Shared and leakage free MAYO

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.

Although pre-quantum threshold signature algorithms have been extensively studied, the state of the art in the creation of post-quantum threshold algorithms remain sparse. Most studies focus on signature algorithms based on structured lattice problems. In particular, few papers have studied the creation of a threshold algorithm based on UOV, despite the simplicity of the scheme.

This paper proposes various algorithms for a set of parties to solve a shared linear system $A\mathbf{x} = \mathbf{y}$ in finite fields of low characteristic.

The first two algorithms securely calculate the determinant of a shared matrix. The first uses recent theoretical results on Newton's polynomials while the second adapts an algorithm by Samuelson and Berkowitz. From these algorithms, we can deduce two algorithms to solve the corresponding linear system. The last algorithm revisits an existing state-of-the-art algorithm by adding noise to the revealed matrix rank. We show that the resulting leakage will be hard to exploit.

These two algorithms enable new threshold instantiations of UOV and UOV-based schemes, in particular MAYO.

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 [BCD+24] and the most efficient MAYO [BCC+21].

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. 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 [dPN25,dPKN$^+$25,BCdP$^+$25,dPENP25].

Given that UOV and its variants are serious candidates in the standardization process and with the announcement of the futur NIST call, designing a threshold scheme based on UOV or MAYO is therefore an important research question.

Related work. In the state of the art, there are two attempts to create a treshold scheme based on UOV or MAYO. In [CS19], the authors examine all the signature submissions to NIST Round 2 and conclude that UOV-based schemes are the simplest to treshold, 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 treshold use are Rainbow and LUOV, two UOV-based schemes. In addition, they build a "tailor-made" way of solving a shared linear system. It is important to mention that LUOV and Rainbow were broken during standardization in [DDVY21] and [Beu22] respectively. In [CEN25], the authors explore the tresholdization of UOV-based algorithms, focusing on UOV and MAYO. They show that the main algorithm proposed in [CS19] 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 $A\mathbf{x} = \mathbf{y}$ 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 [CS19] 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".

In conclusion, building a threshold signature protocol based on UOV does not appear trivial. 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.

Our Contributions. By carefully studying the state of the art, we may note that [MV24] proposes to calculate 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. We have three main contributions:

- First we propose a complete generalization of this algorithm to the case of low characteristic, 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. We obtain a trade-off between the number of communications and the number of communication rounds. The resulting complexity is in $O(n^3)$ communication and $O(n^2)$ round of communication.
- Finally, if the two algorithms above are too costly to implement, we propose an algorithm based on the state-of-the-art, which reveals either the rank of matrix A or the rank of a random matrix that is never invertible.

2 Preliminaries

This section recalls the definitions and results already known, which will be useful throughout the rest of the paper. We recall the notations used in UOV and briefly explain how UOV and MAYO works, then we detail existing algorithms to solve linear systems obliviously.

2.1 Proportion of non-singular matrices

In the rest of the paper we will have to estimate the proportion of non-singular matrices within a finite field of size $q > 2$. We can determine this distribution by using a classical result from [Ran93]:

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

2.2 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 $\mathcal{P}(\mathbf{x}) = (p_1(\mathbf{x}),\dots,p_m(\mathbf{x}))$ is given by:

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

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 calculate 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 n equations and n 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 $\mathbf{y} \in \mathbb{F}_q^m$ 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 calculate 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 n equations and n variables over $\mathbb{F}_q$. If it has no solution, 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 level. 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.

2.3 Existing threshold work on UOV

Notation: As in [CEN25], we will use the Arithmetic Black Box (ABB) model to modelize 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 ([MV24]) *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 ([MV24]) *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.*

In the literature, two papers study the thresholdization of UOV-based signature algorithms: [CS19] and [CEN25]. Both papers propose customized methods that are more efficient than general multiparty computation one. In both [CS19] and [CEN25], the functionality at the core of the protocols is the resolution of a threshold linear system. Formally, the authors attempt to solve the following problem:

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 $[\![\mathbf{x}]\!]$ a shared representation of $\mathbf{x}$ such that:*

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

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

Although explained in [CS19] and [CEN25], we propose to recall, briefly, why this problem appears naturally in any treshold 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 $\mathbf{o}_i \in \mathcal{O}_i$ and 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 calculated, 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 [Pé24]. Since $\mathbf{y}$ is known and $\mathcal{P}$ is a polynomial, it is equivalent to 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 treshold linear system without revealing any information, exactly as formulated in Problem 1.

Solution proposed by [CS19]: In [CS19], the authors propose Algorithm 1, described below.

Algorithm 1 `Solving` $A\mathtt{x} = \mathtt{b}$

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

1: Generate a random $n \times n$ 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 3 *For any $A \in \mathcal{M}_n(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^n$, Algorithm 1 calculates $[\![\mathbf{x}]\!]$, a shared representation of $\mathbf{x}$ that verifies $A\mathbf{x} = \mathbf{b}$. Algorithm 1 requires $O(n^2)$ communications and $O(1)$ rounds of communications. Besides, Algorithm 1 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 n^2 communications, while step 6 requires n. 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 [CEN25] In [CEN25], the authors propose Algorithm 2, described below.

Algorithm 2 `Solving` $A\mathbf{x} = \mathtt{b}$

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 4 *For any $A \in \mathcal{M}_{s\times t}(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^s$, Algorithm 2 calculates $[\![\mathbf{x}]\!]$, a shared representation of $\mathbf{x}$ that verifies $A\mathbf{x} = \mathbf{b}$. Algorithm 2 requires $O(st)$ communication and $O(1)$ round of communications. If A is not invertible, Algorithm 2 reveals the image and kernel of $T = RAS$. In particular, if 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:

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

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. □

3 Shared linear resolution

In the previous section, it has been described two algorithms that solve the Problem 1 in a reasonable number of steps. However, the two algorithms reveal some information. The first one gives the image of A and the second one the rank of

A, this section investigate a solution that only reveal a minimum amount of information. If the characteristic of the field is greater than the size of the matrix, [PS78] used the Faddeev-Leverier 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 UOV and its variants, because the algorithms are mainly defined on $\mathbb{F}_2$ and its extensions, which are all fields of characteristic 2. For a general field of low characteristic, this result was generalized in [Sch93,MV24], by working in an extension of $\mathbb{F}_q$. However, in the annexe of [MV24], 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 [dV25], proposed a generalization of the relations between Newton's polynomials. This section uses the theoretical results demonstrated in [dV25] to create an easier algorithm that allows computation of the determinant with complexity $O(n^{3.5})$, regardless of the characteristic of the field.

3.1 Existing works

First, we will 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 [MV24]. 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 the following proposition due to Leverrier.

Proposition 5 (Leverier) *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 5 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 a inner product. This idea will be combined with a baby-step giant-step approach: precompute $A^2, A^3, \ldots, A^m$ and $A^{2m}, A^{3m}, \ldots, A^{m^2}$ with Proposition 1 for $m = \lfloor \sqrt{n} \rfloor$, 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 1. One of the advantages of the previous protocol is that it is simple to implement. The objective of the next section is to build a simple protocol for calculating the determinant in small characteristics, in the same way of Leverier's result.

3.2 Generalization in low characteristic

In this subsection $\mathbb{F}_q$ will be a field of characteristic r_0 and $n > r_0$. 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:

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

We will use [dV25], where an algorithm is presented for calculating elementary symmetric polynomials solely in matrix form, thereby generalizing Leverier's idea, all technical results are proved in [dV25]. 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 submatrix 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 submatrix by $R(x_1,\ldots,x_n)$. The following proposition will give the generalisation of the Leverier lemma in matrix forms:

Lemma 1 ([dV25]). *The Newton's polynomials satisfies:*

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

and

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

where

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

Thanks to the expression of the previous determinant, we have the following Proposition 6:

Proposition 6 *Let $A\in M_n(\mathbb{F}_q)$ with eigenvalues $\lambda_A=(\lambda_1,\ldots,\lambda_n)\in\overline{\mathbb{F}_q}^n$. 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 $\mathbb{F}_q$. 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 2. To compute $\det(A)$, a naive approach consists of using Proposition 6, but as soon as an eigenvalue is repeated with multiplication at most r_0, the condition won't 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 [Sch93] 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 7 *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 6 on B, by Proposition 7, we need to choose C which verify that $\det(P_B)(0) = \det(P_C) \neq 0$. This can be done with Lemma 2:

Lemma 2. *Let Q an monic irreductible polynome 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 eighenvalues 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 3, which allows secure computation of the determinant.

Algorithm 3 `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 8 *For any $A \in \mathcal{M}_n(\mathbb{F}_q)$, Algorithm 3 calculates $[\![\det(A)]\!]$, a shared representation of $\det(A)$. Algorithm 3 requires $O(1)$ rounds of communication and $O(n^{3.5})$ communications.*

Proof. The correction and termination of the algorithm follows from Proposition 6 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. 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 better complexity for field of low characteristic

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

Theorem 1 ([Sol02]). *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 the Algorithm 4 which allows secure computation of the determinant with a better multiplication complexity than Algorithm 3.

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

Proposition 9 *For any* $A \in \mathcal{M}_n(\mathbb{F}_q)$*, Algorithm 4 calculates* $[\![\det(A)]\!]$*, a shared representation of* $\det(A)$*. Algorithm 4 requires* $O(n^3)$ *communications and* $O(n^2)$ *rounds of communication.*

Proof. The correction and termination of Algorithm 4 follows from Theorem 1. Step 4 requires the multiplication of one matrix of size $(n-i) \times (n-i-1)$ with a vector of size $(n-i-1) \times 1$ that costs $O(n)$ communications and $O(1)$ round of communication. Step 5 requires the multiplication of one vector of size $(n-i-1) \times 1$ with one vector of size $1 \times (n-i-1)$ that costs $O(1)$ communication and $O(1)$ round of communication. The loops require repeating step 4 and 5 $O(n^2)$ times. Step 6 does not require multiplications. Step 7 requires n matrix multiplications that each requires $O(n^2)$ communications. Finally, Algorithm 4 requires a total of $O(n^3)$ communications and $O(n^2)$ round of communication. □

Algorithm 5 `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 3 or 4. 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 10 *For any $A \in \mathcal{M}_{s \times t}(\mathbb{F}_q)$ of rank s and $\mathbf{b} \in \mathbb{F}_q^s$, Algorithm 5 calculates $[\![\mathbf{x}]\!]$, a shared representation of $\mathbf{x}$ that verifies $A\mathbf{x} = \mathbf{b}$ without revealing information. Depending on which algorithm is used at step 2, Algorithm 5 requires $O(s^{3.5})$ communication and $O(1)$ round of communications or $O(s^3)$ communications and $O(s)$ round of communication.*

Proof. The proof follows from Proposition 8, Proposition 9, and Proposition 4.

3.4 Noise on the rank leakage

In some applications, the method described in the previous section may seem too complicated to be used. This section therefore describes another method, similar to those described in [CS19] and [CEN25]. In summary, 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 [CEN25] and we believe that it drastically increases

the complexity of any attack, under reasonable assumptions made explicit in the rest of this subsection. Algorithm 6 describes a way for players to choose a matrix at random. This algorithm will be used in Algorithm 7 to solve a linear system.

Algorithm 6 $\texttt{Randomselection}_{1/2}$

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 3. 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 player i 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 7 $\texttt{Solving } A\mathbf{x} = \mathbf{b}$

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 11 *For any $A \in \mathcal{M}_{s\times t}(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^s$, Algorithm 7 calculates $[\![\mathbf{x}]\!]$, a shared representation of $\mathbf{x}$ that verifies $A\mathbf{x} = \mathbf{b}$. With probability $1/2$, Algorithm 7 reveals the rank of A or the rank of a random non-invertible matrix T'. It requires eight rounds of communication and $4n^3 + 3n^2$ secure multiplications.*

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

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

Eight rounds of communication are required: five for the products on the shares in step 2, step 4, step 8, step 11 and one to open value T'' in step 5, while the call to $\texttt{Randomselection}_{1/2}$ requires two rounds. Finally, step 2 and step 8 requires n^3 secure multiplications. step 4 and step11 respectively requires $(n-1)n^2$ and $2n^2$ multiplications. While step 8 requires n^3 multiplications to open T'' and $\texttt{Randomselection}_{1/2}$ requires $2n^2$. This results in a total of $4n^3 + 3n^2$ secure multiplications.

□

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. 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.

3.5 Security guarantee

Algorithm 2 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 [CEN25], more precisely:

- The authors describe the Arithmetic Black Box (ABB) model and the associated $\mathcal{F}_{\mathbf{ABB}}$ functionality. They then model Algorithm 2 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 7 is at least as secure as the one described in [CEN25].

Theorem 1 *Let* $\mathbf{Leak}$ *be a sequence of matrices resulting from* t *executions of* $\mathcal{F}_{\mathbf{ABB}-\boldsymbol{solve}}$ *and* $\overline{\mathbf{Leak}}$ *be a sequence of matrices resulting from* t *executions of* $\overline{\mathcal{F}}_{\mathbf{ABB}-\boldsymbol{solve}}$*, the functionality where Algorithm 2 has been replaced by Algorithm 7. Let us assume that there is no distinction between a* $\overline{\mathbf{Leak}}$ *element that comes from* T' *or* T'' *in Algorithm 7. 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 7. 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 4. Theorem 1 allow to quantify the proportion of noise in the set of $\overline{\mathbf{Leak}}$ matrices, given by several executions of Algorithm 7. 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_5	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 7. From Table 2 we understand that we have a maximum of 19% significant of information out of $\overline{\mathbf{Leak}}$ with $q = 16$ while with $q = 256$ we only have 1.17% of $\overline{\mathbf{Leak}}$ that is significant. Furthermore in our computation we ignored the noise that is coming from the fact that $T := RAS$ can be non invertible, even when A is invertible. On the negative side, it is important to note that Algorithm 7 will be twice as slow, since it will reject matrices that could have been used.

4 Conclusion and discussion

In this paper we showed two different method to limit leakage during the resolution of a shared linear system. As demonstrated by [CEN25], this construction is one of the core operations that enable the creation of any threshold signature

protocol based on UOV. We effectively got rid of any leakage with our secure determinant computation which means that theoretically we are able to share a UOV signature without leaking anything else than the fact that the matrix we need to invert is not invertible. However, as we evaluated this method is very costly and is unlikely to be used in a applied setting. Therefore, we created another method that limits the rank leakage by adding noise in the protocol. Although this method still leaks a small amount of information, we have endeavoured to provide the most accurate assessment possible of the difficulty of exploiting such a leak. Furthermore, our method is very efficient from a computation perspective.

If, for security reasons, a designer wishes to use an algorithm that does not reveal the rank of the system that is not invertible, it is important to mention that the algorithm presented in this paper is simpler to implement than the one presented in [MV24]. However, it requires that the characteristic of the field be strictly greater than the number of players, whereas our algorithm works even in $\mathbb{F}_2$ and it's extensions. This result lead to the idea that other UOV variants may be more appropriate to construct threshold protocols. We can think about the NIST candidate QR-UOV [FIH+24] that is using large odd characteristic. Even though it is still possible to build protocols based on classic UOV or MAYO, using our algorithm.

The second algorithm works regardless of the field's characteristics, but the added noise will be higher in larger characteristics. In the end, we can hope that our result will bring some new insight upon this understudied subject.

References

BCC+21. Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. Algorithm specifications and supporting documentation, 2021. `https://pqmayo.org/`.

BCD+24. Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Pacques Jatarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. Algorithm specifications and supporting documentation, 2024. `https://www.uovsig.org/`.

BCdP+25. Giacomo Borin, Sofía Celi, Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Threshold signatures reloaded: ML-DSA and enhanced raccoon with identifiable aborts. Cryptology ePrint Archive, Paper 2025/1166, 2025.

Beu22. Ward Beullens. Breaking rainbow takes a weekend on a laptop. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 464–479, Cham, 2022. Springer Nature Switzerland.

CEN25. Sofia Celi, Daniel Escudero, and Guilhem Niot. Share the mayo: Thresholdizing mayo. In Ruben Niederhagen and Markku-Juhani O. Saarinen, editors, *Post-Quantum Cryptography*, pages 165–198, Cham, 2025. Springer Nature Switzerland.

CS19. Daniele Cozzo and Nigel P. Smart. Sharing the luov: Threshold post-quantum signatures. In Martin Albrecht, editor, *Cryptography and Coding*, pages 128–153, Cham, 2019. Springer International Publishing.

DDVY21. Jintai Ding, Joshua Deaton, Vishakha, and Bo-Yin Yang. The nested subset differential attack. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 329–347, Cham, 2021. Springer International Publishing.

dPENP25. Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Simple and efficient lattice threshold signatures with identifiable aborts. Cryptology ePrint Archive, Paper 2025/871, 2025.

dPKN+25. Rafael del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. Unmasking traccoon: A lattice-based threshold signature with an efficient identifiable abort protocol. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025*, pages 423–456, Cham, 2025. Springer Nature Switzerland.

dPN25. Rafael del Pino and Guilhem Niot. Finally! a compact lattice-based threshold signature. In Tibor Jager and Jiaxin Pan, editors, *Public-Key Cryptography – PKC 2025*, pages 169–199, Cham, 2025. Springer Nature Switzerland.

dV25. Sjoerd de Vries. On newton's identities in positive characteristic. *Journal of Algebra*, 668:348–364, April 2025.

FIH+24. Hiroki Furue, Yasuhiko Ikematsu, Fumitaka Hoshino, Tsuyoshi Takagi, Kan Yasuda, Toshiyuki Miyazawa, Tsunekazu Saito, and Akira Nagai. QR-UOV. Technical report, National Institute of Standards and Technology, 2024. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

MV24. Jules Maire and Damien Vergnaud. Secure multi-party linear algebra with perfect correctness. Cryptology ePrint Archive, Paper 2024/508, 2024.

PS78. F.P. Preparata and D.V. Sarwate. An improved parallel processor bound in fast matrix inversion. *Information Processing Letters*, 7(3):148–150, 1978.

Pé24. Pierre Pébereau. 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*, page 92–108, Berlin, Heidelberg, 2024. Springer-Verlag.

Ran93. Dana Randall. Efficient generation of random nonsingular matrices. *Random Structures & Algorithms*, 4, 01 1993.

Sch93. Arnold Schönhage. Fast parallel computation of characteristic polynomials by leverrier's power sum method adapted to fields of finite characteristic. In Andrzej Lingas, Rolf Karlsson, and Svante Carlsson, editors, *Automata, Languages and Programming*, pages 410–417, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

Sol02. Michael Soltys. Berkowitz's algorithm and clow sequences, 2002.