Threshold linear solving in small fields and application to UOV

Paco Azevedo-Oliveira[1,2], Jordan Beraud[2], and Pierre Varjabedian[1,2]

[1] Thales, France
paco.azevedo-oliveira@thalesgroup.com
pierre.varjabedian@thalesgroup.com
[2] Laboratoire de Mathématiques de Versailles, UVSQ, CNRS, Université Paris-Saclay, 78035 Versailles, France
jordan.beraud@uvsq.fr

Abstract. Threshold signatures allow multiple parties to sign a common message by collaborating. More specifically, in a $(t, n)-$threshold signature scheme, at least t out of n parties must collaborate to sign a message. In particular, solving linear systems shared among some parties is a problem that naturally arises in threshold cryptography, and this paper proposes three algorithms for a set of parties to solve a shared linear system $A\mathbf{x} = \mathbf{b}$ in finite fields of low characteristic.

The first two algorithms securely compute the determinant of a shared matrix using recent theoretical results on Newton's polynomials and by adapting an algorithm by Samuelson and Berkowitz. From these results, two algorithms can be deduced to solve the corresponding linear system. On the other hand, the third is a modification of an existing state-of-the-art algorithm.

Although pre-quantum threshold signature algorithms have been extensively studied, the state of the art in the creation of post-quantum threshold algorithms remains sparse. In particular, few papers have studied the creation of a threshold algorithm based on UOV, despite the simplicity of the scheme. The new algorithms presented in this paper enable other threshold instantiations of UOV and UOV-based schemes.

1 Introduction

Since the post-quantum standardization process that began in 2016, signature algorithms based on solving multivariate equations have received a lot of attention from the cryptographic community. Among the schemes selected for Round 2 of the second call for selection by the National Institute of Standards and Technology (NIST), four of them are based on the resolution of multivariate equations, including the most conservative UOV [3] and the most efficient MAYO [2].

Alongside this standardization process, the NIST is expected to issue a call for multi-party threshold signature schemes. In threshold signature algorithms, a set of n parties perform distributed calculations in order to compute a signature

for a message. These calculations are performed interactively, without revealing any secret data, and guarantee security as long as the number of corrupt participants does not exceed a certain threshold t. The NIST call will be divided into two submission categories. The first category concerns the creation of threshold signature schemes based on existing NIST standards, while the second concerns primitives not specified by the NIST. This call from the NIST has encouraged the development of numerous post-quantum threshold schemes, most of which are based on the difficulty of some problems on structured lattices [12,11,4,10]. Given that UOV and its variants are serious candidates in the standardization process and with the announcement of the future NIST call, designing a threshold scheme based on UOV or MAYO is therefore an important research question.

In the state of the art, there are two positive results that enable a threshold scheme based on UOV or MAYO. In [7], the authors examine all the signature submissions to NIST Round 2 and conclude that UOV-based schemes are the simplest to threshold, since all that is needed is to create a protocol that allows parties to solve a linear system in a shared way. The authors conclude that of the nine schemes submitted, the two most suitable for threshold use are Rainbow and LUOV, two UOV-based schemes[3]. In addition, they propose a "tailor-made" way to solve a shared linear system. In [5], the authors explore the thresholdization of UOV-based algorithms, focusing on UOV and MAYO. They show that the main algorithm proposed in [7] gives a lot of information about UOV's secret key and describe the main ideas of an attack against this protocol. They propose an algorithm that promises "minimum leakage": In the case where the system $A\mathbf{x} = \mathbf{b}$ has no solution, the algorithm reveals the rank of the matrix A. They provide explanations as to why this reduced leakage appears secure for their specific threshold setting. However, it is still an open problem to design an algorithm that reveals nothing, quoting [7] p.5: "we leave as an open problem the design of a protocol for linear system solving that, in cases of rank deficiency, reveals only that the matrix is not full rank".

At least one paper, [5], proposes a complete version of UOV and MAYO. However, the protocol for solving a shared linear system provides information about the rank of the matrix in cases where the system has no solution. Since this matrix is directly linked to the secret key, this rank provides non-trivial information about the secret key. To remain conservative, one might wish to eliminate this leakage. For now, there is no known algorithm for solving a linear system of the form $A\mathbf{x} = \mathbf{y}$ which, in the absence of a solution, reveals only the determinant of the matrix A.

As UOV and most of its variants are defined in extensions of $\mathbb{F}_2$, eliminating this leakage amounts precisely to the creation of a protocol for solving linear systems in fields of small characteristic, *i.e.* fields for which the size of the matrix A is strictly greater than the characteristic.

[3] It is important to mention that LUOV and Rainbow were broken during the standardization process in [8] and [1] respectively.

Related work. By carefully studying the state of the art, we may note that [9] proposes to compute the characteristic polynomial in a shared manner. This protocol also enables the calculation of the determinant (which is its smallest degree coefficient). Unfortunately, this algorithm requires that for a matrix $A \in M_n(\mathbb{F})$ there must be $\text{char}(\mathbb{F}) > n$, this is never the case for the matrices considered in UOV or MAYO, which use fields of characteristic 2. The authors briefly explain in the appendix how to generalise this algorithm. They use a transcendental extension $\mathbb{F}[y]/(y^{n+1})$, but as no complete description is given, it is difficult to assess the applicability of this modified algorithm in our specific case.

Our Contributions. We have three main contributions:

- First we propose a complete generalization of the algorithm described in [9] to the case of low characteristic, it allows to securely compute the determinant of a shared matrix A, and thus to create a protocol for solving a linear system that reveals nothing other than the determinant. This leads to a complexity of $O(n^{3.5})$ communications and $O(1)$ rounds of communications.
- Then we revisit Samuelson and Berkowitz's algorithm in a shared manner, which also allow to compute the determinant. It gives a trade-off between the number of communications and the number of communication rounds. The resulting complexity is in $O(n^3)$ communications and $O(n)$ rounds of communications.
- Finally, for contexts where the two algorithms above are too costly to implement, we propose a generalization of the algorithm proposed in [5], which reveals either the rank of matrix A or the rank of a random matrix that is non invertible.

2 Preliminaries

This section recalls the definitions and results already known, which will be useful throughout the rest of the paper.

2.1 Proportion of non-singular matrices

To estimate the proportion of non-singular matrices within a finite field of size $q > 2$ we will use a classical result from [15]:

$$\alpha = \frac{|GL_n(\mathbb{F})|}{|M_n(\mathbb{F})|} = \frac{(q^n-1)(q^n-q)\dots(q^n-q^{n-1})}{q^{n^2}} \geq \left(\frac{q-2}{q-1}\right).$$

2.2 Notation

We will use the Arithmetic Black Box (ABB) to model the secret sharing and arithmetic operations. In the rest of the paper, $[\![x]\!]$ will denote a secret value x shared among the parties. Operations which can be performed locally by the

parties are the addition and multiplication by a constant respectively denoted as $[\![x+y]\!] \leftarrow [\![x]\!]+[\![y]\!]$, and $[\![\lambda x]\!] \leftarrow \lambda[\![x]\!]$. The multiplication of two secrets are done by the parties with an interactive protocol, and is naturally noted $[\![xy]\!] \leftarrow [\![x]\!]\cdot[\![y]\!]$. The notations and operations are generalisable to matrix sharing by viewing matrices coordinate wise. The only exception is that the product of two secret shared matrices can be computed more efficiently with a special protocol than with the product of matrix coordinates. No details are given about the nature of $[\![x]\!]$, since different concrete instantiations of the ABB model depend on the adversarial model. To evaluate the complexity of an algorithm that involves shared matrix products, we will use the following estimates:

Proposition 1 ([9]) *Let two shared matrices A and B where $A \in M_{m,n}(\mathbb{F}_q), B \in M_{n,l}(\mathbb{F}_q)$. Then there exists a secure MPC protocol for computing a shared representation of the product AB with a constant number of rounds and $O(lm)$ communications.*

Proposition 2 ([9]) *Let two shared matrices A and B where $A \in M_{m,n}(\mathbb{F}_q(\zeta)), B \in M_{n,l}(\mathbb{F}_q(\zeta))$ with ζ of degree k. Then there exists a secure MPC protocol for computing a shared representation of the product AB with a constant number of rounds and $O(klm)$ communications.*

Proposition 3 ([9]) *Let $A \in M_n(\mathbb{F}_q)$ be a shared matrix. Then there exist a secure protocol computing the first m power of A with expected $O(mn^2)$ communication and $O(1)$ round.*

2.3 A problem to solve

One define Problem 1, which involves solving a shared linear system in a finite field. Furthermore, we require that, in the absence of a solution, the players learn as little information as possible.

Problem 1 *Given $([\![A]\!], [\![\mathbf{b}]\!])$ a shared representation of $A \in \mathcal{M}_{s,t}(\mathbb{F})$ and $\mathbf{b} \in \mathbb{F}_q^s$, create an algorithm that computes, if it exists, $[\![\mathbf{x}]\!]$ a shared representation of $\mathbf{x}$ such that:*

$$A\mathbf{x} = \mathbf{b},$$

without revealing any information on A, $\mathbf{b}$ and $\mathbf{x}$.

Remark 1. Note that it is not possible to construct an algorithm that solves Problem 1 without providing any information about A. Indeed, if $\mathbf{b}$ is not in the image of A, there is no solution. The least amount of information that can be revealed is therefore the determinant of A, which will in any case be zero if $\mathbf{b}$ is not in the image of A.

Section 3 proposes new algorithms for solving Problem 1. Algorithms that partially solve Problem 1 already exists and will be compared with those new solutions in Section 4.

3 Shared linear resolution

When the characteristic of the field is greater than the size of the matrix, [13] used the Faddeev-Leverrier algorithm, which uses the relation between Newton's polynomials to compute securely the determinant with a complexity $O(n^{2.5})$. Unfortunately, this result does not apply to matrices defined over $\mathbb{F}_2$ and its extensions, which are all fields of characteristic 2. For a general field of low characteristic, this result was generalized in [16,9], by working in an extension of $\mathbb{F}_q$. In the annexe of [9], it is claimed that the resulting algorithm has a complexity of $O(n^{3.5})$. Unfortunately, the algorithm is not clearly defined. This makes it difficult to understand and even to implement. Recently a theoretical paper [18], proposed a generalization of the relations between Newton's polynomials. Using these new theoretical results yields a new algorithm to securely compute the determinant.

3.1 Existing works for large characteristics

First, we review the ideas for calculating the characteristic polynomial of a shared matrix if the characteristic of the field is large enough, everything that follows is derived from and justified in [9]. Let $A \in M_n(\mathbb{F}_q)$ and $\chi_A(X) = X^n + \sum_{i=1}^{n} X^{n-i} d_i$ its characteristic polynomial. Let $(A_i)_{0 \leq i \leq n-1}$ be the following sequence of matrices:

$$\begin{cases} A_0 = A \\ A_i = A\left(A_{i-1} - \frac{1}{i}\mathrm{tr}(A_{i-1})I_n\right) \text{ with } 1 \leq i \leq n-1. \end{cases} \tag{1}$$

Then, d_i can be computed with the formula $d_i = -\frac{1}{i}\mathrm{tr}(A_i)$. This relation can be rewritten in matrix form according to Proposition 4 due to Leverrier.

Proposition 4 (Leverrier) *The coefficients $d_1, \cdots, d_n$ of the characteristic polynomial of a matrix $A \in M_n(\mathbb{F}_q)$ satisfies:*

$$\begin{pmatrix} 1 & & & & \\ t_1 & 2 & & & \\ t_2 & t_1 & 3 & & \\ \vdots & \vdots & \vdots & \ddots & \\ t_{n-1} & t_{n-2} & \cdots & t_1 & n \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{pmatrix} = - \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_n \end{pmatrix}$$

where $t_j := \mathrm{tr}(A^j)$.

When $\mathrm{char}(\mathbb{F}_q) > n$, then the matrix in Proposition 4 is non singular since its determinant is $n!$. Thus, the complexity of calculating the d_i for a shared matrix can be summarized as calculating the t_i on the one hand and solving the linear system securely on the other. The computation of the t_i's will rely on the relation

$$\mathrm{tr}(AB) = \sum_{1 \leq k \leq n} \langle a_k, b^k \rangle$$

where a_k is the k-th row of A and b^k the $k-$th column of B. This can be done with one invocation of a secure protocol to compute an inner product. This idea will be combined with a baby-step giant-step approach: for $m = \lfloor \sqrt{n} \rfloor$, precompute $A^2, A^3, \ldots, A^m$ and $A^{2m}, A^{3m}, \ldots, A^{m^2}$ with Proposition 3 then compute $t_{i+jm} = \mathrm{tr}(A^i A^{jm})$. This is done with a complexity of $O(mn^2)$ communications. Since the linear system is always invertible, after calculating the t_i, a secure matrix inversion protocol can be used to solve the system in $O(n^2)$ communications. The final complexity will be in $O(n^{2.5})$ communications.

Remark 2. The previous protocol is that it is simple to implement and to understand. The objective of the next section is to build a simple protocol for calculating the determinant in small characteristics, in the same way of Leverrier's result.

3.2 A generalization in low characteristic

In this subsection $\mathbb{F}_q$ will be a field of characteristic r_0 and we consider matrices $A \in M_n(\mathbb{F}_q)$ with $n > r_0$. Let $e_k(x_1, \ldots, x_n)$ the k-th symmetric polynomial and $p_k(x_1, \ldots, x_n)$ the k-th Newton's polynomial defined by:

$$\begin{cases} e_k(x_1, \ldots, x_n) = \displaystyle\sum_{1 \leq i_1 < \ldots i_k < n} x_{i_1} x_{i_2} \cdots x_{i_k} \\ \\ p_k(x_1, \cdots, x_n) = \displaystyle\sum_{i=1}^{n} x_i^k. \end{cases}$$

Those polynomials satisfies the Newton's identities:

$$k e_k - p_1 e_{k-1} + p_2 e_{k-2} - \cdots + (-1)^k p_k = 0, \quad k \in \mathbb{N}^*. \tag{2}$$

where $e_k = 0$ for $k > n$. Using those identities for $k \in \{n+1, \ldots, 2n - r_0\}$, one obtains:

$$\sum_{i=k-n}^{k} (-1)^{i-1} e_{k-i} p_i = 0.$$

We define

$$M(x_1, \ldots, x_n) := \begin{pmatrix} p_1 & -p_2 & \cdots & (-1)^n p_{n+1} \\ -p_2 & p_3 & \cdots & (-1)^{n+1} p_{n+2} \\ \vdots & \vdots & \ddots & \vdots \\ (-1)^{n-r_0} p_{n+1-r_0} & \cdots & \cdots & (-1)^{2n-r_0} p_{2n+1-r_0} \end{pmatrix}.$$

The left sub matrix of M of size $(n+1-r_0) \times (n+1-r_0)$ will be noted $P(x_1, \ldots, x_n)$ and the right sub matrix by $R(x_1, \ldots, x_n)$. Lemma 1 gives the generalisation of the Leverrier lemma in matrix forms and it is proved in [18]:

Lemma 1 ([18]). *The Newton's polynomials satisfies:*

$$M(x_1,\dots,x_n)\begin{pmatrix} e_n \\ e_{n-1} \\ \vdots \\ e_1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}. \tag{3}$$

and

$$\det(P(x_1,\dots,x_n)) = P_{n+1-r_0,n}(x_1,\dots,x_n)$$

where

$$P_{d,n}(x_1,\dots,x_n) = \sum_{1\le i_1<\cdots<i_d\le n} e_d(x_{i_1},\cdots,x_{i_d}) \prod_{1\le k<j\le d} (x_{i_k}-x_{i_j})^2.$$

Thanks to the expression of the previous determinant, we have the following Proposition 5:

Proposition 5 *Let $A \in M_n(\mathbb{F}_q)$ with eigenvalues $\lambda_A = (\lambda_1,\dots,\lambda_n) \in \overline{\mathbb{F}_q}^n$. Assume that $P_{n+1-r_0,n}(\lambda_A) \neq 0$ then the coefficients of the characteristic polynomial of A can be computed only with rational operations of $t_j := \mathrm{tr}(A^j)$ for $1 \le j \le 2n+1-r_0$.*

Proof. Using Equation 3 at λ_A gives the following systems:

$$\begin{pmatrix} t_1 & -t_2 & \cdots & (-1)^n t_{n+1} \\ -t_2 & t_3 & \cdots & (-1)^{n+1} t_{n+2} \\ \vdots & \vdots & \ddots & \vdots \\ (-1)^{n-r_0} t_{n+1-r_0} & \cdots & \cdots & (-1)^{2n-r_0} t_{2n+1-r_0} \end{pmatrix} \begin{pmatrix} e_n(\lambda_A) \\ e_{n-1}(\lambda_A) \\ \vdots \\ e_1(\lambda_A) \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}. \tag{4}$$

To compute the coefficients of χ_A, we need to evaluate $e_i(\lambda_A)$ with only t_j. For $k \in \{1,\cdots,r_0-1\}$, using 2 and $p_k(\lambda_A) = t_k$, we can recursively compute $e_k(\lambda_A)$ with t_j for $1 \le j \le k$ since k is invertible in $\mathbb{F}_q$. For $k \ge r_0$, with equation 4, we deduce the following relations:

$$e_k(\lambda_A) = -\sum_{i=0}^{r_0-1} (P(\lambda_A)^{-1} R(\lambda_A))_{n-k+1,i} \times e_{r_0-1-i}(\lambda_A)$$

where the matrix $P(\lambda_A)$ is invertible since $\det(M(\lambda_A)) = P_{n+1-r_0,n}(\lambda_A) \neq 0$. □

Remark 3. To compute $\det(A) = e_n(\lambda_A)$, a naive approach consists of using Proposition 5, but as soon as an eigenvalue is repeated with multiplicity at most r_0, the condition will not be satisfied. We can try to resolve the problem by

multiplying A by a random matrix, which solves the problem as soon as 0 is not an eigenvalue with multiplicity $r > r_0$. However, when the rank of the matrix is low, we fall into this case, which poses a problem. One trick for dealing with all possible cases is to consider the matrix $B = (Ay + C)$, this idea was in the original article [16] where they have specified an explicit choice of C without many details.

Let $R = \mathbb{F}_q[y]/(y^{n+1})$, we will use the following property of this ring:

Proposition 6 *Let $P \in R$, then P is invertible if and only if $P(0) \neq 0$. In this case, if $P = a_0(1 - yQ)$ then $P^{-1} = a_0^{-1} \sum_{k=0}^{n} (yQ)^k$.*

Proof. The proof is the following computation:

$$\begin{aligned} P \times \left(a_0^{-1} \sum_{k=0}^{n} (yQ)^k \right) &= \sum_{k=0}^{n} (yQ)^k - \sum_{k=0}^{n} (yQ)^{k+1} \\ &= 1 - y^{n+1} Q^{n+1} \\ &= 1. \end{aligned}$$

□

Let $B = Ay + C$, to apply Proposition 5 on B, by Proposition 6, we need to choose C which verifies that $\det(P_B)(0) = \det(P_C) \neq 0$. This can be done with Lemma 2:

Lemma 2. *Let Q a monic irreducible polynomial of degree $n - r_0 + 1$ over $\mathbb{F}_q$ and $T = X^{r_0 - 1}Q$. Let C be the companion matrix associated to T then $P_{n+1-r_0,n}(\lambda_C) \neq 0$.*

Proof. The eigenvalues of C are exactly the roots of T which are the roots of Q and 0 with multiplicity $r_0 - 1$. Since Q is monic irreducible, the roots of Q are of multiplicity 1, we will note those $\alpha = (\alpha_1, \ldots, \alpha_{n-r_0+1})$. Then

$$P_{n+1-r_0,n}(\lambda_C) = e_{n-r_0+1}(\alpha) \prod_{i<j} (\alpha_i - \alpha_j)^2 \neq 0.$$

□

Applying these theoretical results yields Algorithm 1, which allows secure computation of the determinant.

Algorithm 1 `Computing det`

Input: $[\![A]\!]$ with $A \in M_n(\mathbb{F}_q)$.
Output: $[\![\det(A)]\!]$.

1: Generate matrix $C \in GL_n(\mathbb{F}_q)$ as in Lemma 2.
2: Compute $[\![B]\!] \leftarrow [\![A]\!]y + C$
3: In R : Compute $[\![\mathrm{tr}(B)]\!], \ldots, [\![\mathrm{tr}(B^{2n-r_0})]\!]$.
4: In R : Compute recursively $[\![e_1(\lambda_B)]\!], \ldots, [\![e_{r_0-1}(\lambda_B)]\!]$ with Newton's identities.
5: Generate a random matrix $[\![S]\!] \in GL_n(R)$.
6: In R : Compute and open $[\![T]\!] \leftarrow [\![S]\!] \cdot [\![P(\lambda_B)]\!]$.
7: In R : Compute T^{-1}.
8: In R : Compute $[\![P(\lambda_B)^{-1}]\!] \leftarrow T^{-1} \cdot [\![S]\!]$.
9: In R : Compute $N := -\sum_{i=0}^{r_0-1}([\![P(\lambda_B)^{-1}]\!][\![R(\lambda_B)]\!])_{1,i} \times [\![e_{r_0-1-i}(\lambda_B)]\!]$.
10: Identify the $n-$th last coefficient N_n of N and output $[\![N_n]\!] = [\![e_n(\lambda_A)]\!] = [\![\det(A)]\!]$.

Proposition 7 *For any $A \in \mathcal{M}_n(\mathbb{F}_q)$, Algorithm 1 computes $[\![\det(A)]\!]$, a shared representation of $\det(A)$. Algorithm 1 requires $O(1)$ rounds of communications and $O(n^{3.5})$ communications.*

Proof. The correction and termination of the algorithm follows from Proposition 5 and the fact that $\det(B) = \det(A)y^n + \cdots + \det(C)$. Step 3 can be done in $O(n^3 \times \sqrt{2n+1-r_0}) = O(n^{3.5})$ communications and $O(1)$ round of communication, using a Baby step Giant step approach with matrices in R and the Proposition 3. Step 4 will require only $r_0 - 1$ communications and $O(1)$ round of communication. Step 5 can be done by generating randomly $[\![M_0]\!] \in GL_n(\mathbb{F}_q)$, $[\![M_1]\!], \ldots, [\![M_{n-1}]\!] \in M_n(\mathbb{F}_q)$ and set $[\![S]\!] = [\![M_0]\!] + \cdots + [\![M_{n-1}]\!]y^{n-1}$ so the complexity will be the same as the one used to generate M_0, therefore $O(n^2)$ communications and $O(1)$ round of communication. Step 6 will require 1 shared matrix multiplication, so $O(n^3)$ communications and $O(1)$ round of communication with the proposition 2. Step 9 will also require 1 shared matrix multiplication and 1 shared matrix-vector multiplication, so $O(n^3)$ communications and $O(1)$ round of communication. Finally, all steps combined require $O(n^{3.5})$ communications and $O(1)$ communication. □

3.3 A revisit of Samuelson-Berkowitz algorithm

In this section, it will be shown that it is possible to achieve $O(n^3)$ communications complexity using the relatively unknown Samuelson-Berkowitz algorithm, which use only multiplications to compute the characteristic polynomial of a matrix. The principle of this algorithm can be explained by Theorem 1:

Theorem 1 ([17]). *Let $A_0 \in \mathcal{M}_n(\mathbb{F}_q)$, let P the vector of coefficients of χ_{A_0} if $A_0 = \begin{pmatrix} a_{11} & R \\ S & A_1 \end{pmatrix}$ then $P = T_0 Q$ where Q is the vector of coefficients of χ_{A_1} and T_0 the lower triangular Toepliz matrix defined by*

– *if* A_0 *is* 1×1,

$$T_0(A_0) = \begin{pmatrix} 1 \\ -a_{11} \end{pmatrix}$$

– *if* A_0 *is* 2×2,

$$T_0(A_0) = \begin{pmatrix} 1 & 0 \\ -a_{11} & 1 \\ -RC & -a_{11} \end{pmatrix}$$

– *In general,*

$$T_0(A_0) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{11} & 1 & 0 & 0 & 0 & 0 & 0 \\ -RS & a_{11} & 1 & 0 & 0 & 0 & 0 \\ -RA_1S & -RS & a_{11} & 1 & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ -RA_1^{n-3}S & -RA_1^{n-4}S & \cdots & -RS & a_{11} & 1 & 0 \end{pmatrix}. \tag{5}$$

Let A_i be the bottom right sub matrix of size $n-i$ of A_0. Applying recursively Theorem 1 yields

$$P = T_0 T_1 \cdots T_{n-1}$$

with

$$T_i = T_0(A_i) \text{ and } A_i = \begin{pmatrix} a_{ii} & R_i \\ S_i & A_{i+1} \end{pmatrix}.$$

Using the structure of the Toepliz matrices yields Algorithm 2 which allows secure computation of the determinant with a better multiplication complexity than Algorithm 1.

Algorithm 2 `Computing det`

Input: $[\![A]\!]$ with $A_0 \in M_n(\mathbb{F}_q)$.
Output: $[\![\det(A)]\!]$.

1: **for** $0 \leq i \leq n-1$ **do**
2: Compute $[\![-R_iS_i]\!] \leftarrow [\![-R_i]\!][\![S_i]\!]$
3: **for** $1 \leq k \leq n-i-3$ **do**
4: Compute $[\![-R_iA_{i+1}^kS_i]\!] \leftarrow [\![-R_i]\!][\![A_{i+1}^k]\!][\![S_i]\!]$
5: Locally construct $[\![T_i]\!]$ with step 2, step 3 using 5.
6: Compute $[\![P]\!] = [\![T_0]\!][\![T_1]\!] \cdots [\![T_{n-1}]\!]$
7: Identify the $n-$th last coefficient of $[\![P]\!]$ and output $[\![P_n]\!] = [\![\det(A)]\!]$.

Proposition 8 *For any* $A \in \mathcal{M}_n(\mathbb{F}_q)$*, Algorithm 2 computes* $[\![\det(A)]\!]$*, a shared representation of* $\det(A)$*. Algorithm 2 requires* $O(n^3)$ *communications and* $O(n)$ *rounds of communications.*

Proof. The correction and termination of Algorithm 2 follows from Theorem 1. The loop in step 3 can be done with Proposition 3 and Proposition 1 by computing all A_i^k with $O(1)$ round of communication and $O((n-i)(n-i-3))$ communication. The main loop in step 1 will require to repeat this n times so at the end of both loops, it will cost $O(n^3)$ communications and $O(n)$ round of communications. Step 7 requires n matrix multiplications that each requires $O(n^2)$ communications. Finally, Algorithm 2 requires a total of $O(n^3)$ communications and $O(n)$ rounds of communications.

□

3.4 New algorithms for linear solving

The previous section provided two algorithms for securely computing the determinant. From these, one can directly derive two algorithms for solving a linear system. Algorithm 3 formally describes both methods.

Algorithm 3 `Securely solving` $A\mathbf{x} = \mathbf{b}$

Input: $([\![A]\!], [\![\mathbf{b}]\!])$ with $A \in M_{s\times t}(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^s$ $(s \leq t)$.
Output: $\perp$ or $[\![\mathbf{x}]\!]$ such that $A\mathbf{x} = \mathbf{b}$.

1: Generate random matrices $[\![R]\!] \in M_s(\mathbb{F}_q)$ and $[\![S]\!] \in M_t(\mathbb{F}_q)$.
2: Compute $[\![T]\!] \leftarrow [\![R]\!] \cdot [\![A]\!] \cdot [\![S]\!]$
3: Choose random sets $I \subset [1, s]$ and $J \subset [1, t]$ of size s.
4: Compute $[\![\det(T_{I,J})]\!]$ with Algorithm 1 or 2. If $[\![\det(T_{I,J})]\!] = 0$, output $\perp$.
5: Open the matrix $[\![T]\!]$.
6: In the clear, compute a right inverse T^{-1} of T.
7: Compute $[\![A^{-1}]\!] \leftarrow [\![S]\!] \cdot T^{-1} \cdot [\![R]\!]$.
8: Generate random scalars $[\![z_1]\!], \ldots, [\![z_{t-s}]\!]$.
9: Compute $[\![\mathbf{z}]\!] \leftarrow \sum_{i=1}^{t-s} [\![z_i]\!] \cdot \beta_i$ where β_i is a basis for $\ker(T)$.
10: Compute and output $[\![\mathbf{x}]\!] \leftarrow [\![A^{-1}]\!] \cdot [\![\mathbf{b}]\!] + [\![S]\!] \cdot [\![\mathbf{z}]\!]$.

Proposition 9 *For any $A \in \mathcal{M}_{s\times t}(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^s$, Algorithm 3 computes $[\![\mathbf{x}]\!]$, a shared representation of $\mathbf{x}$ that verifies $A\mathbf{x} = \mathbf{b}$ without revealing information or $\perp$. Depending on which algorithm is used at step 2, Algorithm 3 requires $O(s^{3.5})$ communications and $O(1)$ rounds of communications or $O(s^3)$ communications and $O(s)$ rounds of communications.*

Proof. The main complexity of the algorithm is the computation of the determinant in step 4. This need $O(s^{3.5})$ with Algorithm 1 and $O(s^3)$ with Algorithm 2. The other steps need at most $O(s^2)$ using Proposition 1.

Remark 4. An open question is remaining. Is it possible to achieve a complexity of $O(s^{2.5})$ as in the case of big characteristic? In any case, such a solution, even if it is secure, may be too expensive. The solution in the following section seems more realistic.

3.5 Noise on the rank leakage

We propose a way for players to randomly choose between the matrix to be inverted and a random matrix that is never invertible, obliviously. This technique allows to add "noise" to the list of recovered ranks. This method is by construction at least as safe as the one described in [5] and we believe that it drastically increases the complexity of any attack, under reasonable assumptions made explicit in Section 4. Algorithm 4 describes a way for players to choose a matrix at random. This algorithm will be used in Algorithm 5 to solve a linear system.

Algorithm 4 $\texttt{Randomselection}_{1/2}$

Input: $([\![A]\!], [\![B]\!])$ with $A \in M_{s\times t}(\mathbb{F}_q)$ and $B \in M_{s\times t}(\mathbb{F}_q)$.
Output: $[\![A]\!]$ or $[\![B]\!]$ with probability $1/2$.

1: Generate a shared random $[\![t]\!] \in \mathbb{F}_2$.
2: Compute $[\![T]\!] := [\![t]\!][\![A]\!] + (1 - [\![t]\!])[\![B]\!]$
3: Output $[\![T]\!]$.

Remark 5. Despite the notations and the fact that $[\![T]\!] = [\![A]\!]$ or $[\![T]\!] = [\![B]\!]$, it is important to note that due to the secret random t, the $i-$th player cannot know whether T_i is a share of A or B. In particular, T_i is different from A_i and B_i a priori.

Algorithm 5 $\texttt{Solving } A\mathbf{x} = \mathbf{b}$

Input: $([\![A]\!], [\![\mathbf{b}]\!])$ with $A \in M_{s\times t}(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^s$.
Output: $\perp$ or $[\![\mathbf{x}]\!]$ such that $A\mathbf{x} = \mathbf{b}$.

1: Generate random matrices $[\![R]\!] \in M_s(\mathbb{F}_q)$ and $[\![S]\!] \in M_t(\mathbb{F}_q)$.
2: Compute $[\![T^{''}]\!] \leftarrow [\![R]\!] \cdot [\![A]\!] \cdot [\![S]\!]$.
3: Generate $[\![u_1]\!], [\![v_1]\!], \ldots, [\![u_{n-1}]\!], [\![v_{n-1}]\!]$, with $[\![u_i]\!] \in \mathbb{F}_q^s$ and $[\![v_i]\!] \in \mathbb{F}_q^t$.
4: Securely compute $[\![T']\!] := \sum_{i=1}^{n-1} [\![u_i]\!][\![^t v_i]\!] \in M_{s\times t}(\mathbb{F}_q)$.
5: Open the matrix $[\![T]\!] := \texttt{Randomselection}_{1/2}([\![T^{''}]\!], [\![T']\!])$.
6: **if** $\text{rank}(T) < s$ **then** output $\perp$.
7: In the clear, compute a right inverse T^{-1} of T.
8: Compute $[\![A]\!]^{-1} \leftarrow [\![S]\!] \cdot T^{-1} \cdot [\![R]\!]$.
9: Generate random scalars $[\![z_1]\!], \ldots, [\![z_{t-s}]\!]$.
10: Compute $[\![\mathbf{z}]\!] \leftarrow \sum_{i=1}^{t-s} [\![z_i]\!] \cdot \beta_i$ where β_i is a basis for $\ker(T)$.
11: Compute $[\![\mathbf{x}]\!] \leftarrow [\![A^{-1}]\!] \cdot [\![\mathbf{b}]\!] + [\![S]\!] \cdot [\![\mathbf{z}]\!]$. Output $[\![x]\!]$.

Proposition 10 *For any $A \in \mathcal{M}_{s\times t}(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^s$, Algorithm 5 computes $[\![\mathbf{x}]\!]$, a shared representation of $\mathbf{x}$ that verifies $A\mathbf{x} = \mathbf{b}$. With probability $1/2$, Algorithm 5 reveals the rank of A or the rank of a random non-invertible matrix T'. It requires $O(1)$ round of communications and $O(st)$ communication.*

Proof. At the end of the protocol, when $\perp$ has not been returned. One has:

$$\begin{aligned} A\mathbf{x} &= A(A^{-1}\mathbf{b} + S\mathbf{z}) \\ &= R^{-1}(RAST^{-1}R\mathbf{b} + RAS\mathbf{z}) \\ &= R^{-1}(I_s R\mathbf{b} + 0) \\ &= \mathbf{b}. \end{aligned}$$

The complexity of each step is computed with Proposition 1 and is at most $O(st)$ communication and $O(1)$ round of communication.

□

Alternative selection functions The $\texttt{Randomselection}_{1/2}$ function describes a way to select equiprobably between two shared matrices in a secure manner. For efficiency, it is useful to provide a function that selects a shared matrix securely with a given probability $p \in [0, 1]$. One way to do this is to generate $[\![t_1]\!], \ldots, [\![t_k]\!] \in \mathbb{F}_2$ and return $(1 + [\![t_1]\!] \times \cdots \times [\![t_k]\!])[\![A]\!] + [\![t_1]\!] \times \cdots \times [\![t_k]\!][\![B]\!]$. It will output $[\![B]\!]$ with probability $1/2^k$. Other more efficient selection functions can be used instead, but for the sake of simplicity we have chosen to present the one we developed ourselves.

4 Application to UOV and MAYO

As explained in the introduction, solving shared linear system is a problem that naturally appears in any UOV based threshold algorithm. We recall the notations used in UOV and briefly explain how UOV works, then we detail existing algorithms to solve linear systems obliviously and compare them with the solutions presented in previous works.

4.1 Summary of UOV

We denote by $\mathbb{F}_q$ the finite field with q elements. To a homogeneous multivariate quadratic polynomial $p(\mathbf{x})$, we can associate his polar form given by:

$$p'(\mathbf{x}, \mathbf{y}) := p(\mathbf{x} + \mathbf{y}) - p(\mathbf{x}) - p(\mathbf{y}).$$

The polar form of a multivariate quadratic map $\mathcal{P}(\mathbf{x}) = (p_1(\mathbf{x}), \ldots, p_m(\mathbf{x}))$ is given by:

$$\mathcal{P}'(\mathbf{x}, \mathbf{y}) = (p'_1(\mathbf{x}, \mathbf{y}), \ldots, p'_m(\mathbf{x}, \mathbf{y})).$$

The UOV trapdoor function is a quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ which vanishes on a secret vector space $\mathcal{O}$ of dimension o (usually with $o = m$). When $\mathcal{O}$ is known, it is easy to compute a preimage for $\mathbf{t} \in \mathbb{F}_q^m$ by $\mathcal{P}$: let $\mathbf{v} \in \mathbb{F}_q^n$, try to solve the following linear system for $\mathbf{o} \in \mathcal{O}$:

$$\mathcal{P}(\mathbf{v} + \mathbf{o}) = \mathcal{P}(\mathbf{v}) + \mathcal{P}(\mathbf{o}) + \mathcal{P}'(\mathbf{v}, \mathbf{o}) = \mathcal{P}(\mathbf{v}) + \mathcal{P}'(\mathbf{v}, \mathbf{o}) = \mathbf{t}.$$

It is a system of m equations and m variables over $\mathbb{F}_q$. Knowledge of this trapdoor naturally allows a signature scheme to be constructed, as follows.

Public Key: The UOV trapdoor function $\mathcal{P}$.

Secret Key: A vector space $\mathcal{O}$ of dimension m such that $\mathcal{P}(\mathcal{O}) = 0$.

Signature algorithm: Let $\mathbf{y} \in \mathbb{F}_q^m$ be a message. A valid signature of $\mathbf{y}$ is an element $\mathbf{x} \in \mathbb{F}_q^n$ such that

$$\mathcal{P}(\mathbf{x}) = \mathbf{y}.$$

The signer can use the knowledge of $\mathcal{O}$ to easily compute a signature for $\mathbf{y}$ as follows. Let $\mathbf{v} \in \mathbb{F}_q^n$, try to solve the following linear system for $\mathbf{o} \in O$:

$$\mathcal{P}(\mathbf{v} + \mathbf{o}) = \mathcal{P}(\mathbf{v}) + \mathcal{P}(\mathbf{o}) + \mathcal{P}'(\mathbf{v}, \mathbf{o}) = \mathcal{P}(\mathbf{v}) + \mathcal{P}'(\mathbf{v}, \mathbf{o}) = \mathbf{y}$$

It is a system of m equations and m variables over $\mathbb{F}_q$. If it has no solution, the signer pick a new value for $\mathbf{v}$ and try again.

Verification algorithm: Simply check that $\mathcal{P}(\mathbf{x}) = \mathbf{y}$.

A few examples of parameters: Table 1 shows parameter sets for the UOV scheme and the MAYO scheme (which is a variant of UOV), for various security levels. We will not go into detail on how the MAYO scheme works.

Parameters	UOV_1	UOV'_1	UOV_3	UOV_5	MAYO_1	MAYO_1	MAYO_3	MAYO_5
n	112	160	184	244	66	78	99	133
m	44	64	72	96	64	64	96	128
o	44	64	72	96	8	18	10	12
q	256	16	256	256	16	16	16	16

Table 1. Summary of current UOV and MAYO parameters.

4.2 Existing threshold work on UOV

In the literature, two papers study the thresholdization of UOV-based signature algorithms: [7] and [5]. Both papers propose customized methods that are more

efficient than general multiparty computation one. In both [7] and [5], proposes solution to Problem 1. We propose to recall, briefly, why Problem 1 appears naturally in any threshold construction of UOV. Let $\mathcal{T}_1, \ldots, \mathcal{T}_r$ be r players who want to conjointly compute a UOV signature. Each of the $\mathcal{T}_i'$s knows $(\mathcal{P}, \mathcal{O}_i)$, $\mathcal{P}$ being the UOV public key and $\mathcal{O}_i \subset \mathbb{F}_q^n$ being a additive share of the secret space $\mathcal{O}$ of dimension o. $[\![\mathcal{O}]\!]$ will denote the shared representation of $\mathcal{O}$, which is defined formally as $[\![\mathcal{O}]\!] := (\mathcal{O}_1, \ldots, \mathcal{O}_r)$ with:

$$\forall i \in \{1, \ldots, r\},\ \mathcal{O}_i = \mathrm{Vect}_{\mathbb{F}_q}(\mathbf{o}_1^{[i]}, \ldots, \mathbf{o}_o^{[i]}) \text{ and } \forall j \in \{1, \ldots, o\},\ \mathbf{o}_j = \sum_{i=1}^{r} \mathbf{o}_j^{[i]}.$$

Therefore, for a message $\mathbf{y} \in \mathbb{F}_q^n$ known to the $\mathcal{T}_i'$s, each player draws a random $\mathbf{v}_i \in \mathbb{F}_q^n$. Then they jointly attempt to solve the following linear system:

$$\mathcal{P}'(\sum_{i=1}^{r} \mathbf{v}_i, \sum_{i=1}^{r} \mathbf{o}_i) = \mathbf{y} + \mathcal{P}(\sum_{i=1}^{r} \mathbf{v}_i).$$

It is important to note that $\mathbf{v}$ cannot be revealed to the $\mathcal{T}_i'$s. If $\mathbf{v}$ is known, once the signature $\mathbf{x} = \mathbf{v} + \mathbf{o}$ has been computed, the $\mathcal{T}_i'$s know a vector $\mathbf{o} \in \mathcal{O}$ and they can retrieve the secret key, according to a classic result described in [14]. Since $\mathbf{y}$ is known and $\mathcal{P}$ is a polynomial, it is equivalent as trying to solve the following system, which is written with a slight abuse of notation:

$$\mathcal{P}'([\![\mathbf{v}]\!], [\![\mathbf{o}]\!]) = [\![\mathbf{y} + \mathcal{P}(\mathbf{v})]\!].$$

Thus, the $\mathcal{T}_i'$s must solve a threshold linear system without revealing any information, exactly as formulated in Problem 1.

Solution proposed by [7]: In [7], the authors propose Algorithm 6, described below.

Algorithm 6 `Solving` $A\mathtt{x} = \mathtt{b}$

Input: $([\![A]\!], [\![\mathbf{b}]\!])$ with $A \in M_s(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^s$.
Output: $\perp$ or $[\![\mathbf{x}]\!]$ such that $A\mathbf{x} = \mathbf{b}$.

1: Generate a random $s \times s$ shared matrix $[\![R]\!]$.
2: Compute $[\![T]\!] \leftarrow [\![A]\!] \cdot [\![R]\!]$.
3: Open the matrix $[\![T]\!]$. If $\det(T) = 0$ output $\perp$.
4: In the clear, compute T^{-1}.
5: Compute $[\![\mathbf{t}]\!] \leftarrow T^{-1}[\![\mathbf{b}]\!]$.
6: Compute and output $[\![\mathbf{x}]\!] \leftarrow [\![R]\!] \cdot [\![\mathbf{t}]\!]$.

Proposition 11 ([7]) *For any $A \in \mathcal{M}_s(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^s$, Algorithm 6 computes $[\![\mathbf{x}]\!]$, a shared representation of $\mathbf{x}$ that verifies $A\mathbf{x} = \mathbf{b}$. Algorithm 6 requires $O(s^2)$*

communications and $O(1)$ rounds of communications. Besides, Algorithm 6 reveals information on the image and kernel of A.

Proof. At the end of the protocol, when $\perp$ has not been returned. One has:

$$[\![\mathbf{x}]\!] = [\![RT^{-1}\mathbf{b}]\!] = [\![RR^{-1}A^{-1}\mathbf{b}]\!] = [\![A^{-1}\mathbf{b}]\!].$$

Three rounds of communications are required: two for the products on the shares in steps 2 and 6 respectively, and one more to open value T in step 3. Finally, step 2 requires s^2 communications, while step 6 requires s. When the protocol returns $\perp$, the matrix T which has a non-trivial kernel and image has been revealed. For instance, if R is invertible it reveals all the image and the kernel of A. □

Solution proposed by [5]: In [5], the authors propose Algorithm 7, described below.

Algorithm 7 `Solving` $A\mathtt{x} = \mathtt{b}$

Input: $([\![A]\!], [\![\mathbf{b}]\!])$ with $A \in M_{s\times t}(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^s$.
Output: $\perp$ or $[\![\mathbf{x}]\!]$ such that $A\mathbf{x} = \mathbf{b}$.

1: Generate random matrices $[\![R]\!] \in M_s(\mathbb{F}_q)$ and $[\![S]\!] \in M_t(\mathbb{F}_q)$.
2: Compute $[\![T]\!] \leftarrow [\![R]\!] \cdot [\![A]\!] \cdot [\![S]\!]$.
3: Open the matrix $[\![T]\!]$. If $\text{rank}(T) < s$ output $\perp$.
4: In the clear, compute a right inverse T^{-1} of T.
5: Compute $[\![A^{-1}]\!] \leftarrow [\![S]\!] \cdot T^{-1} \cdot [\![R]\!]$.
6: Generate random scalars $[\![z_1]\!], \ldots, [\![z_{t-s}]\!]$.
7: Compute $[\![\mathbf{z}]\!] \leftarrow \sum_{i=1}^{t-s} [\![z_i]\!] \cdot \beta_i$ where β_i is a basis for $\ker(T)$.
8: Compute and output $[\![\mathbf{x}]\!] \leftarrow [\![A^{-1}]\!] \cdot [\![\mathbf{b}]\!] + [\![S]\!] \cdot [\![\mathbf{z}]\!]$.

Proposition 12 ([5]) *For any $A \in \mathcal{M}_{s\times t}(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^s$, Algorithm 7 computes $[\![\mathbf{x}]\!]$, a shared representation of $\mathbf{x}$ that verifies $A\mathbf{x} = \mathbf{b}$. Algorithm 7 requires $O(st)$ communications and $O(1)$ rounds of communications. If A is not invertible, Algorithm 7 reveals the image and kernel of $T = RAS$. In particular, when R and S are invertible, it reveals the rank of A.*

Proof. At the end of the protocol, when $\perp$ has not been returned, one has:

$$
\begin{aligned}
A\mathbf{x} &= A(A^{-1}\mathbf{b} + S\mathbf{z}) \\
&= A(ST^{-1}R\mathbf{b}) + AS\mathbf{z} \\
&= R^{-1}(RAST^{-1}R\mathbf{b} + RAS\mathbf{z}) \\
&= R^{-1}(TT^{-1}R\mathbf{b} + T\mathbf{z}) \\
&= R^{-1}(I_s R\mathbf{b} + 0) \\
&= \mathbf{b}.
\end{aligned}
$$

The communication complexity depends on the multiplication of matrix of size $s \times t$ with matrix or vector of compatible size, which can be done in $O(st)$ with Proposition 1. The protocol reveals to the players the matrix $T = RAS$ and if R and S are invertible, the rank of T is that of A. □

4.3 Comparison of different methods

Security guarantee of Algorithm 5 Algorithm 7 reveals the rank of matrix A when A is not invertible, since this matrix is directly related to the secret vector space $\mathcal{O}$, an honest but curious player could use this information to find the secret key. This information is formally studied in [5], more precisely:

- The authors describe the Arithmetic Black Box (ABB) model and the associated $\mathcal{F}_{\mathbf{ABB}}$ functionality. They then model Algorithm 7 as an extended functionality called $\mathcal{F}_{\mathbf{ABB-solve}}$.
- Then, in section 5.5 the authors characterize the leakage of the $\mathcal{F}_{\mathbf{ABB-solve}}$ functionality more precisely. On page 22, one reads: "$\mathcal{F}_{\mathbf{ABB-solve}}$ not only produces a signature, but also a set of positive integers Leaks representing the rank of each matrix A that turned out to be rank-deficient. This arises from the difficulty of determining whether a given secret matrix is full rank without leaking the rank itself. Now, the "most ideal" functionality that models OV-based signing should produce the signature, and nothing else. Since our functionality technically leaks more than such an ideal setting, we discuss the potential implications of this in terms of the unforgeability of the underlying scheme.".

Theorem 1 shows that Algorithm 5 is at least as secure as the one described in [5].

Theorem 1 *Let* $\mathbf{Leak}$ *be a sequence of matrices resulting from t executions of* $\mathcal{F}_{\mathbf{ABB-solve}}$ *and* $\overline{\mathbf{Leak}}$ *be a sequence of matrices resulting from t executions of* $\overline{\mathcal{F}}_{\mathbf{ABB-solve}}$*, the functionality where Algorithm 7 has been replaced by Algorithm 5. Let us assume that there is no distinction between a* $\overline{\mathbf{Leak}}$ *element that comes from* T' *or* T'' *in Algorithm 5. Then, for any attacker* $\mathcal{A}$ *recovering the secret vector space* $\mathcal{O}$ *from any collection of matrices* $\{\mathbf{Leak}\}_{i\in I}$ *with complexity* $P(|I|)$*,* $\mathcal{A}$ *can recover* $\mathcal{O}$ *from any collection* $\{\overline{\mathbf{Leak}}\}_{j\in J}$ *with probability*

$1/2$, $|J| \geq |I|$ *and with an average complexity of* $O\left(\left(1+\frac{1}{pr}\right)^{|I|} P(|I|)\right)$ *with* pr *defined by:*

$$pr := \left\{ \binom{3}{1}\alpha(1-\alpha)^2 + \binom{3}{2}\alpha^2(1-\alpha) + \binom{3}{3}\alpha^3 \right\},$$

where α *denotes the probability for a random matrix in* $M_n(\mathbb{F}_q)$ *to be invertible.*

Proof. Let $\{\overline{\mathbf{Leak}}\}_{j\in J}$ be a set of leaks from algorithm 5. Since $\mathcal{A}$ cannot differentiate between an element of $\overline{\mathbf{Leak}}$ that comes from T' or T'', all she can do is obtain a set I, such that $I \subset J$ and such that for all i in I, $\overline{\mathbf{Leak}}_i$ came from a leak of T''. Formally, by definition the matrix T' is never invertible, while when A is assumed to be random, T'' is invertible with probability:

$$pr := \left\{ \binom{3}{1}\alpha(1-\alpha)^2 + \binom{3}{2}\alpha^2(1-\alpha) + \binom{3}{3}\alpha^3 \right\}$$

Therefore,

$$\mathbb{P}(T \in \overline{\mathbf{Leak}}) = \mathbb{P}(\det(T) = 0) = \frac{1}{2} + \frac{1}{2}pr,$$

and using the law of total probability:

$$\mathbb{P}(T = T'') = \left(\frac{1}{2} + \frac{1}{2}pr\right)\mathbb{P}\left(T = T'' \mid T \in \overline{\mathbf{Leak}}\right) + \left(1 - \left(\frac{1}{2} + \frac{1}{2}pr\right)\right) \times 1.$$

After simplification and because $\mathbb{P}(T = T'') = 1/2$, one obtains:

$$\mathbb{P}\left(T = T'' \mid T \in \overline{\mathbf{Leak}}\right) = \frac{pr}{1+pr}.$$

Thus, the probability that $\mathcal{A}$ will succeed in its attack by taking a random set I, of size $|I|$, is:

$$\left(\frac{pr}{1+pr}\right)^{|I|}.$$

She will succeed at least once in k attempts with a probability of:

$$1 - \left(1 - \left(\frac{pr}{1+pr}\right)^{|I|}\right)^k$$

To obtain a probability of success of, say, $1/2$, set k such that:

$$k \log\left(1 - \left(\frac{pr}{1+pr}\right)^{|I|}\right) \leq \log\left(\frac{1}{2}\right).$$

Using the series expansion of the logarithm, one can simplify this inequality to obtain:

$$k\left(\frac{pr}{1+pr}\right)^{|I|} \geq \log(2).$$

Finally $\mathcal{A}$ will find space $\mathcal{O}$, with a probability $1/2$ from $\{\overline{\mathbf{Leak}}\}_{j\in J}$, with a total complexity of:

$$\left(1+\frac{1}{pr}\right)^{|I|} P(|I|).$$

□

Remark 6. Theorem 1 allow to quantify the proportion of noise in the set of $\overline{\mathbf{Leak}}$ matrices, given by several executions of Algorithm 5. Nevertheless, it is interesting to show the concrete noise and added complexity for specific parameters of UOV and MAYO, as is done in Table 2.

Complexity	UOV_1, UOV_3, UOV_5	UOV'_1	MAYO
q	256	16	16
pr	1.15%	15.75%	15.75%
$\frac{pr}{1+pr}$	1.17%	18.69%	18.69%

Table 2. Evaluation of the complexity.

Given these concrete parameters, we believe that the increase in complexity is detrimental for an attacker seeking to obtain information about the secret vector space $\mathcal{O}$ from a collection of non-invertible matrices generated by Algorithm 5.

Comparison of different algorithms Finally, Table 3 summarizes the various algorithms presented, along with their respective complexities and leakages to solve Problem 1.

Algorithm	Leakage	Communications	Rounds
Cozzo *et al.* (Alg 6)	$\text{Im}(A)$ and $\text{Ker}(A)$	$O(s^2)$	$O(1)$
Celi *et al.* (Alg 7)	$\text{rank}(A)$	$O(st)$	$O(1)$
Noisy rank (Alg 5)	$\text{rank}(A)$ with noise	$O(st)$	$O(1)$
Generalized Newton (Alg 3 using 1)	None	$O(s^{3.5})$	$O(1)$
Revisited Berkowitz (Alg 3 using 2)	None	$O(s^3)$	$O(s)$

Table 3. Comparison of the different algorithms.

5 Conclusion and discussion

In this paper we built three different algorithms to limit leakage during the resolution of a shared linear system in small characteristics. The main application

of these results is to eliminate leakage in existing threshold protocols based on UOV or its variants. However, it remains unclear whether this leak is significant. According to the authors of [5] p.5: "we leave as an open problem the design of a protocol for linear system solving that, in cases of rank deficiency, reveals only that the matrix is not full rank".

This article addresses this open problem and determines the computational cost of eliminating such a leakage. Regarding the risk of a leakage of the rank of matrix A, revealing information about A appears to provide non-trivial knowledge about the secret data. However, the rank of A is statistically close to the maximum rank, as shown in [6]. Furthermore, there are currently no attacks that take rank leakage into account to attack existing threshold solutions. Similarly, implementations of UOV or MAYO will leak information about the number of attempts required to obtain a full-rank matrix, and again, there are no known side-channel attacks that exploit this information. Despite the absence of known attacks, to be conservative, we must conjecture (without proof) that rank leakage does not compromise the security of the scheme.

This article proposes several ways to eliminate this leakage. This prevents any future attacks and avoids relying on an unproven assumption. Furthermore, depending on the use case, the proposed algorithms may not be overly costly.

References

1. Beullens, W.: Breaking rainbow takes a weekend on a laptop. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. pp. 464–479. Springer Nature Switzerland, Cham (2022)
2. Beullens, W., Campos, F., Celi, S., Hess, B., Kannwischer., M.J.: Algorithm specifications and supporting documentation (2021), https://pqmayo.org/assets/specs/mayo-round2.pdf, https://pqmayo.org/
3. Beullens, W., Chen, M.S., Ding, J., Gong, B., Kannwischer, M.J., Jatarin, P., Peng, B.Y., Schmidt, D., Shih, C.J., Tao, C., Yang, B.Y.: Algorithm specifications and supporting documentation (2024), https://drive.google.com/file/d/1b4AzNSr-fNBGupP74LIxDorovm6Qe-Sj/view?usp=drive_link, https://www.uovsig.org/
4. Borin, G., Celi, S., del Pino, R., Espitau, T., Niot, G., Prest, T.: Threshold signatures reloaded: ML-DSA and enhanced raccoon with identifiable aborts. Cryptology ePrint Archive, Paper 2025/1166 (2025), https://eprint.iacr.org/2025/1166
5. Celi, S., Escudero, D., Niot, G.: Share the mayo: Thresholdizing mayo. In: Niederhagen, R., Saarinen, M.J.O. (eds.) Post-Quantum Cryptography. pp. 165–198. Springer Nature Switzerland, Cham (2025)
6. Cogliati, B., Fouque, P.A., Goubin, L., Minaud, B.: New security proofs and techniques for hash-and-sign with retry signature schemes. Cryptology ePrint Archive, Paper 2024/609 (2024), https://eprint.iacr.org/2024/609
7. Cozzo, D., Smart, N.P.: Sharing the luov: Threshold post-quantum signatures. In: Albrecht, M. (ed.) Cryptography and Coding. pp. 128–153. Springer International Publishing, Cham (2019)

8. Ding, J., Deaton, J., Vishakha, Yang, B.Y.: The nested subset differential attack. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. pp. 329–347. Springer International Publishing, Cham (2021)
9. Maire, J., Vergnaud, D.: Secure multi-party linear algebra with perfect correctness. IACR Communications in Cryptology (04 2024). https://doi.org/10.62056/avzojbkrz
10. del Pino, R., Espitau, T., Niot, G., Prest, T.: Simple and efficient lattice threshold signatures with identifiable aborts. Cryptology ePrint Archive, Paper 2025/871 (2025), https://eprint.iacr.org/2025/871
11. del Pino, R., Katsumata, S., Niot, G., Reichle, M., Takemure, K.: Unmasking traccoon: A lattice-based threshold signature with an efficient identifiable abort protocol. In: Tauman Kalai, Y., Kamara, S.F. (eds.) Advances in Cryptology – CRYPTO 2025. pp. 423–456. Springer Nature Switzerland, Cham (2025)
12. del Pino, R., Niot, G.: Finally! a compact lattice-based threshold signature. In: Jager, T., Pan, J. (eds.) Public-Key Cryptography – PKC 2025. pp. 169–199. Springer Nature Switzerland, Cham (2025)
13. Preparata, F., Sarwate, D.: An improved parallel processor bound in fast matrix inversion. Information Processing Letters **7**(3), 148–150 (1978). https://doi.org/https://doi.org/10.1016/0020-0190(78)90079-0, https://www.sciencedirect.com/science/article/pii/0020019078900790
14. Pébereau, P.: One vector to rule them all: Key recovery from one vector in uov schemes. In: Post-Quantum Cryptography: 15th International Workshop, PQCrypto 2024, Oxford, UK, June 12–14, 2024, Proceedings, Part II. p. 92–108. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-62746-0_5, https://doi.org/10.1007/978-3-031-62746-0_5
15. Randall, D.: Efficient generation of random nonsingular matrices. Random Structures & Algorithms **4** (01 1993). https://doi.org/10.1002/rsa.3240040108
16. Schönhage, A.: Fast parallel computation of characteristic polynomials by leverrier's power sum method adapted to fields of finite characteristic. In: Lingas, A., Karlsson, R., Carlsson, S. (eds.) Automata, Languages and Programming. pp. 410–417. Springer Berlin Heidelberg, Berlin, Heidelberg (1993)
17. Soltys, M.: Berkowitz's algorithm and clow sequences (2002), https://arxiv.org/abs/math/0201315
18. de Vries, S.: On newton's identities in positive characteristic. Journal of Algebra **668**, 348–364 (Apr 2025). https://doi.org/10.1016/j.jalgebra.2025.01.010, http://dx.doi.org/10.1016/j.jalgebra.2025.01.010