WCFB: a tweakable wide block cipher

Andrey Jivsov [1]

Abstract. We define a model for applications that process large data sets in a way that enables additional optimizations of encryption operations. We designed a new strong pseudo-random tweakable permutation, WCFB, to take advantage of identified characteristics. WCFB is built with only $2m + 1$ block cipher invocation for m cipherblocks and $\approx 5m$ XOR operations.

WCFB can benefit from commonly occurring plaintext, such as encryption of a 0^{nm} sector, and repeated operations on the same wide block.

We prove the birthday-bound security of the mode, expressed in terms of the security of the underlying block cipher.

A case analysys of disk block access requests by Windows 8.1 is provided.

1 Introduction

The focus of this paper is how to use a standard block cipher as a component to build a "wide" block cipher, or, in other words, to build a secure PRP from another PRP operating on a smaller domain.

The new mode that we propose, WCFB, stands for Wide Cipher FeedBack mode. WCFB is a "tweakable" mode to allow changes to random "wide" blocks in an encrypted data set.

An environment in which any block of the encrypted data set is allowed to be selectively updated calls for multiple authentication tags to provide data integrity. There are scenarios in which the encrypted data expansion due to authentication tags is not an option, thus making the "wide" block cipher the most secure choice available.

Our contribution is to define the operating environment that represents a large class of systems that are candidates to use a wide block encryption mode, the definition of the mode WCFB and explanations how WCFB is optimized for this environment (Section 4), and, finally, the security proof of WCFB (Section 6).

WCFB is designed with simplicity in mind, in particular, it has modular design and XOR-only operations. While the security proof is monotonous and lengthy, it is very simple in principle. The proof is centered around statements about probability of a collision between n-bit blocks.

[1] Symantec Corporation, 350 Ellis Street, Mountain View, CA 94043, USA
crypto@brainhub.org

2 Notations

In this paper we consider a new *wide* block cipher, built with the (*underlying*) block cipher. The size of the wide block cipher is $l = n * m$ bits. In practical applications the $l/8 \geq 512$ bytes, is a power of two, and is usually a fixed value for a given operating system/hardware/data set. Wide block ciphers work with an underlying n-bit block cipher, such as AES-128. One such underlying block cipher call is identified by one BC in Section 1. Each wide block P, C is represented by m n-bit blocks, which are denoted as $P_i, C_i : i \in [0, m-1]$. P_i denotes the block of the plaintext such that $P = P_0||P_1||\cdots||P_{m-1}$, with similar definition for the C. P_0 refers to the block that occupies the lowest $n/8$ bytes of the memory range in which P resides. This indexing is known as a little-endian notation.

The WCFB mode is defined with the following types of operations: the underlying block cipher encryption or decryption and $\mathsf{GF}(2^n)$ additions (*XOR* or $\oplus$).

WCFB is a secret key permutation. Besides using underlying block cipher with its key k, it has a set of $m+1$ derived subkeys k_i, each n bits long regardless of the size of the key k. There is a key schedule for k that the underlying block cipher uses and a set of the subkeys k_i (which can be viewed as another key schedule).

By the *data set* we mean a set of logically related wide blocks P, C. The same key k will be used for a given data set. An encrypted storage disk is an example of a data set.

A *tweak* is denoted by T. Typically it is an offset in a data set, or a block index.

Description of loops in iterative algorithms use compact notation defined next. All loops in this paper are based on the default iteration from 0 to $m-1$ in the ascending direction. The default loop is for $i = 0, 1, \cdots, m-1$, inclusive; it is denoted as $\forall i \curvearrowright$. The same loop in the reverse direction is denoted as $\forall i \curvearrowleft$. When the range of indexes is different from the default one, it is always explicitly noted. When the direction is omitted, such as in $\forall i$, it is not important (and so the random order is possible).

By $\mathsf{Rand}(\cdot)$ we mean a chosen uniformly at random function from the set of all functions mapping $\{0,1\}^n \rightarrow \{0,1\}^n$.

We use the following symbols, respectively, for a definition, equality, assignment: $\doteq$, $=$, $\leftarrow$.

We rely on security notations from Section 2 of [1].

3 Comparison with other modes

Many wide encryption modes were introduced during the first decade of 2000. Modes with provable security are CMC [1], EME2 [2], PEP [3], TET [4], HEH [5], XCB [6], HCTR [7], HCH [8]. EME2 is standardized in the IEEE 1619.2 Section "Wide-Block Encryption" of the IEEE P1619 standard.

There are modes that do not offer security proofs, such as Elephant+CBC [9], modes that do not offer the benefit of a full block permutation such as XTS [10] (XTS is standardized by the IEEE 1619 standard and by the NIST), and there are uses of standard CBC for wide block encryption, most notably, the CBC is one of the two allowed modes in [11].

The market success of wide encryption modes as of year 2014 is very limited. We are not aware of any existing whole disk product that even offers a wide block encryption mode. One likely explanation for this is that the overhead of the the crypto code is perceived as substantial by end-users, especially with solid-state storage media.

An overview of current modes is provided in the Table 1 of HEH [5]. Counting two GF2mul as one BC call, the best mode under this accounting is CMC [1] at $2m+1$ BC operations for a 2-key variant and $2m+2$ for a 1-key variant. [5] lists HEHfp as $m+1$ BC, $2(m-1)$ GF2mul, which by our accounting is equivalent to $2m$ BC, but it is ignoring other operations that are more complex than XORs. Most importantly, it does not account for additional $2(m-1)$ GF2mul that have one of the multiplication factors random but fixed per the data set. This processing is similiar to EME2's, discussed later in this section. In addition, HEHfp includes $(m-1) \cdot \otimes x$ operations. Finally, it has ≈ 6 XORs per BC.

This brings us back to CMC. CMC has $2m+1$ BC among its positives and a lean mixing layer (3 data-dependent XORs). On the other hand, CMC has two key schedules and is incapable to take advantage of caching. CMC's first encryption pass is performed in CBC mode with T used as a BC. This is an unfortunate choice that denies any benefit of caching of ciphertexts for known plaintext. CMC has $\approx 3m$ XORs for the mixing layer, which is equivalent to WCFB's XORs on modern architectures, because $2m$ of WCFB's XORs per n-bit block are data-independent.

Although this may not account for much in practice, two iterations of WCFB are simple back-and-forth pass over the blocks, with the data from an n-bit block interacting only with an adjacent one, for the best CPU cache utilization. CMC performs the mirroring of the block indices between passes which may be a disadvantage for a large m.

EME2 mode is close to CMC as a suitable alternative for our operating environment at $2m+1+m/n$ BC. EME2 matches WCFB's caching capability at the first encryption layer. An implementation optimized for bulk performance will use $\approx 5m$ XORs, plus bit operations for m data-dependent GF2mul. One of the factors in these GF2mul is of special form $y(x) = x^i$, another is data-dependent. One would expect m GF2mul be accomplished as $2m$ bitwise shifts and m XORs in a sequential manner, unless precomputed tables are used. WCFB uses fewer ($5m$) XORs and has no need for precomputed tables. While the sequential processing in EME2 contradicts its main design goal, only a small constant-degree parallelism (per CPU core) may practically be realizable and this limited parallelism can be accomplished with reasonable precomputed tables. Specifically, degree s parallelism will require 2^s entries in the table and we also assumed m n-bit values that are precomputed per data set to achieve $\approx 7m$ complexity of

the mixing layer. WCFB's mixing layer compares well with EME2 and HEHfp: it is a simple XOR sum of n-bit blocks and fully parallelizable.

WCFB is a single-key mode that achieves $2m + 1$ BC and $\approx 5m$ XORs with $2m$ of them data-independent. WCFB has excellent caching capabilities under update scenarios and other repeated access patterns: in the worst case WCFB saves at least one encryption during an update (which corresponds to the encrypted T), while in the best case WCFB can reuse ciphertexts from $m+1$ encryptions of the n-bit plaintext blocks and T.

WCFB's only operation is XORs on the n-bit blocks. WCFB has no data-dependent branching, and thus offers an excellent protection against side-channel attacks.

Although this is not critical in the defined operating environment, WCFB and CMC allow full parallelism in 2 out of 3 layers of the encrypt-mix-encrypt processing within each wide block. Despite not achieving unrestricted 3 out of 3 layer parallelism, EME2 allows parallel execution of either block cipher layers.

The security of WCFB is quadratic in the number of queries, a typical boundary in this category.

Finally, WCFB has the concept of the IV of the data set. While such an IV' could have been constructed as, for example, $\mathsf{IV} \oplus T$ in place of a T, WCFB has clear separation between data set IV and a block identifier in the mode itself.

4 Our contribution

Our main goal is to make encryption of large data sets faster in practice. Two steps towards this goal are presented here.

First, we define the target applications and operating environment in such terms that allow additional optimizations. For example, the wide encryption modes were traditionally valuing the internal parallelism of the mode, i.e. its ability to process multiple n-bit blocks within the nm-wide block, however, Section 1 leads to the idea that multi-wide-block parallelism is an equivalent if not better method to exploit the parallelism. We argue that it is the case for many target applications, and this observation allows simplification of the mode.

Second, we propose the new mode WCFB, which is a "tweakable" mode to allow changes to random "wide" blocks in an encrypted data set. We describe the WCFB in details and highlight its advantages for processing large data sets. Some design ideas employed in WCFB, in particular, the ability to cache the ciphertexts or subkeys, are general techniques with applications beyond WCFB.

For the operating environment defined in Section 1 WCFB is a wide encryption mode with an *operation count* of $\boxed{2m + 1\ \mathsf{BC}}$ and a lean mixing layer at $\boxed{\approx 5m\text{XORs}}$ (see Section 3 for the comparison with other modes).

$2m+1$ operation count is a reasonable threshold for a tweakable wide encryption mode of encrypt-mix-encrypt, given that there is a mixing step, a tweak,

and an IV that need to be "processed". We show next how caching lowers the metric to $2m$ or lower BC.

There are two likely events that can be relied on in this respect: *low entropy* (n-bit) plaintexts and a favorable *usage pattern*.

The encryption of *low-entropy* data is quite common in practice. Any fixed-size data set is expected to use a fixed-value padding, which is typically zeros. There are many high-level operations that fall into this category, such as zeroization of data set blocks; WCFB will only need m BC to accomplish a zeroization request on a wide block, on par with CBC performance.

Consider the *update* usage pattern, which we define as subsequent decrypt and encrypt operations on the same nm-bit block, either performed as a unified operation, or close enough in time so that some intermediate results can be efficiently re-used. This pattern corresponds to a very common access pattern to encrypted data set: reading a random block in an encrypted data set, decrypting it, making modifications to the plaintext, encrypting it, and then writing back at the same position. Further, consider an application that only adds data to a file. While an application adds a byte at a time to a file, at the lower level of the operating system the storage can only be accessed in blocks, given that the storage devices are block devices. If the wide block encryption is employed for the protection of the disk blocks, even consecutive minor file append operations will likely result in update or at least write operations to the same disk block. WCFB can reuse at least $\hat{E}_{k,k_m}(T)$ in update scenarios. The greater benefit is realized on systems with slower BC. See Section 5 for details.

WCFB implementations can fully benefit from caching, primarily owing to WCFB's ECB-style first pass of encryption.

5 Specification of WCFB

The algorithm is defined in Fig. 1. Diagrams in Section C are helpful in visualizing the data flow.

The WCFB follows the encrypt-mix-encrypt approach. It is built from two passes over n-bit blocks that are CFB-like and CBC-like. The mix step corresponds to the XOR of intermediate n-bit values. WCFB can be viewed as having a double nested structure WCFB[$\hat{E}[E]$], where WCFB is, by and large, defined in terms of $\hat{E}$.

The run-time data-dependent input to WCFB encryption or decryption is a wide block P or C, respectively (steps 5-10), and the corresponding tweak T.

Other input values, κ and IV, are fixed for a given data set; they are pre-processed during initialization, steps 1-4. Additional values may need to be calculated to take advantage of caching features of WCFB.

Initialization vector

WCFB requires a unique IV per data set for the same κ (κ is the shared secret, defined below). The uniqueness v.s. randomness condition on the IV is justified because it is only used as a value $\hat{E}_{k,k_0}(\mathsf{IV})$ in a standard CFB mode

Fig. 1. WCFB algorithm

Encryption	Decryption
1: $C_{-1} \doteq \mathsf{IV}$	
2: $k \leftarrow \mathsf{KDF}(\kappa, \mathsf{IV}, 0), \forall_{i=0}^{m} i : k_i \leftarrow \mathsf{KDF}(\kappa, \mathsf{IV}, i+1)$	
3: define $\hat{E}_{k,k_i}(\cdot) \doteq E_k(\cdot \oplus k_i), \hat{E}^{-1}_{k,k_i}(\cdot)$ is its inverse	
4: $P_m \doteq \hat{E}_{k,k_m}(T)$	
5: $\forall i \curvearrowleft$: $P'_i \leftarrow \hat{E}_{k,k_i}(P_i) \oplus P_{i+1}$	$\forall i \curvearrowright$: $P_i \leftarrow \hat{E}_{k,k_i}(C_{i-1}) \oplus C_i$
6: $\forall i : P_i \leftarrow P'_i$	$S \leftarrow \hat{E}_{k,k_m}(\bigoplus_{i=1}^{m-1} P_i)$
7: $P_{m-1} \leftarrow P_{m-1} \oplus P_0$	$P_0 \leftarrow P_0 \oplus S$
8: $S \leftarrow \hat{E}_{k,k_m}(\bigoplus_{i=1}^{m-1} P_i)$	$P_{m-1} \leftarrow P_{m-1} \oplus P_0$
9: $P_0 \leftarrow P_0 \oplus S$	$\forall i \curvearrowleft$: $P_i \leftarrow \hat{E}^{-1}_{k,k_i}(P_i \oplus P_{i+1})$
10: $\forall i \curvearrowright$: $C_i \leftarrow \hat{E}_{k,k_i}(C_{i-1}) \oplus P_i$	

with n-bit block size. IV offers an additional method to segregate data sets, in addition to using a different κ. Note that such a processing of IV adds robustness in practice because high-quality nonces are not critical for WCFB. See (8) for details.

Key set up at the step 2

WCFB mode uses a single symmetric key κ. This key is "expanded" into the main key k and $m+1$ subkeys k_i using a key derivation function $\mathsf{KDF}(\kappa, \mathsf{IV}, i)$, where the parameter i is the index of the returned subkey. This key expansion is performed once per data set and its results are expected to be cached.

WCFB depends on $\{k, k_0, \ldots, k_m\}$ being indistinguishable from selected uniformly at random, even when an attacker has access to an oracle providing $E_k(\cdot), E_k^{-1}(\cdot)$. The last clause is necessary because the discovery of k enables an oracle for $E_k(\cdot), E_k^{-1}(\cdot)$, and this must not provide additional information on other subkeys (thus, $k_i = E_k(i)$ is a poor choice for this reason).

The exact definition for how the keys are derived is left to the final instantiation of WCFB. In many cases, such as when the shared secret κ is obtained through a higher-level key exchange protocol, a KDF is already defined for that protocol. In these cases the KDF is executed more times to get the needed key material.

Alternatively, one could instantiate WCFB as a two-key mode with some κ_0 and κ_1, where $k = \kappa_0$ and $k_i = E_{\kappa_1}(i)$.

Operation count

There are $2m+1$ encryptions, plus one encryption $P_m = \hat{E}_{k,k_0}(\mathsf{IV})$, which is fixed for the data set, and which lifecycle is identical to the lifecycle of the subkeys k_i. It should be cached along with the subkeys.

The rest of operations are $\approx 5m$ XORs and assignments. Each of $\hat{E}$ includes one XOR, therefore, only $\approx 3m$ XORs are data-dependent, and only m BC cannot be fully parallelized (as is the case for CBC encryption).

Update scenario, introduced in Section 4, allows for the caching of pairs P_i, C_i that are shared between decryption of some ciphertext C to corresponding plaintext P, followed by the encryption of a similar to P plaintext, for the same T. We always can reuse P_m (Fig. 1, step 4) that corresponds to the (data-independent) T. The best case scenario represents changes to a single n-bit plaintext block. The first step in the encryption direction is identical to CBC decryption, and in this case only one of the m encryptions on line 5 will be performed with unknown plaintext. Thus, m out of $m+1$ n-bit cached ciphertexts can be reused at the step 5 of encryption in the best case. This makes sense when BC is slow to justify the lookup time.

Concrete security

The security bound is stated by (1) in Section 6. This upper bound means that in order to distinguish 512-byte WCFB with AES-128 from a random permutation with probability of 0.5 an attacker must obtain 2^{58} plaintext/ciphertexts pairs, 512 bytes each, assuming that there is no better attack on the AES-128. This is 2×2^{37} TiB, certainly less than a size of any (single) data set in the near future.

6 Security of WCFB

Theorem 1. *For any attacker A that can perform up to q queries consisting of mn-bit request/response pairs, it holds that the A's advantage to distinguish:*

- *an oracle providing WCFB encryption or decryption instantiated with a random PRP operating on a n-bit domain from*
- *an oracle providing a random tweakable PRP operating on a nm-bit domain*

has the following upper bound:

$$\mathbf{Adv}^{\pm\widetilde{\mathsf{prp}}}_{\mathit{WCFB}[\mathsf{Perm}(n)]}(q) < 0.5q^2(m+2)^2 2^{-n} \tag{1}$$

The theorem means that when we instantiate WCFB with an ideal primitive modeling a block cipher, we get the insecurity directly attributed to WCFB as specified in (1).

Other related "advantage notions" can be obtained from (1) by plugging (1) into inequalities that were proven to hold for wide encryption modes in general. For example, if WCFB is instantiated with a block cipher, we use results from (1) as follows:

$$\mathbf{Adv}^{\pm\widetilde{\mathsf{prp}}}_{\mathsf{WCFB}[E]}(t,q) < \mathbf{Adv}^{\pm\widetilde{\mathsf{prp}}}_{\mathsf{WCFB}[\mathsf{Perm}(n)]}(q) + 2\mathbf{Adv}^{\pm\mathsf{prp}}_{E}(t',q) \tag{2}$$

(2) comes from [1], where we refer the reader in the interest of saving space.

Proof. We will first show that the construction is a pseudo-random function (PRF) in the decryption direction. If the resulting plaintext P is indistinguishable from a random string, a polynomially bound attacker will have no computationally usable method to distinguish the decrypted plaintext from the output of a random function. A particular practical property that follows from the indistinguishability is that a small change anywhere in the C will very likely produce a random-looking P, a feature which narrow-block modes cannot accomplish.

We analyze the decryption steps in the subsections 6.1 – 6.2. The results from these sections are probabilities $\mathsf{Pr}_{5,6,7,8}$ and Pr_9 for respective steps of the WCFB algorithm (formulae (7) and (9)). The values bound the probability to distinguish the WCFB instantiated with a n to n bit PRF from a nm-bit PRF.

$\mathsf{AdvPRP}_{\mathcal{E}[E]} = \binom{q}{2}2^{-n}$ is the advantage to distinguish a PRF E from a PRP E using the $\mathcal{E}[E]$, proven in the Section A.1. Summing it all up, we obtain the probability to distinguish WCFB decryption from a PRP as:

$$\begin{aligned}\mathsf{Pr}_{\mathrm{WCFB}} =& \mathsf{Pr}_{5,6,7,8} + \mathsf{Pr}_9 + \mathsf{AdvPRP}_{\mathcal{E}[E]}\\ \mathsf{Pr}_{\mathrm{WCFB}} <& 0.5q^2(m+2)^2 2^{-n} \quad \forall m > 2\end{aligned}$$

$$\mathbf{Adv}^{\pm\widetilde{\mathsf{prp}}}_{\mathsf{WCFB}[\mathsf{Perm}(n)]}(q) < \mathsf{Pr}_{\mathrm{WCFB}} \tag{4}$$

□

6.1 Analysis of the CFB-decrypt and the mixing layer (steps 5-8 of decryption)

Consider a ciphertext $C = C_0||C_1||...C_m$. We write $\hat{E}_{k,k_i}(x) = E_k(x \oplus k_i)$ for the block cipher encryption applied to individual blocks C_i of C such that the subkey k_i is used for the C_i. We are decrypting C in CFB mode with $\hat{E}_{k,k_i}(\cdot)$ as an underlying block cipher. A unique IV per data set is used.

We model E as a PRF. Our task is to estimate the probability of internal collision between the terms of WCFB, a standard reasoning that identifies such a collision as the event that will make the behavior of a PRF composed of the steps 5,6,7 of the WCFB diverge from the behavior of a random PRF.

After the step 6 we have:

$$\begin{aligned}S' &\doteq \textstyle\bigoplus_{i=0}^{m-2} \hat{E}_{k,k_{i+1}}(C_i) \oplus \bigoplus_{i=1}^{m-1} C_i\\ &= (\textstyle\bigoplus_{i=1}^{m-2} \hat{E}_{k,k_{i+1}}(C_i) \oplus \bigoplus_{i=1}^{m-2} C_i) \oplus \hat{E}_{k,k_1}(C_0) \oplus C_{m-1}\\ &= (\textstyle\bigoplus_{i=1}^{m-2} \hat{E}_{k,k_{i+1}}(C_i) \oplus C_i) \oplus \hat{E}_{k,k_1}(C_0) \oplus C_{m-1} &\text{(5a)}\\ &= \textstyle\bigoplus_{i=0}^{m-2} \hat{E}_{k,k_{i+1}}(C_i) \oplus C_{m-1} \qquad \oplus \bigoplus_{i=1}^{m-2} C_i &\text{(5b)}\\ S &\doteq \hat{E}_{k,k_m}(S') &\text{(5c)}\end{aligned}$$

Observe that the terms $E_{k,k_{i+1}}(C_i) \oplus C_i$ of the sum in brackets in (5a) are equivalent in security to $E_{k,k_{i+1}}(C_i)$, per lemma 4. Intuitively, lemma 4 works here because $m-1$ C_i are not indepenedent quantities, unlike C_{m-1}. We will not pay more attention to the corresponding C_i after moving $\bigoplus_{i=1}^{m-2} C_i$ to the end, as shown in (5b).

What are the events during which the behaviour of the (5a) may differ from a PRF, mapping an mn-bit string to an n-bit string? WCFB is defined only in terms of operations on n-bit blocks. Thus, the events we are interested in will be described in terms of assignments, equality, or XORs of n-bit blocks.

Such events are two types of collisions. The internal collisions: collision between any two $\hat{E}_{k,k_{i+1}}(C_i)$ and any $\hat{E}_{k,k_{i+1}}(C_i)$ with C_{m-1} in (5a). There are also external collisions: when any two sessions or queries return the same S.

An external collision yields $S = \tilde{S}$ for some two queries. Given that the S is the ciphertext returned by $\hat{E}$, the collision means that for same subkey k_m $S' = \tilde{S}'$ with probability $1 - 1/2^n$ (for a PRF $\hat{E}$). For $k_m \neq \tilde{k}_m$ this means that $S'_i \oplus S'_j = k_m \oplus \tilde{k}_m$. We cap the probability of an external collision by the value from lemma 3.

Internal collisions are pair-wise internal collisions between m blocks in a single query.

We have proven the following lemma stating that S is a PRF:

Lemma 1. *For any attacker A that can perform up to q queries consisting of mn-bit request and n-bit response pairs, it holds that the A's advantage to distinguish the S instantiated with a random PRP operating on a n-bit domain from a random PRF has the following upper bound:*

$$\mathbf{AdvS}_{S[E]} \leq \left(\binom{q}{2} + q\binom{m}{2} \right) 2^{-n}, \textit{ where}$$

$$\mathbf{AdvS}_{S[E]} \doteq \mathsf{Pr}[E \leftarrow \mathsf{Rand}(\cdot) : A^{S[E]} = 1] - \mathsf{Pr}[S \leftarrow \mathsf{Rand}(\cdot) : A^{S} = 1]$$

S is defined in Fig. 1.

In step 7 the S is XORed into the first block P_0. In step 8 P_0 is XORed into P_{m-1}. Counting collisions between S, P_0, P_{m-1} in q queries lemma 1 extends to:

$$\boxed{\mathsf{Pr}_{5,6,7,8} < \left(\binom{3q}{2} + q\binom{m}{2} \right) 2^{-n}} \tag{7}$$

6.2 Analysis of the final (step 9) of decryption

The sequence in this step, which proceeds from higher to lower index, is the CBC encryption mode when $\hat{E}^{-1}_{k,k_i}$ is viewed as an encryption block cipher and $P_m = \hat{E}_{k,k_m}(T)$ is the IV, which is random as it is an output of a PRF. In this

section we refer to this CBC-like construction as CBC+. CBC security bounds were shown to be $m(m-1)/(2^n-m)$ in [12]. The lucid proof for the bound of $2\binom{m}{2}2^{-n}$ is in [13]. $m(m-1)/(2^n-m) < m^2/2^n$ iff $m < 2^{n/2}$. These results apply to CBC+.

Previous steps XOR certain value into the block at the index $m-1$, the first block of the CBC+ mode. The XOR into the first block is equivalent to the XOR into the IV in the CBC mode. Next we analyze the effective value of the IV used in CBC+ mode, named here as IV'. After the step 8, before the XOR with $\hat{E}_{k,k_m}(T)$:

$$\mathsf{IV}' = \hat{E}_{k,k_0}(\mathsf{IV}) \oplus S \oplus C_0 \oplus \hat{E}_{k,k_m}(T) \tag{8}$$

The main observation from (8) is that the IV of the CBC+ mode is a function of IV, T, and the entire C through S. Although an attacker controls C_0, S depends on C_0.

Given that CBC+ is IND-CPA secure, the resulting output is the PRF bound by $2\binom{m}{2}2^{-n}$ probability to distinguish this PRF from a random function. Including the $\hat{E}_{k,k_0}(\mathsf{IV})$ into the CBC+ formula,

$$\boxed{\mathsf{Pr}_9 < 2\binom{q}{2}\binom{m+1}{2}2^{-n}} \tag{9}$$

6.3 Analysis of the WCFB encryption

The last step of the encryption direction, step 10, is a standard CFB. Therefore, security claims about CFB apply to WCFB.

The IND-CPA security of CFB decryption with $\hat{E}$ was proven in [14]. When instantiated with random functions, the output of the CFB encryption is indistinguishable from a random function with probability $\binom{m}{2}2^{-n}$.

It is easy to verify from Fig. 1 that the variable S used in encryption and decryption direction of the WCFB is the same quantity (for any corresponding pair of C, P). Lemma 1 estimates the security bounds for the S as a PRF.

The steps 9,10 result in the following blocks C_0, C_1, ..., C_{m-1}, which is CFB, except for the C_0 :

$$\begin{aligned} C_0 &= \hat{E}_{k,k_0}(\mathsf{IV}) \oplus P_0 \oplus S \\ C_1 &= \hat{E}_{k,k_1}(C_0) \oplus P_1 \\ C_2 &= \hat{E}_{k,k_2}(C_1) \oplus P_2 \\ &\cdots \end{aligned}$$

The event we are looking for is a collision between any of qm n-bit blocks (whcih would have had the probability of a collision as $\binom{qm}{2}2^{-n}$), except that the block C_0 needs special attention.

Similar to (7), lemma 1 extends to $(\binom{2q}{2} + q\binom{m}{2})2^{-n}$. Together with CFB bound, we arrive at

$$\boxed{\mathsf{Pr}_{\mathsf{encr}} < \left(\binom{2q}{2} + q\binom{m}{2} + \binom{qm}{2}\right)2^{-n}} \tag{11}$$

(11) is smaller than $\mathsf{Pr}_{5,6,7,8} + \mathsf{Pr}_9$, thus we can use decryption bounds for either direction of the WCFB.

References

1. Halevi, S., Rogaway, P.: A tweakable enciphering mode. In Boneh, D., ed.: Advances in Cryptology - CRYPTO 2003. Volume 2729 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2003) 482–499
2. Halevi, S.: Eme*: Extending eme to handle arbitrary-length messages with associated data. In Canteaut, A., Viswanathan, K., eds.: Progress in Cryptology - INDOCRYPT 2004. Volume 3348 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 315–327
3. Chakraborty, D., Sarkar, P.: A new mode of encryption providing a tweakable strong pseudo-random permutation. In Robshaw, M., ed.: Fast Software Encryption. Volume 4047 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2006) 293–309
4. Halevi, S.: Invertible universal hashing and the tet encryption mode. In Menezes, A., ed.: Advances in Cryptology - CRYPTO 2007. Volume 4622 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007) 412–429
5. Sarkar, P.: Improving upon the tet mode of operation. In Nam, K.H., Rhee, G., eds.: Information Security and Cryptology - ICISC 2007. Volume 4817 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007) 180–192
6. McGrew, D., Fluhrer, S.: The security of the extended codebook (xcb) mode of operation. In Adams, C., Miri, A., Wiener, M., eds.: Selected Areas in Cryptography. Volume 4876 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007) 311–327
7. Wang, P., Feng, D., Wu, W.: Hctr: A variable-input-length enciphering mode. In Feng, D., Lin, D., Yung, M., eds.: Information Security and Cryptology. Volume 3822 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 175–188
8. Chakraborty, D., Sarkar, P.: Hch: A new tweakable enciphering scheme using the hash-encrypt-hash approach. In Barua, R., Lange, T., eds.: Progress in Cryptology - INDOCRYPT 2006. Volume 4329 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2006) 287–302
9. Ferguson, N.: Aes-cbc + elephant diffuser: A disk encryption algorithm for windows vista (2006)
10. Martin, L.: Xts: A mode of aes for encrypting hard disks. Security Privacy, IEEE **8** (2010) 68–69
11. NIAP: Protection profile for software full disk encryption version 1.1. Common Criteria Evaluation and Validation Scheme (2014)
12. Wooding, M.: New proofs for old modes. Cryptology ePrint Archive, Report 2008/121 (2008) `http://eprint.iacr.org/`.
13. Dodis, Y.: Introduction to cryptography, lecture 10. New York University (2012) `http://www.cs.nyu.edu/courses/spring12/CSCI-GA.3210-001/lect/lecture10.pdf`.

14. Alkassar, A., Geraldy, A., Pfitzmann, B., Sadeghi, A.R.: Optimized self-synchronizing mode of operation. In Matsui, M., ed.: Fast Software Encryption. Volume 2355 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2002) 78–91
15. Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. **61** (2000) 362–399
16. Mihir Bellare, P.R.: Cse207 – introduction to modern cryptography. University of San Diego (2012) `http://cseweb.ucsd.edu/users/mihir/cse207/w-prf.pdf`.
17. Kilian, J., Rogaway, P.: How to protect des against exhaustive key search (an analysis of desx). J. Cryptology **14** (2001) 17–35
18. Lucks, S.: The sum of prps is a secure prf. In: Advances in CryptologyEUROCRYPT 2000, Springer (2000) 470–484

A Building blocks

The subsections contain a review of the building blocks of the WCFB. These results are used in the prior sections of the paper.

A.1 $\mathbf{AdvPRP}_{\mathcal{E}[E]}$ for E

Consider a wide block cipher $\mathcal{E}[E]$ that is built with a block cipher E.

We would like to work with E when it is modeled as a PRF and then translate the obtained concrete security results to the same setup but with E modeled as a PRP. $\mathcal{E}$ would be a mode like the WCFB.

As the proof of the following lemma shows, we cannot simply apply the lemma 6 of [1] or Proposition 2.5 of [15] to the $\mathcal{E}$ as a black box. The following lemma and corollary properly address the nested structures such as $\mathcal{E}[E]$, or $\hat{\mathcal{E}}[\mathcal{E}[E]]$.

Lemma 2. *For any distinguisher A that queries their oracle on a q m-block inputs, it holds that*

$$\mathbf{AdvPRP}_{\mathcal{E}[E]} \leq \binom{q}{2} 2^{-n}, \textit{where}$$

$$\mathbf{AdvPRP}_{\mathcal{E}[E]} \doteq \mathsf{Pr}[E \leftarrow \mathsf{Rand}(\cdot) : A^{\mathcal{E}[E]} = 1] - \mathsf{Pr}[E \leftarrow \mathsf{Perm}(\cdot) : A^{\mathcal{E}[E]} = 1]$$

Proof. We find the upper bound of the attack first. While the size of the range and the domain of $\mathcal{E}[E]$ is nm bits, there is an effective way to exploit the limited range of E. The output values of $E(\cdot)$ for unique input will collide in the case of a random function and never in case of a permutation exactly the same way as in the setting of the switching lemma proven for CBC-MAC construction [15]. We fix a block index in $\mathcal{E}[E]$ for the block which value we will be changing from 0 to $2^n - 1$, while setting the rest of blocks to 0^n, and also fix the T. Being a permutation, the $\mathcal{E}[E]$ will take exactly 2^n nm-bit distinct values in case of E instantiated with a PRP, while we will have a collision in the 2^{nm} space after about $2^{n/2}$ steps in case of a PRF.

On the other hand, $\mathbf{AdvPRP}_{\mathcal{E}[E]}$ cannot be less than $\binom{q}{2}2^{-n}$, or $\mathcal{E}[E]$ can be used to attack E in a standard distinguishing experiment (real world E v.s. a PRP in the n-bit domain).

□

Corollary 1. *Lemma 2 holds for nested $\hat{\mathcal{E}}[\mathcal{E}[E]]$.*

Proof. The same argument as in lemma 2 can be extended to nested $\mathcal{E}$. 2^n values in the range of E still permute to 2^n values in the range of $\hat{\mathcal{E}}[\mathcal{E}[E]]$.

□

A.2 Security of $\hat{E}_{k,k_i}(x)$

We evaluate the security of $\hat{E}_{k,k_i}(x)$ for an attacker that has access to *two sessions* I and J simultaneously, corresponding to $\hat{E}_{k,k_i}(x)$ and $\hat{E}_{k,k_j}(x)$. We then show how this setup allows an attacker to distinguish $\hat{E}$ from a random function.

This setup is a natural choice for the internal structure of the WCFB, where XOR between a pair of $\hat{E}$ with different subkeys is a common operation.

In Section A.4 we considered a more common DESX-style construction $\hat{E}_{k,k_i}(x) = E_k(x \oplus k_i) \oplus k_i$, also under the setup of parallel sessions. However, lemma 5 shows that we could not find additional benefits of that more complex construction.

Lemma 3. *Given a PRP $E_k(\cdot) : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, define:*

$$\hat{E}_{k,k_i}(x) \doteq E_k(x \oplus k_i)$$

The probability to distinguish between $\hat{E}_{k,k_i}(\cdot)$ and $\hat{E}_{k,k_j}(\cdot)$ is bound by $\binom{q}{2}2^{-n}$ for q queries. $\mathcal{K}$ is the space of keys for E, $k_i \xleftarrow{\$} \{0,1\}^n$. The probability is the same when $E_k(\cdot)$ is a PRF.

Proof. The default choice in this paper is to model a block cipher as a PRF in the proofs, switching to a PRP at later stages. Here we first assume that the E_k is a PRP. Consider the domain/ranges $\forall x \in \boldsymbol{X} \doteq \{0,1\}^n : \boldsymbol{Y_i} \doteq E_k(x \oplus k_i), \boldsymbol{Y_j} \doteq E_k(x \oplus k_j)$. For a PRP $|\boldsymbol{X}| = |\boldsymbol{Y_i}| = |\boldsymbol{Y_j}| = 2^n$. Each element of $\boldsymbol{Y_i}$ can be paired with an element of $\boldsymbol{Y_j}$, so that for two subkeys k_i, k_j

$$E(a \oplus k_i) = E(b \oplus k_j) \Leftrightarrow \tag{13a}$$

$$a \oplus b = k_i \oplus k_j \tag{13b}$$

The probability of at least one collision is bound by $\binom{q}{2}2^{-n}$ for q queries. A collision event will mean the discovery of the "difference" $\delta = k_i \oplus k_j$ between the subkeys. This is sufficient to break many settings similar to left-or-right security. Knowing δ allows the distinguisher to submit $x \oplus \delta$ to one oracle, say to the oracle with k_j, which will return $\hat{E}_{k,k_j}(x \oplus \delta) = E_k(x \oplus \delta \oplus k_j) = E_k(x \oplus k_i) = \hat{E}_{k,k_i}(x)$,

the output of the oracle with the k_i for x. Then the distinguisher knows exactly the answer that the oracle with the subkey k_i will provide for the value x that it never "seen", with probability 1. An example of a settings would be to distinguish between a session with a pair of random PRPs from a session with a pair $\hat{E}_{k,k_i}$, $\hat{E}_{k,k_j}$. A collision in the latter case gives δ that can be confirmed in the next query; random PRPs will not have such a property attributed to δ.

The proof for the PRF is slightly more difficult because we need to deal with the possibility of $E_k(\cdot)$ colliding on different input. In the event of a collision in formula (13a) we cannot assume (13b). However, a challenger can verify the type of output collision by calculating the candidate δ as above and submitting a pair $t + \delta$ to one oracle, provided that there is an answer for t queried from another oracle. Let us assume that we know the $E_{k,k_i}(t)$. Assuming that $t + \delta$ is new for the oracle that uses k_j, $E_{k,k_i}(t) = E_{k,k_j}(t \oplus \delta)$ if δ is correct with confidence probability $1 - 2^{-n}$. □

A.3 Security of $E_k(x) \oplus x$

This section serves to simplify a few places in the paper by showing that dropping the "outer XOR" of the input added to the output has no consequence for the security. Intuitively, it makes sense that the addition of the plaintext to the ciphertext will not make the ciphertext less predictable.

Lemma 4. *Given a PRF $E_k(x)$, $Y(x) \doteq E_k(x) \oplus x$ is a PRF with identical security.*

Proof. We follow closely the setup of the distinguishing games given in [15], [16].

Consider that there is a distinguisher A^Y that can guess with non-negligible advantage that the output is produced by Y v.s. a random function, given a set of inputs x. Assume there is an environment set up in which A^Y interacts with an oracle Y, sending requests x, getting back $E_k(x) \oplus x$, and at some point, after obtaining sufficient number of pairs (x, y), it produces the conclusion for whether it interacts with the "real" world of Y or with a random function. All the recorded pairs (x, y) represent the only information that the distinguisher possesses to substantiate his conclusion. Such a sequence of (x, y) used for the distinguishing session is called here a *transcript*. A transcript is polynomial in size and it was obtained in polynomial time.

Observe that we can transform the set of obtained pairs by adding $\oplus x$ to the y. This is done in time linear to the size of the transcript. This transforms the transcript for Y into the transcript for E_k.

In doing so we have obtained the polynomial-size transcript for the oracle of E_k and there is a distinguisher A^Y that can distinguish it with non-negligible advantage from a transcript corresponding to a random function.

The remaining technical detail here is that because we rely on the A^Y, the key was selected for the session with Y as an oracle and not E. Note that there is a polynomial-size transcript that gives non-negligible advantage to "break" E for every key $k \in K$. Even though it is unfeasible to build the transcripts for

$|\mathcal{K}|$ keys, we know that there is a polynomial-size transcript for every key that can be selected for an oracle E. [15] defines insecurity of PRF as the maximum advantage over all distinguishers. Therefore, we don't have to produce a distinguisher for E in polynomial time, we only need to show that it exists in standard distinguishing experiments.

By the condition of the lemma, no distinguisher choosing between E and a random function should be able to correctly identify E in polynomial time, given that E is a PRF. It follows that A^Y cannot exist and the Y is a PRF with identical security.

□

A.4 Security of $E_k(x \oplus k_i) \approx E_k(x \oplus k_i) \oplus k_i$

We considered the DESX-like [17] construction, but could not see a benefit of a second subkey. In general, the construction $E_k(x \oplus k_i) \oplus k_i$ is a popular building block in security modes. We show here that its security is equivalent to the security of the $\hat{E}_{k,k_i}(x)$, see Section A.2. Note that we don't offer the oracle access to E_k, only to $\hat{E}_{k,k_i}$. We show this using distinguishing attack identical to the one in lemma 3, which determines whether we are interacting with a session for $E_k(x \oplus k_i) \oplus k_i$ or $E_k(x \oplus k_j) \oplus k_j$. Identical definitions to lemma 3 are omitted.

Lemma 5. *Given a PRP $E_k(\cdot) : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, define:*

$$\hat{E}_{k,k_i}(x) \doteq E_k(x \oplus k_i) \oplus k_i \tag{14}$$

The probability to distinguish between $\hat{E}_{k,k_i}(\cdot)$ and $\hat{E}_{k,k_j}(\cdot)$ is bound by $\binom{q}{2}2^{-n}$ for q queries, identical to the one in lemma 3.

Proof. We are presented with two encryption oracles, the one that uses key k_i and the one that uses the key k_j, referred here as oracles I and oracle J. We cannot repeat queries to I and J, but we can ask each oracle to encrypt the same value.

We rely on collisions to distinguish I and J. The collisions are found using two tables for I and J:

$$\begin{aligned} &\text{for } I : \{a,\ \ y_i(a) = E_k(a \oplus k_i) \oplus E_k((a \oplus 1) \oplus k_i)\} \\ &\text{for } J : \{b,\ \ y_j(b) = E_k(b \oplus k_j) \oplus E_k((b \oplus 1) \oplus k_j)\} \end{aligned}$$

Note how the outer keys k_i, k_j are canceled by the XOR. Name the $\delta = k_i \oplus k_j$. Notice that $\forall a, b \in [0, 2^n - 2]$ there is a 1:1 correspondence between a pair of elements y_i, y_j for some i, j. Specifically, $\forall a \oplus b = \delta \Rightarrow y_i(a) = y_j(b)$. Once the collision is found for some a, b, it can be confirmed if it is due to δ with $y_i(a \oplus 1) = y_j(b \oplus 1)$.

The sum of PRPs is a PRF, as shown in [18], thus the y_i, y_j are PRFs on $\{0,1\}^n$. We will be selecting a, b as distinct uniformly at random values.

At this point the setup is consistent with standard collision search between y_i, y_j, and is identical to the one in lemma 3, leading to the birthday bounds security. □

Note that knowing δ, we can distinguish between I and J deterministically going forward: sending $\forall x, x+\delta$ to I, J will cause the fixed value of XOR between oracle's outputs. This even works regardless of the outer subkey being different from the inner.

B Parallelism within a wide block

In this section we provide support for the design decision not to complicate WCFB with internal parallelism.

First of all, observe that within $\approx 2\times$ block cipher operations WCFB allows unlimited m-degree parallelization of the half of these operations in either encryption or decryption direction, plus its lean "mixing" layer is fully parallelizable, without any need for precomputed tables. Thus, this section is about $\approx 1\times$ block cipher operations that have data dependency due to their chaining.

Parallelism within the wide block is not needed in large number of usage scenarios.

WCFB is designed for random access within a large data set (up to $2^{n/2-\mathsf{log}(m)}$ wide blocks). An example use case is a whole disk encryption product that is implemented as a layer within operating system that "sees" blocks of mn bits and transparently encrypts and decrypts them at the request of an operating system. Let us assume that the T is a block index. In modern operating systems, on average, the block at the index T will not be accessed alone. Operating systems maintain a cache of related blocks through the read-ahead and delayed write strategies and they commonly batch together multiple Input/Output (I/O) operations. On average, some minimum of p blocks will be accessed together. Keep in mind that storage hardware is often doesn't even have capability to access/reference an individual block P; P can only be accessed/referenced within as part of a cluster of blocks instead. Similar argument carries over to database management systems. In general, the systems that are designed for high performance will likely routinely perform parallel I/O with multiple blocks at a time.

Fig.2 shows the case study to support the above claim. We have instrumented a Windows disk filter driver to obtain the listed counters for read requests from disk over the period of 30 min, starting from booting Windows from a powered down state. The user has logged in and used Internet Explorer to browse http://yahoo.com and compile source code. The system was idle for about half the time. The system is a 32-bit Windows 8.1 with 60Gb hard disk, NTFS, and 1 Gb of RAM, installed in a VirtualBox virtual machine. The virtual disk was indicated as an SSD disk in the VirtualBox settings. The sector size in all the requests is 512 bytes.

The fraction of sectors appearing in read requests for which $p < 4$ is less than 0.003, and the fraction of requests for which $p < 8$ is 0.006.

These numbers are slightly better for write requests, see Fig.3.

In serial protocols, such as decryption of a media stream, it is possible to buffer and work on p blocks at a time.

Fig. 2. Sector read access pattern on Windows 8.1. 30 min after boot, Internet Explorer browsing of http://yahoo.com

Description	Counter
Total number of requests	242093
Total number of sectors	10830072
Total sectors appearing in 4 sector groups	10803156
Total sectors appearing in 8 sector groups	10773176
Total sectors not making 4 sector groups	26916
Total sectors not making 8 sector groups	56896
Minimum number of sectors in a request	1
Maximum number of sectors in a request	4096
Average sectors in a request	44
Percentage of sectors in groups < 4 sectors to sectors in groups even to 4	0.3%
Percentage of sectors in groups < 8 sectors to sectors in groups even to 8	0.6%

Fig. 3. Sector write access pattern on Windows 8.1. 30 min after boot, C source code compilation

Description	Counter
Total number of requests	30962
Total number of sectors	3152391
Total sectors appearing in 4 sector groups	3147896
Total sectors appearing in 8 sector groups	3142312
Total sectors not making 4 sector groups	4495
Total sectors not making 8 sector groups	10079
Minimum number of sectors in a request	1
Maximum number of sectors in a request	2048
Average sectors in a request	101
Percentage of sectors in groups < 4 sectors to sectors in groups even to 4	0.2%
Percentage of sectors in groups < 8 sectors to sectors in groups even to 8	0.4%

This shows that there are sufficient number of scenarios where we can count on the minimum of p wide blocks in a request.

WCFB allows unlimited parallel access to separate wide blocks and this is the method by which the parallelism of the system can be leveraged with WCFB. Given p wide blocks, it is possible to implement an interleaved processing method by which individual blocks P_i (or C_i) are processed in parallel among p wide blocks P (or C).

Finally, it is difficult to envision an implementation that will leverage multi-core support (i.e. multithreaded functionality) within an encryption block. The issue here is that in many instantiations of wide block encryption the m is fairly small, 32 is common. Such a low value makes the overhead of asynchronous processing prohibitive. There is mainly one area where internal parallelism can realistically be used, and it is the SIMD capability of the processors. SIMD functionality refers to the parallelism within one CPU core and it is realized by the separate processing pipelines of a single CPU core. SIMD parallelism is limited (value 4 is common) and it is fixed for a CPU model; adding more CPU units to a host will not change the degree of SIMD parallelism. This shows that a mode without limits on internal parallelism has limited upside (e.g. factor of 4) and limited opportunities (e.g. limited to less than 0.3% of the data set).

Fig.2 shows that the added overhead experienced by a mode with high internal parallelism will likely be a disadvantage in this operating environment.

While we acknowledge that there are certain usage patterns in which built-in parallelism of the wide encryption mode might be beneficial, there are also cases in which such a feature only adds complexity and performance penalty.

C WCFB diagrams

Fig. 4 and 5 describe the encryption and decryption of WCFB specified in Fig. 1, respectively. On these diagrams the $E_i(\cdot)$ corresponds to $\hat{E}_{k,k_i}(\cdot)$.

Fig. 4. WCFB encryption diagram

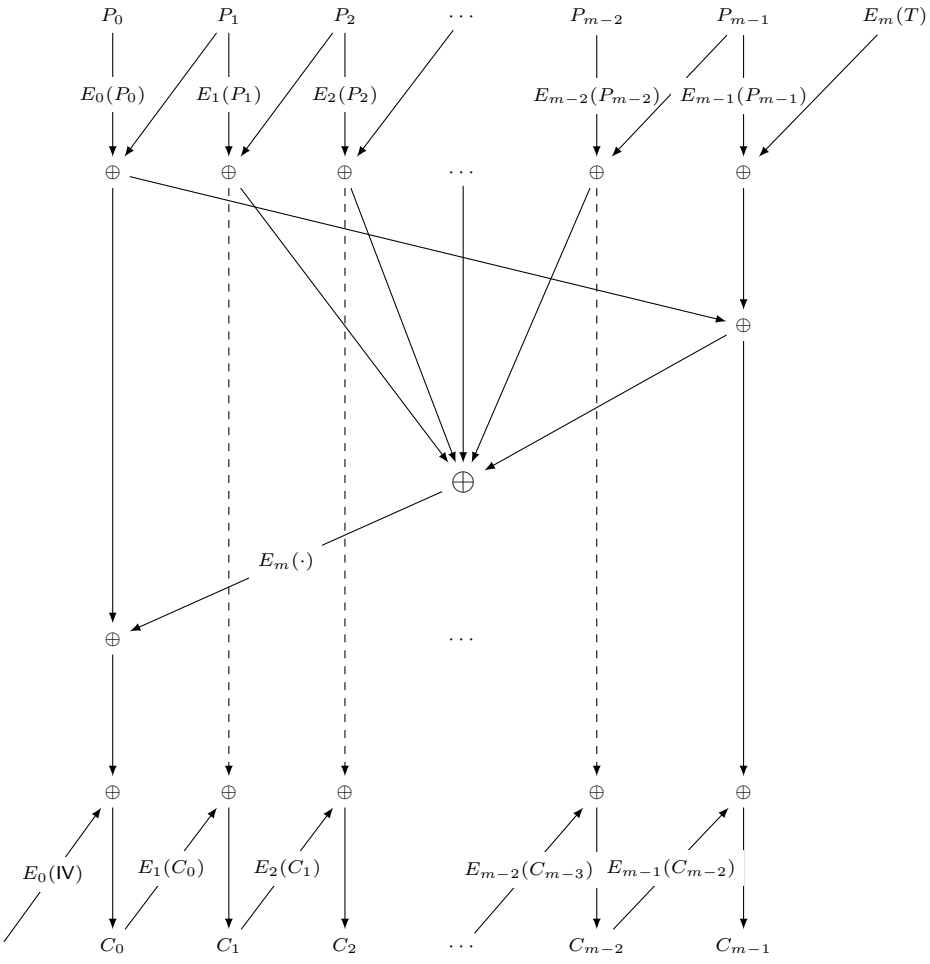
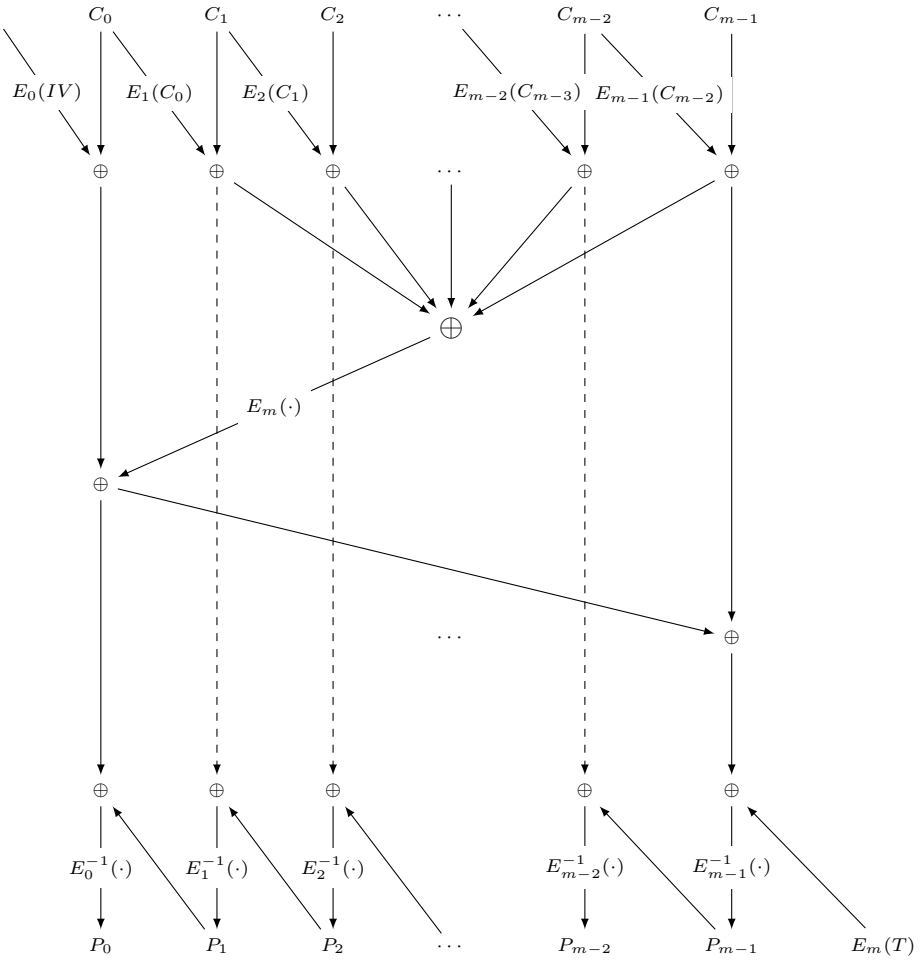


Fig. 5. WCFB decryption diagram



D Performance at 1.45 times the memcpy

We implemented the WCFB encryption algorithm with x86 SSE2 instruction set. We emulated the block cipher encryption and decryption as a one-time pad operation (XOR with a key) in order to time the overhead directly attributed to the WCFB. All WCFB operations, including the use of subkeys in $\hat{E}$, were following the WCFB specification.

We wrote the WCFB algorithm using gcc SSE2 intrinsics, a convenient way to inject SSE2 instructions into a C program. We compiled the code with gcc version 4.7.2 targeting x86_64 Linux with -O6 optimization option. The timing in this section is for a single CPU core (no multithreaded operations).

A useful metric for an algorithm like WCFB is to measure how much is it slower than a standard runtime library's memcpy operation on the same plaintext and ciphertext buffer. Specifically, the unit of measure is the time it takes the memcpy call to copy the plaintext into the ciphertext buffer. C compilers frequently generate code to implicitly use memcpy (or its equivalent) when parameters are passed by value or to implement an assignment between complex types.

We observed that the WCFB operating on a single block is $2.10 - 2.76$ times slower than the corresponding memcpy. This is a worst case scenario when there is only a single nm-bit block available for encryption. Note, however, that the code we wrote takes advantage of all the internal parallelism offered by WCFB: in our implementation the first BC pass and the mixing pass are parallelized at the factor 4 to take advantage of 4 parallel pipelines of the CPU.

Following the method described in the Section B, we timed the appropriately enhanced WCFB implementation at $1.45 - 1.97$ times the time spent by the memcpy.

For a reader who is puzzled by the fact how a performance of a more complex operation such as WCFB can possibly be that close to memcpy's, the answer can be explained by the general-purpose design of the memcpy. The implementation of a memcpy in our case, apparently, doesn't take advantage of the fact that the size of the input buffers are even to n bits and it probably doesn't use SSE2 instructions. Nevertheless, the results are satisfactory because they show that the overhead of WCFB layer puts the WCFB into the realm of operations that software developers are typically considering as negligible in their performance impact.