Research Article

Roberto La Scala* and Alessio Meneghetti

Inner product functional encryption based on the UOV scheme

https://doi.org/10.1515/jmc-2024-0026
Received June 20, 2024; accepted September 12, 2025; published online November 21, 2025

Abstract: We analyze the efficiency and security of the inner product functional encryption (IPFE) protocol introduced in 2021 by Debnath, Mesnager, Dey, and Kundu, specifically when instantiated with UOV. While the scheme offers several advantages, including improvements in key generation and encryption/decryption algorithms, along with compact key sizes, the decryption algorithm remains exponential in complexity with respect to the security parameter. To address this limitation, we propose a variant aimed at reducing the decryption cost. However, this alternative remains impractical at present due to the resulting large ciphertext size.

Keywords: multivariate cryptography; functional encryption; finite fields

MSC 2020: 94A60; 11T06; 11T71

1 Introduction

In recent years, with the progressive improvement in the speed and reliability of data networks, we have witnessed a growing diffusion of "cloud computing". This practice involves entrusting storage functions, software applications, and computation to powerful remote servers capable of meeting the needs of millions or even billions of users. In this context, ensuring the confidentiality of individual user data is of fundamental importance, making the use of specific cryptographic tools indispensable. To allow providers to deliver services through data processing while preserving confidentiality, the paradigms that are emerging as particularly promising are functional encryption [1, 2] and homomorphic encryption [3], briefly FE and HE. These approaches pursue distinct objectives: in HE, the value of functions applied to plaintext data is calculated as a function of the encrypted data and returned to the owner in encrypted form. Only the data owner has the capability to use it, unless they decide to share it with the provider through an additional encryption system. Conversely, in FE, by using specific "functional keys" the provider can directly compute the value of the required functions on plaintext data starting from encrypted data, which are never fully decrypted. These features allow the provider to deliver services to users without requiring further actions from them, except for distributing the necessary functional keys.

***Corresponding author: Roberto La Scala**, Dipartimento di Matematica, Università degli Studi di Bari "Aldo Moro", Via Orabona 4, 70125, Bari, Italy, E-mail: roberto.lascala@uniba.it
Alessio Meneghetti, Dipartimento di Matematica, Università degli Studi di Bari "Aldo Moro", Via Orabona 4, 70125, Bari, Italy, E-mail: alessio.meneghetti@uniba.it

To illustrate the paradigm of functional encryption, we provide a couple of examples of its applications. Consider a hospital that records the medical data of its patients. For research purposes, it could be useful to perform data analysis on these records. Using FE, the hospital can delegate the storage of the records to a cloud service without compromising their confidentiality because the data are encrypted before being sent to the server. At the same time, the hospital can distribute functional keys to researchers, enabling them to conduct medical statistical analyses on the records stored in the cloud for purposes such as evaluating a therapy, without accessing the actual content of the records.

Another common application of FE is performing machine learning on encrypted data while ensuring confidentiality. Specifically, after training a classifier on standard data, the data owner can generate specific functional keys for the functions required by the classifier. In other words, the classifier can perform classification on encrypted data without knowing its plaintext content.

If such computations require knowledge of linear functions of the plaintext data, we refer to the FE scheme as an inner product functional encryption, briefly IPFE, protocol. Some examples of inner products widely used in data analysis are the expected value and the convolution product. Recently, in [4], the authors introduced an IPFE protocol based on multivariate cryptography. We recall that this type of post-quantum cryptographic primitives achieves security through the challenge of computing preimages of generic quadratic polynomial maps $F\colon \mathbb{F}^n \to \mathbb{F}^m$ where $\mathbb{F}$ is a finite field. These primitives are typically digital signatures where signing a vector $v \in \mathbb{F}^m$ corresponds to compute a preimage $u \in F^{-1}(v) \subset \mathbb{F}^n$. The signer holds a secret that allows for efficient computation of such a preimage.

In the present paper, we discuss the possibility of designing multivariate IPFE schemes. We start by discussing the efficiency and security of the IPFE protocol in [4] modified by leveraging the UOV digital signature [5–8]. In this version of the protocol, the property that the quadratic map $F\colon \mathbb{F}^n \to \mathbb{F}^m$ is an "oil-vinegar map" corresponds to the existence of a secret vector subspace $O \subset F^{-1}(0)$. Using this subspace one can obtain elements in the preimages of F without the need for any linear change of coordinates. If otherwise the subspace O is unknown, an opponent faces a problem currently considered indistinguishable from the NP-hard problem of solving a quadratic polynomial system over a finite field. For a recent paper on solving polynomial systems over finite fields, see for instance [9].

Our analysis of the IPFE scheme reveals that the protocol's decryption algorithm incurs a computational cost exponential in the scheme's parameters. To address this inefficiency, we propose a variant aimed at mitigating the issue. However, it appears that circumventing the exponential decryption cost necessitates an increase in ciphertext size, which once again leads to inefficiency. The challenge of developing a practical multivariate IPFE scheme remains an open problem and requires further investigation.

2 Oil-vinegar maps

We start presenting the modern concept of an oil-vinegar map and its corresponding algorithms for the UOV digital signature. References include [5–7].

Let $\mathbb{F}$ be a finite field and denote by $S = \mathbb{F}[x_1, \dots, x_n]$ the algebra of the polynomials with coefficients in the field $\mathbb{F}$ and variables $x_1, \dots, x_n$. For all $1 \le k \le m$, let $f_k \in S$ be a quadratic form, that is, a homogeneous polynomial of degree 2. Let $F = (f_1, \dots, f_m) \in S^m$. By abuse of notation, we also denote by $F\colon \mathbb{F}^n \to \mathbb{F}^m$ the quadratic map such that, for all $v = (v_1, \dots, v_n) \in \mathbb{F}^n$

$$F(v) = (f_1(v), \dots, f_m(v)).$$

Note that each quadratic form f_k $(1 \le k \le m)$ corresponds to a matrix $A_k = \left(A_k^{ij}\right) \in \mathrm{M}_n(\mathbb{F})$ such that, if $x = (x_1, \dots, x_n) \in S^n$ then

$$f_k(x) = xA_kx^T \in S.$$

To avoid issues arising from the characteristic of the field F, we make the assumption that all the matrices A_k are upper triangular, that is, $A_k^{ij} = 0$ whenever $i > j$. We have hence that

$$f_k = \sum_{i \le j} A_k^{ij} x_i x_j.$$

Consider another variable set $\{y_1, \dots, y_n\}$ which is disjoint from the set $\{x_1, \dots, x_n\}$ and let $\bar{S} = \mathbb{F}[x_1, \dots, x_n, y_1, \dots, y_n]$. The polar form of the quadratic form f_k is by definition the bilinear form

$$\bar{f}_k(x, y) = f_k(x + y) - f_k(x) - f_k(y) \in \bar{S}.$$

In matrix terms, we have that

$$\bar{f}_k(x, y) = xA_k y^T + yA_k x^T = x\left(A_k + A_k^T\right) y^T$$

that is

$$\bar{f}_k = \sum_{i,j} \left(A_k^{ij} + A_k^{ji} \right) x_i y_j.$$

Note that $A_k + A_k^T$ is a symmetric matrix with elements along the main diagonal that are divisible by 2. Hence, the monomials $x_i y_i$ $(1 \le i \le n)$ will not appear in the polar forms $\bar{f}_k$ whenever $\mathrm{char}(\mathbb{F}) = 2$. We call $\bar{F} = (\bar{f}_1, \dots, \bar{f}_m) \in \bar{S}^m$ the *polar map* of the quadratic map $F = (f_1, \dots, f_m) \in S$. By abuse of notation, we denote by $\bar{F} \colon \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}^m$ the bilinear map such that, for all $u, v \in \mathbb{F}^n$

$$\bar{F}(u, v) = (\bar{f}_1(u, v), \dots, \bar{f}_m(u, v)).$$

Let $\mathbb{F} = \mathrm{GF}(q)$ be the finite field with q elements and consider the ideal $E = \langle x_1^q - x_1, \dots, x_n^q - x_n \rangle \subset S$. If $J \subset S$ is an ideal and $\bar{\mathbb{F}}$ is the algebraic closure of the field $\mathbb{F}$, we denote

$$V(J) = \{ v \in \bar{\mathbb{F}}^n \mid f(v) = 0 \text{ for all } f \in J \}$$

and $V_{\mathbb{F}}(J) = V(J) \cap \mathbb{F}^n$. The Nullstellensatz over finite fields [10] implies the following result.

Proposition 2.1. *Let $J \subset S$ be an ideal. We have $V(E) = \mathbb{F}^n$ and $V_{\mathbb{F}}(J) = V(J + E)$ where $J + E$ is a radical ideal of S.*

Definition 2.2. Let $F = (f_1, \dots, f_m)$ be a quadratic map and consider the corresponding ideal $I_F = \langle f_1, \dots, f_m \rangle$. We call F an *oil-vinegar map* if there exists a vector subspace $0 \neq O \subset \mathbb{F}^n$ such that $O \subset V_{\mathbb{F}}(I_F) = F^{-1}(0)$. We call O the *oil subspace* and $r = \dim_{\mathbb{F}} O$ the *oil dimension* of F.

Let $l_1, \dots, l_{n-r} \in S$ be linear forms and put $L = (l_1, \dots, l_{n-r}) \in S^{n-r}$. By abuse of notation, we also denote $L \colon \mathbb{F}^n \to \mathbb{F}^{n-r}$ the corresponding linear map. If $O = V_{\mathbb{F}}(I_L)$ where $I_L = \langle l_1, \dots, l_{n-r} \rangle$, then the inclusion $O \subset V_{\mathbb{F}}(I_F)$ is equivalent to the inclusion

$$I_F \subset I_L + E$$

due to the fact that $I_F + E, I_L + E$ are radical ideals. By assuming that the linear forms l_k are linearly independent we have that $r = \dim_{\mathbb{F}} O$.

Observe that the condition $I_F \subset I_L + E$ is equivalent to require that each quadratic form f_i $(1 \le i \le m)$ can be written in the form

$$f_i = \sum_{1 \le j \le n-r} g_{ij} l_j \bmod E$$

where $g_{ij} \in S$ are linear forms.

3 Preimages of oil–vinegar maps

The computation of a preimage of a quadratic map is generally a difficult task because it corresponds to solve a quadratic polynomial system over a finite field. Indeed, it is well-know that the solution to such a general

problem is NP-hard [11]. We show now that computing a preimage of an oil-vinegar map becomes efficient once the oil-subspace is known.

Let $w = (w_1, \dots, w_m) \in \mathbb{F}^m$. If $F = (f_1, \dots, f_m)$ is a quadratic map, computing $v = (v_1, \dots, v_n) \in \mathbb{F}^n$ such that $F(v) = w$ is equivalent to solve, over the base field $\mathbb{F}$, the following system of quadratic equations

$$\begin{cases} f_1(x) = w_1, \\ \quad \vdots \\ f_m(x) = w_m. \end{cases}$$

In other words, the vector v is a preimage of w under the map F. We denote this by writing $v \in F^{-1}(w)$. Assume now that F is an oil–vinegar map and we have knowledge of its oil subspace $O \subset V_{\mathbb{F}}(I_F) = F^{-1}(0)$. In particular, let $O = V_{\mathbb{F}}(I_L) = L^{-1}(0)$ where $L = (l_1, \dots, l_{n-r})$ and $l_k \in S$ are linear independent linear forms. Fix a random vector $u \in \mathbb{F}^n$. If $o \in O$ and hence $F(o) = 0$, then

$$w = F(u + o) = F(u) + F(o) + \bar{F}(u, o) = F(u) + \bar{F}(u, o).$$

To compute a preimage $v = u + o \in F^{-1}(w)$ it is sufficient therefore to solve the system of linear equations $L(x) = 0, \bar{F}(u, x) = w - F(u)$ which is explicitly

$$\begin{cases} l_1(x) = 0, \\ \quad \vdots \\ l_{n-r}(x) = 0, \\ \bar{f}_1(u, x) = w_1 - f_1(u), \\ \quad \vdots \\ \bar{f}_m(u, x) = w_m - f_m(u). \end{cases}$$

Note that if $m = r$ we have a system of n linear equations in exactly n variables. In this case, the matrix of its coefficients has no maximal rank (determinant is zero) with probability $1/q$. If we do not obtain maximal rank, it is sufficient to choose a new random vector $u \in \mathbb{F}^n$. Moreover, by assuming that the equations $f_1(x) = w_1, \dots, f_m(x) = w_m$ are sufficiently generic (complete intersection), one has that the preimage $F^{-1}(w) \subset \mathbb{F}^n$ is an affine variety of dimension $n - m$.

The signing algorithm of the UOV protocol corresponding to the oil-vinegar map $F: \mathbb{F}^n \to \mathbb{F}^m$ is, by definition, the computation of a preimage $v \in F^{-1}(w) \subset \mathbb{F}^n$ for a document $w \in \mathbb{F}^m$. More precisely, the vector w is generally the hash value of a document, so the parameter m is fixed and not very large. The cryptanalysis of the UOV protocol [12] implies that the condition $n > 2m$ is necessary to achieve a secure signature. As we have just seen, an efficient signing algorithm is made possible by the knowledge of the oil subspace $O \subset F^{-1}(0)$. The verification of the signature v simply consists in checking that $F(v) = w$ which involves the cost of evaluating the quadratic map F. Therefore, the public key of UOV is the map F and the private key is the pair (F, O).

4 Generation of oil–vinegar maps

We address now the task of the key generation in the UOV protocol, that is, how to construct an oil–vinegar map $F = (f_1, \dots, f_m)$ from any subspace $O \subset \mathbb{F}^n$ of dimension r. Up to permutating the coordinates of the vector space $\mathbb{F}^n$, we can assume that O is the subspace generated by the rows of a block matrix $(H\ I) \in \mathrm{M}_{r\times n}(\mathbb{F})$ where $H \in \mathrm{M}_{r\times n-r}(\mathbb{F})$ is any matrix and $I \in \mathrm{GL}_r(\mathbb{F})$ is the identity matrix. Consider the upper triangular matrices $A_k \in \mathrm{M}_n(\mathbb{F})$ corresponding to each quadratic form f_k $(1 \le k \le m)$, that is

$$f_k(x) = xA_kx^T.$$

We consider A_k as a block upper triangular matrix of type

$$A_k = \begin{pmatrix} A'_k & A''_k \\ 0 & A'''_k \end{pmatrix}$$

where $A'_k \in \mathrm{M}_{n-r}(\mathbb{F}), A''_k \in \mathrm{M}_{n-r\times r}(\mathbb{F}), A'''_k \in \mathrm{M}_r(\mathbb{F})$ and A'_k, A'''_k are upper triangular matrices. To enforce $O \subset F^{-1}(0)$, or equivalently $F(O) = \{0\}$, corresponds to impose, for each $1 \leq k \leq m$, the following matrix equations

$$(H\ I)A_k(H\ I)^T = 0.$$

From the above equations, one obtains that

$$HA'_kH^T + HA''_k + A'''_k = 0$$

and hence

$$A'''_k = -HA''_k - HA'_kH^T.$$

In other words, by arbitrarily assigning the matrix H that defines the oil subspace $O \subset \mathbb{F}^n$ and arbitrarily choosing the matrices A'_k, A''_k $(1 \leq k \leq m)$, it is always possible to define the matrices A'''_k $(1 \leq k \leq m)$ in such a way that the quadratic map $F = (f_1, \dots, f_m)$ corresponding to the block upper triangular matrices $A_1, \dots, A_m \in \mathrm{M}_n(\mathbb{F})$ satisfies $O \subset F^{-1}(0)$. It is worth noting that the random matrix $A'_k \in \mathrm{M}_{n-r}(\mathbb{F})$ can be directly generated as an upper triangular matrix, whereas the matrix $A'''_k \in \mathrm{M}_r(\mathbb{F})$ is defined as the upper triangular matrix that yields the same quadratic form as the matrix $-HA''_k - HA'_kH^T$.

5 Efficient secret key and signature for UOV

In order to reduce the key sizes and make the signing process more efficient in UOV, we observe the following. Assume as before that the oil subspace $O \subset \mathbb{F}^n$ is the subspace generated by the rows of a block matrix $(H\ I) \in \mathrm{M}_{r\times n}(\mathbb{F})$ where I is the identity matrix of order r. Under this assumption, note that the last r entries of a vector $o' \in O$ can be arbitrarily chosen.

Let $w \in \mathbb{F}^m$ and $u \in \mathbb{F}^n$. We have shown that the knowledge of the oil subspace O implies that a vector $u + o$ $(o \in O)$ such that $F(u + o) = w$ can be computed by solving a system of linear equations. Let now $o' \in O$. Note that if $o \in O$ is such that $L(o) = 0, F(u + o) = w$, then $o - o' \in O$ clearly satisfies $L(o - o') = 0, F(u + o' + (o - o')) = w$. In other words, the computation of a preimage in $F^{-1}(w)$ can be obtained in the same way if we replace the arbitrary vector $u \in \mathbb{F}^n$ with a vector of the form $u + o'$ where $o' \in O$.

Due to the assumption that a basis of the oil subspace O is provided by the rows of a block matrix of type $(H\ I)$, replacing u with $u + o'$ offers the advantage that one can require that the last r entries of the vector o' are precisely the opposite of the corresponding entries of the vector u. In other words, we can assume that the vector $u + o'$ has its last r coordinates all equal to zero.

Let $u = (u'\ 0) \in \mathbb{F}^n$ with $u' \in \mathbb{F}^{n-r}$. Recall that $f_k(x) = xA_kx^T$ $(1 \leq k \leq m)$ where

$$A_k = \begin{pmatrix} A'_k & A''_k \\ 0 & A'''_k \end{pmatrix}$$

with $A'_k \in \mathrm{M}_{n-r}(\mathbb{F}), A''_k \in \mathrm{M}_{n-r\times r}(\mathbb{F}), A'''_k \in \mathrm{M}_r(\mathbb{F})$. We have therefore that

$$f_k(u) = u'A'_ku'^T.$$

Moreover, recall that $\bar{f}_k(x, y) = x\left(A_k + A_k^T\right)y^T$. Since we are assuming that O admits as a basis the row vectors of a block matrix of the form $(H\ I)$, any vector $o \in O$ can be obtained as $o = c(H\ I) = (cH\ c)$ where $c = (c_1, \dots, c_r) \in \mathbb{F}^r$. By computing $\bar{f}_k(u, o)$ one obtains hence

$$\bar{f}_k(u,o) = (u'\ 0)\begin{pmatrix} A'_k + A'^T_k & A''_k \\ A''^T_k & A'''_k + A'''^T_k \end{pmatrix}(cH\ c)^T$$
$$= u'((A'_k + A'^T_k)H^T + A''_k)c^T.$$

By putting

$$C_k = (A'_k + A'^T_k)H^T + A''_k \in \mathrm{M}_{n-r\times r}(\mathbb{F})$$

we finally obtain that

$$\bar{f}_k(u,o) = u'C_k c^T.$$

Once a vector $u = (u'\ 0) \in \mathbb{F}^n$ is fixed, we recall that the UOV signature of $w = (w_1, \dots, w_m) \in \mathbb{F}^m$ consists in solving the following system of linear equations

$$\begin{cases} \bar{f}_1(u,o) = w_1 - f_1(u), \\ \vdots \\ \bar{f}_m(u,o) = w_m - f_m(u) \end{cases}$$

where the vector $o \in O$ is unknown. Since such a vector is parametrized by a vector $c \in \mathbb{F}^r$, once the matrices $(A'_1, C_1, \dots, A'_m, C_m)$ along with the matrix H are stored as the secret key, the signing process involves solving the corresponding linear system, namely

$$\begin{cases} u'C_1c^T = w_1 - u'A'_1u'^T, \\ \vdots \\ u'C_mc^T = w_m - u'A'_mu'^T \end{cases}$$

where the vector $c \in \mathbb{F}^r$ is the unknown. After computing c, we put $o = c(H\ I)$ and the signature is defined as the preimage $v = u + o \in F^{-1}(w)$.

6 Formal description of UOV algorithms

Henceforth, we assume that $m = r = \dim_{\mathbb{F}} O$ and $n > 2m$. Note that in order to set and decrease the value of the parameter m, the vector $w \in \mathbb{F}^m$ typically represents a hash value of the message to be signed.

Algorithm 6.1 Key generation for UOV

Input: $n, m > 0$ integers.
Output: the key pair (sk, pk).

choose at random a matrix $H \in \mathrm{M}_{m\times n-m}(\mathbb{F})$;
for $k = 1$ **to** m **do**
  choose at random an upper triangular matrix $A'_k \in \mathrm{M}_{n-m}(\mathbb{F})$;
  choose at random a matrix $A''_k \in \mathrm{M}_{n-m\times m}(\mathbb{F})$;
  $A'''_k :=$ the upper triangular matrix defining the same quadratic form
    of the matrix $-HA''_k - HA'_kH^T \in \mathrm{M}_m(\mathbb{F})$;
  $A_k := \begin{pmatrix} A'_k & A''_k \\ 0 & A'''_k \end{pmatrix}$;
  $C_k := (A'_k + A'^T_k)H^T + A''_k$;
end for;
$sk := (A'_1, C_1, \dots, A'_m, C_m, H)$;
$pk := (A_1, \dots, A_m)$;
return (sk, pk);

Algorithm 6.2 Generation of the UOV signature

Input: the secret key sk and a vector $w \in \mathbb{F}^m$.
Output: the signature $v \in \mathbb{F}^n$ of w.

$(A'_1, C_1, \dots, A'_m, C_m, H) := sk$;
$flag := true$;
while $flag = true$ **do**
  choose at random a vector $u = (u', 0) \in \mathbb{F}^n$ where $u' \in \mathbb{F}^{n-m}$.
  $b := (u'A'_1u'^T, \dots, u'A'_mu'^T)$;
  $A :=$ the matrix with rows $u'C_1, \dots, u'C_m$;
  if $Ac^T = w - b$ has a solution $c \in \mathbb{F}^m$ **then**
    $flag := false$;
  end if;
end while
$o := c(H\ I)$;
$v := u + o$;
return v;

Algorithm 6.3 Verification of the UOV signature

Input: the public key pk and the vectors $v \in \mathbb{F}^n, w \in \mathbb{F}^m$.
Output: $true$ or $false$.

$(A_1, \dots, A_m) := pk$;
$w' := (vA_1v^T, \dots, vA_mv^T)$;
return the boolean value of $w = w'$;

7 Key sizes in the UOV scheme

The public key of UOV is given by an oil–vinegar map $F\colon \mathbb{F}^n \to \mathbb{F}^m$, that is, by m upper triangular matrices $A_1, \dots, A_m \in \mathrm{M}_n(\mathbb{F})$. Since each element of the finite field $\mathbb{F} = \mathrm{GF}(q)$ is represented by $\lceil \log_2(q) \rceil$ bits, it follows that the size of the public key is

$$\mathrm{size}(pk) = m\frac{n(n+1)}{2}\lceil \log_2(q) \rceil.$$

Recall that the matrices A_k $(1 \le k \le m)$ are block matrices of the form

$$A_k = \begin{pmatrix} A'_k & A''_k \\ 0 & A'''_k \end{pmatrix}$$

where A'_k, A''_k are random matrices and A'''_k is determined by these matrices along with the oil subspace $O \subset \mathbb{F}^n$ $(\dim_{\mathbb{F}} O = m)$. By generating the matrices A'_k, A''_k $(1 \le k \le m)$ using a pseudorandom number generator initialized with a public seed, we have that for the public key $pk = (A_1, \dots, A_m)$ we only need to store the upper triangular matrices $A'''_k \in \mathrm{M}_m(\mathbb{F})$ $(1 \le k \le m)$ in addition to the public seed. We conclude that the size of the *compact public key* is

$$\mathrm{size}(pkc) = \frac{m^2(m+1)}{2}\lceil \log_2(q) \rceil + |\mathrm{seed}_{\mathrm{pub}}|.$$

For the secret key, we need to store the matrix $H \in \mathrm{M}_{m \times n-m}(\mathbb{F})$ such that the rows of the block matrix $(H\ I) \in \mathrm{M}_{m \times n}(\mathbb{F})$ are a basis of the oil subspace. In addition to the matrix H, the UOV signing process requires the upper triangular matrices $A'_k \in \mathrm{M}_{n-m}(\mathbb{F})$ alongside the matrices $C_k \in \mathrm{M}_{n-m \times m}(\mathbb{F})$, for all $k = 1, 2, \dots, m$. The

total size of the secret key is hence

$$\begin{aligned}\text{size}(sk) &= \left(m(n-m) + m\frac{(n-m)(n-m+1)}{2} + m^2(n-m)\right)\lceil \log_2(q)\rceil \\ &= \frac{m(n+m+3)(n-m)}{2}\lceil \log_2(q)\rceil.\end{aligned}$$

As with the public key, we can avoid storing the random matrices A'_k $(1 \le k \le m)$ and H by using a pseudorandom number generator initialized with the public seed used for the public key, along with an additional secret seed. We conclude that the size of the *compact secret key* is

$$\text{size}(skc) = m^2(n-m)\lceil \log_2(q)\rceil + |\text{seed}_{\text{pub}}| + |\text{seed}_{\text{sec}}|.$$

Note that by including the computation of the matrices $C_k = \left(A'_k + A'^T_k\right)H^T + A''_k$ in the UOV signing algorithm, then the size of the compact secret key can be further reduced to

$$|\text{seed}_{\text{pub}}| + |\text{seed}_{\text{sec}}|.$$

In this case, however, the signing process would result in less efficiency.

The size of the UOV signatures is clearly $n\lceil \log_2(q)\rceil$. If the vector $w \in \mathbb{F}^m$ for which we compute a preimage $v \in F^{-1}(w)$ is a hash of the message to be signed and the hashing process includes a random number salt, then the size of the signature is precisely

$$\text{size}(\text{sign}) = n\lceil \log_2(q)\rceil + |\text{salt}|.$$

8 Functional encryption

In this section, we briefly recall the formal notion of Functional Encryption. Let X be the set of *plaintexts* and Y be the set of *ciphertexts*. Denote by K the set of *functional keys* and by V the set of *functional values*. Then, a function $F\colon K \times X \to V$ is given. We also introduce a set K_{sec} of the *secret keys*, a set K_{pub} of the *public keys* and finally a set K_{usr} of the *user secret keys*. Then, we have a function keygen: $K \times K_{\text{sec}} \to K_{usr}$, that is, for any functional key $k \in K$ and for each secret key $sk \in K_{\text{sec}}$, we have a user secret key $usk_k = \text{keygen}(k, sk) \in K_{\text{usr}}$. For any public key $pk \in K_{\text{pub}}$, there is a *functional encryption mapping* $\text{enc}_{pk}\colon X \to Y$. For each user secret key usk_k, we have a *functional decryption mapping* $\text{dec}_{\text{usk}_k}\colon Y \to V$ such that if $y = \text{enc}_{pk}(x)$, then $\text{dec}_{\text{usk}_k}(y) = F(k, x)$.

In practice, the secret key sk and the public key pk are owned by a Master user, a central authority, who distributes pk to Alice to enable them to functionally encrypt the plaintext x, that is, to compute $y = \text{enc}_{pk}(x)$. If Bob is a user corresponding to the functional key k, then the Master distributes to Bob the user secret key $usk_k = \text{keygen}(k, sk)$. With this key, Bob can decrypt the functional value $\text{dec}_{\text{usk}_k}(y) = F(k, x)$ of the plaintext belonging to Alice.

In addition to these algorithms, a protocol of functional encryption, briefly FE, involves a setup function such that for each value of appropriate parameters, it returns a pair (sk, pk) chosen arbitrarily from those satisfying the required parameters. It is assumed that all functions of an FE protocol can be computed efficiently.

If $K = X = \mathbb{F}^d$ and $V = \mathbb{F}$ where $\mathbb{F}$ is a finite field, we have an *inner product functional encryption*, briefly IPFE, if $F(k, x) = \langle k, x\rangle = \sum_i k_i x_i \in V$ where $k = (k_1, \ldots, k_d) \in K$ and $x = (x_1, \ldots, x_d) \in X$.

9 An IPFE protocol based on the UOV scheme

In this section we introduce an IPFE protocol that leverages modern UOV algorithms, inspired by the protocol introduced by Debnath, Mesnager, Dey and Kundu in [4] (See Algorithms 9.1, 9.2, 9.3, and 9.4). As previously explained for general IPFE schemes, the set of plaintexts and the set of functional keys coincide with a vector space $\mathbb{F}^d$ over a finite field $\mathbb{F}$. The set of functional values is thus $\mathbb{F}$.

Let $\{t_1, \dots, t_d\}$ and $\{x_1, \dots, x_n\}$ be two disjoint sets of variables and consider the polynomial algebras $R = \mathbb{F}[t_1, \dots, t_d]$ and $S = \mathbb{F}[x_1, \dots, x_n]$. We put

$$P = R[x_1, \dots, x_n] = \mathbb{F}[t_1, \dots, t_d, x_1, \dots, x_n].$$

Moreover, let $\mathbb{K} = \mathbb{F}(t_1, \dots, t_d)$ denote the field of rational functions corresponding to $R = \mathbb{F}[t_1, \dots, t_d]$. We consider m quadratic forms in the variables $x_1, \dots, x_n$ whose coefficients are polynomials in the variables $t_1, \dots, t_d$. Precisely, if we put $t = (t_1, \dots, t_d) \in R^d$ and $x = (x_1, \dots, x_n) \in S^n$, for each $1 \leq k \leq m$ we define

$$f_k^{(t)}(x) = xA(t)_k x^T \in P$$

where $A(t)_k = (A(t)_k^{ij}) \in \mathrm{M}_n(R)$ is an upper triangular matrix. By evaluating the variable vector $t = (t_1, \dots, t_d)$ at a vector $a = (a_1, \dots, a_d) \in \mathbb{F}^d$, we obtain a quadratic form $f_k^{(a)} \in S$ associated with the matrix $A(a)_k \in \mathrm{M}_n(\mathbb{F})$, which in turn is obtained by evaluating the matrix $A(t)_k$. By setting $F^{(a)} = (f_1^{(a)}, \dots, f_m^{(a)})$, we have a quadratic map $F^{(a)}\colon \mathbb{F}^n \to \mathbb{F}^m$ for every vector $a = (a_1, \dots, a_d) \in \mathbb{F}^d$. This set of maps is clearly obtained by evaluating the quadratic map $F^{(t)} = (f_1^{(t)}, \dots, f_m^{(t)})$, where we can assume that $F^{(t)}\colon \mathbb{K}^n \to \mathbb{K}^m$.

Let now $\{y_1, \dots, y_n\}$ be a set of variables disjoint from the sets $\{x_1, \dots, x_n\}$ and $\{t_1, \dots, t_d\}$. We consider the polynomial algebra $\bar{P} = R[x_1, \dots, x_n, y_1, \dots, y_n]$. The polar form of the quadratic map $f_k^{(t)}$ is by definition the bilinear form

$$\bar{f}_k^{(t)}(x, y) = f_k^{(t)}(x + y) - f_k^{(t)}(x) - f_k^{(t)}(y) \in \bar{P}.$$

In other words, we have

$$\bar{f}_k^{(t)}(x, y) = x(A(t)_k + A(t)_k^T)y^T.$$

Observe that $A(t)_k + A(t)_k^T \in \mathrm{M}_n(R)$ is a symmetric matrix with even elements along the main diagonal. By evaluating the variable vector $t = (t_1, \dots, t_d)$ at a vector $a = (a_1, \dots, a_d) \in \mathbb{F}^d$, we obtain the polar form $\bar{f}_k^{(a)}$ of $f_k^{(a)}$ from the polar form $\bar{f}_k^{(t)}$ of $f_k^{(t)}$.

By setting $\bar{F}^{(t)} = (\bar{f}_1^{(t)}, \dots, \bar{f}_m^{(t)}) \in \bar{P}^m$, we call $\bar{F}^{(t)}$ the *polar map* of the quadratic map $F^{(t)}$. Clearly, $\bar{F}^{(a)}$ is the polar map of the quadratic map $F^{(a)}$, for all $a = (a_1, \dots, a_d) \in \mathbb{F}^d$.

Definition 9.1. We call $F^{(t)} = (f_1^{(t)}, \dots, f_m^{(t)})$ an *oil–vinegar map* if $F^{(a)}$ is an oil–vinegar map, for each $a = (a_1, \dots, a_d) \in \mathbb{F}^d$. If $O(a) \subset \mathbb{F}^n$ is the oil subspace of $F^{(a)}$, we assume that $\dim_{\mathbb{F}} O(a) = m$, for all $a \in \mathbb{F}^d$.

We now illustrate how to construct an oil–vinegar map $F^{(t)} = (f_1^{(t)}, \dots, f_m^{(t)})$. Let $H(t) \in \mathrm{M}_{m \times n-m}(R)$ be a matrix whose entries are linear forms in the variables $t_1, \dots, t_d$ and consider the block matrix $(H(t)I) \in \mathrm{M}_{m \times n}(R)$ where I is the identity matrix of order m. Denote by $O(t) \subset R^n$ the vector subspace generated by the rows of the matrix $(H(t)I)$. Note that if $O(a) \subset F^n$ is the vector subspace generated by the rows of the matrix $(H(a)\ I) \in \mathrm{M}_{m \times n}(\mathbb{F})$, then $\dim_{\mathbb{F}} O(a) = m$ for every $a = (a_1, \dots, a_d) \in \mathbb{F}^d$.

Let $A(t)'_k \in \mathrm{M}_{n-m}(R), A(t)''_k \in \mathrm{M}_{n-m \times m}(R)$ $(1 \leq k \leq m)$ be matrices whose entries are linear forms in R and assume that $A(t)'_k$ is an upper triangular matrix. Denote by $A(t)'''_k \in \mathrm{M}_m(R)$ the upper triangular matrix defining the same quadratic form of the matrix

$$-H(t)A(t)''_k - H(t)A(t)'_k H(t)^T.$$

Since the entries of $A(t)'_k$, $A(t)''_k$ and $H(t)$ are linear forms, it is important to note that the entries of $A(t)'''_k$ are cubic polynomials with homogeneous components of degrees 2 and 3.

We finally define the block upper triangular matrices

$$A(t)_k = \begin{pmatrix} A(t)'_k & A(t)''_k \\ 0 & A(t)'''_k \end{pmatrix} \in \mathrm{M}_n(R)$$

and the corresponding quadratic forms $f_k^{(t)} = xA(t)_k x^T$ $(1 \leq k \leq m)$. By defining $F^{(t)} = (f_1^{(t)}, \dots, f_m^{(t)})$, it is clear that $F^{(a)}$ is an oil–vinegar map with $O(a)$ as its oil subspace.

We present now a direct instantiation of the IPFE protocol in [4] with the modern UOV scheme and discuss its limitations. Up to optimizations in key generation similar to those used for UOV, we have that the secret key sk held by the Master consists of an oil–vinegar map $F^{(t)} = (f_1^{(t)}, \dots, f_m^{(t)})$ along with the corresponding oil subspace $O(t) \subset R^m$. The public key pk given to Alice is the oil–vinegar map $F^{(t)}$ alone, and the user secret key usk_a given to Bob consists of the oil–vinegar map $F^{(a)} = (f_1^{(a)}, \dots, f_m^{(a)})$ along with the oil subspace $O(a) \subset \mathbb{F}^m$.

We observe that the subspaces $O(t)$ and $O(a)$ are assigned using matrices $H(t) \in \mathrm{M}_{m\times n-m}(R)$ and $H(a) \in \mathrm{M}_{m\times n-m}(\mathbb{F})$. To prevent a collusion attack on the protocol that could, for example, determine $H(t)$ from the knowledge of various matrices $H(a_1), \dots, H(a_l)$, it is essential to limit the number s of functional keys $\{a_1, \dots, a_s\} \subset \mathbb{F}^d$ allowed by the protocol relative to the dimension d of the plaintexts $b \in \mathbb{F}^d$.

We will now illustrate how Alice performs the functional encryption of their plaintext $b = (b_1, \dots, b_d) \in \mathbb{F}^d$ using the public key $F^{(t)}$ and how Bob can perform the functional decryption using the user secret key $usk_a = (F^{(a)}, O(a))$. We recall that the goal of the (inner product) functional decryption is to determine the inner product $F(a, b) = \langle a, b\rangle = \sum_i a_i b_i$.

Following the general framework, the functional encryption is defined for any functional key $a = (a_1, \dots, a_d)$. We will see that this implies that the corresponding ciphertext is generally a large data. Alternatively, Alice could construct different ciphertexts for different functional keys. Nonetheless, we will examine the general scheme in the following algorithms.

Algorithm 9.1 IPFE protocol [4] specialized with UOV - Master's Key generation

Input: $d, n, m > 0$ integers.
Output: the key pair (sk, pk).

1: Generate at random the $m \times n - m$ matrix $H(t) \in \mathrm{M}_{m\times n-m}(R)$ whose entries are linear forms in $t_1, \dots, t_d$;
2: **for** $k = 1$ **to** m **do**
3: choose at random an upper triangular matrix $A(t)'_k \in \mathrm{M}_{n-m}(R)$;
4: choose at random m matrices $A(t)''_k \in \mathrm{M}_{n-m\times m}(R)$;
5: $A(t)'''_k :=$ the upper triangular matrix defining the same quadratic form
6: of the matrix $-H(t)A(t)''_k - H(t)A(t)'_k H(t)^T \in \mathrm{M}_m(R)$;
7: $A(t)_k := \begin{pmatrix} A(t)'_k & A(t)''_k \\ 0 & A(t)'''_k \end{pmatrix} \in M_n(R)$;
8: define $C_k(t) := (A(t)'_k + A(t)'^T_k)H(t)^T + A(t)''_k$;
9: **end for**;
10: $sk := (A(t)'_1, C_1(t), \dots, A(t)'_m, C_m(t), H(t))$;
11: $pk := (A(t)_1, \dots, A(t)_m)$;
12: **return** (sk, pk);

Algorithm 9.2 IPFE protocol [4] specialized with UOV - User's Key generation

Input: Master's secret key sk, Bob's functional key $a = (a_1, \dots, a_d)$.
Output: Bob's secret key usk_a.

1: Evaluate the elements $A(t)'_1, C_1(t), \dots, A(t)'_m, C_m(t), H(t)$ in sk at Bob's functional key a:
2: $usk_a := (A(a)'_1, C_1(a), \dots, A(a)'_m, C_m(a), H(a))$;
3: **return** usk_a;

Algorithm 9.3 IPFE protocol [4] specialized with UOV - Encryption

Input: a message $b \in \mathbb{F}^d$, the public key pk.
Output: the cyphertext $\mathrm{CT}^{(t)}(b)$.
1: compute the linear form $\langle t, b\rangle = \sum_i t_i b_i \in R$;
2: generate at random the linear forms $(u(t)_1, \ldots, u(t)_{n-1})$;
3: define $u(t)_n = \langle t, b\rangle - \sum_{i=1}^{n-1} u(t)_i = \langle t, b\rangle$ and call $u(t) = (u(t)_1, \ldots, u(t)_n)$;
4: choose at random $c \in \mathbb{F}^n$ and define $v(t) = u(t) + c$;
5: compute $\bar{u}(t) = F^{(t)}(u(t))$ and $\bar{v}(t) = F^{(t)}(v(t))$;
6: **return** $\mathrm{CT}^{(t)}(b) = (\bar{u}(t), \bar{v}(t), c)$;

Observe that each component of the vectors $\bar{u}(t), \bar{v}(t) \in R^n$ is a polynomial of type

$$\bar{u}(t)_k = \sum_{i \le j} A(t)_k^{ij} u(t)_i u(t)_j, \bar{v}(t)_k = \sum_{i \le j} A(t)_k^{ij} v(t)_i v(t)_j.$$

The (inner product) functional decryption performed by Bob, namely the computation of $\langle a, b\rangle$, involves the following operations.

Algorithm 9.4 IPFE protocol [4] specialized with UOV - Decryption

Input: a ciphertext $\mathrm{CT}^{(t)}(b)$, the keys $(usk_a,\ pk)$.
Output: $\langle a, b\rangle$.
1: evaluate $\mathrm{CT}^{(t)}(b)$ at a, obtaining $(\bar{u} := \bar{u}(a), \bar{v} := \bar{v}(a), c)$;
2: flag := `false`;
3: **while** flag := `false` **do**
4: run Algorithm 6.2 on $\bar{u}$ obtaining $u' = (u'_1, \ldots, u'_n)$;
5: run Algorithm 6.2 on $\bar{v}$ obtaining $v' = (v'_1, \ldots, v'_n)$;
6: **if** $v' - u' = c$ **then**
7: flag := `true`;
8: **end if**;
9: **end while**
10: **return** $\langle a, b\rangle := \sum_{i=1}^{n} u'_i$;

According to the UOV protocol, Bob's knowledge of the oil subspace $O(a)$ enables him to efficiently compute an element of any preimage of the oil-vinegar map $F(a)$. In fact, we recall that such a computation reduces to solving a system of n linear equations in n variables. Moreover, assuming that the polynomial map $F^{(a)}$ is sufficiently generic, the dimension of a preimage, as an algebraic variety, is $n - m$. Consequently, the dimension of the product of preimages $(F^{(a)})^{-1}(\bar{u}) \times (F^{(a)})^{-1}(\bar{v})$ is $d = 2n - 2m$. Under the assumption that n is slightly larger than $2m$, we obtain that $d \approx n$. If we impose the n linear equations corresponding to the condition $v - u = c$ on the pairs $(u, v) \in (F^{(a)})^{-1}(\bar{u}) \times (F^{(a)})^{-1}(\bar{v})$, we obtain essentially a unique solution that coincides with the pair $(u(a), v(a))$. In other words, the probability of finding this pair in the product set $(F^{(a)})^{-1}(\bar{u}) \times (F^{(a)})^{-1}(\bar{v})$ is one out of its number of elements, which can be approximately estimated as $1/q^n$. This explains the correctness of the functional encryption-decryption algorithms described above.

Note that the protocol described in Decryption Algorithm 9.3 has a computational cost of the order of q^{n-m}. This (exponential, and thus infeasible) approach is directly derived from [4], with whom it shares the trial-and-error decryption algorithm: Bob continues to generate valid preimages u and v for $\bar{u}$ and $\bar{v}$ until the check $v - u = c$ is satisfied. Looking back at UOV Signature Algorithm 6.2, this method turns out to be equivalent to keep generating random vectors $u' \in \mathbb{F}^{n-m}$, until the correct preimage u is found.

To solve this issue we consider here a variant in which Alice encodes their information in a particular subspace of R^n, so that, when the ciphertext is specialized by Bob by using their functional key, the decryption algorithm can be speed-up. Before providing the details of the protocol we present the general idea.

Consider a vector subspace $V \subset \mathbb{F}^n$ of dimension ℓ such that its intersection O' with Bob's Oil subspace O has dimension $k\mathrm{lt}\ell$. Without loss of generality, we can assume that V is defined via a linear map

$$\mathcal{V}\colon \mathbb{F}^{\ell} \to \mathbb{F}^n$$

such that its restriction to the first ℓ coordinates of the co-domain is a bijection. If Alice is given $\mathcal{V}$, they can proceed similarly to Algorithm 9.3, with the difference that Steps 2 and 3, used to compute the vector $u(t) = (u(t)_1, \dots, u(t)_n) \in R^n$, are replaced by the following:

2. $u'(t) = (u'(t)_1, \dots, u'(t)_{\ell})$ consists of linear forms in the variables $t_1, \dots, t_d$ such that $\sum_{i=1}^{\ell} u'(t)_i = \langle t, b \rangle$.
3. $u(t) = (u(t)_1, \dots, u(t)_n)$ is computed as $\mathcal{V}(u'(t))$.

Thus, Bob performs the functional decryption using Algorithm 9.4, with the slight modification that the search is confined to the subspace V. In this way, the algorithm runs in $O(q^{\ell-k})$, in which $\ell - k$ can be fixed of the order of $\log(n)$. In this way, the computational cost of the decryption algorithm is polynomial in n instead of exponential in $n - m$.

It is important to note that the variant just described cannot directly be used. Indeed, an attacker can easily guess an element of Bob's Oil subspace by looking at $\mathcal{V}$. To solve this issue, we consider V to be a parametric space $V^{(t)}$, so that, at the cost of increasing the ciphertext's size, an attacker is not capable of guessing the subspace V associated to Bob's oil subspace (which, with this newly introduced notation, is $V^{(a)}$) and thus to directly attack the scheme.

Finally, to further improve the decryption algorithm 9.4, instead of searching for two preimages satisfying the check $v - u = c$ we modify the scheme so that Bob can directly check whether u is correct without generating a corresponding v.

Remark 9.2. A key difference with respect to the IPFE protocol described in Section 9 is the necessity of knowing in advance the user's functional keys. In Section 9 the key-generation is composed by two algorithms run by the authority: Algorithm 9.1 used to obtain the parametric public key pk and the master secret key sk, and Algorithm 9.2 used to specialize the master key sk to obtain users' secret keys usk. In this section instead, during the key generation, the authority generates all the user's secret keys $usk_1, \dots, usk_s$ and the corresponding unique parametric public key pk. No further users can be added after the key-generation, so there is no need to store che master key. This results in the key-generation algorithm 9.5. This choice has been made to mitigate key-recovery attacks that could leverage on the knowledge of $\mathcal{V}^{(t)}$ (which is part of the public key and is linked to users' secret keys).

In Algorithms 9.5, 9.6, and 9.7 we describe in details the key-generation, encryption, and decryption of our variant of the IPFE protocol based on [4] and UOV.

Algorithm 9.5 IPFE protocol - Key generation

Input: $d, n, m, k, \ell > 0$ integers, list of users' functional keys $\{a_1, \ldots, a_s\}$, where $a_j = (a_{j,1}, \ldots, a_{j,d})$.
Output: the list of users' secret keys $usk_1, \ldots, usk_s$, the public key pk.

flag := `false`;
while flag := `false` **do**
  choose at random d linear maps $\mathcal{V}_1, \ldots, \mathcal{V}_d : \mathbb{F}^\ell \to \mathbb{F}^n$;
  define $\mathcal{V}^{(t)} = \sum_{i=1}^{d} t_i \mathcal{V}_i$;
  flag := `true`
  for $j = 1$ **to** s **do**
    Specialize $\mathcal{V}^{(t)}$ in a_j obtaining the space $V^{(a_j)} \subset \mathbb{F}^n$;
    if $\dim(V^{(a_j)}) < \ell$ **then**
      flag := `false`;
    end if;
  end for;
end while;
for $j = 1$ **to** s **do**
  Generate randomly a subspace $O'(a_j)$ of $V(a_j)$ of dimension k
  Generate randomly the Oil Subspace of the j-th user $O(a_j)$ of dimension m s.t. $O(a_j) \cap V^{(a_j)} = O'(a_j)$;
end for;
Generate at random the $m \times n - m$ matrix $H(t) \in \mathrm{M}_{m \times n-m}(R)$ whose entries are linear forms in $t_1, \ldots, t_d$, such that the rows of $(H(a_j)\ I)$ span $O(a_j)$ for any $j = 1, \ldots, s$.
for $k = 1$ **to** m **do**
  choose at random an upper triangular matrix $A(t)'_k \in \mathrm{M}_{n-m}(R)$;
  choose at random m matrices $A(t)''_k \in \mathrm{M}_{n-m \times m}(R)$;
  $A(t)'''_k$:= the upper triangular matrix defining the same quadratic form of the matrix $-H(t)A(t)''_k - H(t)A(t)'_k H(t)^T \in \mathrm{M}_m(R)$;
  $A(t)_k := \begin{pmatrix} A(t)'_k & A(t)''_k \\ 0 & A(t)'''_k \end{pmatrix} \in M_n(R)$;
  define $C_k(t) := (A(t)'_k + A(t)'^T_k)H(t)^T + A(t)''_k$;
end for;
for $j = 1$ **to** s **do**
  $usk_j := (a_j, A(a_j)'_1, C_1(a_j), \ldots, A(a_j)'_m, C_m(a_j), H(a_j))$;
end for;
$pk := (A(t)_1, \ldots, A(t)_m, \mathcal{V}^{(t)})$;
return $(usk_1, \ldots, usk_l, pk)$;

Algorithm 9.6 IPFE protocol - Encryption

Input: a message $b \in \mathbb{F}^\ell$, the public key pk.
Output: the cyphertext $\mathrm{CT}^{(t)}(b)$.

compute the linear form $\langle t, b \rangle = \sum_i t_i b_i \in R$;
generate at random the linear forms $(u'(t)_1, \ldots, u'(t)_{\ell-1})$;
define $u'(t)_\ell = \langle t, b \rangle - \sum_{i=1}^{\ell-1} u'(t)_i = \langle t, b \rangle$ and call $u'(t) = (u'(t)_1, \ldots, u'(t)_\ell)$;
define the quadratic form $u(t) = \mathcal{V}^{(t)}(u'(t))$;
choose at random $c \in \mathbb{F}^n$ and define $v(t) = u(t) + c$;
compute $\bar{u}(t) = F^{(t)}(u(t))$ and $\bar{v}(t) = F^{(t)}(v(t))$;
return $\mathrm{CT}^{(t)}(b) = (\bar{u}(t), \bar{v}(t), c)$;

Algorithm 9.7 IPFE protocol - Decryption

Input: a ciphertext $\mathrm{CT}^{(t)}(b)$, the keys $(usk_j,\ pk)$.
Output: $\langle a_j, b\rangle$.

evaluate $\mathrm{CT}^{(t)}(b)$ at a_j, obtaining $\mathrm{CT}^{(a_j)}(b) = (\bar{u} := \bar{u}(a_j), \bar{v} := \bar{v}(a_j), c)$;
flag := `false`;
while flag := `false` **do**
    run Algorithm 6.2 on $\bar{u}$ by limiting the preimages to V, obtaining $u = (u_1, \ldots, u_n)$;
    if $\bar{v}_i - \bar{u}_i = c(A_i + A_i^T)v^T$ for $i = 1, \ldots, m$ **then**
        flag := `true`;
    end if;
end while
determine $u' = (u'_1, \ldots, u'_\ell)$ from u by the equation $\mathcal{V}^{(a))}(u') = u$;
return $\langle a_j, b\rangle := \sum_{i=1}^{\ell} u'_i$;

10 Key and ciphertext sizes in the IPFE protocol

The public key *pk* possessed by Alice is composed by two parts:
1) a parametric oil–vinegar map $F^{(t)} = (f_1^{(t)}, \ldots, f_m^{(t)})$;
2) a parametric linear map $\mathcal{V}^{(t)}$.

Recall that the map $F^{(t)} = (f_1^{(t)}, \ldots, f_m^{(t)})$ where $f_k^{(t)} = xA(t)_k x^T \in P$ is a quadratic form defined by the matrix

$$A(t)_k = \begin{pmatrix} A(t)'_k & A(t)''_k \\ 0 & A(t)'''_k \end{pmatrix}$$

with $A(t)'_k, A(t)''_k$ being random matrices whose entries are linear forms in R and $A(t)'''_k$ is determined by these matrices and the oil subspace $O(t) \subset R^n$. By generating the random matrices $A(t)'_k, A(t)''_k$ $(1 \le k \le m)$ using a pseudorandom generator from a public seed, for the public key *pk* we only need to store the upper triangular matrices $A(t)'''_k \in \mathrm{M}_m(R)$ $(1 \le k \le m)$ in addition to the public seed. We recall that the entries of $A(t)'''_k$ are cubic polynomials with homogeneous components of degree 2 and 3. Since each element of the finite field $\mathbb{F} = \mathrm{GF}(q)$ is represented by $\lceil \log_2(q) \rceil$ bits, each entry of the upper triangular matrix $A(t)'''_k$ is represented by the following number of bits

$$\left(\frac{d(d+1)}{2} + \frac{d(d+1)(d+2)}{6}\right)\lceil \log_2(q)\rceil = \frac{d(d+1)(d+5)}{6}\lceil \log_2(q)\rceil,$$

so that we have

$$\mathrm{size}(F^{(t)}) = \frac{m^2(m+1)}{2}\frac{d(d+1)(d+5)}{6}\lceil \log_2(q)\rceil + |\mathrm{seed}_{\mathrm{pub}}|.$$

The second part of the public key is $\mathcal{V}^{(t)}$. Recall that $\mathcal{V}^{(t)} = \sum_{i=1}^{\mathrm{d}} t_i \mathcal{V}_i$ where $\mathcal{V}_i \colon \mathbb{F}^\ell \to \mathbb{F}^n$ are randomly generated maps. This implies that it is sufficient to store the seed used. In particular, the set $\{\mathcal{V}_i\}$ can be obtained again by using $\mathrm{seed}_{\mathrm{pub}}$, which has been already accounted for. Putting everything together we have

$$\mathrm{size}(pk) = \frac{m^2(m+1)}{2}\frac{d(d+1)(d+5)}{6}\lceil \log_2(q)\rceil + |\mathrm{seed}_{\mathrm{pub}}|.$$

The user secret key usk_j owned by Bob is given by $a_j, A(a_j)'_k, A(a_j)''_k$ $(1 \le k \le m)$, $H(a_j)$, and $\mathcal{V}(a_j)$. Using the public seed, Bob can obtain $A(t)'_k, A(t)''_k$ and $\mathcal{V}^{(t)}$ which can be transformed into $A(a_j)'_k, A(a_j)''_k$ and $\mathcal{V}^{(a_j)}$ by

evaluation. However, the matrix $H(t)$ in the secret key sk must remain unknown to Bob, so the matrix $H(a_j)$ must be explicitly provided to them. We therefore have

$$\text{size}(usk_j) = m(n-m)\lceil \log_2(q)\rceil + |\text{seed}_{\text{pub}}|.$$

Note that in Algorithm 9.5 Bob is also provided with the matrices

$$C(a)_k = (A(a)'_k + A(a)'^T_k)H(a)^T + A_k(a)4''$$

aiming to speed up the computation of preimages. In this case the size becomes

$$\begin{aligned}&(m(n-m) + m^2(n-m))\lceil \log_2(q)\rceil + |\text{seed}_{\text{pub}}|\\ &\quad = m(m+1)(n-m)\lceil \log_2(q)\rceil + |\text{seed}_{\text{pub}}|.\end{aligned}$$

Finally, we calculate the size of the plaintext and ciphertext. A plaintext is a vector $b = (b_1, \dots, b_d) \in \mathbb{F}^d$, which means it consists of d elements from the field $\mathbb{F}$. We therefore have

$$\text{size}(b) = d\lceil \log_2(q)\rceil.$$

The corresponding ciphertext $\text{CT}^{(t)}(b) = (\bar{u}(t), \bar{v}(t), c)$ consists of the vectors $\bar{u}(t) = (\bar{u}(t)_1, \dots, \bar{u}(t)_m) = F^{(t)}(u(t))$ and $\bar{v}(t) = (\bar{v}(t)_1, \dots, \bar{v}(t)_m) = F^{(t)}(v(t))$ where $u(t) = \mathcal{V}^{(t)}(u'(t))$ and $v(t) = u(t) + c$ are each ℓ quadratic forms in d unknowns. Thus, we have that $\bar{u}(t)_k = u(t)_k A(t)_k u(t)^T_k \in R$ and $\bar{v}(t)_k = v(t)_k A(t)_k v(t)^T_k \in R$ $(1 \le k \le m)$ where

$$A(t)_k = \begin{pmatrix} A(t)'_k & A(t)''_k \\ 0 & A(t)'''_k \end{pmatrix}$$

and $A(t)'_k \in \text{M}_{n-m}(R), A(t)''_k \in \text{M}_{n-m\times m}(R)$ are matrices whose entries are linear forms and $A(t)'_k$ is an upper triangular matrix. Moreover, $A(t)'''_k \in \text{M}_m(R)$ is an upper triangular matrix whose entries are cubic polynomials with homogeneous components of degree 2 and 3. We therefore have that $\bar{u}(t)_k, \bar{v}(t)_k$ are polynomials in d variables of degree 7. We conclude that the size of the ciphertext $\text{CT}^{(t)}(b)$ is

$$\text{size}(\text{CT}^{(t)}(b)) = (2m\binom{d+7}{7} + n)\lceil \log_2(q)\rceil.$$

11 Security and parameters

We start by considering the security of [4] specialized with UOV (Algorithms 9.1, 9.2, 9.3, and 9.4). A first issue when studying the security of a Functional Encryption protocol is that such a protocol is naturally exposed to "collusion attacks" where different functional decryptors, different Bobs, agree to share the information available to them. Specifically, the decryptors could exchange their functional keys a and thus the values of the functions $F(a, b)$, in order to determine the plaintext b that Alice has functionally encrypted. This type of attack is particularly risky in the case of IPFE, where different functional keys $a_1, \dots, a_l \in \mathbb{F}^d$ determine a system of linear equations in the variables $t_1, \dots, t_d$ of type

$$\begin{cases} \langle a_1, t\rangle = \langle a_1, b\rangle \\ \qquad\vdots \\ \langle a_l, t\rangle = \langle a_l, b\rangle \end{cases}$$

The plaintext b is clearly one of the solutions of this linear system. It is therefore absolutely essential that the number of functional keys allowed in the protocol is not too high compared to the dimension d of the plaintexts.

In particular, if l is less than d, than the linear system is underdetermined. Fortunately, this assumption is not too restrictive in practical applications of IPFE.

To perform a security analysis of an FE protocol, the notion of security commonly employed is the paradigm of "simulation-based security". Specifically, let $b \in \mathbb{F}^d$ be a plaintext generated by a challenger CH who possesses the public key, and let $a_1, \dots, a_l \in \mathbb{F}^d$ be the functional keys owned by an adversary AD. We can identify the challenger CH as Alice and the adversary AD as multiple Bobs who collaborate on a collusion attack.

We denote by GAME(0) the so-called "real experiment" of simulation-based security, which is the activity of AD attempting to distinguish different ciphertexts generated by CH from different plaintexts.

Now we consider another activity, denoted by GAME(1), which we consider intermediate between the real experiment GAME(0) and the so-called "ideal experiment" GAME(2) of simulation-based security. We will explain later what the activity of GAME(2) consists of. Indeed, we recall that security in this paradigm is achieved when GAME(0) and GAME(2) are computationally indistinguishable by the adversary AD with only a negligible probability of success.

Starting from the same public key pk, the activity of the challenger CH is replaced in GAME(1) by that of a simulator SIM_1. Given an original plaintext b, the agent SIM_1 generates a distinct plaintext $b^* \neq b$ such that its corresponding ciphertext $\text{CT}^{(t)}(b^*) = (\bar{u}^*(t), \bar{v}^*(t), c^*)$ satisfies, for each $i = 1, 2, \dots, l$

$$\langle a_i, b \rangle = \langle a_i, b^* \rangle. \tag{1}$$

In other words, AD observes the same functional decryption for the plaintexts b and b^* when using the different functional keys a_i it possesses. Hence, functional decryption does not help AD to distinguish between b and b^*. Note that the vector b^* can be efficiently computed by SIM_1 from the vectors b and $a_1, \dots, a_l$ by using the linear equations (1).

Assuming that the randomly selected public key pk consists of a generic quadratic map $F^{(t)}$, the ciphertexts $\text{CT}^{(t)}(b)$ and $\text{CT}^{(t)}(b^*)$ generated by the activities of GAME(0) and GAME(1) respectively, share equal probability within the ciphertext space. Therefore, the adversary AD is left only to try to compute the plaintexts b and b^* in order to distinguish the corresponding ciphertexts. Precisely, if $\{e_k\}_{1 \leq k \leq d}$ is the canonical basis of the vector space $\mathbb{F}^d$, observe that

$$\sum_i u(e_k)_i = \langle e_k, b \rangle = b_k, \sum_i u^*(e_k)_i = \langle e_k, b^* \rangle = b_k^*.$$

In other words, by computing the preimages $u(e_k) \in \mathbb{F}^n$ of the vectors $\bar{u}(e_k) \in \mathbb{F}^m$ under the map $F^{(e_k)}$ $(1 \leq k \leq d)$, the adversary AD obtains the plaintext b. In a similar way, AD can compute b^*. Observe that computing a preimage of a generic quadratic map over a finite field is an NP-hard problem. Thus, AD can distinguish the ciphertexts $\text{CT}^{(t)}(b)$ and $\text{CT}^{(t)}(b^*)$ by computing b and b^*, respectively, with a negligible probability of efficiently solving such a problem. We denote this negligible probability by ϵ_1. Using the language of simulation-based security, we then write

$$|\Pr(\text{GAME}(1)) - \Pr(\text{GAME}(0))| \leq \epsilon_1.$$

Similarly to the activity of GAME(1), the ideal experiment GAME(2) consists in replacing the challenger CH with a simulator SIM_2 who computed a third plaintext b^{**} such that, for each $i = 1, 2, \dots, l$, we have

$$\langle a_i, b \rangle = \langle a_i, b^* \rangle = \langle a_i, b^{**} \rangle.$$

Therefore, the adversary AD cannot distiguish b, b^* and b^{**} using the functional keys he owns. The agent SIM_2 differs from SIM_1 in that the former can use a different public key pk, that is, a distinct polynomial map $F^{(t)}$ from the one used in the real experiment. Since the ciphertext $C^{(t)}(b^{**})$ is computed by the generic polynomial map $F^{(t)}$, this ciphertext is probabilistically uncorrelated with those generated by GAME(0) and GAME(1). This implies that the only way to distinguish the outcomes of these activities from that of GAME(2) is by determining the plaintexts

Table 1: Parameters.

λ	d	n	m	k	ℓ	q	$\mathrm{CT}^{(t)}(b)$
128	20	112	44	19	20	256	78.1MB

b, b^* and b^{**} in order in order to identify the corresponding ciphertexts. Again, this implies the ability of AD to solve generic systems of quadratic equations over a finite field, which is a known NP-hard problem.

Consequently, distinguishing GAME(2) from activities GAME(1) and GAME(0) can be achieved by AD with negligible probability. Specifically, we have

$$|\Pr(\text{GAME}(2)) - \Pr(\text{GAME}(1))| \leq \epsilon_2$$

and thus

$$\begin{aligned}|\Pr(\text{GAME}(2)) - \Pr(\text{GAME}(0))| &\leq |\Pr(\text{GAME}(2)) - \Pr(\text{GAME}(1))| \\ &+ |\Pr(\text{GAME}(1)) - \Pr(\text{GAME}(0))| \leq \epsilon_1 + \epsilon_2 = \epsilon\end{aligned}$$

where ϵ is still a negligible probability.

The security of the proposed IPFE scheme (Algorithms 9.5, 9.6, and 9.7) relies on one hand of the security of [4] instantiated with UOV, and on the other hand on the inability of the attacker to determine (part of) Bob's Oil subspace by looking at the public key $\mathcal{V}(t)$. By construction, $\mathcal{V}(t)$ and $O(t)$ are generated randomly: the only link is that, for any a_j in the predetermined set $\{a_1, \ldots, a_s\}$, $\mathcal{V}(a)$ is a subspace of Bob's oil subspace $O(a)$. This implies that, if $\tilde{a} \notin \{a_1, \ldots, a_s\}$, then $\mathcal{V}(\tilde{a})$ is a random vector subspace of $\mathbb{F}^n$, thus unrelated to the hypothetical corresponding oil subspace $O(\tilde{a})$. In other words, an attacker have to guess Bob's functional key a_j, but a wrong guess $\tilde{a} \neq a_j$ will lead to a specialized public key $(A(\tilde{a})_1, \ldots, A(\tilde{a})_m, \mathcal{V}(\tilde{a}))$ where $O(\tilde{a})$ and $\mathcal{V}(\tilde{a})$ are random and not linked together, thus preventing its ability to decrypt due to the security of UOV. To achieve the required security it is necessary that the number s of valid functional keys (those associated to UOV secret keys) is negligible with respect to q^d, for example

$$\frac{q^d}{|\mathrm{s}|} \geq 2^\lambda$$

with λ the security parameter. Note that this assumption is in line with the already discussed limitation on the number of functional keys to resist against collusion attacks.

In this case, a functional key recovery attack implies the solution of nonlinear system in $d + n$ variables of degree 7 obtained by searching for zeros of the map $F^{(t)} \circ \mathcal{V}^{(t)}(x)$.

In Table 1 we show an example of parameters set corresponding to the instantiation UOV-Ip.

As we can see from Table 1, the main open problem is to determine a more efficient encryption algorithm. In particular, the large ciphertext is a set of polynomial of degree 7 in d variables, and a possibility is to design a method to lower this degree. For example, with the presented parameters, by lowering the degree from 7 to 5 we would obtain a compression of the ciphertext of the order of 94 %, resulting, for the parameters choices in Table 1, in a ciphertext of 4.7MB.

A possibility is to change the key-generation algorithm to obtain a linear (in t) public key $F^{(t)}$. In Algorithm 11.1 we propose a variant of Algorithm 9.5 with this feature, whose security is however still under investigation.

Algorithm 11.1 IPFE protocol - Key generation (variant)

Input: $d, n, m, k, \ell > 0$ integers, list of users' functional keys $\{a_1, \ldots, a_s\}$, where $a_j = (a_{j,1}, \ldots, a_{j,d})$.

Output: the list of users' secret keys $sk_1, \ldots, sk_s$, the public key pk.

flag := `false`;
while flag := `false` **do**
  choose at random d linear maps $\mathcal{V}_1, \ldots, \mathcal{V}_d : \mathbb{F}^\ell \to \mathbb{F}^n$;
  define $\mathcal{V}^{(t)} = \sum_{i=1}^{d} t_i \mathcal{V}_i$;
  flag := `true`
  for $j = 1$ **to** s **do**
    Specialize $\mathcal{V}^{(t)}$ in a_j obtaining the space $V^{(a_j)} \subset \mathbb{F}^n$;
    if $\dim(V^{(a_j)}) < \ell$ **then**
      flag := `false`;
    end if;
  end for;
end while;
for $j = 1$ **to** s **do**
  Generate randomly a subspace $O'(a_j)$ of $V(a_j)$ of dimension k
  Generate randomly the Oil Subspace of the j-th user $O(a_j)$ of dimension m s.t. $O(a_j) \cap V^{(a_j)} = O'(a_j)$;
end for;
Generate at random the $m \times n - m$ matrix $H(t) \in \mathrm{M}_{m \times n-m}(R)$ whose entries are linear forms in $t_1, \ldots, t_d$, such that the rows of $(H(a_j)\ I)$ span $O(a_j)$ for any $j = 1, \ldots, s$.
for $j = 1$ **to** s **do**
  for $k = 1$ **to** m **do**
    choose at random an upper triangular matrix $A'_{j,k} \in \mathrm{M}_{n-m}(\mathbb{F})$;
    choose at random m matrices $A''_{j,k} \in \mathrm{M}_{n-m \times m}(\mathbb{F})$;
    $A'''_{j,k}$:= the upper triangular matrix defining the same quadratic form of the matrix $-H(a_j)A''_{j,k} - H(a_j)A'_{j,k}H(a_j)^T \in \mathrm{M}_m(\mathbb{F})$;
    $A_{j,k} := \begin{pmatrix} A'_{j,k} & A''_{j,k} \\ 0 & A'''_{j,k} \end{pmatrix} \in M_n(\mathbb{F})$;
    define $C_{j,k} := (A'_{j,k} + A'^T_{j,k})H(a_j)^T + A''_{j,k}$;
  end for;
  $usk_j := (a_j, A'_{j,1}, C_{j,1}, \ldots, A'_{j,m}, C_{j,m}, H(a_j))$;
end for;
choose at random m upper triangular matrices $A(t)_k$ in $M_n(\mathbb{F})$ whose entries are linear forms in $t_1, \ldots, t_d$, such that, for any $j = 1, \ldots, s$ and for any $k = 1, \ldots, m$, it holds $A(a_j)_k = A_{j,k}$;
$pk := (A(t)_1, \ldots, A(t)_m, \mathcal{V}^{(t)})$;
return $(usk_1, \ldots, usk_l, pk)$;

Observe that in Algorithm 11.1 the public key is linear in t. By using a public key obtained from Algorithm 11.1 instead of 9.5, Alice will obtain a degree 5 ciphertext, thus mitigating the computational cost and memory requirements of both our encryption and decryption algorithms. We remark however that, even if from a security point of view this last approach seems to be comparable with our proposal, more analyses should be conducted to exclude any vulnerability.

12 Conclusions and further directions

In this paper we discuss the properties and limitations of inner product functional encryption (IPFE) protocols using a state-of-art version of the UOV digital signature. Even though this approach improves the key generation and encryption–decryption algorithms, as well as their security foundations, the decryption algorithm remains exponential. We then propose a variant to mitigate this limitation, whose main drawback is the dimension of the ciphertext. We conclude by proposing a modification which would allow for a more compact public key and smaller ciphertext, whose security is however still under investigation.

Acknowledgments: The authors would like to thank the reviewers for their valuable comments and suggestions, which have significantly contributed to improving the clarity and readability of the manuscript.
Author contribution: All authors accept full responsibility for the entire content of this manuscript, have reviewed all results, approved the final version, and consented to its submission to the journal. The manuscript was prepared collaboratively by all authors.
Conflict of interest: Authors state no conflict of interest.
Research funding: The first author acknowledges the partial support of PNRR MUR projects “Security and Rights in the CyberSpace”, Grant ref. CUP H93C22000620001, Code PE00000014, Spoke 3, and “National Center for HPC, Big Data and Quantum Computing”, Grant ref. CUP H93C22000450007, Code CN00000013, Spoke 10. The same author was co-funded by PRIN MUR project “Algebraic Methods in Cryptanalysis”, Grant ref. CUP H53C24000830006, Code 2022RFAZCJ, and by Università degli Studi di Bari, “Fondo acquisto e manutenzione attrezzature per la ricerca”, Grant ref. DR 3191. Both authors acknowledge membership in INdAM – GNSAGA and UMI – Crittografia e Codici.

References

1. Boneh D, Sahai A, Waters B. Functional encryption: definitions and challenges. In Theory of cryptography. Lecture Notes in Comput. Sci., 6597. Heidelberg: Springer; 2011:253—73 pp.
2. Mascia C, Sala M, Villa I. A survey on functional encryption. Adv Math Commun 2023;17:1251—89.
3. Gentry C. Homomorphic encryption: a mathematical survey. Proc Int Cong Math 2022;2:956—1006.
4. Debnath SK, Mesnager S, Dey K, Kundu N. Post-quantum secure inner product functional encryption using multivariate public key cryptography. Mediterr J Math 2021;18, Paper No. 204:15.
5. Beullens W, Chen M-S, Ding J, Gong B, Kannwischer MJ, Patarin J, et al. UOV: unbalanced oil and vinegar algorithm specifications and supporting documentation, Version 1.0; 2023. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/UOV-spec-web.pdf.
6. Beullens W, Chen M-S, Hung S-H, Kannwischer MJ, Peng B-Y, Shih C-J, et al. Oil and vinegar: modern parameters and implementations. IACR Trans Cryptogr Hardw Embed Syst 2023;2023:321—65.
7. Gringiani A. Multivariate-based cryptography: a revision of MAYO parameters [Master thesis]. Trento: University of Trento; 2022.
8. Kipnis A, Patarin J, Goubin L. Unbalanced oil and vinegar signature schemes. In: Stern J, editor. Advances in Cryptology — EUROCRYPT ’99. EUROCRYPT 1999. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 1999, vol 1592:206—22 pp.
9. La Scala R, Pintore F, Tiwari SK, Visconti A. A multistep strategy for polynomial system solving over finite fields and a new algebraic attack on the stream cipher trivium. Finite Fields Appl 2024;98, Paper No. 102452:33.
10. Ghorpade SR. A note on Nullstellensatz over finite fields. In: Contributions in Algebra and Algebraic Geometry, Contemp. Math., 738. Providence, RI: Amer. Math. Soc.; 2019:23—32 pp.
11. Garey M R, David S J. Computers and intractability. New York: Wh Freeman; 2002, vol 29.
12. Kipnis A, Shamir A. Cryptanalysis of the oil and vinegar signature scheme. In: Advances in Cryptology — CRYPTO ’98, Lecture Notes in Comput. Sci., 1462. Berlin: Springer-Verlag; 1998:257—66 pp.