

# A Note on a CBC-Type Mode of Operation

George Teşeleanu<sup>1,2</sup> 

<sup>1</sup> Advanced Technologies Institute  
10 Dinu Vintilă, Bucharest, Romania  
`tgeorge@dcti.ro`

<sup>2</sup> Simion Stoilow Institute of Mathematics of the Romanian Academy  
21 Calea Grivitei, Bucharest, Romania

**Abstract.** In this paper we formally introduce a novel mode of operation based on the cipher block chaining mode. The main idea of this mode is to use a stateful block cipher instead of a stateless one. Afterwards, we show how to implement our proposal and present a performance analysis of our mode. Next, we provide a concrete security analysis by computing a tight bound on the success of adversaries based on their resources. The results of our performance and security analyses are that this novel mode is more secure than the cipher block chaining mode for large files, but the encryption/decryption time doubles/triples. Therefore, our novel mode is suitable for encrypting large files, when higher security is required, but speed is not paramount. Note that the changes required to transform the software implementations of the cipher block chaining mode into this new mode are minimal, and therefore transitioning to this new mode is straightforward.

## 1 Introduction

One of the most popular classical mode of operation is called Cipher Block Chaining (CBC) mode [16]. The CBC mode is widespread and very widely used, and therefore is standardised in [6, 8, 9, 15]. Implementations of CBC can be found in software libraries such as [1–3, 5].

In this paper we introduce a novel CBC-based mode of operation. More precisely, we use the CBC mode with a stateful block cipher instead of a stateless one. Specifically, rather of using the same key to encrypt each block of data, we modify the key schedule such that after generating the round keys necessary for encryption it also generates an encryption key<sup>3</sup>. Therefore, after encrypting the first block of data, we memorize the encryption key generated by the key schedule. The memorized key is then used to encrypt the second block of data and the process is repeated for the remaining blocks. After finishing all the blocks, the key is reset to the original value and the block cipher is now ready to encrypt the next set of data. Note that resetting the key before each new set of plaintexts avoids synchronisation problems.

---

<sup>3</sup> formed by concatenating one or more additional generated round keys depending on the size of the round and encryption keys

After formalizing our novel CBC-based mode, we provide some implementations details. We start with describing two possible approaches for transforming a stateless block cipher into a stateful one. Since the second one is more suitable for all three ciphers implemented in Mbed TLS [3], namely AES [14], ARIA [7] and Camellia [10], we made the necessary implementation modifications needed to obtain a stateful cipher. To analyze the performance of our mode, we also modified the CBC implementation found in Mbed TLS [3]. We observed that our mode’s encryption/decryption time is two/three times slower than that of classical CBC.

In the last part of the paper, we provide a tight security bound. To achieve this we first compute an upper bound and then we devise an attack that has a success probability close to this bound. Therefore, there is no significantly better bound than the one given in this paper. Based on this security bound, we conclude that this mode is more secure than CBC for large files. Since modifying existing CBC implementations to use a stateful block cipher is straightforward, we recommend switching to this novel mode for large files, when the additional processing time does not lead to bottlenecks. An example of such a use case is the following: suppose you want to store some sensitive data that is rarely accessed, such as backup copies of surveillance footage. In this case, the data is not accessed frequently, so the speed of accessing it is not crucial. However, security is essential to ensure the confidentiality of the data.

*Structure of the paper.* We introduce notations and definitions used throughout the paper in Section 2. In Section 3 we formalize the Running Key CBC mode of operation. Implementation details and a security analysis are provided in Sections 4 and 5. We conclude in Section 6.

## 2 Preliminaries

*Notations and conventions.* Throughout the paper  $|b|$  will denote the bit size of  $b$  and  $\oplus$  the bitwise xor operation. By  $x||y$  we understand the concatenation of the strings  $x$  and  $y$ . Also, we define  $0^s$  and  $1^s$  as a string of  $s$  zeros and ones, respectively. We use the notation  $x \xleftarrow{\$} X$  when selecting a random element  $x$  from a sample space  $X$ . We denote by  $x \leftarrow y$  the assignment of the value  $y$  to the variable  $x$ . The probability that event  $E$  happens is denoted by  $Pr[E]$ . Hexadecimal strings are marked by the prefix  $0x$ . The most  $\ell$  significant bits of  $Q$  are denoted by  $MSB_{\ell}(Q)$ .

For all complexity measures we work in the random access model and all the adversaries are probabilistic polynomial time machines (PPT). When we talk about the running time of an opponent, we will include the actual running time plus the length of the adversary’s description (*i.e.* the length of the random access machine program that describes the adversary). Also, queries are answered in one unity of time.

## 2.1 PRF/PRP

We start by defining the security notions commonly used to quantify the security of a block cipher [20] and then we provide the link between the two notions [12].

**Definition 1 (Pseudorandom Function - PRF, Pseudorandom Permutation - PRP).** *A function  $F : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  is a PRF or PRP if for any PPT algorithm  $A$  which makes at most  $q$  oracle queries, the following advantages*

$$\begin{aligned} ADV_F^{\text{PRF}}(A) &= \left| Pr[A^{F_K(\cdot)} = 1 | K \xleftarrow{\$} \{0, 1\}^k] - Pr[A^{\rho(\cdot)} = 1 | \rho \xleftarrow{\$} \mathcal{F}] \right|, \\ ADV_F^{\text{PRP}}(A) &= \left| Pr[A^{F_K(\cdot)} = 1 | K \xleftarrow{\$} \{0, 1\}^k] - Pr[A^{\pi(\cdot)} = 1 | \pi \xleftarrow{\$} \mathcal{P}] \right| \end{aligned}$$

are negligible, where  $F_K(X) = F(X, K)$ , and  $\mathcal{F}$  and  $\mathcal{P}$  denote the sets of all functions and permutations from  $n$ -bit strings to  $n$ -bit strings.

**Lemma 1 (PRP/PRF Switching Lemma).** *Let  $F : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  be a function. Also, let  $A$  be an adversary that asks at most  $q$  oracle queries. Then*

$$\left| Pr[A^{\rho(\cdot)} = 1 | \rho \xleftarrow{\$} \mathcal{F}] - Pr[A^{\pi(\cdot)} = 1 | \pi \xleftarrow{\$} \mathcal{P}] \right| \leq \frac{q(q-1)}{2^{n+1}}.$$

As a consequence, we have that

$$|ADV_F^{\text{PRF}}(A) - ADV_F^{\text{PRP}}(A)| \leq \frac{q(q-1)}{2^{n+1}}.$$

## 2.2 Symmetric Key Encryption

We further define symmetric key encryption schemes [11] and provide one of the security notions commonly used to model chosen-plaintext attacks [12, 20].

**Definition 2 (Symmetric Key Encryption - SKE).** *A symmetric key encryption (SKE) scheme usually consists of three PPT algorithms: Setup, Encrypt and Decrypt. The Setup algorithm takes as input a security parameter and outputs the secret key. Encrypt takes as input the secret key and a message and outputs the corresponding ciphertext. The Decrypt algorithm takes as input the secret key and a ciphertext and outputs either a valid message or an invalidity symbol (if the decryption failed).*

**Definition 3 (Indistinguishability under Chosen Plaintext Attacks - IND-CPA).** *The security model against chosen plaintext attacks for a SKE scheme  $\mathcal{SE}$  is captured in the following game:*

*Setup( $\lambda$ ): The challenger  $C$  generates the secret key  $K \xleftarrow{\$} \{0, 1\}^k$  and a bit  $b \xleftarrow{\$} \{0, 1\}$ .*

*Queries:* Adversary  $A$  sends  $C$  two messages  $m_0, m_1 \in \{0, 1\}^*$  of the same length. The challenger encrypts  $m_b$  and obtains the ciphertext  $c_b = \mathcal{SE}(m_b, K)$ . The value  $c_b$  is sent to the adversary.

*Guess:* After making a polynomial number of queries, the adversary outputs a guess  $b' \in \{0, 1\}$ . He wins the game, if  $b' = b$ .

The advantage of an adversary  $A$  which runs in time  $t$ , makes at most  $q$  oracle queries, these totaling  $\sigma$  bits and is attacking a SKE scheme is defined as

$$ADV_{\mathcal{SE}}^{\text{IND-CPA}}(A) = |2Pr[b = b'] - 1|$$

where the probability is computed over the random bits used by  $C$  and  $A$ . A SKE scheme is IND-CPA secure, if for any PPT adversary  $A$  the advantage  $ADV_{\mathcal{SE}}^{\text{IND-CPA}}(A)$  is negligible.

We further present the CBC encryption algorithm in Algorithm 1 and then we present its security margins as stated in [11, 20]. Let  $K$  be an  $k$ -bit random key,  $P$  a plaintext and  $C$  a ciphertext.

---

**Algorithm 1: CBC Mode of Operation.**

---

```

1 Function Encrypt( $P, K$ ):
2   if  $|P| \bmod n \neq 0$  or  $|P| = 0$  then return  $\perp$ ;
3    $P_1 \parallel \dots \parallel P_m \leftarrow P$  where  $|P_i| = n$ ;
4    $IV \xleftarrow{\$} \{0, 1\}^n$  and  $C_0 \leftarrow IV$ ;
5   foreach  $i \in [1, m]$  do  $C_i \leftarrow E_K(P_i \oplus C_{i-1})$ ;
6    $C \leftarrow C_1 \parallel \dots \parallel C_m$ ;
7 return  $(IV, C)$ ;

8 Function Decrypt( $IV, C, K$ ):
9   if  $|C| \bmod n \neq 0$  or  $|C| = 0$  then return  $\perp$ ;
10   $C_1 \parallel \dots \parallel C_m \leftarrow C$  where  $|C_i| = n$ ;
11   $C_0 \leftarrow IV$ ;
12  foreach  $i \in [1, m]$  do  $P_i \leftarrow E_K^{-1}(C_i) \oplus C_{i-1}$ ;
13   $P \leftarrow P_1 \parallel \dots \parallel P_m$ ;
14 return  $P$ ;
```

---

**Theorem 1 (Upper Bound on Security of CBC Mode Using a Block Cipher).** Let  $E : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  be a block cipher. Let  $A$  be a PPT adversary against the IND-CPA security of CBC that runs at most time  $t$  and queries at most  $\sigma$   $n$ -bit blocks. Then there exists an adversary  $B$  against the PRP security of  $E$  such that

$$ADV_{CBC}^{\text{IND-CPA}}(A) \leq ADV_E^{\text{PRP}}(B) + \frac{2\sigma^2}{2^n},$$

and where  $B$  makes at most  $\sigma$  queries and runs in time at most  $t + \mathcal{O}(n\sigma)$ .

**Theorem 2 (Lower Bound on Security of CBC Mode Using a Block Cipher).** *Let  $E : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  be a block cipher and let  $\sigma \in [1, \sqrt{2^{n+1}}]$ . Then there exists an adversary  $A$  against the IND-CPA security of CBC such that*

$$ADV_{CBC}^{\text{IND-CPA}}(A) \geq \frac{0.15\sigma^2}{2^n},$$

and which asks one query consisting of  $\sigma$   $n$ -bit blocks and runs in time at most  $\mathcal{O}(n\sigma \log \sigma)$ .

### 3 Running Key CBC SKE

Let  $E : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  be a block cipher. Instantiating the running key CBC (denoted RK-CBC) with  $E$  and with a random initialisation vector (denoted  $IV$ ) leads to a stateless symmetric encryption scheme. Let  $K_f = \{K_i\}_{i>0}$  be a family of distinct  $k$ -bit random keys,  $P$  a plaintext and  $C$  a ciphertext. We present in Algorithm 2 the exact detail of RK-CBC.

---

**Algorithm 2:** Running Key CBC Mode of Operation.

---

```

1 Function Encrypt( $P, K_f$ ):
2   if  $|P| \bmod n \neq 0$  or  $|P| = 0$  then return  $\perp$ ;
3    $P_1 \parallel \dots \parallel P_m \leftarrow P$  where  $|P_i| = n$ ;
4    $IV \xleftarrow{\$} \{0, 1\}^n$  and  $C_0 \leftarrow IV$ ;
5   foreach  $i \in [1, m]$  do  $C_i \leftarrow E_{K_i}(P_i \oplus C_{i-1})$ ;
6    $C \leftarrow C_1 \parallel \dots \parallel C_m$ ;
7 return  $(IV, C)$ ;

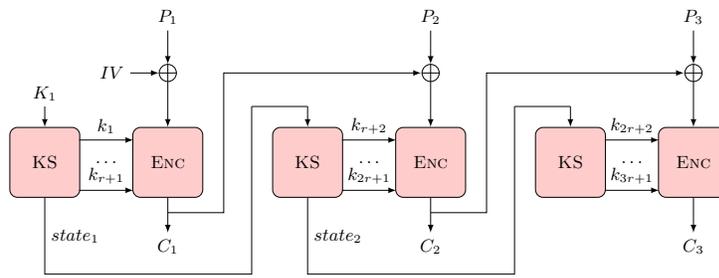
8 Function Decrypt( $IV, C, K_f$ ):
9   if  $|C| \bmod n \neq 0$  or  $|C| = 0$  then return  $\perp$ ;
10   $C_1 \parallel \dots \parallel C_m \leftarrow C$  where  $|C_i| = n$ ;
11   $C_0 \leftarrow IV$ ;
12  foreach  $i \in [1, m]$  do  $P_i \leftarrow E_{K_i}^{-1}(C_i) \oplus C_{i-1}$ ;
13   $P \leftarrow P_1 \parallel \dots \parallel P_m$ ;
14 return  $P$ ;
```

---

### 4 Implementation Details

When studying the security of iterated block ciphers [18], it is assumed that the cipher's round sub-keys are independent. This assumption is reasonable in practice [13, 17, 22]. Using this assumption, we can transform a stateless block cipher into a stateful one using two methods.

In the first method we simply memorize the initial key into a buffer and then we continuously run the key schedule algorithm until all data blocks are encrypted (see Figure 1). More precisely, for an  $r$  round cipher<sup>4</sup> and an  $m$  block plaintext the key schedule algorithm will generate  $(r + 1)m$  keys. Since we assumed that the round sub-keys are independent, we obtain  $m$  independent instantiations of the iterated block cipher. Therefore, we can use it to implement the RK-CBC mode. Unfortunately, not all key schedules can be easily modified to support continuous generation. For example, the AES [14] and ARIA [7] key schedules support this out of the box, but for the Camellia [10] key schedule we could not devise a suitable modification.



**Fig. 1.** The First Method for Implementing RK-CBC when  $m = 3$ .

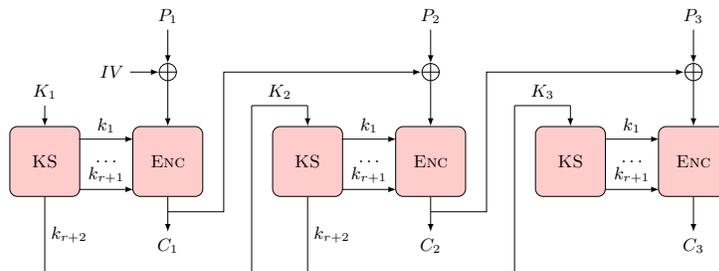
In the second method we modify the key schedule to generate a few additional round keys after computing the round keys needed for encryption. Then we concatenate the additional round keys to obtain a new encryption key, that is used to encrypt the next block of data. The process continues until all blocks are processed. For example, if  $n = k$  we generate only one additional round key  $k_{r+2}$  and set  $K_2 = k_{r+2}$  (see Figure 2).<sup>5</sup> If  $n = 2k$  we have to generate two additional round keys  $k_{r+2}, k_{r+3}$  and set  $K_2 = k_{r+2} || k_{r+3}$ . Using this method we managed to modify all three block ciphers implemented in Mbed TLS [3]. The technical details of the changes can be found in Appendix A.

Taking into account the above discussions, we can transform a CBC implementation into a RK-CBC one by simply resetting the key before encrypting each block of plaintexts and by using either stateful block cipher described above. Note that, compared to CBC, RK-CBC needs an extra buffer to memorize the initial key.

Let  $t_{ks}$  be the time needed run the key schedule once and  $t_{enc}$  be the time needed to compute a block cipher encryption without taking into account  $t_{ks}$ . In terms of performance, the time required to encrypt an  $m$ -block of data using

<sup>4</sup> We consider that an  $r$  round cipher uses  $r + 1$  sub-keys.

<sup>5</sup> Figures 1 and 2 are based on the TikZ found in [19].



**Fig. 2.** The Second Method of Implementing RK-CBC when  $m = 3$  and  $n = k$ .

CBC is  $\mathcal{O}(m \cdot t_{enc})$ .<sup>6</sup> On the other hand, the time required by the RK-CBC is  $\mathcal{O}(m \cdot (t_{ks} + t_{enc}))$ . Therefore, CBC is  $\mathcal{O}((t_{ks} + t_{enc})/t_{enc})$  faster than RK-CBC.

To compute the exact slowdown for Mbed TLS [3], we implemented the RK-CBC mode and run the Mbed TLS benchmark program on a CPU Intel i7-4790 4.00 GHz. Note that to compute the rate of kilobytes per second, Mbed TLS first sets an internal alarm to 1 second and then increments a counter for each 1024 bytes block that is processed, until the alarm is set off. The counter represents the given rate. In the case of the cycle per byte rate, the benchmark program computes the number of cycles needed to process 1024 blocks of size 1024 bytes. Then it divides the resulting number by  $1024 \cdot 1024$  and outputs the rate. The results of our experiments can be found in Figures 3 to 8. Compared to RK-CBC encryption/decryption, classical CBC encryption is 2x/3.5x faster for AES<sup>7</sup>, 2.3x/3.2x faster for ARIA and 2.2x/2.5x faster for Camellia.

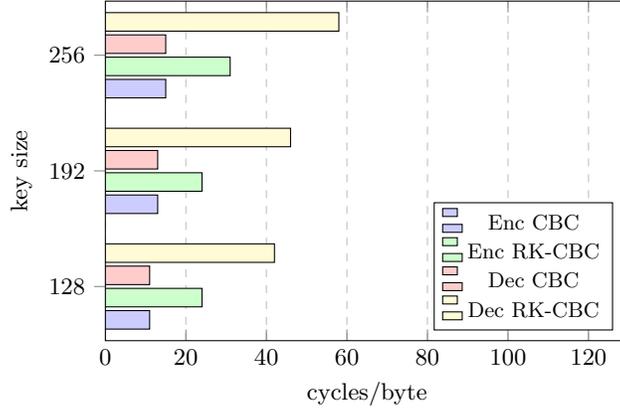
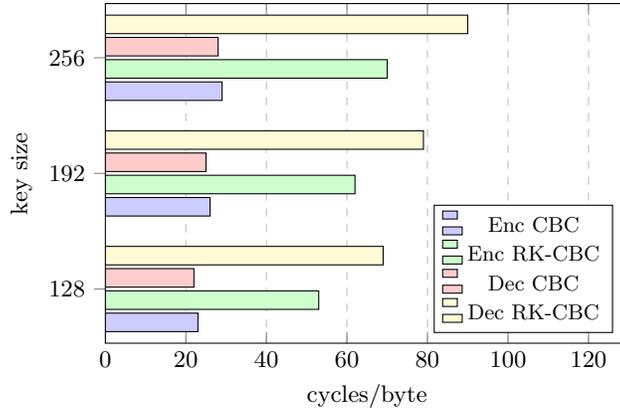
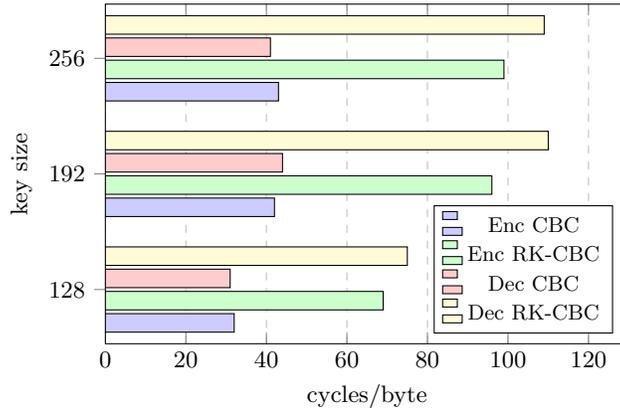
We also carried a series of experiments to test the validity of our assumption (*i.e.* the generated key bits are independent and identically distributed). Therefore, we set  $K_1$  as either  $0^s$  or  $1^s$ , where  $s = 128/192/256$  and we generated  $K_2, K_3 \dots$  using the second method. Then we applied the NIST test suite [4, 21] to check if the key bits of  $K_2, K_3 \dots$  are independent and identically distributed. We obtained that the samples pass all the statistical tests, and have an entropy of 0.99 per bit and 7.9 per byte. Thus, based on the results of the experiments we conducted, we can safely assume the generated keys are random and mutually independent.

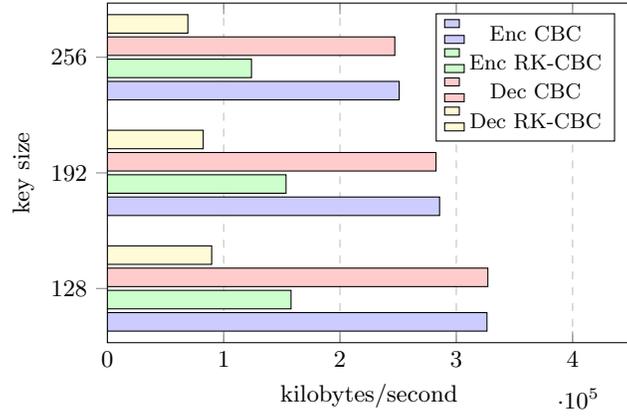
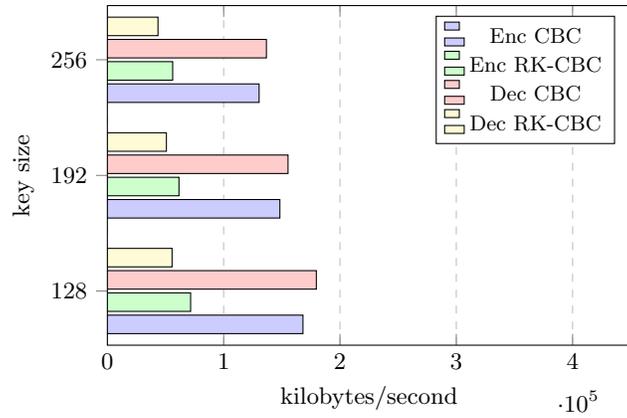
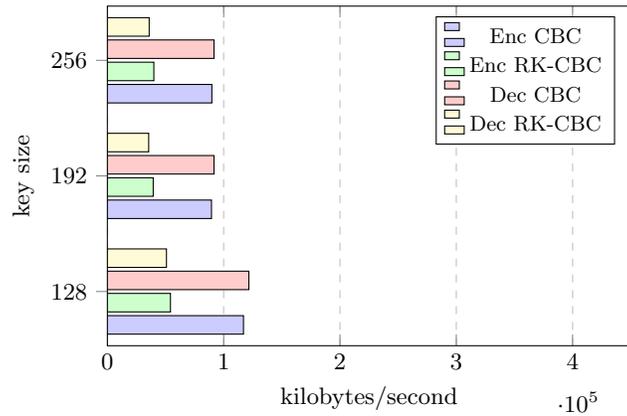
## 5 Security Analysis

Before starting our analysis, some notations are required. Therefore,  $A$  will denote a PPT algorithm which makes at most  $q$  oracle queries, these totaling at most  $\sigma$   $n$ -bit blocks, unless mentioned otherwise. Let  $\sigma' = \sigma - q + 1$  and  $\sigma = \alpha q + r$ ,

<sup>6</sup> Usually, the key schedule is run once at the beginning and the round keys memorized for the duration of encryption, and thus  $t_{ks}$  can be ignored.

<sup>7</sup> Note that we deactivated the AES-NI instructions for a fair performance between the three block ciphers.

**Fig. 3.** AES performance**Fig. 4.** ARIA performance**Fig. 5.** Camellia performance

**Fig. 6.** AES performance**Fig. 7.** ARIA performance**Fig. 8.** Camellia performance

where  $r < q$ . For simplicity, we denote by  $\rho_f$  the oracle that responds to a query of type  $(i, X)$  with  $\rho_i(X)$ , where  $\{\rho_i\}_{i \in [1, \sigma']}$  is a family of distinct random functions. Similarly, we denote by  $\pi_f$  the oracle that responds with  $\pi_i(X)$ , where  $\{\pi_i\}_{i \in [1, \sigma']}$  is a family of distinct random permutations. Lastly, we denote by  $F_f$  the oracle that responds with  $F_{K_i}(X_i)$ , where  $\{K_i\}_{i \in [1, \sigma']}$  is a family of distinct random  $k$ -bit keys and  $F : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  is a function.

For simplicity, we further assume that adversary  $A$  never asks a query that is not composed of  $n$ -bit blocks. In our case, queries that do not satisfy this restriction generate error messages, and thus they are pointless for  $A$ .

Now, we start our security analysis by first studying the security of the RK-CBC mode instantiated with random functions  $\rho_i$  from  $\mathcal{F}$  instead of  $E_{K_i}$ . We denote this version by RK-CBC( $\rho_f$ ). Note that in this case, the decryption algorithm returns  $\perp$  with probability  $1 - 2^{-n}$ , since it does not have a correct decryption property.

**Lemma 2 (Upper Bound on Security of RK-CBC Mode Using a Random Function).** *Let  $A$  be a PPT adversary against the IND-CPA security of the RK-CBC( $\rho_f$ ) mode. Then*

$$ADV_{RK-CBC(\rho_f)}^{\text{IND-CPA}}(A) \leq \frac{\alpha q^2 + r^2}{2^n}.$$

*Proof.* To prove our result, we introduce a series of games that help us to compute the upper bound of the advantage. All the games have the same initialisation phase, which is presented in Algorithm 3. This phase starts by setting  $2\sigma'$  dictionaries to empty and a flag to `false`. Note that, we must have  $2\sigma'$  dictionaries since the adversary can make  $q - 1$  queries using only one block of data and one query using  $\sigma'$  blocks. All other query strategies of the adversary require less than  $\sigma'$  functions.

---

**Algorithm 3:** Initialisation.

---

```

1 for  $i \in [1, \sigma - q + 1]$  do  $\rho_i^0 \leftarrow \emptyset$  and  $\rho_i^1 \leftarrow \emptyset$ ;
2  $bad \leftarrow \text{false}$ ;

```

---

The oracle phase of the games is presented in Algorithm 4. The first game, denoted by  $G_0$ , simulates perfectly the query phase of the IND-CPA game under a family of random function when  $b = 0$ . Similarly,  $G_1$  simulates the case  $b = 1$ . In the last game  $G_2$  we simply remove lines 6 and 7 from Algorithm 4. Therefore, we have

$$\begin{aligned} ADV_{RK-CBC(\rho_f)}^{\text{IND-CPA}}(A) &= \left| Pr[A^{G_1(\cdot)} = 1] - Pr[A^{G_0(\cdot)} = 1] \right| \\ &\leq Pr[A^{G_2(\cdot)} \text{ sets } bad \text{ to } \text{true}], \end{aligned}$$

since all the games are identical until the flag `bad` is set to `true`.

---

**Algorithm 4:** Oracle.

---

**Input:** Two plaintexts  $P_0$  and  $P_1$   
**Output:** A ciphertext  $C$

- 1  $P_1^0 \parallel \dots \parallel P_m^0 \leftarrow P_0$  and  $P_1^1 \parallel \dots \parallel P_m^1 \leftarrow P_1$ ;
- 2  $C_0 \xleftarrow{\$} \{0, 1\}^n$
- 3 **foreach**  $i \in [1, m]$  **do**
- 4      $X_i^0 \leftarrow P_i^0 \oplus C_{i-1}$  and  $X_i^1 \leftarrow P_i^1 \oplus C_{i-1}$ ;
- 5      $C_i \xleftarrow{\$} \{0, 1\}^n$ ;
- 6     **if**  $\rho_i^0(X_i^0) \neq \perp$  **then**  $bad \leftarrow \mathbf{true}$  and  $C_i \leftarrow \rho_i^0(X_i^0)$ ; //include this line only  
       in game  $G_0$
- 7     **if**  $\rho_i^1(X_i^1) \neq \perp$  **then**  $bad \leftarrow \mathbf{true}$  and  $C_i \leftarrow \rho_i^1(X_i^1)$ ; //include this line only  
       in game  $G_1$
- 8      $\rho_i^0(X_i^0) \leftarrow C_i$  and  $\rho_i^1(X_i^1) \leftarrow C_i$ ;
- 9 **end**
- 10  $C \leftarrow C_0 \parallel C_1 \parallel \dots \parallel C_m$
- 11 **return**  $C$

---

The last thing that we need to do is to compute the probability of setting  $bad$  to **true**. Note that all dictionaries start empty and grow with a maximum of one point per query. Therefore, the strategy that maximizes the probability of having a collision in any of the dictionaries is to make  $q$  queries to as many dictionaries as possible. The previous condition is satisfied when the adversary is to making  $r$  queries of size  $\alpha + 1$  blocks and  $q - r$  queries of size  $\alpha$  blocks. Using this strategy we obtain that the probability of setting  $bad$  due to dictionary  $i \in [1, \alpha]$  is at most  $(1 + 2 + \dots + q - 1)/2^n$  and to dictionary  $\alpha + 1$  is at most  $(1 + 2 + \dots + r - 1)/2^n$ . To see this we have to note that each  $C_{i-1}$  is random and independent of the current values stored in  $\rho_i^0$ , and thus each  $X_i^0 = P_i^0 \oplus C_{i-1}$  is random and independent of the current values stored in  $\rho_i^0$ . The same holds for the link between  $X_i^1$  and the values from  $\rho_i^1$ . Keeping all this in mind, we obtain

$$Pr[A^{G_2(\cdot)} \text{ sets } bad \text{ to } \mathbf{true}] \leq 2 \left( \alpha \frac{q(q-1)}{2^{n+1}} + \frac{r(r-1)}{2^{n+1}} \right) \leq \frac{\alpha q^2 + r^2}{2^n},$$

as desired. □

In order to make the switch from random function to random permutation we need an equivalent of the Switching Lemma (Lemma 1). We also introduce an equivalent of Definition 1 that will be useful later.

**Definition 4 (Family of Pseudorandom Functions - F-PRF, Family of Pseudorandom Permutation - F-PRP).** *A function  $F : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  is a F-PRF or F-PRP if for any PPT algorithm  $A$  which makes at most  $q$  oracle queries of type  $(i, \cdot)$ , these totaling at most  $\sigma$   $n$ -bit blocks, the following*

---

**Algorithm 5: Initialisation.**


---

1 **for**  $i \in [1, \sigma - q + 1]$  **do**  $\rho_i \leftarrow \emptyset$ ;  
2  $bad \leftarrow \text{false}$ ;

---

*advantages*

$$ADV_F^{\text{F-PRF}}(A) = \left| Pr[A^{F_f(\cdot)} = 1 | K_i \xleftarrow{\$} \{0, 1\}^k] - Pr[A^{\rho_f(\cdot)} = 1 | \rho_i \xleftarrow{\$} \mathcal{F}] \right|,$$

$$ADV_F^{\text{F-PRP}}(A) = \left| Pr[A^{F_f(\cdot)} = 1 | K_i \xleftarrow{\$} \{0, 1\}^k] - Pr[A^{\pi_f(\cdot)} = 1 | \pi_i \xleftarrow{\$} \mathcal{P}] \right|$$

*are negligible.*

*Remark 1.* Note that when  $q = \sigma$  the F-PRF/F-PRP security notions are identical to the PRF/PRP ones.

**Lemma 3 (F-PRP/F-PRF Switching Lemma).** *Let  $A$  be an adversary that asks at most  $q$  oracle queries, these totaling at most  $\sigma$   $n$ -bit blocks. Then*

$$Pr[\text{switch}] = \left| Pr[A^{\rho_f(\cdot)} = 1 | \rho_i \xleftarrow{\$} \mathcal{F}] - Pr[A^{\pi_f(\cdot)} = 1 | \pi_i \xleftarrow{\$} \mathcal{P}] \right| \leq \frac{0.5(\alpha q^2 + r^2)}{2^n}.$$

*As a consequence, we have that*

$$|ADV_F^{\text{F-PRF}}(A) - ADV_F^{\text{F-PRP}}(A)| \leq \frac{0.5(\alpha q^2 + r^2)}{2^n}.$$

*Proof.* Before we begin the proof, let us first introduce a few notations. Let  $\text{Range}(f)$  be the set of  $n$ -bit strings  $Y$  such that  $f(X) = Y$  for some  $X$ . The set of  $n$ -bit strings that are not in  $\text{Range}(f)$  is denoted by  $\overline{\text{Range}(f)}$ . We also assume that  $A$  never repeats a query, since it will always receive the same answer. Therefore, is not helpful for  $A$  to repeat oracle interrogations.

As before, to prove our result we use a sequence of games. The initialisation and oracle phases are given in Algorithms 5 and 6. The first game  $G_0$  simulates perfectly a random function, while the second one  $G_2$  a random permutation. Hence, we have

$$Pr[\text{switch}] = \left| Pr[A^{G_1(\cdot)} = 1] - Pr[A^{G_2(\cdot)} = 1] \right| \leq Pr[A^{G_2(\cdot)} \text{ sets } bad \text{ to } \text{true}],$$

since the only difference between the two games is line 2 from Algorithm 6.

As in the case of the proof of Lemma 2, the best strategy for  $A$  is to ask as many  $q$  queries of type  $(i, \cdot)$  as possible. This happens when  $A$  queries  $\alpha$  oracles with  $q$  queries and one oracle with  $r$  queries. Therefore, the probability of setting  $bad$  due to one of the  $\alpha$  dictionaries is at most  $q(q-1)/2^{n+1}$  and to the last dictionary is at most  $r(r-1)/2^{n+1}$ . Summing it all up, we obtain that

$$Pr[A^{G_2(\cdot)} \text{ sets } bad \text{ to } \text{true}] \leq \left( \alpha \frac{q(q-1)}{2^{n+1}} + \frac{r(r-1)}{2^{n+1}} \right) \leq \frac{0.5(\alpha q^2 + r^2)}{2^n},$$

---

**Algorithm 6:** Oracle.

---

**Input:** An input  $(i, X)$   
**Output:** An output  $Y$

- 1  $Y \xleftarrow{\$} \{0, 1\}^n$ ;
- 2 **if**  $Y \in \text{Range}(\rho_i)$  **then**  $\text{bad} \leftarrow \text{true}$  and  $Y \xleftarrow{\$} \overline{\text{Range}(\rho_i)}$ ; //include this line only in game  $G_1$ ;
- 3  $\rho_i(X) \leftarrow Y$ ;
- 4 **return**  $Y$

---

as desired.

As a consequence, we have the following inequalities

$$\begin{aligned}
|ADV_{\mathcal{F}}^{\text{F-PRF}}(A) - ADV_{\mathcal{F}}^{\text{F-PRP}}(A)| &\leq \left| Pr[A^{F_f(\cdot)} = 1] - Pr[A^{\rho_f(\cdot)} = 1] \right. \\
&\quad \left. - Pr[A^{F_f(\cdot)} = 1] + Pr[A^{\pi_f(\cdot)} = 1] \right| \\
&\leq \left| Pr[A^{\pi_f(\cdot)} = 1] - Pr[A^{\rho_f(\cdot)} = 1] \right| \\
&\leq \frac{0.5(\alpha q^2 + r^2)}{2^n}.
\end{aligned}$$

□

We further apply Lemma 3 to see what happens if we use random permutations instead random function when instantiating the RK-CBC mode. We denote this version by RK-CBC( $\pi_f$ ).

**Corollary 1 (Upper Bound on Security of RK-CBC Mode Using a Random Permutation).** *Let  $A$  be a PPT adversary against the IND-CPA security of the RK-CBC( $\pi_f$ ) mode. Then*

$$ADV_{\text{RK-CBC}(\pi_f)}^{\text{IND-CPA}}(A) \leq \frac{2(\alpha q^2 + r^2)}{2^n}.$$

*Proof.* Let  $F_\rho = \text{RK-CBC}(\rho_f)$  and  $F_\pi = \text{RK-CBC}(\pi_f)$ . We have that

$$\begin{aligned}
ADV_{\text{RK-CBC}(\pi_f)}^{\text{IND-CPA}}(A) &= \left| Pr[A^{F_\pi(\cdot)} = 1 | b \leftarrow 1] - Pr[A^{F_\pi(\cdot)} = 1 | b \leftarrow 0] \right| \\
&\leq \left| Pr[A^{F_\pi(\cdot)} = 1 | b \leftarrow 1] - Pr[A^{F_\rho(\cdot)} = 1 | b \leftarrow 1] \right| \\
&\quad + \left| Pr[A^{F_\rho(\cdot)} = 1 | b \leftarrow 1] - Pr[A^{F_\rho(\cdot)} = 1 | b \leftarrow 0] \right| \\
&\quad + \left| Pr[A^{F_\rho(\cdot)} = 1 | b \leftarrow 0] - Pr[A^{F_\pi(\cdot)} = 1 | b \leftarrow 0] \right|.
\end{aligned}$$

Lets see why we can use Lemma 3 for the first and third addend. Lets assume that there exists an efficient adversary  $A$  that can distinguish between  $F_\rho(\cdot)$  and

---

**Algorithm 7: Algorithm B.**


---

```

1 Run  $A$ ;
2 while  $A$  makes an oracle query call  $(P)$  do
3    $P_1 \parallel \dots \parallel P_m \leftarrow P$ ;
4    $IV \xleftarrow{\$} \{0, 1\}^n$  and  $C_0 \leftarrow IV$ ;
5   foreach  $i \in [1, m]$  do  $C_i \leftarrow \theta(i, P_i \oplus C_{i-1})$ ;
6    $C \leftarrow C_1 \parallel \dots \parallel C_m$ ;
7   Send  $(IV, C)$  to  $A$ ;
8 end
9  $b \leftarrow A$ ;
10 return  $b$ 

```

---

$F_\pi(\cdot)$ . Then we construct an adversary  $B$  (see Algorithm 7) that can distinguish between  $\rho_f(\cdot)$  and  $\pi_f(\cdot)$ . Let  $\theta(\cdot)$  be either  $\rho_f(\cdot)$  or  $\pi_f(\cdot)$ . Then we have

$$\begin{aligned} Pr[B^{\theta(\cdot)} = 1] &= Pr[B^{\rho_f(\cdot)} = 1] = Pr[A^{F_\rho(\cdot)} = 1], \\ Pr[B^{\theta(\cdot)} = 1] &= Pr[B^{\pi_f(\cdot)} = 1] = Pr[A^{F_\pi(\cdot)} = 1]. \end{aligned}$$

Therefore, we obtain that

$$\begin{aligned} \left| Pr[A^{F_\rho(\cdot)} = 1] - Pr[A^{F_\pi(\cdot)} = 1] \right| &= \left| Pr[B^{\rho_f(\cdot)} = 1] - Pr[B^{\pi_f(\cdot)} = 1] \right| \\ &\leq \frac{0.5(\alpha q^2 + r^2)}{2^n}. \end{aligned}$$

Using Lemma 2 and the previous arguments, we have that

$$\begin{aligned} ADV_{RK-CBC(\pi_f)}^{IND-CPA}(A) &\leq \frac{0.5(\alpha q^2 + r^2)}{2^n} + \frac{\alpha q^2 + r^2}{2^n} + \frac{0.5(\alpha q^2 + r^2)}{2^n} \\ &\leq \frac{2(\alpha q^2 + r^2)}{2^n}, \end{aligned}$$

as desired. □

**Theorem 3 (Upper Bound on Security of RK-CBC Mode Using a Block Cipher).** *Let  $E : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  be a block cipher. Let  $A$  be a PPT adversary against the IND-CPA security of RK-CBC that runs at most time  $t$ . Then there exists an adversary  $B$  against the F-PRP security of  $E$  such that*

$$ADV_{RK-CBC}^{IND-CPA}(A) \leq ADV_E^{F-PRP}(B) + \frac{2(\alpha q^2 + r^2)}{2^n},$$

and where  $B$  makes at most  $q$  queries of type  $(i, \cdot)$  and runs in time at most  $t + \mathcal{O}(n\sigma)$ .

---

**Algorithm 8:** Algorithm  $B$ .

---

```

1 Run  $A$ ;
2  $b \leftarrow \{0, 1\}$ ;
3 while  $A$  makes an oracle query call  $(P_0, P_1)$  do
4    $P_1^b \parallel \dots \parallel P_m^b \leftarrow P_b$ ;
5    $IV \xleftarrow{\$} \{0, 1\}^n$  and  $C_0 \leftarrow IV$ ;
6   foreach  $i \in [1, m]$  do  $C_i \leftarrow \theta(i, P_i \oplus C_{i-1})$ ;
7    $C \leftarrow C_1 \parallel \dots \parallel C_m$ ;
8   Send  $(IV, C)$  to  $A$ ;
9 end
10  $b' \leftarrow A$ ;
11 return  $(b == b')$ 

```

---

*Proof.* We will further construct an adversary  $B$  that uses  $A$  in order to break the F-PRP security of  $E$ . Note that  $B$  has access to an oracle  $\theta(\cdot)$  which is either  $\pi_f(\cdot)$  or  $E_f(\cdot)$ . The exact details of  $B$  are given in Algorithm 8.

It is easy to see that the running time of  $B$  is  $\mathcal{O}(n\sigma)$ . To compute the advantage of  $B$  we must observe that

$$\begin{aligned} Pr[B^{\theta(\cdot)} = 1] &= Pr[B^{E_f(\cdot)} = 1] = ADV_{\text{RK-CBC}}^{\text{IND-CPA}}(A), \\ Pr[B^{\theta(\cdot)} = 1] &= Pr[B^{\pi_f(\cdot)} = 1] = ADV_{\text{RK-CBC}(\pi_f)}^{\text{IND-CPA}}(A). \end{aligned}$$

Therefore, we obtain

$$\begin{aligned} ADV_E^{\text{F-PRP}}(B) &= \left| Pr[B^{E_f(\cdot)} = 1] - Pr[B^{\pi_f(\cdot)} = 1] \right| \\ &= \left| ADV_{\text{RK-CBC}}^{\text{IND-CPA}}(A) - ADV_{\text{RK-CBC}(\pi_f)}^{\text{IND-CPA}}(A) \right| \\ &\geq \left| ADV_{\text{RK-CBC}}^{\text{IND-CPA}}(A) - \frac{2(\alpha q^2 + r^2)}{2^n} \right|, \end{aligned}$$

as desired.  $\square$

**Corollary 2 (Simplified Upper Bound on Security of RK-CBC Mode Using a Block Cipher).** *Let  $E : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  be a block cipher. Let  $A$  be a PPT adversary against the IND-CPA security of RK-CBC that runs at most time  $t$ . Then there exists an adversary  $B$  against the F-PRP security of  $E$  such that*

$$ADV_{\text{RK-CBC}}^{\text{IND-CPA}}(A) \leq ADV_E^{\text{F-PRP}}(B) + \frac{2q\sigma}{2^n},$$

and where  $B$  makes at most  $q$  queries of type  $(i, \cdot)$  and runs in time at most  $t + \mathcal{O}(n\sigma)$ .

*Proof.* According to Theorem 3 we have

$$\begin{aligned} ADV_{\text{RK-CBC}}^{\text{IND-CPA}}(A) &\leq ADV_E^{\text{F-PRP}}(B) + \frac{2(\alpha q^2 + r^2)}{2^n} \\ &\leq ADV_E^{\text{F-PRP}}(B) + \frac{2(\alpha q^2 + qr)}{2^n} \\ &= ADV_E^{\text{F-PRP}}(B) + \frac{2q\sigma}{2^n}, \end{aligned}$$

as desired.  $\square$

We further provide an attack that gives  $A$  an advantage of around  $(\alpha q^2 + r^2)/2^n$  to break the IND-CPA security of RK-CBC. This implies that there is no significantly better bound than the one given in Theorem 3.

**Theorem 4 (Lower Bound on Security of RK-CBC Mode Using a Block Cipher).** *Let  $E : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$  be a block cipher and let  $q \in [1, \sqrt{2^{n+1}}]$ . Then there exists an adversary  $A$  against the IND-CPA security of RK-CBC such that*

$$ADV_{\text{RK-CBC}}^{\text{IND-CPA}}(A) \geq \frac{0.15(\alpha q^2 + r^2)}{2^n},$$

and runs in time at most  $\mathcal{O}(n(\alpha q \log q + r \log r))$ .

*Proof.* Adversary  $A$  makes  $q$  queries of type  $(Z_i, R_i)$ , where for  $i \in [1, r]$  we have  $Z_i \leftarrow 0^{n(\alpha+1)}$  and  $R_i \xleftarrow{\$} \{0, 1\}^{n(\alpha+1)}$ , and otherwise we have  $Z_i \leftarrow 0^{n\alpha}$  and  $R_i \xleftarrow{\$} \{0, 1\}^{n\alpha}$ . If there exists two distinct queries  $q_i$  and  $q_j$  such that blocks  $v$  from the challenger, denoted  $B_v(q_i)$  and  $B_v(q_j)$ , are identical then  $A$  selects lexicographically the first such  $(q_i, q_j, v)$  and output 1 if  $B_{v+1}(q_i) \neq B_{v+1}(q_j)$ . Note that the running time of  $A$  is dominated by the running time needed to sort the data received from each oracle, and thus we obtain a running time of  $\mathcal{O}(n(\alpha q \log q + r \log r))$ .

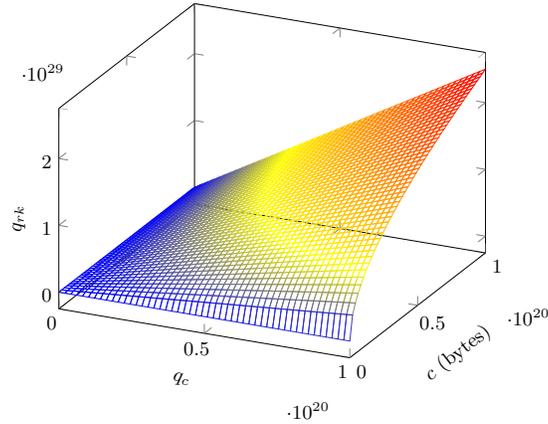
When  $b = 0$ , adversary  $A$  queries identical blocks, and thus if a collision of type  $B_v(q_i) = B_v(q_j)$  occurs we always have  $B_{v+1}(q_i) = B_{v+1}(q_j)$ . Therefore, we always have

$$Pr[A^{\text{RK-CBC}(\cdot)} = 1 | b \leftarrow 0] = 0,$$

and hence

$$ADV_{\text{RK-CBC}}^{\text{IND-CPA}}(A) = Pr[A^{\text{RK-CBC}(\cdot)} = 1 | b \leftarrow 1].$$

When  $b = 1$ , encrypting a random block of data using RK-CBC results in another block of random data, since the image of a uniformly selected value under a permutation remains uniform. Therefore, we can apply the birthday paradox to compute the probability of obtaining  $B_v(q_i) = B_v(q_j)$ . The paradox states that the probability of obtaining a collision when making  $Q \in \{q, r\}$



**Fig. 9.** The curve  $q_{rk} = q_c \sqrt{c/n}$  when  $n = 16$  bytes.

queries to oracle  $v \in [1, \alpha + 1]$  is at least  $0.3Q^2/2^n$ , when  $Q \in [1, \sqrt{2^{n+1}}]$ . Also, note that since  $B_{v+1}(q_i)$  and  $B_{v+1}(q_j)$  are random, the probability of having  $B_{v+1}(q_i) = B_{v+1}(q_j)$  is  $1/2^n$ . Hence, we obtain that

$$\begin{aligned} Pr[A^{\text{RK-CBC}(\cdot)} = 1 | b \leftarrow 1] &\geq \left( \alpha \frac{0.3q^2}{2^n} + \frac{0.3r^2}{2^n} \right) \cdot \left( 1 - \frac{1}{2^n} \right) \\ &\geq \frac{0.15(\alpha q^2 + r^2)}{2^n}, \end{aligned}$$

as desired.  $\square$

**CBC vs. RK-CBC.** To better understand the security gap between the two modes we further consider two scenarios. In the first scenario we assume that we can only encrypt blocks of data of a fixed size  $c = d \cdot n$ , where  $d > 0$ . Therefore, the data requirement for both modes is  $\sigma = q \cdot d$ . According to Theorem 1 we obtain

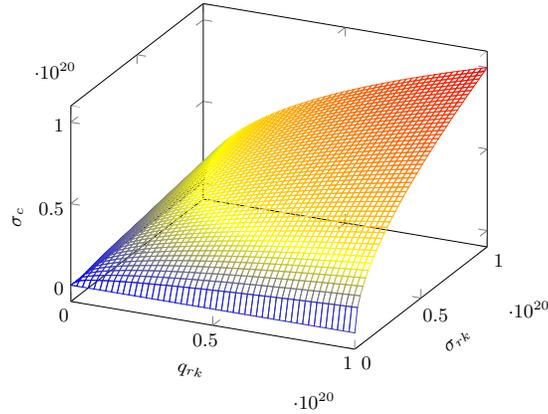
$$ADV_{\text{CBC}}^{\text{IND-CPA}}(A) \leq ADV_E^{\text{PRP}}(B) + \frac{2q^2 d^2}{2^n},$$

and to Theorem 3

$$ADV_{\text{RK-CBC}}^{\text{IND-CPA}}(A) \leq ADV_E^{\text{F-PRP}}(B) + \frac{2q^2 d}{2^n}.$$

Therefore, the free term for RK-CBC is  $d$  times better than the one for CBC. Hence, to obtain the same security margin we have to set the number of RK-CBC queries to  $q_{rk} = q_c \sqrt{d}$  (see Figure 9), where  $q_c$  is the number of CBC queries.

For example, if we consider TLS packages of maximum record size 16 Kilo-bytes we obtain that  $d = 1000$ , which means that we have to make 31x more



**Fig. 10.** The curve  $\sigma_c = \sqrt{q_{rk}\sigma_{rk}}$ .

queries for RK-CBC. If restrict to encrypting DVDs, we have that  $c = 4.37$  Gibibytes and  $d \simeq 2^{28}$ , and thus we have to make  $2^{14}$  times more queries for RK-CBC. Since RK-CBC is slower than CBC (see Section 4) we consider that the security advantage becomes relevant only for large files (*e.g.* DVDs, Blu-rays).

In the second scenario we consider that we can encrypt files of any size. Let  $\sigma_c$  and  $\sigma_{rk}$  be the data requirements from Theorem 1 and Corollary 2. Then the security margins presented in Corollary 2 coincide with the ones from Theorem 1 only if  $\sigma_c^2 = q_{rk}\sigma_{rk}$  (see Figure 10). If we consider that  $\sigma_c = \sigma_{rk} = \sigma$ , then we obtain that  $q_{rk} = \sigma$ . In this case, the only difference is that instead of asking one query consisting of  $\sigma$  blocks (see Theorem 1) we have to ask  $\sigma$  queries each consisting of one block (see Corollary 2). One consequence of this is that if we have an IND-CPA attacker for RK-CBC that asks  $\sigma$  queries, then we can use this attacker to break the IND-CPA security of CBC. More precisely, we only send messages of one block to the CBC oracle, forward the challenges to the RK-CBC attacker and output the bit computed by the RK-CBC attacker. The converse is not true. We can also consider the other extreme case  $q_{rk} = 1$ . This leads to  $\sigma_{rk} = \sigma_c^2$ . Therefore, if we want to make one query as in the case of CBC, we need  $\sigma_c$  times more data for RK-CBC.

## 6 Conclusions

In this work we presented a novel CBC-based mode of operation. Then we provided some implementation details and a performance analysis. Finally, we proved its security by giving an upper bound to polynomial adversaries and, moreover, we showed that this bound is the best possible, by showing a matching attack.

Let  $\sigma_c$  be the data requirements from Theorem 1, and let  $\sigma_{rk}$  and  $q_{rk}$  be the data and query requirements from Corollary 2. From the security analysis

we can see that to match the IND-CPA upper limit of CBC we need to have  $\sigma_c^2 = q_{rk}\sigma_{rk}$ . Therefore, an IND-CPA adversary against the RK-CBC mode requires either more data or more queries. Since obtaining legitimate ciphertexts is problematic in practice, we obtain that the RK-CBC is more secure than CBC. Therefore, because converting existing CBC implementations into RK-CBC ones is straightforward and due to the speed penalty incurred by RK-CBC, we recommend using this novel mode instead of CBC for encrypting large files when processing time is not paramount.

## References

1. Bouncy Castle. <https://www.bouncycastle.org/>
2. Crypto++ Library. <https://www.cryptopp.com/>
3. Mbed TLS. <https://tls.mbed.org/>
4. NIST SP 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation. [https://github.com/usnistgov/SP800-90B\\_EntropyAssessment](https://github.com/usnistgov/SP800-90B_EntropyAssessment)
5. OpenSSL. <https://www.openssl.org/>
6. The AES-CBC Cipher Algorithm and Its Use with IPsec (2003), IETF Standard No. RFC3602
7. Specification of ARIA. Tech. rep., National Security Research Institute (2005)
8. Information Technology - Security Techniques - Modes of Operation for an  $n$ -bit Block Cipher (2017), ISO/IEC Standard No. 10116:2017
9. IEEE Standard for Authenticated Encryption with Length Expansion for Storage Devices (2018), IEEE Standard No. 1619.1
10. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Specification of Camellia - A 128-bit Block Cipher. Tech. rep., Nippon Telegraph and Telephone Corporation and Mitsubishi Electric Corporation (2001)
11. Bellare, M., Rogaway, P.: Introduction to Modern Cryptography. UCSD CSE (2005)
12. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer (2006)
13. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: CRYPTO 1990. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1991)
14. Daemen, J., Rijmen, V.: The Design of Rijndael. Springer (2002)
15. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Methods and Techniques. Tech. rep., NIST Special Publication 800-38A (2001)
16. Ehrsam, W.F., Meyer, C.H., Smith, J.L., Tuchman, W.L.: Message Verification and Transmission Error Detection by Block Chaining (1978), US Patent No. 4,074,066
17. Knudsen, L.R., Mathiassen, J.E.: On the Role of Key Schedules in Attacks on Iterated Ciphers. In: ESORICS 2004. Lecture Notes in Computer Science, vol. 3193, pp. 322–334. Springer (2004)
18. Lai, X., Massey, J.L., Murphy, S.: Markov Ciphers and Differential Cryptanalysis. In: EUROCRYPT 1991. Lecture Notes in Computer Science, vol. 547, pp. 17–38. Springer (1991)
19. Maimuḡ, D.: TikZ for Cryptographers. <https://www.iacr.org/authors/tikz/> (2017)

20. Rogaway, P.: Evaluation of Some Blockcipher Modes of Operation. Tech. rep., Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan (2011)
21. Turan, M.S., Barker, E., Kelsey, J., McKay, K., Baish, M., Boyle, M.: NIST SP 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation. Tech. rep., National Institute of Standards and Technology (2018)
22. Vaudenay, S.: A Classical Introduction to Cryptography: Applications for Communications Security. Springer Science & Business Media (2005)

## A Technical Details

In this appendix we present the changes to the key schedules of AES [14], ARIA [7] and Camellia [10] necessary to convert them into stateful block ciphers.

### A.1 AES

Our presentation is based on [14, Section 3.6]. In order to obtain the key  $K_{next}$  for the next block of data we had to increase the number of rounds  $N_r$  used in the initial key schedule to  $N_r + 1$ . For the 128-bit key schedule we also had to use the additional constant  $RC[11] = 0x6c$ .<sup>8</sup> Therefore, we have  $K_{next} = \text{ExpandedKey}[11]$  for  $k = 128$ ,  $K_{next} = \text{ExpandedKey}[13]$  for  $k = 192$  and  $K_{next} = \text{ExpandedKey}[15]$  for  $k = 256$ .

### A.2 ARIA

Our presentation is based on [7, Section 2.5.2]. The key  $K_{next}$  for the next block of data is  $K_{next} = ek_{14}$  for  $k = 128$  and  $K_{next} = MSB_{192}(ek_{16}||ek_{17})$  for  $k = 192$ . For  $k = 256$  we first had to define the additional round keys

$$\begin{aligned} ek_{18} &= (W_1) \oplus (W_2 \lll 19), \\ ek_{19} &= (W_2) \oplus (W_3 \lll 19), \end{aligned}$$

and then set  $K_{next} = ek_{18}||ek_{19}$ . Note that  $ek_{17}$  is simply  $ek_1$  with the rotation changed from right to left. Therefore, to generate the additional keys we simply continued the previous rationale:  $ek_{18}$  and  $ek_{19}$  are  $ek_2$  and  $ek_3$  with the rotation changed from right to left.

### A.3 Camellia

Our presentation is based on [10, Section 3.4]. To generate the key  $K_{next}$  for the next block of data we had to generate two additional keys  $K_C$  and  $K_D$ . For  $k = 128$  the generation method is provided in Figure 11 and  $K_{next} = K_C$ . For  $k = 192/256$  the generation method is provided in Figure 12.<sup>9</sup>  $K_{next}$  is set as  $MSB_{192}(K_C||K_D)$  for  $k = 192$  and  $K_C||K_D$  for  $k = 256$ .

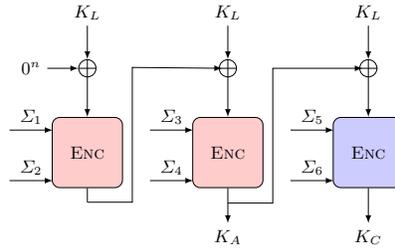
|               |                    |
|---------------|--------------------|
| $\Sigma_7$    | 0xf83d9abfb41bd6b2 |
| $\Sigma_8$    | 0xbe0cd19137e21798 |
| $\Sigma_9$    | 0xbbb9d5dc1059ed8e |
| $\Sigma_{10}$ | 0x29a292a367cd5079 |

**Table 1.** Additional Key Schedule Constants

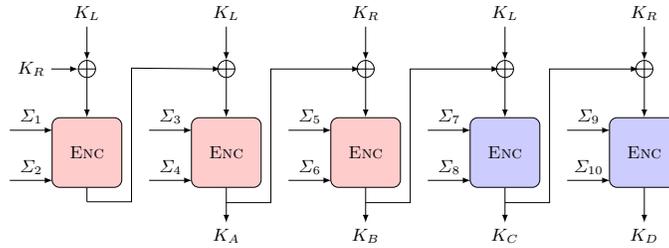
<sup>8</sup> The constant is defined in [14, Section 3.6], but is not required for AES encryption.

<sup>9</sup> Figures 11 and 12 are based on the TikZ found in [19].

In Figures 11 and 12 the red blocks are identical with [10, Figure 8]. Therefore, we simply continued the rationale from [10] to generate  $K_C$  and  $K_D$ . Note that for  $k = 192/256$  we had to generate four additional constants (see Table 1). We used the rationale from [10, Section 3.4]:  $\Sigma_7, \Sigma_8, \Sigma_9$  and  $\Sigma_{10}$  are defined as the continuous values from the second hexadecimal place to the seventeenth hexadecimal place of the hexadecimal representation of the square root of the primes 17, 19, 23 and 29.



**Fig. 11.** Camellia 128-bit Key Schedule.



**Fig. 12.** Camellia 192/256-bit Key Schedule.