electronics

Article

Equivalent Keys: Side-Channel Countermeasure for Post-Quantum Multivariate Quadratic Signatures

David Pokorný , Petr Socha and Martin Novotný *

Department of Digital Design, Faculty of Information Technology, Czech Technical University in Prague, 160 00 Prague, Czech Republic
* Correspondence: martin.novotny@fit.cvut.cz

Abstract: Algorithms based on the hardness of solving multivariate quadratic equations present promising candidates for post-quantum digital signatures. Contemporary threats to implementations of cryptographic algorithms, especially in embedded systems, include side-channel analysis, where attacks such as differential power analysis allow for the extraction of secret keys from the device's power consumption or its electromagnetic emission. To prevent these attacks, various countermeasures must be implemented. In this paper, we propose a novel side-channel countermeasure for multivariate quadratic digital signatures through the concept of equivalent private keys. We propose a random equivalent key to be generated prior to every signing, thus randomizing the computation and mitigating side-channel attacks. We demonstrate our approach on the Rainbow digital signature, but since an unbalanced oil and vinegar is its special case, our work is applicable to other multivariate quadratic signature schemes as well. We analyze the proposed countermeasure regarding its properties such as the number of different equivalent keys or the amount of required fresh randomness, and we propose an efficient way to implement the countermeasure. We evaluate its performance regarding side-channel leakage and time/memory requirements. Using test vector leakage assessment, we were not able to detect any statistically significant leakage from our protected implementation.

Keywords: embedded systems; multivariate quadratic signature; post-quantum cryptography; side-channel security

check for updates

Citation: Pokorný, D.; Socha, P.; Novotný, M. Equivalent Keys: Side-Channel Countermeasure for Post-Quantum Multivariate Quadratic Signatures. *Electronics* **2022**, *11*, 3607. https://doi.org/10.3390/electronics11213607

Academic Editor: Alexander Barkalov

Received: 28 September 2022
Accepted: 1 November 2022
Published: 4 November 2022

1. Introduction

In the past few decades, computer devices have become an essential part of our everyday lives. They are used throughout all industries, including banking, medicine, transportation, and entertainment. Various embedded computer systems are used to control payments, pacemakers, and trains, and most of us carry them in our own pockets every day. In the interconnected world of Industry 4.0, the Internet of Things, cloud computing, and machine learning, our privacy, security, and safety are more endangered than ever [1]. To counteract these modern-day threats, numerous security measures must be taken to provide confidentiality, authenticity, integrity, and more.

Various cryptographic primitives are widely used, including symmetric algorithms such as AES [2] and asymmetric algorithms such as RSA [3] or based on elliptic curves [4]. One of the significant threats to such asymmetric cryptographic algorithms is quantum computation, which is expected to effectively break RSA and ECC cryptosystems. This is due to Shor's algorithm [5], which allows prime factorization and discrete logarithm solving in polynomial time, compared to exponential time on a classical computer. For this reason, the National Institute of Standards and Technology (NIST) of the United States Department of Commerce has initiated a process to standardize quantum-resistant public key cryptographic algorithms. The digital signature candidates in the third round were CRYSTALS-Dilithium [6], FALCON [7], both lattice-based schemes, and Rainbow [8], a multivariate quadratic scheme. Both lattice-based signatures have been currently recommended for standardization, with some multivariate quadratic candidate expected to be further evaluated in the fourth round. In this paper, we focus on multivariate quadratic signature schemes, which, besides Rainbow, also include the Unbalanced Oil and Vinegar (UOV) [9] and LUOV [10] schemes.

Besides classical cryptanalysis, side-channel analysis poses a threat to cryptographic implementations in general. Attacks such as differential power analysis [11] and correlation power analysis [12,13] exploit information leakage in the side-channels, such as power consumption or electromagnetic radiation [14]. More complex attacks include template attacks [15,16] and machine-learning-based attacks [17–19]. Successful side-channel attacks on multivariate quadratic signature schemes were presented in [20,21]. Various countermeasures against these kinds of attacks have been proposed in the literature, typically classified as either hiding or masking. Hiding countermeasures aim to conceal the leaked information in noise, e.g., by using custom secure digital logic [22,23] or additional modules such as noise generators, clock randomization, or current equalizers [24,25]. Masking countermeasures split the sensitive variable into multiple variables using random masks, thus making it difficult for the attacker to predict any intermediate values and attack. Masking countermeasures include threshold implementations [26,27] or domain-oriented masking [28]. Lightweight masking countermeasures have been proposed for multivariate cryptography, based on digest masking using a scalar mask [20,21], introducing only a little masking entropy. A more complex countermeasure, based on splitting the central map solution of Rainbow and randomizing the signing process, was presented in [29]. However, no evaluation (such as test vector leakage assessment) of the proposed countermeasures was presented in either paper. Most recently, a novel cryptanalytic attack on the Rainbow signature scheme was presented in [30], requiring reexamination of the proposed algorithm parameters and their security level.

Our Contribution

In this paper, we propose a novel side-channel countermeasure for multivariate cryptography based on private key randomization. We continue the work presented in [31] and introduce the concept of equivalent private keys, i.e., a class of private keys with the same public key. We propose a new private key to be generated prior to every signing. The adversary would then be forced to target the whole class instead of a fixed private key, making it difficult to mount a side-channel attack. We demonstrate our approach on the Rainbow algorithm. Since the UOV is a special (single-layered) case of Rainbow, our work is applicable to different multivariate cryptography algorithms as well.

We first describe the Rainbow algorithm and the notation used in Section 2. In Section 3, we introduce the general concept of equivalent keys and analyze their properties such as the number of different keys or required random bits. We further propose an efficient way to generate an equivalent key in Section 4, i.e., with fewer computational resources and randomness. Finally, we evaluate the performance of our countermeasure in Section 5 in terms of side-channel leakage and time and memory requirements. We were not able to detect any statistically significant side-channel leakage using an attack-independent test vector leakage assessment; we show that the overhead is reasonable in comparison to another state-of-the-art countermeasure.

2. Rainbow

Rainbow is a post-quantum digital signature, a generalization of the oil and vinegar signature scheme [8]. Its security depends on the fact that solving m quadratic equations for n variables becomes very difficult. These equations are defined by a central map F, which consists of multivariate quadratic polynomials. The central map is structured so that it is possible to solve $F(x_1, \dots, x_n) = (y_1, \dots, y_m)$ using a linear equation solver. In a public key, the special structure of the central map is hidden by two linear maps, S and T, applied on an input and an output of the central map. The central map is layered, where each layer defines two types of variables: vinegar variables for known variables (randomly generated or computed from the prior layer) and oil variables for unknown variables (to be computed using polynomials in the given layer). In this paper, we used the version of the Rainbow algorithm as defined in the submission to the NIST competition [32].

Let $\mathbb{F}$ be a finite field and $v_1, o_1, o_2 \in \mathbb{N}$ the parameters of the two-layered version of Rainbow. The parameters define the number of variables and the sizes of the layers. The first layer consists of o_1 quadratic polynomials with v_1 vinegar variables with indices $V_1 = \{1, 2, \ldots, v_1\}$ and o_1 oil variables with indices $O_1 = \{v_1 + 1, \ldots, v_1 + o_1\}$. The second layer contains o_2 quadratic polynomials with $v_2 = v_1 + o_1$ vinegar variables with indices $V_2 = \{1, 2, \ldots, v_2\}$ and o_2 oil variables with indices $O_2 = \{v_2 + 1, \ldots, v_2 + o_2\}$. Overall, we have $m = o_1 + o_2$ polynomials with $n = v_1 + o_1 + o_2$ variables.

The central map $F = (f^{(v_1+1)}, f^{(v_1+2)}, \ldots, f^{(n)})$ is an m-tuple of n-variate quadratic polynomials. These polynomials, indexed by $k \in O_1 \cup O_2$, are defined as:

$$f^{(k)}(x_1, \ldots, x_n) := \sum_{\substack{i,j \in V_l \\ i \leq j}} \alpha_{i,j}^{(k)} x_i x_j + \sum_{\substack{i \in V_l \\ j \in O_l}} \beta_{i,j}^{(k)} x_i x_j + \sum_{i \in V_l \cup O_l} \gamma_i^{(k)} x_i + \delta^{(k)}, \tag{1}$$

where $\alpha_{i,j}^{(k)}, \beta_{i,j}^{(k)}, \gamma_i^{(k)}, \delta^{(k)} \in \mathbb{F}$ are term coefficients and $l \in \{1, 2\}$ is such that $k \in O_l$.

For simplicity, we changed the notation in Equation (1) in the following sense:

- We restricted the polynomials, omitting the linear and absolute parts of the polynomials in the central map, resulting in quadratic forms. These coefficients are not necessary for Rainbow's security, but add more complexity. The reference implementation submitted to the standardization process considers these coefficients in the documentation, but they were not implemented in the code. Nevertheless, the countermeasure discussed in this paper can be applied to the original polynomials defined in Equation (1) as well.
- We united the α and β coefficients into the λ coefficients for simplicity of notation and also included zero coefficients as described in Equation (2). We did not allow setting any (by definition) zero coefficient to a non-zero value, so there is no change from the original definition.

$$\lambda_{i,j}^{(k)} := \begin{cases} \alpha_{i,j}^{(k)} & (k \in O_1), (i, j \in V_1), (i \leq j), \\ \beta_{i,j}^{(k)} & (k \in O_1), (i \in V_1), (j \in O_1), \\ \alpha_{i,j}^{(k)} & (k \in O_2), (i, j \in V_2), (i \leq j), \\ \beta_{i,j}^{(k)} & (k \in O_2), (i \in V_2), (j \in O_2), \\ 0 & \text{otherwise}. \end{cases} \tag{2}$$

Applying our changes, we obtain a simple notation of the central map polynomials, equivalent to Equation (1):

$$f^{(k)}(\mathbf{x}) = \sum_{i,j \in \hat{n}} \lambda_{i,j}^{(k)} \cdot x_i \cdot x_j, \tag{3}$$

where $k \in O_1 \cup O_2$, $\hat{n} = \{1, 2, \ldots, n\}$ and $\mathbf{x} = (x_1, \ldots, x_n)$.

In Rainbow, the structure of the central map F (described in Equation (1)) is hidden using two random invertible affine maps $S : \mathbb{F}^m \to \mathbb{F}^m$ and $T : \mathbb{F}^n \to \mathbb{F}^n$. Similar to the linear and absolute parts of the central map polynomials, we also omitted translations of the affine maps. Therefore, we used two linear maps represented by regular matrices $\mathbf{S} \in \mathbb{F}^{m \times m}, \mathbf{T} \in \mathbb{F}^{n \times n}$. This is also consistent with the reference implementation.

2.1. Central Map in Matrix Representation

For compact writing, we rewrite the quadratic forms in the central map as a product of matrix–vector multiplication for $k \in O_1 \cup O_2$:

$$\begin{aligned} f^{(k)}(\mathbf{x}) = \mathbf{x}^\mathsf{T}\mathbf{F}^{(k)}\mathbf{x} &= \mathbf{x}^\mathsf{T} \begin{pmatrix} \mathbf{F}_{1,1}^{(k)} & \mathbf{F}_{1,2}^{(k)} & \mathbf{F}_{1,3}^{(k)} \\ \mathbf{0} & \mathbf{F}_{2,2}^{(k)} & \mathbf{F}_{2,3}^{(k)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{x}, \\ \mathbf{F}_{1,1}^{(k)} &= \{\lambda_{i,j}^{(k)}\}_{i,j \in V_1}, \\ \mathbf{F}_{1,2}^{(k)} &= \{\lambda_{i,j}^{(k)}\}_{i \in V_1, j \in O_1}, \\ \mathbf{F}_{1,3}^{(k)} &= \{\lambda_{i,j}^{(k)}\}_{i \in V_1, j \in O_2}, \\ \mathbf{F}_{2,2}^{(k)} &= \{\lambda_{i,j}^{(k)}\}_{i,j \in O_1}, \\ \mathbf{F}_{2,3}^{(k)} &= \{\lambda_{i,j}^{(k)}\}_{i \in O_1, j \in O_2}. \end{aligned} \tag{4}$$

The quadratic form $f^{(k)}(\mathbf{x})$ can be expressed using a matrix $\mathbf{F}^{(k)}$. This matrix is partitioned according to the structure of the central map. Submatrices $\mathbf{F}_{1,1}^{(k)}, \mathbf{F}_{2,2}^{(k)}$ are upper triangulars and correspond to alpha coefficients. Submatrices $\mathbf{F}_{1,2}^{(k)}, \mathbf{F}_{1,3}^{(k)}, \mathbf{F}_{2,3}^{(k)}$ correspond to beta coefficients. Specifically for the first layer, we obtain:

$$(k \in O_1) \implies \mathbf{F}^{(k)} = \begin{pmatrix} \mathbf{F}_{1,1}^{(k)} & \mathbf{F}_{1,2}^{(k)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{5}$$

The central map can be expressed as a vector of m matrices (representing quadratic forms):

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}^{(v_1+1)} \\ \mathbf{F}^{(v_1+2)} \\ \vdots \\ \mathbf{F}^{(n)} \end{pmatrix}. \tag{6}$$

Secret key SK can be expressed and stored as:

$$\mathrm{SK} = (S^{-1}, F, T^{-1}) \leftrightarrow (\mathbf{S}^{-1}, \mathbf{F}, \mathbf{T}^{-1}), \tag{7}$$

where

$$\begin{aligned} &F : \mathbb{F}^n \to \mathbb{F}^m \\ &\mathbf{x} \mapsto y = F(\mathbf{x}) = \begin{pmatrix} f^{(v_1+1)}(\mathbf{x}) \\ f^{(v_1+2)}(\mathbf{x}) \\ \vdots \\ f^{(n)}(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \mathbf{x}^\mathsf{T}\mathbf{F}^{(v_1+1)}\mathbf{x} \\ \mathbf{x}^\mathsf{T}\mathbf{F}^{(v_1+2)}\mathbf{x} \\ \vdots \\ \mathbf{x}^\mathsf{T}\mathbf{F}^{(n)}\mathbf{x} \end{pmatrix}. \end{aligned} \tag{8}$$

The public key contains only a quadratic map P defined as

$$P := S \circ F \circ T. \tag{9}$$

2.2. Signing and Verification Process

For document $\mathbf{d}$, random salt $\mathbf{r}$, and a secret key $\mathrm{SK} = (S^{-1}, F, T^{-1})$, we define

$$\begin{aligned} \mathbf{h} &:= \mathrm{hash}(\mathrm{hash}(\mathbf{d}) || \, \mathbf{r}), \\ \mathbf{y} &:= S^{-1}(\mathbf{h}), \\ \mathbf{x} &:= F^{-1}(\mathbf{y}), \\ \mathbf{z} &:= T^{-1}(\mathbf{x}), \end{aligned} \tag{10}$$

where $\mathbf{h}, \mathbf{y} \in \mathbb{F}^m$ and $\mathbf{x}, \mathbf{z} \in \mathbb{F}^n$. The pair $(\mathbf{z}, \mathbf{r})$ is called a signature.

The signature $(\mathbf{z}, \mathbf{r})$ of the document $\mathbf{d}$ is valid iff $\mathbf{h} = \mathbf{h}'$, where

$$\begin{aligned} \mathbf{h} &:= \mathrm{hash}(\mathrm{hash}(\mathbf{d}) || \, \mathbf{r}), \\ \mathbf{h}' &:= P(\mathbf{z}). \end{aligned} \tag{11}$$

3. Equivalent Key

The main idea of the equivalent key is to change the secret key SK into some other $\overline{\mathrm{SK}}$, which is equivalent to the original one, i.e., SK and $\overline{\mathrm{SK}}$ both have the same public key. This equivalency was previously used for storage, computation, and randomness reduction, where only the normal form of the key was generated [31]. Our countermeasure is based on the generation of the equivalent key before each signing process. We therefore substituted a fixed key with an equivalence class containing a huge number (as shown in Equation (28)) of different, but equivalent, keys.

In our work, the equivalent key can be derived from the original key using two linear maps A and B, represented as matrices $\mathbf{A}$ and $\mathbf{B}$, which change all three maps contained in a secret key without changing their composition:

$$\begin{aligned} P &= S \circ F \circ T, \\ P &= S \circ (A^{-1} \circ A) \circ F \circ (B \circ B^{-1}) \circ T, \\ P &= (S \circ A^{-1}) \circ (A \circ F \circ B) \circ (B^{-1} \circ T). \end{aligned} \tag{12}$$

With this in mind, we define the secret key $\overline{\mathrm{SK}}$ derived from SK as

$$\begin{aligned} \mathrm{SK} &= (S^{-1}, F, T^{-1}), \\ \overline{\mathrm{SK}} &= (A \circ S^{-1}, A \circ F \circ B, T^{-1} \circ B). \end{aligned} \tag{13}$$

The pair (A, B) is called Gauss sustaining transformation, where the representing matrices $\mathbf{A}$ and $\mathbf{B}$ cannot be chosen arbitrarily. They are composed with the central map F, which must maintain its special structure.

The necessary constraint is the invertibility of these matrices due to the signing process, where the inverse central map must be computed. Therefore, $\mathbf{A}$ and $\mathbf{B}$ must be regular; thus, $\det(\mathbf{A}) \neq 0$ and $\det(\mathbf{B}) \neq 0$. The compositions $A \circ S^{-1}$ and $T^{-1} \circ B$ can be computed simply as $\mathbf{A}\mathbf{S}^{-1}$ and $\mathbf{T}^{-1}\mathbf{B}$. The composition $A \circ F \circ B$ is discussed in the following sections.

3.1. Composition $A \circ F$

Let $A : \mathbb{F}^m \to \mathbb{F}^m, \mathbf{h} \mapsto A(\mathbf{h}) = \mathbf{A}\mathbf{h}$, where $\mathbf{A} \in \mathbb{F}^{m \times m}$ is a regular matrix. The composition $A \circ F$ is then

$$(A \circ F)(\mathbf{x}) = \mathbf{A} \begin{pmatrix} f^{(v_1+1)}(\mathbf{x}) \\ f^{(v_1+2)}(\mathbf{x}) \\ \vdots \\ f^{(n)}(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m} A_{1,i} f^{(v_1+i)}(\mathbf{x}) \\ \sum_{i=1}^{m} A_{2,i} f^{(v_1+i)}(\mathbf{x}) \\ \vdots \\ \sum_{i=1}^{m} A_{m,i} f^{(v_1+i)}(\mathbf{x}) \end{pmatrix}, \tag{14}$$

and $\forall k \in \{1, \ldots, m\}$:

$$[A \circ F]_k(\mathbf{x}) = \sum_{i=1}^{m} A_{k,i} \cdot f^{(v_1+i)}(\mathbf{x}) = \sum_{i=1}^{m} A_{k,i} \cdot (\mathbf{x}^\mathsf{T} \mathbf{F}^{(v_1+i)} \mathbf{x}) = \mathbf{x}^\mathsf{T} \sum_{i=1}^{m} (A_{k,i} \cdot \mathbf{F}^{(v_1+i)}) \mathbf{x}. \quad (15)$$

The composition is equivalent to a linear combination of **F** matrices. In the following equation, we show how arbitrary **A** affects the structure of the central map:

$$\sum_{i=1}^{m} A_{k,i} \cdot \mathbf{F}^{(v_1+i)} = \sum_{i=1}^{m} A_{k,i} \cdot \begin{pmatrix} \mathbf{F}_{1,1}^{(v_1+i)} & \mathbf{F}_{1,2}^{(v_1+i)} & \mathbf{F}_{1,3}^{(v_1+i)} \\ \mathbf{0} & \mathbf{F}_{2,2}^{(v_1+i)} & \mathbf{F}_{2,3}^{(v_1+i)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}. \quad (16)$$

To maintain the special structure for the first layer ($k \in \{1, \ldots, o_1\}$), as described in Equation (5), we must fulfill the following restrictions:

$$\begin{aligned} \sum_{i=1}^{m} A_{k,i} \cdot \mathbf{F}_{1,3}^{(v_1+i)} &= \mathbf{0}, \\ \sum_{i=1}^{m} A_{k,i} \cdot \mathbf{F}_{2,2}^{(v_1+i)} &= \mathbf{0}, \\ \sum_{i=1}^{m} A_{k,i} \cdot \mathbf{F}_{2,3}^{(v_1+i)} &= \mathbf{0}. \end{aligned} \quad (17)$$

Since, $\forall i \in O_1 : \mathbf{F}_{1,3}^{(i)}, \mathbf{F}_{2,2}^{(i)}, \mathbf{F}_{2,3}^{(i)}$ are zero matrices, Equation (17) can be rewritten as

$$\begin{aligned} \sum_{i=o_1+1}^{m} A_{k,i} \cdot \mathbf{F}_{1,3}^{(v_1+i)} &= \mathbf{0}, \\ \sum_{i=o_1+1}^{m} A_{k,i} \cdot \mathbf{F}_{2,2}^{(v_1+i)} &= \mathbf{0}, \\ \sum_{i=o_1+1}^{m} A_{k,i} \cdot \mathbf{F}_{2,3}^{(v_1+i)} &= \mathbf{0}. \end{aligned} \quad (18)$$

This restriction can be trivially satisfied by the following condition:

$$(\forall k \in \{1, \ldots, o_1\})(\forall i \in \{o_1 + 1, \ldots, m\}) : A_{k,i} = 0. \quad (19)$$

Regarding the second layer, the arbitrary A maintains its polynomial structure by itself, and therefore, no further restrictions are necessary.

An applicable matrix $\mathbf{A} \in \mathbb{F}^{m \times m}$ is therefore

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{0} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{pmatrix}, \quad (20)$$

where $\mathbf{A}_{1,1} \in \mathbb{F}^{o_1 \times o_1}$, $\mathbf{A}_{2,1} \in \mathbb{F}^{o_2 \times o_1}$, $\mathbf{A}_{2,2} \in \mathbb{F}^{o_2 \times o_2}$. Matrix **A** must be regular; therefore, $\mathbf{A}_{1,1}$ and $\mathbf{A}_{2,2}$ are arbitrary regular matrices and $\mathbf{A}_{1,2}$ is an (possibly singular) arbitrary matrix.

From the matrix **A**, we deduce that the linear combination is performed separately in layers ($\mathbf{A}_{1,1}$ and $\mathbf{A}_{2,2}$), and quadratic polynomials from the first layer can be combined into the second layer ($\mathbf{A}_{2,1}$).

3.2. Composition $F \circ B$

Let $B : \mathbb{F}^n \to \mathbb{F}^n, \mathbf{x} \mapsto B(\mathbf{x}) = \mathbf{B}\mathbf{x}$, where $\mathbf{B} \in \mathbb{F}^{n\times n}$ is a regular matrix. The composition $F \circ B$ is then

$$(F \circ B)(\mathbf{x}) = F \circ (\mathbf{B}\mathbf{x}) = \begin{pmatrix} f^{(v_1+1)}(\mathbf{B}\mathbf{x}) \\ f^{(v_1+2)}(\mathbf{B}\mathbf{x}) \\ \vdots \\ f^{(n)}(\mathbf{B}\mathbf{x}) \end{pmatrix}, \tag{21}$$

and, $\forall k \in \{1, \ldots, m\}$:

$$[(F \circ B)]_k(\mathbf{x}) = f^{(v_1+k)}(\mathbf{B}\mathbf{x}) = (\mathbf{B}\mathbf{x})^\intercal \mathbf{F}^{(v_1+k)}\mathbf{B}\mathbf{x} = \mathbf{x}^\intercal(\mathbf{B}^\intercal\mathbf{F}^{(v_1+k)}\mathbf{B})\mathbf{x}. \tag{22}$$

This composition changes every quadratic polynomial separately. Before we consider the restrictions on the matrix $\mathbf{B}$, let us first introduce the following lemma.

Lemma 1. *Every quadratic form can be represented as an upper triangular matrix.*

Proof. Let $q \in \mathbb{F}[\mathbf{x}]$ be a quadratic form represented by matrix $\mathcal{Q}$, then $\mathcal{U}$ is an upper triangular representation of q, where $\mathcal{U}$ is

$$\forall i,j \in \{1,\ldots,n\} : \mathcal{U}_{i,j} = \begin{cases} \mathcal{Q}_{i,i} & i = j, \\ \mathcal{Q}_{i,j} + \mathcal{Q}_{j,i} & i < j, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

Then,

$$q(\mathbf{x}) = \mathbf{x}^\intercal\mathcal{Q}\mathbf{x} = \sum_{i,j\in\{1,\ldots,n\}} \mathcal{Q}_{i,j}x_ix_j = \sum_{i\in\{1,\ldots,n\}} \mathcal{Q}_{i,i}x_i^2 + \sum_{\substack{i,j\in\{1,\ldots,n\}\\ i<j}} (\mathcal{Q}_{i,j} + \mathcal{Q}_{j,i})x_ix_j = \mathbf{x}^\intercal\mathcal{U}\mathbf{x}. \tag{24}$$

$\square$

Let $\triangledown : \mathbb{F}^{n\times n} \to \mathbb{F}^{n\times n}; \mathcal{Q} \mapsto \mathcal{U}$ be a function, where $\mathcal{U}$ is an upper triangular matrix, such that $\forall \mathbf{x} \in \mathbb{F}^n : \mathbf{x}^\intercal\mathcal{Q}\mathbf{x} = \mathbf{x}^\intercal\mathcal{U}\mathbf{x}$.

In Equation (25), we show how matrix $\mathbf{B}$ changes the structure of a polynomial in the central map. We used the same partitioning of the matrix $\mathbf{B}$ as we used for the matrix $\mathbf{F}^{(k)}$. Size compatibility for the matrix multiplication is guaranteed thanks to the symmetry of block sizes. We further applied the function $\triangledown$, as it does not change the concerned polynomial and it helps us with the wanted structure. It ensures the upper triangular submatrices on the diagonal, which are required by the definition of Rainbow.

$$\begin{aligned} \mathbf{B} &= \begin{pmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} & \mathbf{B}_{1,3} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} & \mathbf{B}_{2,3} \\ \mathbf{B}_{3,1} & \mathbf{B}_{3,2} & \mathbf{B}_{3,3} \end{pmatrix}, \\ k \in O_1 : \triangledown(\mathbf{B}^\intercal\mathbf{F}^{(k)}\mathbf{B}) &= \begin{pmatrix} \mathbf{F}'^{(k)}_{1,1} & \mathbf{F}'^{(k)}_{1,2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \\ k \in O_2 : \triangledown(\mathbf{B}^\intercal\mathbf{F}^{(k)}\mathbf{B}) &= \begin{pmatrix} \mathbf{F}'^{(k)}_{1,1} & \mathbf{F}'^{(k)}_{1,2} & \mathbf{F}'^{(k)}_{1,3} \\ \mathbf{0} & \mathbf{F}'^{(k)}_{2,2} & \mathbf{F}'^{(k)}_{2,3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}. \end{aligned} \tag{25}$$

Solving this system of equations with matrix $\mathbf{B}$ as a variable for arbitrary central map $\mathbf{F}$, where the right sides of the equations are zeroes, we obtain:

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} & \mathbf{0} \\ \mathbf{B}_{3,1} & \mathbf{B}_{3,2} & \mathbf{B}_{3,3} \end{pmatrix}, \tag{26}$$

where non-zero submatrices are arbitrary submatrices, except that matrix $\mathbf{B}$ must be regular, and therefore, submatrices $\mathbf{B}_{1,1}, \mathbf{B}_{2,2}, \mathbf{B}_{3,3}$ must be regular.

For an insight into what is happening to the vector $\mathbf{x}$ (which elements are the variables in the central map), we can partition $\mathbf{x}$ into three different types of variables: vinegar variables of the first layer ($\mathbf{x}_1^v \in \mathbb{F}^{v_1}$), oil variables of the first layer ($\mathbf{x}_1^o \in \mathbb{F}^{o_1}$), and oil variables of the second layer ($\mathbf{x}_2^o \in \mathbb{F}^{o_2}$). If we apply matrix $\mathbf{B}$ to the vector $\mathbf{x}$, we obtain:

$$\mathbf{Bx} = \begin{pmatrix} \mathbf{B}_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} & \mathbf{0} \\ \mathbf{B}_{3,1} & \mathbf{B}_{3,2} & \mathbf{B}_{3,3} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1^v \\ \mathbf{x}_1^o \\ \mathbf{x}_2^o \end{pmatrix} = \begin{pmatrix} \mathbf{B}_{1,1}\mathbf{x}_1^v \\ \mathbf{B}_{2,1}\mathbf{x}_1^v + \mathbf{B}_{2,2}\mathbf{x}_1^o \\ \mathbf{B}_{3,1}\mathbf{x}_1^v + \mathbf{B}_{3,2}\mathbf{x}_1^o + \mathbf{B}_{3,3}\mathbf{x}_2^o \end{pmatrix}. \tag{27}$$

In this situation, the variables to solve are not defined only by vector $\mathbf{x}$, but the vector $\mathbf{Bx}$. To be able to sign, it is necessary not to mix the vinegar variable with the oil variables of each layer. At the beginning of the signing process, we used $\mathbf{B}_{1,1}\mathbf{x}_1^v$ instead of $\mathbf{x}_1^v$. Vector $\mathbf{x}_1^v$ is randomly generated, thus known. We can easily compute $\mathbf{B}_{1,1}\mathbf{x}_1^v$. Next, the first layer is solved, where we computed $\mathbf{B}_{2,1}\mathbf{x}_1^v + \mathbf{B}_{2,2}\mathbf{x}_1^o$ instead of $\mathbf{x}_1^o$. We can immediately substitute $\mathbf{B}_{2,1}\mathbf{x}_1^v$, which is already known. We still obtain a system of the linear equations that has only variables of the vector $\mathbf{x}_1^o$. In the end, the second layer is solved. We can first substitute $\mathbf{B}_{3,1}\mathbf{x}_1^v + \mathbf{B}_{3,2}\mathbf{x}_1^o$, and again, we obtain only a system of linear equations. No oil and vinegar variables (of the same layer) are mixed. This was only an insight into why the inversion of the central map can be still computed. In the implementation, matrix $\mathbf{B}$ is incorporated in the central map (as a generator of the equivalent key).

3.3. Analysis of an Equivalent Key

In this section, we discuss the basic properties of the equivalent key. We concentrate only on equivalent keys that can be generated according to Equation (13), with the maps A and B defined in Equation (20) and Equation (26). We start with the number of equivalent keys, and we calculate entropy where applicable. Then, we state the benefits of our masking scheme. For the analysis in this section, we considered two sets of Rainbow parameters:

- **Ia** (128 bit security): $\mathbb{F}_q = GF(16), v_1 = 36, o_1 = o_2 = 32$;
- **Vc** (256 bit security): $\mathbb{F}_q = GF(256), v_1 = 96, o_1 = 36$ and $o_2 = 64$.

We note that these parameters, while suggested in the NIST standardization process, are probably already inadequate due to the recently presented attacks [30].

Lemma 2. *Let $M_{l,s}(\mathbb{F}_q) = \{l \times s \text{ matrices over } \mathbb{F}_q\}$ be a set. The cardinality of the set is $|M_{l,s}(\mathbb{F}_q)| = q^{l \cdot s}$.*

Lemma 3. *Let $M_n(\mathbb{F}_q) = \{n \times n \text{ matrices over } \mathbb{F}_q\}$, with the matrix multiplication, be the full linear monoid. The order of the monoid is $|M_n(\mathbb{F}_q)| = q^{n^2}$.*

Lemma 4. *Let $GL_n(\mathbb{F}_q) = \{n \times n \text{ invertible matrices over } \mathbb{F}_q\}$, with the matrix multiplication, be the general linear group. The order is $|GL_n(\mathbb{F}_q)| = \prod_{k=0}^{n-1}(q^n - q^k) = q^{n^2}(q^{-n};q)_n$. (q-Pochhammer symbol: $(a;q)_n := \prod_{k=0}^{n-1}(1 - aq^k), n > 0$, defined inter alia in [33]. For example: the number of 10×10 regular matrices over $GF(16)$ is $|GL_{10}(\mathbb{F}_{16})| \approx 0.9336 \cdot 16^{100}$, where $(16^{-10};16)_{10} \approx 0.9336$. Symbol $(q^{-n},q)_n$ denotes the ratio between regular matrices $n \times n$ and all matrices $n \times n$ over $\mathbb{F}_q$.)*

The proofs of these three lemmas are well known [34].

Theorem 1. *The number of different equivalent keys, over a fixed secret key SK, is*

$$q^{v_1(v_1+o_1+o_2)+2(o_1^2+o_1o_2+o_2^2)}(q^{-v_1};q)_{v_1}((q^{-o_1};q)_{o_1})^2((q^{-o_2};q)_{o_2})^2. \tag{28}$$

This is an exact number, where the left part $q^{(\cdots)}$ describes the number of different combinations of matrices $\mathbf{A}$ and $\mathbf{B}$. The remaining part counts for the cases where $\mathbf{A}$ and $\mathbf{B}$ are regular. For GF(16), its value is ≈ 0.7092, and for GF(256) it is ≈ 0.9805. This value is dependent on the values v_1, o_1, o_2, but it converges with these parameters extremely fast. For GF(16), the difference between $v_1 = o_1 = o_2 = 10$ and $v_1 = o_1 = o_2 = 1000$ is less then 10^{-12}.

Proof. (Number of different equivalent keys) We show that the number of equivalent keys is the number of possible non-equal matrices $\mathbf{A}$ multiplied by the number of possible non-equal matrices $\mathbf{B}$. This holds because both matrices are applied to different parts of the secret key (maps S and T), so they cannot interfere with each other, and they are both generated independently. We discuss the number of keys for matrix $\mathbf{A}$ only, and the same procedure can be applied to matrix $\mathbf{B}$. Let $\mathbb{A}$ be the set of matrices with the structure defined in Equation (20) and a regular matrix $\mathbf{S} \in \mathrm{GL}_m(\mathbb{F}_q)$ be the representation of the linear map S.

First, we show that the lower bound of the number of equivalent keys generated by $\mathbb{A}$ is equal to the cardinality of the set $\mathbb{A}$. This is because a regular matrix $\mathbf{S}$ is multiplied by regular matrices from the set $\mathbb{A}$. Thus, two distinct matrices $\mathbf{A}, \mathbf{A}' \in \mathbb{A} \subset \mathrm{GL}$ cannot generate the same product matrix:

$$\forall \mathbf{A}, \mathbf{A}' \in \mathrm{GL}_m(\mathbb{F}_q) : \mathbf{A} \neq \mathbf{A}' \implies \mathbf{AS} \neq \mathbf{A}'\mathbf{S}. \tag{29}$$

Therefore, each distinct matrix in $\mathbb{A}$ generates a different equivalent key; hence, the cardinality of the set $\mathbb{A}$ is the lower bound of the number of different equivalent keys generated by $\mathbb{A}$.

Second, we show that the cardinality of the set $\mathbb{A}$ is also the upper bound of the number of different equivalent keys generated by $\mathbb{A}$, since every equivalent key can be reached by a multiplication with a single $\mathbf{A} \in \mathbb{A}$. Let us examine a product of two matrices $\mathbf{A}, \mathbf{A}' \in \mathbb{A}$:

$$\mathbf{AA}' = \begin{pmatrix} \mathbf{A}_{1,1}\mathbf{A}'_{1,1} & \mathbf{0} \\ \mathbf{A}_{2,1}\mathbf{A}'_{1,1} + \mathbf{A}_{2,2}\mathbf{A}'_{2,1} & \mathbf{A}_{2,2}\mathbf{A}'_{2,2} \end{pmatrix}. \tag{30}$$

The matrix $\mathbf{AA}'$ has the same structure as the matrices $\mathbf{A}$ and $\mathbf{A}'$. On the diagonal, we have the products of the elements of GL, which also result in an element of the GL, and the submatrix $[\mathbf{AA}']_{2,1}$ can be an arbitrary matrix, thus $\mathbf{AA}' \in \mathbb{A}$. In other words, every key reachable by subsequent multiplications with two matrices from $\mathbb{A}$ can be reached by a single multiplication with some matrix from $\mathbb{A}$:

$$(\forall \mathbf{A}, \mathbf{A}' \in \mathbb{A})(\exists \mathbf{A}'' \in \mathbb{A}) : \mathbf{A}'(\mathbf{AS}) = \mathbf{A}''\mathbf{S}. \tag{31}$$

Therefore, the upper bound of the number of equivalent keys generated by $\mathbb{A}$ is the size of the set $\mathbb{A}$.

The previous statements imply that the number of equivalent keys generated by $\mathbb{A}$ is equal to the size of the set $\mathbb{A}$. Every matrix $\mathbf{A} \in \mathbb{A}$ has four parts: two submatrices from GL, one from $M_{o_1,o_2}(\mathbb{F}_q)$, and a zero submatrix. All matrices are independent, so the total number of matrices is the product of the numbers of different submatrices:

$$\begin{aligned} &|\mathrm{GL}_{o_1}(\mathbb{F}_q)| \cdot |\mathrm{GL}_{o_2}(\mathbb{F}_q)| \cdot |M_{o_1,o_2}(\mathbb{F}_q)| \cdot 1 = \\ &= q^{(o_1^2+o_2^2+o_1o_2)}(q^{-o_1};q)_{o_1}(q^{-o_2};q)_{o_2}. \end{aligned} \tag{32}$$

The number of different matrices **B** can be computed in a similar fashion:

$$\begin{aligned}&|\mathrm{GL}_{v_1}(\mathbb{F}_q)| \cdot |\mathrm{GL}_{o_1}(\mathbb{F}_q)| \cdot |\mathrm{GL}_{o_2}(\mathbb{F}_q)| \cdot |M_{v_1,o_1}(\mathbb{F}_q)| \cdot |M_{v_1,o_2}(\mathbb{F}_q)| \cdot |M_{o_1,o_2}(\mathbb{F}_q)| = \\ &= q^{(v_1^2+o_1^2+o_2^2+v_1o_1+v_1o_2+o_1o_2)}(q^{-v_1};q)_{v_1}(q^{-o_1};q)_{o_1}(q^{-o_2};q)_{o_2}.\end{aligned} \tag{33}$$

By multiplying the results in Equation (32) and Equation (33), we obtain the total number of equivalent keys, as stated in Equation (28). $\square$

The number of equivalent keys is approximately $16^{9744} \cdot 0.7097 \approx 2^{38976}$ and $256^{34208} \cdot 0.9805 \approx 2^{273664}$ for the Ia and Vc parameters, respectively. If the matrices **A** and **B** are generated from a uniform distribution, the entropy of the generated equivalent key for the Ia and Vc parameters is ≈38,976 Sh and ≈273,664 Sh, respectively.

The beneficial property of an equivalent key scheme is an option to forget the original key and keep only an equivalent key. We were able to generate an equivalent key from an already-computed equivalent key. The equivalent key (or more of them) can be generated and prepared at any time prior to the signing process. The signing process itself then has no overhead.

4. Efficient Implementation

In Section 3, we discuss the equivalent key defined by matrices **A** and **B**. However, we propose choosing only a subset of these matrices with respect to:

- Calculation performance;
- Amount of fresh randomness necessary;
- Entropy of the generated equivalent key;
- Implementation size and simplicity;
- Total number of keys equivalent to each public key.

With respect to the aforementioned desires, we propose using the following submatrices to generate the equivalent keys:

$$\mathcal{M}_n := \left\{ \mathbf{M} = \begin{pmatrix} 1 & m_1 & 0 & \dots & 0 \\ 0 & 1 & m_2 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_n & 0 & 0 & \dots & 1 \end{pmatrix} \mid \det(\mathbf{M}) \neq 0 \right\}, \tag{34}$$

where $m_1, \dots, m_n \in \mathbb{F}_q$, and the condition for regularity is

$$\det(\mathbf{M}) \neq 0 \iff \prod_{i=1}^{n} m_i \neq (-1)^{n+1}. \tag{35}$$

The equivalent key generator is a tuple $(\mathbf{A}, \mathbf{B})$ defined as

$$\mathbf{A} := \begin{pmatrix} \mathbf{A}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{(2)} \end{pmatrix}, \ \mathbf{B} := \begin{pmatrix} \mathbf{B}^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^{(2)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}^{(3)} \end{pmatrix}, \tag{36}$$

where $\mathbf{A}^{(1)}, \mathbf{B}^{(2)} \in \mathcal{M}_{o_1}, \mathbf{A}^{(2)}, \mathbf{B}^{(3)} \in \mathcal{M}_{o_2}, \mathbf{B}^{(1)} \in \mathcal{M}_{v_1}$. The proposed generator form is further justified in Section 4.1. The generator matrices are defined as

$$\mathbf{A}_{k,l} = \begin{cases} 1 & k = l, \\ a_k & (k \in \{1, \ldots, o_1\}) \wedge (l = |k+1|_{o_1}), \\ a_k & (k \in \{o_1+1, \ldots, m\}) \wedge (l = |k - o_1 + 1|_{o_2} + o_1), \\ 0 & \text{otherwise}, \end{cases}$$
$$\mathbf{B}_{k,l} = \begin{cases} 1 & k = l, \\ b_k^{(1)} & (k \in \{1, \ldots, v_1\}) \wedge (l = |k+1|_{v_1}), \\ b_{k-v_1}^{(2)} & (k \in \{v_1+1, \ldots, v_2\}) \wedge (l = |k - v_1 + 1|_{o_1} + v_1), \\ b_{k-v_2}^{(3)} & (k \in \{v_2+1, \ldots, n\}) \wedge (l = |k - v_2 + 1|_{o_2} + v_2), \\ 0 & \text{otherwise}, \end{cases} \tag{37}$$

where $|a|_b := (a - 1 \bmod b) + 1$ is the modulo operation with offset (e.g., it holds that $|m|_m = m$ and $|m+1|_m = 1$). Indexing from zero would result in using a regular modulo operation instead. Matrices $\mathbf{A}$ and $\mathbf{B}$ can be stored as vectors of their non-trivial values:

$$(a_1, a_2, \ldots, a_m), (b_1^{(1)}, \ldots, b_{v_1}^{(1)}, b_1^{(2)}, \ldots, b_{o_1}^{(2)}, b_1^{(3)}, \ldots, b_{o_2}^{(3)}). \tag{38}$$

For further analysis, in Equation (39), we show the probability that the randomly generated matrix from the definition in Equation (34) (i.e., the matrix with generated values $m_1, \ldots, m_n$) over $\mathbb{F}_q$ is regular. In Equation (39), we omit the set $\{1, \ldots, n\}$ in all quantifiers $\forall i \in \{1, \ldots, n\}$ and $\exists i \in \{1, \ldots, n\}$ since it is always identical, and we write only $\forall i$ and $\exists i$ for simplicity.

$$\begin{aligned} &\mathrm{P}(\det(\mathbf{M}) \neq 0 \mid \mathbf{M} \in \mathcal{M}_n) = \\ &= \mathrm{P}\left(\exists i : m_i = 0 \vee \left(\forall i : m_i \neq 0 \wedge \prod_{i=1}^{n} m_i \neq (-1)^{n+1}\right)\right) \\ &= \mathrm{P}(\exists i : m_i = 0) + \mathrm{P}\left(\forall i : m_i \neq 0 \wedge \prod_{i=1}^{n} m_i \neq (-1)^{n+1}\right) \\ &= 1 - \mathrm{P}(\forall i : m_i \neq 0) + \mathrm{P}\left(\prod_{i=1}^{n} m_i \neq (-1)^{n+1} \,\middle|\, \forall i : m_i \neq 0\right) \cdot \mathrm{P}(\forall i : m_i \neq 0) \\ &= 1 - \left(\frac{q-1}{q}\right)^n + \left(\frac{q-2}{q-1}\right)\left(\frac{q-1}{q}\right)^n \\ &= 1 - \left(\frac{1}{q-1}\right)\left(\frac{q-1}{q}\right)^n. \end{aligned} \tag{39}$$

The non-zero values of the submatrices in $\mathbf{A}$ and $\mathbf{B}$ are to be generated independently from a uniform random distribution, so we can easily express the probability, e.g., a randomly generated matrix $\mathcal{M}_{32}$ over $GF(16)$ is regular with probability 99.15% and matrix $\mathcal{M}_{64}$ over $GF(256)$ with probability 99.69%.

Using the regularity probability, we can express the cardinality of $\mathcal{M}_n$. This can be performed by multiplying the number of all possible matrices by the probability of each matrix over $\mathbb{F}_q$ being regular:

$$|\mathcal{M}_n| = q^n - (q-1)^{n-1}. \tag{40}$$

4.1. Justification of the Selected Generators

In this subsection, we justify the proposed form of our efficient equivalent key generators with respect to the desires mentioned in Section 4.

We propose the form of generators in Equation (34), i.e., the ones on the diagonal and the only non-trivial elements adjacent to it, so that vector–matrix (and analogically, matrix–matrix) multiplication can be efficiently performed as described by formula:

$$\mathbf{M} \in \mathcal{M}_n : \mathbf{M} \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix} = (\mathbf{I} + \widetilde{\mathbf{M}}) \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix} + \begin{pmatrix} m_1 g_2 \\ m_2 g_3 \\ \vdots \\ m_n g_1 \end{pmatrix}. \tag{41}$$

Based on Equation (41), the vector–matrix multiplication $\mathbf{M} \cdot \mathbf{g}$ can be performed simply by copying the vector $\mathbf{g}$ to the result variable, then cyclically shifting the vector $\mathbf{g}$ and adding an elementwise product of the vector $\mathbf{g}$ and the vector of non-trivial values of the matrix $\mathbf{M}$ (as listed in Equation (38)).

Next, we discuss a number of different equivalent keys for our efficient generators. This is equal to the number of possible distinct linear maps of generators, in other words, the number of different matrices that can be obtained by multiplying matrices in $\mathcal{M}_n$ by each other. We believe that these matrices generate the whole group GL_n for $n > 2$, but we were not able to prove this claim in general (we managed to prove the claim for matrices 3×3, 4×4, 5×5 over $\mathbb{Z}_2$, then for 3×3, 4×4 over $\mathbb{Z}_3$, and also, for matrices 3×3 over $\mathbb{Z}_5$ by brute force). Assuming that the following formula holds over a fixed $\mathbb{F}_q$ and for every integer $n > 2$:

$$\begin{aligned} &\forall \mathbf{M} \in \mathrm{GL}_n, \exists k \in \mathbb{N}, \exists \mathbf{M}_1, \ldots, \mathbf{M}_k \in \mathcal{M}_n : \prod_{i=1}^{k} \mathbf{M}_i = \mathbf{M} \\ &\implies \mathrm{card}(\{\text{matrices generated from } \mathcal{M}_n\}) = \mathrm{ord}(\mathrm{GL}_n), \end{aligned} \tag{42}$$

the number of equivalent keys using the proposed efficient generators, with the possibility of generating an equivalent key from another equivalent key, would be

$$N = q^{v_1^2 + 2o_1^2 + 2o_2^2} (q^{-v_1}; q)_{v_1} (q^{-o_1}; q)_{o_1}^2 (q^{-o_2}; q)_{o_2}^2. \tag{43}$$

For parameters Ia and Vc, respectively, the number of transitively reachable equivalent keys (i.e., generating the equivalent key from another equivalent key) would be approximately $16^{5392} \cdot 0.7092 \approx 2^{21567}$ and $256^{20000} \cdot 0.9805 \approx 2^{160000}$, respectively.

The number of possible distinct equivalent keys after one generating process is $|\mathcal{M}_{v_1}| \cdot |\mathcal{M}_{o_1}|^2 \cdot |\mathcal{M}_{o_2}|^2$; for parameters Ia and Vc, respectively, the number of possible equivalent keys after one generation is approximately 2^{656} and 2^{2368}, respectively. Assuming the equivalent key generators are sampled from a uniform distribution, the entropy of the next-generated equivalent key is approximately 656 Sh and 2368 Sh, respectively. Given this entropy, we believe that the probability of the signer using the same equivalent key multiple times is negligible. The number of distinct equivalent keys is summarized in Table 1.

Table 1. Summary of equivalent key variants.

Generator	**Log$_2$ of Number of Equivalent Keys**			
	Single Generated Key		**Transitively Reachable**	
	Ia	**Vc**	**Ia**	**Vc**
General	38,976	273,664	38,976	273,664
Efficient	656	2368	21,567	160,000

We further discuss the number of distinct linear maps generated in equivalent keys, as these are typical targets in a side-channel attack scenario [20,21]. The number of distinct maps S generated by the proposed efficient generator A over a single class of equivalent keys is

$$q^{o_1^2+o_2^2}(q^{-o_1};q)_{o_1}(q^{-o_2};q)_{o_2}. \tag{44}$$

For parameters Ia and Vc, respectively, the number of transitively reachable linear maps S is approximately $16^{2048} \cdot 0.8716 \approx 2^{8192}$ and $256^{5392} \cdot 0.9922 \approx 2^{43136}$, respectively. After one generating process, the number of different linear maps S is $|\mathcal{M}_{o_1}| \cdot |\mathcal{M}_{o_2}|$, that is almost 2^{256} for the Ia parameters and almost 2^{800} for the Vc parameters.

Lastly, we discuss the required fresh randomness. The generators **A** and **B** defined in Equations (20) and (26) use a quadratic number of random elements in regard to parameters m and n. For Rainbow parameters Ia and Vc, respectively, this corresponds to ≈39 kb and ≈274 kb of fresh randomness, respectively. The number of required random elements is linear for our efficient generators defined in Equation (36), leaving us with only 656 bits and 2368 bits of fresh randomness, respectively. This amount of randomness refers to the case when the matrices **A** and **B** are successfully generated according to the requirements, i.e., the generated matrices are regular (the probability of the randomly generated matrices **A** and **B** being regular is 96 % for the Ia parameters and 98.4 % for the Vc parameters).

4.2. *Efficient Computation of Equivalent Keys*

Compositions $A \cdot S^{-1}$ and $T^{-1} \cdot B$ can be computed in the same way as the multiplication in Equation (41).

The composition of $A \circ F$, where A is represented as **A** in Equation (36), is:

$$\forall k \in \{1, \ldots, o_1\} : [A \circ F]_k \leftrightarrow \mathbf{F}^{(v_1+k)} + a_k \mathbf{F}^{(v_1+|k+1|_{o_1})}, \tag{45}$$

$$\forall k \in \{o_1+1, \ldots, m\} : [A \circ F]_k \leftrightarrow \mathbf{F}^{(v_1+k)} + a_k \mathbf{F}^{(v_1+|k-o_1+1|_{o_2}+o_1)}. \tag{46}$$

The composition of $F \circ B$, where B is represented as **B** in Equation (36), is

$$k \in \{v_1, \ldots, n\} : [F \circ B]_k \leftrightarrow \triangledown(\mathbf{B}^\intercal \mathbf{F}^{(k)} \mathbf{B}) = \begin{pmatrix} \triangledown\Big((\mathbf{B}^{(1)})^\intercal \mathbf{F}_{1,1}^{(k)} \mathbf{B}^{(1)}\Big) & (\mathbf{B}^{(1)})^\intercal \mathbf{F}_{1,2}^{(k)} \mathbf{B}^{(2)} & (\mathbf{B}^{(1)})^\intercal \mathbf{F}_{1,3}^{(k)} \mathbf{B}^{(3)} \\ \mathbf{0} & \triangledown\Big((\mathbf{B}^{(2)})^\intercal \mathbf{F}_{2,2}^{(k)} \mathbf{B}^{(2)}\Big) & (\mathbf{B}^{(2)})^\intercal \mathbf{F}_{2,3}^{(k)} \mathbf{B}^{(3)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{47}$$

Upper triangular matrices $\triangledown\Big((\mathbf{B}^{(1)})^\intercal \mathbf{F}_{1,1}^{(k)} \mathbf{B}^{(1)}\Big)$ and $\triangledown\Big((\mathbf{B}^{(2)})^\intercal \mathbf{F}_{2,2}^{(k)} \mathbf{B}^{(2)}\Big)$ in Equation (47) can be efficiently evaluated as described in Section 4.2.1. The remaining parts of Equation (47) consist of matrix multiplications $(\mathbf{B}^{(u)})^\intercal \mathbf{F}_{u,v}^{(k)} \mathbf{B}^{(v)}$, where $(u, v) \in \{(1,2), (1,3), (2,3)\}$. This expression can be computed as:

$$[(\mathbf{B}^{(u)})^\intercal \mathbf{F}_{u,v}^{(k)} \mathbf{B}^{(v)}]_{r,c} = [\mathbf{F}_{u,v}^{(k)}]_{r,c} + b_{|r-1|_s}^{(u)} [\mathbf{F}_{u,v}^{(k)}]_{|r-1|_s,c} + b_{|c-1|_w}^{(v)} [\mathbf{F}_{u,v}^{(k)}]_{r,|c-1|_w} + b_{|r-1|_s}^{(u)} b_{|c-1|_w}^{(v)} [\mathbf{F}_{u,v}^{(k)}]_{|r-1|_s,|c-1|_w}, \tag{48}$$

where $\mathbf{B}^{(u)} \in \mathcal{M}_s$, $\mathbf{B}^{(v)} \in \mathcal{M}_w$, and $\mathbf{F}_{u,v}^{(k)} \in \mathbb{F}^{s \times w}$ is an arbitrary matrix. If the elements are stored sequentially by index k, the computation can be accelerated using word-level parallelism and vector processing.

4.2.1. Algorithm for Upper Triangular Matrices' Evaluation

Algorithm 1 was deduced as described in the following text. First, we rewrote output $\triangledown\Big((\mathbf{B}^{(u)})^\intercal \mathbf{F}_{u,u}^{(k)} \mathbf{B}^{(u)}\Big)$, without writing free variables u and k, as $\triangledown(\mathbf{B}^\intercal \mathbf{F} \mathbf{B})$, where **F** is an upper triangular matrix in $\mathbb{F}^{s \times s}$ and **B** is a matrix from the set $\mathcal{M}_s \subset \mathbb{F}^{s \times s}$, which is defined using the vector of its non-trivial values $(b_1, \ldots, b_s)$. Then,

$$[\triangledown(\mathbf{B}^\intercal \mathbf{F} \mathbf{B})]_{r,c} = \begin{cases} [\mathbf{B}^\intercal \mathbf{F} \mathbf{B}]_{r,r} & r = c, \\ [\mathbf{B}^\intercal \mathbf{F} \mathbf{B}]_{r,c} + [\mathbf{B}^\intercal \mathbf{F} \mathbf{B}]_{c,r} & r < c, \\ 0 & r > c. \end{cases} \tag{49}$$

By expanding matrix multiplication, we obtain

$$[\mathbf{B}^\intercal\mathbf{F}\mathbf{B}]_{r,c} = \mathbf{F}_{r,c} + b_{|r-1|_s}\mathbf{F}_{|r-1|_s,c} + b_{|c-1|_s}\mathbf{F}_{r,|c-1|_s} + b_{|r-1|_s}b_{|c-1|_s}\mathbf{F}_{|r-1|_s,|c-1|_s}. \tag{50}$$

Specifically, for $r = c$, we obtain

$$[\mathbf{B}^\intercal\mathbf{F}\mathbf{B}]_{r,r} = \mathbf{F}_{r,r} + b_{|r-1|_s}(\mathbf{F}_{|r-1|_s,r} + \mathbf{F}_{r,|r-1|_s}) + b^2_{|r-1|_s}\mathbf{F}_{|r-1|_s,|r-1|_s}. \tag{51}$$

One term of $(\mathbf{F}_{|r-1|_s,r} + \mathbf{F}_{r,|r-1|_s})$ is zero, as matrix $\mathbf{F}$ is upper triangular and $|r-1|_s \neq r$.

Algorithm 1 Computation of $[\triangledown\Big((\mathbf{B}^{(u)})^\intercal\mathbf{F}^{(k)}_{u,u}\mathbf{B}^{(u)}\Big)]_{r,c}$.

```
input: matrix $\mathbf{B}^{(u)} \in \mathcal{M}_s$ as vector $(b_1, \dots, b_s), u \in \{1,2\}$
       upper triangular matrix $\mathbf{F}^{(k)}_{u,u} \in \mathbb{F}^{s\times s}, k \in O_1 \cup O_2$
       integers $r, c \in \{1, \dots, s\}$, where $r \le c$
output: element $[\triangledown\Big((\mathbf{B}^{(u)})^\intercal\mathbf{F}^{(k)}_{u,u}\mathbf{B}^{(u)}\Big)]_{r,c}$
 1: $\mathbf{F}_{i,j} := [\mathbf{F}^{(k)}_{u,u}]_{i,j}, \forall i,j \in \{1,\dots,s\}$  (for brevity)
 2: $m := c - r$
 3: $r_1 := |r-1|_s$
 4: $c_1 := |c-1|_s$
 5: $t := \mathbf{F}_{r,c} + b_{r_1}b_{c_1}(r \neq 1\ ?\ \mathbf{F}_{r_1,c_1} : \mathbf{F}_{c_1,r_1})$
 6: if $m \neq s-1$ then
 7:     $t := t + b_{r_1}(r \neq 1\ ?\ \mathbf{F}_{r_1,c} : \mathbf{F}_{c,s})$
 8: else if $\mathrm{char}(\mathbb{F}) \neq 2$ then
 9:     $t := t + 2b_s\mathbf{F}_{s,s}$
10: end if
11: if $m > 1$ then
12:     $t := t + b_{c_1}\mathbf{F}_{r,c_1}$
13: else if $m = 1 \wedge \mathrm{char}(\mathbb{F}) \neq 2$ then
14:     $t := t + 2b_r\mathbf{F}_{r,r}$
15: end if
16: return $t$
```

The second part of Equation (49) with the condition $r < c$ is

$$\begin{aligned}[\mathbf{B}^\intercal\mathbf{F}\mathbf{B}]_{r,c} + [\mathbf{B}^\intercal\mathbf{F}\mathbf{B}]_{c,r} = \mathbf{F}_{r,c} + b_{|r-1|_s}(\mathbf{F}_{|r-1|_s,c} + \mathbf{F}_{c,|r-1|_s}) +& \\ + b_{|c-1|_s}(\mathbf{F}_{r,|c-1|_s} + \mathbf{F}_{|c-1|_s,r}) + b_{|r-1|_s}b_{|c-1|_s}(\mathbf{F}_{|r-1|_s,|c-1|_s} + \mathbf{F}_{|c-1|_s,|r-1|_s}).&\end{aligned} \tag{52}$$

In all sets of parentheses, there is at least one term equal to zero (note that $\mathbf{F}$ is an upper triangular matrix), except for two cases. The exception is iff the indices are the same. The first case is when $|r-1|_s = c \implies c = r + s - 1$. This condition can occur only when $r = 1 \wedge c = s$:

$$\begin{aligned}[\mathbf{B}^\intercal\mathbf{F}\mathbf{B}]_{1,s} + [\mathbf{B}^\intercal\mathbf{F}\mathbf{B}]_{s,1} &= \mathbf{F}_{1,s} + 2b_s\mathbf{F}_{s,s} + b_{s-1}(\mathbf{F}_{1,s-1} + \mathbf{F}_{s-1,1}) + b_sb_{s-1}(\mathbf{F}_{s,s-1} + \mathbf{F}_{s-1,s}) = \\ &= \mathbf{F}_{1,s} + 2b_s\mathbf{F}_{s,s} + b_{s-1}\mathbf{F}_{1,s-1} + b_sb_{s-1}\mathbf{F}_{s-1,s}.\end{aligned} \tag{53}$$

The second case is when $r = |c-1|_s \implies c = r+1$:

$$[\mathbf{B}^\intercal\mathbf{F}\mathbf{B}]_{r,r+1} + [\mathbf{B}^\intercal\mathbf{F}\mathbf{B}]_{r+1,r} = \mathbf{F}_{r,r+1} + b_{|r-1|_s}(\mathbf{F}_{|r-1|_s,r+1} + \mathbf{F}_{r+1,|r-1|_s}) + 2b_r\mathbf{F}_{r,r} + b_{|r-1|_s}b_r(\mathbf{F}_{|r-1|_s,r} + \mathbf{F}_{r,|r-1|_s}). \tag{54}$$

In the last parenthesis in Equation (52), there is no possibility of the same indices, i.e. $|r-1|_s = |c-1|_s$, because $r < c$. In the case of $\mathrm{char}(\mathbb{F}) = 2$, we were able to skip the computation of $2b_s\mathbf{F}_{s,s}$ and $2b_r\mathbf{F}_{r,r}$.

The result can be summarized as:

$$[\triangledown(\mathbf{B}^{\top}\mathbf{F}\mathbf{B})]_{r,c} = \begin{cases} \textit{Equation } (51) & r = c, \\ \textit{Equation } (54) & r = c - 1, \\ \textit{Equation } (53) & r = 1 \wedge c = s, \\ 0 & r > c, \\ \textit{Equation } (52) & \textit{otherwise}. \end{cases} \tag{55}$$

Equation (55) for the condition $r \leq c$ can be rewritten as Algorithm 1.

5. Performance Evaluation

In this section, we present a side-channel leakage evaluation of the presented side-channel countermeasure and a time and memory performance evaluation. We focus on the efficient equivalent keys scheme proposed in Section 4 only. We compare unprotected and protected implementations regarding side-channel leakage in Section 5.1, regarding time in Section 5.2, and regarding memory in Section 5.3.

5.1. Side-Channel Leakage Evaluation

We describe our key-vs.-key t-test methodology in Section 5.1.1, and then, we present our results in Section 5.1.2. We used an attack-independent test vector leakage assessment methodology [35]. We were not able to detect any statistically significant leakage from the protected implementation.

5.1.1. Methodology

We implemented Rainbow with the proposed equivalent key scheme (using the unprotected reference implementation) on a 32-bit STM32F303 microcontroller based on the ARM Cortex-M4 core. To allow for a thorough and feasible side-channel evaluation of the proposed countermeasure, we chose Rainbow parameters $\mathbb{F}_q = GF(16), v_1 = o_1 = o_2 = 8$ to shorten the signing algorithm runtime compared to the Ia or Vc variants. We used the proposed efficient implementation of the equivalent keys scheme. The microcontroller was mounted in a ChipWhisperer C308 stand-alone evaluation board powered by an external 5 V laboratory power supply and clocked by a 7.37 MHz crystal.

The embedded Rainbow implementation receives a random seed from a controlling PC, which is then expanded using a linear congruential generator. The microcontroller then performs multiple signings without any external communication. Based on the generated pseudorandom numbers, the implementation generates the digests to be signed and also chooses one of four predefined private keys. This way, the microcontroller signs multiple randomly generated digests, each one using one of four randomly chosen private keys. The randomly interleaved private keys are a necessary prerequisite for our leakage evaluation methodology, as described later. When the signings are done, the microcontroller sends a checksum of the signatures back to the controlling PC, and the whole process is repeated as many times as necessary. This approach allows for a significant speedup of the measurement process.

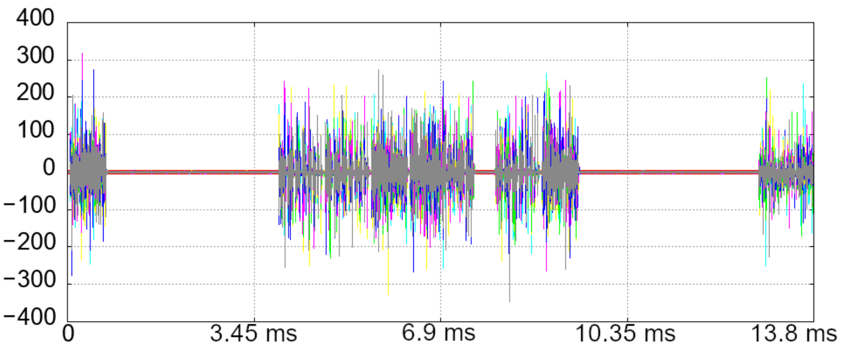
Voltage drops over the core were amplified using the Langer EMV-Technik PA303 preamplifier and sampled using the Picoscope 6404D oscilloscope. A 10Ω shunt resistor was used. The oscilloscope had a 25 MHz bandwidth limiter enabled, and the measurement channel was set in DC 50 Ω mode (since the preamplifier acts like a DC blocker). The sampling rate was 312 MS/s, in our case resulting in over 4 million samples per trace. The measured traces were then decimated 1:10 to 31.2 MS/s to allow for a feasible evaluation while maintaining a good signal-to-noise ratio [36].

The voltage was sampled during each signing, which lasted approximately 13.8 ms. Each decimated power trace consisted of 430,560 samples. The unprotected and protected implementations were evaluated independently. For the unprotected implementation, one of the four original private keys was randomly chosen in every signing. For the protected implementation, an equivalent key of one of the four original private keys was randomly chosen, and the next equivalent key was generated from the previous one.

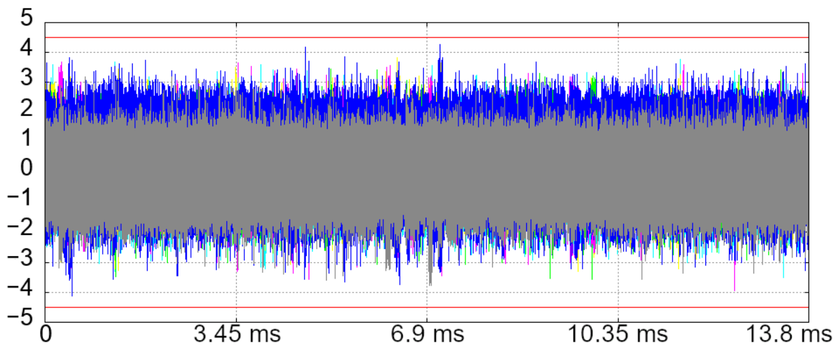
The side-channel leakage was then evaluated using a fixed-fixed t-test methodology [35]. The measured power traces were partitioned into four groups based on the original private key used during the signing. Welch's t-statistic was then computed between every two groups (six evaluations in total), in every sampling point independently. The *null hypothesis* was that the two groups' means are equal, i.e., the original keys are indistinguishable by the mean power consumption. The hypothesis was rejected for high values of the $|t|$-statistic according to the Student's distribution and selected significance level. In side-channel leakage evaluation, the threshold of 4.5 or 5 is typically considered for the $|t|$-statistic, which must be further evaluated carefully with the possibility of both positive and negative false results in mind [35,37].

5.1.2. Results

Figure 1 depicts the results of the leakage evaluations. Figure 1a depicts the results of the evaluation on the unprotected implementation, where every t-test was performed using approximately 40,000 power traces. Figure 1b depicts the results for the protected implementation, where every t-test was performed using approximately one million power traces. Each graph contains six overlaid curves, one for each key-vs.-key t-test. As can be seen in Figure 1a, many peaks reach t-value of 200 and some even exceed 300, in contrast to Figure 1b, where the threshold of 4.5 (marked by red horizontal lines) is not surpassed by the protected implementation. Therefore, we did not detect any statistically significant side-channel leakage from our protected implementation during the signing.



(**a**) Unprotected.



(**b**) Protected.

Figure 1. Results of the t-tests. Each graph contains six overlaid curves. The t-value is depicted on the vertical axis, and time is on the horizontal axis.

5.2. Time Evaluation

We evaluated the time performance on three different platforms:

- The STM32F303 ARM microcontroller (a single-core Cortex-M4);
- A Raspberry Pi 3 B+ single-board computer equipped with the Broadcom BCM2837 ARM microprocessor (a four-core Cortex-A53);
- A desktop computer equipped with an Intel Core i5-2400 processor (four cores).

On the STM32F303 microcontroller, we used the same parameters as for the leakage assessment, i.e., $v_1 = o_1 = o_2 = 8$, where the secret key size is 1776 bytes. On the Raspberry Pi and desktop computer, we used the parameters $v_1 = o_1 = o_2 = 32$, where the secret key size is 95,520 bytes. The Rainbow algorithm runtime was measured including random number generating using a linear congruent generator, excluding hashing. All the implementations were compiled with GCC without any optimizations enabled.

On the STM32F303 microcontroller, the time to sign a document was 13.8 ms, and the average time to generate the equivalent key was 33.28 ms. The whole signing including the equivalent key generation was 3.41-times slower compared to the unsecured signing.

On the Raspberry Pi 3 single-board computer, the average time to sign a document was 7.72 ms, and the average time to generate the equivalent key was 19.87 ms. The whole signing including the equivalent key generation was 3.57-times slower compared to the unsecured signing.

On the desktop computer, the average time to sign a document was 116 µs, and the average time to generate the equivalent key was 420 µs. The whole signing including the equivalent key generation was 4.62-times slower compared to the unsecured signing.

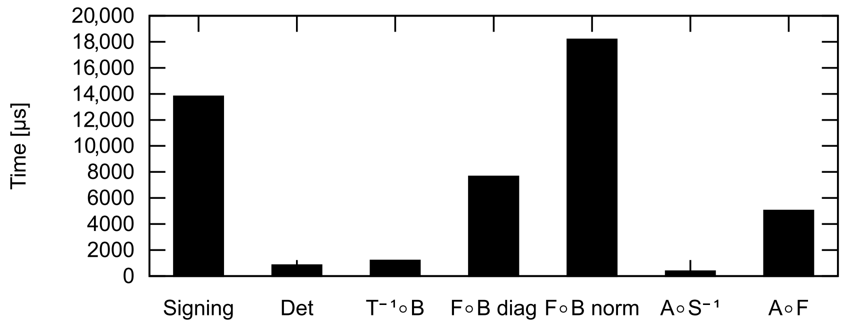
For comparison, the randomization countermeasure of Rainbow proposed in [29] resulted in 3.31-times slower signing. Table 2 summarizes the time overhead comparison. However, our equivalent key can be precomputed any time prior to the signing, and then, the signing itself has no overhead.

Table 2. Time overhead comparison.

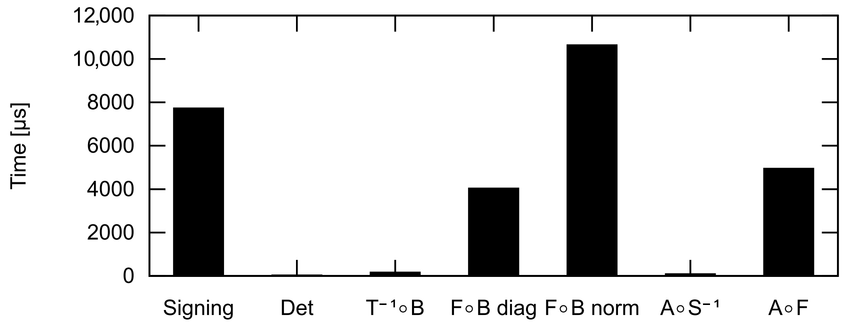
Implementation	Unprotected	Protected	Slowdown
Our countermeasure (STM32F303)	13.8 ms	$13.8 + 33.28 = 47.08$ ms [1]	3.41×
Our countermeasure (RPi)	7.72 ms	$7.72 + 19.87 = 27.59$ ms [1]	3.57×
Our countermeasure (PC)	116 µs	$116 + 420 = 536$ µs [1]	4.62×
Countermeasure from [29]	162,821 cycles	539,338 cycles	3.31×

[1] Our equivalent key can be precomputed any time prior to the signing, and then, the signing itself has no overhead.

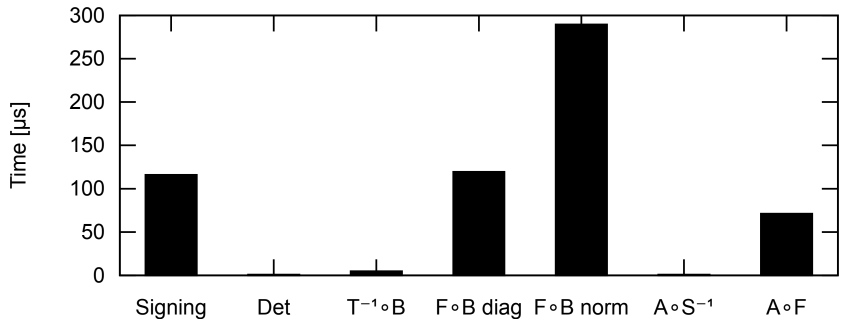
Figure 2 shows the calculation time of the individual generation components. `Det` is the time of the $(\mathbf{A}, \mathbf{B})$ tuple generation including the regularity verification. $\mathtt{T}^{-1} \circ \mathtt{B}$, $\mathtt{A} \circ \mathtt{S}^{-1}$ and $\mathtt{A} \circ \mathtt{F}$ correspond to the calculation of $\mathbf{T}^{-1}\mathbf{B}$, $\mathbf{AS}^{-1}$, and $A \circ F$, respectively. The time of `F∘B diag` refers to Algorithm 1, and `F∘B norm` refers to Equation (48). The time of the signing itself is also included for comparison.



(**a**) STM32F303 ARM microcontroller, $v_1 = o_1 = o_2 = 8$.



(**b**) Raspberry Pi 3 single-board computer, $v_1 = o_1 = o_2 = 32$.



(**c**) Desktop computer, $v_1 = o_1 = o_2 = 32$.

Figure 2. Execution times of the components in equivalent key generation.

5.3. Memory Evaluation

From the memory perspective, the generating of the equivalent key is not an in-place algorithm due to the matrix multiplication in Equation (47). As the key structure is the same as for the unprotected version, we further describe only the extra variables needed for the generation of the equivalent key. The generating of the equivalent key does not need a dynamically allocated memory (as we assumed predefined parameters), and the biggest heap variable is the $(\mathbf{A}, \mathbf{B})$ two-tuple itself, which only takes $(n + m) \cdot \log_2(q)$ bits, where $\log_2(q)$ is the size of an element of $\mathbb{F}_q$ in bits. Other local variables are negligible. Additionally, one can split each part of the mask to be applied separately. This can further reduce extra memory needs to $\max(n, m) \cdot \log_2(q)$ bits.

The implementation submitted to NIST uses a special case of the secret key, where matrices **S** and **T** are specifically selected for memory effectiveness. After the generating of the equivalent key, this special format is no longer possible. As a consequence, we obtained a slightly larger secret key, where the size difference was $(v_1^2 + 2o_1^2 + 2o_2^2) \cdot \log_2(q)$ bits. For the Ia parameters, the difference was 2560 bytes, resulting in a key that was approximately

2.6% larger. Our countermeasure also requires a minor modification of the submitted implementation.

Fresh random bits are needed during the generation and even during the signing process. The amount of randomness is deterministic, except the case where singular matrices are generated and need to be generated again. For the equivalent key generation, the mode (most common value) of the required bits of randomness is $(n+m) \cdot \log_2(q)$, and for the signing process, it is $v_1 \cdot \log_2(q)$ (excluding the hashing salt). The mode is also a minimum number of required bits.

6. Conclusions

In this paper, we proposed a side-channel countermeasure for multivariate quadratic signature schemes. We used the Rainbow signature scheme as our use case. However, the proposed countermeasure is applicable to other schemes such as unbalanced oil and vinegar as well (UOV is equivalent to a single-layer Rainbow). We described the Rainbow algorithm and the notation used.

We proposed an equivalent private key scheme, in which a precomputed randomly generated equivalent key is used for signing instead of the original fixed private key. We examined the scheme in general and described its restrictions and security properties from the theoretical point of view. These include the number of different equivalent keys or reached entropy. We showed that the number of equivalent keys is enormous, and given that, we believe the probability of using the same equivalent key twice is negligible in practice. This efficiently prevents the attacker from mounting attacks such as differential or correlation power analysis.

We further proposed an efficient equivalent key scheme. Our scheme requires significantly less fresh randomness (bounded by the $O(n)$) than the general equivalent key (bounded by the $O(n^2)$) and allows for faster and more efficient computation. We described its properties, similar to the general case. We described an efficient algorithm for the generation of the equivalent key.

We evaluated our proposed countermeasure using attack-independent side-channel leakage assessment with one million power traces, and we were not able to detect any statistically significant information leakage. Lastly, we described the time and memory requirements of the countermeasure. The overhead of our countermeasure is comparable to other relevant countermeasures. Moreover, our equivalent key can be precomputed any time prior to the signing, and then, there is no overhead at the time of signing.

Author Contributions: Conceptualization, D.P. and P.S.; methodology, D.P. and P.S.; software, D.P.; validation, P.S. and D.P.; resources, P.S. and M.N.; data curation, D.P. and P.S.; writing—original draft preparation, D.P. and P.S.; writing—review and editing, P.S. and M.N.; visualization, P.S.; supervision, P.S. and M.N.; project administration, P.S. and M.N.; funding acquisition, M.N. and P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of the Interior of the Czech Republic Grant Number VJ02010010, "Tools for AI-enhanced Security Verification of Cryptographic Devices" in the program Impakt1 (2022–2025).

Data Availability Statement: The data can be provided upon reasonable request to the corresponding author. The side-channel measurements and evaluations were done by the SICAK toolkit, available online: https://github.com/petrsocha/sicak (accessed on 27 September 2022).

Acknowledgments: The authors thank David Kokoška, Jan Onderka, and Vojtěch Miškovský for valuable discussions and proofreading.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Meneghello, F.; Calore, M.; Zucchetto, D.; Polese, M.; Zanella, A. IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices. *IEEE Internet Things J.* **2019**, *6*, 8182–8201. [CrossRef]
2. Daemen, J.; Rijmen, V. *The Design of Rijndael*; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2.

3. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
4. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **1987**, *48*, 203–209. [CrossRef]
5. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **1999**, *41*, 303–332. [CrossRef]
6. Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation. Available online: https://pq-crystals.org/ (accessed on 27 September 2022).
7. Fouque, P.A.; Hoffstein, J.; Kirchner, P.; Lyubashevsky, V.; Pornin, T.; Prest, T.; Ricosset, T.; Seiler, G.; Whyte, W.; Zhang, Z. Falcon: Fast-Fourier Lattice-Based Compact Signatures over NTRU. NIST PQC Project Round 2, Documentation. Available online: https://falcon-sign.info/ (accessed on 27 September 2022).
8. Ding, J.; Schmidt, D. Rainbow, a new multivariable polynomial signature scheme. In Proceedings of the International Conference on Applied Cryptography and Network Security, New York, NY, USA, 7–10 June 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 164–175.
9. Kipnis, A.; Patarin, J.; Goubin, L. Unbalanced oil and vinegar signature schemes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 206–222.
10. Beullens, W.; Preneel, B. Field lifting for smaller UOV public keys. In Proceedings of the International Conference on Cryptology in India, Chennai, India, 10–13 December 2017; Springer: Cham, Switzerland, 2017; pp. 227–246.
11. Kocher, P.; Jaffe, J.; Jun, B. Differential Power Analysis. In *Advances in Cryptology — CRYPTO' 99*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 388–397. [CrossRef]
12. Boer, B.d.; Lemke, K.; Wicke, G. A DPA attack against the modular reduction within a CRT implementation of RSA. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Redwood Shores, CA, USA, 13–15 August 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 228–243.
13. Brier, E.; Clavier, C.; Olivier, F. Correlation Power Analysis with a Leakage Model. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 16–29. [CrossRef]
14. Quisquater, J.J.; Samyde, D. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In Proceedings of the International Conference on Research in Smart Cards, Cannes, France, 19–21 September 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 200–210.
15. Chari, S.; Rao, J.R.; Rohatgi, P. Template attacks. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Redwood Shores, CA, USA, 13–15 August 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 13–28.
16. Rechberger, C.; Oswald, E. Practical template attacks. In Proceedings of the International Workshop on Information Security Applications, Jeju Island, Korea, 23–25 August 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 440–456.
17. Lerman, L.; Poussier, R.; Markowitch, O.; Standaert, F.X. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: Extended version. *J. Cryptogr. Eng.* **2018**, *8*, 301–313. [CrossRef]
18. Hettwer, B.; Gehrer, S.; Güneysu, T. Applications of machine learning techniques in side-channel attacks: A survey. *J. Cryptogr. Eng.* **2020**, *10*, 135–162. [CrossRef]
19. Timon, B. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2019**, *2019*, 107–131. [CrossRef]
20. Park, A.; Shim, K.A.; Koo, N.; Han, D.G. Side-Channel Attacks on Post-Quantum Signature Schemes based on Multivariate Quadratic Equations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 500–523. [CrossRef]
21. Pokorný, D.; Socha, P.; Novotný, M. Side-channel attack on Rainbow post-quantum signature. In Proceedings of the 2021 Design, Automation Test in Europe Conference Exhibition (DATE), Grenoble, France, 1–5 February 2021; pp. 565–568. [CrossRef]
22. Tiri, K.; Verbauwhede, I. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In Proceedings of the Proceedings Design, Automation and Test in Europe Conference and Exhibition, Paris, France, 16–20 February 2004; Volume 1, pp. 246–251.
23. Popp, T.; Mangard, S. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Edinburgh, UK, 29 August–1 September 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 172–186.
24. Lu, Y.; O'Neill, M.; McCanny, J. Evaluation of random delay insertion against DPA on FPGAs. *ACM Trans. Reconfigurable Technol. Syst. (TRETS)* **2010**, *4*, 1–20. [CrossRef]
25. Baddam, K.; Zwolinski, M. Evaluation of dynamic voltage and frequency scaling as a differential power analysis countermeasure. In Proceedings of the 20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07), Bangalore, India, 6–10 January 2007; pp. 854–862.
26. Nikova, S.; Rechberger, C.; Rijmen, V. Threshold implementations against side-channel attacks and glitches. In Proceedings of the International Conference on Information and Communications Security, Raleigh, NC, USA, 4–7 December 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 529–545.
27. Bilgin, B.; Gierlichs, B.; Nikova, S.; Nikov, V.; Rijmen, V. Higher-order threshold implementations. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, 7–11 December 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 326–343.

28. Gross, H.; Mangard, S.; Korak, T. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In Proceedings of the 2016 ACM Workshop on Theory of Implementation Security, Vienna, Austria, 24 October 2016; p. 3.
29. Shim, K.A.; Park, C.M.; Baek, Y.J. Lite-Rainbow: Lightweight Signature Schemes Based on Multivariate Quadratic Equations and Their Secure Implementations. In *Proceedings of the Progress in Cryptology–INDOCRYPT 2015*; Biryukov, A., Goyal, V., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 45–63.
30. Beullens, W. Breaking Rainbow Takes a Weekend on a Laptop. Cryptology ePrint Archive, Report 2022/214, 2022. Available online: https://ia.cr/2022/214 (accessed on 27 September 2022).
31. Wolf, C.; Preneel, B. Equivalent keys in Multivariate Quadratic public key systems. *J. Math. Cryptol.* **2011**, *4*, 375–415. [CrossRef]
32. PQCRainbow.org. Available online: https://www.pqcrainbow.org/ (accessed on 27 September 2022).
33. Andrews, G.E.; Berndt, B. *Ramanujan's Lost Notebook*; Springer: New York, NY, USA, 2005. [CrossRef]
34. Suprunenko, D.; Hirsch, K.; Society, A.M. *Matrix Groups*; Translations of Mathematical Monographs; American Mathematical Society: Providence, RI, USA: 1976.
35. Schneider, T.; Moradi, A. Leakage assessment methodology. *J. Cryptogr. Eng.* **2016**, *6*, 85–99. [CrossRef]
36. O'Flynn, C.; Chen, Z. Synchronous sampling and clock recovery of internal oscillators for side-channel analysis and fault injection. *J. Cryptogr. Eng.* **2015**, *5*, 53–69. [CrossRef]
37. Standaert, F.X. How (not) to use welch's t-test in side-channel security evaluations. In Proceedings of the International Conference on Smart Card Research and Advanced Applications, Montpellier, France, 12–14 November 2018; Springer: Cham, Switzerland, 2018; pp. 65–79.