Full Disk Encryption: Bridging Theory and Practice

Louiza Khati[1,2], Nicky Mouha[3,4,5], and Damien Vergnaud[1]

[1] École Normale Supérieure - CNRS - Inria, Paris, France
[2] Oppida, Montigny-Le-Bretonneux, France
[3] Dept. Electrical Engineering-ESAT/COSIC, KU Leuven, Leuven and iMinds, Ghent, Belgium
[4] Project-team SECRET, Inria, France
[5] National Institute of Standards and Technology, Gaithersburg, MD, USA
`Louiza.Khati@ens.fr, nicky@mouha.be, Damien.Vergnaud@ens.fr`

Abstract. We revisit the problem of Full Disk Encryption (FDE), which refers to the encryption of each sector of a disk volume. In the context of FDE, it is assumed that there is no space to store additional data, such as an IV (Initialization Vector) or a MAC (Message Authentication Code) value. We formally define the security notions in this model against chosen-plaintext and chosen-ciphertext attacks. Then, we classify various FDE modes of operation according to their security in this setting, in the presence of various restrictions on the queries of the adversary. We will find that our approach leads to new insights for both theory and practice. Moreover, we introduce the notion of a *diversifier*, which does not require additional storage, but allows the plaintext of a particular sector to be encrypted to different ciphertexts. We show how a 2-bit diversifier can be implemented in the EagleTree simulator for solid state drives (SSDs), while decreasing the total number of Input/Output Operations Per Second (IOPS) by only 4 %.

Keywords: Disk encryption theory, full disk encryption, FDE, XTS, IEEE P1619, unique first block, diversifier, provable security.

1 Introduction

The term Full Disk Encryption (FDE) is commonly used when every sector of a disk volume is encrypted. There is typically no space to store any additional data, such as an IV or a MAC. As explained by Ferguson [11], generic solutions to store additional data will at least double the number of read and write operations, and will significantly reduce the available disk space. They also change the disk layout, which makes it extremely complicated to enable FDE on existing disks.

With this restriction, FDE cannot offer authentication, or at best "poor-man's authentication" [11], which is to hope that ciphertext changes will result in a plaintext that is random enough to make the application crash. It can also not achieve chosen-plaintext indistinguishability: when the same data is encrypted twice at the same sector index, the resulting ciphertexts will be identical.

Additional efficiency constraints may be imposed on FDE as well. For example, it can be desirable to perform encryption and/or decryption in parallel, which is not possible for inherently sequential constructions where the i-th plaintext block of a sector cannot be processed until all previous plaintext blocks are processed. If there is not enough memory available to store an entire sector, it may be required that encryption and decryption are on-line, meaning that the i-th block of a sector may only depend on the preceding blocks.

These implementation constraints impose severe restrictions on the algorithms that can be used for FDE. A general problem in the domain of FDE, however, is that the security properties of the resulting constructions are not always well-understood. On the other hand, cryptographers often complain about the absence of well-defined cryptographic goals for FDE (see e.g. Rogaway [25]), which are prerequisites to find a good-trade-off between security and efficiency.

Our Contributions. Firstly, we want to measure "how much security is left" within the constraints of FDE. In order to do so, we introduce a theoretical framework to capture that a FDE algorithm behaves as "randomly as possible" subject to different practical constraints. We consider settings where the encryption oracle can be random-up-to-repetition, random-up-to-prefix or random-up-to-block.

For each of the attack settings in the framework, we list an efficient construction that achieves security within this setting. We recall existing security results, and provide new proofs, in particular in the *unique-first-block* (ufb) setting where the Operating System (OS) or application ensures that the first n bits of the plaintext will not be repeated for a particular sector number, where n is the block size of the underlying block cipher.

Our model recalls that the modes of operation CBC (Cipher Block Chaining) and IGE (Infinite Garble Extension), even with a secret IV, do not achieve the security properties that developers often wrongly assume for these constructions. As already shown by Bellare et al. [2], CBC and IGE are not IND-CPA secure up-to-prefix. We will prove, however, that both constructions are IND-CPA secure under the ufb constraint.

Regarding chosen-ciphertext attacks, we point out that Added Redundancy Explicit Authentication (AREA) [12] is not secure when used with CBC or IGE, even when the IV is secret. The insecurity of constructions such as AREA was already shown in 2001 by Jutla [21], but has nevertheless not yet been pointed out in the context of FDE. We recall that there exist constructions that are secure in this setting, such as TC2 and TC3 [26].

Secondly, we revisit the FDE constraints from an engineering point of view. We show that it is possible to produce different ciphertexts for the same plaintext at a particular sector index, without storing additional data. Our solution applies to solid state drives (SSDs), where we show how the SSD firmware can be modified to associate a *diversifier* to every sector. This is done without modifying the data structures of the SSD, but by forcing data to be written to a particular Logical Unit Number (LUN).

For any particular sector, the diversifier value must be unique. But as we will explain later, additional requirements are necessary for performance reasons. When looking at all sectors at any particular point in time, each diversifier value should occur roughly the same number of times. Additionally, this diversifier value can typically only be a few bits long. These requirements put the diversifier in a class by itself, and not as a specific case of a random IV or a nonce (i.e. a number that is only used once).

When we benchmarked our solution in a modified EagleTree simulator [9], we found that it increases the average latency by at most 12 % for reads and 2 % for writes, and that it reduces the SSD throughput (read and write combined) by less than 4 %. This paper provides the first efficient FDE solution that can achieve indistinguishability against chosen-plaintext and chosen-ciphertext attacks.

Related Work. The problem of FDE has been researched extensively, see for example Rogaway [25] for a provable security treatment, or Fruhwirth [12] for an implementer's perspective. The formal requirements of disk encryption are often not clearly stated.

FDE is a topic that has gathered significant interest from industry and standardization, and often leads to application-specific solutions due to the special requirements of full disk encryption. Of particular interest are the elephant diffuser used in Microsoft's BitLocker [11], or IEEE P1619's XTS standard [18], which later became a NIST recommendation [10] as well.

Throughout this paper, we assume that adversary has access to the disk volume at any time. The adversary has (partial) knowledge and even control of the plaintext, and can even change the ciphertext as well. We therefore go beyond just "single point-in-time permanent offline compromise" (see e.g. [13,15]). Read and write operations are assumed to be atomic (on a sector level), so we do not consider blockwise adaptive attacks [19].

Sound key management is required to avoid that the plaintext contains the key, or any function of the key [16]. Physical access threats (e.g. cold boot, DMA, evil maid, or hot plug attacks [14, 22]) are also outside the scope of this paper.

2 Disk Encryption Methods

Data is read and written in a sector-addressable device by fixed-length units called sectors, usually 512 or 4096 bytes long. The OS can access a specific sector by its *sector number s*. We consider the case of an encrypted disk volume where data is encrypted by the OS before being stored.

We list the modes of operation that frequently appear in the context of FDE, whether it be in academic literature or in practical implementations. We also mention other modes with interesting security or efficiency properties.

ECB (Electronic Codebook). In the simplest encryption mode, the plaintext is divided into blocks of n bits, and each block is encrypted separately using an n-bit block cipher. It can readily be used for FDE, even though it is well-known that it does not provide adequate security.

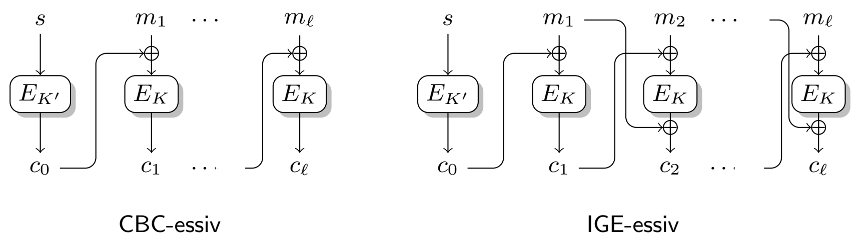

Fig. 1. Description of the CBC-essiv and IGE-essiv modes of operation.

CTR (Counter). This mode uses a counter (incremented for each block) that is encrypted and then XORed with the plaintext block to output the ciphertext block. Typically, the counter is the sector number, bit-shifted to the left over a sufficient number of bits so that the least-significant bits can represent a counter for the number of blocks in one sector. CTR mode is IND-CPA secure [4] under the assumption that the counter is a nonce. In the context of FDE, this assumption does not hold as sectors can be overwritten.

CBC. In this mode, each plaintext block is XORed with the previous ciphertext block (or an IV for the first plaintext block) before being encrypted. To achieve the IND-CPA security notion, it is well-known that the IV has to be a random value [4]. However, for FDE, a first natural idea is to use the sector number as an IV. Fruhwirth [12] proposed to use as an IV the encryption of the sector number by the block cipher keyed with an independent key (see Fig. 1).[6]

IGE. IGE was proposed by Campbell [8] as a variant of CBC mode where each block of plaintext is XORed with the *next* ciphertext block (see Fig. 1). For FDE, since the sector number is not secret, we will consider the variant where the IV is the encryption of the sector number s in which s is not XORed to the first ciphertext block. We will refer to this mode as IGE-essiv.

XTS. XTS [18] applies a tweakable block cipher to every n-bit block of a sector, where the tweak depends on the sector number and on the index of the block within the sector. It uses *ciphertext stealing* when the sector size is not a multiple of n bits, however such sectors sizes are not considered in this paper.

TC1, TC2 and TC3. These modes for tweakable block ciphers were defined by Rogaway and Zhang in [26]. The difference between these constructions is the way the tweak is used:

- In TC1, the tweak is the previous ciphertext block as in the HCBC mode [2];
- In TC2, the tweak is the concatenation of the previous ciphertext block and the previous plaintext block as in the HCBC2 mode [2];
- In TC3 mode, the tweak is the XOR of the previous ciphertext block and the previous plaintext block as in the MHCBC2 mode [23].

[6] Two distinct keys are needed: the message is encrypted with key K, and the IV is encrypted with key $K' \neq K$ (see Fig. 1), in order to avoid an attack by Rogaway [24].

WTBC (Wide Tweakable Block ciphers). In the context of FDE, the block size of a WTBC is equal to the sector size, and the sector number is used as the tweak input. From a security point of view, any change in the plaintext or ciphertext affects the entire sector. A WTBC is typically realized using smaller (tweakable) block ciphers, as for example in the EME (Encrypt-Mix-Encrypt) [17] mode.

Our goal in this paper is to analyze these constructions and to evaluate their security in different models.

3 Security Notions for FDE

In this section, we formalize several security notions for FDE. We first give a formal syntactic definition of block-cipher-based FDE.

It is assumed that the plaintext of a sector is a multiple of n which is the block cipher size. All plaintexts are ℓ blocks of n bits. m^i denotes the i-th block of the plaintext m such that $m = m^1||m^2||...||m^\ell$ where $||$ denotes concatenation of strings. IND-CPA-xx corresponds to IND-CPA up-to-block, IND-CPA up-to-prefix, IND-CPA up-to-repetition and IND-CPA.

Definition 1. *Let k, n and ℓ be three positive integers. A* (k, n, ℓ)-block-cipher-based FDE scheme *is a pair of algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *such that:*

- Enc *is the (deterministic) encryption algorithm which takes as input a key $K \in \{0,1\}^k$, a sector number $s \in \{0,1\}^n$ and a* plaintext $m \in \{0,1\}^{\ell\cdot n}$ *and outputs a* ciphertext $c \in \{0,1\}^{\ell\cdot n}$*;*
- Dec *is the (deterministic) decryption algorithm which takes as input a key $K \in \{0,1\}^k$, a sector number $s \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell\cdot n}$ and outputs a* plaintext $m \in \{0,1\}^{\ell\cdot n}$,

such that $\forall (K, s, m) \in \{0,1\}^{k+n+\ell\cdot n} : \ \mathsf{Dec}(K, s, \mathsf{Enc}(K, s, m)) = m.$

For each security notion, we define two variants: security under Chosen-Plaintext Attack (CPA) where the adversary is given access to the **Encrypt** procedure and security under Chosen-Ciphertext Attack (CCA) where the adversary is also given access to the **Decrypt** procedure. The adversary is not allowed to query the decryption of a ciphertext that was previously returned by **Encrypt** or vice versa.

Indistinguishability up-to-block. Each ciphertext block depends deterministically on the plaintext block, the sector number s and the block position in the plaintext, but behaves as "randomly as possible" subject to this constraint. The corresponding game described in Appendix B uses independent random permutations $\Pi^{(s,i)} : \{0,1\}^n \rightarrow \{0,1\}^n$ for each sector number s and each block position $i \in \{1, \ldots, \ell\}$. We specifically introduce this setting to describe the security goal of XTS, see Rogaway [25] for a formal definition (using a filter function) of what is known to leak by the XTS mode.

Indistinguishability up-to-prefix. For $i \in \{1,\ldots,\ell\}$, the i-th ciphertext block depends deterministically on the sector number s and all previous plaintext blocks at position j for $j \in \{1,\ldots,i\}$, but again behave as "randomly as possible" subject to this constraint. The corresponding game described in Appendix B uses independent random permutations $\Pi^i_m : \{0,1\}^n \rightarrow \{0,1\}^n$ for each sector number s and each plaintext prefix $m \in \{0,1\}^{t \cdot n}$ for $t \in \{1,\ldots,\ell\}$. This notion corresponds to security notion described in [2] for *on-line ciphers*.

Indistinguishability up-to-repetition. Each ciphertext block depends deterministically on the plaintext block and the sector number s, but behaves as "randomly as possible" subject to this constraint. The corresponding game described in Appendix B uses independent random permutations $\Pi^s : \{0,1\}^n \rightarrow \{0,1\}^n$ for each sector number s. It is the best achievable notion for (length-preserving) deterministic encryption [3].

Remark 1. If a construction is IND-CCA under one of these notions, it is also IND-CPA under the corresponding security notion.

Remark 2. If a construction is not IND-CPA under one of these notions, it is also not IND-CCA under the corresponding security notion.

As different ideal-world encryption oracles are used in the various security notions, it is trivially possible to distinguish between the encryption oracles. For example, for a fixed sector number and position on the plaintext, an IND-CPA up-to-block construction always returns the same ciphertext block. This construction does not reach IND-CPA up-to-prefix security, which requires indistinguishablilty up to the longest common prefix for a fixed sector number. It also does not satisfy IND-CPA up-to-repetition security, which requires indistinguishability up to repetition of the plaintext for a given sector number. Conversely, a construction that achieves IND-CPA up-to-prefix security will also not be IND-CPA up-to-block nor IND-CPA up-to-repetition using a similar reasoning.

Analysis of Existing Constructions. We now analyze the FDE modes of operation described in Sect. 2 with respect to these security notions. These results are summarized in Table 1. The properties shown in the three last lines of Table 1 are relevant implementation properties, but are not taken into account in the security proofs.

ECB mode. Unsurprisingly, ECB is not IND-CPA for any of our three security notions.

CTR mode. CTR is not IND-CPA for any of our three security notions. An adversary can simply query the encryption of $m_1 = 0^n||m^2||..||m^\ell$ and $m_2 = 1^n||m^2||..||m^\ell$ for the same sector number s (where $m^2, ..., m^\ell$ can be any n-bit blocks). The first blocks of the obtained ciphertexts c_1 and c_2 will always satisfy $c^1_1 = c^1_2 \oplus 1^n$, whereas this property holds only with probability 2^{-n} in all three random worlds.

Table 1. The security of FDE modes of operation when no diversifier is used. Here, ✓ means that there is a security proof, and ✗ means that there is an attack. Proofs of the security results can be found in Sect. 3. XTS: see [18]. TC1, TC2 and TC3 [26] are generalizations of the HCBC1 [2], HCBC2 [2] and MHCBC [23] constructions. WTBC: wide tweakable block cipher. The * symbol indicates that the property holds for some constructions, but not for others. Here, $x \geq \log_2(\ell)$.

	ECB	CTR	CBC	CBC	IGE	XTS	TC1	TC2/3	WTBC
IV $\rightarrow$	n/a	$s \ll x$	s	$E_{K'}(s)$	$E_{K'}(s)$	s	s	s	s
IND-CPA-block	✗	✗	✗	✗	✗	✓	✗	✗	✗
IND-CPA-prefix	✗	✗	✗	✗	✗	✗	✓	✓	✗
IND-CPA-repetition	✗	✗	✗	✗	✗	✗	✗	✗	✓
IND-CPA	✗	✗	✗	✗	✗	✗	✗	✗	✗
IND-CCA-block	✗	✗	✗	✗	✗	✓	✗	✗	✗
IND-CCA-prefix	✗	✗	✗	✗	✗	✗	✗	✓	✗
IND-CCA-repetition	✗	✗	✗	✗	✗	✗	✗	✗	✓
IND-CCA	✗	✗	✗	✗	✗	✗	✗	✗	✗
on-line enc./dec.	✓	✓	✓	✓	✓	✓	✓	✓	✗
parallelizable enc.	✓	✓	✗	✗	✗	✓	✗	✗	✓*
parallelizable dec.	✓	✓	✓	✓	✗	✓	✓	✗	✓*

CBC mode. The attack on the CTR mode also applies to the variant of the CBC mode where the sector number is used as an IV. In the context of FDE, this attack is known as a Saarinen's watermarking attack [27]. Bellare et al. [2] describe an attack on the CBC-based online cipher that shows that CBC-essiv is not IND-CPA up-to-prefix. This attack can be adapted to construct an adversary $\mathcal{A}$ for our three security notions:

1. $\mathcal{A}$ arbitrarily chooses $(\ell-1)$ blocks $m^2, .., m^\ell$ and a sector number s. $\mathcal{A}$ builds two plaintexts $m_1 = 0^n || m^2 || \ldots || m^\ell$ and $m_2 = 1^n || m^2 || \ldots || m^\ell$ that differ only in their first block.
2. $\mathcal{A}$ queries the **Encrypt** procedure to obtain the encryption of m_1 and m_2 on sector number s: $c_b^1 || c_b^2 || \ldots || c_b^\ell \leftarrow \mathbf{Encrypt}(s, m_b)$ for $b \in \{1, 2\}$.
3. $\mathcal{A}$ builds $m_3 = 1^n || m_3^2 || m^3 || \ldots || m^\ell$ where $m_3^2 = m^2 \oplus c_1^1 \oplus c_2^1$ and queries the **Encrypt** procedure to obtain its encryption: $c_3^1 || ... || c_3^\ell \leftarrow \mathbf{Encrypt}(s, m_3)$
4. $\mathcal{A}$ returns 0 to the **Finalize** procedure if $c_3^2 = c_1^2$ and 1 otherwise.

The equality $c_3^2 = c_1^2$ is always satisfied in the real world but holds only with probability 2^{-n} in all three random worlds.

IGE-essiv mode. The previous attack on the CBC-essiv mode can easily be adapted to show that IGE-essiv is not IND-CPA for any of our three security notions. The attack is identical except that the adversary $\mathcal{A}$ checks whether the equality $c_3^2 = c_1^2 \oplus 1^n$ holds or not.

XTS mode. As explained above, in XTS every plaintext block is encrypted separately using a tweakable block cipher, where the tweak is derived from the sector number and index of the block within the sector. As argued by Rogaway [25], XTS is IND-CPA up-to-block secure. For syntactic reasons, it is not IND-CPA up-to-prefix nor IND-CPA up-to-repetition.

TC1, TC2, TC3 modes. Rogaway and Zhang [26] showed that TC1 is IND-CPA up-to-prefix secure but not IND-CCA up-to-prefix. They also proved that TC2 and TC3 are IND-CCA up-to-prefix (and thus also IND-CPA up-to-prefix) secure. For syntactic reasons, they are not IND-CPA up-to-block nor IND-CPA up-to-repetition.

WTBC modes. Halevi and Rogaway showed that EME is IND-CCA up-to-repetition (and thus IND-CPA up-to-repetition) secure in [17]. For syntactic reasons, these modes are not IND-CPA up-to-block nor IND-CPA up-to-prefix.

4 FDE Security with Unique First Block

Because encryption in the context of FDE is deterministic and length-preserving, encrypting the same plaintext twice will always result in an identical ciphertext. The OS or application may therefore want to use a particular encoding of the plaintext, in order to ensure that the ciphertext will not be repeated. This corresponds to Bellare and Rogaway's Encode-Then-Encipher approach [5] to ensure strong privacy. One thus has to determine which encoding is sufficient to ensure security against CPA. In the context of security "up-to-block," this would require a large overhead, since encoding is then required for every block of a sector (typically 128 bits). However, for schemes that are IND-CPA "up-to-repetition" or "up-to-prefix," it is sufficient to ensure that the beginning of every message is unique. This can be done by prepending either random data or a counter, as suggested by Bellare and Rogaway [5].

In this section, we consider variants of the previous security notions with *unique first block* (ufb) and we prove that IND-CPA security for CBC-essiv and IGE-essiv under this assumption. An application that is aware of this restriction, can therefore format its input such that the first block of every sector is unique. Appendix A gives a concrete example of such an application.

Relation to Previous Security Notions. The only difference between ufb model and the previous model is that for a given sector number s, $\mathcal{A}$ cannot make two queries to encrypt plaintexts that have same first block. So if a construction is secure under a security notion described in Sect. 3, it is still the case in this model. Furthermore, the IND-CPA up-to-prefix, IND-CPA up-to-repetition and IND-CPA security notions become equivalent. This is easy to see: if the first block of plaintext is not repeated, then this is sufficient to ensure that the plaintext prefix or the entire plaintext is will not be repeated either.

Security Results. In this paragraph, we analyze the FDE modes of operation described in Sect. 2 with respect to these security notions for unique first block. These results are summarized in Table 2. For the security proofs of Theorems 1

Table 2. The security of FDE modes of operation when no diversifier is used, but the first plaintext block unique for any given sector. Here, ✓ means that there is a security proof, and ✗ means that there is an attack.

	ECB	CTR	CBC	CBC	IGE	XTS	TC1	TC2/3	WTBC
IV $\rightarrow$	n/a	$s \ll x$	s	$E_{K'}(s)$	$E_{K'}(s)$	s	s	s	s
IND-CPA-block	✗	✗	✗	✗	✗	✓	✗	✗	✗
IND-CPA-prefix	✗	✗	✗	✓	✓	✗	✓	✓	✓
IND-CPA-repetition	✗	✗	✗	✓	✓	✗	✓	✓	✓
IND-CPA	✗	✗	✗	✓	✓	✗	✓	✓	✓
IND-CCA-block	✗	✗	✗	✗	✗	✓	✗	✗	✗
IND-CCA-prefix	✗	✗	✗	✗	✗	✗	✗	✓	✓
IND-CCA-repetition	✗	✗	✗	✗	✗	✗	✗	✗	✓
IND-CCA	✗	✗	✗	✗	✗	✗	✗	✗	✓
on-line enc./dec.	✓	✓	✓	✓	✓	✓	✓	✓	✗
parallelizable enc.	✓	✓	✗	✗	✗	✓	✗	✗	✓*
parallelizable dec.	✓	✓	✓	✓	✗	✓	✓	✗	✓*

and 2, we used the code-based game playing framework [7]. The proofs can be found in Appendix D.

ECB and CTR. The attacks described in Sect. 3 do not make queries with the same first block and the same sector number, and therefore still apply.

CBC-essiv. For this mode, in the attack of Sect. 3, $\mathcal{A}$ makes forbidden queries with the same first block. The following theorem states that CBC-essiv achieves IND-CPA-ufb security if the underlying block cipher E is a Pseudo-Random Function (PRF) [6].

Theorem 1 [**The IND-CPA-ufb Security of CBC-essiv**]
Let $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher. Let $\mathcal{A}$ be an IND-CPA-ufb adversary against the FDE scheme obtained from the CBC-essiv mode on E such that $\mathcal{A}$ runs in time t and makes at most q queries to the ***Encrypt*** *procedure. There exists an adversary $\mathcal{B}$ (attacking the PRF security of E) such that:*

$$\mathbf{Adv}^{ind-cpa-ufb}_{cbc-essiv}(\mathcal{A}) \leq 8 \cdot \mathbf{Adv}^{prf}_{E}(\mathcal{B}) + \frac{q^2(\ell+1)^2}{2^{n-1}}$$

where $\mathcal{B}$ runs in time at most $t' = t + O(q + nq(\ell+1))$ and makes at most $q' = q(\ell+1)$ queries to its oracle.

The following attack shows that it does not achieve IND-CCA-ufb up-to-prefix security:

1. $\mathcal{A}$ chooses a plaintext $m_1 = m_1^1||m_1^2||..||m_1^\ell \in \{0,1\}^{\ell \cdot n}$ and a sector number s and queries the **Encrypt** procedure with (s, m_1) to obtain $c_1^1||c_1^2||..||c_1^\ell$.
2. $\mathcal{A}$ builds a ciphertext $c_2 = c_2^1||c_2^2||..||c_2^\ell$ with $c_2^1 = c_1^2$, $c_2^2 = c_1^3$ and arbitrary $c_2^i \in \{0,1\}^n$ for $i \in \{3, \ldots, \ell\}$ and query the **Decrypt** procedure with (s, c_2) to obtain a plaintext $m_2 = m_2^1||m_2^2||..||m_2^\ell$.

3. $\mathcal{A}$ outputs 0 if $m_2^2 = m_1^3$ and 1 otherwise.

The equality $m_2^2 = m_1^3$ is always satisfied in the real world but this property holds only with probability 2^{-n} in the random world.

IGE-essiv. The following theorem states that the mode IGE-essiv achieves IND-CPA-ufb security:

Theorem 2 **[The IND-CPA-ufb Security of IGE-essiv]**
*Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let $\mathcal{A}$ be an IND-CPA-ufb adversary against the FDE scheme obtained from the IGE-essiv mode on E such that $\mathcal{A}$ runs in time t and makes at most q queries to the **Encrypt** procedure. There exists an adversary $\mathcal{B}$ (attacking the PRF security of E) such that:*

$$\mathbf{Adv}_{ige-essiv}^{ind-cpa-ufb}(\mathcal{A}) \leq 8 \cdot \mathbf{Adv}_{E}^{prf}(\mathcal{B}) + \frac{q^2(\ell+1)^2}{2^{n-1}}$$

where $\mathcal{B}$ runs in time at most $t' = t + O(q + nq(\ell+1))$ and makes at most $q' = q(\ell+1)$ queries to its oracle.

The following attack (inspired by Rohatgi [20]) shows that IGE-essiv does not achieve IND-CCA-ufb up-to-prefix security:

1. $\mathcal{A}$ chooses a plaintext $m_1 = m_1^1||m_1^2||..||m_1^\ell \in \{0,1\}^{\ell \cdot n}$ and a sector number s and queries the **Encrypt** procedure with (s, m_1) to obtain $c_1^1||c_1^2||..||c_1^\ell$.
2. $\mathcal{A}$ builds a ciphertext $c_2 = c_2^1||c_2^2||..||c_2^\ell$ with $c_2^1 = c_1^1$, $c_2^2 = m_1^2 \oplus c_1^3 \oplus m_1^1$ and arbitrary $c_2^i \in \{0,1\}^n$ for $i \in \{3, \ldots, \ell\}$ and query the **Decrypt** procedure with (s, c_2) to obtain a plaintext $m_2 = m_2^1||m_2^2||..||m_2^\ell$.
3. $\mathcal{A}$ outputs 0 if $m_2^2 = m_1^3 \oplus c_1^2 \oplus c_1^1$ and 1 otherwise.

The equality $m_2^2 = m_1^3 \oplus c_1^2 \oplus c_1^1$ is always satisfied in the real world but this property holds only with probability 2^{-n} in the random world.

The TC1, TC2, TC3 and WTBC constructions become IND-CPA with the ufb restriction because TC1/2/3 were IND-CPA up-to-prefix and WTBC constructions were IND-CPA up-to-repetition as explained in Sect. 3.

5 FDE Security with a Diversifier

Typically, IND-CPA cannot be reached for FDE, as the deterministic nature of FDE means that identical plaintexts will result in identical ciphertexts. We worked around this problem in the previous section by imposing a restriction on the plaintext: the first plaintext block must be unique.

Now, we introduce another way to achieve IND-CPA, without imposing restrictions on the plaintext, but still without storing additional data. Instead, we will use a diversifier j, which will be associated to every sector.

To stay within the constraints of FDE, it should somehow be possible to assign a diversifier to every sector without using additional storage. Possible candidates in the particular case of SSDs will be considered in Sect. 6. For now, it is enough to consider that for each encryption, a diversifier is picked among $\{0,1\}^d$, in such a way this diversifier is never repeated for a particular sector. Then two identical plaintexts with the same sector number will have different ciphertexts, a property that could previously not be achieved within the context of FDE.

The combination of the sector number s and the diversifier j is used instead of the sector number in FDE constructions. The combination proposed is simply the concatenation between these two values $s||j$ such as $s \in \{0,1\}^\sigma$, $j \in \{0,1\}^d$ and $n = d + \sigma$.

For the analysis in this section, it suffices that the diversifier is never repeated for a particular sector. As such, the security analysis is the same as if the diversifier were a nonce. However, we will explain in Sect. 6 that efficiency reasons require that at any particular point in time, all diversifier values should occur roughly the same number of times, and that the diversifier must be a rather short value, typically only a few bits.

Security Results. IND-CPA up-to-repetition becomes equivalent to IND-CPA security: the only difference between these notions is that if $\mathcal{A}$ asks to encrypt twice the same query (s,m) the answer will be the same ciphertext but these queries are not allowed any more under the diversifier model. Moreover in IND-CCA game, the adversary is not allowed to query the decryption of a ciphertext what was previously encrypted, or vice versa. It can therefore be seen that IND-CCA up-to-repetition becomes equivalent to IND-CCA. Similarly, since the adversary $\mathcal{A}$ is not allowed to encrypt twice with the same pair $s||j$, the IND-CPA up-to-block property is also equivalent to the other IND-CPA security notions.

Table 3 summarizes the security properties achieved by the FDE modes of operation when used with a diversifier. The IND-CPA attacks of Sect. 3 still carry over to ECB and CBC with a sector-number IV. However CTR mode becomes secure as the counter value is not repeated [4]. The IND-CPA security of XTS, TC1, TC2, TC3 and WTBC follows from the fact that the tweak is not reused. For CBC-essiv and IGE-essiv, the IND-CPA security follows from the proof of Sect. 4: now the first block may be reused, but the IV is unique.

Let us explain the attacks under IND-CCA in Table 3:

- This following attack shows that CTR is not IND-CCA-xx: $\mathcal{A}$ encrypts $(s||j, m)$ with m any plaintext and any $s||j$ and receives c then $\mathcal{A}$ decrypts $(s||j, c')$ where $c' = c \oplus 0^{n-1}1$. $\mathcal{A}$. Then, $m'^1 = m^1 \oplus 0^{n-1}1$ is always satisfied in the real world, but holds only with probability 2^{-n} in the ideal world.
- CBC-essiv and IGE-essiv are not secure: the attacks of Sect. 4 still apply, as they did not perform two encryptions with the same sector number.
- For syntactical reasons, an encryption scheme can only be IND-CCA up-to-block, IND-CCA up-to-prefix, or IND-CCA up-to-repetition. XTS is only IND-CCA up-to-block, TC2 and TC3 are only IND-CCA up-to-prefix and WBTC are only IND-CCA up-to-repetition.

As shown in Table 3, the diversifier shows how to reach IND-CPA security for most commonly-used FDE encryption modes. It also succeeds at providing IND-

CCA security for WTBC constructions, which not achievable in a "classical" FDE model.

Table 3. The security of FDE modes of operation when a diversifier is used. Here, ✓ means that there is a security proof, and ✗ means that there is an attack.

	ECB	CTR	CBC	CBC	IGE	XTS	TC1	TC2/3	WTBC
IV $\rightarrow$	n/a	$s\|j \ll x$	$s\|j$	$E_{K'}(s\|j)$	$E_{K'}(s\|j)$	$s\|j$	$s\|j$	$s\|j$	$s\|j$
IND-CPA-block	✗	✓	✗	✓	✓	✓	✓	✓	✓
IND-CPA-prefix	✗	✓	✗	✓	✓	✓	✓	✓	✓
IND-CPA-repetition	✗	✓	✗	✓	✓	✓	✓	✓	✓
IND-CPA	✗	✓	✗	✓	✓	✓	✓	✓	✓
IND-CCA-block	✗	✗	✗	✗	✗	✓	✗	✗	✗
IND-CCA-prefix	✗	✗	✗	✗	✗	✗	✗	✓	✗
IND-CCA-repetition	✗	✗	✗	✗	✗	✗	✗	✗	✓
IND-CCA	✗	✗	✗	✗	✗	✗	✗	✗	✓
on-line enc./dec.	✓	✓	✓	✓	✓	✓	✓	✓	✗
parallelizable enc.	✓	✗	✓	✗	✗	✓	✗	✗	✓*
parallelizable dec.	✓	✓	✓	✓	✗	✓	✓	✗	✓*

6 Solid State Drive

We now explain the basics of SSD storage, so that we can explain how to modify only the firmware of SSDs to associate a *diversifier* to every sector. This diversifier allows us to encrypt the same plaintext in distinct ways for the same sector number.

SSDs are flash memory devices that are gradually replacing the magnetic Hard Disk Drive (HDD) due to their reliability and performance. Just like HDDs, they are sector-addressable devices. They are indexed by a sector number that is also known as a Logical Block Address (LBA). This ensures that the physical details of the storage device are not exposed to the OS, but are managed by the firmware of the storage device.

For SSDs, the Flash Translation Layer (FTL) stores the mapping between LBAs and Physical Block Addresses (PBAs). This FTL is necessary to ensure an even distribution of writes to every sector number (*wear leveling*) and to invalidate blocks so that they can later be recycled (*garbage collection*). This FTL is necessary due to the physical constraints of flash storage: any physical address can only be written a limited number of times, and rewriting individual sectors is not possible: invalidated sectors can only be recovered in multiples of the *erase block size*.

SSD Components. The flash memory of an SSD is hierarchically organized as a set of flash chips called packages, which are further divided in dies, planes, blocks[7] and pages. Every page consists of one or more sectors, and is the smallest unit that can be written. The smallest unit that can be erased is a block. Invalidated blocks must be erased before writing, and the number of erasures and writes to every block is limited for flash storage.

To abstract away the notion of packages, dies and planes, the Open NAND Flash Interface (ONFI) standard introduces the notion of Logical Unit Number (LUN) as the minimum granularity of parallelism for flash storage. As operations can be issued to several planes in parallel, a LUN corresponds to a plane.

Introducing a Diversifier. In the context of SSD storage, we will show how to associate a *diversifier* to every LBA. This diversifier will not be stored, and will not modify the data structures of the SSD. Instead, the diversifier will impose an additional restriction on the FTL, meaning that the diversifier determines which PBAs can store the data corresponding to a particular LBA.

The intuition is that if the diversifier is selected "randomly" for every write operation, the data will be spread out evenly over the SSD, and the SSD performance should not be affected too much. We will verify this by implementing and benchmarking our proposed solution in Sect. 6.

When a write command is issued, there are various ways to specify a diversifier value for a given LBA. We will prefer to transmit this information in one operation. As such, we do not only avoid the performance drawbacks of issuing several operations to write one sector, but we also do not need to worry about inconsistent states when operations are lost, modified, or reordered.

In particular, we propose to send the diversifier along with the sector data as part of a "fat" sector that is already supported in SATA (Serial ATA) interface for storage devices. This diversifier value will be returned to the OS when a fat sector read command is issued.

In an attempt to make the diversifier as long as possible, we may want to consider the optimal (yet completely unrealistic) scenario: the diversifier uniquely specifies the physical page to which the data must be written. But even then, the size of the diversfier typically be very short: e.g., only 24 bits in case of an 128 GB SSD with 8 kB pages. We will, in fact, choose the diversifier to be much shorter, so that a practical implementation is possible that minimally modifies the SSD firmware. This diversifier cannot be selected at random, as it would be too short to avoid repetitions for a particular sector. But it can also not be a counter, as all diversifier values should be used roughly an equal number of times over all sectors to spread the writes over the entire disk layout.

In our solution, we want to avoid that the SSD needs to send data back to the host for decryption and re-encryption under a different diversifier, as this would affect the SSD performance quite drastically. Therefore, wear leveling and garbage collection operations may not change the diversifier. A straightforward solution is then to make the diversifier correspond to the LUN (or a set of LUNs), which is what we will implement and benchmark in the following section.

[7] These blocks should not be confused with the blocks of the block cipher, nor with the "block" (actually "sector") in the term Logical Block Address (LBA).

Table 4. EagleTree Benchmarks for various diversifier sizes.

	diversifier size (bits)			
	0	1	2	3
read latency (µs)	28.292	28.862	31.619	40.640
write latency (µs)	32.009	32.070	32.470	34.560
read throughput (IOPS)	20860	20493	19050	15634
write throughput (IOPS)	31240	31181	30797	28935
garbage collector reads	1284043	1295530	1356043	1489623
garbage collectorwrites	1284043	1295530	1356041	1489622
erasures	18569	18765	19661	21634

The OS can freely select the diversifier values, however they may not be repeated for any particular sector, and all diversifier values should be used roughly the same number of times. In Appendix A, we give a concrete example of an application where the OS implements such diversifier. More specifically, we show how the OS can ensure randomness and uniqueness even for very short diversifier values.

EagleTree Benchmarks. In order to confirm that the concept of a diversifier is not just feasible but also efficient to implement, we implemented this feature in the EagleTree SSD simulator [9] and performed various benchmarks. An overview of the source code modifications can be found in Appendix C.

The device that is simulated consists of eight packages, each containing four dies of 256 blocks. Each block consists of 128 pages of 4096 bytes. EagleTree currently does not support multiple planes per die. The page read, page write, bus ctrl, bus data and block erase delays are 5 µs, 20 µs, 1 µs, 10 µs and 60 µs respectively. We assume that any latencies incurred by the OS (including sector encryption and decryption) are negligible with respect to these numbers. The SSD has an overprovisioning factor of 0.7.

We simulate an SSD configured with DFTL and the greedy garbage collection policy. The base benchmarks use a simple block scheduler that assigns the next write to whichever package is free. We then compare this performance to various choices of the diversifier value, which determines the package for every write.

The workload used in our benchmarks, is the same as the example in EagleTree's `demo.cpp` file: first a large write is made to the entire logical address space. This write is performed in random order, but without writing to the same address twice. Once this large write is finished, two threads are started up: one performs random reads, and the other performs random writes in the address space. After three million I/O operations, the simulation is stopped.

The benchmark results are shown in Table 4. They suggest not to choose the number of diversifier values to be equal to the number of packages, as the impact on performance is quite significant. Compared to the benchmarks without diversifier, the throughput of the reads and writes drop by 25 % and 7 % respectively.

The average latency increases by 44 % for reads, and 8 % for writes. In this setup, the reads and writes of each garbage collection operation are restricted to one package, and this affects performance quite significantly. Reads suffer more than writes; this is mainly because there is no significant drop in performance for the initial write operations to fill up the SSD.

To avoid the large impact on performance, we must therefore choose the number of diversifier values to be smaller than the number of packages. When the diversifier is two bits, our simulations show an increase of the average latency of 12 % for reads and 1 % for writes, and a reduction of throughput of 9 % for reads and 1 % for writes. The total throughput reduction (reads and writes combined) is at most 4 %. For a diversifier of one bit, latency and throughput are affected by less than 2 %. We also looked into the number of garbage collection and the number of erasures. They worsen by less than 6 % for a diversifier of two bits, but by 16 % for a diversifier of three bits.

7 Conclusion

We presented a theoretical framework for disk encryption and we analyzed several existing constructions against chosen-plaintext and chosen-ciphertext attacks, under different notions of the ideal-world encryption oracle: up-to-repetition, up-to-prefix, or up-to-block.

Using this model, we recalled that IGE-essiv does not have chosen-ciphertext-security under any of the notions that we consider, which shows that the AREA construction proposed by Fruhwirth [12] is insecure. Nevertheless, we proved that IGE-essiv and even CBC-essiv can provide security under chosen-plaintext attacks, under the assumption that the first block of a plaintext is never repeated for the same sector number.

We also revisited FDE from an engineering perspective, and showed how to modify the firmware of a solid-state drive to associate a short "diversifier" to every sector-plaintext pair (s, m). This diversifier makes it possible to encrypt the same plaintext into different ciphertexts, something that was previously impossible without additional storage.

Acknowledgments. Nicky Mouha is supported by a Postdoctoral Fellowship from the Flemish Research Foundation (FWO-Vlaanderen), by a JuMo grant from KU Leuven (JuMo/14/48CF), and by FWO travel grant 12F9714N. Certain algorithms and commercial products are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by NIST, nor does it imply that the algorithms or products identified are necessarily the best available for the purpose. Damien Vergnaud is supported in part by the French ANR JCJC ROMAnTIC project (ANR-12-JS02-0004).

We thank Matias Bjørling, Luc Bouganim, Niv Dayan and Javier Gonzalez for their useful comments and suggestions on SSD technology.

References

1. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: Online ciphers and the hash-CBC construction. In: CRYPTO 2001. LNCS, vol. 2139, pp. 292–309. Springer (2001)
2. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: On-line ciphers and the hash-CBC constructions. Journal of Cryptology 25(4), 640–679 (Oct 2012)
3. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer (2007)
4. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS. pp. 394–403. IEEE Computer Society Press (1997)
5. Bellare, M., Rogaway, P.: Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In: ASIACRYPT 2000. LNCS, vol. 1976, pp. 317–330. Springer (2000)
6. Bellare, M., Rogaway, P.: Introduction to Modern Cryptography. In: UCSD CSE 207 Course Notes. p. 207 (2005)
7. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer (May / Jun 2006)
8. Campbell, C.M.: Design and Specification of Cryptographic Capabilities. IEEE Communications Society Magazine 16(6), 15–19 (November 1978)
9. Dayan, N., Svendsen, M.K., Bjørling, M., Bonnet, P., Bouganim, L.: EagleTree: Exploring the Design Space of SSD-Based Algorithms. PVLDB 6(12), 1290–1293 (2013)
10. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices. NIST SP 800-38E (2010)
11. Ferguson, N.: AES-CBC + Elephant diffuser: A Disk Encryption Algorithm for Windows Vista. `http://www.microsoft.com/en-us/download/details.aspx?id=13866` (2006)
12. Fruhwirth, C.: New methods in hard disk encryption. Master's thesis, Vienna University of Technology (2005)
13. Gjøsteen, K.: Security notions for disk encryption. In: ESORICS 2005. LNCS, vol. 3679, pp. 455–474. Springer (2005)
14. Götzfried, J., Müller, T.: Analysing Android's Full Disk Encryption Feature. JoWUA 5(1), 84–100 (2014)
15. Halcrow, M., Savagaonkar, U., Ts'o, T., Muslukhov, I.: EXT4 Encryption Design Document (public version). Google Technical Report (2015)
16. Halevi, S.: Re: Lrw key derivation (formerly pink-herring). IEEE P1619 Mailing List (May 2006)
17. Halevi, S., Rogaway, P.: A parallelizable enciphering mode. In: CT-RSA 2004. LNCS, vol. 2964, pp. 292–304. Springer (2004)
18. IEEE: IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices. IEEE Std 1619-2007 pp. 1–32 (2008)
19. Joux, A., Martinet, G., Valette, F.: Blockwise-adaptive attackers: Revisiting the (in)security of some provably secure encryption models: CBC, GEM, IACBC. In: CRYPTO 2002. LNCS, vol. 2442, pp. 17–30. Springer (2002)
20. Jutla, C.: Attack on Free-MAC (2000), `https://groups.google.com/d/msg/sci.crypt/4bkzm_n7UGA/5cDwfju6evUJ`

21. Jutla, C.S.: Encryption modes with almost free message integrity. In: EUROCRYPT 2001. LNCS, vol. 2045, pp. 529–544. Springer (2001)
22. Müller, T., Freiling, F.C.: A Systematic Assessment of the Security of Full Disk Encryption. IEEE Trans. Dependable Sec. Comput. 12(5), 491–503 (2015)
23. Nandi, M.: Two new efficient CCA-secure online ciphers: MHCBC and MCBC. In: INDOCRYPT 2008. LNCS, vol. 5365, pp. 350–362. Springer (2008)
24. Rogaway, P.: Nonce-based symmetric encryption. In: FSE 2004. LNCS, vol. 3017, pp. 348–359. Springer (2004)
25. Rogaway, P.: Evaluation of Some Blockcipher Modes of Operation. Tech. rep., CRYPTREC Investigation Report (2011)
26. Rogaway, P., Zhang, H.: Online ciphers from tweakable blockciphers. In: CT-RSA 2011. LNCS, vol. 6558, pp. 237–249. Springer (2011)
27. Saarinen, M.J.O.: Encrypted watermarks and linux laptop security. In: WISA 04. LNCS, vol. 3325, pp. 27–38. Springer (2004)

A Case Studies

We now present three case studies for FDE. They apply the theoretical framework developed in this paper, in order to obtain novel practical solutions that advance the state of the art. For each of the case studies, we explain the advantages of our solutions over existing techniques. The examples are rather concrete for the sake of clarity, but the practical relevance of our results is not limited to these particular examples.

A.1 On-Line Ciphers for FDE

Case Description. We consider an FDE application for which both encryption and decryption must be on-line. Off-line modes of operation cannot be implemented due to a practical restriction: the encryption and decryption must be performed by a hardware security module (HSM) that does not have enough memory to store an entire sector. With this consideration in mind, what FDE security notions can be achieved by this application?

Solution. As encryption and decryption must be on-line, we know that two plaintexts with a common prefix will result in ciphertexts that have a common prefix as well. In the "up-to-prefix" setting, Table 1 explains that neither the commonly-used CBC (e.g. in Microsoft's BitLocker [11]) nor its "improved" IGE variant (see e.g. Fruhwirth [12]) provide chosen-plaintext or chosen-ciphertext security. However, these security properties can be achieved by TC2 and TC3.

Advantages. We are unaware of any applications of TC2 and TC3 in FDE. However, when encryption and decryption must be on-line, these constructions provide up-to-prefix chosen-ciphertext security in a setting where the commonly-used CBC and the often-proposed IGE are insecure.

A.2 FDE-Aware Database Applications

Case Description. After intensive benchmarking, a database application was found to achieve an insufficiently high throughput when encryption is enabled. Without encryption, the performance of the database application is within acceptable limits. To avoid the security risks of storing data without encryption, a decision was made to use a hardware-based FDE based on CBC-essiv.

Database applications are often already aware of the sector size to align read and write operations to sector boundaries,[8] and these operations already contain some wastage.[9] This is also the case for the application that we consider, which additionally also knows whether FDE is enabled for a particular disk volume. For the given application, at least eight bytes of padding are added to the end of every

[8] `https://blogs.msdn.microsoft.com/psssql/2011/01/13/sql-server-new-drives-use-4k-sector-size/`
[9] `https://docs.oracle.com/cd/E18283_01/server.112/e17120/onlineredo002.htm`

record in order to ensure sector alignment. Is it possible for this application to achieve indistinguishability under chosen plaintexts, which implies that identical sector plaintexts will result in distinct ciphertexts?

Solution. CBC-essiv is a common solution for FDE, but unfortunately Table 1 explains that it does not achieve any of the IND-CPA notions that are defined in this paper. However, Table 2 shows that IND-CPA can be achieved for CBC-essiv if the first plaintext block for every sector is unique. The application can decide to reduce the padding at the end of every record by eight bytes, and instead prepend an eight-byte time stamp. Assuming that this time stamp will not repeat, the first plaintext block of every sector is unique, and indistinguishability under chosen plaintext attacks holds. Of course, the application should detect whether FDE using CBC-essiv is enabled for a particular disk, and return an error if this is not the case.

Advantages. To the best of our knowledge, existing solutions would encrypt at the application level, at the disk level using FDE, or would use a combination of both. The FDE is unaware of the application, as it performs encryption at the sector level. But what if the application is aware of the FDE? If that is the case, we explain how the application can generate plaintexts of a particular structure in order to obtain a stronger notion of FDE security, but without performing any encryption at the application level.

A.3 On-the-Fly Firmware Updates

Case Description. A medical monitoring system continuously measures the patient's vital signs. The device has an Internet connection, which it uses to transmit its measurements, as well as to install security-critical on-the-fly firmware updates. The firmware is stored in off-chip Flash memory. Hackers could easily read out previous versions of the firmware, as they were stored without encryption.

To prevent such attacks, the company has decided to use FDE because there is no space to store additional data such as IVs, nonces or tags. However, the company identifies a problem with FDE: if hackers know which sectors in the Flash memory are modified by a security update, they can compare these sectors with older firmware. These hackers can then sniff around in the same part of the code to quickly rediscover the security vulnerability, and possibly exploit it before the devices in the field are updated. How can this problem be avoided?

Solution. It seems that the company wants to represent sectors with identical plaintexts by different ciphertexts, in order to hide which sectors are affected by a firmware update. Classical FDE does not allow this, as there is no space to store additional data such as an IV or nonce. However, Flash-based storage allows us to achieve security against chosen plaintexts, by modifying the Flash controller to introduce a diversifier. Depending on the security requirements against chosen ciphertexts, Table 3 proposes various solutions.

Initially, the diversifier will be the most significant d bits of the hash of the sector number, where d is the size of the diversifier in bits. Every firmware update will have a version number, which will be XORed to the diversifier of every sector. Clearly, the number of firmware updates must be less than the number of diversifier values. Under this restriction, all diversifiers will be unique for a particular sector.

Note that the diversifier cannot be a counter, as this would result in a significant degradation of performance. But it can also not be chosen at random, as this will likely result in collisions due to the small size of the diversifier.

Advantages. An alternative solution may be to randomize all the instructions for every firmware update, but this technique is very error-prone and it will be difficult (if not impossible) to ensure that the device remains in a consistent state, in particular if power is lost during an on-the-fly firmware update. Our solution avoids this: for sectors that are unaffected by the update, the ciphertext changes but the plaintext remains the same.

Changing the FDE key is not an option for an on-the-fly update: while the device is being updated, there is no non-volatile memory available to store the mapping between encryption keys and sector numbers. However, changing the FDE key is possible during scheduled maintenance, when updates do not need to be performed on the fly. The firmware update counter can then be reset. Therefore, as long as the number of on-the-fly updates before scheduled maintenance is less than d, the small size of the diversifier does not present a problem.

B Security Game Definitions

Indistinguishability up-to-block. The first security game (described in Fig. 2) defines *indistinguishability up-to-block.* This notion formalizes that each ciphertext block depends deterministically on the plaintext block, the sector number s and the block position in the plaintext, but behaves as "randomly as possible" subject to this constraint. The game uses independent random permutations $\Pi^{(s,i)} : \{0,1\}^n \to \{0,1\}^n$ for each sector number s and each block position $i \in \{1, \ldots, \ell\}$. We specifically introduce this setting to describe the security goal of XTS, see Rogaway [25] for a formal definition (using a filter function) of what is known to leak by the XTS mode.

Indistinguishability up-to-prefix. The second security game (described in Fig. 3) defines *indistinguishability up-to-prefix.* This notion formalizes that for $i \in \{1, \ldots, \ell\}$, the i-th ciphertext block depends deterministically on the sector number s and all previous plaintext blocks at position j for $j \in \{1, \ldots, i\}$, but again behave as "randomly as possible" subject to this constraint. The game uses independent random permutations $\Pi^i_m : \{0,1\}^n \to \{0,1\}^n$ for each sector number s and each plaintext prefix $m \in \{0,1\}^{t \cdot n}$ for $t \in \{1, \ldots, \ell\}$.

```
procedure Initialize: $b \xleftarrow{\$} \{0,1\}$; $K \xleftarrow{\$} \{0,1\}^k$

procedure Encrypt $(s, m)$:
if $b = 0$ then                                              ▷ Real world
    $c \leftarrow \mathsf{Enc}(K, s, m)$
else                                                         ▷ Random world
    for $i$ from $1$ to $\ell$ do
        if $\Pi^{(s,i)}(m^i) = \mathsf{undefined}$ then
            $\Pi^{(s,i)}(m^i) \xleftarrow{\$} \overline{\mathsf{Rng}}(\Pi^{(s,i)})$
        end if
        $c^i \leftarrow \Pi^{(s,i)}(m^i)$
    end for
    $c \leftarrow c^1 || \ldots || c^\ell$
end if
return $c$

procedure Decrypt $(s, c)$:
if $b = 0$ then                                              ▷ Real world
    $m \leftarrow \mathsf{Dec}(K, s, c)$
else                                                         ▷ Random world
    for $i$ from $1$ to $\ell$ do
        if $(\Pi^{(s,i)})^{-1}(c^i) = \mathsf{undefined}$ then
            $(\Pi^{(s,i)})^{-1}(c^i) \xleftarrow{\$} \overline{\mathsf{Dom}}(\Pi^{(s,i)})$
        end if
        $m^i \leftarrow (\Pi^{(s,i)})^{-1}(c^i)$
    end for
    $m \leftarrow m^1 || .. || m^k$
end if
return $m$

procedure Finalize $(b')$: return $(b = b')$
```

Fig. 2. Security game "up-to-block": IND-CPA-block

Algorithm 1 **findLCP**$(d_{s,t}, \texttt{'d'})$ for $d \in \{m, c\}$

```
$p \leftarrow 0$ ; $\mathsf{pref} \leftarrow 0$ ; $\mathsf{index} \leftarrow 0$
for $j$ from $1$ to $t-1$ do
    for $i$ from $1$ to $\ell$ do
        while $d^i_{s,j} = d^i_{s,t}$ do $p \leftarrow i$
        end while
    end for
    if $p > \mathsf{pref}$ then
        $\mathsf{pref} \leftarrow p$; $\mathsf{index} \leftarrow j$
    end if
end for
return $(\mathsf{index}, \mathsf{pref})$
```

This notion corresponds to the security notion described in [1, 2] for *on-line ciphers*.

Indistinguishability up-to-repetition. The third security game (described in Fig. 4) defines *indistinguishability up-to-repetition*. This notion formalizes that each ciphertext block depends deterministically on the plaintext block and the sector number s, but behaves as "randomly as possible" subject to this constraint. The game uses independent random permutations $\Pi^s : \{0,1\}^n \to \{0,1\}^n$ for each sector number s. It is the best achievable notion for (length-preserving) deterministic encryption [3].

C EagleTree Modifications

Our modified EagleTree simulator is based on the latest commit of EagleTree on GitHub[10] of February 23, 2016, to which we made three small modifications. In particular:

- We added an additional `diversifier` data member to the `Event` class, which is set to a random value every time the operation system generates a write operation in `generate_io()`.
- The `get_free_block_pointer_with_shortest_IO_queue()` function in the `Block_manager_parent` class now has a `diversifier` argument added to it. It restricts I/O operations to the `package` that is specified by the value of `diversifier`.
- In the block manager `Block_manager_parallel`, the `choose_best_address()` and `choose_any_address()` functions were redefined. For write operations that are issued either by the operating system or by the garbage collector, the `package` will be restricted to the value of the diversifier. This value is passed on when `get_free_block_pointer_with_shortest_IO_queue()` is called within this function. No modifications are made for reads and writes to sectors that are not visible to the OS, such as for example the mapping information I/O generated by the FTL.

D Security Proofs

CBC-essiv. The following theorem shows that CBC-essiv achieves IND-CPA-ufb security notion.

Theorem 3 (The IND-CPA-ufb Security of CBC-essiv)
*Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let $\mathcal{A}$ be an IND-CPA-ufb adversary against the FDE scheme obtained from the CBC-essiv mode on E such that $\mathcal{A}$ runs in time t and makes at most q queries to the **Encrypt** procedure. There exists an adversary $\mathcal{B}$ (attacking the PRF security of E) such that:*

$$\mathbf{Adv}^{ind-cpa-ufb}_{cbc-essiv}(\mathcal{A}) \leq 8 \cdot \mathbf{Adv}^{prf}_{E}(\mathcal{B}) + \frac{q^2(\ell+1)^2}{2^{n-1}}$$

[10] `https://github.com/ClydeProjects/EagleTree`

```
procedure Initialize: $b \xleftarrow{\$} \{0,1\}$; $K \xleftarrow{\$} \{0,1\}^k$; $t \leftarrow 0$

procedure Encrypt $(s, m)$:
$t \leftarrow t + 1$
if $b = 0$ then                                                    ▷ Real world
    $c \leftarrow \mathsf{Enc}(K, s, m)$
else                                                               ▷ Random world
    $m_{s,t} \leftarrow m$
    $(\mathsf{index}, p) \leftarrow \mathbf{findLCP}(m_{s,t}, \texttt{'m'})$
    for $i$ from $1$ to $p$ do
        $c^i_{s,t} \leftarrow c^i_{s,\mathsf{index}}$
    end for
    for $i$ from $p + 1$ to $k$ do
        $\Pi^s_{m^1..m^{i-1}}(m^i_{s,t}) \xleftarrow{\$} \overline{\mathsf{Rng}}(\Pi^s_{m^1..m^{i-1}})$
        $c^i_{s,t} \leftarrow \Pi^s_{m^1 m^2 ... m^{i-1}}(m^i_{s,t})$
    end for
end if
return $c_{s,t}$

procedure Decrypt $(s, c)$:
$t \leftarrow t + 1$
if $b = 0$ then                                                    ▷ Real world
    $m \leftarrow \mathsf{Dec}(K, s, c)$
else                                                               ▷ Random world
    $(\mathsf{index}, p) \leftarrow \mathbf{findLCP}(c_{s,t}, \texttt{'c'})$
    for $i$ from $1$ to $p$ do
        $m^i_{s,t} \leftarrow m^i_{s,\mathsf{index}}$
    end for
    for $i$ from $p + 1$ to $k$ do
        $(\Pi^s_{m^1..m^{i-1}})^{-1}(c^i_{s,t}) \xleftarrow{\$} \overline{\mathsf{Dom}}(\Pi^s_{m^1..m^{i-1}})$
        $m^i_{s,t} \xleftarrow{\$} (\Pi^s_{m^1 m^2 ... m^{i-1}})^{-1}(c^i_{s,t})$
    end for
end if
     return $m_{s,t}$

procedure Finalize $(b')$: return $(b = b')$
```

Fig. 3. Security Game "up-to-prefix": IND-CPA-prefix

```
procedure Initialize: $b \xleftarrow{\$} \{0,1\}$; $K \xleftarrow{\$} \{0,1\}^k$

procedure Encrypt $(s, m)$:
if $b = 0$ then                                              ▷ Real world
    $c \leftarrow \mathsf{Enc}(K, s, m)$
else                                                         ▷ Random world
    if $\Pi^s(m) = \mathsf{undefined}$ then
        $\Pi^s(m) \xleftarrow{\$} \overline{\mathsf{Rng}}(\Pi^s)$
    end if
    $c \leftarrow \Pi^s(m)$
end if
return $c$

procedure Decrypt $(s, c)$:
if $b = 0$ then                                              ▷ Real world
    $m \leftarrow \mathsf{Dec}(K, s, c)$
else                                                         ▷ Random world
    if $(\Pi^s)^{-1}(c) = \mathsf{undefined}$ then
        $(\Pi^s)^{-1}(c) \xleftarrow{\$} \overline{\mathsf{Dom}}(\Pi^s)$
    end if
    $m \leftarrow \Pi^s(c)$
end if
return $m$

procedure Finalize $(b')$: return $(b = b')$
```

Fig. 4. Security Game “up-to-repetition”: IND-CPA-repetition

where $\mathcal{B}$ runs in time at most $t' = t + O(q + nq(\ell + 1))$ and makes at most $q' = q(\ell + 1)$ queries to its oracle.

Proof. Let $\mathcal{G}_0$ be the real-or-random security game as defined in Fig. 3. In the following, we consider a sequence of modified games $\mathcal{G}_1, \ldots, \mathcal{G}_4$. We wish to upper bound the advantage of an adversary $\mathcal{A}$ which by definition is

$$\mathbf{Adv}_{cbc-essiv}^{ind-cpa-ufb}(\mathcal{A}) = 2 \cdot Pr[\mathcal{G}_0^{\mathcal{A}} \Rightarrow \text{true}] - 1.$$

– **Game $\mathcal{G}_1$**: Game $\mathcal{G}_1$ is identical to game $\mathcal{G}_0$ except that instead of the keyed block cipher encryption $E(K', \cdot)$ with a key K' randomly chosen in the FDE scheme obtained from the CBC-essiv, we use a random function $\mathcal{F}'$ instead. As in the previous security notions, the function $\mathcal{F}$ is generated *via* the lazy sampling method. The game $\mathcal{G}_1$ is given in detail in Fig. 5.
  We consider an adversary $\mathcal{B}'$ attacking the PRF security of E which runs $\mathcal{A}$ and simulates these two games by replacing the evaluation of E or $\mathcal{F}'$ by its own oracle (so that if $\mathcal{B}'$ is in the real world, $\mathcal{A}$ is in the game $\mathcal{G}_0$ but if $\mathcal{B}'$ is in the random world, $\mathcal{A}$ is in the game $\mathcal{G}_1$). If $\mathcal{A}$ makes q queries to the **Encrypt** procedure than $\mathcal{B}'$ makes $q' = q(\ell + 1)$ queries to its own oracle and we have:

$$Pr[\mathcal{G}_0^{\mathcal{A}} \Rightarrow 1] - Pr[\mathcal{G}_1^{\mathcal{A}} \Rightarrow 1] \leq \mathbf{Adv}_E^{prf}(\mathcal{B}').$$

```
Game $\mathcal{G}_1$
procedure Encrypt $(s, m)$
if $b = 0$ then
    $c^0 \xleftarrow{\$} \{0,1\}^n$
    if $s \in \mathsf{Dom}(\mathcal{F}')$ then
        $c^0 \leftarrow \mathcal{F}'(s)$
    else
        $\mathcal{F}'(s) \leftarrow c^0$
    end if
    for $i$ from $1$ to $\ell$ do
        $X \leftarrow c^{i-1} \oplus m^i$
        $c^i \leftarrow E(K, X)$
    end for
    $c \leftarrow c^1||c^2||..||c^\ell$
else
    $c \xleftarrow{\$} \{0,1\}^{\ell n}$
end if
return $c$
```

Fig. 5. Game $\mathcal{G}_1$ CBC-essiv IND-CPA-ufb

– **Game** $\mathcal{G}_2$: Game $\mathcal{G}_2$ is identical to game $\mathcal{G}_1$ except that instead of the keyed block cipher encryption $E(K,\cdot)$ with a key K randomly chosen in the FDE scheme obtained from the CBC-essiv, we use a random function $\mathcal{F}$ instead. As in the previous security notions, the function $\mathcal{F}$ is generated *via* the lazy sampling method. The game $\mathcal{G}_2$ is given in detail in Fig. 6.
  Again we consider an adversary $\mathcal{B}''$ attacking the PRF security of E which runs $\mathcal{A}$ and simulates these two games by replacing the evaluation of E or $\mathcal{F}$ by its own oracle (so that if $\mathcal{B}''$ is in the real world, $\mathcal{A}$ is in the game $\mathcal{G}_0$ but if $\mathcal{B}''$ is in the random world, $\mathcal{A}$ is in the game $\mathcal{G}_1$). If $\mathcal{A}$ makes q queries to the **Encrypt** procedure than $\mathcal{B}''$ makes $q' = q(\ell+1)$ queries to its own oracle and we have:

$$Pr[\mathcal{G}_1^{\mathcal{A}} \Rightarrow 1] - Pr[\mathcal{G}_2^{\mathcal{A}} \Rightarrow 1] \leq \mathbf{Adv}_E^{prf}(\mathcal{B}'').$$

```
Game $\mathcal{G}_1$
procedure Encrypt $(s, m)$
if $b = 0$ then
    $c^0 \xleftarrow{\$} \{0,1\}^n$
    if $s \in \mathsf{Dom}(\mathcal{F}')$ then
        $c^0 \leftarrow \mathcal{F}'(s)$
    else
        $\mathcal{F}'(s) \leftarrow c^0$
    end if
    for $i$ from 1 to $\ell$ do
        $c^i \xleftarrow{\$} \{0,1\}^n$
        $X \leftarrow c^{i-1} \oplus m^i$
        if $X \in \mathsf{Dom}(\mathcal{F})$ then
            $c^i \leftarrow \mathcal{F}(X)$
        end if
        $\mathcal{F}(X) \leftarrow c^i$
    end for
    $c \leftarrow c^1||c^2||..||c^\ell$
else
    $c \xleftarrow{\$} \{0,1\}^{\ell n}$
end if
return $c$
```

Fig. 6. Game $\mathcal{G}_2$ CBC-essiv IND-CPA-ufb

– **Game** $\mathcal{G}_3$: Game $\mathcal{G}_3$ (see Fig. 7) is identical to game $\mathcal{G}_1$ except that we add Boolean flags in the pseudo-code for the remainder of the proof (see Fig. 7). We have

$$Pr[\mathcal{G}_2^{\mathcal{A}} \Rightarrow 1] = Pr[\mathcal{G}_3^{\mathcal{A}} \Rightarrow 1].$$

– **Game** $\mathcal{G}_4$: The games $\mathcal{G}_3$ and $\mathcal{G}_4$ (see Fig. 7) are identical except if the Boolean flag bad_2 is set to true. We thus have

$$Pr[\mathcal{G}_3^{\mathcal{A}} \Rightarrow 1] - Pr[\mathcal{G}_4^{\mathcal{A}} \Rightarrow 1] \leq Pr[\mathcal{G}_3^{\mathcal{A}} \text{ sets } \mathsf{bad}_2].$$

```
$\boxed{\text{Game } \mathcal{G}_3}$ Game $\mathcal{G}_4$ Game $\mathcal{G}_5$
procedure Encrypt $(s, m)$
if $b = 0$ then
    $c^0 \xleftarrow{\$} \{0,1\}^n$
    if $s \in \mathsf{Dom}(\mathcal{F}')$ then
        $c^0 \leftarrow \mathcal{F}'(s)$
    else
        if $c^0 \in \mathsf{Rng}(\mathcal{F}')$ then $\mathsf{bad}_1 \leftarrow true$
        end if
        $\mathcal{F}'(s) \leftarrow c^0$
    end if
    for $i$ from 1 to $\ell$ do
        $X \leftarrow c^{i-1} \oplus m^i$
        $X \xleftarrow{\$} \{0,1\}^n$
        $c^i \xleftarrow{\$} \{0,1\}^n$
        if $X \in \mathsf{Dom}(\mathcal{F})$ then
            $\boxed{c^i \leftarrow \mathcal{F}(X)}$
            $\mathsf{bad}_2 \leftarrow true$
        end if
        $\mathcal{F}(X) \leftarrow c^i$
    end for
    $c \leftarrow c^1||c^2||..||c^\ell$
else
    $c \xleftarrow{\$} \{0,1\}^{\ell n}$
end if
return $c$
```

Fig. 7. Games $\mathcal{G}_3$, $\mathcal{G}_4$ and $\mathcal{G}_5$ CBC-essiv IND-CPA-ufb. The framed statement is included in game $\mathcal{G}_3$ only. The statement overlined in gray is included in game $\mathcal{G}_5$ only.

– **Game** $\mathcal{G}_5$: In game $\mathcal{G}_5$ (see Fig. 7), the generated ciphertexts in the **Encrypt** procedure are independent of the bit b and we have $Pr[\mathcal{G}_4^{\mathcal{A}} \Rightarrow 1] = 1/2$.

We first evaluate $Pr[\mathcal{G}_5^{\mathcal{A}} \text{ sets } \mathsf{bad}_1]$. For the i-th request to the **Encrypt** procedure, the probability to have a collision for the output of the $\mathcal{F}'$ is at most $\frac{(i-1)}{2^n}$ then for all the q queries:

$$Pr[\mathcal{G}_5^{\mathcal{A}} \text{ sets } \mathsf{bad}_1] \leq \sum_{i=1}^{q} \frac{(i-1)}{2^n} = \frac{q(q-1)}{2^{n+1}}.$$

If the Boolean flag bad_1 is not set to true, then in the security game, the adversary $\mathcal{A}$ never queried the **Encrypt** procedure with two different sector numbers s and s' such that $\mathcal{F}(s) = \mathcal{F}(s')$. In particular, for each query the value c^0 is random and not revealed to $\mathcal{A}$ and by the unique-first-block assumption, it is never used twice with the same plaintext block m^1. We thus have:

$$Pr[\mathcal{G}_4^{\mathcal{A}} \text{ sets } \mathsf{bad}_2 \text{ but not } \mathsf{bad}_1]$$

$$= Pr[\mathcal{G}_5^{\mathcal{A}} \text{ sets } \mathsf{bad}_2 \text{ but not } \mathsf{bad}_1]$$

Finally, in game $\mathcal{G}_5$, we have

$$Pr[\mathcal{G}_5^{\mathcal{A}} \text{ sets } \mathsf{bad}_2] \leq \sum_{i=1}^{(\ell+1)q-1} \frac{i}{2^n} \leq \frac{q^2(\ell+1)^2}{2^{n+1}}.$$

Summing up, we obtain that $\mathbf{Adv}_{cbc-essiv}^{ind-cpa-ufb}(\mathcal{A})$ is upper-bounded by

$$2 \cdot \mathbf{Adv}_E^{prf}(\mathcal{B}') + 2 \cdot \mathbf{Adv}_E^{prf}(\mathcal{B}'') + \frac{q^2(\ell+1)^2}{2^n} + \frac{q(q-1)}{2^n}.$$

Considering an adversary $\mathcal{B}$ attacking the PRF security of E which simply runs $\mathcal{B}'$ with probability $1/2$ and $\mathcal{B}''$ with probability $1/2$, we have

$$\mathbf{Adv}_E^{prf}(\mathcal{B}) \geq \frac{1}{2}\mathbf{Adv}_E^{prf}(\mathcal{B}') \text{ and } \mathbf{Adv}_E^{prf}(\mathcal{B}) \geq \frac{1}{2}\mathbf{Adv}_E^{prf}(\mathcal{B}'')$$

and we obtain the claimed bound.

IGE-essiv. The following theorem shows that IGE-essiv achieves IND-CPA-ufb security notion.

Theorem 4 ***[IND-CPA-ufb Security of IGE-essiv]***
Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let $\mathcal{A}$ be an IND-CPA-ufb adversary against the FDE scheme obtained from the IGE-essiv mode on E such that $\mathcal{A}$ runs in time t and makes at most q queries to the ***Encrypt*** *procedure. There exists an adversary $\mathcal{B}$ (attacking the PRF security of E) such that:*

$$\mathbf{Adv}_{ige-essiv}^{ind-cpa-ufb}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}_E^{prf}(\mathcal{B}) + \frac{q^2(\ell+1)^2}{2^{n-1}}$$

where $\mathcal{B}$ runs in time at most $t' = t + O(q + nq(\ell+1))$ and makes at most $q' = q(\ell+1)$ queries to its oracle.

Proof. This proof is similar to CBC-essiv proof. Let $\mathcal{G}_0$ be the real-or-random security game as defined in Fig. 3. In the following, we consider a sequences of modified games $\mathcal{G}_1, \ldots, \mathcal{G}_5$. We wish to upper bound the advantage of an adversary $\mathcal{A}$ which by definition is

$$\mathbf{Adv}_{ige-essiv}^{ind-cpa-ufb}(\mathcal{A}) = 2 \cdot Pr[\mathcal{G}_0^{\mathcal{A}} \Rightarrow \mathsf{true}] - 1.$$

```
Game $\mathcal{G}_1$
procedure Encrypt $(s, m)$
if $b = 0$ then
    $m^0 \leftarrow 0^n$
    $c^0 \xleftarrow{\$} \{0,1\}^n$
    if $s \in \mathsf{Dom}(\mathcal{F})$ then
        $c^0 \leftarrow \mathcal{F}(s)$
    else
        $\mathcal{F}(s) \leftarrow c^0$
    end if
    for $i$ from $1$ to $\ell$ do
        $Y \xleftarrow{\$} \{0,1\}^n$
        $X \leftarrow c^{i-1} \oplus m^i$
        if $X \in \mathsf{Dom}(\mathcal{F})$ then
            $Y \leftarrow \mathcal{F}(X)$
        end if
        $\mathcal{F}(X) \leftarrow Y$
        $c^i \leftarrow Y \oplus m^{i-1}$
    end for
    $c \leftarrow c^1||c^2||..||c^\ell$
else
    $c \xleftarrow{\$} \{0,1\}^{\ell n}$
end if
return $c$
```

Fig. 8. Game $\mathcal{G}_1$ IGE-essiv IND-CPA-ufb

```
$\boxed{\text{Game } \mathcal{G}_2}$ Game $\mathcal{G}_3$ Game $\mathcal{G}_4$
procedure Encrypt $(s, m)$
if $b = 0$ then
    $m^0 \leftarrow 0^n$
    $c^0 \xleftarrow{\$} \{0,1\}^n$
    if $s \in \mathsf{Dom}(\mathcal{F}')$ then
        $c^0 \leftarrow \mathcal{F}'(s)$
    else
        if $c^0 \in \mathsf{Rng}(\mathcal{F}')$ then $\mathsf{bad}_1 \leftarrow true$
        end if
        $\mathcal{F}'(s) \leftarrow c^0$
    end if
    for $i$ from $1$ to $\ell$ do
        $X \leftarrow c^{i-1} \oplus m^i$
        $X \xleftarrow{\$} \{0,1\}^n$
        $Y \xleftarrow{\$} \{0,1\}^n$
        if $X \in \mathsf{Dom}(\mathcal{F})$ then
            $\boxed{Y \leftarrow \mathcal{F}(X)}$
            $\mathsf{bad}_2 \leftarrow true$
        end if
        $\mathcal{F}(X) \leftarrow Y$
        $c^i \leftarrow Y \oplus m^{i-1}$
        $c^i \xleftarrow{\$} \{0,1\}^n$
    end for
    $c \leftarrow c^1||c^2||..||c^\ell$
else
    $c \xleftarrow{\$} \{0,1\}^{\ell n}$
end if
return $c$
```

Fig. 9. Games $\mathcal{G}_3$, $\mathcal{G}_4$, $\mathcal{G}_5$ IGE-essiv IND-CPA-ufb. The statement in box is omitted in games $\mathcal{G}_4$ and $\mathcal{G}_5$. The statement overlined in gray is included in Game $\mathcal{G}_5$ only.

- **$\mathcal{G}_1$ - $\mathcal{G}_3$:** (see Fig. 8) The only point that changes between CBC-essiv games $\mathcal{G}_0$ to $\mathcal{G}_4$ is that the ciphertext block is not the output of the PRF Π but the output XORed with the previous plaintext block.
- **$\mathcal{G}_4$:** (see Fig. 9) We now consider IGE-essiv's $\mathcal{G}_4$. In $\mathcal{G}_4$, the ciphertext blocks c^i are the result of a XOR between the previous plaintext block and the PRF $\mathcal{F}$ output that is why $\mathcal{G}_4$ the ciphertext blocks c^i are randomly chosen in $\{0,1\}^n$.