Cryptographie à clé publique
Cours 9

Julien Lavauzelle

Université Paris 8

Master 1 mathématiques et applications – parcours ACC

25/03/2024

Séance précédente

Vu à la **séance précédente** :

- Introduction à la cryptographie post-quantique
- Réseaux euclidiens et problèmes difficiles associés
- Chiffrement NTRU
- LWE et chiffrement de Regev

Questions ?

Aujourd'hui

1. Cryptographie fondée sur les systèmes polynomiaux

2. Chiffrement HFE

3. Signature UOV

Plan

Le problème MQ

La **cryptographie « multivariée »** repose sur le problème de calculer des solutions de **systèmes** d'équations polynomiales (de degré ≥ 2) à plusieurs variables.

Le problème MQ

La **cryptographie « multivariée »** repose sur le problème de calculer des solutions de **systèmes** d'équations polynomiales (de degré ≥ 2) à plusieurs variables.

Le problème difficile sous-jacent est le **problème MQ** (pour *multivariate quadratic* : équations de degré 2). On le présente ici dans les corps finis.

Le problème MQ

La **cryptographie « multivariée »** repose sur le problème de calculer des solutions de **systèmes** d'équations polynomiales (de degré ≥ 2) à plusieurs variables.

Le problème difficile sous-jacent est le **problème MQ** (pour *multivariate quadratic* : équations de degré 2). On le présente ici dans les corps finis.

Problème MQ (*Multivariate Quadratic equations*). Soit $\mathbb{F}_q$ un corps fini, et $m, n \geq 1$ deux entiers.

Instance. Un système de m équations quadratiques à n inconnues $x_1, x_2, \ldots, x_n$:

$$\begin{cases} f_1(x_1, \ldots, x_n) = u_1 \\ \qquad\vdots \quad\vdots \\ f_m(x_1, \ldots, x_n) = u_m \end{cases} \qquad \text{où} \qquad f_k(x_1, \ldots, x_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j} \underbrace{x_i\, x_j}_{\text{degré } 2} + \sum_{i=1}^{n} b_i\, x_i + c$$

But. Trouver $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ une solution du système.

Le problème MQ

La **cryptographie « multivariée »** repose sur le problème de calculer des solutions de **systèmes** d'équations polynomiales (de degré ≥ 2) à plusieurs variables.

Le problème difficile sous-jacent est le **problème MQ** (pour *multivariate quadratic* : équations de degré 2). On le présente ici dans les corps finis.

Problème MQ (*Multivariate Quadratic equations*). Soit $\mathbb{F}_q$ un corps fini, et $m, n \geq 1$ deux entiers.

Instance. Un système de m équations quadratiques à n inconnues $x_1, x_2, \ldots, x_n$:

$$\begin{cases} f_1(x_1, \ldots, x_n) = u_1 \\ \qquad\vdots \quad\vdots \\ f_m(x_1, \ldots, x_n) = u_m \end{cases} \qquad \text{où} \qquad f_k(x_1, \ldots, x_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j} \underbrace{x_i\, x_j}_{\text{degré } 2} + \sum_{i=1}^{n} b_i\, x_i + c$$

But. Trouver $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ une solution du système.

Théorème. Le problème **MQ** est NP-difficile sur tous les corps finis.

Déguiser des équations

Pour une application en cryptographie, il faut déduire du problème MQ une **fonction à sens-unique** et **à trappe** ?

Déguiser des équations

Pour une application en cryptographie, il faut déduire du problème MQ une **fonction à sens-unique** et **à trappe** ?

Fait. On sait résoudre **efficacement** des équations polynomiales à **une** variable sur un corps fini.

Déguiser des équations

Pour une application en cryptographie, il faut déduire du problème MQ une **fonction à sens-unique** et **à trappe** ?

Fait. On sait résoudre **efficacement** des équations polynomiales à **une** variable sur un corps fini.

Par exemple, algorithme de recherche de racines de Chien (*Chien search*), ou algorithmes de factorisation de polynômes (algorithme de Berlekamp, de Cantor–Zassenhaus).
$\rightarrow$ leur complexité est polynomiale en $\log q$ et en le degré du polynôme.

Déguiser des équations

Pour une application en cryptographie, il faut déduire du problème MQ une **fonction à sens-unique** et **à trappe** ?

Fait. On sait résoudre **efficacement** des équations polynomiales à **une** variable sur un corps fini.

Par exemple, algorithme de recherche de racines de Chien (*Chien search*), ou algorithmes de factorisation de polynômes (algorithme de Berlekamp, de Cantor–Zassenhaus).
$\rightarrow$ leur complexité est polynomiale en $\log q$ et en le degré du polynôme.

Idée. Pour créer un système d'équations polynomiales multivariées **à trappe**, on passe de 1 variable à m variables :

Procédé :

- on choisit une extension de corps finis $\mathbb{F}_{q^m}/\mathbb{F}_q$,
- pour tout polynôme $f \in \mathbb{F}_{q^m}[x]$, l'équation $f(x) = 0$ (dans $\mathbb{F}_{q^m}$) peut se décomposer comme m équations polynomiales à m variables (dans $\mathbb{F}_q$).

Déguiser des équations : exemple

Exemple : soit $q = 2$ et $m = 3$. Le corps fini $\mathbb{F}_8 = \mathbb{F}_2[\omega]$ avec $\omega^3 = \omega + 1$.

Déguiser des équations : exemple

Exemple : soit $q = 2$ et $m = 3$. Le corps fini $\mathbb{F}_8 = \mathbb{F}_2[\omega]$ avec $\omega^3 = \omega + 1$.
On considère $f(X) = X^5 + X^2 + 1 \in \mathbb{F}_8[X]$.

Déguiser des équations : exemple

Exemple : soit $q = 2$ et $m = 3$. Le corps fini $\mathbb{F}_8 = \mathbb{F}_2[\omega]$ avec $\omega^3 = \omega + 1$.
On considère $f(X) = X^5 + X^2 + 1 \in \mathbb{F}_8[X]$.
Pour tout $x \in \mathbb{F}_8$, on a $x = x_0 + x_1\omega + x_2\omega^2$, avec $x_i \in \mathbb{F}_2$, donc on obtient :

Déguiser des équations : exemple

Exemple : soit $q = 2$ et $m = 3$. Le corps fini $\mathbb{F}_8 = \mathbb{F}_2[\omega]$ avec $\omega^3 = \omega + 1$.
On considère $f(X) = X^5 + X^2 + 1 \in \mathbb{F}_8[X]$.
Pour tout $x \in \mathbb{F}_8$, on a $x = x_0 + x_1\omega + x_2\omega^2$, avec $x_i \in \mathbb{F}_2$, donc on obtient :

$$f(x) = (x_0 + x_1\omega + x_2\omega^2)^5 + (x_0 + x_1\omega + x_2\omega^2)^2 + 1$$

Déguiser des équations : exemple

Exemple : soit $q = 2$ et $m = 3$. Le corps fini $\mathbb{F}_8 = \mathbb{F}_2[\omega]$ avec $\omega^3 = \omega + 1$.
On considère $f(X) = X^5 + X^2 + 1 \in \mathbb{F}_8[X]$.
Pour tout $x \in \mathbb{F}_8$, on a $x = x_0 + x_1\omega + x_2\omega^2$, avec $x_i \in \mathbb{F}_2$, donc on obtient :

$$\begin{aligned} f(x) &= (x_0 + x_1\omega + x_2\omega^2)^5 + (x_0 + x_1\omega + x_2\omega^2)^2 + 1 \\ &= (x_0 + x_1\omega^4 + x_2\omega^8)(x_0 + x_1\omega + x_2\omega^2) + (x_0 + x_1\omega^2 + x_2\omega^4) + 1 \end{aligned}$$

Déguiser des équations : exemple

Exemple : soit $q = 2$ et $m = 3$. Le corps fini $\mathbb{F}_8 = \mathbb{F}_2[\omega]$ avec $\omega^3 = \omega + 1$.
On considère $f(X) = X^5 + X^2 + 1 \in \mathbb{F}_8[X]$.
Pour tout $x \in \mathbb{F}_8$, on a $x = x_0 + x_1\omega + x_2\omega^2$, avec $x_i \in \mathbb{F}_2$, donc on obtient :

$$\begin{aligned} f(x) &= (x_0 + x_1\omega + x_2\omega^2)^5 + (x_0 + x_1\omega + x_2\omega^2)^2 + 1 \\ &= (x_0 + x_1\omega^4 + x_2\omega^8)(x_0 + x_1\omega + x_2\omega^2) + (x_0 + x_1\omega^2 + x_2\omega^4) + 1 \\ &= \text{... (des calculs) ...} \end{aligned}$$

Déguiser des équations : exemple

Exemple : soit $q = 2$ et $m = 3$. Le corps fini $\mathbb{F}_8 = \mathbb{F}_2[\omega]$ avec $\omega^3 = \omega + 1$.
On considère $f(X) = X^5 + X^2 + 1 \in \mathbb{F}_8[X]$.
Pour tout $x \in \mathbb{F}_8$, on a $x = x_0 + x_1\omega + x_2\omega^2$, avec $x_i \in \mathbb{F}_2$, donc on obtient :

$$\begin{aligned} f(x) &= (x_0 + x_1\omega + x_2\omega^2)^5 + (x_0 + x_1\omega + x_2\omega^2)^2 + 1 \\ &= (x_0 + x_1\omega^4 + x_2\omega^8)(x_0 + x_1\omega + x_2\omega^2) + (x_0 + x_1\omega^2 + x_2\omega^4) + 1 \\ &= \text{... (des calculs) ...} \\ &= (x_2x_1 + x_1 + x_2 + 1) + (x_0x_2 + x_1)\omega + (x_0x_2 + x_0x_1 + x_2)\omega^2 \end{aligned}$$

Déguiser des équations : exemple

Exemple : soit $q = 2$ et $m = 3$. Le corps fini $\mathbb{F}_8 = \mathbb{F}_2[\omega]$ avec $\omega^3 = \omega + 1$.
On considère $f(X) = X^5 + X^2 + 1 \in \mathbb{F}_8[X]$.
Pour tout $x \in \mathbb{F}_8$, on a $x = x_0 + x_1\omega + x_2\omega^2$, avec $x_i \in \mathbb{F}_2$, donc on obtient :

$$\begin{aligned} f(x) &= (x_0 + x_1\omega + x_2\omega^2)^5 + (x_0 + x_1\omega + x_2\omega^2)^2 + 1 \\ &= (x_0 + x_1\omega^4 + x_2\omega^8)(x_0 + x_1\omega + x_2\omega^2) + (x_0 + x_1\omega^2 + x_2\omega^4) + 1 \\ &= \text{... (des calculs) ...} \\ &= (x_2x_1 + x_1 + x_2 + 1) + (x_0x_2 + x_1)\omega + (x_0x_2 + x_0x_1 + x_2)\omega^2 \end{aligned}$$

Ainsi, dans la base $(1, \omega, \omega^2)$ de $\mathbb{F}_8/\mathbb{F}_2$, l'équation $f(x) = 0$ correspond à 3 équations :

$$\begin{cases} f_0(\boldsymbol{x}) &= x_2x_1 + x_1 + x_2 + 1 &= 0 \\ f_1(\boldsymbol{x}) &= x_0x_2 + x_1 &= 0 \\ f_2(\boldsymbol{x}) &= x_0x_2 + x_0x_1 + x_2 &= 0 \end{cases}$$

Déguiser des équations

Remarque 1. Pour obtenir un système d'équations **quadratiques** sur $\mathbb{F}_q$, on va choisir f sous une forme particulière :

Déguiser des équations

Remarque 1. Pour obtenir un système d'équations **quadratiques** sur $\mathbb{F}_q$, on va choisir f sous une forme particulière :

$$f(x) = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{m-1} b_i X^{q^i} + c\,.$$

Déguiser des équations

Remarque 1. Pour obtenir un système d'équations **quadratiques** sur $\mathbb{F}_q$, on va choisir f sous une forme particulière :

$$f(x) = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{m-1} b_i X^{q^i} + c\,.$$

En effet, comme $x \mapsto x^{q^i}$ est $\mathbb{F}_q$-linéaire, et vaut l'identité sur $\mathbb{F}_q$, en décomposant sur une base $\mathbb{F}_{q^m}/\mathbb{F}_q$, on obtient des polynômes dont les termes sont de degré au plus 2.

Déguiser des équations

Remarque 1. Pour obtenir un système d'équations **quadratiques** sur $\mathbb{F}_q$, on va choisir f sous une forme particulière :

$$f(x) = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{m-1} b_i X^{q^i} + c\,.$$

En effet, comme $x \mapsto x^{q^i}$ est $\mathbb{F}_q$-linéaire, et vaut l'identité sur $\mathbb{F}_q$, en décomposant sur une base $\mathbb{F}_{q^m}/\mathbb{F}_q$, on obtient des polynômes dont les termes sont de degré au plus 2.

Remarque 2. **Si l'on connaît** la base $\mathbb{F}_{q^m}/\mathbb{F}_q$ qui a été choisie, on peut aisément appliquer la transformation

$$f(x) = 0 \quad \longleftrightarrow \quad \{f_i(\boldsymbol{x}) = 0\}_{1 \le i \le m}$$

Déguiser des équations

Remarque 1. Pour obtenir un système d'équations **quadratiques** sur $\mathbb{F}_q$, on va choisir f sous une forme particulière :

$$f(x) = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{m-1} b_i X^{q^i} + c\,.$$

En effet, comme $x \mapsto x^{q^i}$ est $\mathbb{F}_q$-linéaire, et vaut l'identité sur $\mathbb{F}_q$, en décomposant sur une base $\mathbb{F}_{q^m}/\mathbb{F}_q$, on obtient des polynômes dont les termes sont de degré au plus 2.

Remarque 2. **Si l'on connaît** la base $\mathbb{F}_{q^m}/\mathbb{F}_q$ qui a été choisie, on peut aisément appliquer la transformation

$$f(x) = 0 \quad \longleftrightarrow \quad \{f_i(\boldsymbol{x}) = 0\}_{1 \le i \le m}$$

Idée pour la crypto : on va **cacher** cette base en appliquant des transformations affines sur les variables $\boldsymbol{x}$ et sur le système final.

Plan

Le système de chiffrement HFE : génération de clefs

J. Patarin introduit en 1996 le système de chiffrement **HFE** (*hidden field equations*).

Le système de chiffrement HFE : génération de clefs

J. Patarin introduit en 1996 le système de chiffrement **HFE** (*hidden field equations*).

Paramètres du système : une extension de corps fini $\mathbb{F}_{q^m}/\mathbb{F}_q$ et sa base associée.

HFE : GÉNÉRATION DE CLEFS

Le système de chiffrement HFE : génération de clefs

J. Patarin introduit en 1996 le système de chiffrement **HFE** (*hidden field equations*).

Paramètres du système : une extension de corps fini $\mathbb{F}_{q^m}/\mathbb{F}_q$ et sa base associée.

HFE : GÉNÉRATION DE CLEFS

1. Alice choisit aléatoirement un polynôme $f(X) \in \mathbb{F}_{q^m}[X]$ de la forme
$$f(X) = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{m-1} b_i X^{q^i} + c\,.$$
et calcule les polynômes $f_1(x_1,\ldots,x_m),\ldots,f_m(x_1,\ldots,x_m) \in \mathbb{F}_q[x_1,\ldots,x_m]$ associés.

Le système de chiffrement HFE : génération de clefs

J. Patarin introduit en 1996 le système de chiffrement **HFE** (*hidden field equations*).

Paramètres du système : une extension de corps fini $\mathbb{F}_{q^m}/\mathbb{F}_q$ et sa base associée.

HFE : GÉNÉRATION DE CLEFS

1. Alice choisit aléatoirement un polynôme $f(X) \in \mathbb{F}_{q^m}[X]$ de la forme
$$f(X) = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{m-1} b_i X^{q^i} + c\,.$$
et calcule les polynômes $f_1(x_1,\ldots,x_m),\ldots,f_m(x_1,\ldots,x_m) \in \mathbb{F}_q[x_1,\ldots,x_m]$ associés.
2. Alice choisit deux **transformations affines inversibles** $R, S : \mathbb{F}_q^m \to \mathbb{F}_q^m$

Le système de chiffrement HFE : génération de clefs

J. Patarin introduit en 1996 le système de chiffrement **HFE** (*hidden field equations*).

Paramètres du système : une extension de corps fini $\mathbb{F}_{q^m}/\mathbb{F}_q$ et sa base associée.

HFE : GÉNÉRATION DE CLEFS

1. Alice choisit aléatoirement un polynôme $f(X) \in \mathbb{F}_{q^m}[X]$ de la forme
$$f(X) = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{m-1} b_i X^{q^i} + c\,.$$
et calcule les polynômes $f_1(x_1,\ldots,x_m),\ldots,f_m(x_1,\ldots,x_m) \in \mathbb{F}_q[x_1,\ldots,x_m]$ associés.
2. Alice choisit deux **transformations affines inversibles** $R, S : \mathbb{F}_q^m \to \mathbb{F}_q^m$
3. Alice calcule
$$\begin{pmatrix} g_1(\boldsymbol{x}) \\ g_2(\boldsymbol{x}) \\ \vdots \\ g_m(\boldsymbol{x}) \end{pmatrix} = R \cdot \begin{pmatrix} f_1(S\cdot\boldsymbol{x}) \\ f_2(S\cdot\boldsymbol{x}) \\ \vdots \\ f_m(S\cdot\boldsymbol{x}) \end{pmatrix}$$

Le système de chiffrement HFE : génération de clefs

J. Patarin introduit en 1996 le système de chiffrement **HFE** (*hidden field equations*).

Paramètres du système : une extension de corps fini $\mathbb{F}_{q^m}/\mathbb{F}_q$ et sa base associée.

HFE : GÉNÉRATION DE CLEFS

1. Alice choisit aléatoirement un polynôme $f(X) \in \mathbb{F}_{q^m}[X]$ de la forme
$$f(X) = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{m-1} b_i X^{q^i} + c\,.$$
et calcule les polynômes $f_1(x_1,\ldots,x_m),\ldots,f_m(x_1,\ldots,x_m) \in \mathbb{F}_q[x_1,\ldots,x_m]$ associés.
2. Alice choisit deux **transformations affines inversibles** $R, S : \mathbb{F}_q^m \to \mathbb{F}_q^m$
3. Alice calcule
$$\begin{pmatrix} g_1(\boldsymbol{x}) \\ g_2(\boldsymbol{x}) \\ \vdots \\ g_m(\boldsymbol{x}) \end{pmatrix} = R \cdot \begin{pmatrix} f_1(S\cdot\boldsymbol{x}) \\ f_2(S\cdot\boldsymbol{x}) \\ \vdots \\ f_m(S\cdot\boldsymbol{x}) \end{pmatrix}$$
4. Les clefs sont :
   - **clé privée** : les transformations R et S et le polynôme $f \in \mathbb{F}_{q^m}[x]$
   - **clé publique** : les polynômes $g_i(\boldsymbol{x}) \in \mathbb{F}_q[x_1,\ldots,x_m]$.

Chiffrement HFE : chiffrement et déchiffrement

L'espace des **clairs** est $\mathbb{F}_q^m$, l'espace des **chiffrés** est aussi $\mathbb{F}_q^m$.

Chiffrement HFE : chiffrement et déchiffrement

L'espace des **clairs** est $\mathbb{F}_q^m$, l'espace des **chiffrés** est aussi $\mathbb{F}_q^m$.

HFE : CHIFFREMENT

Pour chiffrer un message $\boldsymbol{a} = (a_1, \ldots, a_m) \in \mathbb{F}_q^m$:

1. calculer $\boldsymbol{y} = (g_1(\boldsymbol{a}), g_2(\boldsymbol{a}), \ldots, g_m(\boldsymbol{a}))$,
2. retourner $\boldsymbol{y}$.

Chiffrement HFE : chiffrement et déchiffrement

L'espace des **clairs** est $\mathbb{F}_q^m$, l'espace des **chiffrés** est aussi $\mathbb{F}_q^m$.

HFE : CHIFFREMENT

Pour chiffrer un message $\boldsymbol{a} = (a_1, \ldots, a_m) \in \mathbb{F}_q^m$:

1. calculer $\boldsymbol{y} = (g_1(\boldsymbol{a}), g_2(\boldsymbol{a}), \ldots, g_m(\boldsymbol{a}))$,
2. retourner $\boldsymbol{y}$.

HFE : DÉCHIFFREMENT

1. Calculer $\boldsymbol{b} = R^{-1}(\boldsymbol{y}) \in \mathbb{F}_q^m$.
2. Calculer $\mathcal{Z} \subset \mathbb{F}_{q^m}$, l'ensemble des solutions z de l'équation $f(z) = b$, où $b \in \mathbb{F}_{q^m}$ est l'élément associé à $\boldsymbol{b}$.
3. Calculer $S^{-1}(\mathcal{Z})$ et reconnaître le message $\boldsymbol{a}$.

Chiffrement HFE : chiffrement et déchiffrement

L'espace des **clairs** est $\mathbb{F}_q^m$, l'espace des **chiffrés** est aussi $\mathbb{F}_q^m$.

HFE : CHIFFREMENT

Pour chiffrer un message $\boldsymbol{a} = (a_1, \ldots, a_m) \in \mathbb{F}_q^m$:

1. calculer $\boldsymbol{y} = (g_1(\boldsymbol{a}), g_2(\boldsymbol{a}), \ldots, g_m(\boldsymbol{a}))$,
2. retourner $\boldsymbol{y}$.

HFE : DÉCHIFFREMENT

1. Calculer $\boldsymbol{b} = R^{-1}(\boldsymbol{y}) \in \mathbb{F}_q^m$.
2. Calculer $\mathcal{Z} \subset \mathbb{F}_{q^m}$, l'ensemble des solutions z de l'équation $f(z) = b$, où $b \in \mathbb{F}_{q^m}$ est l'élément associé à $\boldsymbol{b}$.
3. Calculer $S^{-1}(\mathcal{Z})$ et reconnaître le message $\boldsymbol{a}$.

Remarque. Pour reconnaître le message $\boldsymbol{a}$ parmi toutes les solutions, on peut y ajouter de la redondance, un haché, ou encore utiliser un procédé de remplissage (*padding*).

HFE : cryptanalyse

Le chiffrement HFE a été attaqué par Kipnis et Shamir, en 1999, qui retrouvent la clef privée à partir de la clé publique en temps polynomial.

Cryptanalysis of the HFE Public Key Cryptosystem. A. Kipnis et A. Shamir. CRYPTO. **1999**.

HFE : cryptanalyse

Le chiffrement HFE a été attaqué par Kipnis et Shamir, en 1999, qui retrouvent la clef privée à partir de la clé publique en temps polynomial.

Cryptanalysis of the HFE Public Key Cryptosystem. A. Kipnis et A. Shamir. CRYPTO. **1999**.

Soit $G(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})) : \mathbb{F}_q^m \to \mathbb{F}_q^m$ la clé publique.

HFE : cryptanalyse

Le chiffrement HFE a été attaqué par Kipnis et Shamir, en 1999, qui retrouvent la clef privée à partir de la clé publique en temps polynomial.

Cryptanalysis of the HFE Public Key Cryptosystem. A. Kipnis et A. Shamir. CRYPTO. **1999**.

Soit $G(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})) : \mathbb{F}_q^m \to \mathbb{F}_q^m$ la clé publique.

Informellement, l'idée est de remarquer que la partie quadratique de G peut être modélisée comme une forme quadratique, dont **la matrice associée a un rang particulièrement faible**.

HFE : cryptanalyse

Le chiffrement HFE a été attaqué par Kipnis et Shamir, en 1999, qui retrouvent la clef privée à partir de la clé publique en temps polynomial.

Cryptanalysis of the HFE Public Key Cryptosystem. A. Kipnis et A. Shamir. CRYPTO. **1999**.

Soit $G(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \dots, g_m(\boldsymbol{x})) : \mathbb{F}_q^m \to \mathbb{F}_q^m$ la clé publique.

Informellement, l'idée est de remarquer que la partie quadratique de G peut être modélisée comme une forme quadratique, dont **la matrice associée a un rang particulièrement faible**.

On se ramène alors à une version d'un problème appelé **MinRank** (étant données des matrices $\boldsymbol{M}_1, \dots, \boldsymbol{M}_n$, trouver une combinaison linéaire des matrices qui a faible rang).

HFE : cryptanalyse

Le chiffrement HFE a été attaqué par Kipnis et Shamir, en 1999, qui retrouvent la clef privée à partir de la clé publique en temps polynomial.

Cryptanalysis of the HFE Public Key Cryptosystem. A. Kipnis et A. Shamir. CRYPTO. **1999**.

Soit $G(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})) : \mathbb{F}_q^m \to \mathbb{F}_q^m$ la clé publique.

Informellement, l'idée est de remarquer que la partie quadratique de G peut être modélisée comme une forme quadratique, dont **la matrice associée a un rang particulièrement faible**.

On se ramène alors à une version d'un problème appelé **MinRank** (étant données des matrices $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$, trouver une combinaison linéaire des matrices qui a faible rang).

Ce problème est NP-difficile, mais il s'avère que l'**instance choisie pour HFE est facile**, et résoluble par des techniques de « relinéarisation » (des attaques par calcul de **bases de Gröbner** fonctionnent également).

HFE : cryptanalyse

Le chiffrement HFE a été attaqué par Kipnis et Shamir, en 1999, qui retrouvent la clef privée à partir de la clé publique en temps polynomial.

Cryptanalysis of the HFE Public Key Cryptosystem. A. Kipnis et A. Shamir. CRYPTO. **1999**.

Soit $G(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \dots, g_m(\boldsymbol{x})) : \mathbb{F}_q^m \to \mathbb{F}_q^m$ la clé publique.

Informellement, l'idée est de remarquer que la partie quadratique de G peut être modélisée comme une forme quadratique, dont **la matrice associée a un rang particulièrement faible**.

On se ramène alors à une version d'un problème appelé **MinRank** (étant données des matrices $\boldsymbol{M}_1, \dots, \boldsymbol{M}_n$, trouver une combinaison linéaire des matrices qui a faible rang).

Ce problème est NP-difficile, mais il s'avère que l'**instance choisie pour HFE est facile**, et résoluble par des techniques de « relinéarisation » (des attaques par calcul de **bases de Gröbner** fonctionnent également).

Remarque. Des variantes de HFE (par exemple, multi-HFE, HFE$^-$,...) ont également été attaquées.

HFE : cryptanalyse

Le chiffrement HFE a été attaqué par Kipnis et Shamir, en 1999, qui retrouvent la clef privée à partir de la clé publique en temps polynomial.

Cryptanalysis of the HFE Public Key Cryptosystem. A. Kipnis et A. Shamir. CRYPTO. **1999**.

Soit $G(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})) : \mathbb{F}_q^m \to \mathbb{F}_q^m$ la clé publique.

Informellement, l'idée est de remarquer que la partie quadratique de G peut être modélisée comme une forme quadratique, dont **la matrice associée a un rang particulièrement faible**.

On se ramène alors à une version d'un problème appelé **MinRank** (étant données des matrices $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$, trouver une combinaison linéaire des matrices qui a faible rang).

Ce problème est NP-difficile, mais il s'avère que l'**instance choisie pour HFE est facile**, et résoluble par des techniques de « relinéarisation » (des attaques par calcul de **bases de Gröbner** fonctionnent également).

Remarque. Des variantes de HFE (par exemple, multi-HFE, HFE$^-$,...) ont également été attaquées.

Conclusion partielle. En général, peu de solutions pratiques de chiffrement à clef publique basées sur des systèmes polynomiaux multivariés. En revanche, beaucoup de **signatures numériques**.

Plan

Principe *oil and vinegar*

Soit $\mathbb{F}_q$ un corps fini, $q = 2^8$ typiquement. Soit également $v > o \geq 1$ avec $n \coloneqq v + o$.

Principe *oil and vinegar*

Soit $\mathbb{F}_q$ un corps fini, $q = 2^8$ typiquement. Soit également $v > o \geq 1$ avec $n \coloneqq v + o$.
On appelle

- $x_1, \ldots, x_v$ les variables de type « vinaigre » (*vinegar*), et on note $V \coloneqq \{1, \ldots, v\}$;
- $x_{v+1}, \ldots, x_n$ les variables de type « huile » (*oil*), et on note $O \coloneqq \{v+1, \ldots, n\}$.

Principe *oil and vinegar*

Soit $\mathbb{F}_q$ un corps fini, $q = 2^8$ typiquement. Soit également $v > o \geq 1$ avec $n \coloneqq v + o$.
On appelle

- $x_1, \ldots, x_v$ les variables de type « vinaigre » (*vinegar*), et on note $V \coloneqq \{1, \ldots, v\}$;
- $x_{v+1}, \ldots, x_n$ les variables de type « huile » (*oil*), et on note $O \coloneqq \{v+1, \ldots, n\}$.

On note enfin $\boldsymbol{x} = (x_1, \ldots, x_v, x_{v+1}, \ldots, x_n)$.

Principe *oil and vinegar*

Soit $\mathbb{F}_q$ un corps fini, $q = 2^8$ typiquement. Soit également $v > o \geq 1$ avec $n \coloneqq v + o$.
On appelle

- $x_1, \ldots, x_v$ les variables de type « vinaigre » (*vinegar*), et on note $V \coloneqq \{1, \ldots, v\}$;
- $x_{v+1}, \ldots, x_n$ les variables de type « huile » (*oil*), et on note $O \coloneqq \{v+1, \ldots, n\}$.

On note enfin $\boldsymbol{x} = (x_1, \ldots, x_v, x_{v+1}, \ldots, x_n)$.

Pour $k \geq v+1$, on définit un polynôme quadratique en les variables $\{x_i\}_{1 \leq i \leq n}$:

$$f_k(\boldsymbol{x}) = \sum_{i \in V, j \in O} \alpha_{i,j}^{(k)} x_i x_j + \sum_{i,j \in V, i \leq j} \beta_{i,j}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)}.$$

Principe *oil and vinegar*

Soit $\mathbb{F}_q$ un corps fini, $q = 2^8$ typiquement. Soit également $v > o \geq 1$ avec $n \coloneqq v + o$.
On appelle

- $x_1, \ldots, x_v$ les variables de type « vinaigre » (*vinegar*), et on note $V \coloneqq \{1, \ldots, v\}$;
- $x_{v+1}, \ldots, x_n$ les variables de type « huile » (*oil*), et on note $O \coloneqq \{v+1, \ldots, n\}$.

On note enfin $\boldsymbol{x} = (x_1, \ldots, x_v, x_{v+1}, \ldots, x_n)$.

Pour $k \geq v+1$, on définit un polynôme quadratique en les variables $\{x_i\}_{1 \leq i \leq n}$:

$$f_k(\boldsymbol{x}) = \sum_{i \in V, j \in O} \alpha_{i,j}^{(k)} x_i x_j + \sum_{i,j \in V, i \leq j} \beta_{i,j}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)}.$$

Remarque fondamentale. En temps polynomial en n, on peut calculer une application réciproque de :

$$\begin{array}{rccc} F : & \mathbb{F}_q^n & \to & \mathbb{F}_q^o \\ & \boldsymbol{x} = (x_1, \ldots, x_v, x_{v+1}, \ldots, x_n) & \mapsto & (f_{v+1}(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) \end{array}$$

Principe *oil and vinegar*

Soit $\mathbb{F}_q$ un corps fini, $q = 2^8$ typiquement. Soit également $v > o \geq 1$ avec $n \coloneqq v + o$.
On appelle

- $x_1, \ldots, x_v$ les variables de type « vinaigre » (*vinegar*), et on note $V \coloneqq \{1, \ldots, v\}$;
- $x_{v+1}, \ldots, x_n$ les variables de type « huile » (*oil*), et on note $O \coloneqq \{v+1, \ldots, n\}$.

On note enfin $\boldsymbol{x} = (x_1, \ldots, x_v, x_{v+1}, \ldots, x_n)$.

Pour $k \geq v+1$, on définit un polynôme quadratique en les variables $\{x_i\}_{1 \leq i \leq n}$:

$$f_k(\boldsymbol{x}) = \sum_{i \in V, j \in O} \alpha_{i,j}^{(k)} x_i x_j + \sum_{i,j \in V, i \leq j} \beta_{i,j}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)}.$$

Remarque fondamentale. En temps polynomial en n, on peut calculer une application réciproque de :

$$\begin{array}{rccc} F : & \mathbb{F}_q^n & \to & \mathbb{F}_q^o \\ & \boldsymbol{x} = (x_1, \ldots, x_v, x_{v+1}, \ldots, x_n) & \mapsto & (f_{v+1}(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) \end{array}$$

Comment ?

Principe *oil and vinegar*

Soit $\mathbb{F}_q$ un corps fini, $q = 2^8$ typiquement. Soit également $v > o \geq 1$ avec $n \coloneqq v + o$.
On appelle

- $x_1, \ldots, x_v$ les variables de type « vinaigre » (*vinegar*), et on note $V \coloneqq \{1, \ldots, v\}$;
- $x_{v+1}, \ldots, x_n$ les variables de type « huile » (*oil*), et on note $O \coloneqq \{v+1, \ldots, n\}$.

On note enfin $\boldsymbol{x} = (x_1, \ldots, x_v, x_{v+1}, \ldots, x_n)$.

Pour $k \geq v + 1$, on définit un polynôme quadratique en les variables $\{x_i\}_{1 \leq i \leq n}$:

$$f_k(\boldsymbol{x}) = \sum_{i \in V, j \in O} \alpha_{i,j}^{(k)} x_i x_j + \sum_{i,j \in V, i \leq j} \beta_{i,j}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)}.$$

Remarque fondamentale. En temps polynomial en n, on peut calculer une application réciproque de :

$$\begin{array}{rccc} F : & \mathbb{F}_q^n & \to & \mathbb{F}_q^o \\ & \boldsymbol{x} = (x_1, \ldots, x_v, x_{v+1}, \ldots, x_n) & \mapsto & (f_{v+1}(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) \end{array}$$

Comment ?

1. On choisit **aléatoirement** des valeurs $x_1, x_2, \ldots, x_v$, et en substituant dans les équations, on obtient un système linéaire de o équations en o inconnues.
2. On essaie de l'inverser par élimination gaussienne.
3. Si cela ne fonctionne pas, on recommence.

La signature UOV : *unbalanced oil and vinegar*

La signature **UOV** (*unbalanced oil and vinegar*), par J. Patarin en 1997.

Remarque. « *Unbalanced* » = $\mathbf{v} > o$.

La signature UOV : *unbalanced oil and vinegar*

La signature **UOV** (*unbalanced oil and vinegar*), par J. Patarin en 1997.

Remarque. « *Unbalanced* » = $\mathbf{v} > o$.

Intuition. Si on ne connaît pas quelles sont les variables « **vinaigre** » et « huile » (voire, si elles sont « mélangées ») alors on ne peut pas inverser F.

La signature UOV : *unbalanced oil and vinegar*

La signature **UOV** (*unbalanced oil and vinegar*), par J. Patarin en 1997.

Remarque. « *Unbalanced* » = $\mathbf{v} > o$.

Intuition. Si on ne connaît pas quelles sont les variables « **vinaigre** » et « huile » (voire, si elles sont « mélangées ») alors on ne peut pas inverser F.

UOV : GÉNÉRATION DE CLEFS

1. Alice choisit uniformément $L_1 : \mathbb{F}_q^o \to \mathbb{F}_q^o$ et $L_2 : \mathbb{F}_q^n \to \mathbb{F}_q^n$ deux applications affines inversibles.
2. Alice choisit uniformément $F : \mathbb{F}_q^n \to \mathbb{F}_q^o$ comme précédemment :

$$f_k(\boldsymbol{x}) = \sum_{i \in V, j \in O} \alpha_{i,j}^{(k)} x_i x_j + \sum_{i,j \in V, i \leq j} \beta_{i,j}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)} .$$

et

$$\begin{array}{rccc} F : & \mathbb{F}_q^n & \to & \mathbb{F}_q^o \\ & \boldsymbol{x} & \mapsto & (f_{v+1}(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) \end{array}$$

3. La clé publique est $P := L_1 \circ F \circ L_2$, la clé secrète est formée de L_1, L_2 et F.

La signature UOV : *unbalanced oil and vinegar*

On signe un message quelconque $\boldsymbol{m} \in \{0,1\}^*$. La signature $\boldsymbol{s}$ est un élément de $\mathbb{F}_q^n$.

La signature UOV : *unbalanced oil and vinegar*

On signe un message quelconque $\boldsymbol{m} \in \{0,1\}^*$. La signature $\boldsymbol{s}$ est un élément de $\mathbb{F}_q^n$.

On se donne une **fonction de hachage** cryptographique $H : \{0,1\}^* \to \mathbb{F}_q^o$, connue de tous.

La signature UOV : *unbalanced oil and vinegar*

On signe un message quelconque $\boldsymbol{m} \in \{0,1\}^*$. La signature $\boldsymbol{s}$ est un élément de $\mathbb{F}_q^n$.

On se donne une **fonction de hachage** cryptographique $H : \{0,1\}^* \to \mathbb{F}_q^o$, connue de tous.

UOV : SIGNATURE

Alice souhaite signer $\boldsymbol{m} \in \{0,1\}^*$.

1. Elle hache son message : $\boldsymbol{h} = H(\boldsymbol{m})$.
2. Elle calcule successivement $\boldsymbol{z} = L_1^{-1}(\boldsymbol{h})$, $\boldsymbol{z'} = F^{-1}(\boldsymbol{z})$ et , $\boldsymbol{s} = L_2^{-1}(\boldsymbol{z'})$.
3. Elle retourne la signature $\boldsymbol{s}$.

La signature UOV : *unbalanced oil and vinegar*

On signe un message quelconque $\boldsymbol{m} \in \{0,1\}^*$. La signature $\boldsymbol{s}$ est un élément de $\mathbb{F}_q^n$.

On se donne une **fonction de hachage** cryptographique $H : \{0,1\}^* \to \mathbb{F}_q^o$, connue de tous.

UOV : SIGNATURE

Alice souhaite signer $\boldsymbol{m} \in \{0,1\}^*$.

1. Elle hache son message : $\boldsymbol{h} = H(\boldsymbol{m})$.
2. Elle calcule successivement $\boldsymbol{z} = L_1^{-1}(\boldsymbol{h})$, $\boldsymbol{z'} = F^{-1}(\boldsymbol{z})$ et , $\boldsymbol{s} = L_2^{-1}(\boldsymbol{z'})$.
3. Elle retourne la signature $\boldsymbol{s}$.

UOV : VÉRIFICATION

Bob souhaite vérifier si la signature $\boldsymbol{s}$ correspond au message $\boldsymbol{m}$.

1. Bob calcule $\boldsymbol{h} = H(\boldsymbol{m})$
2. Bob calcule $P(\boldsymbol{s})$ (P est la clé publique), et compare le résultat avec $\boldsymbol{h}$.
3. S'il y a égalité, la signature est acceptée. Sinon elle est rejetée.

Sécurité de UOV

Informellement, la **sécurité de UOV** repose sur :

- le problème MQ,
- l'impossibilité de retrouver la décomposition $P = L_1 \circ F \circ L_2$.

Sécurité de UOV

Informellement, la **sécurité de UOV** repose sur :

- le problème MQ,
- l'impossibilité de retrouver la décomposition $P = L_1 \circ F \circ L_2$.

Des attaques **exponentielles** (mais efficaces pour les paramètres proposés) existent, ce qui implique de choisir des clés très grandes.

Sécurité de UOV

Informellement, la **sécurité de UOV** repose sur :

- le problème MQ,
- l'impossibilité de retrouver la décomposition $P = L_1 \circ F \circ L_2$.

Des attaques **exponentielles** (mais efficaces pour les paramètres proposés) existent, ce qui implique de choisir des clés très grandes.

Notons par exemple :

Sécurité de UOV

Informellement, la **sécurité de UOV** repose sur :

- le problème MQ,
- l'impossibilité de retrouver la décomposition $P = L_1 \circ F \circ L_2$.

Des attaques **exponentielles** (mais efficaces pour les paramètres proposés) existent, ce qui implique de choisir des clés très grandes.

Notons par exemple :

- La résolution directe du problème **MQ** sous-jacent. Pour cela on utilise un calcul de **bases de Gröbner** par les algorithmes F_4 ou F_5 de Faugère.

Sécurité de UOV

Informellement, la **sécurité de UOV** repose sur :

- le problème MQ,
- l'impossibilité de retrouver la décomposition $P = L_1 \circ F \circ L_2$.

Des attaques **exponentielles** (mais efficaces pour les paramètres proposés) existent, ce qui implique de choisir des clés très grandes.

Notons par exemple :

- La résolution directe du problème **MQ** sous-jacent. Pour cela on utilise un calcul de **bases de Gröbner** par les algorithmes F_4 ou F_5 de Faugère.
- La recherche d'une combinaison linéaire de faible rang (sur $\mathbb{F}_q$) des applications coordonnées de P (clé publique). C'est une version du problème **MinRank**, lui aussi résoluble par calcul de bases de Gröbner en temps exponentiel.

Sécurité de UOV

Informellement, la **sécurité de UOV** repose sur :

- le problème MQ,
- l'impossibilité de retrouver la décomposition $P = L_1 \circ F \circ L_2$.

Des attaques **exponentielles** (mais efficaces pour les paramètres proposés) existent, ce qui implique de choisir des clés très grandes.

Notons par exemple :

- La résolution directe du problème **MQ** sous-jacent. Pour cela on utilise un calcul de **bases de Gröbner** par les algorithmes F_4 ou F_5 de Faugère.
- La recherche d'une combinaison linéaire de faible rang (sur $\mathbb{F}_q$) des applications coordonnées de P (clé publique). C'est une version du problème **MinRank**, lui aussi résoluble par calcul de bases de Gröbner en temps exponentiel.
- Recherche brute des variables « huile ».

Sécurité de UOV

Informellement, la **sécurité de UOV** repose sur :

- le problème MQ,
- l'impossibilité de retrouver la décomposition $P = L_1 \circ F \circ L_2$.

Des attaques **exponentielles** (mais efficaces pour les paramètres proposés) existent, ce qui implique de choisir des clés très grandes.

Notons par exemple :

- La résolution directe du problème **MQ** sous-jacent. Pour cela on utilise un calcul de **bases de Gröbner** par les algorithmes F_4 ou F_5 de Faugère.
- La recherche d'une combinaison linéaire de faible rang (sur $\mathbb{F}_q$) des applications coordonnées de P (clé publique). C'est une version du problème **MinRank**, lui aussi résoluble par calcul de bases de Gröbner en temps exponentiel.
- Recherche brute des variables « huile ».
- D'autres : réduction à des sous-corps, recherche de sous-espaces particuliers, etc.

Sécurité de UOV

Informellement, la **sécurité de UOV** repose sur :

- le problème MQ,
- l'impossibilité de retrouver la décomposition $P = L_1 \circ F \circ L_2$.

Des attaques **exponentielles** (mais efficaces pour les paramètres proposés) existent, ce qui implique de choisir des clés très grandes.

Notons par exemple :

- La résolution directe du problème **MQ** sous-jacent. Pour cela on utilise un calcul de **bases de Gröbner** par les algorithmes F_4 ou F_5 de Faugère.
- La recherche d'une combinaison linéaire de faible rang (sur $\mathbb{F}_q$) des applications coordonnées de P (clé publique). C'est une version du problème **MinRank**, lui aussi résoluble par calcul de bases de Gröbner en temps exponentiel.
- Recherche brute des variables « huile ».
- D'autres : réduction à des sous-corps, recherche de sous-espaces particuliers, etc.

Remarque. La version équilibrée ($v = o$) de UOV a été attaquée en temps polynomial par Kipnis et Shamir.

La signature Rainbow

La signature **Rainbow** proposait une amélioration qui consiste à « empiler » $m \geq 2$ couches de problèmes **UOV**.

Elle avait été proposée pour standardisation au NIST, avec le choix $m = 3$.

`https://www.pqcrainbow.org`

Tailles de clefs et signature :

sécurité (bits)	128	192	256
$(q, v_1, v_2 - v_1, v_3 - v_2)$	$(16, 36, 32, 32)$	$(256, 68, 32, 48)$	$(256, 96, 36, 64)$
clé publique	157.8 ko	861.3 ko	1.89 Mo
clé privée	101.2 ko	611.3 ko	1.38 Mo
taille de signature	66 o	164 o	204 o

La signature Rainbow

La signature **Rainbow** proposait une amélioration qui consiste à « empiler » $m \geq 2$ couches de problèmes **UOV**.

Elle avait été proposée pour standardisation au NIST, avec le choix $m = 3$.

`https://www.pqcrainbow.org`

Tailles de clefs et signature :

sécurité (bits)	128	192	256
$(q, v_1, v_2 - v_1, v_3 - v_2)$	$(16, 36, 32, 32)$	$(256, 68, 32, 48)$	$(256, 96, 36, 64)$
clé publique	157.8 ko	861.3 ko	1.89 Mo
clé privée	101.2 ko	611.3 ko	1.38 Mo
taille de signature	66 o	164 o	204 o

Attaque récente. Mais à la conférence CRYPTO en 2022, Beullens donne une attaque sur la clé d'une durée de quelques jours.

La signature Rainbow

La signature **Rainbow** proposait une amélioration qui consiste à « empiler » $m \geq 2$ couches de problèmes **UOV**.

Elle avait été proposée pour standardisation au NIST, avec le choix $m = 3$.

`https://www.pqcrainbow.org`

Tailles de clefs et signature :

sécurité (bits)	128	192	256
$(q, v_1, v_2 - v_1, v_3 - v_2)$	$(16, 36, 32, 32)$	$(256, 68, 32, 48)$	$(256, 96, 36, 64)$
clé publique	157.8 ko	861.3 ko	1.89 Mo
clé privée	101.2 ko	611.3 ko	1.38 Mo
taille de signature	66 o	164 o	204 o

Attaque récente. Mais à la conférence CRYPTO en 2022, Beullens donne une attaque sur la clé d'une durée de quelques jours.

La signature **UOV** reste proposée comme candidate (dans de nombreuses versions) à la compétition du NIST.

Fin de la partie cours

Questions ?