ABC - A New Framework for Block Ciphers

Uri Avraham[1] Eli Biham[1] Orr Dunkelman[2]

[1]Computer Science Department,
Technion - Israel Institute of Technology,
Haifa 32000, Israel.
Email: uria@cs.technion.ac.il, biham@cs.technion.ac.il
[2]Computer Science Department,
The Weizmann Institute of Science,
Rehovot 76100, Israel.
Email: orr.dunkelman@weizmann.ac.il

Abstract. We suggest a new framework for block ciphers named Advanced Block Cipher, or shortly ABC. ABC has additional non-secret parameters that ensure that each call to the underlying block cipher uses a different pseudo-random permutation. It therefore ensures that attacks that require more than one block encrypted under the same secret permutation cannot apply. In particular, this framework protects against dictionary attacks, and differential and linear attacks, and eliminates weaknesses of ECB and CBC modes. This new framework shares a common structure with HAIFA [3], and can share the same logic with HAIFA compression functions. We analyze the security of several modes of operation for ABCs block ciphers, and suggest a few instances of ABCs.

1 Introduction

Block ciphers are widely used in a large variety of modern applications. Traditionally, a block cipher is an algorithm that accepts a plaintext block and a key as inputs, and outputs a ciphertext block. To encrypt long messages, *modes of operation* are used to iterate the block cipher operation. Several modes of operation were designed to supply encryption with various desirable properties, such as being stateful and disabling an adversary who commits a chosen-plaintext attack from controlling the plaintext input of the block cipher. However, all these modes of operation, even when combined with an ideal block cipher, have their pitfalls. In particular, the ciphetexts are usually distinguishable from random sequences. Moreover, an adversary that obtains many ciphertext blocks may still gain some information on the corresponding plaintext, that she should not be able to gain in an ideal situation.

In this paper we suggest a new framework for block-ciphers aimed to prevent such pitfalls when used properly. Our framework introduces two additional input parameters to the underlying block cipher – *a salt* and *a counter*. By avoiding repeated key-salt-counter combinations, we guarantee that each permutation is used (at most) once, and thus many known attacks, such as dictionary attack, differential attack, and linear attack become inapplicable.

Our new frameworks also eliminates known weaknesses of ECB and CBC modes. For example, in ECB, two equal ciphertext blocks indicate two equal plaintext blocks. In CBC, one can learn the XOR result of two plaintext blocks if their corresponding ciphertext blocks are equal. These weaknesses disappear when replacing the underlying block cipher in the mode of operation with an ABC and when using the ABC properly (i.e., never repeating the same key-salt-counter combination). Later in this paper we discuss modes of operations for ABCs.

This work can be seen as a sibling of the HAIFA framework for hash functions [3], which fixes many of the pitfalls of traditional hash functions. In our work we borrow these new parameters introduced in HAIFA (i.e., a salt and a counter) to the world of block ciphers in order to protect against many security pitfalls. The ability to borrow new ideas from the world of hash functions and to implement them in the world of block ciphers naturally leads to a discussion about the correlation between the two worlds. We discuss the possibility of building a compression function based on an ABC, taking advantage of their similar APIs.

1.1 Related Work

Seeds for some of our new ideas can be found in [14], where the authors suggest a block cipher that has an additional input called a *tweak*. This tweak is used to provide some variability to the encryption, without necessarily ensuring that each block uses a different permutation. In [14], the authors suggest several modes of operation for their new framework, of which one is the Tweakable Authenticated Encryption mode (TAE) where the concatenation of a nonce and the block index is used as a tweak. In our work the nonce (salt) and the index (counter) parameters are separated. This separation is natural due to the difference in the functionality of these two parameters. The salt should be a fixed value in all the ABC calls that are required for an encryption of a single message while the counter should be unique in every such call. The salt value can either be defined by the protocol or chosen by the encrypting party, while the counter value in every ABC call is defined by the protocol and the encrypting party cannot control it. Our framework also has better security than the security of tweakable block ciphers in some of the cases.

A major part of our work is investigating the security of our framework. For this analysis we adapt some security notions that were previously defined in [2].

1.2 Outline of This Paper

In Section 2 we describe our new ABC framework. In Section 3 we introduce the notations and definitions that we use throughout the paper. In Section 4 we define some security notions and provide security proofs for various properties. In Section 5 we discuss the similarity between stream ciphers and encryption schemes that are based on our new framework, and in Section 6 we discuss the relation of our new framework to hash functions. Finally, in Section 7 we suggest several instances for ABC block ciphers that conform to the new framework and examine these implementations.

2 The Framework

2.1 The ABC Block Ciphers

Definition 1 *A* salt S is a non-secret parameter that can be considered as defining a family of block ciphers.

The salt value should be chosen by the encrypting party or defined by the protocol (e.g., increased by one for every message).

Definition 2 *A* counter t is a counter or a dithering, intended to introduce diversity between blocks.

The idea is that, when using an ABC as a building block for an encryption scheme (such as a mode of operation), the counter values for the instances of the ABC will be chosen by the encryption scheme and not by the encrypting party. This way, the counter is a parameter which is not subjected to manipulation by any of the parties.

Our new ABC framework introduces the use of a block counter and a salt as inputs to the underlying block cipher and defined as follows:

Definition 3 *An* advanced block cipher *(or shortly an* ABC*) is a function of the form:*

$$E : \underbrace{\{0,1\}^n}_{plaintext} \times \underbrace{\{0,1\}^k}_{key} \times \underbrace{\{0,1\}^s}_{salt} \times \underbrace{\{0,1\}^c}_{counter} \rightarrow \underbrace{\{0,1\}^n}_{ciphertext},$$

such that the ciphertext block is computed by $C = E_{K,S,t}(M)$*, where* E *is an* ABC*,* K *is the secret key,* S *is a salt, and* t *is a counter.*

The ABC framework aims at avoiding attacks that take advantage of the iteration of the encryption permutation by making sure that the same combination of key, salt and counter will never repeat. To achieve this, we demand that the same key-salt combination will never repeat for different messages, and that the same counter value will not repeat in two different ABC calls that are made while encrypting a single message. This demand limits the maximal length of an encrypted message to 2^c blocks, but for sufficiently large values of c this is not expected to be a problem.

As said, the salt should be defined by the protocol or chosen by the encrypting party in a way that ensures that the same key-salt combination never repeats. This requirement allows salt values to be reused as long as the keys are different. An even stronger requirement would never allow a salt value repeat for two different messages (even if the keys that are used to encrypt these messages are different). Such a requirement provides an even better security.

Note that like block ciphers, or any other kind of cryptographic primitives, ABCs are not necessarily secure, e.g., the identity ABC defined by $E_{K,S,t}(M) = M$ for every K, S and t, is clearly insecure. In this paper we assume that the ABCs and the (traditional) block ciphers are secure, and discuss the security features of such primitives when combined in modes of operation.

When discussing the security of modes of operations for ABCs, we assume that the underlying ABCs are ideal, and when we develop ABCs we try to make them as close as possible to ideal ABCs.

Definition 4 *An* ideal block cipher *is a family of random permutations* $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *such that for every key* $K \in \{0,1\}^k$ *the permutation* $E_K(\cdot)$ *is chosen uniformly at random from the set of all possible permutations over* $\{0,1\}^n$.

Definition 5 *An* ideal ABC *is a family of random permutations such that for every* (K, S, t) *combination the permutation* $E_{K,S,t}(\cdot)$ *is chosen uniformly at random from the set of all possible permutations over* $\{0,1\}^n$.

2.2 Using the New Cipher in Modes of Operation

Modes of operation are used for the encryption of long messages. A mode of operation makes use of several calls to an underlying block cipher. ABCs can also be used in modes of operation, and in this section we discuss a family of modes of operation for ABCs.

Definition 6 *An* advanced mode of operation *(or shortly* AMode*) is a mode of operation in which the underlying cipher of the mode is an ABC, and when encrypting a single multi-block message under the mode of operation, all the calls to the underlying ABC get the same salt input, and each call gets a different counter.*

Notation 1 *Let* X *be a conventional mode of operation. We denote by* $\mathrm{AX_{K,S}}$ *the AMode that has the same structure as* X *but in which the underlying cipher is an ABC that gets* S *as the salt of all calls to the ABC, and the block index as the counter of each ABC.*

Example 1 $\mathrm{AECB_{K,S}}$ *is the AMode in which the* i*'th block satisfies:* $C_i = E_{K,S,i}(M_i)$.

Example 2 $\mathrm{ACBC_{K,S}}$ *is the AMode in which* $C_0 = IV$ *and* $C_i = E_{K,S,i}(M_i) \bigoplus C_{i-1}$ *for every* $i = 1, ..., m$, *where* m *is the number of blocks.*

As we show in Section 4, the AECB mode is indistinguishable from a random permutation when the underlying ABC it uses is ideal. Therefore, it looks like for "good" ciphers, there is no security motivation to use AModes with feedback. In particular, in the new framework, we get rid of the main pitfall of the ECB mode – encryption of the same plaintext block under the same key always results with the same ciphertext block. One might still want to use AModes with feedbacks for different purposes. For example, AOFB might be desirable for its fast online computations, and ACBC might be desirable for those that wish to mix feedbacks into encryption, and for transforming the cipher into a MAC. We show that, in our framework, these modes of operation are also secure.

2.3 Reasonable Modes of operation and Reasonable AModes

We are specifically interested in a family of modes of operation and a family of AModes that we call *reasonable modes* and *reasonable AModes*.

Definition 7 *Let X be a mode of operation. Let E be any block cipher. We say that X is a* reasonable mode *if it fulfills all of the following requirements:*

1. *X is length preserving (the ciphertext is of the same length as the plaintext), and both plaintext and ciphertext have the same number of blocks, where the length of each block is n bits (recall that n is the block size of E).*
2. *The number of E encryptions performed calculated while encrypting a message M under X_K^E is equal to the length of M in blocks. In particular, there exist some functions $f_1(\cdot, ..., \cdot)$ and $f_2(\cdot, ..., \cdot)$ such that $C_i = f_2(E_K(f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})), IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})$, where:*
   - *(a) $f_2(\cdot; IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})$ is a permutation over $\{0,1\}^n$ for every possible combination of $IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}$.*
   - *(b) C_i is a permutation of M_i, i.e., the encryption under X_K^E is invertible and therefore the ciphertext can be decrypted. Formally, the function $h_{E,K,IV,i,M_1,...,M_{i-1},C_1,...,C_{i-1}}(M_i)$ $= f_2(E_K(f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})), IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})$ is a permutation over $\{0,1\}^n$ for every possible combination of $E, K, IV, i, M_1, ..., M_{i-1}, C_1, ..., C_{i-1}$.*

We associate every instance of E with a block number i, and we denote the plaintext input and the ciphertext output of the i'th instance of E by (x_i, y_i), respectively.

The above definition deals with modes of operation in which the block size is the same for all blocks. This definition can be trivially extended to deal with modes of operation in which the block size varies, as long as $|M_i| = |C_i|$ for any i. This extended definition covers all the standard cases where the last block can be shorter than n bits.

The conditions in Definition 7 might seem complicated, but most of the widely used modes of operation, such as ECB, CBC, OFB, and CTR, fulfill these conditions and therefore considered to be reasonable modes.

Example 3 *ECB is a reasonable mode where $f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) = M_i$ and $f_2(y_i, IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) = y_i$.*

Example 4 *CBC is a reasonable mode where $f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) = C_{i-1} \bigoplus M_i$ (for $i > 1$) and $f_2(y_i, IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) = y_i$.*

Example 5 *OFB is a reasonable mode where $f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) = C_{i-1} \bigoplus M_{i-1}$ and $f_2(y_i, IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) = y_i \bigoplus M_i$.*

Definition 8 *An AMode X is a* reasonable AMode *if it satisfies all of the conditions of Definition 7, where the block cipher E_K is substituted by the* ABC *$E_{K,S,t}$, for a key K, a salt S, and a counter t.*

Lemma 1 *Let X be a reasonable mode. Then* AX *is a reasonable* AMode.

The proof of Lemma 1 is given in Appendix A.

3 Notations and Definitions

Table 1 lists the notations that we use in this paper.

We define three kinds of **symmetric encryption schemes**.

K	Key
k	Key size in bits
S	Salt
s	Salt size in bits
t	Counter
c	Counter size in bits
M	Plaintext message
m	number of blocks of a message M. ($M = M_1 \| \| ... \| \| M_m$)
C	Ciphertext
M_i, C_i	The i'th block of the plaintext M, or the ciphertext C
$\|Y\|$	Length in bits of the string Y
$X_K^E(M)$	The encryption of the message M under the block cipher E using the key K and the mode of operation X
$X_K^{E^{-1}}(C)$	The decryption of the ciphertext C under the block cipher E using the key K and the mode of operation X
$X_{K,S}^E(M)$	The encryption of the message M under the ABC E using the key K, the salt S and the AMode X
$X_{K,S}^{E^{-1}}(C)$	The decryption of the ciphertext C under the ABC E using the key K, the salt S and the AMode X
$a \xleftarrow{R} A$	The operation of choosing an item a uniformly at random out of a set A.

Table 1. Notations

Definition 9 *A* traditional symmetric encryption scheme *is a pair of deterministic algorithms* $(\mathcal{E}, \mathcal{D})$, *that accept a key and plaintext/ciphertext as parameters. Formally* $\mathcal{E}$ *and* $\mathcal{D}$ *are defined as:*

$$\mathcal{E} : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C} \quad ; \quad \mathcal{D} : \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M},$$

where $\mathcal{K}$ *is the key space,* $\mathcal{M}$ *is the message space and* $\mathcal{C}$ *is the ciphertext space. We demand that for any* $K \in \mathcal{K}$ *and* $M \in \mathcal{M}$ *it holds that* $\mathcal{D}_K(\mathcal{E}_K(M)) = M$.

The traditional symmetric encryption schemes discussed in this paper are traditional block-ciphers and traditional modes of operations.

Definition 10 *A* salted-countered symmetric encryption scheme *is a pair of deterministic algorithms* $(\mathcal{E}, \mathcal{D})$, *that accept as parameters (on top of the key and the plaintext/ciphertext) also a counter and a salt. Formally* $\mathcal{E}$ *and* $\mathcal{D}$ *are defined as:*

$$\mathcal{E} : \mathcal{M} \times \mathcal{K} \times \mathcal{S} \times \mathcal{T} \rightarrow \mathcal{C} \quad ; \quad \mathcal{D} : \mathcal{C} \times \mathcal{K} \times \mathcal{S} \times \mathcal{T} \rightarrow \mathcal{M},$$

where $\mathcal{S}$ *is the salts space and* $\mathcal{T}$ *is the counter space. We demand that for any* $K \in \mathcal{K}$,$S \in \mathcal{S}$, $t \in \mathcal{T}$ *and* $M \in \mathcal{M}$ *it holds that* $\mathcal{D}_{K,S,t}(\mathcal{E}_{K,S,t}(M)) = M$.

The salted-countered symmetric encryption schemes discussed in this paper are ABCs.

Definition 11 *A* salted symmetric encryption scheme *is a pair of deterministic algorithms* $(\mathcal{E}, \mathcal{D})$, *that accept as parameters (on top of the key and the plaintext/ciphertext) also a salt but no counter. Formally* $\mathcal{E}$ *and* $\mathcal{D}$ *are defined as:*

$$\mathcal{E} : \mathcal{M} \times \mathcal{K} \times \mathcal{S} \rightarrow \mathcal{C} \quad ; \quad \mathcal{D} : \mathcal{C} \times \mathcal{K} \times \mathcal{S} \rightarrow \mathcal{M},$$

We demand that for any $K \in \mathcal{K}$,$S \in \mathcal{S}$ *and* $M \in \mathcal{M}$ *it holds that* $\mathcal{D}_{K,S}(\mathcal{E}_{K,S}(M)) = M$.

The salted-countered symmetric encryption schemes discussed in this paper are AModes.

For each of the definitions above, we note that $\mathcal{E}$ (and similarly $\mathcal{D}$) uniquely define the symmetric encryption scheme. We will therefore use $\mathcal{E}$ to describe the encryption scheme – it will be apparent from the context whether $\mathcal{E}$ represents the encryption scheme or the encryption function itself.

4 Security

In this section we discuss the security of ABCs and AModes. We analyze the security of our framework against known generic attacks, and we measure the security of AModes according to some widely-used security notions. In order to examine the security features of the AModes, we assume that the underlying ABC is ideal and that any security pitfall results from the mode itself. Nevertheless, we also consider in some cases a realistic ABC and prove that if the ABC is "secure enough" then the AMode using it is also "secure enough".

The proofs for all the theorems that appear in this section are given in Appendix A.

4.1 Security Against Generic Attacks

As discussed in the introduction, the ABC framework is secure against generic attacks that require many plaintexts encrypted under the same secret permutation. Examples for such attacks are the dictionary attack, differential attack, linear attack and time-memory tradeoff attacks.

In the dictionary attack the adversary collects pairs of plaintext blocks and their respective ciphertexts. She gains information about the encryption permutation without necessarily learning anything about the secret key. Later on she can use the knowledge that she had gained to decrypt ciphertexts or to encrypt plaintexts. This kind of attack is useless against protocols that use ABCs properly, since in such protocols the proper use of the salt and counter inputs ensures that no encryption permutation is used more than once. Thus, any information that the adversary has gained is useless for future encryption or decryption of plaintexts/ciphertexts.

The differential and linear attacks require a large amount of data encrypted under the secret encryption permutation $E_K(\cdot)$. These attacks are inapplicable for the ABC frmaework since a proper use ensures that an encryption permutation is never used more than once, and since in the ideal case, the different permutations used by the ABC for different values of salt-counter combinations are independent. Note that for realistic underlying ABCs, as for block ciphers, the security analysis should consider differential and linear attacks on the particular design. Moreover, it should consider the possibility of extending these attacks using the new parameters, the salt and the counter. In Section 7, where we suggest some instances of ABCs, we make this kind of analysis.

In time-memory tradeoff attacks [12], a large amount of pre-computation, equivalent to exhaustive search, can be used for breaking the encryption many times in the future. As discussed in Section 2, the salt might be reused with different keys. If this is the case (e.g., every time the key is changed the salt is reset to zero) then the pre-computation will be amortized among many instances of the attack, and thus the time-memory tradeoff attack will work against the framework just as they do for traditional block ciphers. On the other hand, if the salt is never reused (not even after changing the key) then the pre-computation cannot be amortized, and therefore time-memory tradeoffs become as inefficient as an exhaustive search.

4.2 Security Notions

Goldwasser and Micali [11] were the first to formally define security notions for encryption schemes. In [2] security notions for symmetric encryption were defined and examined. We adapt some of these notions to ABCs and their extra arguments while we use some of the ideas introduced in [16].

The security notions we consider are indistinguishability from random bits (defined in Section 4.2.2) and semantic security (defined in Section 4.2.3). We consider the security of AModes in the terms of the security notions mentioned above, both in the chosen-plaintext model (CPA), where the adversary is allowed to ask for encryption of messages and in the chosen-ciphertext model (CCA) where the adversary is allowed to ask for decryptions of ciphertexts.

We examine the security of our model in the information-theory sense (rather than in the computational sense). I.e., the adversaries that we consider are limited by the amount of information that they are allowed to have rather than in the time complexity of their computations.

Definition 12 *Let $\mathcal{E}$ be a symmetric encryption scheme. An $\mathcal{E}$-*CPA **adversary** *is an adversary that has access to an oracle that answers queries of one of three forms: (M) – if $\mathcal{E}$ is a traditional symmetric encryption scheme,*

(M, S) – *if* $\mathcal{E}$ *is a salted symmetric encryption scheme, or* (M, S, t) – *if* $\mathcal{E}$ *is a salted-countered symmetric encryption scheme (where* $M \in \mathcal{M}$*,* $S \in \mathcal{S}$ *and* $t \in \mathcal{T}$*).*

Definition 13 *An* $\mathcal{E}$-CCA **adversary** *is an adversary that has access to an oracle that answers queries of one of three forms:* (C) – *if* $\mathcal{E}$ *is a traditional symmetric encryption scheme,* (C, S) – *if* $\mathcal{E}$ *is a salted symmetric encryption scheme, or* (C, S, t) – *if* $\mathcal{E}$ *is a salted-countered symmetric encryption scheme (where* $C \in \mathcal{C}$*,* $S \in \mathcal{S}$ *and* $t \in \mathcal{T}$*).*

As stated in Section 2, the salt is chosen either by the encrypting party or by the protocol, and the counter is selected by the AMode. For our analysis, we let the adversary to have more power by letting her choose the salt as long as the same salt never repeats. Similarly, if she attacks an ABC rather than an AMode, we let her chose the salt and the counter as long as the same combination of salt-counter never repeats.

Definition 14 *Let* E *be an* ABC*. An* E-CPA *adversary is said to be* salt-counter-respecting *when no two queries it calls have the same salt-counter combination.*

Definition 15 *Let* E *be an* ABC*. Let* X *be an* AMode*. An* X^E-CPA *adversary is said to be* salt-respecting *when no two queries it calls have the same salt.*

We note that the above definitions hold only for *encryption* queries and not for decryption queries. A good practice is to make sure that every message is encrypted using a unique key-salt combination, but if for some reason different messages were encrypted using the same key-salt combination then it should be possible to decrypt the resulting ciphertexts. Therefore, we allow a CCA adversary to use the same salt for different queries.

4.2.1 Distinguishers for ABCs An advanced block cipher is a family of permutations $E : \{0,1\}^n \times \{0,1\}^k \times \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n$. By fixing a key $K \in \{0,1\}^k$, an ABC defines a *salted-countered*-family of permutations of the form: $E_K : \{0,1\}^n \times \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n$. By fixing a key K, a salt S and a counter t we define a single permutation $E_{K,S,t} : \{0,1\}^n \rightarrow \{0,1\}^n$.

Let $SCPerm(s, c, n)$ be the set of all possible salted-countered-families of permutations of the form: $\Pi : \{0,1\}^n \times \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n$.

We adapt the definitions of the advantage of a block-cipher distinguisher defined in [2] to our needs. Let $E : \{0,1\}^n \times \{0,1\}^k \times \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n$ be an ABC. Let A be an adversary that has access to a salted-countered-family of permutations $\pi : \{0,1\}^n \times \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n$. The advantage of an adversary A in distinguishing E from a random family of permutations is defined as:

$$\mathrm{Adv}_E^{\mathrm{prp}}(A) \triangleq \Pr\left[A^{E_K(\cdot,\cdot,\cdot)} \middle| K \xleftarrow{R} \{0,1\}^k\right] - \Pr\left[A^{\pi(\cdot,\cdot,\cdot)} \middle| \pi \xleftarrow{R} SCPerm(s, c, n)\right].$$

We denote by $\mathrm{Sec}_E^{\mathrm{prp}}(q)$ the maximum advantage of any salt-counter-respecting adversary A that is allowed to make queries to a salted-countered-family of permutations, and makes no more than q queries. Denote by $\mathcal{A}_q^{scr}$ the set of all salt-counter-respecting adversaries that make no more than q queries. The maximum is taken over all possible such adversaries, i.e., $\mathrm{Sec}_E^{\mathrm{prp}}(q) \triangleq \max_{A \in \mathcal{A}_q^{scr}} \mathrm{Adv}_E^{\mathrm{prp}}(A)$.

Lemma 2 (given without a proof) *Let* E *be an ideal* ABC*. Then every possible total length of queries* σ *that allows the adversary to be salt-counter-respecting satisfies* $\mathrm{Sec}_E^{\mathrm{prp-CPA}}(\sigma) = 0$.

We later show that if E is an ABC for which $\mathrm{Sec}_E^{\mathrm{prp}}(q)$ is small then using a reasonable AMode in which the underlying ABC is E results with a secure encryption scheme.

4.2.2 Indistinguishability from Random Bits This security notion evaluates the ability of an adversary to distinguish the encryption (decryption) of a message from the encryption (decryption) of an equal length random string of bits.

Let $\mathcal{E}$ be a traditional symmetric encryption scheme. Consider the following two oracles, both answer queries of the form $Q = (M)$ with a string $C \in \{0,1\}^{|\mathcal{E}_K(M)|}$. The first oracle is the real encryption oracle $O_{\mathcal{E}_\mathcal{K}}$ that answers the query $Q = (M)$ with $C = \mathcal{E}_K(M)$. The second is a fake encryption oracle $O_R^\mathcal{E}$ that answers the same query Q with a random string of $|\mathcal{E}_K(M)|$ bits.

The $\mathrm{ind}-\mathrm{CPA}$ advantage of an $\mathcal{E}-\mathrm{CPA}$ adversary A is defined as:

$$\mathrm{Adv}_\mathcal{E}^{\mathrm{ind-CPA}}(A) \triangleq \Pr\left[A^{O_{\mathcal{E}_\mathcal{K}}(\cdot)} = 1 \middle| K \xleftarrow{R} \mathcal{K}\right] - \Pr\left[A^{O_R^\mathcal{E}(\cdot)} = 1\right].$$

For the CCA variant of this notion, consider two other oracles, both answer queries of the form $Q = (C)$ with a string $M \in \{0,1\}^{|\mathcal{D}_K(C)|}$. The first is the real decryption oracle, $O_{\mathcal{D}_\mathcal{K}}$ that answers the query $Q = (C)$ with $M = \mathcal{D}_K(C)$. The second is a fake decryption oracle, $O_R^\mathcal{D}$ that answers the same query Q with a random string of $|\mathcal{D}_K(C)|$ bits.

The $\mathrm{ind}-\mathrm{CCA}$ advantage of an adversary A against an encryption scheme $\mathcal{E}$ is defined as:

$$\mathrm{Adv}_\mathcal{E}^{\mathrm{ind-CCA}}(A) \triangleq \Pr\left[A^{O_{\mathcal{D}_\mathcal{K}}(\cdot)} = 1 \middle| K \xleftarrow{R} \mathcal{K}\right] - \Pr\left[A^{O_R^\mathcal{D}(\cdot)} = 1\right].$$

We adapt these security notions also to the salted-symmetric encryption scheme, in which the queries are of the form $Q = (M, S)$. Thus, if $\mathcal{E}$ is a salted-symmetric encryption scheme, then

$$\mathrm{Adv}_\mathcal{E}^{\mathrm{ind-CPA}}(A) \triangleq \Pr\left[A^{O_{\mathcal{E}_\mathcal{K}}(\cdot,\cdot)} = 1 \middle| K \xleftarrow{R} \mathcal{K}\right] - \Pr\left[A^{O_R^\mathcal{E}(\cdot,\cdot)} = 1\right],$$
$$\mathrm{Adv}_\mathcal{E}^{\mathrm{ind-CCA}}(A) \triangleq \Pr\left[A^{O_{\mathcal{D}_\mathcal{K}}(\cdot,\cdot)} = 1 \middle| K \xleftarrow{R} \mathcal{K}\right] - \Pr\left[A^{O_R^\mathcal{D}(\cdot,\cdot)} = 1\right].$$

It is important to note that the adversary A must not repeat the same query twice (or otherwise it will be trivial to distinguish between the real and the random oracles). When discussing AModes, this demand is being respected automatically in the CPA variant of the notion since we consider only salt-respecting adversaries. Since CCA adversaries are allowed to repeat the same salt, we have to explicitly prohibit them from repeating the same query more than once.

We use the abbreviated notations of O_E for the oracle $O_{\mathcal{E}_\mathcal{K}}$ or $O_{\mathcal{D}_\mathcal{K}}$ with a random key K, and O_R for $O_R^\mathcal{E}$ or $O_R^\mathcal{D}$, where it is clear from the context whether the attack is a chosen-plaintext attack or a chosen-ciphertext attack.

We denote by $\mathrm{Sec}_\mathcal{E}^{\mathrm{ind-CPA}}(\sigma)$ the maximum advantage taken over all $\mathcal{E}$-CPA salt-respecting adversaries that use no more than a total of σ blocks in their queries. Similarly, we denote by $\mathrm{Sec}_\mathcal{E}^{\mathrm{ind-CCA}}(\sigma)$ the maximum advantage taken over all $\mathcal{E}$-CCA adversaries that use no more than a total of σ blocks in their queries. Denote by $\mathcal{A}_\sigma$ the set of all adversaries that use no more than a total of σ blocks in their queries, and denote by $\mathcal{A}_\sigma^{sr}$ the set of all salt-respecting adversaries that use no more than a total of σ blocks in their queries. Formally,

$$\mathrm{Sec}_\mathcal{E}^{\mathrm{ind-CPA}}(\sigma) \triangleq \max_{A \in \mathcal{A}_\sigma^{sr}} \mathrm{Adv}_\mathcal{E}^{\mathrm{ind-CPA}}(A) \quad ; \quad \mathrm{Sec}_\mathcal{E}^{\mathrm{ind-CCA}}(\sigma) \triangleq \max_{A \in \mathcal{A}_\sigma} \mathrm{Adv}_\mathcal{E}^{\mathrm{ind-CCA}}(A).$$

The following lemmas examine the ind-CPA security AECB and ACBC AModes.

Lemma 3 *Let E be an ideal* ABC. *Every possible total length of queries, σ that allows the adversary to be salt-respecting (i.e., $\sigma \leq 2^{c+s}$) satisfies* $\mathrm{Sec}_{\mathrm{AECB}^{\mathrm{E}}}^{\mathrm{ind-CPA}}(\sigma) = 0$.

Lemma 4 *Let E be an ideal* ABC. *Every possible total length of queries σ that allows the adversary to be salt-respecting satisfies* $\mathrm{Sec}_{\mathrm{ACBC}^{\mathrm{E}}}^{\mathrm{ind-CPA}}(\sigma) = 0$.

The same result can be achieved for any reasonable AMode, as discussed in the following theorem and conclusion.

Theorem 1 *Let X be a reasonable* AMode*. Let E be an* ABC*. Every possible total length of queries σ that allows the adversary to be salt-respecting satisfies* $\mathrm{Sec}_{X^E}^{\mathrm{ind-CPA}}(\sigma) \leq \mathrm{Sec}_E^{\mathrm{prp}}(\sigma)$.

Following Theorem 1 and Lemma 2, we conclude the following conclusion:

Conclusion 1 *If E is an ideal ABC, and X is a reasonable* AMode*, then* $\mathrm{Sec}_{X^E}^{\mathrm{ind-CPA}}(\sigma) = 0$ *for every σ that allows the adversary to be salt-respecting.*

As seen in Conclusion 1, our framework achieves perfect indistinguishability security against chosen-plaintext attacks when the underlying ABC is ideal. This is not the case when considering chosen-ciphertext attacks because of the ability of the adversary who employs the attack to repeat the same salt-counter combination more than once in its queries to the decryption oracle. A CCA adversary can simply ask for a decryption of $C = C_1||C_2$ and $C' = C_1$, both with the same salt value S, and return '1' if and only if the corresponding messages, M, M' begin with the same block. This attack works for every reasonable AMode.

However, we claim that the indistinguishability security against chosen-ciphertext attacks of our framework is not worse than the security against chosen-ciphertext attacks of the conventional framework. In particular, the following theorem proves that the ind-CCA security of AECB is not worse than the ind-CCA security of ECB.

Theorem 2 *Let $\tilde{E}$ be an ideal* ABC *and let E be an ideal block cipher. For every σ it holds that* $\mathrm{Sec}_{\mathrm{ECB}^E}^{\mathrm{ind-CCA}}(\sigma) \geq \mathrm{Sec}_{\mathrm{AECB}^{\tilde{\mathrm{E}}}}^{\mathrm{ind-CCA}}(\sigma)$.

4.2.3 Semantic Security The semantic security notion was defined first in [11] and was adapted to the symmetric scheme in [2]. This security notion evaluates the ability of an adversary to learn something on a plaintext from its corresponding ciphertext. We adapt this security notion to the parameterized encryption scheme. Let $(\mathcal{E}, \mathcal{D})$ be a salted-symmetric encryption scheme. Let A be an adversary. For the CPA variant of the notion, we consider an $\mathcal{E}$-CPA adversary, and for the CCA variant we consider an $\mathcal{E}$-CCA adversary.

The adversary plays a game of two stages: In the first stage, after A calls its oracle(s) with its queries, it defines some valid distribution function over the message space such that all the messages with non-zero probability are of the same length. In the second stage, A is provided with a salt S^* chosen uniformly at random, from all the salts that A has not used in its queries, and a ciphertext $C^* = \mathcal{E}_{K,S^*}(M^*)$ for some plaintext M^* chosen at random from the message space according to the distribution function defined by A. The adversary then outputs a pair (α, f), where f is a function, defined for all non-zero-probability messages, that can be computed by the adversary. A wins if $\alpha = f(M^*)$. Let M' be a message chosen at random from the message space, according to the distribution function defined by A and independently from M^*. The advantage of an adversary A is defined in this case as:

$$\mathrm{Adv}_{\mathcal{E}}^{\mathrm{sem-ATK}}(A) \triangleq \Pr_{M^*,K,A}[\alpha = f(M^*)\big|C^*] - \Pr_{M',K,A}[\alpha = f(M')],$$

where $\mathrm{ATK} = \mathrm{CPA}$ if A employs a chosen-plaintext attack and $\mathrm{ATK} = \mathrm{CCA}$ if A employs a chosen-ciphertext attack.

We denote by $\mathrm{Sec}_{\mathcal{E}}^{\mathrm{sem-ATK}}(\sigma)$ the maximum advantage taken over all $\mathcal{E} - \mathrm{ATK}$ adversaries that use no more than a total of σ blocks in their queries. Formally,

$$\mathrm{Sec}_{\mathcal{E}}^{\mathrm{sem-ATK}}(\sigma) \triangleq \max_{A \in \mathcal{A}_\sigma} \mathrm{Adv}_{\mathcal{E}}^{\mathrm{sem-ATK}}(A).$$

Theorem 3 *Let X be a reasonable* AMode*, and let E be an ideal* ABC*. Then* $\mathrm{Sec}_{X^E}^{\mathrm{sem-ATK}}(\sigma) = 0$*, where* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA}\}$.

4.3 Security Pitfalls of Conventional Modes of Operation

Many of the conventional modes of operation are distinguishable from random permutations even when the underlying block cipher is indistinguishable from a random permutation. Moreover, the following distinguishers allow an adversary who obtains a ciphertext, to gain some information on the corresponding plaintext. Ideally, the adversary should not gain such information.

4.3.1 ECB Obviously, The conventional ECB mode does not provide the indistinguishability security notion defined in Section 4.2.2. In the simplest attack the adversary A asks for the encryption of $M = M_1||M_2$ where $M_1 = M_2$ and gets $C = C_1||C_2$. It returns '1' if $C_1 = C_2$ and '0' otherwise. Therefore every underlying block cipher E satisfies $\mathrm{Adv}^{\mathrm{ind-CCA}}_{\mathrm{ECB}^E}(A) = 1 - 2^{-n}$, and the adversary can achieve even a higher advantage by using longer messages. In practice, this means that an adversary that obtains a ciphertext in which two blocks are equal learns that the two corresponding blocks of the plaintext are equal as well.

4.3.2 CBC The conventional CBC mode provides no security against chosen-ciphertext attacks in the indistinguishability sense (ind-CCA). Consider the following attack: The adversary A asks for the decryption of $C = C_1||C_1$ and gets $M = M_1||M_2$ as an answer. It returns '1' if $M_1 \bigoplus M_2 = C_1 \bigoplus IV$ and '0' otherwise. The advantage of A is $\mathrm{Adv}^{\mathrm{ind-CCA}}_{\mathrm{CBC}^E}(A) = 1 - 2^{-n}$, and the adversary can get even a higher advantage by using longer ciphertexts.

The CBC mode does not provide good ind-CPA security as well. When encrypting a message longer than $2^{n/2}$ blocks, it is expected (due to the birthday paradox) that two of the ciphertext blocks will collide (i.e., $C_i = C_j$ for some $i \neq j$), and thus an adversary that has access only to an encryption oracle, can check whether $M_i \bigoplus M_j = C_{i-1} \bigoplus C_{j-1}$ and distinguish between the encryption and random bits with high probability.

4.3.3 TBC In [14], the authors suggest several modes of operation for tweakable block ciphers. In one of them, called TAE, the tweak is used as a concatenation of a nonce (that has the same functionality as our salt) and a counter. This mode is equivalent to our AECB, and therefore provides the same security properties. But the tweak is not limited to this kind of usage, and when it is used differently the result can be an insecure mode. An example for this is the Tweakable Block Chaining (TBC) mode, suggested in [14]. The TBC mode is illustrated in Figure 1. The indistinguishability security of reasonable AModes is better than the indistinguishability security of TBC in some cases, and not worse in the others.

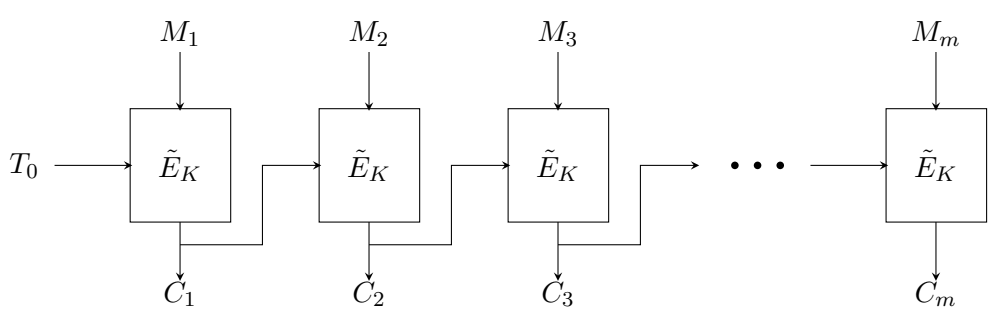

Fig. 1. TBC mode of operation

Let $\tilde{E}$ be an ideal tweakable block cipher, let E be an ideal ABC, and let X be a reasonable AMode. An ind-CPA adversary that attacks $TBC^{\tilde{E}}$ can ask for the encryption of a message $M = M_1||...||M_m$, where all the message blocks are equal. If $m \geq 2^{n/2}$, it is excepted that there exist some $1 \leq i < j \leq m$ such that $C_i = C_j$. If the output is a result of a real TBC encryption, rather than a random string, then for every $1 \leq l \leq m - j$

it holds that $C_{i+l} = C_{j+l}$. If this is the case then the adversary can guess that she is facing the real encryption oracle. Otherwise, she is definitely facing the random oracle. Therefore, following Conclusion 1, for every $\sigma \geq 2^{n/2}$ it holds that $\mathrm{Sec}^{\mathrm{ind-CPA}}_{TBC^{\tilde{E}}}(\sigma) > \mathrm{Sec}^{\mathrm{ind-CPA}}_{X^E}(\sigma)$.

An ind-CCA adversary $\tilde{A}$ that attacks $TBC^{\tilde{E}}$ and is limited to a total of two blocks in its queries can ask for the decryption of the message $T_0||T_0$, where T_0 is also the initial tweak, and obtain the oracle's answer – $M = M_1||M_2$. $\tilde{A}$ outputs '1' if $M_1 = M_2$, and '0' otherwise. The advantage of $\tilde{A}$ is given by $\mathrm{Adv}^{\mathrm{ind-CCA}}_{TBC^{\tilde{E}}}(\tilde{A}) = 1 - 2^{-n}$. When an ind-CCA adversary A that attacks X^E is limited to a total of two blocks in its queries, then the two blocks must be decrypted with a different salt-counter combination. Therefore, $\mathrm{Adv}^{\mathrm{ind-CCA}}_{X^E}(A) = 0$, and thus, $\mathrm{Sec}^{\mathrm{ind-CCA}}_{TBC^{\tilde{E}}}(2) > \mathrm{Sec}^{\mathrm{ind-CCA}}_{X^E}(2)$.

5 Comparison with Stream Ciphers and CTR Mode

A main requirement for our framework is that the same combination of key, salt, and counter should never be used twice. Apparently, if this requirement is fulfilled, then there is no reason that the function $E_{K,S,t}(\cdot)$ will be a pseudo-random permutation. It is enough to demand that for every combination of S, t and M, the function $f_{S,t,M} : \{0,1\}^k \rightarrow \{0,1\}^n$, defined by $f_{S,t,M}(K) = E_{K,S,t}(M)$, is a pseudo-random function. For example, assume that we have a pseudo-random function $g : \{0,1\}^k \times \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n$, and define $f_{S,t,M}(K) = g(K,S,t) \bigoplus M$. Thus, the AECB encryption using the underlying ABC defined by $E_{K,S,t}(M) = g(K,S,t) \bigoplus M$ can be viewed as a stream cipher in which the message M is XORed with the stream $g(K,S,t)$. Under the assumption that the same (K,S,t) combination is never used more than once, this encryption is secure, but if these assumptions do not hold for some reason then it is no longer secure since an adversary who holds a single (M,C) pair is able to calculate the appropriate $g(K,S,t)$ for every block of M and therefore can encrypt (decrypt) any message (ciphertext) with the same K and S values. Therefore, constructing an ABC which is a pseudo-random permutation for every (K,S,t) combination provides us with a system that is more tolerant for misuse and guarantees some security even when the same (K,S,t) is used more than once.

A good example is the CTR mode [7]. In the CTR mode, the message is XORed with a stream generated by encrypting a counter block. Formally, $C = C_1||...||C_m$ where $C_i = M_i \bigoplus E_K(ctr + i)$ for some number ctr. In [9] it is suggested that the ℓ least significant bits of the counter block are used as a counter while the other $n - \ell$ bits are used as a nonce. In this particular case, the CTR mode can be seen as a special case of AMode where its encryption function $E^{\mathrm{CTR}}_{K,S,t}(M) = M \bigoplus E_K(S + t)$, where S is a multiply of 2^ℓ, $0 \leq t < 2^\ell$, and the maximal length of a message is 2^l blocks.

We note, however, that our framework with the explicit separation of the salt and the counter has some advantages over the CTR mode and other stream ciphers:

1. Tolerance for misuse — In CTR mode, just as in any stream cipher, a reuse of the same stream (i.e., a reuse of the same nonce and counter) results with an immediate compromise of the security. When the same stream is used for two different messages M_1, M_2 then $M_1 \bigoplus M_2$ can be calculated from the corresponding ciphertexts C_1, C_2. Moreover, an adversary who has a pair (M,C) of plaintext-ciphertext blocks for a specific nonce S and a specific counter t can calculate the encryption of any other message with the same nonce and counter.
2. Time-memory-data tradeoffs — Time memory-data tradeoffs are widely discussed for stream ciphers. If the salt values are reused every time the key is changed, then the security of our framework against these attacks is the same as for stream ciphers. In the more strict version in which the salt is never repeated (not even after changing the key) then these attacks become inapplicable.
3. The security of our framework is a little bit better than the security of the CTR mode (particularly in the indistinguishability and semantic senses). This is because of the fact that in CTR mode, for every message $M = M_1||...||M_m$ and for every $1 \leq i < j \leq m$ it is certain that $C_i \bigoplus C_j \neq M_i \bigoplus M_j$. For example, an ind-CPA adversary can take advantage of this fact. The adversary asks for the encryption of

a message $M = M_1||...||M_{2^{n/2}}$ in which all of the $2^{n/2}$ blocks are identical (i.e., $M_i = M_j$ for every $1 \leq i < j \leq 2^{n/2}$). If the adversary receives an answer from the real encryption oracle it is guaranteed that $C_i \neq C_j$ for every $1 \leq i < j \leq 2^{n/2}$. If the adversary receives a random string as an answer then it is expected that there are some $1 \leq i < j \leq 2^{n/2}$ such that $C_i = C_j$.

6 The Relation to Hash Functions

In [3] the HAIFA framework for cryptographic hash function is presented. This framework introduces the idea of salt and bits-counter for compression functions. A natural question to ask is how can these new compression functions and block ciphers relate. In this section we discuss the possibility of using a single underlying primitive to build both ABCs and compression functions. We focus on the well known Davies-Meyer construction [5]. Other possible constructions were studied in [15] – some of them could be used for our purpose as well.

6.1 Using Davies-Meyer Construction

The Davies-Meyer construction is used for building a compression function, C_{DM} using a block cipher E. Given a block cipher $E : \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n$, the compression function $C_{DM} : \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n$ is defined as: $C_{DM}(h, M) = E_M(h) \bigoplus h$. Here, k is the length of the message block for the compression function and n is the length of the chaining value.

A major drawback of the Davies-Meyer construction is that it is easy to find fixpoints. One can find a fixpoint (h^*, M^*) simply by fixing a message block M^* and decrypting the zero constant by $h^* = E^{-1}_{M^*}(0)$. Such a fixpoint can be used for a second-preimage attack as shown in [6].

We note that with our new block cipher framework and the HAIFA framework, this problem is fixed. We can use a Davies-Meyer construction to build a HAIFA compression function $C^{HAIFA}_{DM} : \{0,1\}^n \times \{0,1\}^b \times \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n$ (where n is the length of the chaining value and $b = k$ is the block size) using an ABC, E. Such a compression function will be defined as $C^{HAIFA}_{DM}(h, M, S, \#bits) = E_{M,S,f(\#bits)}(h) \bigoplus h$, for some function f that outputs a unique counter value for every possible $\#bits$ value (typically, $f = \lceil \frac{\#bits}{n} \rceil$). The $\#bits$ parameter of the compression function (t parameter in E) prevents attacks that take advantage of the easy-to-find fixpoints. It is still possible to find a tuple $(h^*, M^*, S^*, \#bits^*)$ such that $C_{DM}(h^*, M^*, S^*, \#bits^*) = h^*$, but for any $\#bits' \neq \#bits^*$ (and specifically for the value of $\#bits'$ that matches the next block) it holds, with a very high probability, that $C_{DM}(h^*, M^*, S^*, \#bits') \neq h^*$. Moreover, it is difficult to find a message block M' such that $C_{DM}(h^*, M', S^*, \#bits') = h^*$. Therefore, a fixpoint cannot be used to expand the message and create a second preimage.

It might be better, for performance reasons, to use the message block of the compression function as the *salt* of the ABC and not as the key, so the key scheduling algorithm is not re-executed for every block. However, the Davies-Meyer construction might not be practical since it demands that the key size (or salt size) of the cipher equals the size of compression function's message block and that the size of the compression function's chaining value equals the size of the cipher's block. This is not the case with today's compression functions and symmetric block ciphers. The typical size for message blocks in hash functions today is much larger than the typical key size of block ciphers and the typical size of a chaining value is much larger than the typical size of ciphers' block. The size of the chaining value of a compression function should be large enough so finding collisions will be difficult, and the size of the compression function's message block should be large enough for performance reasons. Increasing the lengths of the block cipher's key and message block would increase the complexity of the block cipher's execution (in particular the key schedule), and will influence especially on encryption of short messages.

7 Simple ABCs Based on AES

In this section we describe three AES-based ABCs, all of which with 128-bit keys, 128-bit salts and 64-bit counters.

7.1 ABC1

Our first ABC uses AES [17] as a black-box and is implemented as follows:

$$ABC1_{K,S,t}(M) = \text{AES-128}_{K'}(\text{AES-128}_K(\text{AES-128}_{K'}(M) \oplus t') \oplus t'),$$

where $K' = \text{AES-128}_K(S)$ and $t' = t||t$.

7.2 ABC2

ABC2 modifies the key scheduling algorithm of AES-256 to allow a mixing of the salt and the counter. The key scheduling algorithm is modified as follows: given a 128-bit key K and a 128-bit salt S, an intermediate 256-bit key K' is computed by $K' = K||\text{AES-128}_K(S)$. The intermediate key K' is then expanded using the original AES-256 key scheduling algorithm, to the (15) round keys $RK_1[0], ..., RK_1[14]$. The counter is mixed into five of these round keys – $RK_1[2], RK_1[4], RK_1[7], RK_1[10]$, and $RK_1[12]$: each time the counter is mixed into a round key it is XORed into two consecutive (cyclicly) columns of the round key, when the counter is represented as big endian. The counter is mixed into the following round keys: columns 0,1 of $RK_1[2]$; 1,2 of $RK_1[4]$; 2,3 of $RK_1[7]$; 3,0 of $RK_1[10]$ and 0,1 of $RK_1[12]$. The result is another set of round keys RK_2. Encryption is performed using the AES-256 encryption algorithm with the resulting round keys $RK_2[0], ..., RK_2[14]$. Note that an efficient implementation of ABC2 does not have to compute the key scheduling for every counter. Instead, it computes RK_1 once and XORs the counter into the right locations during the encryption process.

In order to allow a full diffusion of a counter before the next counter mixing occurs, we selected to have at least two AES rounds between consecutive counter mixings. A full diffusion of a counter mixing ensures that all of the bytes of the state are influenced by the counter. The next counter mixing, changes only two of the four columns of the state, while keeping the other two unchanged.

7.3 ABC3

ABC3 modifies the key scheduling algorithm of AES-128 to allow a mixing of the salt and the counter. The key scheduling algorithm is modified as follows: given a 128-bit key K and a 128-bit salt S, three temporary keys K_1, K_2, and K_3 are calculated. Each of the temporary keys is expanded by the original key scheduling algorithm of AES-128 into the (11) round keys. Then, a fourth set of round keys is calculated as the XOR of the three sets of round keys. Formally,

$$\begin{aligned}
&K_1 = K \quad ; \quad RK_1 = KS(K_1) \\
&K_2 = \text{AES-128}_{K_1}(S) \quad ; \quad RK_2 = KS(K_2) \\
&K_3 = \text{AES-128}_{RK_1 \oplus RK_2}(K) \quad ; \quad RK_3 = KS(K_3) \\
&RK_4 = RK_1 \oplus RK_2 \oplus RK_3,
\end{aligned}$$

where KS is the key scheduling algorithm of AES-128. The notation AES-128_{RK}, where RK is a set of round keys rather than a key, refers to the AES algorithm that uses the round keys RK instead of deriving them from a 128-bit key.

The counter is then mixed into five round keys of RK_4 – columns 0,1 and columns 2,3 of $RK_4[1]$; 1,2 of $RK_4[3]$; 2,3 of $RK_4[5]$; 3,0 of $RK_4[7]$; and columns 0,1 and columns 2,3 of $RK[9]$. The result is another set of round keys RK_5. Encryption is performed using the AES-128 encryption algorithm with the round keys $RK_5[0], ..., RK_5[10]$. As in ABC2, an efficient implementation of ABC3 does not have to compute the key scheduling for every counter. Instead, it computes RK_4 once and XORs the counter into the right locations during the encryption process.

7.4 Security of ABC1, ABC2 and ABC3

The main motivation behind the ABC framework is to provide security against attacks that take advantage of a repeating permutation. We claim that, for any of the ABCs suggested above, equivalent key-salt-counter combinations do not exist and therefore these ABCs really provide this kind of security. We also analyze the security of the ABCs suggested above against a variety of known attacks.

7.4.1 Equivalent Keys In an ideal ABC it is very unlikely that two different key-salt-counter combinations $(K_1, S_1, t_1) \neq (K_2, S_2, t_2)$ specify the same permutation. Therefore, like in block ciphers, the design of an ABC should result with permutations that do not collide.

In ABC1 the AES algorithm is used three times. Due to the use of K in the middle encryption, every pair of (K, S) generates a unique set of round keys for the triple encryption. Therefore, we claim that there are no pairs $(K_1, S_1) \neq (K_2, S_2)$ and a counter t for which $\mathrm{ABC1}_{K_1,S_1,t}(M) = \mathrm{ABC1}_{K_2,S_2,t}(M)$ for every M. Moreover, we claim that there do not exist different key-salt-counter combinations $(K_1, S_1, t_1) \neq (K_2, S_2, t_2)$ for which $\mathrm{ABC1}_{K_1,S_1,t_1}(M) = \mathrm{ABC1}_{K_2,S_2,t_2}(M)$ for every M.

In ABC2, for two different key-salt pairs $(K_1, S_1) \neq (K_2, S_2)$ the derived keys K_1', K_2' are necessarily different. Therefore, there is a difference in at least one of the first two round keys. Such a difference diffuses to other round keys. In case that $(K_1, S_1) = (K_2, S_2)$ but $t_1 \neq t_2$ then there is necessarily a difference in round keys 2, 4, 7, 10, and 12, that are influenced by the counter. Thus, every two different key-salt-counter combinations $(K_1, S_1, t_1) \neq (K_2, S_2, t_2)$ necessarily produce a different set of round keys RK_2. We claim that there are no two such different key-salt-counter combinations that define the same permutation.

In ABC3 every key-salt pair generates a unique set of round keys (RK_1, RK_2). We claim that every key-salt pair generates a unique set of round keys (RK_1, RK_2, RK_3), and that $RK_4 = RK_1 \oplus RK_2 \oplus RK_3$ is also unique for each key-salt pair. Moreover, we claim that every key-salt-counter combination results with a unique set of round keys RK_5, which specifies a unique permutation.

7.4.2 Security of ABC1 Against Any Known or Chosen Plaintext Attack We show a reduction from any known/chosen plaintext attack on ABC1 to a known/chosen plaintext attack on AES. We conclude that the security of ABC1 against known/chosen plaintext attacks (e.g., differential attack, linear attack and time-memory tradeoff attacks) is similar to the security of AES against these attacks.

Consider a known/chosen plaintext attack A on ABC1. We build an attack B on AES-128 using the attack A. Whenever A asks for the encryption of a plaintext block M, with a salt S, and with a counter t, B asks for $K' = \text{AES-128}_K(S)$. Then B computes $V_1 = \text{AES-128}_{K'}(M) \oplus t'$, asks for $V_2 = \text{AES-128}_K(V_1)$, and computes $C = \text{AES-128}_{K'}(V_2 \oplus t')$, thus receiving $C = \mathrm{ABC2}_{K,S,t}(M)$. Once A announces the recovered key K (or any information on the key), B announces the same key K (or the same information that A announced). Therefore, B learns exactly the same information on the key as A does.

7.4.3 Security of ABC2 and ABC3 Against Linear Attacks ABC2 and ABC3 are based on the AES algorithm and use its S Box. The maximal bias of a linear approximation of this S Box, as shown in [4], is 2^{-3}. Using a computer program we found that the lower bound of active S Boxes in a linear characteristic is 80 for ABC2, and 55 for ABC3. Therefore, the bias of any linear approximation of the full ABC2 and ABC3 is bounded by 2^{-240} and 2^{-165}, respectively. We note that the introduction of the counter might allow the adversary to attack the cipher using a shorter characteristic of 12 rounds in ABC2, or 9 rounds in ABC3. The number of active S Boxes in 12 rounds of ABC2 is lower bounded by 75 and the number of active S Boxes in 9 rounds of ABC is lower bounded by 51. Therefore the bias of a linear approximation of 12 rounds of ABC2 or 9 rounds of ABC3 are bounded by 2^{-225} and 2^{-151}, respectively. These biases would require over 2^{300} known plaintexts for a linear attack while there are only 2^{128} plaintexts in the whole block space, which makes this attack impossible. We note that in our model an adversary can use more than 2^{128} plaintext-counter combinations for a linear attack, but the number of encryptions required for such an attack is larger than 2^{128} – the number of encryptions required for an exhaustive search, and therefore such an attack is also unapplicable.

	ABC1	ABC2	ABC3	AES
Key scheduling avg speed (in cpu cycles)	545.25	625.62	924	162.14
Key scheduling standard deviation	3.19	2.53	4.87	3.06
Key scheduling speed median (in cpu cycles)	541	627	925	162
Encryption avg speed (in cpu cycles)	825.82	385.47	326.11	240.32
Encryption speed standard deviation	35.05	1.09	9.41	6.64
Encryption speed median (in cpu cycles)	808	385	322	238

Table 2. Performance of the different implementations

7.4.4 Security of ABC2 and ABC3 Against Extended Differential Attacks Unlike traditional differential attacks, in differential attacks on ABCs the adversary is not limited to introduce the differences through the plaintext. She can also introduce the differences through the salt or through the counter.

Definition 16 *We use the term* extended differential characteristic *for a differential characteristic that may include a salt difference and/or a counter difference in addition to the plaintext difference. An attack that uses an extended differential characteristic is called* extended differential attack.

We note that using an extended differential characteristic that has a salt difference is useless when attacking ABC2 or ABC3, since in both ABCs the salt is encrypted by the secret key before it is used. Thus, an adversary who uses such a characteristic has no information about the differential that is actually in use. Therefore, we limit our analysis to the case where there is no salt difference. Obviously, a counter difference can partially "fix" a plaintext difference and slow down the diffusion of the difference. Therefore, we expect that the diffusion of differences will be slightly slower in ABC2 and ABC3 than in AES. Nevertheless, we claim that both ABCs are still secure against extended differential attacks, and indeed, a simulation that checks all possible differential trails shows that after six rounds of ABC2 and after seven rounds of ABC3 there are always at least 25 active S Boxes. In a full run of 14 rounds of ABC2 there are at least 53 active S Boxes, and in a full run of 10 rounds of ABC3 there are at least 30 active boxes. Thus, ABC2 does not have any extended differential characteristic (without a salt-difference) that has probability higher than $(2^{-6})^{53} = 2^{-318}$. Similarly, ABC3 does not have such an extended differential characteristic that has probability higher than $(2^{-6})^{30} = 2^{-180}$.

7.5 Performance of Our ABCs

In order to check the performance of our suggested ABCs, we used an Intel Xeon E5540 processor with a 2.53GHz CPU and cache size of 8192KB that runs a Red Hat Enterprise Linux Server release 5.5. We used parts of the AES code of Brian Gladman [10] for our ABCs implementations. The results are summarized in Table 2 along with the performance results of Gladman's AES-128 code for comparison.

Bibliography

[1] Mihir Bellare, John Black, Philip Rogaway, *OCB: A Block-Cipher Mode of Operation*, ACM Transactions on Information and System Security (TISSEC), vol. 6, no. 3, pp. 365-403, August 2003. Earlier version in Eighth ACM Conference on Computer and Communications Security (CCS-8), ACM Press, 2001.

[2] Mihir Bellare, Anand. Desai, Eron Jokipii, Philip Rogaway, *A Concrete Security Treatment of Symmetric Encryption*, proceedings of the 38th Annual Symposium on Foundations of Computer Science, IEEE, 1997, pp. 394–403.

[3] Eli Biham, Orr Dunkelman, *A Framework for Iterative Hash Functions - HAIFA*, Second NIST Cryptographic Hash Workshop, 2006.

[4] Joan Daemen and Vincent Rijmen, *AES Proposal: Rijndael* NIST AES proposal, 1998.

[5] D. W. Davies and W. L. Price, *The Application of Digital Signatures Based on Public-Key Cryptosystems*, Proceeding of Fifth International Computer Communications Conference, pp. 525–530, 1980.

[6] Richard Drews Dean, *Formal Aspects of Mobile Code Security*, Ph.D Dissertation, Princeton University, January 1999.

[7] Whitfield Diffie and Martin Edward Hellman, *Privacy and Authentication: An Introduction to Cryptography*, Proceedings of the IEEE, Vol. 67, No. 3, pp. 397-427, March 1979.

[8] Danny Dolev, Cynthia Dwork, Moni Naor, *Non-Malleable Cryptography*, SIAM Journal on computing, Vol. 30, No. 2, pp. 391–437, 2000.

[9] Morris Dworkin, *Recommendation for Block Cipher Modes of Operation - Methods and Techniques*, NIST Special Publication 800-38A, 2001.

[10] Brian Gladman, *AES Source Code*, http://www.gladman.me.uk/.

[11] Shafi Goldwasser and Silvio Micali, *Probabilistic Encryption*, Special issue of Journal of Computer and Systems Sciences, Vol. 28, No. 2, pp. 270-299, April 1984.

[12] Martin E.Hellman, *A Cryptanalytic Time-Memory Tradeoff*, IEEE Transactions on Information Theory, Vol. 26, pp. 401-406, July 1980.

[13] John Kelsey and Bruce Schneier, *Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work*, EUROCRYPT 2005, pp. 474-490.

[14] Moses Liskov, Ronald L. Rivest, David Wagner, *Tweakable Block Ciphers*, Advances in Cryptology – CRYPTO 2002, Lecture Notes in Computer Science 2442, pp. 31–46, Springer-Verlag, 2002.

[15] Bart Preneel, René Govaerts and Joos Vandewalle, *Hash Functions Based on Block Ciphers: A Synthetic Approach*, Crypto 1993, pp. 368-378.

[16] Philip Rogaway, *Nonce-Based Symmetric Encryption*, Fast Software Encryption (FSE) 2004, Lecture Notes in Computer Science vol. 3017, pp. 348-359, Springer, 2004.

[17] US National Institute of Standards and Technology, *Advanced Encryption Standard*, Federal Information Processing Standards Publications No. 197, 2001.

A Security Proofs

A.1 Proof of Lemma 1

Lemma 1 *Let X be a reasonable mode. Then* AX *is a reasonable* AMode.

Proof. AX is built by replacing the instances of a block cipher in X with instances of an ABC. Let $\tilde{E}$ be an ABC with a block size of n bits, and let E be a block cipher with a block size of n bits. We show that AX fulfills all the conditions of a reasonable AMode.

1. X is a reasonable mode of operation and therefore, $X_K^E(\cdot)$ is a length preserving encryption for every key K. Therefore, $\mathrm{AX}_{\tilde{\mathrm{K}},\mathrm{S}}^{\tilde{\mathrm{E}}}$, by its definition, is also a length preserving encryption for every key K and a salt S. The number of plaintext blocks in $\mathrm{AX}_{\tilde{\mathrm{K}},\mathrm{S}}^{\tilde{\mathrm{E}}}$ is equal to the number of plaintext blocks in X_K^E and the number of ciphertext blocks in $\mathrm{AX}_{\tilde{\mathrm{K}},\mathrm{S}}^{\tilde{\mathrm{E}}}$ is equal to the number of ciphertext blocks in X_K^E. Therefore, the number of plaintext blocks in $\mathrm{AX}_{\tilde{\mathrm{K}},\mathrm{S}}^{\tilde{\mathrm{E}}}$ equals the number of ciphertext blocks. Moreover, all the plaintext and ciphertext blocks of $\mathrm{AX}_{\tilde{\mathrm{K}},\mathrm{S}}^{\tilde{\mathrm{E}}}$ are n-bits blocks.
2. Let f_1 and f_2 be the functions such that
$$C_i = f_2(E_K(f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})), IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}).$$
By its definition, AX is the AMode in which
$$C_i = f_2(\tilde{E}_{K,S,i}(f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})), IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}).$$
   (a) By its definition, $f_2(\cdot; IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})$ is a permutation over $\{0,1\}^n$.
   (b) Denote $g_{E,K,IV,i,M_1,...,M_{i-1},C_1,...,C_{i-1}}(M_i)$
   $= f_2(E_K(f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})), IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})$,
   and denote $\tilde{g}_{\tilde{E},K,S,IV,i,M_1,...,M_{i-1},C_1,...,C_{i-1}}(M_i)$
   $= f_2(\tilde{E}_{K,S,i}(f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})), IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})$. Let $\tilde{E}^{(S,i)}$ be the block cipher that is defined by $\tilde{E}_K^{(S,i)}(M) = \tilde{E}_{K,S,i}(M)$. Now we can write:
$$\tilde{g}_{\tilde{E},K,S,IV,i,M_1,...,M_{i-1},C_1,...,C_{i-1}}(M_i) = g_{\tilde{E}^{(S,i)},K,IV,i,M_1,...,M_{i-1},C_1,...,C_{i-1}}(M_i).$$
   The function on the right of the last equation is a permutation. The function on the left side of the equation ($\tilde{g}$) equals the i'th block of $\mathrm{AX}_{\tilde{\mathrm{K}},\mathrm{S}}^{\tilde{\mathrm{E}}}$. Thus in AX, C_i is a permutation of M_i.

QED.

A.2 Proof of Theorem 1

Theorem 1 *Let X be a reasonable* AMode. *Let E be an* ABC. *Every possible total length of queries σ that allows the adversary to be salt-respecting satisfies* $\mathrm{Sec}_{X^E}^{\mathrm{ind-CPA}}(\sigma) \leq \mathrm{Sec}_E^{\mathrm{prp}}(\sigma)$.

Proof. Let A be an ind-CPA adversary against X^E that uses q queries of total length of σ blocks and that achieves the maximal advantage such an adversary can achieve, i.e., $\mathrm{Adv}_{X^E}^{\mathrm{ind-CPA}}(A) = \mathrm{Sec}_{X^E}^{\mathrm{ind-CPA}}(\sigma)$. We build an adversary B against E that uses σ queries and such that $\mathrm{Adv}_E^{\mathrm{prp}}(B) = \mathrm{Adv}_{X^E}^{\mathrm{ind-CPA}}(A)$.

Let O_B be the oracle that answers B's queries (O_B can be either $E_K(\cdot,\cdot,\cdot)$ or $\pi(\cdot,\cdot,\cdot)$). B simulates an oracle O for A. O works as follows: whenever A makes a query $Q = (S, M = M_1||...||M_m)$ to O, B makes m queries to its own oracle O_B: $Q_i = (S, i, x_i)$ for $1 \leq i \leq m$, and gets an answer y_i, where x_i is the plaintext input of the i'th instance of E in X (note that B is able to calculate x_i when necessary). From the answers it gets for its queries, B calculates $C_i = f_2(y_i, IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})$ and returns $C = C_1||...||C_m$. B lets A run on O and answers as A.

First, we notice that when $O_B = E_K(\cdot,\cdot,\cdot)$ then $O = O_{X_K^E}(\cdot,\cdot)$. This follows immediately from the building of O. Second, we claim that when $O_B = \pi(\cdot,\cdot)$ then $O = O_R^{X^E}(\cdot,\cdot)$. This is because X is a reasonable mode and as such, C_i is a permutation of y_i which is a random string in that case. Therefore, for every $1 \leq i \leq m$, C_i is actually chosen uniformly at random from $\{0,1\}^n$ and thus, C is a random string of length $|X_{K,S}^E(M)|$.

Now, we can write:
$$\begin{aligned}\mathrm{Adv}_E^{\mathrm{prp}}(B) &= \Pr\left[B^{E_K(\cdot,\cdot,\cdot)} = 1 \middle| K \xleftarrow{R} \{0,1\}^k\right] - \Pr\left[B^{\pi(\cdot,\cdot,\cdot)} = 1 \middle| \pi \xleftarrow{R} SCPerm(s,c,n)\right] \\ &= \Pr\left[A^{O_{X_K^E}(\cdot,\cdot)} = 1 \middle| K \xleftarrow{R} \{0,1\}^k\right] - \Pr\left[A^{\pi(\cdot,\cdot)} = 1\right] = \mathrm{Adv}_{X^E}^{\mathrm{ind-CPA}}(A).\end{aligned}$$

QED.

A.3 Proof of Theorem 2

Theorem 2 *Let $\tilde{E}$ be an ideal* ABC *and let E be an ideal block cipher. For every σ it holds that* $\mathrm{Sec}^{\mathrm{ind-CCA}}_{ECB^E}(\sigma) \geq \mathrm{Sec}^{\mathrm{ind-CCA}}_{\mathrm{AECB}^{\tilde{\mathrm{E}}}}(\sigma)$.

In order to prove Theorem 2 we need to prove the following two lemmas.

Lemma 5 *Let E be an ideal* ABC. *Let A be an* ind-CCA *adversary against $AECB^E$ that uses no more than a total of σ blocks in its queries. Then there exists a deterministic* ind-CCA *adversary A' that uses no more than a total of σ blocks in its queries and such that* $\mathrm{Adv}^{\mathrm{ind-CCA}}_{AECB^E}(A) \leq \mathrm{Adv}^{\mathrm{ind-CCA}}_{AECB^E}(A')$.

Proof. Consider a representation of A as a decision tree. Let v be a state in the decision tree where A performs a coin-flip. Let $u_1, ..., u_m$ be the children of v, so upon arriving to state v, A flips a coin and, according to the result, decides to which one of v's children it should move. Let A_v be the event in which A reaches state v. The advantage of A can be written as

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{ind-CCA}}_{AECB^E}(A) &= \Pr\left[A^{AECB^E_K(\cdot,\cdot)} = 1 \middle| K \xleftarrow{R} \{0,1\}^k\right] - \Pr\left[A^{O_R(\cdot,\cdot)} = 1\right] \\
&= \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_E\right] - \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_R\right] \\
&= \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_E \wedge A_v\right] \cdot \Pr\left[A_v \middle| O_E\right] + \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_E \wedge \neg A_v\right] \cdot \Pr\left[\neg A_v \middle| O_E\right] \\
&\quad - \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_R \wedge A_v\right] \cdot \Pr\left[A_v | O_R\right] - \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_R \wedge \neg A_v\right] \cdot \Pr\left[\neg A_v \middle| O_R\right].
\end{aligned}$$

We note that $\Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| A_v\right] = \sum_{i=1}^{m} \Pr\left[u_i \middle| A_v\right] \cdot \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| A_{u_i}\right]$.

Define

$$j = \arg\max_{1 \leq i \leq m} \{ \Pr\left[A_v \middle| O = O_E\right] \cdot \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_E \wedge A_{u_i}\right] - \Pr\left[A_v \middle| O = O_R\right] \cdot \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_R \wedge A_{u_i}\right] \}.$$

Now, consider the adversary $A^{(v)}$ of which decision tree is identical to the decision tree of A, besides the fact that upon reaching the state v, $A^{(v)}$ does not flip a coin, but moves to u_j.

For any state w, denote by O^w_E the event in which $O = O_E$ and the adversary (which can be either A or $A^{(v)}$ – according to the context) reaches state w. Denote by $O^{\neg w}_E$ the event in which $O = O_E$ and the adversary does not reach state w. Similarly, denote by O^w_R and $O^{\neg w}_R$ the events in which $O = O_R$ and the adversary reaches or not, respectively, state w.

Obviously, $A^{(v)}$ performs one flip-coin less then A and

$$
\begin{aligned}
&\mathrm{Adv}^{\mathrm{ind-CCA}}_{AECB^E}(A^{(v)}) = \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O = O_E\right] - \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O = O_R\right] \\
&= \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_E^v\right] \cdot \Pr\left[{A^{(v)}}_v \middle| O_E\right] + \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_E^{\neg v}\right] \cdot \Pr\left[\neg {A^{(v)}}_v \middle| O_E\right] \\
&\quad - \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_R^v\right] \cdot \Pr\left[{A^{(v)}}_v \middle| O_R\right] - \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_R^{\neg v}\right] \cdot \Pr\left[\neg {A^{(v)}}_v \middle| O_R\right] \\
&= \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_E^{u_j}\right] \cdot \Pr\left[{A^{(v)}}_v \middle| O_E\right] - \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_R^{u_j}\right] \cdot \Pr\left[{A^{(v)}}_v \middle| O_R\right] \\
&\quad + \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_E^{\neg v}\right] \cdot \Pr\left[\neg {A^{(v)}}_v \middle| O_E\right] - \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_R^{\neg v}\right] \cdot \Pr\left[\neg {A^{(v)}}_v \middle| O_R\right] \\
&\geq \sum_{i=1}^{m} \Pr\left[u_i \middle| A_v\right] \cdot \left(\Pr\left[A_v \middle| O_E\right] \cdot \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O_E^{u_i}\right] - \Pr\left[A_v \middle| O_R\right] \cdot \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_R^{u_i}\right]\right) \\
&\quad + \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_E^{\neg v}\right] \cdot \Pr\left[\neg {A^{(v)}}_v \middle| O_E\right] - \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_R^{\neg v}\right] \cdot \Pr\left[\neg {A^{(v)}}_v \middle| O_R\right] \\
&= \sum_{i=1}^{m} \Pr\left[u_i \middle| A_v\right] \cdot \Pr\left[A_v \middle| O_E\right] \cdot \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O_E^{u_i}\right] - \sum_{i=1}^{m} \Pr\left[u_i \middle| A_v\right] \cdot \Pr\left[A_v \middle| O_R\right] \cdot \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O_R^{u_i}\right] \\
&\quad + \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_E^{\neg v}\right] \cdot \Pr\left[\neg {A^{(v)}}_v \middle| O_E\right] - \Pr\left[{A^{(v)}}^{O(\cdot,\cdot)} = 1 \middle| O_R^{\neg v}\right] \cdot \Pr\left[\neg {A^{(v)}}_v \middle| O_R\right] \\
&= \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O_E^v\right] \cdot \Pr\left[A_v \middle| O_E\right] + \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O_E^{\neg v}\right] \cdot \Pr\left[\neg A_v \middle| O_E\right] \\
&\quad - \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O_R^v\right] \cdot \Pr\left[A_v \middle| O_R\right] - \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O_R^{\neg v}\right] \cdot \Pr\left[\neg A_v \middle| O_R\right] \\
&= \mathrm{Adv}^{\mathrm{ind-CCA}}_{AECB^E}(A).
\end{aligned}
$$

By induction, we can create an adversary A' which does not perform any flip-coins and for which $\mathrm{Adv}^{\mathrm{ind-CCA}}_{AECB^E}(A') \geq \mathrm{Adv}^{\mathrm{ind-CCA}}_{AECB^E}(A)$.

QED.

Definition 17 *The sequence of queries made by an* ind-CCA *adversary A to an oracle O, and the answers received for them is denoted by* stream*.*

When discussing a *deterministic* adversary A, the output of the adversary in a single run depends only on the stream generated by A's run (and since this is a deterministic adversary, the output actually depends only on the answers received by the oracle). Thus, the contribution of a single stream u that A accepts to the advantage of A is the difference between the probability of getting u when A runs on O_E and the probability of getting u when A runs on O_R. Formally, denote by U the set of all streams upon which A outputs '1'. Then, $\mathrm{Adv}^{\mathrm{ind-CCA}}_{\mathcal{E}}(A) = \sum_{u \in U} (\Pr[u|O = O_E] - \Pr[u|O = O_R])$.

Lemma 6 *Let X be a reasonable* AMode*. Let E be an ideal* ABC*. Let A be a deterministic* ind-CCA *adversary against X^E that uses no more than a total of σ blocks in its queries, and such that $\mathrm{Adv}^{\mathrm{ind-CCA}}_{X^E}(A) \geq 0$. If there are $1 \leq \ell$ streams that A accepts (outputs '1') then there exists a deterministic adversary B that uses no more than a total of σ blocks in its queries, accepts $2^n \leq \ell'$ streams and such that $\mathrm{Adv}^{\mathrm{ind-CCA}}_{X^E}(B) \geq \mathrm{Adv}^{\mathrm{ind-CCA}}_{X^E}(A)$.*

Proof. If $2^n \leq \ell$ then $B = A$ and we are done. If $1 \leq \ell < 2^n$ then let u be a stream that A accepts and such that $\Pr[u|O = O_E] - \Pr[u|O = O_R] \geq 0$ (obviously, there is at least one such stream). Let S_1 be the first salt in u and let C_1 be the answer to the first block of the first query in u. Let π be a permutation over $\{0,1\}^n$. Consider the stream u_π which is identical to u accepts that every answer block c to a query block with $S = S_1, t = 1$ in u is replaced with $\pi(c)$ in u_π. Since E is an ideal ABC then for each key, K, the permutation $E_{K,S_1,1}(\cdot)$ is chosen uniformly at random. Therefore, $\Pr[u_\pi|O = O_E] = \Pr[u|O = O_E]$. Of course, $\Pr[u_\pi|O = O_R] = \Pr[u|O = O_R] = 2^{-|u| \cdot n}$, where $|u|$ is the number of blocks in the queries of u. We obtain that $\Pr[u_\pi|O = O_E] - \Pr[u_\pi|O = O_R] = \Pr[u|O = O_E] - \Pr[u|O = O_R] \geq 0$.

Now, let U_A be the set of streams that A accepts. Let $\{\pi_i\}_{i=1}^{2^n}$ be a set of 2^n permutations such that $\pi_i(C_1) \neq \pi_j(C_1)$ for every $i \neq j$. Let $U_B = U_A \bigcup_{i=1}^{2^n} u_{\pi_i}$ and let B be the adversary that accepts the streams that are in U_B and rejects the other streams. B accepts at least 2^n streams and

$$\begin{aligned}\mathrm{Adv}_{X^E}^{\mathrm{ind-CCA}}(B) &= \sum_{u \in U_B} (\Pr[u|O = O_E] - \Pr[u|O = O_R]) \\ &= \sum_{u \in U_A} (\Pr[u|O = O_E] - \Pr[u|O = O_R]) + \overbrace{\sum_{u \in U_B \setminus U_A} (\Pr[u|O = O_E] - \Pr[u|O = O_R])}^{\geq 0} \\ &\geq \mathrm{Adv}_{X^E}^{\mathrm{ind-CCA}}(A)\end{aligned}$$

QED.

Now we can prove Theorem 2.

Proof. Let A be an ind-CCA adversary against $AECB^{\tilde{E}}$ that uses no more than a total of σ blocks in its queries. W.l.g., A is deterministic (following Lemma 5). In case A does not accept any streams then $\mathrm{Adv}_{AECB^{\tilde{E}},A}^{\mathrm{ind-CCA}} = 0$. In case that there are streams that A accepts then w.l.g., there are at least 2^n such streams (following Lemma 6).

Every sequence that A accepts is generated by O_R with probability of at least $2^{-n\sigma}$ and therefore, in case that there are sequences that A accepts, $\Pr[A^{O_R(\cdot,\cdot)} = 1] \geq 2^{-n(\sigma-1)}$. Therefore, we can write

$$\mathrm{Adv}_{AECB^{\tilde{E}}}^{\mathrm{ind-CCA}}(A) = \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_E\right] - \Pr\left[A^{O(\cdot,\cdot)} = 1 \middle| O = O_R\right] \leq 1 - 2^{-n(\sigma-1)},$$

and conclude that $\mathrm{Sec}_{\mathrm{AECB^{\tilde{E}}}}^{\mathrm{ind-}cca}(\sigma) \leq 1 - 2^{-n(\sigma-1)}$.

Now, consider an ind-CCA adversary B that works against ECB^E and is defined as follows: B asks a single query of σ blocks, all equal and accepts (outputs '1') if and only if all σ blocks of the answer are equal.

Obviously, $\Pr[B^{O_E(\cdot)} = 1] = 1$ and $\Pr[B^{O_R(\cdot)} = 1] = 2^{-n(\sigma-1)}$. Therefore we get

$$\mathrm{Sec}_{ECB^E}^{\mathrm{ind-CCA}}(\sigma) \geq \mathrm{Adv}_{ECB^E}^{\mathrm{ind-CCA}}(B) = 1 - 2^{-n(\sigma-1)} \geq \mathrm{Sec}_{\mathrm{AECB^{\tilde{E}}}}^{\mathrm{ind-CCA}}(\sigma).$$

QED.

A.4 Proof of Theorem 3

Theorem 3 *Let X be a reasonable* AMode*, and let E be an ideal* ABC*. Then* $\mathrm{Sec}_{X^E}^{\mathrm{sem-ATK}}(\sigma) = 0$*, where* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA}\}$.

In order to prove Theorem 3 we need to prove the following lemmas.

Lemma 7 *Let X be a reasonable AMode. Let E be an ideal* ABC*. Then, for every message $M = M_1||...||M_m$, and every $1 \leq i \leq m$, $c \in \{0,1\}^{|M_i|}$, $K \in \{0,1\}^k$, $S \in \{0,1\}^s$, it holds that $\Pr[C_i = c] = 2^{-|M_i|}$, where $C = X_{K,S}^E(M)$ and when the randomness results from choosing the random permutations $E_{K,S,t}$ for different counter values t.*

Proof. Since E is an ideal ABC then $E_{K,S,i}$ is a random permutation chosen uniformly from the permutation spaces. Therefore, for every possible $x_i \in \{0,1\}^{|M_i|}$ and for every $y \in \{0,1\}^{|M_i|}$ we get that $\Pr[y_i = y] = \Pr[E_{K,S,i}(x_i) = y] = 2^{-|M_i|}$.

Since X is a reasonable AMode in which C_i is a permutation of y_i then for every $c \in \{0,1\}^{|M_i|}$ we get that $\Pr[C_i = c] = 2^{-|M_i|}$.

QED.

Lemma 8 *Let X be a reasonable AMode. Let E be an ideal* ABC*. Then, for every message M, and every $c \in \{0,1\}^{|M|}$, $K \in \{0,1\}^k$, $S \in \{0,1\}^s$, it holds that $\Pr[C = c] = 2^{-|M|}$, where $C = X^E_{K,S}(M)$ and when the randomness results from choosing the random permutations $E_{K,S,t}$ for different counter values t.*

Proof. According to Lemma 7, $\Pr[C_i = c_i] = 2^{-|M_i|}$ for every $1 \leq i \leq m$, (where m is the number of blocks in M) and for every $c_i \in \{0,1\}^{|M_i|}$. Since X is a reasonable AMode then all the instances of E in X define independent permutations and therefore,

$$\Pr[C = c] = \prod_{i=1}^{m} \Pr[C_i = c_i] = \prod_{i=1}^{m} 2^{-|M_i|} = 2^{-\sum_{i=1}^{m} |M_i|} = 2^{-|M|}.$$

QED.

Lemma 9 *Let X be a reasonable AMode, let E be an ideal* ABC*, and let M be a message chosen at random according to some valid distribution function γ (i.e., γ gives non-zero probability only to messages of length $\ell = |M|$).*

Then, for every $m, c \in \{0,1\}^\ell$, $K \in \{0,1\}^k$, $S \in \{0,1\}^s$, it holds that $\Pr[M = m|C = c] = \gamma(m)$, where $C = X^E_{K,S}(M)$ and when the randomness results from choosing the random permutations $E_{K,S,t}$ for different counter values t.

Proof. In Lemma 8 we show that $\Pr[C = c] = 2^{-|M|}$ for every message M, and every $c \in \{0,1\}^{|M|}$, $K \in \{0,1\}^k$, $S \in \{0,1\}^s$. Therefore, we can write $\Pr[C = c|M] = 2^{-|M|}$. By applying Bayes' law, and following that $\Pr[C = c] \neq 0$:

$$\Pr[M = m|C = c] = \frac{Pr[M = m] \cdot \Pr[C = c|M = m]}{Pr[C = c]} = \frac{\gamma(M) \cdot 2^{-\ell}}{2^{-\ell}} = \gamma(M).$$

QED.

Now we can prove Theorem 3.

Proof. For every (α, f) that A can output it holds that

$$\Pr[\alpha = f(M^*)|C^*] = \sum_{\substack{m:\gamma(m)>0,\\ f(m)=\alpha}} Pr[m|C^*] = \sum_{m:f(m)=\alpha} \gamma(m).$$

On the other hand,

$$\Pr[\alpha = f(M')] = \sum_{\substack{m:\gamma(m)>0,\\ f(m)=\alpha}} \gamma(m) = \Pr[\alpha = f(M^*)|C^*].$$

QED.