Analysis of REDOG: the Pad Thai Attack

Alex Pellegrini, Marc Vorstermans

Eindhoven University of Technology, the Netherlands
alex.pellegrini@live.com, m.h.l.vorstermans@tue.nl

Abstract. This paper introduces the Pad Thai message recovery attack on REDOG, a rank-metric code-based encryption scheme selected for the second round of evaluation in the Korean Post-Quantum Cryptography (KPQC) competition. The attack exploits the low rank weight of a portion of the ciphertext to construct multiple systems of linear equations, one of which is noise-free and can be solved to recover the secret message. The Pad Thai attack significantly undermines the security of REDOG, revealing that its provided security is much lower than originally claimed.

1 Introduction

In this paper we introduce the Pad Thai attack, a message recovery attack on REDOG [KHL+23], a code-based public-key encryption system that uses rank-metric codes, specifically Gabidulin codes [Gab85]. REDOG (REinforced modified Dual-Ouroboros based on Gabidulin codes) is built on the hardness of solving the rank decoding problem. It is a candidate in the Korean competition on post-quantum cryptography (KPQC) and has progressed to the second round.

Research into rank-metric codes began with Delsarte in 1978 [Del78], and they were later independently rediscovered by Gabidulin in 1985 [Gab85]. Unlike Delsarte, Gabidulin focused on rank-metric codes that are linear over an extension field. The first cryptographic scheme using rank-metric codes, GPT, was introduced in 1991 by Gabidulin, Paramonov, and Tretjakov [GPT91]. However, the original GPT scheme was broken by Overbeck [Ove05,Ove08], who demonstrated structural attacks that could recover both the secret and public keys.

In 2021, Kim, Kim, Galvez, and Kim [KKGK21] proposed a new rank-metric code-based scheme as a modification of the Dual-Ouroboros public-key encryption scheme [GGH+20]. This scheme employed Gabidulin codes to avoid decryption failures. Lau, Tan, and Prabowo [LTP21] analyzed the scheme and suggested a modification that uses a technique introduced by Loidreau [Loi17] of selecting a certain secret invertible matrix having entries in a space of small dimension. These revisions culminated in the REDOG public-key encryption system [KHL+22]. In 2023, Lange, Pellegrini, and Ravagnani [LPR23] analyzed REDOG, identifying both its incorrectness and vulnerability to algebraic attacks on the rank syndrome decoding problem. They also proposed fixes that improved both correctness and security. With these improvements, an updated version of REDOG [KHL+23] advanced to the second round of the KPQC competition.

1.1 Our Contribution

We present the Pad Thai message recovery attack on REDOG [KHL+23]. In REDOG, part of the public matrix is constructed using a strategy introduced by Loidreau [Loi17]. For REDOG's chosen parameters, this approach imposes a very low upper bound on the rank weight of the noise added to the codeword encoding the message. As a result, many entries of the error vector share the same value. The Pad Thai attack leverages this large number of identical entries in the error vector to create a noise-free system of linear equations, which can be uniquely solved to recover the secret message. We also analyze the complexity and success probability of our attack algorithm, demonstrating that it significantly undermines REDOG's security, which falls short of its claimed robustness.

2 Preliminaries and Background Notions

In this section, we introduce the necessary concepts and notation regarding rank-metric codes and the Pad Thai attack. We use the same notation as in [LPR23].

Let $\mathbb{F}_{2^m}$ denote a finite field with 2^m elements throughout this paper. Let $\{\alpha_1, \ldots, \alpha_m\}$ be a basis of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$. For any $x \in \mathbb{F}_{2^m}$, we can represent x by $(X_1, \ldots, X_m) \in \mathbb{F}_2^m$ as it can be uniquely written as $x = \sum_{i=1}^{m} X_i \alpha_i$, $X_i \in \mathbb{F}_2$ for all $1 \leq i \leq m$. We refer to this as the *vector representation* of x. We may extend this process to $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{F}_{2^m}^n$, defining a map $\mathsf{Mat} : \mathbb{F}_{2^m}^n \to \mathbb{F}_2^{m \times n}$ by:

$$\mathbf{v} \mapsto \begin{bmatrix} V_{11} & V_{21} & \ldots & V_{n1} \\ V_{12} & V_{22} & \ldots & V_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ V_{1m} & V_{2m} & \ldots & V_{nm} \end{bmatrix}.$$

Definition 2.1. *The rank weight of a vector* $\mathbf{v} \in \mathbb{F}_{2^m}^n$ *is defined as* $\mathsf{wt}_R(\mathbf{v}) := \mathsf{rk}_q(\mathsf{Mat}(\mathbf{v}))$ *and the rank distance between* $\mathbf{v}, \mathbf{w} \in \mathbb{F}_{2^m}^n$ *is defined as* $d_R(\mathbf{v}, \mathbf{w}) := \mathsf{wt}_R(\mathbf{v} - \mathbf{w})$.

We remark that the rank distance of two vectors is independent from the choice of the basis of $\mathbb{F}_{2^m}$. For $\mathbf{v} \in \mathbb{F}_{2^m}^n$, we denote the space spanned by $\mathbf{v}$ as $\langle \mathbf{v} \rangle$. More concretely, if we refer to this space, we mean the $\mathbb{F}_2$-subspace of $\mathbb{F}_2^m$ spanned by the columns of $\mathsf{Mat}(\mathbf{v})$.

Rank-metric codes are an alternative to the traditional hamming-metric codes. Therefore, we briefly introduce the Hamming weight and the Hamming distance.

Definition 2.2. *The Hamming weight of a vector* $\mathbf{v} \in \mathbb{F}_{2^m}^n$ *is defined as the number of non-zero entries of* $\mathbf{v}$*, i.e.* $\mathsf{wt}_H(\mathbf{v}) := \#\{i \in \{1, \ldots, n\} \mid v_i \neq 0\}$. *The Hamming distance between vectors* $\mathbf{v}, \mathbf{w} \in \mathbb{F}_{2^m}^n$ *is defined as* $d_H(\mathbf{v}, \mathbf{w}) := \mathsf{wt}_H(\mathbf{v} - \mathbf{w})$.

A code, together with its minimal distance and correction capability are defined as follows.

Definition 2.3. *Let $n, k, d \in \mathbb{N}$ and let D be a distance metric. Then, an $[n, k, d]$-code C with respect to D over $\mathbb{F}_{2^m}$ is a k-dimensional $\mathbb{F}_{2^m}$-linear subspace of $\mathbb{F}_{2^m}^n$ with minimum distance*

$$d := \min_{\mathbf{x},\mathbf{y} \in C,\, \mathbf{x} \neq \mathbf{y}} D(\mathbf{x}, \mathbf{y})$$

and correction capability $\lfloor (d-1)/2 \rfloor$.

For our result, we only consider $D = d_R$ or $D = d_H$, i.e. the rank distance or hamming distance, respectively. If $D = d_R$ (resp. $D = d_H$) then the code C is also called a *rank-metric* (resp. *Hamming-metric*) code. In this paper we consider all codes to be linear over the field extension $\mathbb{F}_{2^m}$.

The final notions that we require concerning codes are those of the generator matrix and parity check matrix of a code. A matrix $\mathbf{G}$ is called a *generator matrix* of a code C if the rows of $\mathbf{G}$ span C. A matrix $\mathbf{H}$ is called a *parity check matrix* of a code C if C is the right-kernel of $\mathbf{H}$.

REDOG utilizes *Gabidulin codes* [Gab85] in their system specification, which is a well-known family of rank metric codes. In this paper, we mainly use Gabidulin codes as a black box. For more details on Gabidulin codes, we refer to the original paper.

Remark 2.4. We have omitted definitions regarding the system specification of REDOG as these will not be used for the Pad Thai attack. This included definitions such as *Moore matrices*, *circulant matrices*, and *isometries*, for which we refer to the original REDOG paper [KHL+23].

To analyze the performance of the Pad Thai attack, we estimate the number of *basic operations* that the attack needs to carry out. As we work over a finite field, we regard any *field operation* as one basic operation. Note that REDOG is defined over an extension field, for which we count the number of operations in the base field.

3 System Specification

This section introduces the specification of REDOG. We use the same notation as in [LPR23].

The system parameters are positive integers $(n, k, \ell, q, m, r, \lambda, t_1, t_2)$, with $\ell < n$ and $t_1 + \lambda t_2 \leq r \leq \lfloor (n-k)/2 \rfloor$. Furthermore, a hash function $\mathsf{hash} : \{0,1\}^* \to \{0,1\}^{\ell_{\mathsf{hash}}}$ for some positive integer ℓ_{hash} is chosen. Since the input of hash will be elements of $\mathbb{F}_{2^m}^{2n-k}$, we assume that such input is internally transformed into a string of symbols in $\{0,1\}$.

The REDOG PKE consists of three algorithms, $\mathsf{Keygen}, \mathsf{Encrypt}, \mathsf{Decrypt}$. The Keygen algorithm, see Algorithm 3.1, outputs a public key $\mathsf{pk} = (\mathbf{M}, \mathbf{F})$ and a secret key $\mathsf{sk} = (\mathbf{P}, \mathbf{H}, \mathbf{S}, \Phi)$.

Algorithm 3.1 REDOG-2-Keygen

Input : $\emptyset$.
Output : A public key $\mathsf{pk} \in \mathbb{F}_{2^m}^{\ell \times 2n-k}$ and a secret key $\mathsf{sk} = (\mathbf{P}, \mathbf{H}, \mathbf{S}, \Phi)$ with $\mathbf{P} \in \mathbb{F}_{2^m}^{n \times n}$, $\mathbf{H} \in \mathbb{F}_{2^m}^{(n-k)\times(2n-k)}$, $\mathbf{S} \in \mathbb{F}_{2^m}^{(n-k)\times(n-k)}$ and Φ a Gabidulin decoder for the code with parity check matrix $\mathbf{H}$.

1. Select a parity check matrix $\mathbf{H} = (\mathbf{H}_1 \mid \mathbf{H}_2)$ of a $[2n-k, n]$ Gabidulin code, so that $\mathbf{H}_2 \in \mathsf{GL}_{n-k}(\mathbb{F}_{2^m})$. Let Φ be the syndrome decoder that corrects r errors;
2. Select a full rank matrix $\mathbf{M} \in \mathbb{F}_{2^m}^{\ell \times n}$ and an isometry $\mathbf{P} \in \mathbb{F}_{2^m}^{n \times n}$ (with respect to the rank metric);
3. Select a λ-dimensional $\mathbb{F}_2$-subspace $\Lambda \subset \mathbb{F}_{2^m}$ containing 1 and select a random circulant matrix $\mathbf{S}^{-1} \in \mathsf{GL}_{n-k}(\mathbb{F}_{2^m})$ having entries only in Λ;
4. Compute $\mathbf{F} = \mathbf{M}\mathbf{P}^{-1}\mathbf{H}_1^T \left(\mathbf{H}_2^T\right)^{-1} \mathbf{S}$ and publish the public key $\mathsf{pk} = (\mathbf{M}, \mathbf{F})$. Store the secret key $\mathsf{sk} = (\mathbf{P}, \mathbf{H}, \mathbf{S}, \Phi)$.

The encryption algorithm, see Algorithm 3.2, takes as input a message $\mathsf{msg} \in \mathbb{F}_{2^m}^{\ell}$ and a public key $\mathsf{pk} = (\mathbf{M}, \mathbf{F})$. It outputs a ciphertext $\mathbf{c}$ of the form $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$.

Algorithm 3.2 REDOG-2-Encrypt

Input : A message $\mathsf{msg} \in \mathbb{F}_{2^m}^{\ell}$ and a public key $\mathsf{pk} = (\mathbf{M}, \mathbf{F})$.
Output : A ciphertext $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in \mathbb{F}_{2^m}^{n} \times \mathbb{F}_{2^m}^{n-k}$.

1. Generate uniformly random $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2) \in \mathbb{F}_{2^m}^{2n-k}$ with $\mathbf{e}_1 \in \mathbb{F}_{2^m}^{n}$ having $\mathsf{wt}_R(\mathbf{e}_1) = t_1$ and $\mathbf{e}_2 \in \mathbb{F}_{2^m}^{n-k}$ having $\mathsf{wt}_R(\mathbf{e}_2) = t_2$;
2. Compute $\mathbf{m} = \mathsf{msg} + \mathsf{hash}(\mathbf{e})$;
3. Compute $\mathbf{c}_1 = \mathbf{m}\mathbf{M} + \mathbf{e}_1$ and $\mathbf{c}_2 = \mathbf{m}\mathbf{F} + \mathbf{e}_2$ and output $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$.

The decryption algorithm, see Algorithm 3.3, takes as input a ciphertext of the form $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$ and a secret key $\mathsf{sk} = (\mathbf{P}, \mathbf{H}, \mathbf{S}, \Phi)$. It outputs a message $\mathsf{msg} \in \mathbb{F}_{2^m}^{\ell}$.

Algorithm 3.3 REDOG-2-Decrypt

Input : A ciphertext $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in \mathbb{F}_{2^m}^{n} \times \mathbb{F}_{2^m}^{n-k}$ and a secret key $\mathsf{sk} = (\mathbf{P}, \mathbf{H}, \mathbf{S}, \Phi)$.
Output: The message $\mathsf{msg} \in \mathbb{F}_{2^m}^{\ell}$ corresponding to $\mathbf{c}$.

1. Compute $\mathbf{c}' = \mathbf{c}_1\mathbf{P}^{-1}\mathbf{H}_1^T - \mathbf{c}_2\mathbf{S}^{-1}\mathbf{H}_2^T = \mathbf{e}'\mathbf{H}^T$ where the vector $\mathbf{e}' := (\mathbf{e}_1\mathbf{P}^{-1}, -\mathbf{e}_2\mathbf{S}^{-1})$;
2. Decode $\mathbf{c}'$ using Φ to obtain $\mathbf{e}'$. Recover $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$ using $\mathbf{P}$ and $\mathbf{S}$;
3. Solve $\mathbf{m}\mathbf{M} = \mathbf{c}_1 - \mathbf{e}_1$;
4. Output $\mathsf{msg} = \mathbf{m} - \mathsf{hash}(\mathbf{e})$.

Note that $\mathsf{wt}_R(\mathbf{e}') = t_1 + \lambda t_2 \leq r$, so that indeed Φ can be applied to $\mathbf{c}'$ (Step 2 of Algorithm 3.3) to obtain error vector $\mathbf{e}'$.

3.1 Suggested Parameters

We list the suggested parameters of REDOG for 128, 192 and 256 bits of security submitted to round 2 of the KpqC competition.

Table 1. Suggested parameters by [KHL+23]. The security parameter is given in the number of bits.

Security parameter	$(n, k, \ell, q, m, r, \lambda, t_1, t_2)$
128	$(30, 6, 25, 2, 59, 12, 3, 6, 2)$
192	$(44, 8, 37, 2, 83, 18, 3, 12, 2)$
256	$(58, 10, 49, 2, 109, 24, 3, 15, 3)$

4 The Pad Thai Attack

In this section, we describe our attack on REDOG which succeeds in recovering the messages corresponding to REDOG's ciphertexts. We first give an overview of the attack.

4.1 Overview

We break down the description of the Pad Thai attack into two steps. We aim to construct a system of linear equations which can be solved uniquely for $\mathbf{m}$, and subsequently for $\mathbf{e}_1$ and $\mathbf{e}_2$. Finally, we recover the message by computing $\mathsf{msg} = \mathbf{m} - \mathsf{hash}(\mathbf{e}_1 \mid \mathbf{e}_2)$.

First Step. The goal of the first step is to construct a system of linear equations starting from the relation $\mathbf{c}_2 = \mathbf{mF} + \mathbf{e}_2$. To construct this system, we combine columns of $\mathbf{F}$ and the corresponding entries of $\mathbf{c}$ in order to obtain a system of equations $\mathbf{c}_2' = \mathbf{mF}' + \mathbf{e}_2'$ where $\mathbf{e}_2'$ has only t_2 nonzero entries whose positions are known.

Assume that we know the mentioned system. Observe that, for all security levels of REDOG, we have $t_2 = 2$ or $t_2 = 3$, which means that $n - k - t_2$ entries in $\mathbf{c}_2'$ are error-free. Let $i_1, \ldots, i_{t_2} \in \{1, \ldots, n-k\}$ be such that $e_{2,i_j} \neq 0$ for $j \in \{1, \ldots, t_2\}$. Take $\mathbf{F}'' \in \mathbb{F}_{2^m}^{\ell \times (n-k-t_2)}$ as the submatrix of $\mathbf{F}'$ consisting of the columns F_i' for $i \neq i_1, \ldots, i_{t_2}$. Similarly, compute $\mathbf{c}_2'' \in \mathbb{F}_{2^m}^{n-k-t_2}$ by taking the entries c_i' where $i \neq i_1, \ldots, i_{t_2}$. Then, the message $\mathbf{m}$ satisfies

$$\mathbf{c}_2'' = \mathbf{mF}'', \tag{1}$$

which is an underdetermined system as we have ℓ unknowns of $\mathbf{m}$ and $n-k-t_2$ equations, where $n-k-t_2 < \ell$ for every security level. In reality, $\ell - n + k = 1$ for every security level, which means that we are $t_2 + 1$ equations short.

Second Step. In order to uniquely compute $\mathbf{m}$ we need to pad the system in (1) with t_2+1 extra error-free equations by combining some of the equations from $\mathbf{c}_1 = \mathbf{m}\mathbf{M} + \mathbf{e}_1$. Let $c'_{1,i} = \mathbf{m}M'_i$ for $i = 1, \ldots, t_2 + 1$ be such error-free equations. We can add these equations to (1) and obtain a new system

$$(\mathbf{c}''_2 \mid c'_{1,1} \mid \cdots \mid c'_{1,t_2+1}) = \mathbf{m}\left(\mathbf{F}'' \mid M'_1 \mid \cdots \mid M'_{t_2+1}\right). \tag{2}$$

In REDOG's specification, $\mathbf{M}$ is chosen uniformly at random among the full-rank matrices in $\mathbb{F}_{2^m}^{\ell \times n}$. Moreover, $\mathbf{F}$ is assumed to be another random matrix by [KHL+23, Problem 2] so we can safely assume that $\left(\mathbf{F}'' \mid M'_1 \mid \cdots \mid M'_{t_2+1}\right) \in \mathbb{F}_{2^m}^{\ell \times \ell}$ is a random matrix, thus having full rank with high probability. We can now compute $\mathbf{m}$ by inverting the system (2) and recover msg.

4.2 First Step

We describe a method to produce a system of equations $\mathbf{c}'_2 = \mathbf{m}\mathbf{F}' + \mathbf{e}'_2$ where the Hamming weight $\mathsf{wt}_H(\mathbf{e}'_2) = t_2$ and the positions of non-zero entries of $\mathbf{e}'_2$ are known. This can be done because of the following observation.

Remark 4.1. Let F_i denote the i-th column of $\mathbf{F}$, then $c_{2,i} = \mathbf{m}F_i + e_{2,i}$. Assume that $e_{2,i} = e_{2,j}$ for some i, j. Then

$$c_{2,i} + c_{2,j} = \mathbf{m}(F_i + F_j) + e_{2,i} + e_{2,j} = \mathbf{m}(F_i + F_j).$$

Let $\alpha_1, \ldots, \alpha_{t_2} \in \mathbb{F}_{2^m}^*$ be such that $\langle \mathbf{e}_2 \rangle_{\mathbb{F}_2} = \langle \alpha_1, \ldots, \alpha_{t_2} \rangle_{\mathbb{F}_2}$. So each entry $e_{2,i}$ of $\mathbf{e}_2$ can assume a value in an $\mathbb{F}_2$-vector subspace of $\mathbb{F}_{2^m}$ containing 2^{t_2} elements. This suggests that REDOG's encryption algorithm chooses $\mathbf{e}_2$ among $2^{(n-k)t_2}$ possibilities (actually, less than $2^{(n-k)t_2}$ as the rank weight constraint $\mathsf{wt}_R(\mathbf{e}_2) = t_2$ must also hold). Label the unknown values of $\langle \mathbf{e}_2 \rangle_{\mathbb{F}_2}$ as $\{0, \alpha_1, \alpha_2, \ldots, \alpha_{2^{t_2}-1}\}$, where

$$\alpha_j = \sum_{h=1}^{t_2} z_{j,h}\alpha_h \tag{3}$$

for some $z_{j,h} \in \mathbb{F}_2$ for every $j = t_2 + 1, \ldots, 2^{t_2} - 1$. Most of the time we will handle 0 separately, but for convenience, we define $\alpha_0 = 0$.

In the following, let $2^{\{1,\ldots,n\}}$ denote the set of subsets of $\{1, \ldots, n\}$, and thus $\left(2^{\{1,\ldots,n\}}\right)^t$ a vector of t such subsets.

Definition 4.2 (Set of arrangements of subsets). *Let $t \in \mathbb{N}$ be a positive integer. We define the set of arrangements of t disjoint subsets over n elements as the set of ordered tuples in $\left(2^{\{1,\ldots,n\}}\right)^t$ defined as*

$$A_{t,n} := \left\{ \mathbf{a} \in \left(2^{\{1,\ldots,n\}}\right)^t \mid \bigcup_{i=1}^{t} a_i = \{1, \ldots, n\}, a_i \cap a_j = \emptyset \ \forall i \neq j \right\}.$$

Proposition 4.3. *Let $\alpha = \{\alpha_1, \ldots, \alpha_{t_2}\} \subset \mathbb{F}_{2^m}^*$ be a set of $\mathbb{F}_2$-linearly independent elements. There exists a one-to-one correspondence between the set $E_{\alpha,n-k} := \{\mathbf{e} \in \mathbb{F}_{2^m}^{n-k} \mid \langle \mathbf{e} \rangle_{\mathbb{F}_2} \subseteq \langle \alpha_1, \ldots, \alpha_{t_2} \rangle_{\mathbb{F}_2}\}$ and $A_{2^{|\alpha|},n-k}$.*

Proof. For $\mathbf{e} \in E_{\alpha,n-k}$, denote by

$$\mathbf{e}^0 := \{i \in \{1, \ldots, n-k\} \mid e_i = 0\}$$

and by

$$\mathbf{e}^{\alpha_j} := \{i \in \{1, \ldots, n-k\} \mid e_i = \alpha_j\}$$

the positions where $\mathbf{e}$ is 0 and α_j for all $j = 1, \ldots, 2^{t_2} - 1$, respectively. We prove that the map

$$\begin{aligned} \varphi_{\alpha,n-k} : E_{\alpha,n-k} &\to A_{2^{|\alpha|},n-k} \\ \mathbf{e} &\mapsto (\mathbf{e}^0, \mathbf{e}^{\alpha_1}, \ldots, \mathbf{e}^{\alpha_{2^{t_2}-1}}) \end{aligned}$$

is a bijection by showing that it is both injective and surjective. Let $\mathbf{e}, \mathbf{f} \in E_{\alpha,n-k}$ be such that $\mathbf{e} \neq \mathbf{f}$. Then there exists $i \in \{1, \ldots, n-k\}$ such that $e_i \neq f_i$. Write $e_i = \alpha_{j_1}$ and $f_i = \alpha_{j_2}$ for some $j_1, j_2 \in \{0, \ldots, 2^{t_2} - 1\}$ with $j_1 \neq j_2$, then $\mathbf{e}^{\alpha_{j_1}} \neq \mathbf{f}^{\alpha_{j_1}}$. It follows that $\varphi_{\alpha,n-k}(\mathbf{e}) \neq \varphi_{\alpha,n-k}(\mathbf{f})$.

On the other hand, let $\mathbf{a} \in A_{2^{|\alpha|},n-k}$ and let $\mathbf{e} \in \mathbb{F}_{2^m}^{n-k}$ be such that $e_j = 0$ for every $j \in a_1$ and $e_j = \alpha_{i-1}$ for every $j \in a_{i-1}$ and every $i = 2, \ldots, 2^{t_2}$. Clearly, $\langle \mathbf{e} \rangle_{\mathbb{F}_2} \subseteq \langle \alpha_1, \ldots, \alpha_{t_2} \rangle_{\mathbb{F}_2}$ and $\varphi_{\alpha,n-k}(\mathbf{e}) = \mathbf{a}$.

□

Definition 4.4 (Good basis of a vector). *Let $\mathbf{v} \in \mathbb{F}_{2^m}^n$ be such that $\mathsf{wt}_R(\mathbf{v}) = t$. A set of elements $\{\alpha_1, \ldots, \alpha_t\} \subset \mathbb{F}_{2^m}^t$ such that $\langle \mathbf{v} \rangle_{\mathbb{F}_2} = \langle \alpha_1, \ldots, \alpha_t \rangle_{\mathbb{F}_2}$, is called a good basis for $\mathbf{v}$ if for every $j \in \{1, \ldots, t\}$ there exists an $i \in \{1, \ldots, n\}$ such that $v_i = \alpha_j$.*

Remark 4.5. It is clear that a good basis exists for every $\mathbf{e} \in \mathbb{F}_{2^m}$. To see that, let $t = \mathsf{wt}_R(\mathbf{e})$ and define the basis $\{\alpha_1, \ldots, \alpha_t\}$ of $\langle \mathbf{e} \rangle_{\mathbb{F}_2}$ by taking the leftmost t entries in $\mathbf{e}$ which are linearly independent over $\mathbb{F}_2$. Then $\{\alpha_1, \ldots, \alpha_t\}$ is a good basis for $\mathbf{e}$.

Denote by $A'_{2^{t_2},n-k}$ the subset of $A_{2^{t_2},n-k}$ so that $a_i \neq \emptyset$ for $i = 2, \ldots, t_2 + 1$.

Definition 4.6 (Arrangement of a vector). *Let $\mathbf{e} \in \mathbb{F}_{2^m}^{n-k}$ with $\mathsf{wt}_R(\mathbf{e}) = t_2$ and $\alpha = \{\alpha_1, \ldots, \alpha_{t_2}\} \subset \mathbb{F}_{2^m}^{t_2}$ be a good basis for $\mathbf{e}$. Then we call $\varphi_{\alpha,n-k}(\mathbf{e})$ the arrangement of $\mathbf{e}$ with respect to α.*

Observe that, given a good basis α for $\mathbf{e}$, the arrangement of $\mathbf{e}$ w.r.t. α is in $A'_{2^{t_2},n-k}$.

Algorithm 4.7 RearrangeSystem

Input: An arrangement $\mathbf{a} \in A'_{2^{t_2},n-k}$, a REDOG's partial ciphertext $\mathbf{c}_2 \in \mathbb{F}_{2^m}^{n-k}$ corresponding to a message $\mathbf{m} \in \mathbb{F}_{2^m}^{\ell}$ under the partial public key $\mathbf{F} \in \mathbb{F}_{2^m}^{\ell\times(n-k)}$.
Output: A vector $\mathbf{c}''_2 \in \mathbb{F}_{2^m}^{n-k-t_2}$ and a matrix $\mathbf{F}'' \in \mathbb{F}_{2^m}^{\ell\times(n-k-t_2)}$.

1. Fix elements $x_i \in a_i$ for every $i = 2, \ldots, t_2+1$;
2. Construct $\mathbf{F}'' \in \mathbb{F}_{2^m}^{\ell\times(n-k-t_2)}$ and $\mathbf{c}''_2$ by computing the following columns and values:
   (a) $F''_j = F_j$ and $c''_{2,j} = c_{2,j}$ for every $j \in a_1$;
   (b) $F''_j = F_j + F_{x_i}$ and $c''_{2,j} = c_{2,j} + c_{2,x_i}$ for all $j \in a_i \setminus \{x_i\}$ and $i = 2, \ldots, t_2+1$;
   (c) $F''_j = F_j + \sum_{h=1}^{t_2} z_{i-1,h} F_{x_h+1}$ and $c''_{2,j} = c_{2,j} + \sum_{h=1}^{t_2} z_{i-1,h} c_{2,x_h+1}$ for all $j \in a_i$ and $i = t_2+2, \ldots, 2^{t_2}$.
3. Return $\mathbf{F}''$ and $\mathbf{c}''_2$, the matrix $\mathbf{F}''$ and vector $\mathbf{c}''$ punctured at $x_i, i = 2, \ldots, t_2+1$.

Proposition 4.8. *Let $\mathbf{c}_2 = \mathbf{mF} + \mathbf{e}_2$ be a REDOG's partial ciphertext and $\varphi_{\alpha,n-k}(\mathbf{e}_2)$ be the arrangement of $\mathbf{e}_2$ w.r.t a good basis α. Then Algorithm 4.7 returns $\mathbf{c}''_2 \in \mathbb{F}_2^{n-k-t_2}$ and $\mathbf{F}'' \in \mathbb{F}_{2^m}^{\ell\times(n-k-t_2)}$ such that*

$$\mathbf{c}''_2 = \mathbf{mF}''.$$

Proof. The algorithm repeatedly applies the observation in Remark 4.1. Each $j \in a_1$ has that $c_{2,j}$ is error free. Each $j \in a_i, i = 2, \ldots, t_2+1$ has $c_{2,j} = \mathbf{m}F_j + \alpha_{i-1}$. The algorithm selects one such index as x_i and then applies Remark 4.1 to cancel the α_{i-1} in all other $c_{2,j}$ for $j \in a_i \setminus \{x_i\}$. Eventually, $\mathbf{c}''_2$ is punctured at x_i so that only those entries without error remain.

Similarly, all $c_{2,j}$ with $j \in a_i$ for $i = t_2+2, \ldots, 2^{t_2}$ have error α_{i-1} added and (3) states the coefficients $z_{i-1,h}$ representing α_{i-1} in the basis. Again using that c_{x_h} contributes α_{h-1} shows that the third case produces an error-free $c''_{2,j}$ for $j \in a_i$.

In total, the matrix and vector are punctured at the t_2 positions of the x_i, thus producing $n-k-t_2$ error free equations $c''_{2,j} = \mathbf{m}F''_j$. □

Proposition 4.8 together with Algorithm 4.7 provides a method that transforms the system of equations $\mathbf{c}_2 = \mathbf{mF} + \mathbf{e}_2$ into a smaller system $\mathbf{c}''_2 = \mathbf{mF}''$ that does not involve any noise. As the system is underdetermined, we present an algorithm in the next section that exploits REDOG's partial ciphertext $\mathbf{c}_1$ to obtain the necessary remaining equations for the system.

Remark 4.9. Note that Proposition 4.8 assumes knowledge of the arrangement of $\mathbf{e}_2$. We want to stress that, since the basis α is unknown, knowing the arrangement of $\mathbf{e}_2$ w.r.t. α does not necessarily mean knowing $\mathbf{e}_2$. This assumption will be satisfied as we iterate over all possible arrangements of $\mathbf{e}_2$.

4.3 Second Step

In this subsection, we investigate how to pad the system equations $\mathbf{c}''_2 = \mathbf{mF}''$ with t_2+1 additional equations to uniquely determine $\mathbf{m}$. The idea is to construct

these extra equations, combining equations from $\mathbf{c}_1 = \mathbf{mM} + \mathbf{e}_1$. Observe that since $\mathsf{wt}_R(\mathbf{e}_1) = t_1$, then any set $\{e_{1,i_1}, \ldots, e_{1,i_{t_1+1}}\}$ of $t_1 + 1$ entries of $\mathbf{e}_1$ is linearly dependent, i.e. there exist $z_1, \ldots, z_{t_1+1} \in \mathbb{F}_2$ not all zero such that

$$\sum_{j=1}^{t_1+1} z_j e_{1,i_j} = 0.$$

This suggests that given a set of $t_1 + 1$ equations of $\mathbf{c}_1 = \mathbf{mM} + \mathbf{e}_1$ one can search the space of $\mathbb{F}_2$-linear combinations for $t_1 + 1$ non-zero combinations of the equations, which cancels the error factor.

Remark 4.10. Observe that we need to make sure that the columns M_{i_j} for $j = 1, \ldots, t_1 + 1$ are linearly independent, as otherwise we might run into

$$\sum_{j=1}^{t_1+1} z_j M_{i_j} = 0$$

which is a useless equation. However, the probability for this to happen is negligible for each parameter set.

Since we need $t_2 + 1$ extra equations to pad the system, we need to find $t_2 + 1$ equations simultaneously with this method. In total, we obtain a linear system of $\ell = n - k + 1$ equations that can be solved to recover the message.

4.4 The Full Attack

For each system rearrangement that we perform in the first step, we need to test all paddings in the second step. Testing the solution of each system we construct implies computing a candidate message $\mathbf{m}' \in \mathbb{F}_{2^m}^{\ell}$ and candidate errors $\mathbf{e}'_1 \in \mathbb{F}_{2^m}^{n}$ and $\mathbf{e}'_2 \in \mathbb{F}_{2^m}^{n-k}$ and checking whether the rank weights of $\mathbf{e}'_1$ and $\mathbf{e}'_2$ match t_1 and t_2, respectively. Therefore, combining the two steps described in this section, we obtain Algorithm 4.11. We present an argument for the correctness of the Pad Thai attack.

Proposition 4.12. *Algorithm 4.11, under the assumption that matrix* $\mathbf{G}$ *in Step 5.(d) is invertible, recovers a valid message* msg *corresponding to a REDOG's ciphertext* $\mathbf{c}$ *under public key* $\mathsf{pk} = (\mathbf{M}, \mathbf{F})$.

Proof. Let $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in \mathbb{F}_{2^m}^{2n-k}$ be a REDOG's ciphertext corresponding to a message $\mathsf{msg} \in \mathbb{F}_{2^m}^{\ell}$ under the public key $\mathsf{pk} = (\mathbf{M}, \mathbf{F}) \in \mathbb{F}_{2^m}^{\ell \times (2n-k)}$.

As the algorithm executes Step 1 for all arrangements $\mathbf{a} \in A'_{2^{t_2}, n-k}$, by Proposition 4.8, there exists an arrangement of $\mathbf{e}_2$ (w.r.t. a good basis α) that outputs $\mathbf{F}'', \mathbf{c}''$ such that

$$\mathbf{c}'' = \mathbf{m}'\mathbf{F}''$$

for $\mathbf{c}_2 = \mathbf{m}'\mathbf{F} + \mathbf{e}_2$. Matrix $\mathbf{F}''$ has size $\ell \times (n - k - t_2)$.

Algorithm 4.11 PadThaiAttack

Input: A REDOG's ciphertext $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in \mathbb{F}_{2^m}^{2n-k}$ corresponding to a message $\mathsf{msg} \in \mathbb{F}_{2^m}^{\ell}$ under the public key $\mathsf{pk} = (\mathbf{M}, \mathbf{F}) \in \mathbb{F}_{2^m}^{\ell \times (2n-k)}$.
Output: The message msg.

For each arrangement $\mathbf{a} \in A'_{2^{t_2}, n-k}$ do:

1. Let $\mathbf{F}'', \mathbf{c}'' = \mathsf{RearrangeSystem}(\mathbf{a}, \mathbf{c}_2, \mathbf{F})$;
2. Pick random sets $J_1, \ldots, J_{t_2+1} \subset \{1, \ldots, n\}$ with $|J_i| = t_1 + 1$;
3. Let $\mathbf{M}_{J_i}$ be the matrix consisting of columns of $\mathbf{M}$ indexed by J_i;
4. If $\mathsf{rk}(\mathbf{M}_{J_i}) < t_1 + 1$ for some $i \in \{1, \ldots, t_2 + 1\}$ then go to Step 2.
5. For every $(\mathbf{v}_1, \ldots, \mathbf{v}_{t_2+1}) \in \left(\mathbb{F}_2^{t_1+1}\right)^{t_2+1}$ do:
   (a) Compute $M'_i = \mathbf{M}_{J_i} \mathbf{v}_i^\top$ for each $i = 1, \ldots, t_2 + 1$;
   (b) Let $\mathbf{c}_{1,J_i}$ be the vector consisting of the entries of $\mathbf{c}_1$ indexed by J_i;
   (c) Compute $c'_{1,i} = \mathbf{c}_{1,J_i} \mathbf{v}_i^\top$ for each $i = 1, \ldots, t_2 + 1$;
   (d) Let $\mathbf{G} := (\mathbf{F}'' \mid M'_1, \ldots, M'_{t_2+1})$ and $\mathbf{y} := (\mathbf{c}'' \mid c'_{1,2}, \ldots, c'_{1,t_2+1})$. If $\mathbf{G}$ is not invertible, continue at the next iteration of Step 5;
   (e) Compute $\mathbf{m}' = \mathbf{y}\mathbf{G}^{-1}$;
   (f) Compute $\mathbf{e}'_1 = \mathbf{c}_1 - \mathbf{m}'\mathbf{M}$ and $\mathbf{e}'_2 = \mathbf{c}_2 - \mathbf{m}'\mathbf{F}$;
   (g) If $\mathsf{wt}_R(\mathbf{e}'_1) = t_1$ and $\mathsf{wt}_R(\mathbf{e}'_2) = t_2$ then return $\mathsf{msg} = \mathbf{m}' - \mathsf{hash}(\mathbf{e}'_1 \mid \mathbf{e}'_2)$.

In Steps 2-4, the algorithm constructs $t_2 + 1$ matrices $\mathbf{M}_{J_1}, ..., \mathbf{M}_{J_{t_2+1}}$, each of rank exactly $t_1 + 1$. The algorithm simply repeats this process until $t_2 + 1$ matrices of desired rank are constructed.

In Steps 5.(a)-(c), the columns $M'_i = \mathbf{M}_{J_i} \mathbf{v}_i^T$ and vectors $c'_{1,i} = \mathbf{c}_{1,J_i} \mathbf{v}_i^T$ are constructed, where $\mathbf{v}_i \in \mathbb{F}_2^{t_1+1}$ for each $i = 1, \ldots, t_2 + 1$. Since $\mathsf{wt}_R(\mathbf{e}_1) = t_1$, any set $\{e_{1,i_1}, \ldots, e_{1,i_{t_1+1}}\}$ of $t_1 + 1$ entries of $\mathbf{e}_1$ is linearly dependent, i.e. for there exists some $z_1, \ldots, z_{t_1+1} \in \mathbb{F}_2$ not all zero such that

$$\sum_{j=1}^{t_1+1} z_j e_{1,i_j} = 0.$$

Thus, at some iteration of Steps 5.(a)-5.(c), some columns $M'_1, \ldots, M'_{t_2+1}$ and vectors $c'_{1,1}, \ldots, c'_{1,t_2+1}$ are found that satisfy

$$(c'_{1,1} \mid \cdots \mid c'_{1,t_2+1}) = \mathbf{m}' \left(M'_1 \mid \cdots \mid M'_{t_2+1}\right).$$

In total, for some arrangement $\mathbf{a}' \in A'_{2^{t_2}, n-k}$ and some vectors $\mathbf{v}'_1, ..., \mathbf{v}'_{t_2+1} \in \left(\mathbb{F}_2^{t_1+1}\right)^{t_2+1}$ there exists a matrix $\mathbf{G} := (\mathbf{F}'' \mid M'_1, \ldots, M'_{t_2+1})$ and vector $\mathbf{y} := (\mathbf{c}'' \mid c'_{1,2}, \ldots, c'_{1,t_2+1})$ so that $\mathbf{y} = \mathbf{m}'\mathbf{G}$, where $\mathbf{G}$ has size $\ell \times \ell$. Under the assumption that $\mathbf{G}$ is invertible, the algorithm determines $\mathbf{m}' = \mathbf{y}\mathbf{G}^{-1}$, which allows the algorithm to compute $\mathbf{e}'_1$ and $\mathbf{e}'_2$ in Step 5.(f). Then in Step 5.(g), if $\mathsf{wt}(\mathbf{e}'_1) = t_1$ and $\mathsf{wt}(\mathbf{e}'_2) = t_2$, the algorithm recovers the correct message corresponding to the REDOG's ciphertext $\mathbf{c}$. □

In the next section, we give the complexity analysis of our attack and point out some areas of improvement.

Remark 4.13. The matrix $\mathbf{G}$ in Step 5.(d) of Algorithm 4.11 needs to be invertible for the attack to be successful. This is a direct consequence of Lemma 5.3, which can be applied to show that $\mathbf{G}$ is invertible with probability ~ 1. A more elaborate argument is presented in Section 5.1.

5 Analysis of the Pad Thai Attack

In this section, we describe the complexity of our attack on REDOG described in Algorithm 4.11. Let us start with the following easy lemma.

Lemma 5.1. *The cardinality of* $A_{2^{t_2},n-k}$ *is* $2^{t_2(n-k)}$.

Proof. By Proposition 4.3 there is a bijection between $A_{2^{t_2},n-k}$ and $E_{\alpha,n-k}$ for a fixed set $\alpha = \{\alpha_1, \ldots, \alpha_{t_2}\} \subset \mathbb{F}_{2^m}^{t_2}$. The number of elements in $E_{\alpha,n-k}$ is clearly $2^{t_2(n-k)}$.

□

A first assessment of the complexity of the Pad Thai attack is given in the following proposition.

Proposition 5.2. *The Algorithm 4.11 recovers the message* msg *corresponding to a REDOG ciphertext* $\mathbf{c}$ *under public key* $\mathsf{pk} = (\mathbf{M}, \mathbf{F})$ *in*

$$\mathcal{O}(2^{(t_1+1)(t_2+1)+t_2(n-k)}\ell^\omega m^2) \tag{4}$$

field operations, where $2 \leq \omega \leq 3$ *is the matrix multiplication exponent.*

Proof. The algorithm consists of two nested cycles. The first cycle iterates over all arrangements $A'_{2^{t_2},n-k}$, which is a subset of $A_{2^{t_2},n-k}$ whose cardinality is reported in Lemma 5.1.

The most expensive steps of each cycle of Algorithm 4.11 are Steps 2.(c) in Algorithm 4.7 and 5.(e). The former computes $2^{t_2}-t_2-1$ sums $F_j+\sum_{h=1}^{t_2} z_{i-1,h}F_{x_{h+1}}$, i.e. the sum of t_2 elements of $\mathbb{F}_{2^m}^{\ell}$ for a total number of operations in $\mathbb{F}_2$ in $\mathcal{O}(2^{t_2}mt_2\ell)$. The latter happens in the nested cycle and inverts a matrix $\mathbf{G} \in \mathbb{F}_{2^m}^{\ell\times\ell}$. Its cost, using schoolbook multiplications in finite fields, is in $\mathcal{O}(\ell^\omega m^2)$ and is performed $2^{(t_1+1)(t_2+1)}$ times for each outer cycle. For each parameter set in Table 1 we have that

$$2^{t_2}mt_2\ell < 2^{(t_1+1)(t_2+1)}\ell^\omega m^2.$$

Combining with the number of outer cycles we obtain the claimed complexity.

□

The following table reports the updated security provided by REDOG based on our attack.

Table 2. $\log_2$ of the complexity of the Pad Thai attack for each security level of REDOG according to equation (4) with $\omega = 2.807$.

Security parameter	$(n, k, \ell, q, m, r, \lambda, t_1, t_2)$	Pad Thai attack
128	$(30, 6, 25, 2, 59, 12, 3, 6, 2)$	**93.8**
192	$(44, 8, 37, 2, 83, 18, 3, 12, 2)$	**138.37**
256	$(58, 10, 49, 2, 109, 24, 3, 15, 3)$	**237.3**

Table 2 suggests that the combination of parameters of REDOG security level 256 has a smaller loss of security under the Pad Thai attack compared to security levels 128 and 192. This can be explained by the choices for parameter t_2.

5.1 Success Probability

The matrix $\mathbf{F} \in \mathbb{F}_{2^m}^{\ell \times n-k}$ is assumed to be indistinguishable from random as per [KHL+23, Problem 2], hence we can consider every output matrix $\mathbf{F}'' \in \mathbb{F}_{2^m}^{\ell \times n-k-t_2}$ of Algorithm 4.7 as random too. Furthermore, given that each pad consists of a combination of random columns of an actual random matrix $\mathbf{M} \in \mathbb{F}_{2^m}^{\ell \times n}$, we conclude that the entire matrix $\mathbf{G} \in \mathbb{F}_{2^m}^{\ell \times \ell}$ can be considered as a matrix chosen uniformly at random. We only need to estimate the probability that $\mathbf{G}$ is invertible in order for the attack to succeed when the right system has been set up. To this end, we can use the following result of [LPR23].

Lemma 5.3 ([LPR23, Lemma 4.2]). *Let V be a t-dimensional subspace $V \subseteq \mathbb{F}_q^m$ and let $S \in V^s$ be a uniformly random s-tuple of elements of V. The probability $p(q, s, t)$ that $\langle S_i \mid i \in \{1, \ldots, s\} \rangle = V$ is*

$$p(q,s,t) = \begin{cases} 0 & \text{if} \quad 0 \leq s < t; \\ \sum_{i=0}^{t} \begin{bmatrix} t \\ i \end{bmatrix}_q (-1)^{t-i} q^{s(i-t)+\binom{t-i}{2}} & \text{otherwise}, \end{cases} \tag{5}$$

where $\begin{bmatrix} t \\ i \end{bmatrix}_q$ is the q-binomial coefficient, counting the number of subspaces of dimension i of $\mathbb{F}_q^t$, and $\binom{a}{b} = 0$ for $a < b$. In particular, this probability does not depend on m or on the choice of V, but only on its dimension.

By setting $V = \mathbb{F}_{2^m}^{\ell}$ and $s = t = \ell$ in the above lemma, we obtain that the set of columns of $\mathbf{G}$ spans the entire space V. In other words, $\mathbf{G}$ is invertible with probability ~ 1 for every cycle and every security level. As a result, we are assured that the attack succeeds at recovering the secret msg with probability ~ 1.

We want to point out a scenario where $\left(M'_1 \mid \cdots \mid M'_{t_2+1}\right)$, generated in Step 5.(a) of Algorithm 4.11, does not have full rank. This may occur when for some

distinct $i, j \in \{1, \ldots, t_2+1\}$, we have that $J_i \cap J_j \neq \emptyset$. We denote this intersection by S. Let $\mathbf{M}_{J_i}, \mathbf{M}_{J_j}$ be the matrices consisting of columns of $\mathbf{M}$ indexed by J_i, J_j respectively. Let $\{e_{1,k}\}_{k\in J_i}, \{e_{1,k}\}_{k\in J_j}$ be the set of error terms corresponding to $\mathbf{M}_{J_i}, \mathbf{M}_{J_j}$ respectively. In case the sets $\{e_{1,k}\}_{k\in J_i\setminus S}, \{e_{1,k}\}_{k\in J_j\setminus S}$ are linearly independent and $\{e_{1,k}\}_{k\in S}$ is not, then there exists a unique linear combination so that

$$\sum_{k\in J_i} z_k e_{1,k} = 0 \text{ and } \sum_{k\in J_j} z_k e_{1,k} = 0$$

for $z_k = 1$ for $k \in S$ and $z_k = 0$ otherwise. Consequently, the vectors $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{F}_2^{t_1+1}$ that create a error-free equation will lead to outputting $M_i' = M_j'$ in Step 5.(a). This case will be rejected in Step 5.(d) as $\mathbf{G}$ is not invertible. If this is the only linear combination that makes the error vanish, the algorithm will not recover the correct message. We have estimated the probability of this event to happen, having performed 5000 runs per parameter set. The results are reported in Table 3. The Sage code used for the experiments can be found in Appendix A.

Table 3. Estimated probability of the event $M_i' = M_j'$ to happen in step 5.(d) of Algorithm 4.11 per security level. The total number of runs is 5000.

Security parameter	Estimated probability $M_i' = M_j'$
128	0.0124
192	0.001
256	0.0004

5.2 Attack Improvements

In this subsection, we point out an interesting behavior of Algorithm 4.7 of the first step of our attack. We observe that Algorithm 4.7 rearranges the system $\mathbf{c}_2 = \mathbf{mF} + \mathbf{e}_2$ depending only on the arrangement of $\mathbf{e}_2$. Here is an example for $t_2 = 2$.

Example 5.4. Let $\mathbf{e}, \mathbf{f} \in \mathbb{F}_{2^m}^{n-k}$ be the vectors

$$\mathbf{e} = (0, 0, \alpha, \alpha, \beta, \alpha+\beta)$$

and

$$\mathbf{f} = (0, 0, \beta, \beta, \alpha+\beta, \alpha).$$

These two vectors have the same arrangement w.r.t. the bases $\{\alpha, \beta\}$ and $\{\beta, \alpha+\beta\}$, respectively. Now, let $\mathbf{c_e} = \mathbf{mF} + \mathbf{e}$ and $\mathbf{c_f} = \mathbf{mF} + \mathbf{f}$ and let $x_1 = 3$ and $x_2 = 5$. Let also $\mathbf{a}$ be the arrangement of $\mathbf{e}$ (equivalently, of $\mathbf{f}$). Then, on inputs $(\mathbf{a}, \mathbf{c_e})$ and $(\mathbf{a}, \mathbf{c_f})$, Algorithm 4.7 produces the same output.

This means that we can run the outer cycle on a subset of the arrangements $A'_{2^{t_2},n-k}$ as Algorithm 4.7 does not distinguish between errors having the same

arrangement. Indeed, for each arrangement there are $y = \prod_{i=0}^{t_2-1}(2^{t_2} - 2^i)$ other arrangements for which Algorithm 4.7 produces the same output. The updated complexity becomes therefore

$$\mathcal{O}(2^{(t_1+1)(t_2+1)+t_2(n-k)}\ell^{\omega}m^2y^{-1}). \tag{6}$$

The updated values are as follows.

Table 4. $\log_2$ of the complexity of the Pad Thai attack for each security level of REDOG according to equation (6) with $\omega = 2.807$.

Security parameter	$(n, k, \ell, q, m, r, \lambda, t_1, t_2)$	Pad Thai attack
128	$(30, 6, 25, 2, 59, 12, 3, 6, 2)$	**91.22**
192	$(44, 8, 37, 2, 83, 18, 3, 12, 2)$	**135.78**
256	$(58, 10, 49, 2, 109, 24, 3, 15, 3)$	**229.9**

Another improvement comes from noting that the complexity estimates consider all arrangements in $A_{2^{t_2},n-k}$ including those that correspond to error vectors $\mathbf{e}_2$ that have rank weight $\mathsf{wt}_R(\mathbf{e}_2) \leq t_2$. Considering only arrangements for error vectors with rank weight exactly t_2 slightly reduces the number of arrangements that we need to iterate over in the first step.

Remark 5.5. We want to stress that the values in Table 2 and Table 4 are overestimates due to the fact that Algorithm 4.11 iterates over $A'_{2^{t_2},n-k}$, which is a subset of $A_{2^{t_2},n-k}$ and that we assume schoolbook arithmetic in $\mathbb{F}_{2^m}$.

6 Impact of Pad Thai attack on key sizes

During the production of the final version of the paper, the REDOG authors have suggested a new set of parameters for REDOG, presented in Table 5. We report these parameters to indicate their influence on the key and ciphertext sizes. These are presented in Table 6. Where the key sizes grow by a factor of approximately 1.2-1.6 for security levels 128 and 192, we surprisingly notice a decrease in the key size for security level 256. This is explained by the REDOG team by stating that the parameter set for security level 256 was conservative, and thus increasing t_2 seems to be sufficient to reach a desired level of security.

Remark 6.1. We do not endorse using the parameters in Table 5 before they have been further analyzed.

Table 5. Newly suggested parameters by the REDOG authors, see https://groups.google.com/g/kpqc-bulletin/c/t-q5PfMtbYU. The security parameter is given in the number of bits.

Security parameter	$(n, k, \ell, q, m, r, \lambda, t_1, t_2)$
128	$(36, 6, 31, 2, 67, 15, 3, 6, 3)$
192	$(47, 6, 43, 2, 89, 20, 3, 11, 3)$
256	$(56, 10, 49, 2, 109, 24, 3, 11, 4)$

Table 6. Old and new REDOG key and ciphertext sizes in KB.

Security parameter	$\text{size}_{\mathsf{pk}}$	$\text{size}_{\mathsf{sk}}$	$\text{size}_{\mathsf{ct}}$
128_{old}	9.72	0.68	0.39
128_{new}	16.73	0.95	0.54
192_{old}	29.99	1.42	0.81
192_{new}	41.11	1.68	0.96
256_{old}	69.11	2.47	1.41
256_{new}	66.50	2.37	1.36

7 Conclusion

This paper presents the Pad Thai message recovery attack on REDOG, a rank metric code-based encryption scheme that is candidate to the second round of evaluations of KPQC. The Pad Thai attack exploits the low-rank weight of the error vectors and recovers the message by constructing a noise-free system of linear equations. This significantly reduces the security of the Pad Thai attack by approximately 37, 56, and 27 bits for security levels 128, 192, and 256 respectively.

Acknowledgments. The authors would like to express their gratitude to Tanja Lange for the valuable suggestions and comments regarding the development of the Pad Thai attack.

Appendix

A Sage Code for Experiments

```
#n,l,m,q,t1 = 30,25,59,2,6
#n,l,m,q,t1 = 44,37,83,2,12
n,l,m,q,t1 = 58,49,109,2,15
```

```
Fqm = GF(q**m)
Fq = GF(q)

C = codes.LinearRankMetricCode(random_matrix(Fqm,l,n), Fq)
def get_random_error(n,t):
    while True:
        e = vector([Fqm.random_element() for i in range(t)])
        rk = C.rank_weight_of_vector(e)
        if rk == t:
            A = random_matrix(Fq, t,n)
            return e*A

count = 0
for i in range(5000):
    e = get_random_error(n,t1)

    #for testing security levels 128 and 192,
    only J1,J2,J3 are needed

    J1 = list(sample([0..n-1],t1+1))
    J2 = list(sample([0..n-1],t1+1))
    J3 = list(sample([0..n-1],t1+1))
    J4 = list(sample([0..n-1],t1+1))

    Jint12 = [i for i in J1 if i in J2]
    Jint13 = [i for i in J1 if i in J3]
    Jint14 = [i for i in J1 if i in J4]
    Jint23 = [i for i in J2 if i in J3]
    Jint24 = [i for i in J2 if i in J4]
    Jint34 = [i for i in J3 if i in J4]

    e1 = vector([e[i] for i in J1])
    e2 = vector([e[i] for i in J2])
    e3 = vector([e[i] for i in J3])
    e4 = vector([e[i] for i in J4])

    rk1 = C.rank_weight_of_vector(e1)
    rk2 = C.rank_weight_of_vector(e2)
    rk3 = C.rank_weight_of_vector(e3)
    rk4 = C.rank_weight_of_vector(e4)

    if Jint12 != [] and rk1==t1 and rk2==t1:
        eint12 = vector([e[i] for i in Jint12])
        if C.rank_weight_of_vector(eint12) != len(Jint12):
            count +=1
```

```
    if Jint13 != [] and rk1==t1 and rk3==t1:
        eint13 = vector([e[i] for i in Jint13])
        if C.rank_weight_of_vector(eint13) != len(Jint13):
            count +=1

    if Jint14 != [] and rk1==t1 and rk4==t1:
        eint14 = vector([e[i] for i in Jint14])
        if C.rank_weight_of_vector(eint14) != len(Jint14):
            count +=1

    if Jint23 != [] and rk2==t1 and rk3==t1:
        eint23 = vector([e[i] for i in Jint23])
        if C.rank_weight_of_vector(eint23) != len(Jint23):
            count +=1

    if Jint24 != [] and rk2==t1 and rk4==t1:
        eint24 = vector([e[i] for i in Jint24])
        if C.rank_weight_of_vector(eint24) != len(Jint24):
            count +=1

    if Jint34 != [] and rk3==t1 and rk4==t1:
        eint34 = vector([e[i] for i in Jint34])
        if C.rank_weight_of_vector(eint34) != len(Jint34):
            count +=1

print("count:", count)
```

References

Del78. Philippe Delsarte. Bilinear forms over a finite field, with applications to coding theory. *Journal of Combinatorial Theory, Series A*, 25(3):226–241, 1978.

Gab85. Ernst M. Gabidulin. Theory of codes with maximum rank distance. *Problems of Information Transmission*, 21(1):1–12, 1985.

GGH+20. Philippe Gaborit, Lucky Galvez, Adrien Hauteville, Jon-Lark Kim, Myeong Jae Kim, and Young-Sik Kim. Dual-ouroboros: An improvement of the mcnie scheme. *Advances in Mathematics of Communications*, 14(2):301–306, 2020.

GPT91. Ernst M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a non-commutative ring and thier applications in cryptology. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 482–489, Brighton, UK, April 8–11, 1991. Springer, Berlin, Heidelberg, Germany.

KHL+22. Jon-Lark Kim, Jihoon Hong, Terry Shue Chien Lau, YounJae Lim, Chik How Tan, Theo Fanuela Prabowo, and Byung-Sun Won. REDOG. Submission to KpqC Round 1, 2022.

KHL+23. Jon-Lark Kim, Jihoon Hong, Terry Shue Chien Lau, YounJae Lim, and Byung-Sun Won Bo seung Yang. REDOG. Submission to KpqC Round 2, 2023.

KKGK21. Jon-Lark Kim, Young-Sik Kim, Lucky Erap Galvez, and Myeong Jae Kim. A modified Dual-Ouroboros public-key encryption using Gabidulin codes. *Appl. Algebra Eng. Commun. Comput.*, 32(2):147–156, 2021.

Loi17. Pierre Loidreau. A new rank metric codes based encryption scheme. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 3–17, Utrecht, The Netherlands, June 26–28, 2017. Springer, Cham, Switzerland.

LPR23. Tanja Lange, Alex Pellegrini, and Alberto Ravagnani. On the security of REDOG. In *ICISC (2)*, volume 14562 of *Lecture Notes in Computer Science*, pages 282–305. Springer, 2023.

LTP21. Terry Shue Chien Lau, Chik How Tan, and Theo Fanuela Prabowo. On the security of the modified Dual-Ouroboros PKE using Gabidulin codes. *Appl. Algebra Eng. Commun. Comput.*, 32(6):681–699, 2021.

Ove05. Raphael Overbeck. A new structural attack for GPT and variants. In *Mycrypt*, volume 3715 of *Lecture Notes in Computer Science*, pages 50–63. Springer, 2005.

Ove08. R. Overbeck. Structural attacks for public key cryptosystems based on Gabidulin codes. *Journal of Cryptology*, 21(2):280–301, April 2008.