Politecnico di Torino

Politecnico di Torino

Master's Degree in Cybersecurity Engineering
A.a. 2024/2025
Graduation session October 2025

Olivier2

An Oil and Vinegar Based Cryptosystem

Supervisors:
Prof. Pagnin Elena
Chalmers University of Technology
Prof. Sanna Carlo
Politecnico di Torino

Candidate:
Cornaggia Emanuele

Abstract

This thesis investigates the post-quantum encryption scheme proposed by *Esposito et al.* [EFR24], which is based on the Multivariate Quadratic (MQ) problem. The work presents the scheme within a well-established cryptographic framework, and formalizes a new security assumption that was only implicitly defined in the original proposal. The thesis then analyzes weaknesses in the original construction, proposes potential improvements, and derives appropriate security parameters against the best-known attacks. It further develops an IND-CCA-2 secure KEM, and a key exchange (KE) variant of the scheme, providing practical implementations for each. Finally, the thesis identifies a key-recovery attack that compromises both the original and the modified versions, offering insights into the limitations of using the Oils and Vinegars as a trapdoor for an encryption scheme.

Acknowledgments

This thesis would not have been possible without the support and encouragement of many individuals during my two years in Turin and my six-month stay in Gothenburg.

First and foremost, I would like to express my deepest gratitude to my main supervisor, Prof. Elena Pagnin, who welcomed me into her research team at Chalmers University. Working with her has been a truly inspiring experience. She introduced me to the world of research by inviting me to conferences, involving me in group discussions, and guiding me through regular meetings that shaped and motivated me to pursue a research career beyond this thesis. Her overall guidance, trust, and immense support have been instrumental in my personal and professional growth.

I am also sincerely grateful to the other members of the Crypto Team at Chalmers: Lucia Lavagnino, Hanna Ek, Christoph Egger, Adrián Pérez Keilty, and Kelsey Melissaris, for their warm welcome and for making my stay so enriching. In particular, I would like to thank Hanna and Lucia for introducing me to the Olivier paper and guiding me through the UOV schemes, patiently addressing all my doubts. Our regular meetings and the insightful feedback they provided were key to the completion of this work. I am especially thankful to Lucia, who supported me with everything I needed during my time in Sweden, from finding accommodation to helping me navigate everyday life. She was always available and incredibly kind, and I am truly grateful for that.

I would also like to thank my internal supervisor, Prof. Carlo Sanna, for his invaluable help with the administrative and bureaucratic aspects related to my stay in Sweden and the submission of this thesis. His assistance ensured that everything ran smoothly throughout the process.

My heartfelt thanks also go to all the people I met at the university, with whom I shared projects, discussions, and friendships. In particular, I would like to thank Alessandro Genova, who had a profound impact on me. He inspired me to look deeper, even into the simplest things that might not seem worth exploring at first glance. This mindset proved essential in my research, as many of my findings emerged from such deeper investigations.

Finally, I would like to express my deepest gratitude to my family for their unwavering support during these two additional years of study. I am especially thankful to my uncles, who hosted me in Turin for one and a half years, making it possible for me to pursue this master's degree. I am also grateful to my grandparents, my other aunt, my sister and her partner, my cousins, and my nephews, who have always been there for me and have brought great joy and strength to my life.

Contents

1 Introduction

Cryptography enables confidential communication between networks and authentication to services. Without it, there would be no reliable way to prove over a communication channel that someone is who they claim to be. This is a fundamental aspect of today's world, as many critical services like banking, government, and healthcare now heavily depend on the internet. In the online banking example, demonstrating the ownership of a given account and the right to submit transactions is essential for the existence of this service. At the same time, a user wants to be sure that he is connecting to the bank's website instead of a fake one created by someone else just to steal users' credentials. Asymmetric cryptography enables the online bank to demonstrate its identity. In asymmetric cryptography a user generate two keys that are linked to each other, one is meant to be kept secret, and takes the name of secret key, and the other one is meant to be published (public key), so that every time he connects to the online banking he compute the encryption function of a chosen asymmetric cryptography algorithm using a random value as input, and only the owner of the secret key linked to that public key will be able to retrive the original input. This concludes the authentication of the online banking server. In order to create a strong asymmetric cryptography system, "hard problems" of mathematics are exploited. Those problems are believed to be (almost) impossible to solve in a reasonable amount of time. Inverting this problem is considered hard.

Instead of using random instances of the problem, a trapdoor is embedded in the instance, making inversion easy; this trapdoor is referred to as the secret key. The owner of the secret can solve the hard problem efficiently, while those who don't have the secret cannot solve it or recover the input. The two most used hard problems are based on the difficulty of factoring large numbers and the discrete logarithm. In 1994, Peter Shor demonstrated that solving this problem efficiently is possible, but to do so, a Quantum Computer is needed. Recent progress in the field of Quantum Computing has resulted in an urgency to develop cryptographic algorithms that cannot be broken by a Quantum Computer. An algorithm based on a problem for which a significant quantum advantage is not yet known is said to be quantum-resistant. Even tho it's estimated that a Quantum computer that is powerful enough to break a current cryptosystem will not be built until 2030, a very important threat that we are undergoing right now it's the collection of todays encrypted data, using current non-quantum resistant algorithm, and the recovery of the original message by breaking the system in the future using Quantum Computer, this attack is called "Harvest Now, Decrypt Later". This problem is very concerning because private data, including government data, intellectual property, medical, and financial records, should remain private even in the future. To mitigate this attack, some browsers already implement a post-quantum encryption algorithm on top of current encryption algorithms, but further research is needed to improve and evaluate this new post-quantum cryptosystem.

If we still want to utilize internet services in the future, making this a priority is essential.

1.1 NIST competition

To further emphasize the importance of this area of research, it's worth mentioning that the National Institute of Standards and Technology opened a call for quantum-resistant algorithms in 2017, and it is still open today. The NIST is a U.S. federal agency that develops and promotes measurement standards. NIST's standards are widely respected and adopted globally, particularly in cybersecurity and cryptography, where compliance with NIST guidelines is often required in government and defense systems.

1.2 Thesis' structure

Chapter 2 provides a comprehensive introduction to the proposed cryptosystem. The chapter outlines the motivation for its design, presents the system model, and details the algorithms that constitute key generation, encryption, and decryption. Chapter 3 focuses on the security foundations of the scheme. It introduces a new security assumption, outlines the security goals of the scheme, and proves them under the new assumption. Chapter 4 analyzes the best-known attack on the scheme and derives security parameters that guarantee resistance against those attacks. Chapter 5 explores the applicability of the cryptosystem in the broader cryptographic landscape. In particular, it discusses how the proposed public-key encryption scheme can be employed as a building block for constructing advanced cryptographic protocols. Chapter 6 presents the practical implementation of the cryptosystem. It describes the design decisions, provides details on the development environment, and reports experimental results evaluating the scheme's performance and efficiency in practice. Chapter 7 presents the results evaluating the efficiency of the implemented system. Chapter 8 describes a key recovery attack that compromises both the original and the modified schemes. Chapter 9 concludes the study, presenting the main insights and lessons learned from the evaluation of the scheme.

2 Olivier2 Cryptosystem

Olivier is a Multivariate public key encryption algorithm proposed by *Esposito et al.*[EFR24]. The scheme does not rely on a previously known security assumption, but introduces a new one. This new security assumption is inspired by the *Patartin* Oil and Vinegar assumption[Pat97]. It exploits the fixing of vinegar variables to obtain a linear system, but it also adds a linear combination of random quadratic polynomials to make the system stronger. This is necessary as the parameters of the OV map are considered weak.

Olivier's revised version is presented below, with the differences from the original scheme highlighted in grey.

Let $n, v, u, \ell \in \mathbb{N}$. For convenience, define $o = n - v$ and $m = n + u$. Unless otherwise specified, the system is considered over an arbitrary finite field $\mathbb{F}_q$

2.1 Key Generation

1. Generate a homogeneous OV system $\mathcal{F}$ of n equations in n variables
2. Generate a random invertible matrix $\mathcal{T}$
3. Generate a system of random homogeneous polynomials $\mathcal{Q}$ of u equations in n variables
4. Generate a $n \times u$ coefficient matrix Λ
5. Compute $\hat{\mathcal{P}}_1 = \mathcal{F} \circ \mathcal{T} + \Lambda\mathcal{Q}$
6. Let $\hat{\mathcal{P}}_2 = \mathcal{Q}$
7. Sample a random permutation of m indexes Π
8. Apply the permutation Π to $\hat{\mathcal{P}} = (\hat{\mathcal{P}}_1 | \hat{\mathcal{P}}_2)$ to obtain the final public system $\mathcal{P}$ of m equations in n variables

Public Key $\mathcal{P} = \Pi\hat{\mathcal{P}}$

$$\hat{\mathcal{P}} = \begin{cases} \mathcal{F} \circ \mathcal{T} + \Lambda\mathcal{Q} & \text{if } 1 \leq i \leq n \\ \mathcal{Q} & \text{if } n+1 \leq i \leq m \end{cases}$$

Private Key $\mathcal{F}, \mathcal{T}^{-1}, \Lambda, \Pi$

2.2 Encryption

1. Let $\mathbf{m} \in \mathbb{F}_q^{\ell}$ denote the to-be-encrypted message
2. Sample a random vector $\mathbf{r} \xleftarrow{\$} \mathbb{F}_q^{n-\ell}$
3. Evaluate the system of equations $c \leftarrow \mathcal{P}(\mathbf{m} \| \mathbf{r})$

2.3 Decryption

1. Invert the permutation $\hat{\mathcal{P}}(\mathbf{x}) = \mathcal{P}(\mathbf{x})\Pi^{-1}$ (where Π^{-1} is the inverse permutation)
2. Remove the $\Lambda\mathcal{Q}(\mathbf{x})$ polynomials and recover the evaluation on the OV map $\mathcal{F} \circ \mathcal{T}(\mathbf{x}) = \hat{\mathcal{P}}_1(\mathbf{x}) - \Lambda\mathcal{Q}(\mathbf{x}) = \mathbf{t}$
3. To recover the $\mathbf{x} = \mathbf{v} + \mathbf{o}$ iterate over all possible values of $\mathbf{v}$ and solve for $\mathbf{o}$
4. If $\mathcal{Q}(\mathbf{v} + \mathbf{o}) = \mathcal{Q}(\mathbf{x})$, $\mathbf{v} + \mathbf{o} = \mathbf{x}$ with high probability, otherwise go back to step 3 and continue the iteration over all possible $\mathbf{v}$
5. The recovered message $\mathbf{m}$ is the first ℓ bits of $\mathbf{x}$

2.4 Proof of Correctness

1. Applying the inverse permutation to c gives the evaluation on $\mathbf{x}$ of $\hat{\mathcal{P}}$. $\Pi^{-1}c = \Pi^{-1}\Pi\hat{\mathcal{P}}(\mathbf{x}) = \hat{\mathcal{P}}(\mathbf{x})$
2. To each evaluation of $\hat{\mathcal{P}}_1(\mathbf{x})$, it been summed a linear combination of the evaluation of $\mathcal{Q}(\mathbf{x})$. By removing this addition to each $\hat{\mathcal{P}}_1^{(1)}(\mathbf{x})$ the evaluation of $\mathbf{x}$ on the OV map is obtained. $(\mathcal{F} \circ \mathcal{T})^{(i)}(\mathbf{x}) = \hat{\mathcal{P}}^{(i)}(\mathbf{x}) - \sum_{j=1}^{u} \lambda_{ij}\mathcal{Q}_j = t^{(i)}$ $(i = 1, \ldots, n)$
3. Once $\mathbf{v}$ is fixed, the polar form is a linear system $\mathcal{F} \circ \mathcal{T}(\mathbf{v} + \mathbf{o}) = (\mathcal{F} \circ \mathcal{T})(\mathbf{v}) + (\mathcal{F} \circ \mathcal{T})(\mathbf{o}) + (\mathcal{F} \circ \mathcal{T})'(\mathbf{v}, \mathbf{o}) = \mathbf{t}$
4. The linear system $(\mathcal{F} \circ \mathcal{T})'(\mathbf{v}, \mathbf{o})$ may not have full rank, this mean that $\mathcal{F} \circ \mathcal{T}(\mathbf{v} + \mathbf{o}) = \mathbf{t}$ might have multiple solutions and does not necessarily hold that $\mathbf{x} = \mathbf{v} + \mathbf{o}$. The correct solution must also satisfy $\mathcal{Q}(\mathbf{v} + \mathbf{o}) = \mathcal{Q}(\mathbf{x})$. The chance of collision is negligible over n.
5. To recover the original message $\mathbf{m}$ note that $\mathbf{x} = \mathbf{v} + \mathbf{o} = \mathbf{m}\|\mathbf{r}$ where $\mathbf{m}$ are the first ℓ bits of $\mathbf{x}$

2.5 Differences from the original scheme

2.5.1 Nondeterministic encryption

In the original scheme, only the message to be exchanged $\mathbf{m}$ was encrypted; retransmitting the same message would yield an identical ciphertext, resulting in an unacceptable leakage of information. Moreover, a public-key encryption scheme with a deterministic encryption algorithm cannot achieve CPA-security. To mitigate this, one may concatenate $\mathbf{m}$ with a random value $\mathbf{r}$, ensuring that repeated encryptions of the same plaintext produce distinct ciphertexts (except when the same randomness $\mathbf{r}$ is reused)

2.5.2 Permutation

In the original paper, the public key is given by $\hat{\mathcal{P}}$. The authors note that the most efficient known attack against the scheme exploits this information. By introducing a random permutation Π, this attack is no longer possible unless an attacker is able to distinguish the polynomial in $\hat{\mathcal{P}}_1$ from

the polynomial $\hat{\mathcal{P}}_2$. Another difference closely related to this aspect is the requirement that the random polynomial $\mathcal{Q}$ be homogeneous; otherwise, it would lead to trivial distinguishing when the polynomial has a constant term. This change does not affect the security of the scheme.

2.5.3 Generalization over any finite field

The schema was originally proposed only over $\mathbb{F}_2$. Its generalization to $\mathbb{F}_q$ offers two main advantages. First, choosing $\Lambda \in \mathbb{F}_q$ increases the difficulty of distinguishing between $\hat{\mathcal{P}}_1$ and $\hat{\mathcal{P}}_2$ compared to the case $\Lambda \in \mathbb{F}_2$. Second, while transmitting a 128-bit message over $\mathbb{F}_2$ requires 128 variables, in $\mathbb{F}_q$ only $\frac{128}{\log 2q}$ variables are needed, allowing for a smaller system.

3 Security Analysis

The schema cannot rely on the UOV assumption because the decryption process requires iterating over all vinegar variables, totaling q^v. If v is too large, decryption becomes computationally infeasible. In a secure UOV system, the number of vinegar variables is typically between two and three times the number of oil variables. Such parameter sizes would make our system too small and easily breakable by direct attacks. Moreover, in UOV, the number of equations m equals the number of oil variables. This property is suitable for signature schemes, where multiple valid solutions exist (hence signatures are not unique). In contrast, for a public encryption system, each ciphertext must correspond to a unique pre-image. Therefore, the number of equations must be greater than or equal to the number of variables.

An alternative approach to ensuring the uniqueness of the pre-image is to add random polynomials, requiring that the recovered solution produce the same result as the correct pre-image for both the OV map and the random polynomials. However, there is no clear advantage in using more equations than variables; hence, the proposed system employs n equations in n variables.

For readers unfamiliar with the UOV assumption, a formal definition is provided in Appendix C.3. Intuitively, this assumption relies on hiding a structured linear space on which encryption and decryption can be efficiently performed within a general multivariate quadratic (MQ) system, while claiming that the resulting system is computationally indistinguishable from a random MQ instance.

The security of the scheme comes from masking the weak OV system by summing random polynomials to it. To express this security notion, a new assumption needs to be introduced.

3.1 Olivier Assumption

Olivier Problem *For $\mathcal{O} \in \mathbb{F}_q^o$, we let $MQ_{n,v,q}(\mathcal{O})$ denote the set of $\mathcal{P} \in MQ_{n,v,q}$ that vanish on the evaluation of $o \in \mathcal{O}$. The Olivier problem asks to distinguish a random multivariate quadratic map $\mathcal{P} \in MQ_{n,q}$ from $MQ_{n,v,q}(\mathcal{O}) + \Lambda\mathcal{Q}$ for a random $\mathcal{O} \in \mathbb{F}_q^o$ and for a subset $\mathcal{Q} \subseteq MQ_{n,u,q}$.*

Let $\mathcal{A}$ be a Olivier distinguisher algorithm. We say that the distinguishing advantage of $\mathcal{A}$ is

$$\mathrm{Adv}^{\mathrm{Olivier}}_{n,v,u,q}(\mathcal{A}) = \left| \Pr[\mathcal{A}(\mathcal{P}) = 1 | \mathcal{P} \leftarrow MQ_{n,q}] - \Pr\left[\mathcal{A}(\mathcal{P}) = 1 \;\middle|\; \begin{array}{c} \mathcal{O} \in \mathbb{F}_q^o \\ MQ_{n,v,q}(\mathcal{Q}) + \Lambda\mathcal{Q} \\ \mathcal{Q} \subseteq MQ_{n,u,q} \end{array} \right] \right|$$

Evidence for the difficulty of this problem can be seen in the following cases.

For each $MQ_{n,v,q}(\mathcal{O}) + \Lambda\mathcal{Q}$ it is possible to retrieve $\mathcal{F} \circ \mathcal{T}^{(i)}$ by:

Solving the MinRank instance. Each polynomial can be represented as a matrix. By construction, each $\mathcal{F} \circ \mathcal{T}$ has rank at most $2v$, while $\mathcal{Q}$ has maximal or nearly maximal rank with high probability. Hence, the expression

$$P^{(i)} = \mathcal{F} \circ \mathcal{T}^{(i)} + \sum_{j=1}^{u} \mathcal{Q}$$

defines a MinRank instance with rank at most $2v$. The MinRank problem is known to be NP-hard.

Building a linear system. Let $x_{j,k}$ denote each variable of the linearized system, and let $a_{j,k}$ be the coefficient corresponding to the upper-triangular matrix $\mathcal{F} \circ \mathcal{T}^{(i)}$. Let λ denote the i-th row of the Λ map. The evaluation of a single polynomial of the form

$$MQ_{n,v,q}(\mathcal{O}) + \Lambda\mathcal{Q}$$

can be expressed as

$$x_{1,1}a_{1,1} + x_{1,2}a_{1,2} + \cdots + x_{n,n}a_{n,n} + \lambda_1\mathcal{Q}_1(\mathbf{x}) + \cdots + \lambda_u\mathcal{Q}_u(\mathbf{x}) = \mathcal{P}^{(i)}(\mathbf{x}).$$

This equation can be represented as a linear system, where $a_{j,k}$ and $\lambda^{(i)}$ are the unknowns. By evaluating the polynomial at multiple points $\mathbf{x}^{(l)}$ for $l = 1, 2, \ldots, z$, we obtain:

$$\begin{bmatrix} x_{1,1}^{(1)} & x_{1,2}^{(1)} & \cdots & x_{n,n}^{(1)} & \mathcal{Q}_1(\mathbf{x}^{(1)}) & \cdots & \mathcal{Q}_u(\mathbf{x}^{(1)}) \\ x_{1,1}^{(2)} & x_{1,2}^{(2)} & \cdots & x_{n,n}^{(2)} & \mathcal{Q}_1(\mathbf{x}^{(2)}) & \cdots & \mathcal{Q}_u(\mathbf{x}^{(2)}) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{1,1}^{(z)} & x_{1,2}^{(z)} & \cdots & x_{n,n}^{(z)} & \mathcal{Q}_1(\mathbf{x}^{(z)}) & \cdots & \mathcal{Q}_u(\mathbf{x}^{(z)}) \end{bmatrix} \begin{bmatrix} a_{1,1}^{(i)} \\ a_{1,2}^{(i)} \\ \vdots \\ a_{n,n}^{(i)} \\ \lambda_1^{(i)} \\ \vdots \\ \lambda_u^{(i)} \end{bmatrix} = \begin{bmatrix} \mathcal{P}^{(i)}(\mathbf{x}^{(1)}) \\ \mathcal{P}^{(i)}(\mathbf{x}^{(2)}) \\ \vdots \\ \mathcal{P}^{(i)}(\mathbf{x}^{(z)}) \end{bmatrix}. \tag{1}$$

It can be observed that the domain of $\mathbf{x}$ has dimension $\frac{n(n+1)}{2}$. Since $\lambda \cdot \mathcal{Q}(\mathbf{x})$ is a linear combination of $\mathbf{x}$, the matrix in (1) can have at most $\frac{n(n+1)}{2}$ linearly independent rows. Consequently, the final system consists of $\frac{n(n+1)}{2}$ equations in $\frac{n(n+1)}{2} + u$ unknowns. Therefore, the system admits an exponential number of solutions in u, i.e., q^u.

It is important to note that these two attacks take into consideration only a single polynomial $MQ_{n,v}(\mathcal{O}) + \Lambda\mathcal{Q}$. It may be possible to infer additional information from an attack that includes all the structured polynomials. Those attacks are more suitable in the Olivier2 version, where the order of the equations is permuted: an attacker can choose a polynomial and with chance $\frac{n}{u}$ it will be a structured polynomial $MQ_{n,v,q}(\mathcal{O})+\Lambda\mathcal{Q}$. Then, in the resulting solutions, the other structured polynomials will have the coefficient λ equal to 0. This adds further problems to the attacker as

the number of possible solutions grows even more. For example, in the MinRank scenario, there will be an exponential amount of spurious solutions.

To make the encryption algorithm more resistant to this type of attacks, the proposed schema Oliver 2 make use of a random permutation in order to hide if the polynomial has the form $\mathcal{F} \circ \mathcal{T}^{(i)} + \sum_{j=1}^{u} \mathcal{Q}$ or is a random polynomial $\mathcal{Q}$. This can be expressed in another assumption

3.2 Olivier2 Assumption

Olivier2 Problem *For $\mathcal{O} \in \mathbb{F}_q^o$, we let $MQ_{n,v,q}(\mathcal{O})$ denote the set of $\mathcal{P} \in MQ_{n,v,q}$ that vanish on the evaluation of $o \in \mathcal{O}$. The Olivier problem asks to distinguish a random multivariate quadratic map $\mathcal{P} \in MQ_{n,q}$ from $MQ_{n,v,q}(\mathcal{O}) + \Lambda\mathcal{Q}$ and to distinguish the polynomial of the form $MQ_{n,v,q}(\mathcal{O}) + \Lambda\mathcal{Q}$ from the polynomial $\mathcal{Q} \subseteq MQ_{n,u,q}$ for a random $\mathcal{O} \in \mathbb{F}_q^o$, a subset $\mathcal{Q} \subseteq MQ_{n,u,q}$ and the permutation Π_m of the total system of $m = n + u$ polynomials*

Let $\mathcal{A}$ be a Olivier2 distinguisher algorithm. We say that the distinguishing advantage of $\mathcal{A}$ is

$$\mathrm{Adv}^{\mathrm{Olivier}}_{n,v,u,q}(\mathcal{A}) = \left| \Pr[\mathcal{A}(\mathcal{P}) = 1 | \mathcal{P} \leftarrow MQ_{n,q}] - \Pr\left[\mathcal{A}(\mathcal{P}) = 1 \;\middle|\; \begin{array}{c} \mathcal{O} \in \mathbb{F}_q^o \\ MQ_{n,v,q}(\mathcal{Q}) + \Lambda\mathcal{Q} \\ \mathcal{Q} \subseteq MQ_{n,u,q} \\ \Pi_m \end{array} \right] \right|$$

Lemma. Olivier Assumption ⇒ Olivier2 Assumption

Proof. If $\mathcal{A}$ can break the assumption of an Olivier2 system, where random and structured polynomials are mixed together, then it can recover Π. Once the attacker knows Π, the Olivier2 assumption reduces to the original Olivier assumption. □

3.3 CPA-Indistingushability

Theorem. Under the Olivier Assumption, Olivier2 is CPA-secure.

Proof. Define the following experiment.

$\mathcal{D}_{\mathcal{A}}(n, v, u, q, \ell, \mathcal{P})$

1. Give $n, v, u, \ell, q, \mathcal{P}$ to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ outputs two messages m_0 and m_1 of the same length ℓ
3. Generate a random bit $b \xleftarrow{\$} \{0, 1\}$.
4. $r \xleftarrow{\$} \{0, 1\}^{n-\ell}$.
5. Give $\mathcal{P}(m_b||r)$ to the adversary $\mathcal{A}$.
6. The adversary $\mathcal{A}$ outputs a bit b'.
7. Return 1 (**success**) if $b' = b$, and 0 (**failure**) if $b' \neq b$.

Suppose $\mathcal{A}$ is a PPT adversary with non-negligible advantage (1-negl(λ)). Let $\mathcal{D}_{\mathcal{A}}$ be a distinguisher for the Olivier Problem, which takes as input a map $\mathcal{P}$ and must distinguish whether:

1. $\mathcal{P} \in MQ_n, q$
2. $\mathcal{P} \in \{MQ_{n,v,q}(\mathcal{O}) + \Lambda\mathcal{Q}, \mathcal{Q}\}$ where $\mathcal{O} \in \mathbb{F}_q^o, \mathcal{Q} \in MQ_{n,u,q}, \Lambda \in \{0,1\}^{n \times u}$

The distinguisher has control over $\mathbf{m}$, so It can arbitrarily choose the first ℓ variables of the MQ system, and obtain a quadratic system of n equations in $n - \ell$ variables.

Case 1 $\mathcal{P}$ is a random quadratic map and does not follow the structure of the Olivier public key. The ciphertext $c = \mathcal{P}(\mathbf{m}_b \| \mathbf{r})$ is the evaluation of a random MQ map on the input $\mathbf{m}_b \| \mathbf{r}$. Since $\mathbf{r}$ is chosen uniformly at random from $\mathbb{F}_q^{n-\ell}$, the output c is statistically close to uniform over F_q^m. Thus the ciphertext reveals no information about b, and $\mathcal{A}$'s guess is no better then random $\Pr[\mathcal{D}_{\mathcal{A}}(\mathcal{P} \in MQ_n)] = \Pr[b' = b] = \frac{1}{2}$

Case 2 $\mathcal{P}$ is a valid public key of the Olivier cryptosystem. The CPA experiment simulated is identical to the real CPA-security game for Olivier. Thus $\Pr[\mathcal{D}_{\mathcal{A}}(\mathcal{P} \in \{MQ_{n,v,q}(\mathcal{O}) + \Lambda\mathcal{Q}, \mathcal{Q}\}] = \Pr[b' = b] = \frac{1}{2} + \text{negl}(\lambda)$.

$$\text{Adv}^{\text{Olivier}}_{n,v,u,q}(\mathcal{A}) = \left| \frac{1}{2} - \left(\frac{1}{2} + \text{negl}(\lambda) \right) \right| = \text{negl}(\lambda)$$

4 Security Analysis against known attacks

There are two groups of attacks against multivariate public key schemes:

- **Direct attacks** They try to solve the instance of the MQ Problem. There are two standard algorithms, namely the XL algorithm and the F4/F5 algorithm.
- **Structural attacks** They try to exploit the fact that the scheme is not a random instance of the MQ problem, and they try to find the composition $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$

4.1 Direct attacks against Olivier

4.1.1 Gröbner basis computation

A Gröbner basis is a special type of generating set for an ideal in a polynomial ring, constructed so that the structure of the ideal and its associated algebraic variety can be effectively studied. This makes many otherwise difficult algebraic problems algorithmically tractable. In particular, Gröbner bases enable the systematic solution of systems of multivariate polynomial equations by providing a simplified representation of their solution variety.

Gröbner Basis A finite subset $G = \{g^{(1)}, \ldots, g^{(n)}\} \subset I$ is a Gröbner basis of I (with respect to a monomial order) if: $\text{LT}(I) = < \text{LT}(g^{(1)}), \ldots, \text{LT}(g^{(n)}) >$

Elimination Ideal Let $k[x_1, \ldots, x_n]$ be a polynomial ring over a field k, and let $I \subseteq k[x_1, \ldots, x_n]$ be an ideal. Fix a lexicographic order $x_1 \succ x_2 \succ \cdots \succ x_n$. For $0 \leq \ell \leq n$, define the ℓ-th elimination ideal $I_\ell = I \cap k[x_{\ell+1}, \ldots, x_n]$.

If $I = \langle f_1, \ldots, f_2 \rangle$, I_ℓ consists of all consequences of $f_1 = \cdots = f_s = 0$ which eliminate the variables $x_1, \ldots, x_\ell$

Elimination Theorem Let G be a Gröbner basis of I with respect to lex order $x_1 \succ x_2 \succ \cdots \succ x_n$. For each $0 \leq \ell \leq n$, $G_\ell = G \cap k[x_{\ell+1}, \ldots, x_n]$ is a Gröbner basis of the elimination ideal I_ℓ

G_{n-1} is a set of univariate polynomials in x_n. Once a solution for that variable has been found, it can be substituted in G to obtain another set of univariate polynomials, and so on until the set of solutions is found.

F4/F5 Most efficient algorithms known to compute the Gröbner basis of an ideal of a multivariate polynomial ring. [Fau99][Fau02]

F4 is more efficient for non-regular systems, while F5 is more efficient for regular sequences; their time complexity is

$$O\left(\binom{n + D_{\text{reg}}}{D_{\text{reg}}}^{\omega}\right) \tag{2}$$

where ω is the coefficient of matrix multiplication and D_{reg} is the degree of regularity of the polynomial system.

4.1.2 XL Algorithm

The XL Algorithm can be seen as a combination of a bounded degree Gröbner basis and linearization. The basic idea of this technique is to generate from each polynomial equation a large number of higher degree variants by multiplying it with all the possible monomials of some bounded degree, and then linearize the expanded system[Cou+00]

Relinearization The basic idea of relinearization is to add to the given system of linear equations in the y_{ij} additional nonlinear equations which express the fact that these variables are related rather than independent. In its simplest form, relinerization is based on the commutativity of multiplication of 4-tuples of variables: for any a, b, c, d, $(x_a, x_b)(x_c, x_d) = (x_a, x_c)(x_b, x_d) = (x_a, x_d)(x_b, x_c)$ and thus $y_{ab}y_{cd} = y_{ac}y_{bd} = y_{ad}y_{bc}$

Each independent equation obtained by relinearization exists (in a different form) in XL, and thus, XL can be seen as a simplified and improved version of relinearization

1. **Multiply** Generate all the products $\prod_{j=1}^{k} x_i l_i \in \mathcal{I}_D$ with $k \leq D - 2$
2. **Linearize** Consider each monomial in the x_i of degree $\leq D$ as a new variable and perform Gaussian elimination on the equations obtained in 1.

   The ordering of the monomials must be such that all the terms containing one variable (say x_1) are eliminated last.
3. **Solve** Assume that step 2 yields at least one univariate equation in the powers of x_1. Solve this equation over the finite fields (e.g., with Berlekamp's algorithm)
4. **Repeat** Simplify the equations and repeat the process to find the values of the other variables

Gröbner Basis and XL Algorithm One way of implementing the XL algorithm is to combine the equations in an organised way, rather than to multiply them by all the possible monomials. This would lead to the classical Gröbner-bases algorithms

Several variants of the XL Algorithm have been proposed:

- **FXL** Fix the values of some variables before applying the XL-Algorithm. If the values are correctly guessed, it may help find a solution to the system at a lower degree D
- **XL2** Optimized for GF2. It makes use of the field equation $x_i^2 - x_i = 0$ during the elimination process
- **PWXL** Combines XL and parallel Wiederman solver[FK21]

The complexity of the optimized implementation of Block Wiedemann XL by Cheng, Chou, Niederhagen, and Yang[Che+16] of an instance with m random homogeneous equations in n variables can be estimated at the cost of

$$O\left(3\binom{n-1+D}{D}^2\binom{n+1}{2}\right) \tag{3}$$

4.2 Structural attacks against Olivier

Lemma (Kipnis-Shamir cryptanalysis of Oil and Vinegar[KPG99][KS98])[Péb23]
Let G be a UOV public key with parameters n, m, q. Then the following holds:

i. *If $n = 2m$, there exists a probabilistic algorithm performing a key recovery attack against G in time $O(n^\omega)$.*

ii. *If $n > 2m$, there exists a probabilistic algorithm performing a key recovery attack against G in time $O(q^{n-2m}n^\omega)$.*

iii. *If $n < 2m$, there exists a deterministic algorithm performing a key recovery attack against G in time $O(n^\omega)$.*

It comes from this Lemma that the UOV map used by Olivier2 is weak and if an attacker can separate the $\Lambda\mathcal{Q}$ part from $\mathcal{F} \circ \mathcal{T}$, the system is easily broken.

4.2.1 Recovering Lambda

4.2.1.1 MinRank Problem

The MinRank problem is the problem of finding a non-trivial linear combination of K matrices $M_1, \ldots, M_K$, of size $m \times n$, having rank smaller than or equal to a target rank $r \leq \min m, n$. (i.e. find the coefficients $\lambda_i \in \mathbb{Z}_2$ such that $M = \sum_{i=1}^{K} \lambda_i M_i$ has rank $\leq r$

MinRank Attack 1 In Olivier[EFR24] the UOV matrices $\mathcal{F} \circ \mathcal{T}$ have rank less than or equal to $2v$. The system can be modeled as $\mathcal{F} \circ \mathcal{T}^{(i)} = \sum_{j=1}^{u} \lambda_i \mathcal{Q}^{(j)}$ where i is any of the matrix of $\mathcal{G}$. The solution λ_j is the i-th row of the matrix Λ.

This attack has complexity $O(u^3(2v+1)\binom{4v+2}{2v}^2)$ for a single matrix $\mathcal{F} \circ \mathcal{T}^{(i)}$, hence $O(nu^3(2v+1)\binom{4v+2}{2v}^2)$ to recover the whole system $\mathcal{F} \circ \mathcal{T}$.

In Olivier2, with the introduction of a permutation, the attacker cannot apply this attack because he would need to distinguish the random polynomials from the structured ones. The complexity of the attack is then increased by a factor of $\binom{n+u}{u}$.

MinRank Attack 2 If an attacker cannot distinguish the structured polynomials from the random ones, he could perform a MinRank attack on the whole system. The result obtained will be the linear combinations of all $\mathcal{F} \circ \mathcal{T}$ matrices (totaling $q^n - 1$ results). This exponential number of parasitic solutions makes it hard to distinguish which combinations identify the polynomial $\mathcal{F} \circ \mathcal{T}$. It is also important to note that the solutions found are not really parasitic as the combination that gives, for example, $\mathcal{F} \circ \mathcal{T}^{(1)} + \mathcal{F} \circ \mathcal{T}^{(2)}$ will also contain the sum of the coefficients $\lambda^{(1)} + \lambda^{(2)}$. It is not clear yet if it is possible to exploit this information to build an easier system. This attack needs to be explored further.

The complexity of the attack is $O((n+u)^3(2v+1)\binom{4v+2}{2v}^2)$

4.2.1.2 Building a linear system

Given a vector $\mathbf{o} \in \mathcal{O}$, its evaluation will produce a random evaluation on $\mathcal{Q}(\mathbf{o})$ and the linear combination $\lambda^{(i)} + \sum_{j=1}^{u} \mathcal{Q}(\mathbf{o})$ for each structured polynomial.

It seems reasonable to assume that, given a random vector, it is possible to check if the vector is in $\mathcal{O}$ only if some $\lambda^{(i)}$ are known.

In Olivier, if an attacker can recover u vector of $\mathcal{O}$, then he can easily recover Λ.

For each structured polynomial an attacker can build the system 1 exploiting the knowledge of $\sum x_{ij} a_{ij} = 0$

$$\begin{bmatrix} \mathcal{Q}_1(\mathbf{x}^{(1)}) & \cdots & \mathcal{Q}_u(\mathbf{x}^{(1)}) \\ \mathcal{Q}_1(\mathbf{x}^{(2)}) & \cdots & \mathcal{Q}_u(\mathbf{x}^{(2)}) \\ \vdots & & \\ \mathcal{Q}_1(\mathbf{x}^{(z)}) & \cdots & \mathcal{Q}_u(\mathbf{x}^{(z)}) \end{bmatrix} \begin{bmatrix} \lambda_1^{(i)} \\ \vdots \\ \lambda_u^{(i)} \end{bmatrix} = \begin{bmatrix} \mathcal{P}^{(i)}(\mathbf{x}^{(1)}) \\ \mathcal{P}^{(i)}(\mathbf{x}^{(2)}) \\ \vdots \\ \mathcal{P}^{(i)}(\mathbf{x}^{(z)}) \end{bmatrix} \tag{4}$$

The number of solutions of the system is given by q^{u-z} (when $u = z$ the solution is unique).

In Olivier2, the attack is still valid, but it's slightly more complex. An attacker could guess that a polynomial is structured with chance $\frac{n}{n+u}$ and use as vector of unknowns λ' where if the i-th polynomial is $a\mathcal{Q}$ polynomial, then $\lambda'^{(i)} = \lambda^{(i)}$, if it is a structured polynomial then $\lambda'^{(i)} = 0$.

To break a single $\lambda^{(i)}$ it is sufficient that $G(\mathbf{x})^{(i)} = 0$. If an attacker can guess u of those $\mathbf{x}$ then he could recover $\lambda^{(i)}$ for Olivier (or $n+u$ vector for $\lambda'^{(i)}$ for Olivier2). If an attacker guesses randomly, the chance of finding a single vector that satisfies the equation is $\frac{1}{q}$, so an attacker could sample randomly $q \cdot u$ vectors and expect that u of them satisfy the equation. Then he would need to choose exactly the u vector out of the $q \cdot u$. This happens with chance $\binom{q \cdot u}{u}$. Hence, randomly guessing a vector is worse than randomly guessing a coefficient of $\lambda^{(i)}$.

4.2.2 Recovering the Oil Space

It is not known if $\mathcal{O}$ can be recovered from the mixed system, before recovering the matrix Λ

4.3 Security parameters choice

The NIST post-quantum call proposes 5 security levels:

Using the value MAXDEPTH = 2^{40}, Olivier2 satisfies Level 1 of security with parameters derived in the following way:

The security parameters identified are: $q, u, n, v, D_{\text{reg}}, D_{\text{XL}}$

AES 128	2^{170} MAXDEPTH quantum gates or 2^{143} classical gates
SHA3-256	2^{146} classical gates
AES 192	2^{233} MAXDEPTH quantum gates or 2^{207} classical gates
SHA3-384	2^{210} classical gates
AES 256	2^{298} MAXDEPTH quantum gates or 2^{272} classical gates
SHA3-512	2^{274} classical gates

Table 1: NIST level of security

Choice of q The choice of field is a primary parameter, upon which others are selected, as the security of the system comes for the most part from those other parameters. The field is typically selected with efficiency considerations in mind, as it largely determines the overall system size. It is often chosen to be a power of two, as this configuration offers greater efficiency for bit-level operations on digital hardware. As an example, $q = 2$ is chosen.

Choice of n The CPA-IND and CCA2-IND experiment showed that the after the adversary choose the variables ℓ, he can obtain a reduced system of m equations in $n - \ell$ variables. The resulting system is where the security parameters are chosen to satisfy the previous security goals. Note that to send b bits of information, set $\ell = \frac{b}{\log_2 q}$ and derive the security from $n - \ell$, the choice is done by estimating the degree of regularity of the system and the D_{XL}

Choice of u The choice of u mostly impacts the hardness of recovering the map Λ and recovering the weak UOV polynomials. It is also important that no vector is evaluated to 0 on $\mathcal{Q}$, and this happens with $(q^n - 1)q^{-m}$ probability. From the attack identified from the construction of a linear system 1, 2^u is the number of possible solutions of the system; this value can be iterated over using enumeration. This attack in the post quantum model is improved by the Grover algorithms, reducing the security to $2^{u/2}$. To obtain a security of 2^{130} quantum operations, set $u = 260$.

Degree of regularity It is standard to estimate D_{reg} with the assumption that the system is semi-regular. D_{reg} is the first d such that the coefficient of t^d in $\frac{(1-t^2)^m}{(1-t)^{n+1}}$ is non-positive. It is important to notice that this estimation works better with higher values of q, and for $q = 2$, it's likely an underestimation.

For $q = 2, \ell = 128, n = 293, m = 553\omega = 2.376$ the formula 2 gives a 143-bit security (substituting n with $n - \ell$).

Degree of the algorithm XL To estimate D_{XL} it has been used the program written by *Cheng et al.*[Che+16]

For $q = 2, \ell = 128, n = 331, m = 591$ the algorithms returns $D_{\mathrm{XL}} = 13$ and formula 3 gives 151-bit security (substituting n with $n - \ell$).

Choice of v The number of vinegar is the most critical aspect of the schema. These parameters cannot be too big, as the efficiency of the decryption function depends exponentially on v. Moreover, even tho it's clear that it is a key parameter in the security of the system, no significant attack has been identified. As previously stated, it is not clear if the equations derived from the MinRank2 Attack might be exploited to create a more vulnerable system. A system that has $v = 15$ will achieve a 149-bit security on this attack.

Similarly, the security parameters for other values of q have been identified for a key exchange

or message of 128 bit:

q	n	u	ℓ
2	331	260	128
4	202	130	64
8	162	87	43
16	140	65	32

5 Construction from Olivier2 PKE

Using Oliver2 Public Key Encryption as a foundational building block, several cryptographic algorithms can be constructed from it.

5.1 Key Encapsulation Mechanism

Using the Fujisaki–Okamoto transform [FO99], a CPA-secure public-key encryption scheme can be transformed into a CCA2-secure key encapsulation mechanism (KEM) in the random oracle model.

Algorithm 1 Fujisaki-Okamoto KEM from PKE

procedure KeyGen
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}()$
    return $(\mathsf{pk}, \mathsf{sk})$
end procedure

procedure Encaps(pk)
    $(k \| r) \xleftarrow{\$} \{0,1\}^{\ell_k + \ell_r}$ ▷ Sample random key material
    $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, k \| r)$
    $K \leftarrow \mathsf{KDF}(k \| r \| c)$
    return (c, K)
end procedure

procedure Decaps(sk, c)
    $k \| r \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, c)$
    $K \leftarrow \mathsf{KDF}(k \| r \| c)$
    return K
end procedure

5.2 Key Exchange

The Olivier decryption iterates over all possible vinegar values, so on average, a solution is expected to be found after $\frac{q^v}{2}$ tries. This implies a significant amount of computational effort is wasted. However, instead of performing decryption on one ciphertext at a time, the scheme can be made more efficient: after fixing a vinegar value, it can check whether any ciphertext in a batch corresponds to that vinegar. In practice, multiple ciphertexts can be arranged in matrix form, allowing operations to be performed simultaneously on the matrix. In this setting, the average number of vinegar

iterations is given by $\frac{q^v}{2^k}$, where k is the number of ciphertexts sent. It is important to note that increasing k also increases both computational complexity and communication cost, so k must be chosen carefully.

Since Bob recovers only one of the keys Alice sent, the parties must still agree on which key was selected. One approach is for Bob to communicate the chosen ciphertext, or he could send the hash of the randomness r used during encryption. Alternatively, both parties may accept that Bob obtains one of the keys from a known subset. Concretely, Bob can initialize a symmetric channel with the recovered key and send an authenticated message. Alice can then attempt to decrypt this message using each candidate key until one succeeds.

Lemma. The Olivier2 KE constructed using the KEM derived from the Fujisaki-Okamoto is IND-ME-CCA2.

Proof. A KEM realized with the Fujisaki-Okamoto transform is also IND-ME-CCA2. These results come from the security game in which the proof is modeled.[FO99] □

6 System Design

To implement Olivier2, some remarks on security and optimization need to be discussed.

6.1 Helper functions

Vinegar Iteration Exhaustively iterating over all vinegar values in a fixed order must be avoided, as it would expose the scheme to timing attacks. In such an attack, a passive adversary could infer information about the vinegar value of the encrypted message based on the decryption time. This concern is even more critical in the Multiple Encryption setting, where the first ciphertext to decrypt would always correspond to the same order, introducing a bias in key establishment. To mitigate this, the iteration process should be randomized, with a fresh randomization applied for each invocation of the decryption function. For efficiency reasons, the value returned is incremented by 1 to avoid generating the value 0, which corresponds to the vinegar zero vector. This solution is skipped as the linear system obtained is a trivial homogeneous system with q^o possible solutions. This means that a message cannot be recovered with chance $\frac{1}{q^v}$.

The proposed approach is both fast and efficient. As an alternative, one might employ a Linear Feedback Shift Register (LFSR), which could offer improved performance in hardware implementations..

Algorithm 2 ScrambleIndex

Input: h integer that represent the previous vinegar value
Input: rnd random integer
Output: vg integer that represent the next vinegar value

1: b $\leftarrow q^v$
2: a = 59 ▷ This value can be chosen arbitrarily must be coprime with v
3: **return** $(a \cdot h + rnd) \pmod{b} + 1$

Obtaining the linear system Fixing the vinegar values transforms the system into a linear one. The quadratic equations, originally represented in matrix form, reduce to linear equations that can be expressed as vectors. These vectors are then assembled into a matrix, on which Gaussian elimination can be applied to solve the system. When the vinegar values are fixed, terms of the form x_i or $x_i * x_j$ where $i = j \neq 1$ are vinegar values, evaluate to constants. These constants must be subtracted from the system's evaluation to correctly account for their contribution.

Algorithm 3 FixVinegars

Input: F_{matrices} in upper triangular form
Input: vg the current vinegar value
Output: My a matrix representing the linear system obtained while fixing the vinegar values
Output: The constant term arised when fixing the vinegar values

$M_y \leftarrow 0_{n,o}$
constant $\leftarrow 0_n$
for $j = 0$ to $n - 1$ **do** do
    sum $\leftarrow 0$
    for $i = 0$ to $v - 1$ **do** do
        $\text{temp}_i \leftarrow 0$
        for $k = i$ to $v - 1$ **do** do
            $\text{temp}_i \leftarrow F_{\text{matrices}}[j][i, k] \cdot \text{vg}[k]$ ▷ When fixing v the terms $v_i \cdot v_j$ will result in a constant term for each polynomial
        end for
        sum $\leftarrow$ sum + $\text{vg}[i] \cdot \text{temp}_i$
    end for
    constant $[j] \leftarrow sum$
end for
for $j = 0$ to $n - 1$ **do** do
    $A \leftarrow 0_{o,v}$
    for $i = 0$ to $n - v - 1$ **do** do
        for $k = 0$ to $v - 1$ **do** do
            bit1 $\leftarrow F_{\text{matrices}}[j][v + i, k]$
            bit2 $\leftarrow F_{\text{matrices}}[j][k, v + i]$
            $A[i, k] \leftarrow$ bit1 + bit2
        end for
    end for
    for $i = 0$ to $n - v - 1$ **do** do
        sum $\leftarrow 0$
        for $k = 0$ to v **do** do
            sum $=\leftarrow$ sum + $A[i, k] \cdot \text{vg}[k]$ ▷ The coefficient of a variable is multiplied for the coefficient of the fixed vinegar value
        end for
        $M_y[j, i] \leftarrow$ sum
    end for
end for
return M_y, constant

Computing EF By reducing a matrix to its Row Echelon Form with leading ones, the complete solution set of the system can be obtained directly.

Algorithm 4 EF

Input: B matrix
Output: EF of B with leading ones

```
pivot_row ← 0
pivot_col ← 0
while pivot_row < n and pivot_col < m do do
    next_pivot_row ← pivot_row
    while next_pivot_row < n and B[next_pivot_row, pivot_col] == 0 do do
        next_pivot_row ← next_pivot_row + 1
    end while
    if next_pivot_row == n then
        pivot_col ← pivot_col + 1                ▷ Move to next column if there is no pivot
        continue
    end if
    if next_pivot_row != pivot_row then
        Swap(B[pivot_row], B[next_pivot_row])
    end if
    B[pivot_row] = B[pivot_row]· inverse(B[pivot_row, pivot_col])        ▷ Scale the pivot row
    for row = 0 to n − 1 do do                                           ▷ Eliminate other rows
        if row != pivot_row and B[row, pivot_col] != 0 then do
            factor ← B[row, pivot_col]
            B[row] ← B[row] - factor · B[pivot_row]
        end if
    end for
    pivot_row ← pivot_row + 1
    pivot_col ← pivot_col + 1
end while
return B
```

Extract Free Indices from the EF

Algorithm 5 ExtractFreeIndices

Input: M – matrix in echelon form (EF) with r rows and c columns
Output: Array of free (non-pivot) column indices

pivot_cols ← [ ]
for $i = 0$ to $\min(r, c) - 1$ **do**
    found ← **False**
    for $j = 0$ to $c - 1$ **do** ▷ If there is no column with only one pivot, then that column is a free index
        if $M[i, j] \neq 0$ **then**
            pivot_cols.append(j)
            found ← **True**
            break
        end if
    end for
    if ¬found **then**
        pivot_cols.append(-1)
    end if
end for
pivot_cols_set ← $\{\text{col} \in \text{pivot\_cols} : \text{col} \geq 0\}$
free_indices ← $\{j : 0 \leq j < c \text{ and } j \notin \text{pivot\_cols\_set}\}$
return free_indices

Extract all solutions from the EF

Algorithm 6 ExtractAllSolutions

Input: M augmented matrix in echelon form, size $r \times (c+1)$
Input: pivot_cols – array of pivot column indices (or -1 for zero rows)
Input: free_indices – array of free variable indices
Output: Array of all possible solutions to the system

```
solutions ← ∅
k ← |free_indices|
for each v ∈ {0, 1, ..., q − 1}^k do                ▷ Iterate over all q^k assignments
    x ← 0_c                                          ▷ Initialize solution vector
                                                     ▷ Assign free variables
    for i = 0 to k − 1 do
        x[free_indices[i]] ← v[i]
    end for
                                                     ▷ Back-substitute to find pivot variables
    for row = 0 to min(r, |pivot_cols|) − 1 do
        if pivot_cols[row] ≥ 0 then
            pivot_idx ← pivot_cols[row]
            s ← 0
            for each j ∈ free_indices do
                s ← s + M[row, j] · x[j]
            end for
            if M[row, pivot_idx] ≠ 0 then
                x[pivot_idx] ← (M[row, c] − s)/M[row, pivot_idx]
            else
                x[pivot_idx] ← 0
            end if
        end if
    end for
    solutions ← solutions ∪ {x}
end for
return solutions
```

Excluding some solutions After fixing the vinegars the linear system to be solved is given by $M_{\bar{\mathbf{y}}} \begin{pmatrix} y_{v+1} \\ \vdots \\ y_n \end{pmatrix} = W_{\bar{\mathbf{y}}}$, were $W_{\bar{\mathbf{y}}} = (W - \text{constant})$ is given by the evaluation of $\mathcal{F} \circ \mathcal{T}(\mathbf{x})$. Instead of solving a system for each vector in W, it is possible to extract a subset of possible solutions. The $M_{\bar{\mathbf{y}}}$ matrix has dimension $n \times o$, hence there are at least v dependent vector. These dependencies must also hold in the $W_{\bar{\mathbf{y}}}$ matrix. This can be verified by building a $\tilde{W}_{\bar{\mathbf{y}}}$ matrix.

$$\tilde{W}^{(i)}_{\bar{\mathbf{y}}} = \begin{cases} W^{(i)}_{\bar{\mathbf{y}}} & \text{If } i \text{ is an independent index of } M_{\bar{\mathbf{y}}} \\ \sum_{j=1} d_j W^{(j)}_{\bar{\mathbf{y}}} - W^{(i)}_{\bar{\mathbf{y}}} & \text{If } i \text{ is a dependent index of } M_{\bar{\mathbf{y}}} \text{ where } d_j \text{ are the coefficients of the dependencies} \end{cases}$$

If a row of $M_{\bar{\mathbf{y}}}$ is the zero row, set the corresponding row of $\tilde{W}_{\bar{\mathbf{y}}}$ to 0.

Algorithm 7 FindIndependentRows

Input: B matrix
Output: Array of independent indices

```
original_indices ← [0, 1, ..., n − 1]
pivot_indices ← []
⋮                                   ▷ Apply Algorithm 6.1 (EF) on B and track original indexes
if next_pivot_row != pivot_row then
    Swap(B[pivot_row], B[next_pivot_row])
    Swap(original_indices[pivot_row], original_indices[next_pivot_row])
end if
pivot_indices[rank] ← original_indices[pivot_row]
⋮
return pivot_indices
```

Algorithm 8 FindDependencies

Input: B matrix in EF
Input: independent_indices – array of independent indices
Output: Dictionary of dependencies and coefficients for each dependent row

dependent_indices $\leftarrow \{i : i \notin \text{independent\_indices}\}$
dependencies $\leftarrow \{\}$
for each dep_idx in dependent_indices **do**
    $b \leftarrow$ matrix[dep_idx].copy()
    $A_{\text{ind}} \leftarrow$ matrix[independent_indices]
    $k \leftarrow |\text{independent\_indices}|$
    augmented $\leftarrow [A_{\text{ind}}^T \mid b]$ ▷ Form augmented matrix
    pivot_positions $\leftarrow$ FindIndependentPositions(augmented) ▷ Like 6.1 but at the end of every while iteration save pivot_row and pivot_col
    consistent $\leftarrow$ **True**
    for row $= 0$ to $m - 1$ **do**
        if row has no pivot in first k columns **and** rref[row, k] $\neq 0$ **then**
            consistent $\leftarrow$ **False**
            break
        end if
    end for
    if consistent **then**
        Extract coefficients from rref[:, k] at pivot positions
        dependencies[dep_idx] $\leftarrow$ non-zero coefficient pairs
    else
        dependencies[dep_idx] $\leftarrow$ [ ]
    end if
end for
return dependencies

6.2 Key Generation

Algorithm 9 KeyGen()

Output: Private key components $\mathcal{F}, \mathcal{S}^{-1}, \mathcal{Q}, \Lambda$, perm_indices
Output: Public key $\mathcal{P}$

$F_{\text{Matrices}} \leftarrow []$
for $i = 1$ to n **do**
    Generate a $OV_{\text{n, v}}$ polynomial
    F_{Matrices}.append(polynomial)
end for
Generate a random matrix S and its inverse S^{-1}
$G_{\text{Matrices}} = S^T \cdot F_{\text{Matrices}} \cdot S$
$Q_{\text{Matrices}} \leftarrow []$
for $i = 1$ to u **do**
    Generate a random polynomial
    Q_{Matrices}.append(polynomial)
end for
$P_{\text{Matrices}} \leftarrow []$
$P_{\text{Matrices,append}}(G_{\text{Matrices}} + \Lambda Q_{\text{Matrices}})$
$P_{\text{Matrices.append}}(Q_{\text{Matrices}})$
perm_indices $\leftarrow$ RandomPermutation(m)
$P_{\text{matrices}} \leftarrow \text{Shuffle}(P_{\text{matrices}}[i], \text{perm\_indices})$
return sk $= (F_{\text{matrices}}, S^{-1}, Q_{\text{matrices}}, \Lambda, \text{perm\_indices})$, pk $= P_{\text{matrices}}$

6.3 Encaps

Algorithm 10 Encaps(pk)

Input: Public key $\mathcal{P}$ **Output:** Shared key K **Output:** Ciphertext c

$(k \| r) \xleftarrow{\$} \{0,1\}^{\ell_k + \ell_r}$ ▷ Sample random key material
msg $\leftarrow$ ConstructVector($k \| r$, n) ▷ Embed $k \| r$ into message space
$c \leftarrow \text{msg} \cdot P_{\text{matrices}} \cdot \text{msg}^T$
$K \leftarrow \mathsf{KDF}(k \| r \| c)$ ▷ Derive final shared key
return K, c

To accelerate the encapsulation of multiple ciphertexts, one can exploit the properties of the quadratic form to extract the diagonal of the computation

$$M[i,:] \cdot P_{\text{matrices}}[i] \cdot M[i,:]^T,$$

where M denotes the matrix of messages.

6.4 Decaps

Important remarks about the decryption algorithm:

- In systems that allows parallel computing, the iteration of the vinegar and the consequent operations can be done in parallel
- At line 21, the algorithm checks whether the number of free variables in the solved system exceeds a given threshold `exit`. This check is necessary because the number of possible solutions grows exponentially with the number of free variables, following $q^{|\text{free\_variables}|}$. If, for a particular system, this value becomes too large, the algorithm would remain stuck in the loop.

Algorithm 11 Decaps(sk, c)

Input: The secret key sk **Input:** The encrypted message c **Output:** Shared key K

inv_indices = ArgSort(perm_indices)
Reordered_c = Shuffle(c, inv_indices)
W = reordered_c[: n] + ($\Lambda \cdot$ reordered_c[n :]^T)^T
total = $q^v - 1$
for $h = 1$ to *total* **do**
    j = ScrambleIndex(k, v, rnd)
    vg = FieldConversion(j)
    My, term = FixVg(F_{matrices}, vg)
    W_{vg} = $W-$ term
    ind = FindIndependentRows(My)
    dep = FindDependencies(My, ind)
    $\tilde{W}_{\text{vg}}$ = copy(W_{vg})
    for $i \in$ dep **do**
        $\tilde{W}_{\text{vg}}[i] = \sum W_{\text{vg}}[\text{dep}.i] \cdot \text{dep}.i.\text{coeff}$
    end for
    if all($\tilde{W}$[dep]) == 0 **then**
        Ay = RREF(M[ind]$\|W_{\text{vg}}$[ind])
        rank = Rank(M[ind])
        rank_augmented = Rank(Ay)
        free_vars = o - rank
        if free_vars > exit **then**
            continue
        end if
        if rank $\neq$ rank_augmented **then**
            continue
        end if
        pivot_cols = ExtractPivotColumns(Ay, rank, o)
        free_indexes = ExtractFreeIndices(Ay, rank, o)
        solutions = ExtractAllSolutions(Ay, pivot_cols, free_indexes, q)
        for each oils $\in$ solutions **do**
            $y \leftarrow$ vg $\|$ oils
            $x \leftarrow S^{-1} \cdot y$
            evaluation $\leftarrow 0_u$
            for $i = 0$ **to** $2n - 1$ **do**
                evaluation[i] = $x \cdot Q_{\text{Matrices}} \cdot x^T$
            end for
            if evaluation == reordered_ciphertext[$n..2n - 1$] **then**
                $(k\|r) \leftarrow$ ExtractKeyMaterial(x, $\ell_k + \ell_r$) ▷ Extract from msg
                $c' \leftarrow \text{msg} \cdot P_{\text{matrices}} \cdot \text{msg}^T$ ▷ Re-encrypt
                if $c' == c$ **then** ▷ Implicit rejection check
                    $K \leftarrow \mathsf{KDF}(k\|r\|c)$ ▷ Derive shared key
                    return K
                end if
            end if
        end for
    end if
end for
return $\perp$ ▷ Decapsulation failure / implicit rejection

Algorithm 12 MultipleDecaps(sk, c)

Input: The secret key sk
Input: An array of encrypted messages C
Output: The first decrypted messages of C

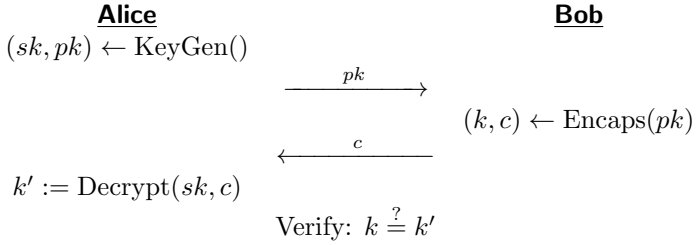
$\vdots$
$W = \text{reordered\_c}[:, : n] + (\Lambda \cdots \text{reordered\_c}[:, n :]^T)^T$ ▷ W is a Matrix
for $h = 1$ to *total* **do**
  for $i \in$ dep **do**
    $\tilde{W}_{\text{vg}}[:, i] = \sum \tilde{W}_{\text{vg}}[:, \text{dep}.i] \cdot \text{dep}.i.\text{coeff}$ ▷ $\tilde{W}_{\text{vg}}$ is a Matrix
  end for
  compatible_columns $\leftarrow k \in \tilde{W}_{\text{vg}}$ such that $\tilde{W}_{\text{vg}}$[k, j] == 0 ▷ Check the columns of the Matrix
  if |compatible_columns| == 0 **then** break
  end if
  for k in compatible_columns **do**
    $\vdots$ ▷ Solve the linear system
    return x ▷ Return the first solution
  end for
end for
return $\perp$ ▷ Decapsulation failure / implicit rejection

6.5 Olivier2

Algorithm 13 KEM

Output: True if the key message recovered is the same, false otherwise

sk, pk $\leftarrow$ KeyGen()
$k, c \leftarrow$ Encaps(pk, msg)
$k' :=$ Decrypt(sk, c)
return $k == k'$

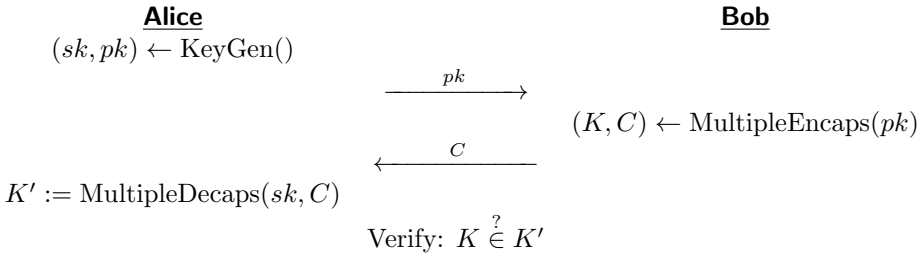


Algorithm 14 KE

Input: A list of messages M

Output: True if the messages recovered are the same, false otherwise

1: sk, pk ← KeyGen()
2: $K, C \leftarrow$ MultipleEncaps(pk)
3: $K' :=$ MultipleDecryptKE(sk, C)
4: **return** $K \in (K')$



7 Benchmark

The system has been implemented on a 2019 MacBook Pro equipped with an Intel Core i5 1.4 GHz Quad-Core processor. The implementation was carried out in the C programming language, chosen for its efficiency. The library used for computations over GF(2) is [AB25], while the library used for computations over GF(2^e) is [Alb25]. Those libraries implement fast arithmetic with dense matrices over their given field.

7.1 Execution time

The goal is to find a value for v that allows the algorithm to run within an acceptable time. This implementation has not been optimized for the target computer architecture, and the code could be made more efficient. The time target has been arbitrarily set at 200 ms. For reference, an optimized implementation of the Classic McEliece cryptosystem[Ber+22] with NIST level-1 security parameters achieves approximately 22 ms for the combined execution of the Encaps and Decaps functions on the underlying machine. Furthermore, the Decryption process can be parallelized; this has been mentioned in the "Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process" document by NIST as a positive feature. An implementation that uses parallelization could improve the results obtained.

7.1.1 GF(2)

The safe parameters for GF2 identified in Table 4.3 are $n = 331$ and $u = 260$.

7.1.1.1 KEM

The vinegar parameter obtained is 5. As noted in the System Design section under the Vinegar Iteration function, one out of every 32 keys cannot be recovered. This makes the KEM over GF(2)

unsuitable for the Olivier2 public key encryption.

7.1.1.2 KE

The vinegar parameter obtained is 10, together with $t = 256$. In Key Exchange, the ciphertexts sent are t, unlike in KEM, where only a single ciphertext is sent. Therefore, for a key to be unrecoverable, all vectors must be unrecoverable. For these parameters, this occurs with probability $(\frac{1}{2^{10}})^{256}$, which is negligible.

It has also been noted that as t increases, the complexity of Encaps also increases, making it even slower than MultipleDecaps.

7.1.2 GF(2^e)

7.1.2.1 KEM

The same problem identified for GF2 is also present in GF(2^e) for each parameter set, concluding that the KEM construction is not suitable.

7.1.2.2 KE

The vinegar and t values identified for each q are reported in the following table:

q	v	t
4	3	96
8	2	128
16	1	32

Operations in GF(2^e) are significantly more expensive than in GF2, and the Encaps algorithm reaches the execution time of 200ms with a much lower t compared to GF2.

7.2 Key sizes and Communication Cost

With the exception of the permutation array, all the other structures are represented by matrices. Each element of a matrix can be represented in multiple bits, according to the formula $\log_2 q$ bits.

Secret Key. The secret key consists of the following components:

- $\mathcal{F}$: OV matrices, represented as $n \times n$ upper triangular matrices of the form

$$f = \begin{bmatrix} v \times v & v \times o \\ 0_{o \times v} & 0_{o \times o} \end{bmatrix}.$$

- $\mathcal{T}^{-1}$: a full $n \times n$ matrix.
- $\mathcal{Q}$: u upper triangular $n \times n$ matrices.
- Λ: an $n \times u$ matrix.
- Π: a permutation, packed using $n \cdot \lceil \log_2(n) \rceil$ bits.

The sizes of these components in bits are given by:

$$\begin{aligned}
|\mathcal{F}| &= n(\frac{v(v+1}{2} + vo)\log_2(q), \\
|\mathcal{T}^{-1}| &= n^2 \log_2(q), \\
|\mathcal{Q}| &= u \cdot \frac{n(n+1)}{2}\log_2(q), \\
|\Lambda| &= nu \log_2(q), \\
|\Pi| &= n \cdot \lceil \log_2(n) \rceil.
\end{aligned}$$

For the four proposed parameter sets, the corresponding expanded secret key sizes, in kilobytes, are 5281KB, 2700KB, 2003KB, and 1668KB, respectively.

The matrices $\mathcal{F}$, $\mathcal{Q}$, and Λ can be generated by expanding a random seed, significantly reducing the overall key size. For a security level 1 target according to NIST, the seed length is 192 bits.

For the four proposed parameter sets, the corresponding compact secret key sizes, in bytes, are 14092B, 10427B, 10028B, and 9964B, respectively.

Public Key. The public key consists of the multivariate system $\mathcal{P}$, which can be represented by $n + u$ upper triangular $n \times n$ matrices.

The size of the public key in bytes is

$$|\mathcal{P}| = \frac{(n+u) \cdot \frac{n(n+1)}{2}\log_2(q)}{8}.$$

For the four proposed parameter sets, the corresponding secret key sizes, in kilobytes, are 3964KB, 1662KB, 1204KB, and 988KB, respectively.

As shown in UOVsig[NIS25] and MAYO[Beu+23], part of the public key of a UOV system can be expanded by a seed; however, this approach is not possible in this construction, as the public map is not fully UOV, but it's mixed with random polynomials. Consequently, the key size cannot be reduced.

Communcation costs

The transmission cost of the key exchange is determined by the t evaluations of the public key sent by Bob. Each evaluation consists of $n + u$ elements of a vector. The total size in bits is given by

$$t(n+u)\log_2(q).$$

For the four proposed parameter sets, the corresponding communication costs, in bytes, are 18912B, 7968B, 11952B, and 3280B, respectively.

q	csk	esk	pk	comm
2	14,092	5,407,707	4,059,135	18,912
4	10,092	2,764,320	1,701,749	7,968
8	10,028	2,050,202	1,232,831	11,952
16	9,964	1,707,265	1,011,675	3,280

7.3 Final considerations

The Olivier2 cryptosystem presents different problems.

During the implementation of the KEM, it was observed that the failure rate of the Decaps algorithm is excessively high. This occurs when the vinegar vector takes the value of the zero vector. There is no practical way to sample vectors that completely avoid this case; therefore, the KEM construction cannot be realized.

This issue does not arise in the KE construction, as it is sufficient for at least one of the vectors to yield a valid solution. However, the KE construction introduces a drawback that the KEM does not have: encrypting multiple messages becomes increasingly inefficient for large values of t, making it the main bottleneck of the system.

Although the attack on the system that relied on the parameter v has been mitigated by introducing the permutation, it is likely that the value of v chosen to maintain efficiency is too small. As a result, the bias in the structured polynomials may make them distinguishable from truly random ones.

Part of the system's inefficiency arises from the additional randomness required to satisfy the IND-CPA and IND-CCA2 security notions. For $q = 2$ and $q = 4$, the final parameter sets achieve a security level of 271 bits against direct attacks. If the scheme could be modified to make the encryption algorithm non-deterministic while using a smaller value of n, the overall efficiency of the scheme could be improved.

Another major source of inefficiency in the system arises from the large key sizes, particularly in the public key, which cannot be expanded from a seed. The communication cost is not excessively high for small values of t, but it remains significantly worse than in other post-quantum proposals. Furthermore, even if a reduced system with a higher value of v could be obtained to make `MultipleDecaps` more efficient, the parameter t would still need to be increased further.

In conclusion, even if a reduced system satisfying the IND-CPA and IND-CCA2 security notions could be achieved, its performance would still not improve enough to justify adopting this algorithm over other proposals submitted to NIST. Structural limitations, such as the limited number of vinegar variables and the need to compute multiple ciphertexts, remain significant obstacles. Overcoming them would require a fundamental redesign of the underlying structure.

8 Breaking the Olivier cryptosystem

As mentioned in Chapter 4, there is no currently known attack to recover the oil space from this structure; however, after implementing the cryptosystem, one attack becomes obvious.

It has been noted while using the KEM that the algorithm Decrypt fails for $\frac{1}{q^v}$ input. Those inputs are the ones where the vinegar value is the zero vector.

Theorem. If the DECAPS algorithm is unable to produce a solution and halts, then the plaintext corresponding to that ciphertext has a vinegar value of the $\mathcal{F} \circ \mathcal{T}$ map equal to the zero vector.

Proof. The vinegar vector of the $\mathcal{F}$ maps that cause the algorithm to halt is the zero vector. This can be observed from the decryption process: when the vinegar component of $\mathcal{F}$ to be iterated is the zero vector, the resulting system becomes trivial and thus unsolvable. Let $\mathcal{O}$ denote the oil space on which the polynomial $\mathcal{F} \circ \mathcal{T}$ vanishes. It follows that $\mathcal{O}' = \mathcal{T}(\mathcal{O})$ [Beu20]. Applying $\mathcal{T}^{-1}$

to the zero vector $[0, 0, \ldots, 0] \in \mathcal{O}'$ yields the corresponding $[0, 0, \ldots, 0] \in \mathcal{O}$. Let $\mathbf{x}$ be a vector such that the KEM fails to return a valid result. From the polar decomposition $\mathbf{x} = \mathbf{v} + \mathbf{o}$, and since $\mathbf{v} = [0, 0, \ldots, 0]$, it follows that $\mathbf{o} = \mathbf{x}$, and therefore $\mathbf{x} \in \mathcal{O}$. □

This can be exploited in the key-encapsulation (KE) construction by sending a single vector ($t = 1$). If MultipleDecaps fails, the vector lies in the oil space. (Decapsulation may also fail when the linear system has too many free variables; however, that event has low probability and can be neglected.)

The attack is not prevented by forbidding the choice $t = 1$, since an attacker can query MultipleDecaps with $t > 1$ repeatedly using the same ciphertexts. A vector that belongs to the oil space will never produce the exchanged key.

Once u vectors of the oil space have been recovered, the attacks described in Section 4 can be applied: the Λ-map can be recovered completely, yielding weak OV maps that can be inverted efficiently using the Kipnis–Shamir attack. [KS98]

If $t = 1$ is an admissible parameter for the MultipleDecaps algorithm, the expected number of calls required to find a single vector in the oil space is q^v. Hence, recovering u such vectors requires on the order of uq^v calls. Although this complexity is exponential, the holder of the secret key who runs the decryption algorithm faces essentially the same exponential cost to solve the underlying linear system; therefore, the attack is feasible in practice under the same asymptotic constraints.

If $t > 1$, the attacker's workload increases only slightly, because the vinegar iteration at decryption must be performed in a randomized manner. The interaction can be modelled as follows: the attacker (Eve) sends t candidate values to Alice and Alice returns a uniformly random one. After N independent repetitions, the probability that a particular value has never been returned by Alice is $(1 - 1/t)^N$. Thus, to achieve confidence $1 - \delta$ that a value is never returned because it lies in the oil space, one can choose

$$N \;\geq\; \frac{\ln \delta}{\ln(1 - 1/t)} \;\approx\; t \ln \frac{1}{\delta},$$

So the number of required repetitions scales linearly with t (up to logarithmic factors in the desired confidence).

To determine the expected number of picks N required to see all allowed messages present in a sample of size t, we model the process as the *coupon collector problem*.

Define:

- Domain size: q^n
- Number of forbidden messages: q^o (values in the oil space)
- Number of allowed messages: $A = q^n - q^o$ (values not in the oil space)
- Number of messages Eve sends each trial: t
- Number of trials: N

In a sample of size t, the expected number of allowed messages is

$$A_s = t \cdot \frac{A}{q^n} = t \left(1 - \frac{q^o}{q^n}\right).$$

The expected number of picks to see all A_s allowed messages in the sample is

$$\mathbb{E}[N_{\text{picks}}] \approx A_s \cdot H_{A_s} \approx A_s(\ln A_s + \gamma),$$

where H_{A_s} is the A_s-th harmonic number and $\gamma \approx 0.5772$ is the Euler–Mascheroni constant.

To recover u vectors, the expected number of queries required is $\frac{u}{t \cdot \frac{q^o}{q^n}} \mathbb{E}[N_{\text{picks}}]$. Simplifying the formula and substituting $\frac{q^o}{q^n}$ with q^{-v}, the complexity becomes

$$u \cdot q^v \left(1 - q^{-v}\right) \cdot \left(\ln\left(t\left(1 - q^{-v}\right)\right) + \gamma\right) .$$

The attack complexity is again dominated by q^v, which broke the system.

It is also worth noting that the efficiency can be further improved by replacing Eve's returned message with a newly sampled one. For instance, once Eve has collected $t-1$ vectors of the oil space, to obtain the next one, is it sufficient that she send all of the $t-1$ vectors, together with a randomly sampled one, and then she is in the same scenario as $t = 1$.

Example with the previously identified parameters:

- $n = 311$, $o = 306$, $q = 2$, $t = 256$
- Domain size: $q^n = 2^{311}$
- Number of forbidden messages: $q^o = 2^{306}$
- Total allowed messages: $A = q^n - q^o$
- Messages to be recovered: $u = 260$

The expected number of allowed messages in the sample is given by

$$A_s = t\left(1 - \frac{q^o}{q^n}\right) = 256\left(1 - \frac{2^{306}}{2^{311}}\right) = 256 \cdot \frac{31}{32} \approx 248$$

The expected number of picks to see all allowed messages in the sample

$$\mathbb{E}[N_{\text{picks}}] \approx A_s(\ln A_s + \gamma) \approx 248 \cdot (\ln 248 + 0.5772) \approx 248 \cdot 6.09 \approx 1509$$

Using the final formula, the estimated number of queries required to break these parameters is $60264 \approx 2^{15}$.

The attack completely breaks the Olivier system. The Olivier2 variant is also broken: the attack incurs only a polynomially larger cost. The number of vectors that must be recovered increases to $n + u$, and the probability of correctly guessing whether a given polynomial is structured (rather than random) is $\frac{n}{n+u}$.

9 Conclusion

After analyzing the scheme, it becomes clear that the Oil-and-Vinegar trapdoor is unsuitable for encryption: the case $\mathbf{v} = \mathbf{0}$ cannot be avoided. This forced abort leaks critical information that an adversary can exploit to break the system. Although the attack's complexity grows exponentially in v, so the scheme is theoretically secure (for large v the probability of randomly sampling a vinegar vector equal to $\mathbf{0}$ is negligible), the decryption trapdoor likewise runs in time exponential in v, with the same dominating parameter q^v. Consequently, the scheme is insecure in practice.

References

[AB25] Martin Albrecht and Gregory Bard. *The M4RI Library – Version 20250722*. The M4RI Team. 2025. URL: https://bitbucket.org/malb/m4ri.

[Alb25] Martin Albrecht. *The M4RIE Library – Version 20250722*. The M4RIE Team. 2025. URL: https://bitbucket.org/malb/m4rie.

[Ber+22] Daniel J. Bernstein et al. *Classic McEliece: conservative code-based cryptography*. NIST Submission. Oct. 2022. URL: https://classic.mceliece.org/nist/mceliece-submission-20221023.pdf.

[Beu+23] W. Beullens et al. *MAYO: A Multivariate Quadratic Signature Scheme*. Tech. rep. National Institute of Standards and Technology, 2023. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/mayo-spec-web.pdf.

[Beu20] Ward Beullens. *Improved Cryptanalysis of UOV and Rainbow*. Cryptology ePrint Archive, Paper 2020/1343. 2020. URL: https://eprint.iacr.org/2020/1343.

[Che+16] Chen-Mou Cheng et al. *Solving Quadratic Equations with XL on Parallel Architectures - extended version*. Cryptology ePrint Archive, Paper 2016/412. 2016. URL: https://eprint.iacr.org/2016/412.

[Cou+00] Nicolas T. Courtois et al. "Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations (XL)". In: — (2000). Available as "xlfull.pdf" at minrank.org. URL: http://www.minrank.org/xlfull.pdf.

[EFR24] Antonio Corbo Esposito, Rosa Fera, and Francesco Romeo. *OliVier: an Oil and Vinegar based cryptosystem*. 2024. arXiv: 2405.08375 [math.AC]. URL: https://arxiv.org/abs/2405.08375.

[Fau02] Jean-Charles Faugère. "A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)". In: *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation (ISSAC '02)*. New York, NY, USA: ACM, 2002, pp. 75–83. DOI: 10.1145/780506.780516. URL: https://doi.org/10.1145/780506.780516.

[Fau99] Jean-Charles Faugère. "A new efficient algorithm for computing Gröbner bases (F4)". In: *Journal of Pure and Applied Algebra* 139.1–3 (1999), pp. 61–88. DOI: 10.1016/S0022-4049(99)00005-5. URL: https://doi.org/10.1016/S0022-4049(99)00005-5.

[FK21] H. Furue and M. Kudo. "Polynomial XL: A Variant of the XL Algorithm Using Macaulay Matrices over Polynomial Rings". In: 2021/1609 (2021). URL: https://eprint.iacr.org/2021/1609.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. "How to Enhance the Security of Public-Key Encryption at Minimum Cost". In: *Public Key Cryptography (PKC 1999)*. Springer, 1999, pp. 53–68. DOI: 10.1007/3-540-48910-X_6.

[KPG99] Aviad Kipnis, Jacques Patarin, and Louis Goubin. "Unbalanced Oil and Vinegar Signature Schemes". In: *Advances in Cryptology – EUROCRYPT '99*. Ed. by Jacques Stern. Vol. 1592. Lecture Notes in Computer Science. Prague, Czech Republic: Springer, 1999, pp. 206–222. DOI: 10.1007/3-540-48910-X_15. URL: https://doi.org/10.1007/3-540-48910-X_15.

[KS98] Aviad Kipnis and Adi Shamir. "Cryptanalysis of the Oil & Vinegar Signature Scheme". In: *Advances in Cryptology – CRYPTO '98*. Ed. by Hugo Krawczyk. Vol. 1462. Lecture Notes in Computer Science. Santa Barbara, California, USA: Springer, 1998, pp. 257–266. DOI: 10.1007/BFb0055733. URL: https://doi.org/10.1007/BFb0055733.

[NIS25] NIST Post-Quantum Cryptography Project. *UOV: Unbalanced Oil and Vinegar Specification Document*. Tech. rep. National Institute of Standards and Technology, Feb. 2025. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/uov-spec-round2-web.pdf.

[Pat97] Jacques Patarin. "The Oil and Vinegar Signature Scheme". Presented at the Dagstuhl Workshop on Cryptography, transparencies. Sept. 1997.

[Péb23] Pierre Pébereau. *One vector to rule them all: Key recovery from one vector in UOV schemes*. Cryptology ePrint Archive, Paper 2023/1131. 2023. URL: https://eprint.iacr.org/2023/1131.

A Post-quantum cryptography

The most important classes of post-quantum cryptographic systems are:

- **Hash-based cryptography** relies on the collision resistance and preimage resistance of cryptographic hash functions. These schemes, such as the Merkle signature scheme and its modern variants (e.g., XMSS and SPHINCS+), are primarily used for digital signatures. Their security is well-understood and depends only on the security of the underlying hash function.

- **Code-based cryptography** uses the hardness of decoding a random linear error-correcting code. The most prominent example is the McEliece cryptosystem, proposed in 1978, which remains unbroken despite decades of cryptanalysis. Code-based schemes tend to have large public keys but offer fast encryption and decryption operations.

- **Lattice-based cryptography** relies on the hardness of computational problems on lattices, such as the Shortest Vector Problem (SVP) and the Learning With Errors (LWE) problem. These schemes are highly efficient and versatile, supporting encryption, digital signatures, and even fully homomorphic encryption. Examples include Kyber (encryption) and Dilithium (signatures), both standardized by NIST in 2024.

- **Multivariate cryptography** is based on the difficulty of solving systems of multivariate quadratic equations over finite fields, a problem known to be NP-hard. These schemes are particularly efficient for digital signatures, with examples including UOV and GeMSS.

- **Isogeny-based cryptography** relies on the difficulty of finding isogenies (structure-preserving maps) between elliptic curves. Although this field is newer and still under active research, it promises relatively small key sizes compared to other post-quantum schemes. The Supersingular Isogeny Key Encapsulation (SIKE) protocol was one of the best-known candidates until it was recently broken. Another promising approach is CSIDH (Commutative Supersingular Isogeny Diffie–Hellman), which operates over supersingular elliptic curves defined over

the rationals and allows for a group action similar to classical Diffie-Hellman. CSIDH provides elegant mathematical simplicity and potential for efficient implementations, although its security level is still being thoroughly studied.

- **Symmetric key quantum resistance** focuses on adapting symmetric algorithms to resist quantum attacks. While Grover's algorithm can theoretically halve the effective key length of symmetric ciphers, increasing key sizes (e.g., using AES-256 or SHA-512) provides sufficient security against quantum adversaries.

Many cryptographic schemes base their security on a particular class of computational problems known as *NP-hard* problems.

For instance, the security of lattice-based cryptography relies on problems such as the Shortest Vector Problem (SVP) and the Learning With Errors (LWE) problem. Multivariate cryptography depends on the difficulty of solving systems of nonlinear polynomial equations over finite fields, while code-based cryptography relies on the hardness of decoding random linear codes. These problems belong to this class, which is believed to be *quantum-resistant*.

It is important to understand that any problem in this class can be reduced to any other problem within the same class. Therefore, if there exists an efficient algorithm that solves one NP-hard problem, then there exists an efficient algorithm that solves any NP-hard problem.

P $\subseteq$ NP holds by definition, but it remains an open question whether P = NP. Asymmetric cryptography fundamentally relies on the assumption that P $\neq$ NP, since if P = NP, every problem that can be verified in polynomial time could also be solved in polynomial time. This would undermine the concept of a secret key, allowing anyone to efficiently solve the underlying hard problems and invalidate the existence of one-way functions.

B Multivariate Quadratic systems

In multivariate cryptography, the system of polynomials is typically chosen to have degree 2. The motivation behind this choice lies in the better efficiency of the resulting schemes and the fact that the corresponding system remains NP-hard even at the lowest degree. Furthermore, any system of higher degree can be reduced to a quadratic one through a polynomial-time reduction, which does not increase the asymptotic security of the system.

A MQ system of m equations in n variables can be expressed as

$$\mathcal{P}(\mathbf{x}) = (p^{(1)}(x_1, \ldots, x_n), p^{(2)}(x_1, \ldots, x_n), \ldots, p^{(m)}(x_1, \ldots, x_n)) \text{ where:}$$

$$\begin{aligned}
p^{(1)}(x_1, \ldots, x_n) &= \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^{(1)} x_i x_j + \sum_{i=1}^{n} p_i^{(1)} x_i + p_0^{(1)} \\
p^{(2)}(x_1, \ldots, x_n) &= \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^{(2)} x_i x_j + \sum_{i=1}^{n} p_i^{(2)} x_i + p_0^{(2)} \\
&\vdots \\
p^{(m)}(x_1, \ldots, x_n) &= \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^{(m)} x_i x_j + \sum_{i=1}^{n} p_i^{(m)} x_i + p_0^{(m)}
\end{aligned}$$

Representation of a MQ system Each polynomial can be represented as a matrix. This is important for creating an efficient system, as matrix operations are highly optimized for hardware implementation. Each component $p^{(k)}$ $(k = 1, \ldots, m)$ can be written as a matrix-vector product using an upper triangular $(n+1) \times (n+1)$ matrix $P^{(k)}$ of the form:

$$P^{(k)} = \begin{pmatrix} p_{11}^{(k)} & p_{12}^{(k)} & p_{13}^{(k)} & \cdots & p_{1n}^{(k)} & p_{1}^{(k)} \\ 0 & p_{22}^{(k)} & p_{23}^{(k)} & \cdots & p_{2n}^{(k)} & p_{2}^{(k)} \\ 0 & 0 & p_{33}^{(k)} & \cdots & p_{3n}^{(k)} & p_{3}^{(k)} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & p_{nn}^{(k)} & p_{n}^{(k)} \\ 0 & 0 & \cdots & 0 & 0 & p_{0}^{(k)} \end{pmatrix}$$

To evaluate the polynomial on any vector $(\mathbf{x}) = (x_1, \ldots, x_n)$, compute $p^{(k)}(\mathbf{x}) = (\mathbf{x}) \cdot P^{(k)} \cdot (\mathbf{x})^T$ $k = (1, \ldots, m)$

Homogeneous system A homogeneous system is a polynomial system where each polynomial has the maximum degree. This is often used in MQ cryptography because the system remains NP-hard, and its efficiency is improved. The matrix of a homogeneous quadratic polynomial will have dimension $n \times n$ and will not contain the column of the linear terms and the row of the constant terms.

$$P^{(k)} = \begin{pmatrix} p_{11}^{(k)} & p_{12}^{(k)} & p_{13}^{(k)} & \cdots & p_{1n}^{(k)} \\ 0 & p_{22}^{(k)} & p_{23}^{(k)} & \cdots & p_{2n}^{(k)} \\ 0 & 0 & p_{33}^{(k)} & \cdots & p_{3n}^{(k)} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & p_{nn}^{(k)} \end{pmatrix}$$

Symmetric Matrix

The symmetric matrix corresponding to a multivariate homogeneous quadratic polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$ is the $n \times n$ matrix

$$\begin{cases} Q = \begin{pmatrix} p_{11} & \frac{1}{2}p_{12} & \frac{1}{2}p_{13} & \cdots & \frac{1}{2}p_{1n} \\ \frac{1}{2}p_{21} & p_{22} & \frac{1}{2}p_{23} & \cdots & \frac{1}{2}p_{2n} \\ \frac{1}{2}p_{31} & \frac{1}{2}p_{32} & p_{33} & \cdots & \frac{1}{2}p_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \frac{1}{2}p_{n1} & \frac{1}{2}p_{n2} & \cdots & \frac{1}{2}p_{nn-1} & p_{nn} \end{pmatrix} & \text{if char}(\mathbb{F}) \text{ is odd} \\ Q = \begin{pmatrix} 0 & p_{12} & p_{13} & \cdots & p_{1n}^{(k)} \\ 0 & 0 & p_{23} & \cdots & p_{2n} \\ 0 & 0 & 0 & \cdots & p_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & p_{12} & p_{13} & \cdots & p_{1n} \\ 0 & 0 & p_{23} & \cdots & p_{2n} \\ 0 & 0 & 0 & \cdots & p_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}^T & \text{if char}(\mathbb{F}) \text{ is even} \end{cases}$$

Symmetric matrices are useful in the description of a system and often exploited in cryptanalysis.

Polar or **differential form** The differential of a multivariate quadratic system is symmetric and bilinear in $\mathbf{x}$ and $\mathbf{y}$

$$\mathcal{DP}(\mathbf{x}, \mathbf{y}) = \mathcal{P}(\mathbf{x} + \mathbf{y}) - \mathcal{P}(\mathbf{x}) = \mathcal{P}(\mathbf{y}) + \mathcal{P}(\mathbf{0})$$

In the case of a homogeneous quadratic system, the differential is given by

$$\mathcal{DP}(\mathbf{x}, \mathbf{y}) = \mathcal{P}(\mathbf{x} + \mathbf{y}) - \mathcal{P}(\mathbf{x}) = \mathcal{P}(\mathbf{y})$$

since $\mathcal{P}(\mathbf{0}) = 0$ holds.

C Multivariate Public Key Cryptography

Problem of Solving Polynomial Systems (PoSSo) Given a system $\mathcal{P} = (p^{(1)}, \ldots, p^{(m)})$ of m nonlinear polynomial equations in the variables $x_1, \ldots, x_n$, find values $\bar{x_1}, \ldots, \bar{x_n}$ such that:

$$p^{(1)}(\bar{x_1}, \ldots, \bar{x_n}) = \cdots = p^{(m)}(\bar{x_1}, \ldots, \bar{x_n}) = 0$$

Solving a multivariate polynomial system is proven to be NP-complete even in the simplest case of quadratic polynomials over $\mathbb{F}_2$. In fact, it can be shown to be equivalent to the SAT3 problem.

To understand why this problem is hard, consider the system of 4 multivariate polynomials in $\mathbb{F}_2[x_1, x_2, x_3]$

$$\begin{cases} x_1x_2 + x_2x_3 + x_3 + 1 = 0 \\ x_1 + x_2 + x_3 = 0 \\ x_1 + x_1x_3 + x_2x_3 + 1 = 0 \\ x_1x_2 + x_1x_3 + x_2 + x_2x_3 = 0 \end{cases}$$

The system can be solved using the brute force approach by trying all possible vectors $(a_1, a_2, a_3) \in \mathbb{F}_2^3$. It is easily seen that the number of possible solution grow exponentially with the number of variables (2^n).

The system can be linearized by substituting the variable x_{ij} with the variable x_{ij} to obtain a linear system, which can be efficiently solved by applying Gaussian elimination.

$$\begin{cases} x_{12} + x_{23} + x_3 + 1 = 0 \\ x_1 + x_2 + x_3 = 0 \\ x_1 + x_{13} + x_{23} + 1 = 0 \\ x_{12} + x_{13} + x_2 + x_{23} = 0 \end{cases}$$

The system obtained has 4 equations and 6 variables. As stated by the Rouchè-Capelli Theorem, the possible solutions are 2^{6-4}. This can be verified by solving the system and obtaining:

$$\begin{cases} x_1 = x_2 + x_3 \\ x_1x_2 = x_3 + 1 \\ x_1x_3 = x_2 + x_3 + 1 \\ x_2x_3 = 0 \end{cases}$$

As expected the system have 2 free variables; $x_2, x_3 \in \mathbb{F}_2$ so the possible solutions have the form $(0, 0, x_2 + x_3, x_2, x_3 + 1, x_3)$. They are:

- $(0, 1, 1, 0, 0, 0)$
- $(1, 0, 0, 0, 0, 1)$
- $(1, 1, 0, 1, 0, 0)$
- $(0, 0, 1, 1, 0, 1)$

The coefficient matrix of the solutions is given by:

$$\begin{pmatrix} x_1 & x_1x_2 & x_1x_3 & x_2 & x_2x_3 & x_3 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

It can be easily seen that rows 1, 2, and 4 are not valid solutions:

- $x^1 \cdot x^2 \neq x_1x_2$ and $x^1 \cdot x^3 \neq x_1x_3$
- $x^1 \cdot x^3 \neq x_1x_3$
- $x^1 \cdot x^3 \neq x_1x_3$ and $x^2 \cdot x^3 \neq x_2x_3$

The only solutions where $x^1 \cdot x^2 = x_1x_2$, $x^1 \cdot x^3 = x_1x_3$, and $x^2 \cdot x^3 = x_2x_3$ are satisfied are the solutions 3, which is a valid solution for the original system. The other 3 solutions are called parasitic.

In general, there are an exponential number of parasitic solutions. Hence, while this method yields the solution in polynomial time (Gaussian elimination has complexity $\mathcal{O}(n^3)$ or $\approx \mathcal{O}(n^{2.3729})$ using Coppersmith-Winograd optimization), the time complexity bound now becomes the time to test all solutions multiplied by the number of solutions to test for, which are exponential, leading to an exponential time complexity.

After the substitution, in general, there are up to $\frac{n(n+1)}{2}$ variables (which is the number of monomials of degree 2 in n variables). If the system has exactly $\frac{n(n+1)}{2}$ independent quadratic equations in the variables n variables, then the linearized system has at most one solution. If instead, the number of independent quadratic equations r is less then $\frac{n(n+1)}{2}$ the number of total solution is $q^{\frac{n(n+1)}{2}-r}$.

This result provides an important security bound for constructing a multivariate quadratic cryptosystem. And it hints at the fact that the best-known methods to solve MQ instances involve finding a model that increases the number of independent equations.

C.1 Construction methods for MPKC's

The **Public Key** is given by the representation of the polynomial system $\mathcal{P}$ while the **Secret Key** is given by the secret information used to efficiently solve the system (i.e., find a pre-image for a target)

C.1.1 Encryption scheme

- **Key Generation**(λ) Alice choose a security parameter λ and obtains a key pair(sk, pk).
- **Encryption**(m, pk) Bob chooses a message (represented as a sequence of $x_1 \ldots, x_n$) and evaluates the polynomials to the public key, obtaining c, which he will send to Alice.
- **Decryption**(c, sk) Alice uses the secret key to find the pre-image used by Bob to obtain the encrypted message.

C.1.2 Signature scheme

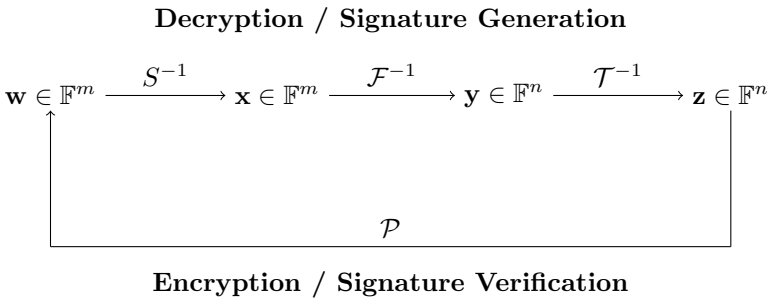
- **Key Generation**(λ) Alice choose a security parameter λ and obtains a key pair(sk, pk).
- **Signature**(m, sk) Alice chooses a message (represented as a result of the evaluation of the public key) and, using the secret key, finds a pre-image s of that message.
- **Verification**(m, s,pk) Bob check that the evaluation of s on the public key is equal to m.

C.2 The trapdoor construction

The key components are:

- The **Central Map** $\mathcal{F} : (x_1, \ldots, x_n) \in \mathbb{F}_q^n \to (f^{(1)}(x_1, \ldots, x_n), \ldots, f^{(m)}(x_1, \ldots, x_n)) \in \mathbb{F}_q^m$ is given by a system of structured polynomial. These polynomials are not random, but they embed a hidden structure that can be easily reversed.
- Two **bijective linear (or affine) transformations** that are used to hide the hidden structure of the central map: $\mathcal{S} \in \mathrm{GL}_n(\mathbb{F}_q)$ and $\mathcal{T} \in \mathrm{GL}_m(\mathbb{F}_q)$
- The **Public map** $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$

General workflow:



C.2.1 The IP Problem

Since the multivariate system is not based on randomly generated polynomials, the security of the cryptosystem cannot depend solely on the MQ problem. If an adversary can recover the central map, the system is trivially solved. The hardness of recovering the linear transformations used to hide the central map is expressed by the Isomorphism of Polynomials. In contrast to the MQ problem, the hardness of the IP problem has never been defined; it is assumed to be hard, as there are no currently known attacks that can recover any central map from the public map. When such a system is declared broken, it is because the central map selected was weak. This is the main drawback of multivariate cryptography.

Isomorphism of Polynomials with One Secret (IP1S) Given nonlinear multivariate system $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{B} = \mathcal{A} \circ \mathcal{T}$ for a linear or affine map $\mathcal{T}$, find a map $\mathcal{T}'$ such that $\mathcal{B} = \mathcal{A} \circ \mathcal{T}'$.

Isomorphism of Polynomials with Two Secrets (IP2S) Given nonlinear multivariate system $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{B} = \mathcal{S} \circ \mathcal{A} \circ \mathcal{T}$ for some linear or affine maps $\mathcal{S}$ and $\mathcal{T}$, find two maps $\mathcal{S}'$ and $\mathcal{T}'$ such that $\mathcal{B} = \mathcal{S}' \circ \mathcal{A} \circ \mathcal{T}'$.

Extended Isomorphism of Polynomials (EIP) Given a special class $\mathcal{C}$ of nonlinear multivariate system and a nonlinear multivariate system $\mathcal{P}$ which can be written as $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ with affine maps $\mathcal{S}$ and $\mathcal{T}$ and $\mathcal{F} \in \mathcal{C}$, find a decomposition of $\mathcal{P}$ of the form $\mathcal{P} = \mathcal{S}' \circ \mathcal{F}' \circ \mathcal{T}'$ with affine maps $\mathcal{S}'$ and $\mathcal{T}'$ and $\mathcal{F}' \in \mathcal{C}$.

C.3 Oil and Vinegars

The Oil and Vinegar scheme is a signature scheme introduced by Patarin[Pat97].

Let $\mathbb{F} = \mathbb{F}_q$ be a finite field with q elements and o, v be integers. The number of equations of the scheme is o and the number of variables is given by $n = o + v$.

(The original version used $o = v$. This has been proved to be insecure by Kipnis and Shamir. Currently, secure parameters are $2o < v < 3o$. And the name of the scheme has been changed to Unbalanced Oil and Vinegars (UOV)).

The key idea is to construct equations of the form $\sum\sum v_i v_j + \sum\sum v_i o_j$, so that after fixing the v variables the system will become linear in o variables, which can be then solved efficiently using Gaussian elimination (this happens because the quadratic term $v_i v_j$ becomes a number, $v_i o_j$ becomes o_j multiplied by a coefficient, and there are no $o_i o_j$ terms).

To prevent an adversary from performing the same operations, another structure is added to the Oil and Vinegar scheme. Namely, an affine map $\mathcal{T}$ whose purpose is to "shuffle" the variables in order to make it harder to reconstruct the original division in vinegars and oils. UOV is based on EIP.

More formally:

Let $V = \{1, \ldots, \}$ and $O = \{v + 1, \ldots, n\}$ be index set. Denote the variables x_i with $i \in V$ vinegar variables, and with $i \in O$ oil variables.

$$f^{(k)}(x_1, \ldots, x_n) = \sum_{i \in V, j \in V} \gamma_{ij}^{(k)} + \sum_{i \in V, j \in O} \gamma_{ij}^{(k)} + \sum_{i=1}^{n} \beta_i^{(k)} x_i + \alpha^{(k)} \text{ with } (k = 1, \ldots, o)$$

Private Key

- $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^o = (f^{(1)}, \ldots, f^{(o)})$
- $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n \in \mathrm{GL}_n(\mathbb{F}_q)$

Public Key $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ consisting of o quadratic polynomials in n variables

Signature Generation Let $\mathbf{m} \in \mathbb{F}^o$ the message to be signed/

1. Find a preimage $\mathbf{y} \in \mathbb{F}^n$ of $\mathbf{m}$ under the central map $\mathcal{F}$ (i.e. $\mathcal{F}(\mathbf{y}) = \mathbf{m}$)
   (a) Choose random values for the vinegar variables $y_1, \ldots, y_n$ and substitute them into the polynomials $f^{(1)}, \ldots, f^{(o)}$.
   (b) Solve the resulting linear system of o equations in the o Oil variables $y_{v+1}, \ldots, y_n$ by Gaussian elimination. (If the system has no solution, we can simply try again by sampling different vinegar values)
2. Compute the signature $\sigma \in \mathbb{F}^n$ by $\sigma = \mathcal{T}^{-1}(\mathbf{y})$

Signature Verification If $\mathcal{P}(\sigma) = \mathbf{m}$ the signature is valid

C.3.1 Alternative Description of the UOV trapdoor

[Beu20]

The UOV trapdoor function is a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ that vanishes on a secret linear subspace $\mathcal{O} \subset \mathbb{F}_q^n$ of dimension $\dim(\mathcal{O}) = m$, i.e. $\mathcal{P}(\mathbf{o}) = 0$ for all $\mathbf{o} \in \mathcal{O}$

Given a target $\mathbf{t} \in \mathbb{F}_q^m$ its preimage $\mathbf{x} \in \mathbb{F}_q^n$ (i.e. $\mathcal{P}(\mathbf{x}) = \mathbf{t}$) can be found by using the bipolar form.

1. Pick a random vector $\mathbf{v} \in \mathbb{F}_q^n$ (note that here $\mathbf{v}$ is not in $\mathbb{F}_q^v$)
2. Solve the system $\mathcal{P}(\mathbf{v} + \mathbf{o}) = \mathbf{t}$ for a vector $\mathbf{o} \in \mathcal{O}$

$$P(\mathbf{v}+\mathbf{o}) = \underbrace{P(\mathbf{v})}_{\text{fixed by choice of } \mathbf{v}} + \underbrace{P(\mathbf{o})}_{=0} + \underbrace{P'(\mathbf{v},\mathbf{o})}_{\text{linear function of } \mathbf{o}} = t.$$

If the system has no solutions, sample another random $\mathbf{v} \in \mathbb{F}_q^n$ and retry.

OV problem *For* $\mathbf{O} \in \mathbb{F}_q^{(n-o)\times o}$, *let* $\mathrm{MQ}_{n,m,q}(\mathbf{O})$ *denote the set of multivariate maps* $\mathcal{P} \in \mathrm{MQ}_{n,m,q}$ *that vanish on the rowspace of* $(\mathbf{O}^\top \quad \mathbf{I}_o)$. *The OV problem asks to distinguish a random multivariate quadratic map* $\mathcal{P} \in \mathrm{MQ}_{n,m,q}$ *from a random multivariate quadratic map in* $\mathrm{MQ}_{n,m,q}(\mathbf{O})$ *for a random* $\mathbf{O} \in \mathbb{F}_q^{(n-o)\times o}$.

Let $\mathcal{A}$ *be an OV distinguisher algorithm. We say the distinguishing advantage of* $\mathcal{A}$ *is:*

$$\mathrm{Adv}^{\mathrm{OV}}_{n,m,o,q}(\mathcal{A}) = \left| \Pr\left[\mathcal{A}(\mathcal{P}) = 1 \,\middle|\, \mathcal{P} \leftarrow \mathrm{MQ}_{m,n,q}\right] - \Pr\left[\mathcal{A}(\mathcal{P}) = 1 \,\middle|\, \begin{array}{l} \mathbf{O} \leftarrow \mathbb{F}_q^{(n-o)\times o} \\ \mathcal{P} \leftarrow \mathrm{MQ}_{n,m,q}(\mathbf{O}) \end{array}\right] \right|.$$

The OV problem is believed to be hard.

References

Jintai Ding, Albrecht Petzoldt, and Dieter S. Schmidt. *Multivariate Public Key Cryptosystems*. Advances in Information Security, vol. 80. Springer, New York, 2020. DOI: 10.1007/978-1-0716-0987-3.