C);
    return M;
```

Fig. 2. $G_{\mathrm{CPA}}(\omega, \nu)$ is the $\mathrm{CPA}_{\Pi}^{(\omega,\nu)}$-Game and $G_{\mathrm{CCA3}}(\omega, \nu)$ the $\mathrm{CCA3}_{\Pi}^{(\omega,\nu)}$-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 $\$^{\mathrm{AE}}(H, M)$ returns a string of length $|M| + \mathrm{TAG}(H, M)$, this string is on-line compatible if $\omega = \mathrm{OAE}$. V denotes the last block of the header representing the nonce/initial value.

answer is already known. It is easy to see that we can rewrite the term given in Definition 3 as

$$\Big|\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right] \tag{1}$$

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

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 3 using the game G_{CCA3} given in Figure 2 as

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

We denote $\mathbf{Adv}_{\Pi}^{\mathrm{CCA3}(\omega,\nu)}(q, t, \ell)$ as the maximum advantage over all $\mathrm{CCA3}(\omega, \nu)$ 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 $\mathrm{TAG}(H, M)$ as follows.

Definition 4. *Let $G_{CPA}(\omega, \nu)$ be the $CPA_{\Pi}^{\omega,\nu}$ game given in Figure 2. Fix an adversary A. The advantage of A breaking Π is defined as*

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

```
Game $G_{INT-CTXT}$

Initialize($\nu$)
    $K \leftarrow \mathcal{K}()$;

Finalize()
    return win;

Encrypt(H,M)
    if ($\nu = $ NR and
            $V \in B$) then
        return $\bot$;
    $C \leftarrow \mathcal{E}_K$(H,M);
    $B \leftarrow B \cup \{V\}$;
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(H,C)\}$;
    return $C$;

Verify(H,C)
    M $\leftarrow \mathcal{D}_K$(H,C);
    if $((H,C) \notin Q$
            and $M \neq \bot$) then
        win $\leftarrow$ true;
    return $(M \neq \bot)$;
```

Fig. 3. 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 5. *Let $G_{\text{INT-CTXT}}(\nu)$ be the* INT-CTXT$_{\Pi}^{\nu}$ *game given in Figure 3. 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 [4]. It simply states the equivalence of a scheme being CCA3 secure and both INT-CTXT and CPA secure. These statements hold in the on-line and offline 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 the full version of this paper [14].

5 The On-Line Authenticated Encryption Scheme McOE-X

In this section, we present McOE-X, a construction for an OAE scheme. We prove that McOE-X achieves our two-fold goal. First, it guarantees a certain minimum, well defined, security against a nonce-ignoring adversary. And, second, we show – in the full version of the paper [14] – that the complete McOE family of OAE schemes (including McOE-X) is fully secure against a nonce-respecting adversary.

Since we already have presented two McOE-X instances in Section 2, we proceed by formally defining McOE-X and giving its pseudocode. Indeed this is very similar to the results presented in Section 2, but here our definitions are slightly more general. Instead of fixing the key computation function to $K \oplus V$, where R is the chaining value and K the secret key, we here use a key derivation function $\varphi(K, R)$. By this we make sure that our proof also works for tweakable block ciphers - with K as key and R as tweak - leading to more efficient design.

Definition 6 (McOE-X). *Let $k, n \in \mathbb{N}$ with $k \geq n$, $E \in \texttt{Block}(k, n)$, and $\varphi : \{0,1\}^k \times \{0,1\}^v \to \{0,1\}^k$ such that $\varphi(K, \cdot)$ is injective. The encryption function takes a header $H \in D_n^{L_H}$, a message M and returns a ciphertext C and a tag $T \in D_n$. The decryption function takes a header $H \in D_n^{L_H}$, a ciphertext C and a tag $T \in D_n$ and returns either a plaintext M or the fail symbol $\perp$.*

(i) 'Non-TS'. *Let $M, C \in D_N^L$ for some integer L, then* McOE-X *is defined by the algorithms* ***EncryptAuthenticate*** *and* ***DecryptAuthenticate*** *given in Table 5.*

(ii) 'TS'. *Let $M, C \in D_N^L || \{0,1\}^{l^*}$ for some integers L and l^*, $0 < l^* < n$, then* McOE-X*/TS is defined by the algorithms* ***EncryptAuthenticateSplitTag*** *and* ***DecryptAuthenticateSplitTag*** *given in Table 5.*

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

Theorem 2

(i) *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a* McOE-X *scheme as in Definition 6 (i). We further assume that the block cipher E is secure against related key attacks. Then*

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

(ii) *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a* McOE-X *scheme as in Definition 6 (ii). We further assume that the block cipher E is secure against related key attacks. Then*

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

Proof. The proof of (i) follows from Theorem 1 together with Lemmas 1 and 2. Due to the lack of of space the proof of (ii) it is skipped here and is available in the full version of the paper [14].

Table 5. Instances of McOE-X: Left side is for messages whose size is evenly divisible by the block size n; Right side is for arbitrarily sized messages (TS-variant); see text for details

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

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

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

```
DecryptAuthenticateSplitTag($H, C, T$)
 1. $U \leftarrow \varphi(K, 0^n)$
 2. for $i = 1, \ldots, L_H - 1$ do
        $U \leftarrow \varphi(K, H_i \oplus E_U(H_i))$
 3. $\tau \leftarrow E_U(H_{L_H})$
 4. $U \leftarrow \varphi(K, H_{L_H} \oplus \tau)$
 5. for $i = 1, \ldots, L$ do
        $M_i \leftarrow E_U^{-1}(C_i)$
        $U \leftarrow \varphi(K, M_i \oplus C_i)$
 6. $C^* \leftarrow C_{L+1} || T[0 \ldots n - l^* - 1]$
 7. $M^* \leftarrow E_U^{-1}(C^*)$
 8. $U \leftarrow \varphi(K, M^* \oplus C^*)$
 9. $M^* \leftarrow M^* \oplus E_{K \oplus 1^n}(|C_L|)$
10. Parse $M_L || \tau'[0 \ldots n - l^* - 1] \leftarrow M^*$
11. $T' \leftarrow E_U(\tau)$
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 $(M_1, ..., M_L)$ else
    return $\perp$
```

Lemma 1. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a* McOE-X *scheme as in Definition 6 (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 in blocks of the queries to $\mathcal{E}$ and $\mathcal{D}$. Then,*

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

Proof (Lemma 1). Our bound is derived by game playing arguments. Consider games G_1-G_3 of Figure 4 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 3. Definition 5 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 specific McOE-X counterparts as of Definition 6. 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 is initialized to $\{\varphi(K, 0^n)\}$ and then collects new key-input values U which are computed during the encryption or verification process (in lines 204, 207, 213, 223, 226, 232 and 237). We note that, since φ is injective, a collision for the chaining values follows if there is a collision in the U values.

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[3]. It follows that

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

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 (i) either have 'found' a collision for $E_{\varphi(K,X)}(Y) \oplus Y$, *i.e.* she stumbles over (X, Y) and (X', Y') such that $E_{\varphi(K,X)}(Y) \oplus Y = E_{\varphi(K,X')}(Y') \oplus Y'$ or, (ii), must have found a preimage of $\varphi(K, 0^n)$, which is always the starting point of our chain. Note that that value $\varphi(K, 0^n)$ is initially stored in the set B. In both cases, the variable `bad` would have been set to `true`, and it follows [8] that

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

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

```
**Initialize**()
    $K \xleftarrow{\$} \mathcal{K}()$;
    $B \leftarrow \{\varphi(K, 0^n\}$;
```

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

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

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

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

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

Fig. 4. Games G_1-G_3 for the proof of Lemma 1. Game G_3 contains the code in the box while G_2 does not.

We now describe the new game G_4. It is equal to G_3 *except* that the block cipher E and its inverse 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 $E_K(A)$ is replaced by an invocation of $\mathbf{EncryptBlock}_K(A)$ and the call $E_K^{-1}(A)$ is replaced by an invocation of $\mathbf{DecryptBlock}_K(A)$. We now upper bound the difference between G_3 and G_4.

So, by definition of G_4, we have

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

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 E or E^{-1}) such that we always have enough randomness left for our result.

Case 1: *H has already been used in an Encrypt or Verify query before and $U \in B$.* Since we already have computed τ in the past, the chance of success is upper bounded by the probability $\Pr[E_U^{-1}(T) = \tau]$ which can be upper bounded by $1/(2^n - (q+\ell))$.

Case 2: *H has never been used before, also U has never been used as a chaining value.* Then the tagging operation uses a 'new key' – essentially due since φ is injective – and therefore the output of $E_U(\tau)$ is uniformly distributed and the success probability is $\leq 1/2^n$.

Case 3: *$H \in A$ but U has never been used as a chaining value.* The chance of success is upper bounded by $\Pr[E_U^{-1}(T) = \tau]$ which can be upper bounded by $1/2^n$.

Note that 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. □

Lemma 2. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a* McOE-X *scheme as in Definition 6 (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,*

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

The proof is given in the full version of this paper [14].

6 Discussion

New Challenges for Research. At the 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 ordinary 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 that 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 don't 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 as, 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.

Acknowledgments. We like to thank Jakob Wenzel for very helpful comments, Phil Rogaway for making us aware of the Galois field native instructions, and the participants of the Dagstuhl Seminar on Symmetric Cryptography 2012 for inspiring discussions.

References

1. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: Online Ciphers and the Hash-CBC Construction. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 292–309. Springer, Heidelberg (2001)
2. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: On-Line Ciphers and the Hash-CBC Constructions. IACR Cryptology ePrint Archive, 2007:197 (2007)

3. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: Online Ciphers and the Hash-CBC Construction. Cryptology ePrint Archive, Report 2007/197; full version of [1] (2007), http://eprint.iacr.org/
4. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. J. Cryptology 21(4), 469–491 (2008)
5. Bellare, M., Rogaway, P.: Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 317–330. Springer, Heidelberg (2000)
6. Bellare, M., Rogaway, P., Wagner, D.: The EAX Mode of Operation. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 389–407. Springer, Heidelberg (2004)
7. Black, J.A., Cochran, M., Shrimpton, T.: On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 526–541. Springer, Heidelberg (2005)
8. Black, J.A., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 320. Springer, Heidelberg (2002)
9. Borisov, N., Goldberg, I., Wagner, D.: Intercepting Mobile Communications: The Insecurity of 802.11. In: MOBICOM, pp. 180–189 (2001)
10. Buonanno, E., Katz, J., Yung, M.: Incremental Unforgeable Encryption. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 109–124. Springer, Heidelberg (2002)
11. Intel Corporation. AES-NI Sample Library v1.2 (2010), http://software.intel.com/en-us/articles/download-the-intel-aesni-sample-library/
12. Daemen, J.: Hash Function and Cipher Design: Strategies Based on Linear and Differential Cryptanalysis. Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium (March 1995)
13. Dworkin, M.: 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)
14. Fleischmann, E., Forler, C., Lucks, S.: McOE: A Foolproof On-Line Authenticated Encryption Scheme. IACR Cryptology ePrint Archive, 2011:644 (2011)
15. Fouque, P.-A., Martinet, G., Valette, F., Zimmer, S.: On the Security of the CCM Encryption Mode and of a Slight Variant. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 411–428. Springer, Heidelberg (2008)
16. Gladman, B.: Brian Gladman's AES Implementation (June 19, 2006), http://gladman.plushost.co.uk/oldsite/AES/index.php
17. Gligor, V.D., Donescu, P.: Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 92–108. Springer, Heidelberg (2002)
18. Goldwasser, S., Micali, S.: Probabilistic Encryption. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
19. Gueron, S., Kounavis, M.E.: Efficient implementation of the Galois Counter Mode using a carry-less multiplier and a fast reduction algorithm. Inf. Process. Lett. 110(14-15), 549–553 (2010)
20. Hotz, G.: 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. Iwata, T.: New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 310–327. Springer, Heidelberg (2006)
23. Iwata, T.: Authenticated Encryption Mode for Beyond the Birthday Bound Security. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 125–142. Springer, Heidelberg (2008)
24. Iwata, T., Yasuda, K.: BTM: A Single-Key, Inverse-Cipher-Free Mode for Deterministic Authenticated Encryption. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 313–330. Springer, Heidelberg (2009)
25. Iwata, T., Yasuda, K.: HBS: A Single-Key Mode of Operation for Deterministic Authenticated Encryption. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 394–415. Springer, Heidelberg (2009)
26. Jutla, C.S.: Encryption Modes with Almost Free Message Integrity. J. Cryptology 21(4), 547–578 (2008)
27. Katz, J., Yung, M.: Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 284–299. Springer, Heidelberg (2001)
28. Kohno, T.: Attacking and Repairing the WinZip Encryption Scheme. In: ACM Conference on Computer and Communications Security, pp. 72–81 (2004)
29. Kohno, T., Viega, J., Whiting, D.: CWC: A High-Performance Conventional Authenticated Encryption Mode. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 408–426. Springer, Heidelberg (2004)
30. Iwata, T.: New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 310–327. Springer, Heidelberg (2006)
31. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable Block Ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002)
32. Lucks, S.: Ciphers Secure against Related-Key Attacks. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 359–370. Springer, Heidelberg (2004)
33. Lucks, S.: Two-Pass Authenticated Encryption Faster Than Generic Composition. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 284–298. Springer, Heidelberg (2005)
34. McGrew, D.A., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer, Heidelberg (2004)
35. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: Skein source code and test vectors, `http://www.skein-hash.info/downloads`
36. Paterson, K.G., Watson, G.J.: Plaintext-Dependent Decryption: A Formal Security Treatment of SSH-CTR. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 345–361. Springer, Heidelberg (2010)
37. Rogaway, P.: Authenticated-Encryption with Associated-Data. In: ACM Conference on Computer and Communications Security, pp. 98–107 (2002)
38. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer, Heidelberg (2004)
39. Rogaway, P.: Nonce-Based Symmetric Encryption. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 348–359. Springer, Heidelberg (2004)

40. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: ACM Conference on Computer and Communications Security, pp. 196–205 (2001)
41. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006)
42. Rogaway, P., Shrimpton, T.: Deterministic Authenticated-Encryption: A Provable-Security Treatment of the Key-Wrap Problem. Cryptology ePrint Archive, Report 2006/221; full version of [41] (2006), http://eprint.iacr.org/
43. Rogaway, P., Zhang, H.: Online Ciphers from Tweakable Blockciphers. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 237–249. Springer, Heidelberg (2011)
44. Sabin, T.: Vulnerability in Windows NT's SYSKEY encryption. BindView Security Advisory (1999), http://marc.info/?l=ntbugtraq&m=94537191024690&w=4
45. Wu, H.: 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

We now give a short overview on one of the attack patterns we have successfully used (cf. Table 2). A more detail led analysis (including more attack patterns) can be found in the full version of this paper [14].

Cipher-block-chaining (CBC) is an unauthenticated encryption mode which is sometimes used as the encryption component of an AE scheme. It is well known that, for constant nonces, the ciphertext of two different plaintexts do reveal the full keystream. 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, as was already indicated in Table 2 using the following 'linear tag' attack pattern. Schemes susceptible to this attack are CWC [29], GCM [34], EAX [6], and CHM [22].

Linear Tag Attack. Assume an AE scheme which generate a keystream $S = F_K(V)$ depending on a secret key K and a nonce V encryption a message M by computing a ciphertext $C = S \oplus M$. For AE schemes using the encrypt-then-authenticate paradigm, we 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 following attack:

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