ABC - A New Framework for Block Ciphers

Uri Avraham
ABC - A New Framework for Block Ciphers

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Uri Avraham

Submitted to the Senate of the Technion — Israel Institute of Technology
Shvat 5772 Haifa February 2012
The research thesis was written under the supervision of Prof. Eli Biham and Dr. Orr Dunkelman in the Computer Science Department.

I would like to express my gratitude to my advisors, Prof. Eli Biham, and Dr. Orr Dunkelman for their guidance during my studies. I thank them both.

The generous financial support of the Technion and MAFAT is gratefully acknowledged.
Contents

Abstract

Abbreviations and Notations 2

1 Introduction 3

1.1 Traditional Block Ciphers . . . . . . . . . . . . . . . . . . . 3
1.2 Tweakable Block Ciphers . . . . . . . . . . . . . . . . . . . 5
1.3 Our Contribution . . . . . . . . . . . . . . . . . . . . . . . . 6
   1.3.1 Advanced Block Ciphers . . . . . . . . . . . . . . . . 6
   1.3.2 Our Results . . . . . . . . . . . . . . . . . . . . . . . . 7
1.4 Outline of This Thesis . . . . . . . . . . . . . . . . . . . . . 8

2 Traditional Modes of Operation 9

2.1 Electronic Code Book (ECB). . . . . . . . . . . . . . . . . 9
2.2 Cipher Block Chaining (CBC) . . . . . . . . . . . . . . . . . 10
2.3 Output Feedback (OFB) . . . . . . . . . . . . . . . . . . . . 11
2.4 Cipher Feedback (CFB) . . . . . . . . . . . . . . . . . . . . 11
2.5 Counter Mode (CTR) . . . . . . . . . . . . . . . . . . . . . . 13

3 Advanced Block Ciphers and Advanced Modes of Operation 14

3.1 Advanced Block Ciphers . . . . . . . . . . . . . . . . . . . . 14
3.2 Advanced Modes of Operation . . . . . . . . . . . . . . . . . 16
   3.2.1 Reasonable Modes of operation and Reasonable AModes 17
3.3 Encryption Schemes . . . . . . . . . . . . . . . . . . . . . . 20

4 Security Analysis of ABCs 22

4.1 Attacks that Take Advantage of a Repeating Permutation . . 22
4.1.1 Dictionary Attacks .................................. 22
4.1.2 Statistical Attacks .................................. 23
4.1.3 Time-Memory Tradeoff Attacks ..................... 23
4.2 Theoretical Security Notions for Symmetric Key Encryption ... 24
4.2.1 Adversaries ......................................... 24
4.2.2 Distinguishers for Block Ciphers and for ABCs .... 26
4.2.3 Indistinguishability ................................. 28
4.2.4 Semantic Security .................................. 40

5 The Relation to Other Cryptographic Primitives .......... 47
5.1 Stream Ciphers ........................................ 47
5.2 Hash Functions ....................................... 48
5.2.1 Using Davies-Meyer Construction ................. 48

6 Simple ABCs Based on AES .............................. 51
6.1 Proposals for ABCs ................................... 51
6.1.1 ABC1 .............................................. 51
6.1.2 ABC2 .............................................. 52
6.1.3 ABC3 .............................................. 53
6.2 Security of Our Proposals ............................... 54
6.2.1 Equivalent Keys ................................... 54
6.2.2 Security of ABC1 Against Any Known or Chosen Plain-
text Attack ............................................. 55
6.2.3 Security of ABC2 and ABC3 Against Linear Attacks . 56
6.2.4 Security of ABC2 and ABC3 Against Extended Diff-
erential Attacks ...................................... 57
6.2.5 A Pitfall of Using ABC3 in a Davies-Meyer Construction 58
6.3 Performance of ABC1, ABC2, and ABC3 ............... 59

Abstract in Hebrew ...........................................
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A traditional block cipher</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>ECB mode of operation</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>CBC mode of operation</td>
<td>11</td>
</tr>
<tr>
<td>2.3</td>
<td>OFB mode of operation</td>
<td>12</td>
</tr>
<tr>
<td>2.4</td>
<td>CFB mode of operation</td>
<td>12</td>
</tr>
<tr>
<td>2.5</td>
<td>CTR mode of operation</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>Advanced block cipher</td>
<td>15</td>
</tr>
<tr>
<td>4.1</td>
<td>TBC mode of operation. Note that $C_i$ is used as the tweak of the $i + 1$ block cipher call.</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>Semantic security. The adversary wins if $\alpha = f(M^*)$</td>
<td>42</td>
</tr>
</tbody>
</table>
List of Tables

1.1 An overview of the security results. .......................... 7
4.1 The indistinguishability security of different modes of operation and AModes. ........................................... 41
6.1 The minimum number of active S Boxes in a differential characteristic as a function of the number of rounds. ...... 58
6.2 Performance of the different implementations .................. 60
Abstract

There is no arguing about the importance of encryption in today’s world. We make frequent use, sometimes unaware, of encryption. We encrypt information that we send over the Internet, cellular phone calls, cable broadcasts, and more.

A large portion of this encryption is done using block ciphers. Block ciphers provide us with fast encryption, and can be used as building blocks of various encryption schemes (called modes of operation).

Block ciphers have always conformed to a specific interface with two inputs – a plaintext block to encrypt and a key – resulting in a ciphertext block of the same size as the plaintext block.

In this thesis we suggest a new framework for block ciphers named Advanced Block Cipher, or shortly ABC. This framework defines a new interface, and new modes of operation. ABC introduces two additional non-secret parameters that ensure that each call to the underlying block cipher behaves like a different pseudo-random permutation. It therefore ensures that attacks that require more than one block of encrypted data cannot apply. In particular, this framework protects against dictionary attacks, differential and linear attacks, and eliminates the weaknesses of most modes of operation. This new framework shares a common structure with the HAIFA hash function framework, 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 several instances of ABCs.
### Abbreviations and Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$</td>
<td>Key</td>
</tr>
<tr>
<td>$k$</td>
<td>Key size in bits</td>
</tr>
<tr>
<td>$S$</td>
<td>Salt</td>
</tr>
<tr>
<td>$s$</td>
<td>Salt size in bits</td>
</tr>
<tr>
<td>$t$</td>
<td>Counter</td>
</tr>
<tr>
<td>$c$</td>
<td>Counter size in bits</td>
</tr>
<tr>
<td>$M$</td>
<td>Plaintext message</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of blocks of a message $M (M = M_1</td>
</tr>
<tr>
<td>$C$</td>
<td>Ciphertext</td>
</tr>
<tr>
<td>$M_i, C_i$</td>
<td>The $i$'th block of the plaintext $M$, or the ciphertext $C$</td>
</tr>
<tr>
<td>$</td>
<td>Y</td>
</tr>
<tr>
<td>$X^E_K(M)$</td>
<td>The encryption of the message $M$ under the block cipher $E$, using the key $K$, and the mode of operation $X$</td>
</tr>
<tr>
<td>$X^{-1}_K(C)$</td>
<td>The decryption of the ciphertext $C$ under the block cipher $E$, using the key $K$, and the mode of operation $X$</td>
</tr>
<tr>
<td>$X^E_{K,S}(M)$</td>
<td>The encryption of the message $M$ under the ABC $E$, using the key $K$, the salt $S$, and the AMode $X$</td>
</tr>
<tr>
<td>$X^{-1}_{K,S}(C)$</td>
<td>The decryption of the ciphertext $C$ under the ABC $E$, using the key $K$, the salt $S$, and the AMode $X$</td>
</tr>
<tr>
<td>$a \overset{R}{\leftarrow} A$</td>
<td>The operation of choosing an item $a$ uniformly at random out of a set $A$</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Encryption is being used nowadays in almost any mean of electronic communications. Sensitive information transferred over the internet is being encrypted, as well as cellular phone calls, military radio communication, and many more. There are two types of encryption methods. The first is symmetric encryption (or secret key encryption), in which the communicating parties share a secret key known only to them. Examples for symmetric encryption schemes are block ciphers and stream ciphers. The second is public-key encryption, in which a public key is known to everyone and is used to encrypt messages, while the decryption key (known as the secret key or private key) is known only to the owner of the key.

This thesis concentrate on block ciphers. Two users who wish to communicate using the block ciphers must have a common secret key in advance. Knowing the secret key enables to encrypt and decrypt messages easily, but it should be difficult doing so without it. It should be therefore computationally difficult for an eavesdropper (i.e., a party that does not know the secret) that had gained plaintexts and their corresponding ciphertexts to learn any information on the secret key, or to be able to encrypt or decrypt additional blocks.

1.1 Traditional Block Ciphers

Block ciphers are symmetric encryption primitives that allow to encrypt messages of fixed size – the block size. Traditional block ciphers have always
conformed to a simple API that consists of two inputs – the message block and the secret key, and one output – the ciphertext block. Formally, a block cipher is a function of the form $E : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$, where $n$ is the block size, and $k$ is the key size. For every key $K \in \{0,1\}^k$, the function $E_K : \{0,1\}^n \to \{0,1\}^n$ is a permutation. Figure 1.1 describes a traditional block cipher. We denote the plaintext by $M$, and the ciphertext by $C = E_K(M)$.

The first modern civilian block cipher is considered to be Lucifer [33], which was developed by IBM in the early 1970’s. A variant of Lucifer was adopted by the National Bureau of Standards (later called NIST) as the Data Encryption Standard (DES) [35]. Over the years, DES has been studied thoroughly, and many attacks were devised to break it [7, 9, 27, 32]. Still, DES remained the standard de-facto for many years, as the attacks were not practically applicable in the real world. However, the relatively short key size of DES (56 bits) made exhaustive search applicable [15, 18, 25, 31, 37] on the faster and stronger hardware of the 1990’s, and via distributed computation over the internet.

The short key size, short block size, and the fact that DES was relatively inefficient in software implementation, led NIST to initiate a public process for finding a replacement for DES. The proposals were required to have 128-bit blocks, and to support three key sizes (128, 192 and 256 bits). 15 candidates were submitted and thoroughly analyzed. Eventually, on October 2000, the Rijndael cipher [10] was chosen and announced as the Advanced Encryption Standard (AES) [34].

Figure 1.1: A traditional block cipher
Since a block cipher has a fixed block size, the encryption of long messages is done by iterating the block cipher operation in a construction called *mode of operation*. Four modes of operation were accepted by NIST as official modes of operation for DES [36] – ECB, CBC, CFB, and OFB. Another mode of operation – CTR, was added as a fifth NIST recommendation in 2001 [17]. In ECB every block is encrypted separately and independently – this fact makes it vulnerable to various attacks, and it is therefore not recommended for regular use. In CBC every ciphertext block is dependant on the corresponding message block and on all the previous message blocks as well. This makes CBC stronger than ECB, and it is the most popular mode of operation, but still, when decrypting a ciphertext, each message block depends only on two message blocks, which may be considered a weakness in certain circumstances. OFB and CTR modes both generate a stream that is XORed into the plaintext to generate the ciphertext. Since the stream is generated out of a short key, and using a deterministic algorithm, these modes of operation expose some security weaknesses that are not related to the underlying block cipher. The example of OFB with blocks of 63 bits or fewer also emphasizes the need of provable security. These variants of OFB have been considered secure at the time. However, they were later found to be less secure than expected [12]\footnote{64-bit OFB remains secure.}. Following this observation, these insecure versions were discarded by NIST. The CFB mode generates a stream in which each block depends also on previous message blocks. As in CBC, it is sometimes possible to gain information about the plaintext only by looking at the ciphertext. All these modes of operation and their pitfalls are discussed in detail in Chapter 2.

### 1.2 Tweakable Block Ciphers

In [26], the authors suggest a new kind of block ciphers that has another input called *tweak*, in addition to the plaintext and the key inputs. This tweak is used to provide some variability to the encryption, without necessarily ensuring that each block uses a different tweak or a different permutation. The main motivation behind the tweak is to provide an ad-hoc parameter for feedbacks that were traditionally fed using other parameters (mainly the plaintext). In [26], 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. While the TAE mode of operation provides security measures of an ABC, and can be considered as an ABC (and is functionally equivalent to an ABC instance we call AECB – defined in Chapter 3), other modes of operation use the tweak in different ways that do not assure that the tweak is unique for every call to the block cipher.

ABCs differ from tweakable block ciphers in the guidelines of use and in the inherited separation between the salt and the counter. While the TAE mode and a certain mode for ABCs are functionally similar, the implicit separation between the salt and the counter, provided by ABC, captures our understanding of the need for the two different parameters. Moreover, this separation considers performance needs, by allowing the counter mixing, which is done for every message block, to be a faster (even if more superficial) process, while more time and effort can be invested in the less frequent process of the salt mixing.

It is also important to note that the motivations behind the two frameworks are different. While the main motivation behind the tweakable block cipher is to introduce the feedback separately from the plaintext block, the motivation behind ABC is to ensure that no two encryptions of plaintext blocks are done using the same permutation, and thus increase security.

1.3 Our Contribution

1.3.1 Advanced Block Ciphers

The security pitfalls of the traditional modes of operation led us to suggest a new framework for block ciphers. We base our new framework on the same ideas that were used to improve hash functions in HAIFA [8]. 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. We analyze the security of our framework according to commonly-used security notions and prove that it is secure at least as the traditional framework.
1.3.2 Our Results

We analyze the security of our framework according to two commonly used security notions – indistinguishability from random bits, and semantic security. The indistinguishability notion aims to measure the ability of an adversary to distinguish between the output of the encryption (or decryption) operation and a string of random bits. The semantic security notion aims to measure the ability of an adversary that obtains some plaintext/ciphertext pairs to learn some non-trivial information on the decryption (or encryption) of another ciphertext (or plaintext). The two notions are described in details in Chapter 4. We show that our framework achieves the optimal security against the most important types of attacks (including all known and chosen plaintext attacks), and that it is at least as secure as the traditional framework in other cases. Table 1.1 overviews our security results.

In the second part of our work we suggest three algorithms for Advanced Block Ciphers – all based on AES. We analyze the security of our suggestions against practical security threats such as differential and linear attacks, and claim that they are secure. We also analyze the performance of our suggestions and show that the encryption time for two of them is only slightly longer than the encryption time for AES-128 – a small investment for the highly improved security.
1.4 Outline of This Thesis

In Chapter 2 we discuss the most common traditional modes of operation, and give some examples of their pitfalls. In Chapter 3 we introduce our new framework – we define the Advanced Block Cipher framework, and discuss the way of using it for encrypting long messages. In Chapter 4 we examine the security of our framework in terms of various security criteria, and compare it to the security of the traditional framework. In Chapter 5 we discuss the relation of our new framework to hash functions. Finally, in Chapter 6 we suggest several instances for ABC block ciphers that conform to the new framework and examine these suggestions.
Chapter 2

Traditional Modes of Operation

A block cipher has a fixed block size of $n$ bits. For encrypting messages longer than $n$ bits, *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.

Throughout this thesis we will denote by $M = M_1||...||M_m$ the message divided into $n$-bit blocks (where the last block might contain padding that was added to the original message), and by $C = C_1||...||C_m$ the ciphertext divided into $n$-bit blocks.

Next, we will review some prominent traditional modes of operation. We will note few security weaknesses of each mode. A more formal security treatment will be given in Chapter 4.

2.1 Electronic Code Book (ECB)

The simplest mode of operation is the Electronic Code Book (ECB) in which the message $M$ is padded to create a string of size which is a multiple of the block length $n$. Then the message is divided into $n$-bit blocks $M = M_1||...||M_m$, and each block $M_i$ is encrypted using the underlying block-cipher to create the ciphertext block $C_i = E_K(M_i)$. Figure 2.1 describes the
The ECB mode is not recommended for general use. When using it for encryption, an adversary can learn information about the plaintext only by looking at the ciphertext. For example, if \( C_i = C_j \) for some \( i, j \) then the adversary learns that \( M_i = M_j \). If \( C_i \neq C_j \) then the adversary learns that \( M_i \neq M_j \). Moreover, an adversary that gains a plaintext-ciphertext pair \((M = M_1 \| \cdots \| M_m, C = C_1 \| \cdots \| C_m)\) learns the encryption of \( M_i \) for every \( 1 \leq i \leq m \), and can later encrypt (decrypt) other messages (ciphertexts) of her own that are composed of the blocks that she had already learned, without knowing the secret key.

2.2 Cipher Block Chaining (CBC)

Another well-known mode of operation is the Cipher Block Chaining (CBC). As in ECB, the message \( M \) is padded and divided into \( m \) blocks, each block is of size of \( n \) bits. The plaintext block is XORed with the previous ciphertext block prior to its encryption. Formally, \( C_i = E_K(M_i \oplus C_{i-1}) \) for every \( 1 \leq i \leq m \), where \( C_0 = IV \) for some known initial value \( IV \). Figure 2.2 describes the CBC mode.

CBC is considered to be a secure mode and it is widely used in many applications. Nevertheless, in some cases it is possible to learn some information about the plaintext only by looking at the ciphertext. If \( C_i = C_j \) for some \( i \neq j \) then the adversary can learn that \( M_i \oplus M_j = C_{i-1} \oplus C_{j-1} \). When encrypting long messages of \( 2^{n/2} \) blocks or more, it is expected with
high probability, due to the birthday paradox, that there will be some \( i, j \) for which \( C_i = C_j \).

### 2.3 Output Feedback (OFB)

The output feedback mode of operation (OFB) generates a stream \( V = V_1||...||V_m \) that is XORed into the plaintext to generates the ciphertext. Formally, \( C_i = M_i \oplus V_i \), where \( V_i = E_K(V_{i-1}) \) for \( 1 \leq i \leq m \), and \( V_0 = IV \) for some initial value \( IV \). Figure 2.3 describes the OFB mode.

An adversary that is given a plaintext-ciphertext pair \( (M = M_1||...||M_m, C = C_1||...||C_m) \) and the value of \( IV \), can easily calculate the corresponding stream \( V_{IV} = V_1||...||V_m \). The adversary can encrypt or decrypt other messages using the same key and \( IV \) (and therefore protocols usually do not allow repeating an \( IV \)). Moreover, the adversary can encrypt or decrypt other messages using \( IV = V_i \), where \( 1 \leq i < m \).

### 2.4 Cipher Feedback (CFB)

The cipher feedback mode (CFB) is defined as follows: \( C_i = M_i \oplus E_K(C_{i-1}) \), for \( 1 \leq i \leq m \), where \( C_0 = IV \) for some initial value \( IV \). Figure 2.4 describes the CFB mode.

As in CBC mode, two equal ciphertext blocks imply some information about the plaintext. More specifically, if \( C_i = C_j \), then an adversary can learn that \( M_{i+1} \oplus M_{j+1} = C_{i+1} \oplus C_{j+1} \).
Figure 2.3: OFB mode of operation

Figure 2.4: CFB mode of operation
2.5 Counter Mode (CTR)

As OFB, Counter mode (CTR) also generates a stream $V = V_1 || ... || V_m$ that is XORed into the plaintext. As in OFB, the ciphertext block is calculated as $C_i = M_i \oplus V_i$. However, here $V_i = E_K(IV||i)$ for some nonce $IV$. A good practice is to never repeat the same $IV$ value and therefore never use the same $IV||i$ combination more than once. Figure 2.5 describes the CTR mode of operation.

When it is used correctly, CTR mode is considered to be a secure mode. It generates a pseudo-random stream that does not allow the adversary to learn much about the plaintext, when she is given only the ciphertext. Still, some information, even if not very significant, can be learned. In a CTR stream it is guaranteed that $V_i \neq V_j$ for every $i \neq j$. Therefore, an adversary can learn that $M_i \oplus M_j \neq C_i \oplus C_j$ for every $i \neq j$. In an ideal encryption scheme, even such a minor detail cannot be learned.
Chapter 3

Advanced Block Ciphers and Advanced Modes of Operation

As mentioned in Chapter 1, an advanced block cipher is a block cipher that has two additional inputs – a salt, and a counter. In this chapter we describe in detail the advanced block cipher framework, the new inputs, their purpose, and the way we recommend using them.

3.1 Advanced Block Ciphers

Definition 3.1 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., incremented by one for every new message).

Definition 3.2 A counter $t$ is a counter or a dithering, intended to introduce diversity between different blocks of the same message.

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

\[
E : \{0,1\}^n \times \{0,1\}^k \times \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n,
\]

such that the ciphertext block is computed by \( C = E_{K,S,t}(M) \), where \( E \) is an advanced block cipher, \( K \) 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 never repeats. To achieve this, we demand that the same key-salt combination never repeats for different messages, and that the same counter value does 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 real constraint.

As mentioned before, 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 to repeat for two different messages (even if the keys that are used to encrypt these messages are different). The latter requirement provides security against an even larger variety of attacks, as can be seen in Chapter 4.

### 3.2 Advanced Modes of Operation

Similarly to block ciphers, ABCs can be used to encrypt data streams longer than the block length. Following the differences between block ciphers and ABCs, the modes of operation for block ciphers need an adaptation to the ABC framework. In this section we discuss a family of modes of operation for ABCs.

**Definition 3.4** An advanced mode of operation (or shortly AMode) is a mode of operation in which the underlying cipher 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 key and salt inputs, and each call gets a different counter.

We usually use the counter as the index of the block but it should be emphasized that the counter can be the result of any invertible function on the block’s index.

**Notation 3.1** Let $X$ be a traditional mode of operation. We denote by $AX_{K,S}$ 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 3.1** $AECB_{K,S}$ is the AMode in which the $i$’th block satisfies: $C_i = E_{K,S,i}(M_i)$.

**Example 3.2** $ACBC_{K,S}$ is the AMode in which $C_0 = IV$ and $C_i = E_{K,S,i}(M_i) \oplus C_{i-1}$ for $i = 1, \ldots, m$, where $m$ is the number of blocks.

As we show in Chapter 4, the AECB mode is indistinguishable from a random permutation when the underlying ABC conforms to certain conditions. Therefore, it looks that whenever “good” block ciphers are used, 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 wish 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 (in order to prevent an adversary to control the inputs of the ABC), and for transforming the cipher into a MAC. The good security measures, provided by our framework, hold for all these advanced modes of operation, and each AMode has additional properties as exemplified above.

3.2.1 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. This family of modes tries to capture those modes of operation which are length preserving, and in which the encryption of a message block can depend only on the current block and the previous message blocks (i.e., modes that can be used for an online encryption of a message that is received block by block).

Definition 3.5 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, without the IV, if exists, 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 calls to the block cipher $E$ encryptions performed calculated while encrypting a message $M$ under $X_E$ is equal to the length of $M$ in blocks.

We associate every invocation of $E_{K,S}$ with a block number $i$, and we denote the plaintext input and the ciphertext output of the $i$'th invocation of $E_{K,S}$ by $(x_i, y_i)$, respectively. I.e., $y_i = E_{K,S,i}(x_i)$ In a reasonable Amode, there exist two functions $f_1(\cdot, \ldots, \cdot)$ and $f_2(\cdot, \ldots, \cdot)$.
such that $x_i = f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})$, and $C_i = f_2(y_i, IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})$, 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 $IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}$. This means that the $i$'th invocation of $E_K,S$ has an injective influence over the $i$'th ciphertext block, and is not, for example, being ignored.

(b) $C_i$ is a permutation of $M_i$, i.e., the encryption under $X^E_K$ is invertible and therefore the ciphertext can be decrypted. Formally, the function $h_{E,K,IV,i,M_1,...,M_{i-1},C_1,...,C_{i-1}}(M_i) = f_2(y_i, IV, i, M_1, ..., M_i, C_1, ..., C_{i-1})$ is a permutation over $\{0, 1\}^n$ for every possible combination of $K, IV, i, M_1, ..., M_{i-1}, C_1, ..., C_{i-1}$, and for every block cipher $E$. This means that the $i$'th plaintext block has an injective influence over the $i$'th ciphertext block.

(c) The calculation time of each of the functions $f_1(\cdot, ..., \cdot)$ and $f_2(\cdot, ..., \cdot)$ is independent in the block’s index $i$. The time complexity of the calculation of each of the functions is $O(1)$.

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 3.5 might seem complicated, but most of the widely used modes of operation, such as ECB, CBC, OFB, CFB, and CTR, fulfill these conditions and therefore we consider them to be reasonable modes.

**Example 3.3** 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 3.4** CBC is a reasonable mode where

$$f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) = \begin{cases} C_{i-1} \oplus M_i, & \text{for } i > 1 \\ IV \oplus M_i, & \text{for } i = 1 \end{cases},$$
and \( f_2(y_i, IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) = y_i \).

**Example 3.5** OFB is a reasonable mode where

\[
\begin{align*}
    f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) &= \begin{cases} 
        C_{i-1} \oplus M_{i-1}, & \text{for } i > 1 \\
        IV, & \text{for } i = 1 
    \end{cases}, \\
    f_2(y_i, IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) &= y_i \oplus M_i.
\end{align*}
\]

**Definition 3.6** An AMode \( X \) is a reasonable AMode if it satisfies all of the conditions of Definition 3.5, 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 3.1** 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, \( X^E_K(\cdot) \) is a length preserving encryption for every key \( K \). Therefore, \( AX^E_{K,S} \), by its definition, is also a length preserving encryption for every key \( K \) and a salt \( S \). The number of plaintext blocks in \( AX^E_{K,S} \) is equal to the number of plaintext blocks in \( X^E_K \) and the number of ciphertext blocks in \( AX^E_{K,S} \) is equal to the number of ciphertext blocks in \( X^E_K \). Therefore, the number of plaintext blocks in \( AX^E_{K,S} \) equals the number of ciphertext blocks. Moreover, all the plaintext and ciphertext blocks of \( AX^E_{K,S} \) are \( n \)-bit blocks.

2. Let \( f_1 \) and \( f_2 \) be the functions such that \( x_i = f_1(IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) \), and \( C_i = f_2(y_i, IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) \). By its definition, \( AX \) is the AMode in which \( C_i = f_2(\tilde{E}_{K,S,i}(x_i), IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) \).

(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 \( g_{E,K,IV,i,M_1,...,M_{i-1},C_1,...,C_{i-1}}(M_i) \)
\( = f_2(E_K(x_i), IV, i, M_1, ..., M_i, C_1, ..., C_{i-1}) \),
Let \( \tilde{E}^{(S,i)} \) be the block cipher that is defined by \( \tilde{E}^{(S,i)}_K(M) = \tilde{E}_{K,S,i}(M) \). Now we can write:
\[
\tilde{g}_{E,K,S,IV,i,M_1,...,M_{i-1},C_1,...,C_{i-1}}(M_i) = \tilde{g}_{E^{(S,i)},K,IV,i,M_1,...,M_{i-1},C_1,...,C_{i-1}}(M_i).
\]
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 \( AX^{E^{(S,i)}}_{K,S} \). Thus in \( AX \), \( C_i \) is a permutation of \( M_i \).

\[\blacksquare\]

### 3.3 Encryption Schemes

Now, that we have discussed block ciphers, modes of operation, ABCs and AModes, we can define three kind of symmetric encryption schemes. These definitions will allow us to later treat the different encryption schemes in a generalized way, and to easily compare them.

**Definition 3.7** A traditional symmetric encryption scheme is a pair of algorithms \( (E, D) \), that accept a key and a plaintext/ciphertext as parameters, and outputs a ciphertext/plaintext. Formally \( E \) and \( D \) are defined as:

\[
E : \mathcal{M} \times \mathcal{K} \to \mathcal{C},
\]
\[
D : \mathcal{C} \times \mathcal{K} \to \mathcal{M},
\]

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 \( D_K(E_K(M)) = M \).

The traditional symmetric encryption schemes we discuss in our work are the modes of operation reviewed in Chapter 2.

**Definition 3.8** A salted-countered symmetric encryption scheme is a pair of algorithms \( (E, D) \), that accept as parameters (in addition to the key and
the plaintext/ciphertext) also a counter and a salt, and outputs a ciphertext/plaintext. Formally $E$ and $D$ are defined as:

$$E : \mathcal{M} \times \mathcal{K} \times \mathcal{S} \times \mathcal{T} \to \mathcal{C},$$

$$D : \mathcal{C} \times \mathcal{K} \times \mathcal{S} \times \mathcal{T} \to \mathcal{M},$$

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 $D_{K,S,t}(E_{K,S,t}(M)) = M$.

The salted-countered symmetric encryption schemes we discuss in our work are ABCs.

**Definition 3.9** A salted symmetric encryption scheme is a pair of algorithms $(E, D)$, that accept as parameters (in addition to the key and the plaintext/ciphertext) also a salt but no counter, and outputs a ciphertext/plaintext. Formally $E$ and $D$ are defined as:

$$E : \mathcal{M} \times \mathcal{K} \times \mathcal{S} \to \mathcal{C},$$

$$D : \mathcal{C} \times \mathcal{K} \times \mathcal{S} \to \mathcal{M}.$$

We demand that for any $K \in \mathcal{K}, S \in \mathcal{S}$, and $M \in \mathcal{M}$ it holds that $D_{K,S}(E_{K,S}(M)) = M$.

The salted symmetric encryption schemes we discuss in our work are AModes.

For each of the definitions above, we note that $E$ (and similarly $D$) uniquely defines the symmetric encryption scheme. We will therefore use $E$ to describe the encryption scheme – it will be apparent from the context whether $E$ represents the encryption scheme or the encryption function itself.
Chapter 4

Security Analysis of ABCs

This chapter deals with the security of the ABC framework. We discuss the security of the framework against some well known practical attacks, and also discuss its security according to some widely used theoretical security measures.

4.1 Attacks that Take Advantage of a Repeating Permutation

Many of the attacks that are used for breaking ciphers and cryptographic protocols make use of a large amount of information encrypted using the same key, i.e., the same permutation. The desired amount of information might be collected since the same permutation is used over and over again when a mode of operation is in use. With ABCs and AModes this is not the case, since each permutation is used at most once. In the next sections we overview such generic attacks that make use of the repeating permutation.

It is important to emphasize that there are also attacks that are not based on the repetition of the same permutations, such as related-key attacks [5, 6, 24]. Thus, when designing a particular ABC, such attacks should also be considered.

4.1.1 Dictionary Attacks

In the dictionary attack, the adversary collects pairs of plaintext blocks and their respective ciphertexts. She gains information about the encryption per-
mutation 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.

4.1.2 Statistical Attacks

Statistical attacks, such as differential and linear cryptanalysis, as well as their many generalizations, usually require a large amount of data encrypted under the secret encryption permutation $E_K(\cdot)$. If the permutations defined by different combinations of key, salt and counter are, indeed, independent, then these attacks are inapplicable on ABCs, since a proper use ensures that an encryption permutation is never used more than once. Note that for realistic ABCs, as for block ciphers, the security analysis should consider statistical 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 Chapter 6, where we suggest some instances of ABCs, we make this kind of analysis.

4.1.3 Time-Memory Tradeoff Attacks

In time-memory tradeoff attacks [22], a large amount of pre-computation, equivalent to exhaustive search, can be used for breaking the block cipher many times in the future, where each instance is successfully broken faster than exhaustive search. As discussed in Chapter 3, the salt might be reused with different keys. If this is the case, and the salt is being reused in a predictable way (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 Theoretical Security Notions for Symmetric Key Encryption

Goldwasser and Micali [21] were the first to formally define security notions for encryption schemes. The encryption schemes that they considered were public key encryption schemes. Later, security notions for symmetric-key encryption were defined and examined by Bellare et. al, in [3].

The security notions we consider for our analysis are indistinguishability from random bits (discussed in Section 4.2.3) and semantic security (discussed in Section 4.2.4). 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 the encryption of messages and in the chosen-ciphertext model (CCA) where the adversary is allowed to ask for the decryption of ciphertexts.

4.2.1 Adversaries

Definition 4.1 Let $E$ be one of the following encryption schemes: a mode of operation, an ABC, or an AMode. An $E$-CPA adversary is an adversary that has access to oracles that answer queries of one of three forms:

- $(M)$ if $E$ is a mode of operation,
- $(S, M)$ if $E$ is an AMode,
- $(S, t, M)$ if $E$ is an ABC,

(Where $M \in M$, $S \in S$, and $t \in T$). The answer of an oracle for a query is a string of the same length as the length of the ciphertext $C$, which is the result of the encryption of $M$ under some key $K \in K$ (and using the salt $S$ and the counter $t$ where needed).

Definition 4.2 An $E$-CCA adversary is an adversary that has access to oracles that answer queries of one of three forms:

- $(C)$ if $E$ is a traditional symmetric-key encryption scheme,
- $(S, C)$ if $E$ is a salted symmetric-key encryption scheme,
- $(S, t, C)$ if $E$ is a salted-countered symmetric-key encryption scheme,
(where $C \in C$, $S \in S$, and $t \in T$). The answer of an oracle for a query is a string of the same length as the length of the plaintext $P$, which is the result of the decryption of $C$ under some key $K \in \mathcal{K}$ (and using the salt $S$ and the counter $t$ where needed).

As stated in Chapter 3, the salt is chosen either by the encrypting party or by the protocol, and the counter is selected by the AMode. In our analysis, we allow even more powerful adversaries who can choose the salt as long as the same salt never repeats. Similarly, if the adversary attacks an ABC rather than an AMode, we allow her to choose the salt and the counter that are used by the oracle, as long as the same combination of salt-counter never repeats.

**Definition 4.3** 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 4.4** Let $E$ be an ABC. Let $X$ be an AMode. An $XE$-CPA adversary is said to be salt-respecting when no two queries it calls have the same salt.

As mentioned before, the requirement that the same salt-counter combination never repeats is necessary for the security of our framework. When discussing a reasonable AMode, it is enough to require that any adversary should be salt-respecting — since in a reasonable AMode the counter value never repeats in the same message, it is guaranteed that when the adversary is salt-respecting the same salt-counter combination never repeats. When discussing an ABC it is necessary to require salt-counter respecting in order that the same salt-counter combination never repeats.

We note that we only demand that encryption queries do not use previously-used combinations of salt and counter. We do not demand this for decryption queries as the adversary can then control the salt. 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. For this reason the security of our framework against
CCA adversaries is not as good as its security against CPA adversaries (as discussed later in this chapter).

4.2.2 Distinguishers for Block Ciphers and for ABCs

A block cipher is a family of permutations $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$. Fixing a key $K \in \{0,1\}^k$ defines a single permutation. In a strong block cipher, when choosing the key randomly, the resulting permutation should “appear” like a random permutation. The PRP-security notion [4] described next, aims to measure the indistinguishability of a random block cipher permutation from a totally random permutation. In order to do so, we examine a “game” played by an adversary, in which she makes queries, and get in response the result of operating a permutation on those queries. She then has to determine, whether the permutation was chosen randomly from the set of permutations defined by the block cipher, or was chosen randomly from the set of all possible permutations. During the entire game, the adversary can make off-line calls to the block cipher.

Let $\text{Perm}(n)$ be the set of all possible permutations over $\{0,1\}^n$. Let $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher. Let $A$ be an adversary that has access to a permutation $\pi : \{0,1\}^n \rightarrow \{0,1\}^n$. The advantage of an adversary $A$ in distinguishing $E$ from a random permutation is defined as:

$$\text{Adv}_{prp}^E(A) \triangleq \Pr \left[ A^{E_K(\cdot)} = 1 \mid K \overset{R}{\leftarrow} \{0,1\}^k \right] - \Pr \left[ A^{\pi(\cdot)} = 1 \mid \pi \overset{R}{\leftarrow} \text{Perm}(n) \right].$$

We note that $A$ is allowed to make off-line calls to the block cipher $E$ which is under inspection. In these encryption/decryption calls, $A$ has to specify keys of her choice. We emphasize that making these queries to $E$ is the only way $A$ can calculate $E$ encryptions/decryptions with a key of her own choice. I.e., $A$ does not have the description of $E$. We denote by $A_{q_E,q}$ the set of all adversaries that make no more than $q_E$ off-line calls, and no more than $q$ queries to the oracle. We denote by $\text{Sec}_{prp}^{E}(q_E, q)$ the maximum over the advantages of adversaries in $A_{q_E,q}$. I.e.,

$$\text{Sec}_{prp}^{prp}(q_E, q) \triangleq \max_{A \in A_{q_E,q}} \text{Adv}_{prp}^E(A).$$

Definition 4.5 An ideal block cipher of $n$-bit block is a block cipher $E$ of whose key space contains $2^n!$ different keys, and such that for each permu-
utation $\pi \in \text{Perm}(n)$ there is a single key $k_\pi$ such that $E_{k_\pi}(\cdot) = \pi(\cdot)$.

Lemma 4.1 Let $E : \{0,1\}^k \times \{0,1\}^n \leftarrow \{0,1\}^n$ be an ideal block cipher. Then, for any $q$, and any $q_E$, it holds that

\[ \text{Sec}_{E}^{\text{prp}}(q_E, q) = 0. \]

Proof. The operation of choosing a random key uniformly from the key space of an ideal block cipher and then using the permutation defined by that key is equivalent to the operation of choosing a random permutation uniformly from $\text{Perm}(n)$.

An advanced block cipher $E : \{0,1\}^n \times \{0,1\}^k \times \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n$ is a family of $2^k$ mappings from $\{0,1\}^s \times \{0,1\}^c$ to $\text{Perm}(n)$. By fixing a key $K \in \{0,1\}^k$, an ABC defines a single mapping of the form: $E_K : \{0,1\}^s \times \{0,1\}^c \rightarrow \text{Perm}(n)$. We adapt the above definitions of the advantage of a block-cipher distinguisher to the world of ABCs.

Let $\text{SCP}erm(s,c,n)$ be the set of all possible mappings from $\{0,1\}^s \times \{0,1\}^c$ to $\text{Perm}(n)$.

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 mapping from $\{0,1\}^s \times \{0,1\}^c$ to $\text{Perm}(n)$. The advantage of an adversary $A$ in distinguishing $E$ from a random family of such mappings is defined as:

\[
\text{Adv}_{E}^{\text{prp}}(A) \triangleq \text{Pr}[A^{E_K(\cdot,\cdot)} = 1 | K \leftarrow \{0,1\}^k] - \text{Pr}[A^{f(\cdot,\cdot)} = 1 | f \leftarrow \text{SCP}erm(s,c,n)].
\]

We note that $A$ is allowed to make off-line calls to the ABC $E$ which is under inspection. In these encryption/decryption calls, $A$ has to specify keys of her choice, and is allowed to choose the salt and the counter of every call as her will. Again, this is the only way $A$ can compute $E$ encryptions/decryptions with a key of her choice. We denote by $A_{qE,q}^{\text{scr}}$ the set of all salt-counter-respecting adversaries that make no more than $q_E$ off-line calls to $E$ and no more than $q$ queries. We denote by $\text{Sec}_{E}^{\text{prp}}(qE,q)$ the maximum over the advantages of adversaries in $A_{qE,q}^{\text{scr}}$. I.e.,

\[
\text{Sec}_{E}^{\text{prp}}(qE,q) \triangleq \max_{A \in A_{qE,q}^{\text{scr}}} \text{Adv}_{E}^{\text{prp}}(A).
\]
**Definition 4.6**  An ideal ABC of $n$-bit block, $s$-bit salt, and $c$-bit counter is an ABC $\tilde{E}$ whose key space contains $(2^n!)^{2^s+c}$ different keys, and such that for each mapping $f \in \text{SCP}_{\text{Perm}}(s,c,n)$ there is a single key $k_f$ such that $\tilde{E}_{k_f}(\cdot,\cdot,\cdot) = f(\cdot,\cdot,\cdot)$.

**Lemma 4.2** Let $E$ be an ideal ABC. For any number of queries $q$ that allows the adversary to be salt-counter-respecting, and any $q_E$, 

$$\text{Sec}_{E}^{\text{prp-CPA}}(q_E, q) = 0.$$  

**Proof.** The operation of choosing a random key uniformly from the key space of an ideal block cipher and then using the mapping defined by that key is equivalent to the operation of choosing a random mapping uniformly from $\text{SCP}_{\text{Perm}}(s,c,n)$.

Later in this chapter we show that if $E$ is an ABC for which $\text{Sec}_{E}^{\text{prp-CPA}}(q_E, q)$ is small, then using a reasonable AMode in which the underlying ABC is $E$, results with a secure encryption scheme.

### 4.2.3 Indistinguishability

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

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

The ind-CPA advantage of an $E$-CPA adversary $A$ is defined as: 

$$\text{Adv}_{E}^{\text{ind-CPA}}(A) \equiv \Pr[A^{O_{E_K}(\cdot)} = 1 | K \xleftarrow{\text{R}} \mathcal{K}] - \Pr[A^{O_{E_R}^F(\cdot)} = 1].$$

For the CCA variant of this notion, consider two other oracles, both answer queries of the form $Q = (C)$ with a string $M \in \{0,1\}^{D_K(C)}$. The

\(^1\)According to the original definition in [3], the random oracle encrypts a random message of $|M|$ bits. For any reasonable mode or reasonable AMode this is equivalent to returning a random string of $|E_K(M)|$ bits.
first is the real decryption oracle, $O_{DK}$ that answers the query $Q = (C)$ with $M = DK(C)$, for a randomly chosen key $K$. The second is a fake decryption oracle, $O_{DR}$ that answers the same query $Q$ with a random string of $|DK(C)|$ bits.

The ind-CCA advantage of an adversary $A$ against an encryption scheme $E$ is defined as:

$$\text{Adv}_E^{\text{ind-CCA}}(A) \triangleq \Pr[A^{O_{DK}(\cdot) = 1} | K \leftarrow R] - \Pr[A^{O_{DR}(\cdot) = 1} | K \leftarrow R].$$

We adapt these security notions also to the salted-symmetric-key encryption scheme, in which the queries are of the form $Q = (S, M)$ -- in the CPA case, or $Q = (S, C)$ -- in the CCA case. Thus, if $E$ is a salted-symmetric-key encryption scheme, then

$$\text{Adv}_E^{\text{ind-CPA}}(A) \triangleq \Pr[A^{O_{EK}(\cdot, \cdot) = 1} | K \leftarrow R] - \Pr[A^{O_{ER}(\cdot, \cdot) = 1} | K \leftarrow R],$$

$$\text{Adv}_E^{\text{ind-CCA}}(A) \triangleq \Pr[A^{O_{DK}(\cdot, \cdot) = 1} | K \leftarrow R] - \Pr[A^{O_{DR}(\cdot, \cdot) = 1} | K \leftarrow R].$$

It is important to note that the adversary $A$ must not repeat the same query twice (or otherwise it is 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.

Also, the adversary is allowed to make her own off-line calls to the encryption scheme $E$. When doing so, she chooses all the parameters (including the secret key) of $E$. When discussing a mode of operation or an AMode, the adversary is also allowed to make her own off-line calls to the underlying block cipher or ABC (rather than just to the mode of operation or the AMode). Again, this is the only way $A$ can compute $E$ encryptions/decryptions with a key of her choice -- she does not have the description of the block cipher or the ABC.

We use the abbreviated notations of $O_E$ for the oracle $O_{E_K}$ or $O_{DK}$ with a random key $K$, and $O_R$ for $O_{EK}$ or $O_{DR}$, where it is clear from the context whether the attack is a chosen-plaintext attack or a chosen-ciphertext attack.

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

$$\text{Sec}_{E}^{\text{ind}-\text{CPA}}(\sigma_E, \sigma) \triangleq \max_{A \in A_{\sigma_E, \sigma}^{sr}} \text{Adv}_{E}^{\text{ind}-\text{CPA}}(A),$$

$$\text{Sec}_{E}^{\text{ind}-\text{CCA}}(\sigma_E, \sigma) \triangleq \max_{A \in A_{\sigma_E, \sigma}} \text{Adv}_{E}^{\text{ind}-\text{CCA}}(A).$$

Indistinguishability of Traditional Modes of Operation

In this section, we calculate bounds for the CPA indistinguishability of some of the commonly used modes of operation, recognized by NIST.

ECB — The ECB mode is obviously an insecure mode of operation. This also reflects in its poor indistinguishability security. The best distinguisher for ECB is the one that asks for the encryption of a $\sigma$-block message of $\sigma$ equal blocks. The adversary outputs ‘1’ if and only if all the ciphertext blocks are equal. The advantage of such an adversary is $1 - 2^{-n(\sigma-1)}$. For the CCA case, a similar CCA-adversary achieves the same result.

CBC — For large enough values of $\sigma$, a possible CPA-adversary $A$ against CBC can ask for the encryption of a $\sigma$-block message consists of random blocks. For $\sigma \geq 2^{n/2+1}$, and assuming that the ciphertext blocks are pseudo-random, we get by the birthday paradox that with probability of at least 1/2 there are some $i \neq j$ for which $C_i = C_j$. If indeed there are such blocks, and if the answer is the result of a real CBC encryption then necessarily (as explained in Chapter 2) $M_i \oplus M_j = C_{i-1} \oplus C_{j-1}$. If the answer is a random string then the last equation holds with probability of $2^{-n}$. Thus, if $A$ outputs ‘1’ when there are some two equal ciphertext blocks $C_i = C_j$ and $M_i \oplus M_j = C_{i-1} \oplus C_{j-1}$, its advantage is lower bounded by $\text{Adv}_{CBC}^{\text{ind}-\text{CPA}}(A) > (1 - e^{-\frac{\sigma(\sigma-1)}{2^{n+1}}})(1 - 2^{-n})$. For large enough values
of $\sigma$ this bound is non-negligible. For example, for $\sigma = 2^{n/2+1}$ we get that $Adv_{C_{CBC}^{ind-CPA}}(A) > (1 - e^{-1})(1 - 2^{-n}) \approx 0.63$. This advantage can be used to lower bound $Sec_{C_{CBC}^{ind-CPA}}(q_E, q)$.

For the CCA case we have an even better distinguisher. Consider the adversary $A$ that asks for the decryption of $C = IV, \alpha || \alpha$, for some $IV, \alpha \in \{0, 1\}^n$. $A$ outputs ‘1’ if and only if $M_2 = M_1 \oplus \alpha \oplus IV$. The advantage$^2$ of $A$ is $Adv_{C_{CBC}^{ind-CCA}}(A) = 1 - 2^{-n}$.

**OFB** — The stream produced by OFB is cyclic, and the probability that the first $\sigma$ blocks of the OFB stream form a cycle (i.e., the cycle-length is no longer than $\sigma$), equals to $\min \{1, \sigma \cdot 2^{-n}\}$. If the cycle-length is longer than $\sigma$, (i.e, the first $\sigma$ blocks do not form a cycle) then there are no collisions within the first $\sigma$ blocks.

Let $A$ be an adversary that asks for the encryption of a $\sigma$-block message $M$. When receiving the answer $C$, $A$ calculates the stream $V = M \oplus C$. $A$ outputs ‘1’ if $V$ can possibly be produced by OFB mode of operation with some block cipher $E$. I.e., $A$ outputs ‘1’ if and only if there are no collisions in the stream or the stream is cyclic. The probability of a cycle being generated by the random oracle can be upper bounded by:

$$\Pr[\text{cycle}|O_R] \leq \frac{2^{-n}}{2^n - 1}.$$  

And therefore, the advantage of $A$ can be lower bounded by:

$$Adv_{C_{ CBC}^{ind-CPA}}(A) = 1 - \Pr[\text{no collisions}|O_R] - \Pr[\text{cycle}|O_R] \geq 1 - e^{-\sigma(\sigma+1)} - \frac{2^{-n}}{2^n - 1}$$

And thus we can lower bound $Sec_{C_{OFBE}^{ind-CPA}}(0, \sigma)$. For example, $Sec_{C_{OFBE}^{ind-CPA}}(0, 2^{n/2+1}) > 1 - e^{-1} - \frac{2^{-n}}{2^n - 1} \approx 0.63$

**CTR** — In a CTR stream that is XORed into the plaintext/ciphertext there are no two equal blocks. Thus, a possible adversary $A$ can ask for the

---

$^2$The advantage of this attack can be enlarged to $1 - 2^{-n(\sigma - 1)}$ by asking for the decryption of $C = IV, \alpha || \ldots || \alpha$, and making sure that $M_i = M_1 \oplus \alpha \oplus IV$ for all $1 < i \leq \sigma$.
encryption of a $\sigma$-block message $M$. When receiving the answer $C$, $A$ calculates the stream $V = M \oplus C$. $A$ outputs ‘1’ if there are no colliding blocks in $V$. Otherwise, it outputs ‘0’. The advantage of $A$ can be lower bounded by $\text{Adv}^{\text{ind-CPA}}_{\text{CTR}}(A) \geq 1 - e^{-\frac{\pi(\sigma-1)}{2^n}}$. For example, for $\sigma = 2^{n/2+1}$, the advantage of $A$ can be bounded by:

$$\text{Adv}^{\text{ind-CPA}}_{\text{CTR}}(A) \geq 1 - e^{-\frac{2^n}{2^{n/2+1}}} \geq \left(1 - e^{-1}\right)(1 - 2^{-n}).$$

This can be used to lower bound $\text{Sec}^{\text{ind-CPA}}_{\text{CTR}}(qE; q)$.

**Indistinguishability of Tweakable Block Ciphers**

In [26], 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 [26]. In TBC, the $i$’th block of the ciphertext is used as the tweak for the $i + 1$ block. The TBC mode is illustrated in Figure 4.1. The TBC mode of operation is an example for a tweakable encryption scheme which is less secure than AModes. The indistinguishability security of reasonable AModes is better than the indistinguishability security of TBC in some cases, and not worse in the others.

Let $\tilde{E}$ be an ideal tweakable block cipher. Denote $C_0 \triangleq T_0$. A possible ind-CPA adversary $A$, 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. The higher the value of $m$, the higher the probability that there are two colliding blocks $C_i = C_j$ (for $0 \leq i < j < m$). If the output is the result of a real TBC encryption, rather than a random string, then for every $1 \leq \ell \leq m - j$ it holds that $C_{i+\ell} = C_{j+\ell}$. If this is the case then the adversary can deduce that she is facing the real encryption oracle. Otherwise, she is definitely facing the fake oracle. The probability for a cycle that ends before the last cipher block, when $A$ is facing the real oracle is equal to the probability of having two colliding blocks $C_i = C_j$ (for $0 \leq i < j < m$). (i.e, higher
Figure 4.1: TBC mode of operation. Note that $C_i$ is used as the tweak of the $i + 1$ block cipher call.

than $1 - e^{-\frac{\sigma(\sigma-1)}{2^{n+1}}}$). The probability for a cycle that ends before the last cipher block, when $A$ is facing the random oracle is not higher than the probability for getting a stream in which there is a collision $C_i = C_j$ (where $1 \leq i < j < m$) and in which $C_{i+1} = C_{j+1}$.

Thus, the advantage of an adversary $A$ that asks for the encryption of a message $M = M_1 || \ldots || M_m$, where all the message blocks are equal, and outputs 1 only when a cycle that ends before the last block exists in the ciphertext, can be lower bounded by $Adv_{TBC}^{ind-CPA}(A) > (1 - e^{-\frac{\sigma(\sigma-1)}{2^{n+1}}})(1 - 2^{-n})$. As we later show (Corollary 4.1), $Sec_{XE}^{ind-CPA}(\sigma, \sigma) = 0$ for any reasonable AMode $X$, and any ideal ABC $E$, and thus, for every $\sigma \geq 2^{n/2+1}$ it holds that $Sec_{TBC}^{ind-CPA}(\sigma, \sigma) > Sec_{XE}^{ind-CPA}(\sigma, \sigma)$.

Additionally, an ind-CCA adversary $\tilde{A}$ that attacks $TBC^{E}$ 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 $Adv_{TBC}^{ind-CCA}(\tilde{A}) = 1 - 2^{-n}$. When an ind-CCA adversary $A$ that attacks $XE$ is limited to a total of two blocks in its queries, there are two possible cases:

1. The two block are decrypted with a different salt-counter combination. In this case the advantage is 0.

2. The two blocks are decrypted using the same salt-counter combina-
tion, which means that the blocks are not equal. In such a case, the advantage is not higher than $2^{-n}$. And thus, $\text{Sec}_{\text{TAE}}^{\text{ind-CPA}}(0, 2) > \text{Sec}_{X}^{\text{ind-CPA}}(0, 2)$.

**Indistinguishability of AModes**

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

**Lemma 4.3** Let $E$ be an ideal ABC. For every possible total length of queries, $\sigma$ that allows the adversary to be salt-respecting (i.e., $\sigma \leq 2^{c+s}$), and for every $\sigma_E$ it holds that $\text{Sec}_{\text{AECB}}^{\text{ind-CPA}}(\sigma_E, \sigma) = 0$.

**Lemma 4.4** Let $E$ be an ideal ABC. For every possible total length of queries $\sigma$ that allows the adversary to be salt-respecting (i.e., $\sigma \leq 2^{c+s}$), and for every $\sigma_E$ it holds that $\text{Sec}_{\text{ACBC}}^{\text{ind-CPA}}(\sigma_E, \sigma) = 0$.

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

**Theorem 4.1** Let $X$ be a reasonable AMode. Let $E$ be an ABC. For every possible total length of queries $\sigma$ that allows the adversary to be salt-respecting, and for every $\sigma_E$ it holds that $\text{Sec}_{X}^{\text{ind-CPA}}(\sigma_E, \sigma) \leq \text{Sec}_{E}^{\text{prp}}(\sigma_E, \sigma)$.

**Proof.** In [26], the authors prove the security of the TAE mode. Our proof for the security of reasonable AModes is similar. Let $A$ be a salt-respecting ind-CPA adversary against $X^E$ that uses $q$ queries of total length of $\sigma$ blocks, $\sigma_E$ off-line calls to the underlying ABC, and that achieves the maximal advantage such an adversary can achieve, i.e., $\text{Adv}_{X}^{\text{ind-CPA}}(A) = \text{Sec}_{X}^{\text{ind-CPA}}(\sigma_E, \sigma)$. We build an adversary $B$ against $E$ that uses $\sigma$ queries, $\sigma_E$ off-line calls to $E$, and such that $\text{Adv}_{E}^{\text{prp}}(B) = \text{Adv}_{X}^{\text{ind-CPA}}(A)$.

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 $Q = (S, M = M_1||\ldots||M_m)$ to $O$, $B$ chooses an IV (if needed) in the manner defined by $X$, 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 \( C = IV, C_1||...||C_m \). \( B \) lets \( A \) run on \( O \). Whenever \( A \) makes an off-line call to \( E \), \( B \) makes the same call. When \( A \) finishes its run and answers, \( B \) answers as \( A \).

First, we note that if \( A \) uses a total of \( \sigma \) blocks for its queries then \( B \) makes \( \sigma \) unique queries to \( O_B \). This is due to the fact that \( A \) is salt-respecting. Also, obviously, \( B \) makes exactly \( \sigma_E \) off-line calls to \( E \). We also note that when \( O_B = E_K(\cdot, \cdot, \cdot) \) then \( O = O_X^E(\cdot, \cdot, \cdot) \). This follows immediately from the building of \( O \). At last, we claim that when \( O_B = \pi(\cdot, \cdot, \cdot) \) then \( O = O_X^E(\cdot, \cdot, \cdot) \). 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 and independently at random from \( \{0, 1\}^n \) and thus, \( C \) is a random string of length \( |X^E_{K,S}(M)| \).

Now, we can write:

\[
\text{Adv}_{\text{prp}}^E(B) = \Pr \left[ \begin{array}{c}
B^{E_K(\cdot, \cdot, \cdot)} = 1 \\
B^{\pi(\cdot, \cdot, \cdot)} = 1
\end{array} \right] \left| \begin{array}{c}
K \leftarrow \{0, 1\}^k \\
\pi \leftarrow \text{SCP}(s, c, n)
\end{array} \right] = \Pr \left[ \begin{array}{c}
A^{O_X^E(\cdot, \cdot, \cdot)} = 1 \\
A^{\pi(\cdot, \cdot)} = 1
\end{array} \right] - \Pr \left[ \begin{array}{c}
A^{\pi(\cdot, \cdot)} = 1
\end{array} \right] = \text{Adv}_{\text{ind-CPA}}^X(A).
\]

Following Theorem 4.1 and Lemma 4.2, we conclude the following corollary:

**Corollary 4.1** If \( X \) is a reasonable AMode, and \( E \) is an ideal ABC, then \( \text{Sec}_{\text{ind-CPA}}^{X^E}(\sigma_E, \sigma) = 0 \) for every \( \sigma \) that allows the adversary to be salt-respecting, and every \( \sigma_E \).

As seen in Corollary 4.1, 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.
Consider the CCA adversary that asks for the decryption of \( C = C_1 || C_2 \) and \( C' = C_1 \) (in case IV is used by the mode of operation, the same IV is used with both ciphertexts), and returns ‘1’ if and only if the corresponding messages, \( M, M' \) begin with the same block. For any reasonable mode (for regular block ciphers), this adversary has an overwhelming success rate (of \( 1 - 2^{-n} \)). It is easy to see that this adversary can be easily adapted to foil any reasonable AMode, by asking for the decryption of \( C \) and \( C' \) under the same salt (and the same IV, if used by the AMode), resulting in the same attack with the same advantage.

Hence, 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. Although we do not claim that AECB is ind-CCA secure, we claim that it is not worse than ECB in this respect.

To present another advantage of the ABC framework over standard block ciphers, we consider the case of ECB and AECB. For ECB, the trivial CPA adversary shown earlier, achieves advantage of \( 1 - 2^{-n} \) after one query with two blocks (namely, asking for the decryption of \( C = C_1 || C_1 \) and returning ‘1’ if and only if the two blocks of the received plaintext are equal). For AECB (and for that matter, any reasonable AMode), such an attack fails. Moreover, it is easy to show that most CCA adversaries against AModes with two blocks of message, has advantage of at most \( 2^{-n} \). This advantage can be achieved if the adversary uses two different single-block ciphertexts, with the same salt (and the same IV, if used by the AMode), and outputs ‘1’ if and only if the two plaintexts received as answers are different one from the other\(^3\).

When discussing \( \sigma = 3 \) the advantage of the AECB adversary is no longer negligible. Still, the security of AECB is better than that of ECB. While for three ciphertext blocks the advantage of the generic adversary is \( 1 - 2^{-n} \), the CCA adversary against ECB that asks for the decryption of a single ciphertext \( C = C_1 || C_1 || C_1 \)) and returns ‘1’ if and only if the received plaintext is composed of three identical blocks, has advantage \( 1 - 2^{-2n} \).

\(^3\)The best advantage against AOFB and ACTR that can be achieved by using two single-block messages is also \( 1 - 2^{-n} \).
Theorem 4.2 If $E$ is an ideal block cipher, and $	ilde{E}$ is an ideal ABC then for every $\sigma$ and $\sigma_E$ it holds that

$$\text{Sec}_{\text{ECB}}^{\text{ind-CCA}}(\sigma_E; \sigma) \geq \text{Sec}_{\text{AECB}}^{\text{ind-CCA}}(\sigma_E; \sigma).$$

In order to prove Theorem 4.2 we use the following two lemmas.

Lemma 4.5 Let $E$ be an ideal ABC. Let $A$ be an ind-CCA adversary against $\text{AECB}_E$ that uses no more than a total of $\sigma$ blocks in its queries, and no more than $\sigma_E$ off-line calls to the underlying ABC. Then there exists a deterministic ind-CCA adversary $A'$ that uses no more than a total of $\sigma$ blocks in its queries and no more than $\sigma_E$ off-line calls to the ABC, and such that $\text{Adv}^{\text{ind-CCA}}_{\text{AECB}_E}(A) \leq \text{Adv}^{\text{ind-CCA}}_{\text{AECB}_E}(A').$

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 $u_1, \ldots, u_m$ 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

$$\text{Adv}^{\text{ind-CCA}}_{\text{AECB}_E}(A) = \Pr[A^{\text{AECB}_E(\cdot)} = 1|K \overset{R}{\leftarrow} \{0, 1\}^k] - \Pr[A^{O_R(\cdot)} = 1]$$

$$= \Pr[A^{O(\cdot)} = 1|O = O_E] - \Pr[A^{O(\cdot)} = 1|O = O_R]$$

$$= \Pr[A^{O(\cdot)} = 1|O = O_E \land A_v] \cdot \Pr[A_v|O_E]$$

$$+ \Pr[A^{O(\cdot)} = 1|O = O_E \land \neg A_v] \cdot \Pr[\neg A_v|O_E]$$

$$- \Pr[A^{O(\cdot)} = 1|O = O_R \land A_v] \cdot \Pr[A_v|O_R]$$

$$- \Pr[A^{O(\cdot)} = 1|O = O_R \land \neg A_v] \cdot \Pr[\neg A_v|O_R].$$

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

Let

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

Now, consider the adversary $A^{(v)}$ whose 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 immediately to \( u_j \).

For any state \( w \), we denote by \( O_E^w \) 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 \( O_E^{-w} \) the event in which \( O = O_E \) and the adversary does not reach state \( w \). Similarly, we denote by \( O_R^w \) and \( O_R^{-w} \) the events in which \( O = O_R \) and the adversary reaches or not, respectively, state \( w \).

Obviously, \( A^{(v)} \) performs one flip-coin less than \( A \) and satisfies:

\[
\text{Adv}_{AECB}^{\text{ind-CCA}} (A^{(v)}) = Pr[A^{(v)}_{O(\cdot)} = 1 \mid O = O_E] - Pr[A^{(v)}_{O(\cdot)} = 1 \mid O = O_R] = Pr[A^{(v)}_{O(\cdot)} = 1 \mid O = O_E] - Pr[A^{(v)}_{O(\cdot)} = 1 \mid O = O_R]
\]

\[
= \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_E^w \right] \cdot \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_R^{-w} \right] - \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_R^w \right] \cdot \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_E^{-w} \right]
\]

\[
= \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_E^w \right] \cdot \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_R^{-w} \right] - \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_R^w \right] \cdot \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_E^{-w} \right]
\]

\[
+ \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_R^{-w} \right] \cdot \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_E^w \right] - \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_R^{-w} \right] \cdot \Pr \left[ A^{(v)}_{O(\cdot)} = 1 \mid O_E^w \right]
\]

\[
\geq \sum_{i=1}^{n} \Pr [u_i] \cdot \Pr [A_i \mid O_E] \cdot \Pr [A^{(v)}_{O(\cdot)} = 1 \mid O_R^w] \cdot \Pr [A^{(v)}_{O(\cdot)} = 1 \mid O_E^{-w}] - \Pr [A^{(v)}_{O(\cdot)} = 1 \mid O_R^{-w}] \cdot \Pr [A^{(v)}_{O(\cdot)} = 1 \mid O_E^w]
\]

\[
+ \sum_{i=1}^{n} \Pr [u_i] \cdot \Pr [A_i \mid O_E] \cdot \Pr [A^{(v)}_{O(\cdot)} = 1 \mid O_R^w] \cdot \Pr [A^{(v)}_{O(\cdot)} = 1 \mid O_E^{-w}] - \Pr [A^{(v)}_{O(\cdot)} = 1 \mid O_R^{-w}] \cdot \Pr [A^{(v)}_{O(\cdot)} = 1 \mid O_E^w]
\]

\[
= \text{Adv}_{AECB}^{\text{ind-CCA}} (A)^{\prime} \cdot \Pr [\text{Adv}_{AECB}^{\text{ind-CCA}} (A)^{\prime}] \geq \text{Adv}_{AECB}^{\text{ind-CCA}} (A).
\]

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

**Definition 4.7** We denote the sequence of queries made by an adversary \( A \) to oracle \( O \), and to the actual block cipher/ABC, and the answers received for them 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, and since the block cipher or the ABC are deterministic as well, 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, $\text{Adv}_{X E}^{\text{ind-CCA}}(A) = \sum_{u \in U} (\Pr[u|O = O_E] - \Pr[u|O = O_R])$.

**Lemma 4.6** 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 $\sigma$ blocks in its queries, no more than $\sigma_E$ off-line calls to the underlying ABC, and such that $\text{Adv}_{X E}^{\text{ind-CCA}}(A) \geq 0$. If there are $1 \leq \ell \leq \sigma$ streams that $A$ accepts (outputs ‘1’) then there exists a deterministic adversary $B$ that uses no more than a total of $\sigma$ blocks in its queries, no more than $\sigma_E$ off-line calls to the underlying ABC, accepts $2^n \leq \ell$ streams and such that $\text{Adv}_{X E}^{\text{ind-CCA}}(B) \geq \text{Adv}_{X E}^{\text{ind-CCA}}(A)$.

**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 $\pi$ be a permutation over $\{0,1\}^n$. Consider the stream $u_\pi$ which is identical to $u$ except that every answer block $c$ to a query block with $S = S_1, t = 1$ in $u$ is replaced with $\pi(c)$ in $u_\pi$. 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, $\Pr[u_\pi|O = O_R] = \Pr[u|O = O_R] = 2^{-|u| - n}$, 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

$$\text{Adv}_{X E}^{\text{ind-CCA}}(B) = \sum_{u \in U_B} (\Pr[u|O = O_E] - \Pr[u|O = O_R])$$

$$= \sum_{u \in U_A} (\Pr[u|O = O_E] - \Pr[u|O = O_R])$$

$$+ \sum_{u \in U_B \setminus U_A} (\Pr[u|O = O_E] - \Pr[u|O = O_R]) \geq 0$$

$$\geq \text{Adv}_{X E}^{\text{ind-CCA}}(A)$$
Now we can prove Theorem 4.2:

**Proof.** Let $A$ be an ind-CCA adversary against $\widetilde{AECB}^E$ that uses no more than a total of $\sigma$ blocks in its queries, and no more than $\sigma_E$ offline calls to the underlying ABC. Without loss of generality, $A$ is deterministic (following Lemma 4.5). In case $A$ does not accept any streams, then $\text{Adv}_{\widetilde{AECB}^E}^{\text{ind-CCA}}(A) = 0$. In case that there are streams that $A$ accepts then, without loss of generality, there are at least $2^n$ such streams (following Lemma 4.6).

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, $\Pr[A^{O_R(\cdot)} = 1] \geq 2^{-n(\sigma-1)}$. Therefore, we can write

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

and conclude that $\text{Sec}_{\widetilde{AECB}^E}^{\text{ind-CCA}}(\sigma_E, \sigma) \leq 1 - 2^{-n(\sigma-1)}$.

Now, consider an ind-CCA adversary $B$ that works against $ECB^E$ and is defined as follows: $B$ asks a single query of $\sigma$ blocks, all equal and accepts (outputs ‘1’) if and only if all $\sigma$ 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

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

Table 4.1 summarizes the CPA-indistinguishability security of the modes of operation and AModes we discuss in this section.

### 4.2.4 Semantic Security

The semantic security notion was first defined in [21] and was adapted to symmetric-key encryption schemes in [3]. The semantic security notion evaluates the ability of an adversary to learn something on a plaintext from its corresponding ciphertext. Let $(E, D)$ be a traditional symmetric-key encryption scheme. Let $A$ be an adversary. For the CPA variant of the notion,
Table 4.1: The indistinguishability security of different modes of operation and AModes.

<table>
<thead>
<tr>
<th>Mode</th>
<th>ind-CPA Security</th>
<th>Minimal $\sigma$ for which $S_{E_{\text{ind-CPA}}}^{\text{ind-CPA}}(0, \sigma) \geq 0.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECB</td>
<td>$1 - 2^{-n(\sigma-1)}$</td>
<td>2</td>
</tr>
<tr>
<td>CBC</td>
<td>$(1 - e^{-2^{(\sigma+1)}})(1 - 2^{-n})$</td>
<td>$1.17 \cdot 2^{n/2}$ (approx.)</td>
</tr>
<tr>
<td>OFB</td>
<td>$1 - e^{-2^{(\sigma+1)}} - 2^{-n}$</td>
<td>$1.17 \cdot 2^{n/2}$ (approx.)</td>
</tr>
<tr>
<td>CTR</td>
<td>$\geq 1 - e^{-2^{(\sigma+1)}}$</td>
<td>$1.17 \cdot 2^{n/2}$ (approx.)</td>
</tr>
<tr>
<td>TBC</td>
<td>$(1 - e^{-2^{(\sigma+1)}})(1 - 2^{-n})$</td>
<td>$1.17 \cdot 2^{n/2}$ (approx.)</td>
</tr>
<tr>
<td>AECB</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>ACBC</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>AOFB</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>ACTR</td>
<td>0</td>
<td>N/A</td>
</tr>
</tbody>
</table>

we consider an $\mathcal{E}$-CPA adversary, and for the CCA variant we consider an $\mathcal{E}$-CCA adversary.

The adversary plays a game against the encryption (decryption) oracle. The sequence of the game is as follows:

1. The adversary makes queries, and receives the answers for them from the oracle.
2. The adversary defines a distribution function $\gamma$ over the message space, such that all the messages with non-zero probability are of the same length.
3. The adversary is then given the challenge $C^*$ such that $C^* = \mathcal{E}_K(M^*)$ for some plaintext $M^*$ chosen at random from the message space according to $\gamma$.
4. The adversary makes more queries to the oracle.
5. The adversary outputs a pair $(\alpha, f)$, where $f$ is a function, defined for all non-zero-probability messages, that can be computed by the adversary. $A$ wins if $\alpha = f(M^*)$. 

41
Figure 4.2: Semantic security. The adversary wins if $\alpha = f(M^*)$

Figure 4.2 describes the flow of messages between the adversary and the oracle.

During the entire game, $A$ is allowed to make off-line calls to the encryption scheme $\mathcal{E}$. In these calls she chooses all the parameters of $\mathcal{E}$ (including the secret key) according to her will. When discussing a mode of operation or an AMode, $A$ is also allowed to make off-line calls to the underlying block cipher or ABC (rather than just to the mode of operation or the AMode).

Let $M'$ be a message chosen at random from the message space, according to $\gamma$, the distribution function defined by $A$. The advantage of an adversary $A$ is defined as:

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

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

We denote by $\text{Sec}_{\mathcal{E}}^{\text{sem-ATK}}(\sigma, \sigma)$ the maximum advantage taken over all $\mathcal{E}$-ATK adversaries that use no more than a total of $\sigma$ blocks in their queries,
and no more than $\sigma_E$ off-line calls to the encryption scheme. Formally, 

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

We adapt this security notion to our framework. Here, in the CPA scenario the adversary makes queries of the form (M,S), and in the CCA scenario the adversary makes queries of the form (C,S). The challenge given to the adversary $A$ after the first stage is a pair ($C^*, S^*$), such that $S^*$ is a salt that $A$ has not used so far in its queries, and such that $C^* = E_K, S^*(M^*)$ for some plaintext $M^*$ chosen at random from the message space according to $\gamma$. We demand, of course, that the CPA-adversary is a salt-respecting adversary. This means not only that the adversary is not allowed to use the same salt in two different queries, but also that after being challenged, the adversary is not allowed to use $S^*$ in its queries. We do not require CCA adversaries to be salt-respecting.

**Semantic Security of Traditional Modes**

In [3], the authors show reductions among some security notions. From these reductions it can be deduced that 

$$\text{Sec}_{E}^{\text{sem-ATK}}(\sigma_E, \sigma, q) \geq \frac{1}{q} \text{Sec}_{E}^{\text{ind-ATK}}(\sigma_E, \sigma, q),$$

where $q$ is the maximal number of queries that the adversaries are allowed to make (rather than their total length in blocks, given by $\sigma$ and $\sigma_E$). While we do not limit the number of queries in our model, all the examples for adversaries given in Section 4.2.3 use a single query. Therefore, all the security bounds for indistinguishability that are given in Section 4.2.3, apply also to semantic security.

**Semantic Security of AModes**

**Theorem 4.3** Let $X$ be a reasonable AMode, and let $E$ be an ideal ABC. Then, 

$$\text{Sec}_{X^E}^{\text{sem-CPA}}(\sigma_E, \sigma) = 0,$$

In order to prove Theorem 4.3 we use the following two lemmas.

**Lemma 4.7** Let $X$ be a reasonable AMode, let $E$ be an ideal ABC, and let $A$ be a sem-CPA adversary against $X^E$. Let $u$ be a stream generated by the run of $A$, and let $M^*$ be the message chosen randomly by the oracle (according
to $\gamma$) in order to generate the challenge $C^*$. Then, for every message $C$, such that $|C| = |M^*|$, it holds that $\Pr[C^* = C|M^* \land u] = 2^{-|M^*|}$, when the randomness is over the choice of the key $K$.

**Proof.** Let $K_u$ be the set of all keys that can generate $u$. We note that the salt $S^*$ that is used to encrypt $M^*$ does not appear in $u$, and therefore, and since $E$ is an ideal ABC, exactly $2^{-|M^*|}$ of the keys in $K_u$ encrypt $M^*$ to $C$ under the salt $S^*$.

**Lemma 4.8** Let $X$ be a reasonable AMode, let $E$ be an ideal ABC, and let $A$ be a semi-CPA adversary against $X^E$. Let $u$ be a stream generated by the run of $A$, and let $\ell$ be the length of messages allowed by $\gamma$. Then, for every message $C$, such that $|C| = \ell$ it holds that $\Pr[C^* = C|u] = 2^{-\ell}$, when the randomness is over the choice of the key $K$.

**Proof.** Bearing in mind Lemma 4.7, we can write:

$$\Pr[C^* = C|u] = \sum_{M^*} \Pr[C^* = C|M^* \land u] \cdot \Pr[M^*]$$

$$= \sum_{M^*} 2^{-\ell} \cdot \Pr[M^*] = 2^{-\ell}$$

We would like to emphasize that Lemmas 4.7 and 4.8 deal with ideal ABCs, and therefore we are able to disregard $\sigma_E$ in their proofs. Since the ABC is ideal then making off line queries to the ABC would not help the adversary gain any new useful information.

Now we can prove Theorem 4.3:

**Proof.** Let $u$ be a stream generated by the run of $A$, and let $C^*$ be the challenge given to $A$ by the oracle. Using Bayes’ Theorem, we get that for every possible message $M$,

$$\Pr[M^* = M|u \land C^*] = \frac{\Pr[M^* = M] \cdot \Pr[u|M^* = M] \cdot \Pr[C^*|M^* = M \land u]}{\Pr[u] \cdot \Pr[C^*|u]}.$$
Since the random choice of \(M^*\) is independent of the stream \(u\), then \(\Pr[u|M^* = M] = \Pr[u]\), and we can write:

\[
\Pr[M^* = M|u \land C^*] = \frac{\Pr[M^* = M] \cdot \Pr[C^*|M^* = M \land u]}{\Pr[C^*|u]}
\]

\[
= \Pr[M^* = M] = \gamma(M).
\]

The advantage of \(A\) can be written as:

\[
\text{Adv}^{\text{sem-CPA}}_{XE}(A) = \Pr[\alpha = f(M^*)|C^*] - \Pr[\alpha = f(M')]
\]

\[
= \sum_{M:f(M)=\alpha} \Pr[M^* = M|C^*] - \sum_{M':f(M')=\alpha} \Pr[M']
\]

\[
= \sum_{M:f(M)=\alpha} \gamma(M) - \sum_{M':f(M')=\alpha} \gamma(M') = 0.
\]

For the CCA security analysis of our framework, we follow the common distinction between \textit{a priori} CCA (also known as CCA1), and \textit{a posteriori} CCA (also known as CCA2) [20]. The \textit{a priori} CCA notion, allows the adversary to make her queries only \textit{before} she is given the challenge, but not afterwards, while the \textit{a posteriori} CCA notion allows the adversary to make queries also \textit{after} the challenge was given. We therefore define two types of adversaries:

**Definition 4.8** Let \(\mathcal{E}\) be an encryption scheme. An \(\mathcal{E}\)-prio-CCA adversary is an \textit{a priori} CCA-adversary.

**Definition 4.9** Let \(\mathcal{E}\) be an encryption scheme. An \(\mathcal{E}\)-post-CCA adversary is an \textit{a posteriori} CCA-adversary.

**Definition 4.10** Let \(\mathcal{E}\) be an encryption scheme. The \textit{a priori} CCA security of \(\mathcal{E}\) is defined as:

\[
\text{Sec}^{\text{sem-prio-CCA}}_{\mathcal{E}}(\sigma_E, \sigma) \triangleq \max_{A \in \mathcal{A}_{\text{prio}}^{\mathcal{E}}} \text{Adv}^{\text{sem-CCA}}_{\mathcal{E}}(A),
\]

where \(\mathcal{A}_{\text{prio}}^{\mathcal{E}}\) is the set of all \textit{a priori} CCA adversaries against \(\mathcal{E}\) that use no more than a total of \(\sigma\) blocks in their queries, and no more than \(\sigma_E\) off-line calls to the encryption scheme.
Definition 4.11 Let $\mathcal{E}$ be an encryption scheme. The posteriori CCA security of $\mathcal{E}$ is defined as:

$$\text{Sec}_{\mathcal{E}}^{\text{sem-post-CCA}}(\sigma_E, \sigma) \triangleq \max_{A \in \mathcal{A}_{\text{post}}^{\sigma}} \text{Adv}_{\mathcal{E}}^{\text{sem-CCA}}(A),$$

where $\mathcal{A}_{\text{post}}^{\sigma}$ is the set of all posteriori CCA adversaries against $\mathcal{E}$ that use no more than a total of $\sigma$ blocks in their queries, and no more than $\sigma_E$ off-line calls to the encryption scheme.

While we cannot claim our framework to be secure against a posteriori CCA-adversary\(^4\), we can prove it is secure against a priori CCA-adversaries, as we do in the following theorem.

**Theorem 4.4** Let $X$ be a reasonable AMode, and let $E$ be an ideal ABC. Then,

$$\text{Sec}_{X^E}^{\text{sem-prio-CCA}}(\sigma_E, \sigma) = 0.$$

The proof is similar to the proof of Theorem 4.3. The similarity is based on the fact that in both cases (CPA and posteriori-CCA) the challenge $C^*$ is the only ciphertext that the adversary sees along the game that was encrypted using $S^*$.

\(^4\)The posteriori CCA-adversary can use the salt $S^*$ that was used for the challenge in the queries she makes after the challenge was given, and thus may gain some non-trivial information about the challenge $M^*$. 

46
Chapter 5

The Relation to Other Cryptographic Primitives

In this chapter, we discuss the relations between ABCs and other cryptographic primitives, including stream ciphers and hash functions.

5.1 Stream Ciphers

In Corollary 4.1, we show that an AMode with an underlying ideal ABC results in perfect ind-CPA security. However, it is possible to gain the same ind-CPA security using a stream cipher, which does not require the complexity of an ideal ABC. Consider the following ABC: \( E_{K,S,t}(M) = M \oplus f_K(S,t) \) whose key space contains \( 2^{s+c+n} \) keys, and such that for every possible mapping \( g : \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n \) there exists a single key \( K \) such that \( f_K(\cdot,\cdot) = g(\cdot,\cdot) \). \( E \) has a much smaller key space than the key space of an ideal ABC. Nevertheless, \( \text{Sec}^{\text{PP}}_E(q) = 0 \) for any \( q \) that allows salt-counter respecting. Thus, from Theorem 4.1 it holds that \( \text{Sec}^{\text{ind-CPA}}_{X_E}(\sigma) = 0 \), for any reasonable AMode \( X \). We note, that using \( E \) in AECB mode is equivalent for using a stream cipher generated by \( f(\cdot,\cdot) \) by querying it for a given fixed salt and a counter which is incremented. It looks like we can use such stream ciphers (that are much simpler than ideal ABCs) and gain the same security as ideal ABCs provide. However, using an ideal ABC with an AMode that does not produce a stream (such as AECB, or ACBC) provides better tolerance for misuse.
In any stream cipher, a reuse of the same stream results with an immediate compromise of the security. When the same stream is used for two different messages \( M_1, M_2 \), then \( M_1 \oplus M_2 \) can be calculated from the corresponding ciphertexts \( C_1, C_2 \). Moreover, when a misuse of the protocol allows the same IV or nonce to be reused, and thus the same stream to be reused, an adversary who has a pair \((M, C)\) of plaintext-ciphertext blocks can calculate the encryption of any other message using the same stream. When considering an ideal ABC and an AMode such as AECB or ACBC, such attacks are impossible, even when the same key-salt-counter combination is used more than once due to some fault or misuse.

We note that AModes such as AOFB, and ACTR generate streams, and therefore, do not provide good tolerance for misuse as AModes such as AECB and ACBC offer.

Another advantage of our framework over stream ciphers is in its security against time-memory-data tradeoffs. Time memory-data tradeoffs are widely discussed for stream ciphers. Indeed, 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 which initialize their IV whenever the key is changed. But in the more strict version in which the salt is never repeated (not even after changing the key) then these attacks become impractical.

### 5.2 Hash Functions

The HAIFA framework for cryptographic hash function introduced the idea of salt and bit-counter for compression functions [8]. A natural question to ask is how can these new compression functions and block ciphers relate. In this chapter 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 [11]. Other possible constructions were studied in [28] – some of them could be used for our purpose as well.

#### 5.2.1 Using Davies-Meyer Construction

The Davies-Meyer construction is widely 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: $C_{DM}(h, M) = E_M(h) \oplus h$. Here, $k$ is the length of the message block for the compression function and $n$ is the length of the chaining value.

The Davies-Meyer construction is used in many hash functions, usually with a custom block cipher, designed specifically for the use of the compression function. Common block ciphers are unsuitable for the Davies-Meyer construction, since this construction demands that the key size of the cipher is equal to the message block length, and that the size of the compression function’s chaining value is equal to the cipher’s block length. 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 internal collisions is 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 increases the complexity of the block cipher’s execution (in particular its key schedule), and will especially affect encryption of short messages. Despite these issues, given the good understanding of designing and constructing block ciphers that the cryptographic community has developed, it is still a convenient way to construct a compression function using a custom block cipher and the Davies-Meyer construction.

We can also use a Davies-Meyer construction to build a HAIFA compression function $C_{DM}^{HAIFA} : \{0,1\}^n \times \{0,1\}^b \times \{0,1\}^s \times \{0,1\}^c \rightarrow \{0,1\}^n$ (where $n$ is the length of the chaining value and $b = k$ is the block size) using an ABC, $E$. Such a compression function is defined as $C_{DM}^{HAIFA}(h, M, S, \#bits) = E_{M,S,f(\#bits)}(h) \oplus h$, for some function $f$ that outputs a unique counter value for every possible $\#bits$ value (typically, $f = \#bits$).

A major security drawback of the Davies-Meyer construction is that it is easy to find fixpoints in the compression function. One can easily find a fixpoint $(h^*, M^*)$ simply by fixing a message block $M^*$, decrypting the zero constant and setting $h^* = E_{M^*}^{-1}(0)$. Such a fixpoint can be used for a second-preimage attack as shown in [13].

We note that with our new block cipher framework and the HAIFA
framework, finding a fixpoint (although possible) cannot be used for a second-preimage attack (as in [13]). The \(\#\text{bits}\) parameter of the compression function (the \(t\) parameter in \(E\)) prevents attacks that take advantage of the easy-to-find fixpoints. It is still possible to find a tuple \((h^*, M^*, S^*, \#\text{bits}^*)\) such that \(C_{DM}(h^*, M^*, S^*, \#\text{bits}^*) = h^*\), but for any \(\#\text{bits}' \neq \#\text{bits}^*\) (and specifically for the value of \(\#\text{bits}'\) that matches the next block) it holds, with a very high probability, that \(C_{DM}(h^*, M^*, S^*, \#\text{bits}') \neq h^*\). Moreover, it is difficult to find a message block \(M'\) such that \(C_{DM}(h^*, M', S^*, \#\text{bits}') = h^*\). Therefore, a fixpoint cannot be concatenated to itself in order to expand the message like the second pre-image attacks require.

We note that due to performance issues it might be better to use the message block of the compression function as the \textit{salt} of the ABC and not as the key, so the key scheduling algorithm is not re-executed for every block, since that in a good ABC design, an update of the salt is cheaper than a full key-scheduling.
Chapter 6

Simple ABCs Based on AES

In this chapter we describe three AES-based ABCs, all of which with 128-bit keys, 128-bit salts and 64-bit counters. We then analyze the security and performance of each of the suggestions.

6.1 Proposals for ABCs

In this section we suggest three AES-based ABCs. The first one, ABC1, uses AES as a black box. It is therefore very easy to implement, using an out-of-the-box AES implementation, but its speed is much slower than the speed of AES-128, since it uses several AES-128 calls. ABC2 and ABC3 both modify the round keys used by the AES algorithm, mixing the salt and the counter into them, and thus producing an ABC. ABC2 is based on AES-256, and ABC3 is based on AES-128.

In each of our proposals the counter mixing is designed to be faster than the salt mixing, since it is much more frequent. I.e., more time and effort are invested in the process of salt mixing which occurs once for every message, rather than for every single block.

6.1.1 ABC1

Our first ABC is a variant of a triple-AES structure that uses AES [34] as a black-box and is implemented as follows:

\[ ABC1_{K,S,t}(M) = AES-128_{K'}(AES-128_{K}(M \oplus t') \oplus t'), \]
where $K' = \text{AES-128}_K(S)$ and $t' = t||t$.

6.1.2 ABC2

ABC2 is based on AES-256. It modifies the key scheduling algorithm of AES-256 to allow a mixing of the salt and the counter into the state. 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 as $K' = K||\text{AES-128}_K(S)$. 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 then mixed into five of these round keys – $RK_1[2]$, $RK_1[4]$, $RK_1[7]$, $RK_1[10]$, and $RK_1[12]$, in order to generate the set of round-keys that will be used during the encryption. Each time the counter is mixed into a round key it is XORed into two consecutive (cyclicly) columns of the round key. 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 $RK_1[12]$ (where column 0 is the most significant column of the round key). The counter $t$ is another set of round keys $RK_2$. The counter $t$ is represented in big endian notation, i.e., the most significant column of the counter is column 0 and the most significant byte of the counter is byte 0, which belongs to column 0.

Formally, $RK_2$ is given by:

\[
RK_2[i] = RK_1[i], \quad \text{(for } i = 0, 1, 3, 5, 6, 8, 9, 11, 13, 14) \\
RK_2[7] = RK_1[7] \oplus 0^{64}|t[0]|t[1], \\
RK_2[10] = RK_1[10] \oplus t[1]|0^{64}|t[0], \\
RK_2[12] = RK_1[12] \oplus t[0]|t[1]|0^{64},
\]

where $t[0]$ and $t[1]$ are columns 0 and 1 of the counter, respectively.

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 the counter before the next mixing of the counter occurs, we selected to have at least two AES rounds between consecutive counter mixings — two AES rounds are enough to ensure full diffusion [10]. A full diffusion of a counter mixing ensures that all of the bytes of the state are influenced by the injected counter. The next counter mixing, changes only two of the four columns of the state, while keeping the other two unchanged.

### 6.1.3 ABC3

In order to achieve faster encryption than the encryption offered by ABC1 and ABC2, we suggest ABC3, which is based on AES-128. 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,

\[
\begin{align*}
K_1 &= K \quad ; \quad RK_1 = KS(K_1) \\
K_2 &= AES-128_{K_1}(S) \quad ; \quad RK_2 = KS(K_2) \\
K_3 &= AES-128_{RK_1 \oplus RK_2}(K) \quad ; \quad RK_3 = KS(K_3) \\
RK_4 &= RK_1 \oplus RK_2 \oplus RK_3,
\end{align*}
\]

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-128 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 $RK_4[9]$. The result is another set of round
keys $RK_5$. Formally, $RK_5$ is given by:

\[
RK_5[i] = RK_4[i], \quad \text{(for } i = 0, 2, 4, 6, 8, 10) \\
RK_5[1] = RK_4[1] \oplus t[0]||t[1]||t[0]||t[1], \\
RK_5[5] = RK_4[5] \oplus 0^{64}||t[0]||t[1], \\
RK_5[7] = RK_4[7] \oplus t[1]||0^{64}||t[0], \\
RK_5[9] = RK_4[9] \oplus t[0]||t[1]||t[0]||t[1],
\]

where $t[0]$ and $t[1]$ are columns 0 and 1 of the counter, respectively.

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.

### 6.2 Security of Our Proposals

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 offer the expected security. We also analyze the security of the ABCs suggested above against a variety of known attacks, such as differential attack and linear attack.

#### 6.2.1 Equivalent Keys

An important requirement from an ABC would be that the different permutations are independent from each other. Thus, we would like it to be difficult to find two different key-salt-counter combinations $(K_1, S_1, t_1) \neq (K_2, S_2, t_2)$ that specify the same permutation.

There are $2^{128!}$ possible permutations over blocks of 128 bits. An ABC of 128-bit key, 128-bit salt, and 64-bit counter, consists of only $2^{320}$ of these permutations. Thus, if each such permutation is chosen randomly and independently, the probability for colliding permutations is negligible. However,
in actual ABCs, such as in our proposals, the permutations are not chosen independently, and therefore we need to examine them carefully and thoroughly.

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, with extremely high probability, there are no pairs $(K_1, S_1) \neq (K_2, S_2)$ and a counter $t$ for which $ABC1_{K_1,S_1,t}(M) = ABC1_{K_2,S_2,t}(M)$ for every $M$. We believe that the probability that there exist different key-salt-counter combinations $(K_1, S_1, t_1) \neq (K_2, S_2, t_2)$ for which $ABC1_{K_1,S_1,t_1}(M) = ABC1_{K_2,S_2,t_2}(M)$ for every $M$, is extremely low.

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. This difference cannot be canceled by the counter neither, since the counter is not mixed until the second round (the third round-key). Of course, 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 believe 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. We believe that every key-salt-counter combination results with a unique set of round keys $RK_5$, which specifies a unique permutation.

### 6.2.2 Security of ABC1 Against Any Known or Chosen Plaintext Attack

We show a reduction from any known/chosen plaintext attack on ABC1 to an adaptive known/chosen plaintext attack on AES. We conclude that the security of ABC1 against known/chosen plaintext attacks (e.g., differential attack, linear attack or time-memory tradeoff attacks) is not worse than the security of AES-128 against these attacks.
Consider an adversary $A$ that commits a known/chosen plaintext attack on ABC1. We build an adaptive adversary $B$ that commits a known/chosen plaintext attack on AES-128 using the adversary $A$. Whenever $A$ asks for the encryption of a plaintext block $M$, with a salt $S$, and with a counter $t$, $B$ asks for $K' = AES_{128K'}(S)$. Then $B$ computes $V_1 = AES_{128K'}(M) \oplus t'$, asks for $V_2 = AES_{128K}(V_1)$, and computes $C = AES_{128K'}(V_2 \oplus t')$, thus receiving $C = ABC1_{K,S,t}(M)$. $B$ provides $A$ with $C$ as an answer. 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.

If $A$ uses $q$ pairs $(M, C)$ of known/chosen plaintext for its attack, then $B$ uses at most $2q$ pairs for its own attack (for every pair used by $A$, $B$ might has to ask for the encryption of the salt used with this pair as well). The running time of $B$ is equal to $2q$ AES encryption operations (for computing $V_1$ and $C$) added to the complexity time of $A$.

### 6.2.3 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 [10], is $2^{-3}$. Using a computer program we found out that the lower bound on the number of active S Boxes in a linear characteristic is 80 for the full ABC2, and 55 for the full 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. By using a fixed plaintext, a fixed salt, and different counters, the adversary can cancel the influence of the plaintext bits over the linear characteristic. This way, after two ABC2 rounds (or after one ABC3 round), the state is the same for all the encryptions required by the attack (since the counter has not been introduced yet). Therefore, the probability

---

1 Without the influence of the counter, the propagation of a difference during the ABC2 or ABC3 encryption would be exactly as the propagation of a difference during AES-256 or AES-128 encryption, respectively. Therefore, if the bits of the counter are not part of the linear characteristic, then the analysis made for AES is valid for ABC2 and ABC3 as well. Thus, for characteristics that do not include bits of the counter we get the same analysis of AES, saying that the bias of any 8-round linear characteristic is no more than $2^{-150}$. 

---

56
for any such linear characteristic is influenced only by the remaining rounds (12 in ABC2, and 9 in ABC3). Still, 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 ABC3 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 of no threat to the security of ABC2 and ABC3.

6.2.4 Security of ABC2 and ABC3 Against Extended Differential Attacks

Unlike traditional differential attacks, in differential attacks on ABCs the adversary is not limited to introducing the differences through the plaintext. She can also introduce the differences through the salt or through the counter.

**Definition 6.1** 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 a plaintext difference. An attack that uses an extended differential characteristic is called an 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 (unless the full AES-128 allows this property, which implies a significant weakness in AES). 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 shows that after six rounds of ABC2 and after seven first 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 \(2^{-318}\). Similarly, ABC3 does not have such an extended differential characteristic that has probability higher than \(2^{-180}\). The full analysis of the minimal number of differentially active S Boxes, for any number of ABC2/ABC3 rounds is given in Table 6.1.

### 6.2.5 A Pitfall of Using ABC3 in a Davies-Meyer Construction

It has been brought to our attention that when using ABC3 in a Davies-Meyer construction, the finding of a pre-image for the resulting compression function can be sped-up by a factor of about 1.5 [1]. The attack is based on the following observation: Let \(K\) be a key, and let \(S\) be a salt such that

---

<table>
<thead>
<tr>
<th># rounds</th>
<th>ABC2</th>
<th>AES-256</th>
<th>ABC3</th>
<th>AES-128</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>9</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>25</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>26</td>
<td>9</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>30</td>
<td>19</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>25</td>
<td>34</td>
<td>25</td>
<td>34</td>
</tr>
<tr>
<td>8</td>
<td>26</td>
<td>50</td>
<td>26</td>
<td>50</td>
</tr>
<tr>
<td>9</td>
<td>28</td>
<td>51</td>
<td>26</td>
<td>51</td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>55</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>11</td>
<td>33</td>
<td>59</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>40</td>
<td>75</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>46</td>
<td>76</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>53</td>
<td>80</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6.1: The minimum number of active S Boxes in a differential characteristic as a function of the number of rounds.
$AES_K(S) = K$ (given a key $K$, $S$ can be easily found by decrypting $K$ under itself). Thus, we get that $K_2 = K_1$, $K_3 = AES_0(K)$, and $RK_4 = RK_3$. Therefore, only one AES-128 encryption ($K_3 = AES_0(K)$) and one calculation of the AES-128 key-scheduling algorithm are needed for the calculation of $RK_4$ (instead of three AES-128 key-scheduling calculations and two AES-128 encryptions, when using the full algorithm). According to our performance analysis (Table 6.2), it is possible, in such a case, to calculate $RK_4$ in 400 CPU cycles, instead of 924 CPU cycles. Now, given a target value $y$ and an initial value $IV$, we find a pre-image of a Davies-Meyer compression function, based on ABC3, by employing an exhaustive search. For every candidate message block $M$, we chose the salt $S_M = AES^{-1}_M(M)$. Now, we can check our candidate by calculating $ABC3^{-1}_{M,S_M,y}(y \oplus IV)$ and checking whether the result equals $IV$. Since we can calculate $ABC3^{-1}_{M,S_M,y}(y \oplus IV)$ using only one AES-128 key-scheduling calculation and one AES-128 encryption (instead of three key-scheduling calculations, and two AES-128 encryptions), and since that calculating $S_M$ for each message block $M$ involves another single calculation of the AES-128 key-scheduling algorithm and another single AES-128 encryption, then the cost of checking each candidate is about 804 cpu cycles, instead of 1180 CPU cycles using the full calculation of ABC3 (key scheduling + encryption). The result is a speed-up by a factor of almost 1.5.

### 6.3 Performance of ABC1, ABC2, and ABC3

In order to check the performance of ABC1, ABC2, and ABC3, we used an Intel Xeon E5540 processor with a 2.53GHz CPU, cache size of 8192KB, and 8.175GB RAM that runs a Red Hat Enterprise Linux Server release 5.5. We used parts of the AES code of Brian Gladman [19] for our ABCs implementations. We used gcc 4.1.2 compiler with ‘-O3’ optimization for compiling our code. The results are summarized in Table 6.2 along with the performance results of Gladman’s AES-128 and AES-256 code for comparison.
<table>
<thead>
<tr>
<th></th>
<th>ABC1</th>
<th>ABC2</th>
<th>ABC3</th>
<th>AES-128</th>
<th>AES-256</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key Scheduling</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>545.60</td>
<td>625.62</td>
<td>924</td>
<td>162.14</td>
<td>211</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>11</td>
<td>2.53</td>
<td>4.87</td>
<td>3.06</td>
<td>6.38</td>
</tr>
<tr>
<td>Median</td>
<td>545</td>
<td>627</td>
<td>925</td>
<td>162</td>
<td>212</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>691.54</td>
<td>334.78</td>
<td>255.52</td>
<td>240.32</td>
<td>318.07</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>34.37</td>
<td>9.86</td>
<td>0.74</td>
<td>6.64</td>
<td>12.92</td>
</tr>
<tr>
<td>Median</td>
<td>690</td>
<td>334</td>
<td>255</td>
<td>238</td>
<td>318</td>
</tr>
</tbody>
</table>

Table 6.2: Performance of the different implementations
Bibliography


תבנית חדשה לפני בלוקים סימטריים

אורי אברם
תבנית חדשה ל撺י בלוקים סימטרים

חיבור על מחקר

לשם مليוי חלקי של הדירישות לכתבת התואר
מגישר למדעי בנведущה המחשב

אוריא אברבה

הוגס לסטט הטכנולוגי – מכון טכנולוגי לישראל
שבט התחעיב
חיפה
פברואר 2012
המחקר נָעָשֶׁה בְּנַחַיָּית פָּרֹופ', אלְיָבִי בְיַה עֲדָר', אוֹר דְּנֵכְלֵם בַּפּוֹקָלֶה לְמַדִּיעַ הַמְּחֻשָּׁב.

אנְיוֹּ מַוְּדָה לְשֵׁנִי הַמֶּנְחָה שֶל, פָּרֹופ', אלְיָבִי בְיַה, אוֹר דְּנֵכְלֵם, עַל הַמַּעֲנַכָּה, הָנֵחְיָטָם.

אָנִי מָזוֹדָה לְשֵׁנִי הַמֶּנְחָה שֶל, פָּרֹופ', אלְיָבִי בְיַה, אוֹר דְּנֵכְלֵם, עַל הַמַּעֲנַכָּה, הָנֵחְיָטָם.

עַץָּתָה חֵסְבָה לָאוֹרְד' כֶּל חֶדְרָא.

אָנִי מָזוֹדָה לְשֵׁנִי הַמֶּנְחָה (ולַמְּפַאָלַת) עַל הַחַמִכָּה הַכְּסֵפִיָּה הָנֵדָיבָה בְּחַשְׁתַּלֶמֶטְוַי.
התקינו

הצפנה מופיעה בעולמנו של今天我们 בימים רב שחר של תחומי. ישנם ושישים שמות חרות, לוחות
אטלט א טמיט,屾🙇‍♀️urally, אנו הצפנה מופיעה יッシュ על גני רשת האינטernetes, שיתוף
בתקשורות שלילתיות, תקשורת חברתית, שידורי בבל החוצה.

צופן בלוקים מתאימים לציפי מקדים מפורטים בהצפנה של יומינו. הם עליכ הנכון הבלי של שי
קלטים והם בבל, המילים לแพงה, המפתית מנדלקת לשלוחה
ולמקלבל, והפלט לאות התוכל המוזמן, שעון אלבל בבל הקלטים. כאשר קבוצה את
המפתית מתקבלת מיום ומגיעה לפשיט ויתור לוחבונה הממפה בין בבל הקלט בבל הפלה.

על מהד הוצפף וה돈ת אופני תקניים בולר דקORIA בלוק מפורטים בבל
משתמשי באלגוריתמים CTR - ECB, CBC, OFB, CFB. ישנם אופני 회תקופות של
מפעלים, כמו גם פעולות ריבים, דמייה המה - (NIST) ומפקד לציפי
קבע הסדרת התוכני הנסגרים על ידי מדמות התוכני התוכני התוכני התוכני (XOR) בתוכני
бавו אופני פעולות של יומינו (באותיות) ישן חלש מובנה ההגנה לכל שוש אייר היא بيان
אוסף בבלוקים פורמלטים לכל מאמץ את היילוב הב買った מתמיד, שילוב מבחרית פעולות
על מהד הוצפף וה돈ת אופני תקניים, היו עלחל השוק מידי על ההודע המסורית
ולItemCount.

כבﺷדד זה או מתיעב בחינה הידיעה של צפיני בלוקים别人的 פעולות מתאימות לפנים
המוחנים בשתיות והתיקון של הוא מתוכני בבלוקים של פעולות מתכונה
(counter) המזה (salt) בעלת היחידה, ABC - Advanced Block Cipher (AECB)
עד שיי קולים: מלח המילה (ממח) לשון

המנון הווה כול מתוכני הבולים של ימי שניים זר צים לע עץ למחבל הפנים/פועלת
והดวง היוד. המנון מתוכני בבלוקים מתכונה בהיחוד מבורות, נ롣ה בבל
קריאת אתפוכן מבולים המתכונה, ממוקם עם, ויאני nau כים פוליתא האת שע
הוחזק. המוסיפים

המלון הווה כולם שנותר בבולים שעון ב bácת הוצפף/השון, כ שוש אמא אาะה

א
The use of repeated frequent plaintexts under a single key, salt salt, for encryption shall not be used. The counter salt shall be set to a new, different value every time.

Shifatokhut hashtokhut, hash tokhut, is not to be used in the same function, and even to be combined.

The practice of using the same salt in encryption, and even to be combined.

Advanced - A Modes of Operation (Modes of Operation - Advanced), and also the definition of a new function for encryption under a single key, salt.

The counter salt shall be set to a new, different value every time.

Encrypted text is encrypted text, and even to be combined.

The practice of using the same salt in encryption, and even to be combined.

Advanced - A Modes of Operation (Modes of Operation - Advanced), and also the definition of a new function for encryption under a single key, salt.

The counter salt shall be set to a new, different value every time.

Encrypted text is encrypted text, and even to be combined.

Advanced - A Modes of Operation (Modes of Operation - Advanced), and also the definition of a new function for encryption under a single key, salt.

The counter salt shall be set to a new, different value every time.

Encrypted text is encrypted text, and even to be combined.

Advanced - A Modes of Operation (Modes of Operation - Advanced), and also the definition of a new function for encryption under a single key, salt.

The counter salt shall be set to a new, different value every time.

Encrypted text is encrypted text, and even to be combined.
אן מתיחסים את בטיחותם שנגד התוכן שונים קונן התוכןزارצאות ל-NIST
tוכן ליניארי, מתוכנות את ביטויים מביתיות לאללי שאם
תופעות ותאימות ל- AES.
ולאובך יעילה ויעי, אנו מצפים לשיפור בהדרגה את רמת האבטחה של השפה.