On Improving Performance of One Block Ciphers Mode of Operation Used for Protection of Block-Oriented System Storage Devices

Georgy V. Firsov, Alisa M. Koreneva

***Abstract*—In the end of 2022 in Russian Federation a block ciphers mode of operation named DEC (Disk Encryption with Counter) for protection of block-oriented storage devices was adopted as recommendations for standardization. Due to its operational properties, it is complicated to use it for system partition encryption. In modern software for disk encryption, XTS mode of operation is widely spread. However, properties of the XTS mode lead to degradation of its cryptographic qualities. Previously the authors introduced XEH (Xor-Encrypt-Hash) mode of operation, that mitigates weaknesses of the XTS mode. This paper describes a block ciphers mode of operation XEHf (XEH fast), aimed to improve performance of the XEH mode. Its security is proven in chosen ciphertext attack setting, and its operational properties are studied.**

***Keywords*—Block ciphers mode of operation, Block-oriented storage devices, Full disk encryption, Cryptographic protection of information, Provable security, Symmetric cryptography.**

I. Introduction

Many software solutions for full disk encryption (FDE) are known: VeraCrypt, Apple FileVault, Microsoft BitLocker, etc. These solutions are intended to protect data stored on hard drives from being read by an unauthorized party (hereinafter referred to as adversary). The adversary may act inside of controlled zone and have direct access to data storage device. If the data is stored in encrypted form, passive adversary gains nothing from viewing the data.

Most of existing solutions for FDE utilize specially designed cryptographic algorithms such as block ciphers modes of operation. The most widespread mode is called XEX-based Tweaked-codebook mode with ciphertext Stealing (XTS). The XTS mode is described in NIST SP 800-38E “Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices”. Notwithstanding its widespread use, the XTS mode is considered insecure, since several attacks on this mode are known [1][2]. Some of these attacks are theoretical, they provide an upper bound of the XTS mode security [2]. Other attacks are practical, e.g. plaintext-recovery ones, but they require more data to be processed [1].

Manuscript received November 15, 2023.

G. V. Firsov is with National Research Nuclear University MEPhI, Moscow, Russian Federation, Security Code LLC, Moscow, Russian Federation (corresponding author, e-mail: G.Firsov@securitycode.ru).

A. M. Koreneva, PhD is with Financial University under the Government of the Russian Federation, Moscow, Russian Federation, Security Code LLC, Moscow, Russian Federation (e-mail: A.Koreneva@securitycode.ru).

In the Russian Federation, recommendations for standardization were recently adopted defining a new block ciphers mode of operation called Disk Encryption With Counter (DEC) [3]. The DEC mode requires counters associated with each sector and partition (logical disk) to be stored [4]. This complicates the mode’s usage for system partition encryption (the partition intended to store operation system files). Consider 32 GB disk with 512-byte sectors. At least 256 MB of additional storage is required to store counters necessary for encrypting this disk in the DEC mode with block ciphers standardized in the Russian Federation. At the same time, the standard size of system EFI (Extensible Firmware Interface) partition (the partition that stores data available at boot time) is 100 MB, and extending this partition could not be performed using standard applications built into a Windows operating system (OS).

In our previous works a modification of the XTS mode called XEH (Xor-Encrypt-Hash) is introduced [5][6]. The aim of the XEH mode is to mitigate its predecessor’s weaknesses. The XEH mode is proven to be secure in chosen ciphertext attack (CCA) setting and has up-to-birthday-bound security. Further, this mode has better performance compared to Encrypt-Mix-Encrypt approach given two universal hash functions invocation being faster than encrypting each block in a sector using a block cipher [6]. Universal hash functions used in the XEH mode consist of finite field operations with data blocks represented as finite field elements. Even though these operations may be implemented using special processor instructions, the universal hash functions performance could be improved.

In this paper, we introduce a modification of the XEH mode called XEHf, which stands for Xor-Encrypt-Hash fast. The XEHf mode is aimed to improve performance of its predecessor via using a different universal hash function after encrypting blocks with a block cipher.

The rest of the paper has the following structure. In Section II, we introduce the main notation and terms used in the rest of the paper. Section III contains definition of the XEHf mode. Section IV discussed security properties for the proposed mode. In Section V, we compare the XEHf mode with some existing modes. In Section VI, we summarize results of our study.

II. Preliminaries

Full disk encryption systems encrypt the entire storage device space. Regardless of its physical structure the device is logically split into one or more *partitions*. Partition is a “logical disk” that one can observe in a file explorer built into an operating system. Each partition consists of *sectors*. Sector is the smallest chunk of consecutive data that can be

read from or written to the disk. All sectors have the same size (usually 512 or 4096 KB). Each sector has its own number (we denote it by SN) that is unique inside a corresponding partition.

Consider a fixed partition. Since sector numbers are unique inside the partition, they could be used to "randomize" encryption, i.e. to make the same data to be encrypted into different ciphertexts being located in different sectors. This property could be achieved by using *tweakable encryption schemes*. Notion of tweakable encryption scheme (in regard to block ciphers) is introduced in [7]. In the current paper, we give an adapted definition. A tweakable encryption scheme $\tilde{\mathcal{E}}$ is a family of functions $\tilde{E}: \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{C}$, where $\mathcal{M}$ is a message space, $\mathcal{C}$ is a ciphertext space, $\mathcal{K}$ is a key space, $\mathcal{T}$ is a tweak space. All sets $\mathcal{K}$, $\mathcal{M}$, $\mathcal{C}$ and $\mathcal{T}$ are nonempty. Functions in the family are "indexed" by key. For every $K \in \mathcal{K}$ we write $\tilde{E}_K(\cdot,\cdot)$ for $\tilde{E}(K,\cdot,\cdot)$, where the function $\tilde{E}$ is called *encryption function* of scheme $\tilde{\mathcal{E}}$. The encryption function is bijective on its last variable. Corresponding function $\tilde{E}^{-1}$ is called *decryption function* of the scheme. The scheme satisfies the *correctness requirement*, if the following holds: $\forall K \in \mathcal{K}\ \forall T \in \mathcal{T}\ \forall m \in \mathcal{M}: \tilde{E}_K^{-1}\left(T, \tilde{E}_K(T,m)\right) = m$.

An *untweakable encryption scheme* $\mathcal{E}$ or simply *encryption scheme* is a family of functions $E: \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ "indexed" by key. Partially applied function $E(K,\cdot)$ is bijective. As before, for every $K \in \mathcal{K}$ we write $E_K(\cdot)$ for $E(K,\cdot)$. The scheme satisfies the correctness requirement, if the following equality holds: $\forall K \in \mathcal{K}\ \forall m \in \mathcal{M}: E_K^{-1}\big(E_K(m)\big) = m$.

An untweakable encryption scheme could be regarded as a tweakable encryption scheme with $\mathcal{T} = \{\lambda\}$, where λ is some fixed value, e.g. an empty binary string. That is, the notion of tweakable encryption scheme generalizes the notion of untweakable encryption scheme.

Block cipher $\mathcal{E}$ is an untweakable encryption scheme, where $\mathcal{M} = \mathcal{C} = V_l$, V_l is a set of binary strings of length $l \in \mathbb{N}$. Number l is called *blocksize*.

Number of blocks in a sector is denoted by n. In this paper, we assume that sector length in bits is multiple of n, i.e. sector length in bits equals nl. Further, the following inequality holds: $0 < n < 2^l$.

Let $\mathbb{F}$ be a finite field: $\mathbb{F} = GF(2)[x]/p(x)$, where $p(x) = x^{128} + x^7 + x^2 + x + 1$ for $l = 128$ and $p(x) = x^{64} + x^4 + x^3 + x + 1$ for $l = 64$. We explicitly define the field for $l \in \{64, 128\}$, because these are blocksizes of standardized block ciphers from [8]. The field could be similarly defined for other values of l by choosing proper irreducible polynomial $p(x)$. The primitive element x of the field $\mathbb{F}$ is denoted by α.

Consider an arbitrary set X. The set of all permutations of the set X is denoted by $\mathrm{S}(X)$. We write $x \xleftarrow{\$} X$ to denote the process of assigning to the variable x a random, uniformly distributed element from the set X.

Lemma 1 (Lemma 2 of [6]). Let X be an arbitrary set. Let values $x_1, \dots, x_n$ be independent and uniformly distributed on X. Then n-tuple $x = (x_1, \dots, x_n)$ is uniformly distributed on X^n.

Lemma 2 (Lemma 1 of [6]). Let X be an arbitrary set. Let π be a permutation chosen from $\mathrm{S}(X)$ according to some distribution (not necessary uniform). Let x be an element chosen uniformly from X (this choice is independent from the choice of π). Then the value $\pi(x)$ is uniformly distributed on X.

Lemma 3 (adapted Theorem A.1 of [9]). Let X be a set and $|X| = N$. Let we uniformly and independently choose q elements from X. Let A be an event of at least two chosen elements being equal, then:

$$\Pr[A] \le \frac{q(q-1)}{2N}.$$

In proposed in this paper XEHf mode, block-wise almost universal hash functions are used. The following definition is adapted from [10] and [11]. Let $\mathcal{F}: \mathcal{K} \times \mathcal{D} \to \mathcal{R}^n$ be a keyed family of functions ("indexed" by key), where $\mathcal{K}$ is a key space, $\mathcal{D}$ and $\mathcal{R}$ are arbitrary sets, n is a positive integer. $\mathcal{F}$ is said to be (ϵ_1, ϵ_2)-block-wise almost universal (BAU) if for every $x, x' \in \mathcal{D}$ and for every $i, i' \in \{1, \dots, n\}$ holds:

$$\Pr_K\left[y_i = y'_{i'}\right] \le \epsilon_1, \text{ if } i \ne i',$$
$$\Pr_K\left[y_i = y'_{i'}\right] \le \epsilon_2, \text{ otherwise,}$$

where $(y_1, \dots, y_n) = \mathcal{F}(K, x)$, $(y'_1, \dots, y'_n) = \mathcal{F}(K, x')$, $(x, i) \ne (x', i')$. Subscript "K" denotes, that probability is taken over the uniform choice of $K \in \mathcal{K}$.

In the present paper we use RND-fdeCCA-sector notion from [6] to analyze cryptographic properties of the proposed XEHf mode. This security notion considers an active adversary that may encrypt and decrypt any (allowed) piece of data. In practice, it means that the adversary has direct access to storage device. Such adversary may write some data using FDE subsystem interface (as a legitimate user), read encrypted data directly from the disk and vice versa.

We briefly describe the RND-fdeCCA-sector security notion below. Let $\tilde{\mathcal{E}}$ be a tweakable encryption scheme with $\mathcal{M} = \mathcal{C} = V_{nl}$. In probabilistic experiment RND-fdeCCA-sector, two parties interact with each other by sending queries and responding to them. An adversary $\mathcal{A}$ sends queries to a challenger **Exp**. Each query consists of data and location of the data on disk (e.g. sector number). The challenger provides two oracles: encryption and decryption ones. Encryption oracle encrypts the data from query, decryption oracle, in turn, decrypts the data. The adversary makes $q_e \ge 0$ queries to the encryption oracle, and $q_d \ge 0$ queries to the decryption one.

Consider two worlds: real and random. In real world, encryption and decryption oracles use the scheme $\tilde{\mathcal{E}}$ to process adversary's requests. Key for the scheme is chosen once before the first query and remains unchanged during the experiment. The key is unknown to the adversary. In random world, the encryption and decryption oracles merely return uniformly random ciphertext and plaintext respectively. The adversary's goal is to distinguish these two worlds by analyzing responses to its queries. The adversary returns either 0 or 1 (we write $\mathcal{A} \Rightarrow 0$ or $\mathcal{A} \Rightarrow 1$) for "real" or "random" worlds respectively.

All plaintexts and ciphertexts have the same length. The adversary never repeats its queries and never makes "pointless" queries. By "pointless" we mean such queries which the adversary already "knows" the answer for. That is the adversary never queries for decryption of a ciphertext received from the encryption oracle, and never queries for

encryption of a plaintext received from the decryption oracle.

The adversary's advantage is defined as follows:

$$\mathbf{Adv}_{\tilde{\mathcal{E}}}^{\mathrm{RND-fdeCCA-sector}}(\mathcal{A}) = \\ = \Pr\left[\mathcal{A}^{\tilde{E}_K,\tilde{E}_K^{-1}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\$,\$} \Rightarrow 1\right],$$

where superscripts after $\mathcal{A}$ denote oracles, which it interacts with.

Let $\{a_1, \dots, a_s\}$ be a set of restrictions on adversary's resources. Let $\mathcal{A}(a_1, \dots, a_s)$ is be set of adversaries which resources satisfy given restrictions $a_1, \dots, a_s$. We write:

$$\mathbf{Adv}_{\tilde{\mathcal{E}}}^{\mathrm{RND-fdeCCA-sector}}(a_1, \dots, a_s) = \\ = \max_{\mathcal{A}\in\mathcal{A}(a_1,\dots,a_s)} \mathbf{Adv}_{\tilde{\mathcal{E}}}^{\mathrm{RND-fdeCCA-sector}}(\mathcal{A}).$$

III. Specification of XEHf mode

The main difference between the new XEHf mode and its ancestor the XEH mode is a different block-wise universal hash function used after encrypting data blocks with a block cipher.

The XEHf mode uses two block-wise universal functions $f: \mathbb{F} \times \mathbb{F} \times \mathbb{F}^n \to \mathbb{F}^n$ and $\phi: \mathbb{F} \times \mathbb{F}^n \to \mathbb{F}^n$, that are defined as follows:

$$f(\tau_3, \tau_4, \boldsymbol{y}) = \left(y_1 + Y_{\tau_3,\tau_4}, \cdots, y_{n-1} + Y_{\tau_3,\tau_4}, Y_{\tau_3,\tau_4}\right), \\ \phi(\tau_3, \boldsymbol{y}) = \left(Z_{\tau_3}, y_2 + Z_{\tau_3}, \cdots, y_n + Z_{\tau_3}\right), \quad (1)$$

where $y = (y_1, \dots, y_n)$, $Y_{\tau_3,\tau_4} = \tau_4 + \sum_{j=1}^{n} y_j \tau_3^{n-j}$, where $Z_{\tau_3} = \sum_{j=1}^{n} y_j \tau_3^{j-1}$.

In addition, we define the following functions:

$$g(\tau_2, \tau_3, \tau_4, \boldsymbol{y}) = f(\tau_3, \tau_4, \boldsymbol{y}) + \mathbf{a}_{\tau_2}, \\ \psi(\tau_1, \tau_3, \boldsymbol{y}) = \phi(\tau_3, \boldsymbol{y}) + \mathbf{a}_{\tau_1}, \quad (2)$$

where $\mathbf{a}_{\tau_i} = (\alpha^0\tau_i, \alpha^1\tau_i, \dots, \alpha^{n-1}\tau_i)$ for $i \in \{1,2\}$, $\alpha = x$ is a primitive element of $\mathbb{F}$.

For simplicity, we write $f_{\tau_3,\tau_4}(\cdot)$ for $f(\tau_3, \tau_4, \cdot)$, $g_{\tau_2,\tau_3,\tau_4}(\cdot)$ for $g(\tau_2, \tau_3, \tau_4, \cdot)$, $\phi_{\tau_3}(\cdot)$ for $\phi(\tau_3, \cdot)$ and $\psi_{\tau_1,\tau_3}(\cdot)$ for $\psi(\tau_1, \tau_3, \cdot)$. For fixed τ_i, $i \in \{1, \dots, 4\}$ functions g_{τ_2,τ_3,τ_4} and ψ_{τ_1,τ_3} are permutations.

Lemma 4. Let $\mathbb{F}$ be an arbitrary field. Further let τ_2, τ_3, τ_4 be uniformly and independently chosen elements of $\mathbb{F}$. Then g is $(1/|\mathbb{F}|, (n-1)/|\mathbb{F}|)$-BAU.

We prove Lemma 4 in Appendix A.

Lemma 5 (Lemma 4 of [6]). Let $\mathbb{F}$ be an arbitrary field. Further let τ_1, τ_3 be uniformly and independently chosen elements of $\mathbb{F}$. Then ψ is $(1/|\mathbb{F}|, (n-1)/|\mathbb{F}|)$-BAU.

Let $\Delta_l: V_l \to \mathbb{F}$ be a function, that maps a binary string $a = (a_0, \dots, a_{l-1})$ of V_l to element $\tilde{a} = \sum_{i=0}^{l-1} a_i x^i$ of $\mathbb{F}$, where $a_i \in \{0, 1\}$ for each index i. Let $\nabla_l: \mathbb{F} \to V_l$ be an inverse function of Δ_l.

Let $\mathcal{E}$ be a block cipher with key space $\mathcal{K}$, blocksize l and encryption function E. The XEHf mode uses two independent block cipher keys $K, K' \in \mathcal{K}$, and four subkeys $\tau_1, \tau_2, \tau_3, \tau_4$ that are derived from a sector number SN as follows:

$$\tau_1 = \Delta_l\left(E_K(SN)\right), \\ \tau_2 = \Delta_l\left(E_{K'}\left(\nabla(\tau_1)\right)\right), \\ \tau_3 = \Delta_l\left(E_{K'}(SN)\right), \\ \tau_4 = \Delta_l\left(E_K\left(\nabla(\tau_3)\right)\right).$$

Let $\mathrm{XEHf}^{E_K,E_{K'}}$ be a tweakable encryption scheme constructed from a block cipher $\mathcal{E}$. Encryption and decryption functions of the scheme are denoted by $\mathbf{Enc}_{K,K'}$ and $\mathbf{Dec}_{K,K'}$ respectively. These functions are shown in Fig. 1.

IV. Security of XEHf mode

Let $\mathrm{XEHf}^{\pi,\pi'}$ be a tweakable encryption scheme, where two uniformly random permutations $\pi, \pi' \in \mathrm{S}(V_l)$ are used instead of block cipher in the XEHf mode. The permutation π is used instead of E_K, and permutation π' is used instead of $E_{K'}$. The key space of the scheme $\mathcal{K}'$ is $\mathrm{S}(V_l)^2$, i.e. permutations π and π' are used as a key.

Replacing a block cipher with π and π' allows us to analyze combinatorial properties of the XEHf mode without taking block cipher properties into account. The concrete security of XEHf mode is summarized in the following theorem.

Theorem 1. (XEHf security) Let $\pi \in \mathrm{S}(V_l)$ and $\pi' \in \mathrm{S}(V_l)$ be random independent permutations. Fix positive integers l, n, q. Then:

$$\mathbf{Adv}_{\mathrm{XEHf}^{\pi,\pi'}}^{\mathrm{RND-fdeCCA-sector}}(q) \le \frac{2(n+2)^2 q^2}{2^l},$$

where $q = q_e + q_d$.

We prove Theorem 1 in Appendix B.

V. Discussion and comparison with existing solutions

The XEHf mode is compared with the following modes:

Function $\mathbf{Enc}_{K,K'}(SN, m_1, \dots, m_n)$	Function $\mathbf{Dec}_{K,K'}(SN, c_1, \dots, c_n)$
Derive subkeys	*Derive subkeys*
$\tau_1 \leftarrow \Delta_l\left(E_K(SN)\right)$	$\tau_1 \leftarrow \Delta_l\left(E_K(SN)\right)$
$\tau_2 \leftarrow \Delta_l\left(E_{K'}\left(\nabla(\tau_1)\right)\right)$	$\tau_2 \leftarrow \Delta_l\left(E_{K'}\left(\nabla(\tau_1)\right)\right)$
$\tau_3 \leftarrow \Delta_l\left(E_{K'}(SN)\right)$	$\tau_3 \leftarrow \Delta_l\left(E_{K'}(SN)\right)$
$\tau_4 \leftarrow \Delta_l\left(E_K\left(\nabla(\tau_3)\right)\right)$	$\tau_4 \leftarrow \Delta_l\left(E_K\left(\nabla(\tau_3)\right)\right)$
Encrypt	*Decrypt*
$(mm_1, \dots, mm_n) \leftarrow \psi_{\tau_1,\tau_3}\left(\Delta_l(m_1), \dots, \Delta_l(m_n)\right)$	$(cc_1, \dots, cc_n) \leftarrow g_{\tau_2,\tau_3,\tau_4}\left(\Delta_l(c_1), \dots, \Delta_l(c_n)\right)$
for $i \leftarrow 1$ **to** n **do**	**for** $i \leftarrow 1$ **to** n **do**
$\quad cc_i \leftarrow E_K\left(\nabla_l(mm_i)\right)$	$\quad mm_i \leftarrow E_K^{-1}\left(\nabla_l(cc_i)\right)$
$(c_1, \dots, c_n) \leftarrow g_{\tau_2,\tau_3,\tau_4}^{-1}(cc_1, \dots, cc_n)$	$(m_1, \dots, m_n) \leftarrow \psi_{\tau_1,\tau_3}\left(\Delta_l(mm_1), \dots, \Delta_l(mm_n)\right)$
Return result	*Return result*
return $\left(\nabla_l(c_1), \dots, \nabla_l(c_n)\right)$	**return** $\left(\nabla_l(m_1), \dots, \nabla_l(m_n)\right)$

Figure 1. Encryption (left) and decryption (right) under XEHf mode using block cipher $\mathcal{E}$ with two independent keys $K, K' \in \mathcal{K}$. Functions E_κ and E_κ^{-1} for $\kappa \in \{K, K'\}$ are encryption and decryption functions of the block cipher $\mathcal{E}$ on the key κ.

DEC and XTS, that are standardized modes for block-oriented storage devices, and XEH, which is the baseline of the XEHf mode.

The main difference between XEHf and XEH modes is the use of a different function g after encrypting blocks with a block cipher. For the XEHf mode, this function depends on three subkeys τ_2, τ_3 and τ_4. For the XEH mode similar function (we denote it by h instead of g to prevent ambiguity) depends on two subkeys τ_2, τ_3 and is defined as follows [6]:

$$h(\tau_2,\tau_3,\boldsymbol{y}) = q(\tau_3,\boldsymbol{y}) + \mathbf{a}_{\tau_2},$$
$$q(\tau_3,\boldsymbol{y}) = \left(y_1 + Q_{\tau_3},\cdots,Q_{n-1} + Q_{\tau_3}, Q_{\tau_3}\right), \quad (3)$$

where $Q_{\tau_3} = \left(\sum_{j=1}^{n} y_j \cdot \tau_3^{n-j}\right) + \left(\sum_{j=1}^{n-1} y_j \cdot \mathfrak{F}_l(j)\right)$, and $\mathfrak{F}_l: \mathbb{Z}_{2^l} \to \mathbb{F}$ is a function that maps an element $r = \sum_{i=0}^{l-1} a_i 2^i$ of the ring $\mathbb{Z}_{2^l}$ to an element $\tilde{r} = \sum_{i=0}^{l-1} a_i x^i$ of the field $\mathbb{F}$, $a_i \in \{0,1\}$ for $i \in \{0,\dots,l-1\}$.

It is clearly seen that:

$$Q_{\tau_3} = y_n + \sum_{j=1}^{n-1} y_j \cdot \left(\tau_3^{n-j} + \mathfrak{F}_l(j)\right). \quad (4)$$

Consider the following matrix:

$$\begin{array}{c|cc} \tau_3 & & y_n \\ \tau_3 & (\tau_3 + \mathfrak{F}_l(n-1)) & \cdot\ y_{n-1} \\ \tau_3^2 & (\tau_3^2 + \mathfrak{F}_l(n-2)) & \cdot\ y_{n-2} \\ \vdots & & \vdots \\ \tau_3^{n-1} & (\tau_3^{n-1} + \mathfrak{F}_l(1)) & \cdot\ y_1 \end{array} \quad (5)$$

The matrix (5) consists of n rows. Each row represents a step in computing the value of Q_{τ_3}. The right column (to the right of the vertical line) consists of all summands of the sum (4). The left column represents an auxiliary register. While moving down the rows of the matrix, the value of this register is multiplied by τ_3, except for the step from the first row to the second one. Initially, the register contains τ_3. We observe, that computing the value of Q_{τ_3} requires $2n-3$ multiplications and $2n-2$ additions in the field $\mathbb{F}$. After that according to (3), the value of Q_{τ_3} is added to each block, except for the last one. It requires $n-1$ finite field additions. Next, we add the value of $\mathbf{a}_{\tau_2}$. This step requires n finite field additions and $n-1$ finite field multiplications by the primitive element $\alpha = x$.

Next, consider the function g used in the XEHf mode. To compute its value, the value of f should be computed first. According to Horner's rule, computing the value of Y_{τ_3,τ_4} requires $n-1$ finite field multiplications and n additions. The value of Y_{τ_3,τ_4} is added to each block, except for the last one, which requires $n-1$ finite field additions. Then, the value of $\mathbf{a}_{\tau_2}$ is added to the result of the previous step. It requires n finite field additions and $n-1$ finite field multiplications by the primitive element α.

Total number of subkeys and finite field operations (additions, multiplications and multiplications by primitive element) required to compute functions g and h is shown in Table I.

TABLE I. TOTAL NUMBER OF SUBKEYS AND FINITE FIELD OPERATIONS REQUIRED TO COMPUTE FUNCTIONS g AND h. NUMBER OF ADDITIONS, MULTIPLICATIONS AND MULTIPLICATIONS BY PRIMITIVE ELEMENT ARE DENOTED BY A, M AND MP RESPECTIVELY.

Function	Subkeys	A	M	MP
g	3	$3n-1$	$n-1$	$n-1$
h	2	$4n-3$	$2n-3$	$n-1$

From Table I, we observe that the function g requires less multiplications and additions than h, but uses 3 subkeys instead of 2. Each subkey is produced by invoking a block cipher encryption function. Therefore, the XEHf mode is more efficient than the XEH mode, if $n-2$ multiplications and $n-2$ additions in finite field are performed faster than one invocation of block cipher encryption function.

Similarly to the XEH and XTS modes and in contrast to the DEC mode, the XEHf mode does not require any additional data such as counters, initialization vectors, etc. The DEC mode uses one half-block counter for each partition and one half-block counter for each sector. These counters should be stored on a storage device reducing disk space available to user. Total amount of additional data required for encryption of 32 GB of data on system disk with 512-byte sector using standardized block ciphers is shown in Table II. From Table II, we observe that this amount exceeds 100 MB, and therefore, the additional data could not be stored in system EFI partition, since the standard size of this partition is exactly 100 MB, and it cannot be extended with built-in Windows OS tools. This makes the XEHf mode appropriate for encrypting of system disk.

TABLE II. COMPARISON OF ADDITIONAL DATA AMOUNT FOR THE DEC AND XEHF MODES REQUIRED FOR ENCRYPTION OF 32 GB SYSTEM DISK WITH SECTOR SIZE OF 512 BYTES.

Cipher (mode)	Blocksize	Amount of additional data
Magma (DEC)	64 bits	256 MB
Magma (XEHf)		-
Kuznyechik (DEC)	128 bits	512 MB
Kuznyechik (XEHf)		-

Furthermore, performance measurements are performed for the XTS, XEH and XEHf modes. The DEC mode is excluded from the performance comparison due to impossibility of creating equivalent experimental functioning conditions.

Abovementioned modes are implemented in C programming language with SSE2 instructions set support. In performance comparison each mode uses Kuznyechik as a block cipher [8]. The experiment is conducted on a computer with 2.6 GHz Intel(R) Core(TM) i7-9750H CPU, 8 Gb of DDR4 RAM, and 64-bit macOS 14.1 operating system. During the experiment, 512- and 4096-byte sectors are encrypted and decrypted multiple times, and the average processing time is taken into account. Each value is normalized for a corresponding value for XTS mode (thus, all normalized values for XTS mode are equal to 1). The results are shown in Table III.

TABLE III. RELATIVE TIME OF ENCRYPTING AND DECRYPTING OF 512- AND 4096-BYTE SECTORS IN XTS, XEH AND XEHF MODES.

Mode	Encryption		Decryption	
	512 bytes	4096 bytes	512 bytes	4096 bytes
XTS	1	1	1	1
XEH	1.078	1.051	1.081	1.065
XEHf	1.055	1.037	1.051	1.044

From Table III, one can see, that performance degradation relative to XTS mode does not exceed 9% for XEH and 6% for XEHf. This decrease of performance is substantiated by

additional computations of f and ϕ. The XEHf mode performs all operations faster, than its ancestor.

VI. Conclusion

In this paper we introduce a new provably secure block ciphers mode of operation XEHf, which stands for “XEH fast”, aimed to improve performance of the XEH mode [5][6].

Cryptographic and operational properties of XEHf mode are investigated. The mode is proven to be secure against adaptive adversary in Chosen Ciphertext Attack (CCA) setting. The mode uses block-wise universal hash functions, which properties are essential for the mode’s security.

Performance comparison with existing modes is performed. The XEHf mode runs 3% faster on average compared to the XEH mode due to more efficient hash function.

References

[1] Isobe, T., & Minematsu, K. (2020). “Plaintext recovery attacks against XTS beyond collisions” in K. G. Paterson, D. Stebila (eds.), *Selected Areas in Cryptography - SAC 2019*, 103–123. Springer, Cham.

[2] Firsov, G., & Koreneva, A. (2022). On One Block Cipher Mode of Operation Used to Protect Data on Block-Oriented Storage Devices. *Modern Information Technologies and IT- Education*, *18*(3), 691–701.

[3] R 1323565.1.042-2022. Information technology. Cryptographic protection of information. Block ciphers mode of operation designed to protect of data storage medium with a block-oriented structure. (2022). Russian National Bureau of Standards.

[4] Bogdanov, D., & Nozdrunov, V. (2021). Some properties of one mode of operation of block ciphers. In *10th* Workshop *on Current Trends in Cryptology (CTCrypt 2021). Pre-proceedings* (pp. 12–17).

[5] Firsov, G., & Koreneva, A. (2023). On one block cipher mode of operation for protection of block-oriented storage devices. *Applied Discrete Mathematics. Supplement*, *16*(1), 52–56.

[6] Firsov, G., & Koreneva, A. (2024). On improved security bounds of one block ciphers mode of operation for protection of block-oriented system storage devices. *Journal of Computer Virology and Hacking Techniques.*

[7] Liskov, M., Rivest, R. L., & Wagner, D. (2010). Tweakable block ciphers. *Journal of Cryptology*, *24*(3), 588–613.

[8] GOST 34.12-2018. Information technology. Cryptographic protection of information. Block ciphers. (2018). Russian National Bureau of Standards.

[9] Bellare, M., & Rogaway, P. (2005). *Introduction to Modern Cryptography*.

[10] Halevi, S. (2007). “Invertible Universal Hashing and the TET Encryption Mode” in Menezes, A. (ed), *Advances in Cryptology - CRYPTO 2007. CRYPTO 2007. Lecture Notes in Computer Science. 4622*, 412–429. Springer, Berlin, Heidelberg.

[11] Sarkar, P. (2009). Efficient tweakable enciphering schemes from (block-wise) universal hash functions. *IEEE Transactions on Information Theory*, *55*(10), 4749–4760.

APPENDIX A. THE PROOF OF LEMMA 4

Proof The proof uses the same idea as proof of Lemma 4 of [6]. Fix some $(y_1, \dots, y_n) \in \mathbb{F}^n$ and $(y'_1, \dots, y'_n) \in \mathbb{F}^n$. Further let $(z_1, \dots, z_n) = g_{\tau_2,\tau_3,\tau_4}(y_1, \dots, y_n)$ and $(z'_1, \dots, z'_n) = g_{\tau_2,\tau_3,\tau_4}(y'_1, \dots, y'_n)$.

Case $i \neq i'$. Without loss of generality, we assume $i < i'$. First, consider $i' < n$. We have:

$$z_i - z'_{i'} = \tau_2 \cdot \left(\alpha^{i-1} - \alpha^{i'-1}\right) + R_1, \quad (6)$$

where $R_1 = \left(y_i + Y_{\tau_3,\tau_4}\right) - \left(y'_i + Y'_{\tau_3,\tau_4}\right)$. From (6) immediately follows that event $z_i = z'_{i'}$ is equivalent to the event $\tau_2 = R_1 \cdot \left(\alpha^{i'-1} - \alpha^{i-1}\right)^{-1}$. Since the right part of the equation is independent from τ_2, which is chosen uniformly, the probability of the last event equals $1/|\mathbb{F}|$.

Next, consider $i' = n$. We have:

$$z_i - z'_{i'} = \tau_2 \cdot \left(\alpha^{i-1} - \alpha^{i'-1}\right) + R_2,$$

where $R_2 = \left(y_i + Y_{\tau_3,\tau_4}\right) - Y'_{\tau_3,\tau_4}$. As before, R_2 is independent from τ_2. Using similar idea we conclude, that the probability of z_i and $z'_{i'}$ being equal given $i' = n$ equals $1/|\mathbb{F}|$. Hence:

$$\Pr_{\tau_2,\tau_3,\tau_4}\left[z_i = z'_{i'}\right] \leq \frac{1}{|\mathbb{F}|}, i \neq i'. \quad (7)$$

Case $i = i'$. In this case, $(y_1, \dots, y_n) \neq (y'_1, \dots, y'_n)$ always holds. First, suppose $i < n$. Then:

$$z_i - z'_{i'} = \left(y_i + Y_{\tau_3,\tau_4}\right) - \left(y'_i + Y'_{\tau_3,\tau_4}\right).$$

From (1) and (2), we have: $y_i + Y_{\tau_3,\tau_4} = R_3 + \sum_{j=1}^{n-1} y_j \tau_3^{n-1}$, where $R_3 = y_i + \tau_4 + y_n$. Similarly we have $y'_i + Y'_{\tau_3,\tau_4} = R'_3 + \sum_{j=1}^{n-1} y'_j \tau_3^{n-1}$, $R'_3 = y'_i + \tau_4 + y'_n$. Event $z_i = z'_{i'}$ is equivalent to the event:

$$0 = (R_3 - R'_3) + \sum_{j=1}^{n-1} \left(y_j - y'_j\right) \tau_3^{n-j}. \quad (8)$$

There exists at least one non-zero coefficient of polynomial (8). Therefore, the equality (8) holds if and only if τ_3 is a root of the polynomial.

Degree of the polynomial (8) does not exceed $n - 1$, and hence, there exist no more than $n - 1$ roots. The polynomial coefficients are all independent from τ_3, which is chosen uniformly. Therefore, the probability of z_i and $z'_{i'}$ being equal given $i = i' \wedge i < n$ does not exceed $(n - 1)/|\mathbb{F}|$.

Similar argument shows similar bound for case $i = n$. Hence:

$$\Pr_{\tau_2,\tau_3,\tau_4}\left[z_i = z'_{i'}\right] \leq \frac{n-1}{|\mathbb{F}|}, i = i'. \quad (9)$$

From (7) and (9), we conclude that g is $\left(\frac{1}{|\mathbb{F}|}, \frac{n-1}{|\mathbb{F}|}\right)$-BAU by definition. This completes the proof.

APPENDIX B. THE PROOF OF THEOREM 1

Let $\mathcal{A}$ be an adversary. We write "$\mathcal{A}[GAME] \Rightarrow 1$" to denote that adversary $\mathcal{A}$ returned 1 in security game (experiment) named "GAME". We write "$GAME: bad = 1$" to denote that the "bad" flag equals 1 in the end of security game "GAME".

Proof This proof of the XEHf mode security is based on game-substitution argument. The following four games are introduced:

- *Game XEHf*. In this game the adversary interacts with a challenger, that uses $\text{XEHf}^{\pi,\pi'}$ encryption scheme to process requests. Permutations π and π′ are built via "lazy sampling" technique, i.e. whenever the value of $\pi(x)$ is required, we choose uniformly an "unused" value and define $\pi(x)$ to equal this value. The same steps are performed, whenever the value of $\pi^{-1}(y)$ is required. The permutation π′ is built in the similar way.
- *Game RND1*. This game differs from the previous one in the way of building π and π′ functions. In this game, we do not check if a newly chosen value is "unused". Hence, π and π′ are not necessary permutations.
- *Game RND2*. In this game, the challenger generates just random binary strings of proper length as responses to the adversary's queries. After handling all queries, the challenger checks if there is a collision in either domain or range of π and/or π′ functions.
- *Game NON* (for "noninteractive"). In this game, we consider stronger condition when the adversary sends both plaintext and ciphertext in each query. This game's purpose is to upper bound the probability of a collision in either domain or range of π and π′ functions.

We describe the algorithm of "lazy sampling" in more detail. Using this algorithm, we build π and π′ permutations in games XEHf and RND1. Since π and π′ are build the same way, we describe this algorithm once.

Let $\omega \in \{\pi, \pi'\}$. Let $\mathbf{D}_\omega$ and $\mathbf{R}_\omega$ be domain and range of permutation ω respectively. These sets are used to track the values, for which the permutation is defined. Both sets are initially empty.

Whenever the value of $\omega(x)$ is required, an algorithm $\mathbf{Smp}_\omega$ ("Smp" stands for "sample") is invoked. Corresponding algorithm $\mathbf{Smp}_{\omega^{-1}}$ exists for sampling ω^{-1}. These algorithms are shown in Fig. B.1. For game XEHf, we preserve the shaded statements, and for game RND1, we do not.

We use the "bad" flag, which could be either 0 or 1. Initially this flag is set to 0 and could be modified during the game under certain conditions in algorithms $\mathbf{Smp}_\omega$ and $\mathbf{Smp}_{\omega^{-1}}$.

Algorithm $\mathbf{Smp}_\omega(x)$	Algorithm $\mathbf{Smp}_{\omega^{-1}}(y)$
Choose a value	*Choose a value*
$y \xleftarrow{\$} V_l$	$x \xleftarrow{\$} V_l$
Perform checks	*Perform checks*
if $y \in \mathbf{R}_\omega$ **then**	**if** $x \in \mathbf{D}_\omega$ **then**
$bad \leftarrow 1$	$bad \leftarrow 1$
$y \xleftarrow{\$} V_l \setminus \mathbf{R}_\omega$	$x \xleftarrow{\$} V_l \setminus \mathbf{D}_\omega$
if $x \in \mathbf{D}_\omega$ **then**	**if** $y \in \mathbf{R}_\omega$ **then**
$bad \leftarrow 1$	$bad \leftarrow 1$
$y \leftarrow \omega(x)$	$x \leftarrow \omega^{-1}(y)$
Save value	*Save value*
$\omega(x) \leftarrow y$	$\omega(x) \leftarrow y$
$\mathbf{D}_\omega \leftarrow \mathbf{D}_\omega \cup \{x\}$	$\mathbf{D}_\omega \leftarrow \mathbf{D}_\omega \cup \{x\}$
$\mathbf{R}_\omega \leftarrow \mathbf{R}_\omega \cup \{y\}$	$\mathbf{R}_\omega \leftarrow \mathbf{R}_\omega \cup \{y\}$
Return result	*Return result*
return y	return x

Figure B.1. Algorithms $\mathbf{Smp}_\omega$ *and* $\mathbf{Smp}_{\omega^{-1}}$ *for* $\omega \in \{\pi, \pi'\}$. *We preserve shaded statements for game XEHf and omit them for game RND1.*

In all games we never redefine subkeys. Let $\mathbf{S} \subseteq V_l \times \mathbb{N}$ be a set of pairs that consist of a sector number and a query number, which the sector number was processed for the first time on. Let $\mathbf{T} \subseteq \mathbb{F}^4 \times \mathbb{N}$ be a set of corresponding subkeys. Sets $\mathbf{S}$ and $\mathbf{T}$ are related to each other: if $(SN, j) \in \mathbf{S}$, and $\tau_1, \tau_2, \tau_3, \tau_4$ are computed using sector number SN, then $(\tau_1, \tau_2, \tau_3, \tau_4, j) \in \mathbf{T}$.

Game XEHf. The challenger's encryption and decryption oracles use encryption and decryption functions of the scheme $\mathrm{XEHf}^{\pi,\pi'}$ respectively. In more details, the encryption and decryption oracles invoke algorithms **RespEnc** and **RespDec** respectively. To track currently handled query sequential number, these algorithms maintain an internal counter cnt. To handle a sector number, these algorithms invoke algorithm **Twk**. The algorithm **Twk** is shown in Fig. B.2, algorithms **RespEnc** and **RespDec** are shown in Fig. B.3.

Algorithm **Twk**(SN, j)

if $\exists k\colon k < j \wedge (SN, j) \in \mathbf{S}$ **then**
  $\tau_1, \tau_2, \tau_3, \tau_4 \leftarrow \hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3, \hat{\tau}_4$ s.t. $(\hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3, \hat{\tau}_4, k) \in \mathbf{T}$
else
  $\tau_1 \leftarrow \Delta_l\big(\mathbf{Smp}_{\pi}(SN)\big)$
  $\tau_2 \leftarrow \Delta_l\Big(\mathbf{Smp}_{\pi'}\big(\nabla_l(\tau_1)\big)\Big)$
  $\tau_3 \leftarrow \Delta_l\big(\mathbf{Smp}_{\pi'}(SN)\big)$
  $\tau_4 \leftarrow \Delta_l\Big(\mathbf{Smp}_{\pi}\big(\nabla_l(\tau_3)\big)\Big)$
  $\mathbf{S} \leftarrow \mathbf{S} \cup \{(SN, j)\}$
  $\mathbf{T} \leftarrow \mathbf{T} \cup \{(\tau_1, \tau_2, \tau_3, \tau_4, j)\}$
return $\tau_1, \tau_2, \tau_3, \tau_4$

*Figure B.2. Algorithm **Twk** for games XEHf and RND1.*

Game RND1. The only difference from the previous game is definition of algorithms $\mathbf{Smp}_{\omega}$ and $\mathbf{Smp}_{\omega^{-1}}$ for $\omega \in \{\pi, \pi'\}$. The shaded statements in Fig. B.1 are omitted. Hence, π and π' are not necessary permutations.

Note, that games XEHf and RND1 are identical until the "bad" flag is set to 1. Therefore, we have:

$$\Pr[\mathcal{A}[XEHf] \Rightarrow 1] - \Pr[\mathcal{A}[RND1] \Rightarrow 1] \leq \leq \Pr[RND1\colon bad = 1].$$

Game RND2. We change the structure of a game. In this game, the challenger responds on every query with a random uniform binary string. Each string is chosen independently. After processing all queries, challenger checks if there exist a collision in either domain or range of π and/or π'. If such collision exists, then the "bad" flag is set to 1.

More precisely, we modify algorithms **Twk**, **RespEnc** and **RespDec**. Hereinafter $\mathbf{D}_{\omega}$ and $\mathbf{R}_{\omega}$ are in principle multisets. The algorithm **Twk** does not invoke $\mathbf{Smp}_{\omega}$ anymore. Instead, it uniformly chooses subkeys and directly modifies $\mathbf{D}_{\omega}$ and $\mathbf{R}_{\omega}$. Algorithms **RespEnc** and **RespDec** are shown in Fig. B.4. Algorithm **Twk** is shown in Fig. B.5.

After responding on the last adversary's query, the "bad" flag is set to 1 if there is a collision in at least one of the sets $\mathbf{D}_{\pi}$, $\mathbf{R}_{\pi}$, $\mathbf{D}_{\pi'}$ and $\mathbf{R}_{\pi'}$.

The adversary in RND2 game receives uniformly random n-tuples of elements from V_l. In RND1 game, encryption and decryption oracles uniformly and independently choose values cc_k and mm_k, $k \in \{1, \dots, n\}$ on each query. Applying Lemma 1 and then Lemma 2 (both g_{τ_2,τ_3,τ_4} and ψ_{τ_1,τ_3} are permutations for every $\tau_1, \tau_2, \tau_3, \tau_4$), we conclude that games RND1 and RND2 are indistinguishable by the adversary $\mathcal{A}$, because the adversary receives uniformly random binary strings in both games. Therefore, we have:

$$\Pr[\mathcal{A}[RND1] \Rightarrow 1] = \Pr[\mathcal{A}[RND2] \Rightarrow 1],$$
$$\Pr[RND1\colon bad = 1] = \Pr[RND2\colon bad = 1].$$

Algorithm **Twk**(SN, j)

if $\exists k\colon k < j \wedge (SN, j) \in \mathbf{S}$ **then**
  $\tau_1, \tau_2, \tau_3, \tau_4 \leftarrow \hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3, \hat{\tau}_4$ s.t. $(\hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3, \hat{\tau}_4, k) \in \mathbf{T}$
else
  for $j \leftarrow 1$ **to** 4 **do**
    $\hat{\tau}_j \xleftarrow{\$} V_l$
    $\tau_j \leftarrow \Delta_l(\hat{\tau}_j)$
  $\mathbf{S} \leftarrow \mathbf{S} \cup \{(SN, j)\}$
  $\mathbf{T} \leftarrow \mathbf{T} \cup \{(\tau_1, \tau_2, \tau_3, \tau_4, j)\}$
  $\mathbf{D}_{\pi} \leftarrow \mathbf{D}_{\pi} \cup \{SN, \hat{\tau}_3\}$
  $\mathbf{R}_{\pi} \leftarrow \mathbf{R}_{\pi} \cup \{\hat{\tau}_1, \hat{\tau}_4\}$
  $\mathbf{D}_{\pi'} \leftarrow \mathbf{D}_{\pi'} \cup \{SN, \hat{\tau}_1\}$
  $\mathbf{R}_{\pi'} \leftarrow \mathbf{R}_{\pi'} \cup \{\hat{\tau}_2, \hat{\tau}_3\}$
return $\tau_1, \tau_2, \tau_3, \tau_4$

*Figure B.5. Algorithm **Twk** for game RND2.*

Algorithm **RespEnc**$(SN, \mathbf{m})$

Process sector number
$\tau_1, \tau_2, \tau_3, \tau_4 \leftarrow \mathbf{Twk}(SN, cnt)$
$cnt \leftarrow cnt + 1$
Encrypt
$(mm_1, \dots, mm_n) \leftarrow \psi_{\tau_1,\tau_3}\big(\Delta_l(m_1), \dots, \Delta_l(m_n)\big)$
for $i \leftarrow 1$ **to** n **do**
  $cc_i \leftarrow \mathbf{Smp}_{\pi}\big(\nabla_l(mm_i)\big)$
$(c_1, \dots, c_n) \leftarrow g^{-1}_{\tau_2,\tau_3,\tau_4}(cc_1, \dots, cc_n)$
Return result
return $\big(\nabla_l(c_1), \dots, \nabla_l(c_n)\big)$

Function **RespDec**$(SN, \mathbf{c})$

Process sector number
$\tau_1, \tau_2, \tau_3, \tau_4 \leftarrow \mathbf{Twk}(SN, cnt)$
$cnt \leftarrow cnt + 1$
Decrypt
$(cc_1, \dots, cc_n) \leftarrow g_{\tau_2,\tau_3,\tau_4}\big(\Delta_l(c_1), \dots, \Delta_l(c_n)\big)$
for $i \leftarrow 1$ **to** n **do**
  $mm_i \leftarrow \mathbf{Smp}_{\pi^{-1}}\big(\nabla_l(cc_i)\big)$
$(m_1, \dots, m_n) \leftarrow \psi_{\tau_1,\tau_3}\big(\Delta_l(mm_1), \dots, \Delta_l(mm_n)\big)$
Return result
return $\big(\nabla_l(m_1), \dots, \nabla_l(m_n)\big)$

*Figure B.3. Algorithms **RespEnc** and **RespDec** for games XEHf and RND1.*

Algorithm **RespEnc**$(SN, \mathbf{m})$

Process sector number
$\tau_1, \tau_2, \tau_3, \tau_4 \leftarrow \mathbf{Twk}(SN, cnt)$
$cnt \leftarrow cnt + 1$
"Encrypt"
$(c_1, \dots, c_n) \xleftarrow{\$} V_l^n$
Update multisets
$(mm_1, \dots, mm_n) \leftarrow \psi_{\tau_1,\tau_3}(\Delta_l(m_1), \dots, \Delta_l(m_n))$
$(cc_1, \dots, cc_n) \leftarrow g_{\tau_2,\tau_3,\tau_4}(\Delta_l(c_1), \dots, \Delta_l(c_n))$
$\mathbf{D}_\pi \leftarrow \mathbf{D}_\pi \cup \{\nabla_l(mm_1), \dots, \nabla_l(mm_n)\}$
$\mathbf{R}_\pi \leftarrow \mathbf{R}_\pi \cup \{\nabla_l(cc_1), \dots, \nabla_l(cc_n)\}$
Return result
return $(c_1, \dots, c_n)$

Function **RespDec**$(SN, \mathbf{c})$

Process sector number
$\tau_1, \tau_2, \tau_3, \tau_4 \leftarrow \mathbf{Twk}(SN, cnt)$
$cnt \leftarrow cnt + 1$
"Decrypt"
$(m_1, \dots, m_n) \xleftarrow{\$} V_l^n$
Update multisets
$(mm_1, \dots, mm_n) \leftarrow \psi_{\tau_1,\tau_3}(\Delta_l(m_1), \dots, \Delta_l(m_n))$
$(cc_1, \dots, cc_n) \leftarrow g_{\tau_2,\tau_3,\tau_4}(\Delta_l(c_1), \dots, \Delta_l(c_n))$
$\mathbf{D}_\pi \leftarrow \mathbf{D}_\pi \cup \{\nabla_l(mm_1), \dots, \nabla_l(mm_n)\}$
$\mathbf{R}_\pi \leftarrow \mathbf{R}_\pi \cup \{\nabla_l(cc_1), \dots, \nabla_l(cc_n)\}$
Return result
return $(m_1, \dots, m_n)$

*Figure B.4. Algorithms **RespEnc** and **RespDec** for game RND2.*

In game RND2, the adversary interacts with two random oracles, since these oracles respond with uniformly and independently chosen random binary strings. Hence:

$$\Pr[\mathcal{A}[RND2] \Rightarrow 1] = \Pr\left[\mathcal{A}^{\$,\$} \Rightarrow 1\right].$$

We have the following upper bound on the adversary's advantage:

$$\begin{aligned}
&\mathbf{Adv}_{\mathrm{XEHf}^{\pi,\pi'}}^{\mathrm{RND-fdeCCA-sector}}(\mathcal{A}) = \\
&= \Pr\left[\mathcal{A}^{\tilde{E}_K, \tilde{E}_K^{-1}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\$,\$} \Rightarrow 1\right] = \\
&= \Pr[\mathcal{A}[XEHf] \Rightarrow 1] - \Pr[\mathcal{A}[RND2] \Rightarrow 1] = \qquad (10)\\
&= \Pr[\mathcal{A}[XEHf] \Rightarrow 1] - \Pr[\mathcal{A}[RND1] \Rightarrow 1] \leq \\
&\leq \Pr[RND1: bad = 1] = \\
&= \Pr[RND2: bad = 1].
\end{aligned}$$

Game NON. In this game, we consider stronger condition. The adversary sends both plaintext and ciphertext in each query. More formally, it makes $q = q_e + q_d$ queries. The j-th query has the following form: $(SN_j, \mathbf{m}_j, \mathbf{c}_j, t_j)$, where SN_j is a sector number, $\mathbf{m}_j$ is an n-block message (plaintext), $\mathbf{c}_j$ is an n-block ciphertext, t_j is either 0 or 1 denoting encryption or decryption query respectively. We denote i-th block of $\mathbf{m}_j$ ($\mathbf{c}_j$) by $m_{j,i}$ ($c_{j,i}$) for $i \in \{1, \dots, n\}$. Oracles perform the same actions as in previous game, except for response generation, since response is provided by adversary.

The adversary's queries are such that they maximize the probability of the "bad" flag being set and are not pointless. The adversary never makes the same query twice.

Now we could get rid of the concrete adversary and find an upper bound on the value of $\mathbf{Adv}_{\mathrm{XEHf}^{\pi,\pi'}}^{\mathrm{RND-fdeCCA-sector}}(q)$ via bounding the probability $\Pr[RND2: bad = 1]$.

Collision analysis. Let $Coll_X$ be an event of a collision in multiset X for $X \in \{\mathbf{D}_\pi, \mathbf{R}_\pi, \mathbf{D}_{\pi'}, \mathbf{R}_{\pi'}\}$. By inclusion-exclusion principle we have:

$$\begin{aligned}
\Pr[RND2: bad = 1] \leq &\Pr\left[Coll_{\mathbf{D}_\pi}\right] + \\
&+ \Pr\left[Coll_{\mathbf{R}_\pi}\right] + \\
&+ \Pr\left[Coll_{\mathbf{D}_{\pi'}}\right] + \qquad (11)\\
&+ \Pr\left[Coll_{\mathbf{R}_{\pi'}}\right].
\end{aligned}$$

Consider j-th query. Let $(mm_{j,1}, \dots, mm_{j,n}) = \psi_{\tau_{1,j},\tau_{3,j}}\left(\Delta_l(m_{j,1}), \dots, \Delta_l(m_{j,n})\right)$, $(cc_{j,1}, \dots, cc_{j,n}) = g_{\tau_{2,j},\tau_{3,j},\tau_{4,j}}\left(\Delta_l(c_{j,1}), \dots, \Delta_l(c_{j,n})\right)$, where $\tau_{1,j}$, $\tau_{2,j}$, $\tau_{3,j}$, $\tau_{4,j}$ are subkeys for the j-th query.

First, consider the multiset $\mathbf{R}_\pi$. This multiset consists of the following values: $\nabla_l(\tau_{1,j})$, $\nabla_l(\tau_{4,j})$, $\nabla_l(cc_{j,1})$, …, $\nabla_l(cc_{j,n})$ for every $j \in \{1, \dots, q\}$.

Let $Coll_{\tau,g}$ be an event that there is at least one pair (i, j) such that $SN_i \neq SN_j$ and $(\tau_{2,i}, \tau_{3,i}, \tau_{4,i}) = (\tau_{2,j}, \tau_{3,j}, \tau_{4,j})$. From law of total probability, we have:

$$\Pr\left[Coll_{\mathbf{R}_\pi}\right] \leq \Pr\left[Coll_{\tau,g}\right] + \Pr\left[Coll_{\mathbf{R}_\pi} | \overline{Coll_{\tau,g}}\right], \qquad (12)$$

where event $\overline{Coll_{\tau,g}}$ is complement of $Coll_{\tau,g}$.

First, consider the event $Coll_{\tau,g}$. There are at most $\binom{q}{2} \leq q^2/2$ pairs of different sector numbers. Consider a pair (i, j) such that $SN_i \neq SN_j$. The probability of $(\tau_{2,i}, \tau_{3,i}, \tau_{4,i})$ and $(\tau_{2,j}, \tau_{3,j}, \tau_{4,j})$ being equal is 2^{-3l}, because all subkeys are chosen uniformly and independently. Hence:

$$\Pr\left[Coll_{\tau,g}\right] \leq \frac{q^2}{2 \cdot 2^{3l}}. \qquad (13)$$

Next, we assume that the event $\overline{Coll_{\tau,g}}$ occurs, i.e. all subkeys are different for different sector numbers. Let $p \leq q$ be the total number of different sector numbers occurred among adversary's queries. We write all these sector numbers in some order (the specific order is not important in this context): $(SN^1, \dots, SN^p)$, where superscript denote sequence numbers in particular order.

We rewrite all values of $cc_{j,1}$, …, $cc_{j,n}$ into p matrices. Each such matrix contains values computed for queries with the same sector number. These matrices are of the following form:

$$\mathcal{CC}_t = \begin{Vmatrix} cc_{k_1,1} & \cdots & cc_{k_1,n} \\ \vdots & \ddots & \vdots \\ cc_{k_{q_t},1} & \cdots & cc_{k_{q_t},n} \end{Vmatrix}, \qquad (14)$$

where $t \in \{1, \dots, p\}$, and $\{k_1, \dots, k_{q_t}\} \subseteq \{1, \dots, q\}$ are numbers of queries that contain sector number SN^t, i.e. $SN_{k_1} = \cdots = SN_{k_{q_t}} = SN^t$. Note that $\sum_{t=1}^{p} q_t = q$.

Now we bound the probability of a collision in columns of matrices (14). There are n columns in t-th matrix, in each column there are $\binom{q_t}{2}$ elements. Therefore, there are at most $n \cdot \binom{q_t}{2}$ different pairs of elements in columns of t-th matrix. By Lemma 4, the probability of collision in such pair does not exceed $\frac{n-1}{2^l}$. Therefore, we have the following upper bound on probability of collision among such pairs:

$$\sum_{t=1}^{p}\frac{n(n-1)\cdot\binom{q_t}{2}}{2^l}\le\sum_{t=1}^{p}\frac{q_t^2 n(n-1)}{2\cdot 2^l}\le$$
$$\le\frac{n(n-1)}{2\cdot 2^l}\left(\sum_{t=1}^{p}q_t^2\right)^2=$$
$$=\frac{1}{2}\cdot\frac{n(n-1)q^2}{2^l}.$$

The number of remaining pairs does not exceed $\binom{|\mathbf{R}_\pi|}{2}\le |\mathbf{R}_\pi|^2/2$. Probability of a collision in such pair is not greater, than $1/2^l$. Note that $|\mathbf{R}_\pi|\le(n+2)q$. Hence, the probability of collision among these pairs is less or equal to $(n+2)^2q^2/(2\cdot 2^l)$.

The probability of collision in $\mathbf{R}_\pi$ given $\overline{Coll_{\tau,g}}$ is bounded as follows:

$$\Pr\left[Coll_{\mathbf{R}_\pi}\middle|\overline{Coll_{\tau,g}}\right]\le$$
$$\le\frac{1}{2}\left(\frac{n(n-1)q^2}{2^l}+\frac{(n+2)^2q^2}{2^l}\right). \quad (15)$$

From (12) given (13) and (15), we have:

$$\Pr\left[Coll_{\mathbf{R}_\pi}\right]\le$$
$$\le\frac{1}{2}\left(\frac{n(n-1)q^2}{2^l}+\frac{(n+2)^2q^2}{2^l}+\frac{q^2}{2^{3l}}\right).$$

Similar analysis shows the following upper bound for $\Pr\left[Coll_{\mathbf{D}_\pi}\right]$. The only difference is that we apply Lemma 5 instead of Lemma 4, and instead of the event $Coll_{\tau,g}$, we consider an event $Coll_{\tau,\psi}$ of existence of at least on pair (i,j) such that $SN_i\ne SN_j$ and $(\tau_{1,i},\tau_{3,i})=(\tau_{1,j},\tau_{3,j})$:

$$\Pr\left[Coll_{\mathbf{D}_\pi}\right]\le$$
$$\le\frac{1}{2}\left(\frac{n(n-1)q^2}{2^l}+\frac{(n+2)^2q^2}{2^l}+\frac{q^2}{2^{2l}}\right).$$

Next, consider the multiset $\mathbf{R}_{\pi'}$. It consists of at most $2q$ values. By Lemma 3, we have the following upper bound on probability of collision is this multiset:

$$\Pr\left[Coll_{\mathbf{R}_{\pi'}}\right]=\frac{1}{2}\cdot\frac{2q(2q-1)}{2^l}<\frac{2q^2}{2^l}.$$

Similar analysis shows the same upper bound for $\Pr\left[Coll_{\mathbf{D}_{\pi'}}\right]$.

Given abovementioned upper bounds on summands in (11), we have:

$$\Pr[RND2:bad=1]\le$$
$$\le\frac{n(n-1)q^2}{2^l}+\frac{(n+2)^2q^2}{2^l}+\frac{q^2}{2\cdot 2^{3l}}+$$
$$+\frac{q^2}{2\cdot 2^{2l}}+\frac{4q^2}{2^l}\le$$
$$\le\frac{n(n-1)q^2}{2^l}+\frac{(n+2)^2q^2}{2^l}+\frac{q^2}{8\cdot 2^l}+$$
$$+\frac{q^2}{4\cdot 2^l}+\frac{4q^2}{2^l}=$$
$$=\frac{q^2}{2^l}\left(n(n-1)+(n+2)^2+\frac{35}{8}\right)\le$$
$$\le\frac{2(n+2)^2q^2}{2^l}.$$

Hence, advantage of an adversary, that makes q queries in total, has the following upper bound:

$$\mathbf{Adv}^{\mathrm{RND-fdeCCA-sector}}_{\mathrm{XEHf}^{\pi,\pi'}}(q)\le\frac{2(n+2)^2q^2}{2^l}.$$

This completes the proof.