Multivariate cryptography – Optimizations and the MQ problem

SLMath summer school: Introduction to Quantum-Safe Cryptography (IBM Zurich)

Simona Samardjiska

July, 2024

Institute for Computing and Information Sciences
Radboud University

Schedulle (tentative)

- **Monday - Designs**
  - General
  - Classic designs
- **Tuesday - Design and general MQ solving techniques**
  - Public key optimization techniques
  - Algorithms for solving the MQ problem
- **Wednesday - Cryptanalysis**
  - MinRank
  - Equivalent keys attacks
- **Thursday - Cryptanalysis and provably secure designs**
  - Attacks on UOV
  - Fiat-Shamir signatures I
- **Friday - Provably secure designs**
  - Fiat-Shamir signatures II

Notations

- $\mathbb{F}_q$ – finite field of q elements,
- $\mathbb{F}_q^m$ – vector space of vectors $(u_1, u_2, \dots, u_m)$ over $\mathbb{F}_q$
- $\mathbb{F}_{q^m}$ – extension field of $\mathbb{F}_q$ of degree m
- $\mathbb{F}_q[x_1, \dots, x_n]$ – ring of polynomials over $\mathbb{F}_q$ in the variables $x_1, \dots, x_n$
- polynomial ideal - subset of $\mathbb{F}_q[x_1, \dots, x_n]$ closed under linear combination with polynomial coefficients
- $\mathrm{GL}_n(\mathbb{F}_q)$ – general linear group of degree n over $\mathbb{F}_q$.
- $\mathbf{x} = (x_1, \dots, x_n)^\top$ – column vectors in $\mathbb{F}_q^n$, $\mathbf{x}^\top = (x_1, \dots, x_n)$ – row vectors in $\mathbb{F}_q^n$
- $p(x_1, \dots, x_n) = \sum_{1 \le i \le j \le n} \alpha_{ij} x_i x_j$ – quadratic form
  - matrix form $\bar{\mathbf{P}} = \mathbf{P} + \mathbf{P}^\top$, where $\mathbf{P}_{ij} = \alpha_{ij}/2$ over char $\neq 2$ or $\mathbf{P}_{ij} = \alpha_{ij}$ over char $= 2$

Public key optimization techniques

Equivalent keys

- Let $(\mathcal{F}, \mathbf{S}, \mathbf{T})$ be a private key for the public key $\mathcal{P}$ of a multivariate scheme
- $(\mathcal{F}, \mathbf{S}, \mathbf{T}) \simeq (\mathcal{F}', \mathbf{S}', \mathbf{T}')$ (**the keys are equivalent**) if and only if:

$$(\mathbf{T} \circ \mathcal{F} \circ \mathbf{S} = \mathbf{T}' \circ \mathcal{F}' \circ \mathbf{S}')$$

  and $(\mathcal{F}', \mathbf{S}', \mathbf{T}')$ can be used as a private key of $(\mathcal{F}, \mathbf{S}, \mathbf{T})$.
- How to find an equivalent key?

$$\begin{aligned} \mathcal{P} &= \mathcal{T} \circ \mathcal{F} \circ \mathcal{S} \Leftrightarrow \\ \mathcal{P} &= \underbrace{\mathcal{T} \circ \Sigma^{-1}} \circ \underbrace{\Sigma \circ \mathcal{F} \circ \Omega} \circ \underbrace{\Omega^{-1} \circ \mathcal{S}} \Leftrightarrow \\ \mathcal{P} &= \quad \mathcal{T}' \quad \circ \quad \mathcal{F}' \quad \circ \quad \mathcal{S}' \end{aligned}$$

  - we try to find the matrices Σ and Ω.

Equivalent keys

- Let $(\mathcal{F}, \mathbf{S}, \mathbf{T})$ be a private key for the public key $\mathcal{P}$ of a multivariate scheme
- $(\mathcal{F}, \mathbf{S}, \mathbf{T}) \simeq (\mathcal{F}', \mathbf{S}', \mathbf{T}')$ (**the keys are equivalent**) if and only if:

$$(\mathbf{T} \circ \mathcal{F} \circ \mathbf{S} = \mathbf{T}' \circ \mathcal{F}' \circ \mathbf{S}')$$

  and $(\mathcal{F}', \mathbf{S}', \mathbf{T}')$ can be used as a private key of $(\mathcal{F}, \mathbf{S}, \mathbf{T})$.
- How to find an equivalent key?

$$\begin{aligned}
\mathcal{P} &= \mathcal{T} \circ \mathcal{F} \circ \mathcal{S} \Leftrightarrow \\
\mathcal{P} &= \underbrace{\mathcal{T} \circ \Sigma^{-1}} \circ \underbrace{\Sigma \circ \mathcal{F} \circ \Omega} \circ \underbrace{\Omega^{-1} \circ \mathcal{S}} \Leftrightarrow \\
\mathcal{P} &= \quad \mathcal{T}' \quad \circ \quad \mathcal{F}' \quad \circ \quad \mathcal{S}'
\end{aligned}$$

  - we try to find the matrices Σ and Ω.

Equivalent keys

- It is actually a **cryptanalytical technique**, used quite often
- But it can be used for reduction of **the size of the public key** :)
  - **Recall that the public keys are huge**
  - For example of UOV Level 1 it is 412KB
  - with the optimization it is 66KB
- This optimization introduces weaknesses as we will see in the next lectures ...
  - if not used properly
  - for side-channel analysis

Equivalent keys

- It is actually a **cryptanalytical technique**, used quite often
- But it can be used for reduction of **the size of the public key** :)
  - **Recall that the public keys are huge**
  - For example of UOV Level 1 it is 412KB
  - with the optimization it is 66KB
- This optimization introduces weaknesses as we will see in the next lectures ...
  - if not used properly
  - for side-channel analysis

Equivalent keys

- It is actually a **cryptanalytical technique**, used quite often
- But it can be used for reduction of **the size of the public key** :)
  - **Recall that the public keys are huge**
  - For example of UOV Level 1 it is 412KB
  - with the optimization it is 66KB
- This optimization introduces weaknesses as we will see in the next lectures ...
  - if not used properly
  - for side-channel analysis

Equivalent keys for UOV

- Central map: $\mathcal{F}^{(s)}(x_1,\ldots,x_n) = \sum_{i,j\in V, i\leqslant j} \alpha_{ij}^{(s)} x_i x_j + \sum_{i\in V, j\in O} \beta_{ij}^{(s)} x_i x_j$
- $\mathcal{F}^{(s)}$ in an upper triangular matrix form: $\mathbf{F}^{(s)} = \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$
- $\mathcal{P} = \mathcal{F} \circ \mathbf{S}$ also in matrix form:

$$\begin{aligned}
\mathbf{P}^{(s)} = \mathbf{S}^\top \mathbf{F}^{(s)} \mathbf{S} \quad &= \quad \text{(we are looking for an equivalent key)} \\
&= \mathbf{S}^\top (\Omega^{-1})^\top \, \Omega^\top \mathbf{F}^{(s)} \Omega \, \Omega^{-1} \mathbf{S} \\
&= \underbrace{\left(\mathbf{S}^\top (\Omega^{-1})^\top\right)}_{\begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}^\top} \circ \underbrace{\left(\Omega^\top \mathbf{F}^{(s)} \Omega\right)}_{\begin{pmatrix} \omega_1^\top & \omega_3^\top \\ 0 & \omega_4^\top \end{pmatrix} \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \omega_1 & 0 \\ \omega_3 & \omega_4 \end{pmatrix}} \circ \underbrace{\left(\Omega^{-1} \mathbf{S}\right)}_{\begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}} \\
&= \begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}^\top \begin{pmatrix} \mathbf{F'}_1^{(s)} & \mathbf{F'}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix} \quad \text{- An equivalent key!}
\end{aligned}$$

Equivalent keys for UOV

- Central map: $\mathcal{F}^{(s)}(x_1, \ldots, x_n) = \sum\limits_{i,j \in V, i \leqslant j} \alpha_{ij}^{(s)} x_i x_j + \sum\limits_{i \in V, j \in O} \beta_{ij}^{(s)} x_i x_j$
- $\mathcal{F}^{(s)}$ in an upper triangular matrix form: $\mathbf{F}^{(s)} = \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$
- $\mathcal{P} = \mathcal{F} \circ \mathbf{S}$ also in matrix form:

$$\begin{aligned}
\mathbf{P}^{(s)} = \mathbf{S}^\top \mathbf{F}^{(s)} \mathbf{S} &= \quad \text{(we are looking for an equivalent key)} \\
&= \mathbf{S}^\top (\Omega^{-1})^\top \, \Omega^\top \mathbf{F}^{(s)} \Omega \, \Omega^{-1} \mathbf{S} \\
&= \left(\mathbf{S}^\top (\Omega^{-1})^\top\right) \circ \left(\Omega^\top \mathbf{F}^{(s)} \Omega\right) \circ \left(\Omega^{-1} \mathbf{S}\right) \\
&= \overbrace{\begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}^\top}\overbrace{\begin{pmatrix} \omega_1^\top & \omega_3^\top \\ 0 & \omega_4^\top \end{pmatrix} \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \omega_1 & 0 \\ \omega_3 & \omega_4 \end{pmatrix}}\overbrace{\begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}} \\
&= \begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}^\top \begin{pmatrix} \mathbf{F'}_1^{(s)} & \mathbf{F'}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix} \quad \text{- An equivalent key!}
\end{aligned}$$

Equivalent keys for UOV

- Central map: $\mathcal{F}^{(s)}(x_1,\ldots,x_n) = \sum_{i,j\in V, i\leqslant j} \alpha_{ij}^{(s)} x_i x_j + \sum_{i\in V, j\in O} \beta_{ij}^{(s)} x_i x_j$
- $\mathcal{F}^{(s)}$ in an upper triangular matrix form: $\mathbf{F}^{(s)} = \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$
- $\mathcal{P} = \mathcal{F} \circ \mathbf{S}$ also in matrix form:

$$
\begin{aligned}
\mathbf{P}^{(s)} = \mathbf{S}^\top \mathbf{F}^{(s)} \mathbf{S} &= \quad \text{(we are looking for an equivalent key)} \\
&= \mathbf{S}^\top (\Omega^{-1})^\top \; \Omega^\top \mathbf{F}^{(s)} \Omega \; \Omega^{-1} \mathbf{S} \\
&= \left(\mathbf{S}^\top (\Omega^{-1})^\top\right) \circ \left(\Omega^\top \mathbf{F}^{(s)} \Omega\right) \circ \left(\Omega^{-1} \mathbf{S}\right) \\
&= \overbrace{\begin{pmatrix} \mathrm{I}_v & \mathrm{S}_1 \\ 0 & \mathrm{I}_m \end{pmatrix}^\top}\overbrace{\begin{pmatrix} \omega_1^\top & \omega_3^\top \\ 0 & \omega_4^\top \end{pmatrix} \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \omega_1 & 0 \\ \omega_3 & \omega_4 \end{pmatrix}}\overbrace{\begin{pmatrix} \mathrm{I}_v & \mathrm{S}_1 \\ 0 & \mathrm{I}_m \end{pmatrix}} \\
&= \begin{pmatrix} \mathrm{I}_v & \mathrm{S}_1 \\ 0 & \mathrm{I}_m \end{pmatrix}^\top \begin{pmatrix} \mathbf{F}_1'^{(s)} & \mathbf{F}_2'^{(s)} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \mathrm{I}_v & \mathrm{S}_1 \\ 0 & \mathrm{I}_m \end{pmatrix} \quad \text{- An equivalent key!}
\end{aligned}
$$

Equivalent keys for UOV

- Central map: $\mathcal{F}^{(s)}(x_1, \ldots, x_n) = \sum_{i,j \in V, i \leqslant j} \alpha_{ij}^{(s)} x_i x_j + \sum_{i \in V, j \in O} \beta_{ij}^{(s)} x_i x_j$
- $\mathcal{F}^{(s)}$ in an upper triangular matrix form: $\mathbf{F}^{(s)} = \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$
- $\mathcal{P} = \mathcal{F} \circ \mathbf{S}$ also in matrix form:

$$\begin{aligned}
\mathbf{P}^{(s)} = \mathbf{S}^\top \mathbf{F}^{(s)} \mathbf{S} &= \quad \text{(we are looking for an equivalent key)} \\
&= \mathbf{S}^\top (\Omega^{-1})^\top \, \Omega^\top \mathbf{F}^{(s)} \Omega \, \Omega^{-1} \mathbf{S} \\
&= \left(\mathbf{S}^\top (\Omega^{-1})^\top\right) \circ \left(\Omega^\top \mathbf{F}^{(s)} \Omega\right) \circ \left(\Omega^{-1} \mathbf{S}\right) \\
&= \overbrace{\begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}^\top}\ \overbrace{\begin{pmatrix} \omega_1^\top & \omega_3^\top \\ 0 & \omega_4^\top \end{pmatrix} \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \omega_1 & 0 \\ \omega_3 & \omega_4 \end{pmatrix}}\ \overbrace{\begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}} \\
&= \begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}^\top \begin{pmatrix} \mathbf{F}'^{(s)}_1 & \mathbf{F}'^{(s)}_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix} \quad \text{- An equivalent key!}
\end{aligned}$$

Equivalent keys for UOV

- Central map: $\mathcal{F}^{(s)}(x_1,\ldots,x_n) = \sum_{i,j\in V, i\leqslant j} \alpha_{ij}^{(s)} x_i x_j + \sum_{i\in V, j\in O} \beta_{ij}^{(s)} x_i x_j$
- $\mathcal{F}^{(s)}$ in an upper triangular matrix form: $\mathbf{F}^{(s)} = \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$
- $\mathcal{P} = \mathcal{F} \circ \mathbf{S}$ also in matrix form:

$$
\begin{aligned}
\mathbf{P}^{(s)} = \mathbf{S}^\top \mathbf{F}^{(s)} \mathbf{S} &= \quad \text{(we are looking for an equivalent key)} \\
&= \mathbf{S}^\top (\Omega^{-1})^\top \, \Omega^\top \mathbf{F}^{(s)} \Omega \, \Omega^{-1} \mathbf{S} \\
&= \left(\mathbf{S}^\top (\Omega^{-1})^\top\right) \circ \left(\Omega^\top \mathbf{F}^{(s)} \Omega\right) \circ \left(\Omega^{-1} \mathbf{S}\right) \\
&= \overbrace{\begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}^\top} \overbrace{\begin{pmatrix} \omega_1^\top & \omega_3^\top \\ 0 & \omega_4^\top \end{pmatrix} \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \omega_1 & 0 \\ \omega_3 & \omega_4 \end{pmatrix}} \overbrace{\begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}} \\
&= \begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}^\top \begin{pmatrix} \mathbf{F'}_1^{(s)} & \mathbf{F'}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I}_v & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix} \quad \text{- An equivalent key!}
\end{aligned}
$$

Key generation for UOV using equivalent keys

- From the key equation $\mathcal{P} = \mathcal{F} \circ \mathbf{S}$ in matrix form: $\mathbf{P}^{(s)} = \mathbf{S}^\top \mathbf{F}^{(s)} \mathbf{S}$:

$$\begin{pmatrix} \mathbf{P}_1^{(s)} & \mathbf{P}_2^{(s)} \\ \mathbf{0} & \mathbf{P}_4^{(s)} \end{pmatrix} = \texttt{Upper}\left(\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{S}_1^\top & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \right)$$
$$= \begin{pmatrix} \mathbf{F}_1^{(s)} & (\mathbf{F}_1^{(s)} + \mathbf{F}_1^{(s)\top})\mathbf{S}_1 + \mathbf{F}_2^{(s)} \\ \mathbf{0} & \texttt{Upper}\,(\mathbf{S}_1^\top \mathbf{F}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{pmatrix}.$$

- The **standard key generation (not using equivalent keys)** would
  - Expand from secret seed $\mathbf{F}_1^{(s)}, \mathbf{F}_2^{(s)}$ and $\mathbf{S}_1$
  - Calculate

$$\begin{aligned} \mathbf{P}_1^{(s)} &= \mathbf{F}_1^{(s)} \quad \text{and} \\ \mathbf{P}_2^{(s)} &= (\mathbf{P}_1^{(s)} + \mathbf{P}_1^{(s)\top})\mathbf{S}_1 + \mathbf{F}_2^{(s)} \quad \text{and} \\ \mathbf{P}_4^{(s)} &= \texttt{Upper}\,(\mathbf{S}_1^\top \mathbf{F}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{aligned}$$

  - Note immediatelly that we must have a public $\mathbf{F}_1^{(s)}$

Key generation for UOV using equivalent keys

- From the key equation $\mathcal{P} = \mathcal{F} \circ \mathbf{S}$ in matrix form: $\mathbf{P}^{(s)} = \mathbf{S}^\top \mathbf{F}^{(s)} \mathbf{S}$:

$$\begin{pmatrix} \mathbf{P}_1^{(s)} & \mathbf{P}_2^{(s)} \\ \mathbf{0} & \mathbf{P}_4^{(s)} \end{pmatrix} = \mathtt{Upper}\left(\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{S}_1^\top & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \right)$$
$$= \begin{pmatrix} \mathbf{F}_1^{(s)} & (\mathbf{F}_1^{(s)} + \mathbf{F}_1^{(s)\top})\mathbf{S}_1 + \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathtt{Upper}\ (\mathbf{S}_1^\top \mathbf{F}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{pmatrix}.$$

- The **standard key generation (not using equivalent keys)** would
  - Expand from secret seed $\mathbf{F}_1^{(s)}, \mathbf{F}_2^{(s)}$ and $\mathbf{S}_1$
  - Calculate

$$\begin{aligned} \mathbf{P}_1^{(s)} &= \mathbf{F}_1^{(s)} \quad \text{and} \\ \mathbf{P}_2^{(s)} &= (\mathbf{P}_1^{(s)} + \mathbf{P}_1^{(s)\top})\mathbf{S}_1 + \mathbf{F}_2^{(s)} \quad \text{and} \\ \mathbf{P}_4^{(s)} &= \mathtt{Upper}\ (\mathbf{S}_1^\top \mathbf{F}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{aligned}$$

  - Note immediatelly that we must have a public $\mathbf{F}_1^{(s)}$

Key generation for UOV using equivalent keys

- From the key equation $\mathcal{P} = \mathcal{F} \circ \mathbf{S}$ in matrix form: $\mathbf{P}^{(s)} = \mathbf{S}^\top \mathbf{F}^{(s)} \mathbf{S}$:

$$\begin{pmatrix} \mathbf{P}_1^{(s)} & \mathbf{P}_2^{(s)} \\ \mathbf{0} & \mathbf{P}_4^{(s)} \end{pmatrix} = \mathtt{Upper}\left(\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{S}_1^\top & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \right)$$
$$= \begin{pmatrix} \mathbf{F}_1^{(s)} & (\mathbf{F}_1^{(s)} + \mathbf{F}_1^{(s)\top})\mathbf{S}_1 + \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathtt{Upper}\ (\mathbf{S}_1^\top \mathbf{F}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{pmatrix}.$$

- The **standard key generation (not using equivalent keys)** would
  - Expand from secret seed $\mathbf{F}_1^{(s)}, \mathbf{F}_2^{(s)}$ and $\mathbf{S}_1$
  - Calculate

$$\begin{aligned} \mathbf{P}_1^{(s)} &= \mathbf{F}_1^{(s)} \quad \text{and} \\ \mathbf{P}_2^{(s)} &= (\mathbf{P}_1^{(s)} + \mathbf{P}_1^{(s)\top})\mathbf{S}_1 + \mathbf{F}_2^{(s)} \quad \text{and} \\ \mathbf{P}_4^{(s)} &= \mathtt{Upper}\ (\mathbf{S}_1^\top \mathbf{F}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{aligned}$$

  - Note immediatelly that we must have a public $\mathbf{F}_1^{(s)}$

Key generation for UOV using equivalent keys

- From the key equation $\mathcal{P} = \mathcal{F} \circ \mathbf{S}$ in matrix form: $\mathbf{P}^{(s)} = \mathbf{S}^\top \mathbf{F}^{(s)} \mathbf{S}$:

$$\begin{aligned}\begin{pmatrix} \mathbf{P}_1^{(s)} & \mathbf{P}_2^{(s)} \\ \mathbf{0} & \mathbf{P}_4^{(s)} \end{pmatrix} &= \texttt{Upper}\left(\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{S}_1^\top & \mathbf{I} \end{pmatrix}\begin{pmatrix} \mathbf{F}_1^{(s)} & \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{I} & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I} \end{pmatrix}\right) \\ &= \begin{pmatrix} \mathbf{F}_1^{(s)} & (\mathbf{F}_1^{(s)} + \mathbf{F}_1^{(s)\top})\mathbf{S}_1 + \mathbf{F}_2^{(s)} \\ \mathbf{0} & \texttt{Upper}\ (\mathbf{S}_1^\top \mathbf{F}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{pmatrix}.\end{aligned}$$

- The **standard key generation (not using equivalent keys)** would
  - Expand from secret seed $\mathbf{F}_1^{(s)}, \mathbf{F}_2^{(s)}$ and $\mathbf{S}_1$
  - Calculate

$$\begin{aligned}\mathbf{P}_1^{(s)} &= \mathbf{F}_1^{(s)} \quad \text{and} \\ \mathbf{P}_2^{(s)} &= (\mathbf{P}_1^{(s)} + \mathbf{P}_1^{(s)\top})\mathbf{S}_1 + \mathbf{F}_2^{(s)} \quad \text{and} \\ \mathbf{P}_4^{(s)} &= \texttt{Upper}\ (\mathbf{S}_1^\top \mathbf{F}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)})\end{aligned}$$

  - Note immediatelly that we must have a public $\mathbf{F}_1^{(s)}$

Key generation for UOV using equivalent keys

- Again the key equation

$$\begin{pmatrix} \mathbf{P}_1^{(s)} & \mathbf{P}_2^{(s)} \\ \mathbf{0} & \mathbf{P}_4^{(s)} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_1^{(s)} & (\mathbf{F}_1^{(s)} + \mathbf{F}_1^{(s)\top})\mathbf{S}_1 + \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathtt{Upper}\ (\mathbf{S}_1^\top \mathbf{F}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{pmatrix}.$$

- Note immediatelly that we must have $\mathbf{F}_1^{(s)}$
- **The new key generation (using equivalent keys)**
  - Expands from secret seed: $\mathbf{S}_1$ and from public seed: $\mathbf{P}_1^{(s)}, \mathbf{P}_2^{(s)}$
  - Calculate

$$\begin{aligned} \mathbf{F}_1^{(s)} &= \mathbf{P}_1^{(s)} \quad \text{and} \\ \mathbf{F}_2^{(s)} &= \mathbf{P}_2^{(s)} - (\mathbf{P}_1^{(s)} + \mathbf{P}_1^{(s)\top})\mathbf{S}_1 \quad \text{and} \\ \mathbf{P}_4^{(s)} &= \mathtt{Upper}\ (\mathbf{S}_1^\top \mathbf{P}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{aligned}$$

- Only $\mathbf{P}_4^{(s)}$ needs to be stored as **non-compressible public key**
- for UOV parameters, more than 5/6 reduction of public key

Key generation for UOV using equivalent keys

- Again the key equation

$$\begin{pmatrix} \mathbf{P}_1^{(s)} & \mathbf{P}_2^{(s)} \\ \mathbf{0} & \mathbf{P}_4^{(s)} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_1^{(s)} & (\mathbf{F}_1^{(s)} + \mathbf{F}_1^{(s)\top})\mathbf{S}_1 + \mathbf{F}_2^{(s)} \\ \mathbf{0} & \mathtt{Upper}\ (\mathbf{S}_1^\top \mathbf{F}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{pmatrix}.$$

- Note immediatelly that we must have $\mathbf{F}_1^{(s)}$
- **The new key generation (using equivalent keys)**
  - Expands from secret seed: $\mathbf{S}_1$ and from public seed: $\mathbf{P}_1^{(s)}, \mathbf{P}_2^{(s)}$
  - Calculate

$$\begin{aligned} \mathbf{F}_1^{(s)} &= \mathbf{P}_1^{(s)} \quad \text{and} \\ \mathbf{F}_2^{(s)} &= \mathbf{P}_2^{(s)} - (\mathbf{P}_1^{(s)} + \mathbf{P}_1^{(s)\top})\mathbf{S}_1 \quad \text{and} \\ \mathbf{P}_4^{(s)} &= \mathtt{Upper}\ (\mathbf{S}_1^\top \mathbf{P}_1^{(s)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(s)}) \end{aligned}$$

- Only $\mathbf{P}_4^{(s)}$ needs to be stored as **non-compressible public key**
- for UOV parameters, more than 5/6 reduction of public key

LUOV [Beullens et al. '17]

- **Lifting of coefficients + key generation with equivalent keys**
  - Coefficient live in ground field, but polynomials and solutions live in extension field
  - Significant reduction in key sizes
  - NIST Second round candidate
  - Unfortunately, proven insecure by Ding et al.'19

MAYO [Beullens '21]

- Submitted to NIST in additional signature round
- Currently, one of the most promising candidates!
- **UOV with small oil space + key generation with equivalent keys**
- 'Whipping' technique to expand the oil space so that signing is possible
  - Various approaches for whipping possible
  - Not yet well understood? More research necessary

The $\mathcal{MQ}$ problem

The MQ problem

Computational MQ problem

Given: m multivariate polynomials $p_1, p_2, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ of degree 2

Find: (if any) a vector $(u_1, \ldots, u_n) \in \mathbb{F}_q^n$ such that

$$\begin{cases} p_1(u_1, \ldots, u_n) = & 0 \\ p_2(u_1, \ldots, u_n) = & 0 \\ \quad \ldots & \\ p_m(u_1, \ldots, u_n) = & 0 \end{cases}$$

How hard is it actually?

- **Easy** when $m >$ number of monomials of degree 2
  - linearize and solve as a system of linear equations
- hardest case $n \approx m$
- Complexity well understood for "random" systems (correct: systems without structure)
  - Gröbner bases, XL, Joux-Vitse algorithms

Computational MQ problem

Given: m multivariate polynomials $p_1, p_2, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ of degree 2

Find: (if any) a vector $(u_1, \ldots, u_n) \in \mathbb{F}_q^n$ such that

$$\begin{cases} p_1(u_1, \ldots, u_n) = & 0 \\ p_2(u_1, \ldots, u_n) = & 0 \\ \quad \ldots & \\ p_m(u_1, \ldots, u_n) = & 0 \end{cases}$$

How hard is it actually?

- **Easy** when $m >$ number of monomials of degree 2
  - linearize and solve as a system of linear equations
- hardest case $n \approx m$
- Complexity well understood for "random" systems (correct: systems without structure)
  - Gröbner bases, XL, Joux-Vitse algorithms

Computational MQ problem

Given: m multivariate polynomials $p_1, p_2, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ of degree 2

Find: (if any) a vector $(u_1, \ldots, u_n) \in \mathbb{F}_q^n$ such that

$$\begin{cases} p_1(u_1, \ldots, u_n) = & 0 \\ p_2(u_1, \ldots, u_n) = & 0 \\ \quad \ldots & \\ p_m(u_1, \ldots, u_n) = & 0 \end{cases}$$

How hard is it actually?

- **Easy** when $m >$ number of monomials of degree 2
  - linearize and solve as a system of linear equations
- hardest case $n \approx m$
- Complexity well understood for "random" systems (correct: systems without structure)
  - Gröbner bases, XL, Joux-Vitse algorithms

The MQ problem

Computational MQ problem

Given: m multivariate polynomials $p_1, p_2, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ of degree 2

Find: (if any) a vector $(u_1, \ldots, u_n) \in \mathbb{F}_q^n$ such that

$$\begin{cases} p_1(u_1, \ldots, u_n) = & 0 \\ p_2(u_1, \ldots, u_n) = & 0 \\ \quad \ldots & \\ p_m(u_1, \ldots, u_n) = & 0 \end{cases}$$

How hard is it actually?

- **Easy** when $m >$ number of monomials of degree 2
  - linearize and solve as a system of linear equations
- hardest case $n \approx m$
- Complexity well understood for "random" systems (correct: systems without structure)
  - Gröbner bases, XL, Joux-Vitse algorithms

The MQ problem

Computational MQ problem

Given: m multivariate polynomials $p_1, p_2, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ of degree 2

Find: (if any) a vector $(u_1, \ldots, u_n) \in \mathbb{F}_q^n$ such that

$$\begin{cases} p_1(u_1, \ldots, u_n) = & 0 \\ p_2(u_1, \ldots, u_n) = & 0 \\ \quad \ldots & \\ p_m(u_1, \ldots, u_n) = & 0 \end{cases}$$

How hard is it actually?

- **Easy** when $m >$ number of monomials of degree 2
  - linearize and solve as a system of linear equations
- hardest case $n \approx m$
- Complexity well understood for "random" systems (correct: systems without structure)
  - Gröbner bases, XL, Joux-Vitse algorithms

Computational MQ problem

Given: m multivariate polynomials $p_1, p_2, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ of degree 2

Find: (if any) a vector $(u_1, \ldots, u_n) \in \mathbb{F}_q^n$ such that

$$\begin{cases} p_1(u_1, \ldots, u_n) = & 0 \\ p_2(u_1, \ldots, u_n) = & 0 \\ \quad \ldots & \\ p_m(u_1, \ldots, u_n) = & 0 \end{cases}$$

How hard is it actually?

- **Easy** when $m >$ number of monomials of degree 2
    - linearize and solve as a system of linear equations
- hardest case $n \approx m$
- Complexity well understood for "random" systems (correct: systems without structure)
    - Gröbner bases, XL, Joux-Vitse algorithms

The MQ problem

Computational MQ problem

Given: m multivariate polynomials $p_1, p_2, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ of degree 2

Find: (if any) a vector $(u_1, \ldots, u_n) \in \mathbb{F}_q^n$ such that

$$\begin{cases} p_1(u_1, \ldots, u_n) = & 0 \\ p_2(u_1, \ldots, u_n) = & 0 \\ \quad \ldots & \\ p_m(u_1, \ldots, u_n) = & 0 \end{cases}$$

How hard is it actually?

- **Easy** when $m >$ number of monomials of degree 2
  - linearize and solve as a system of linear equations
- hardest case $n \approx m$
- Complexity well understood for "random" systems (correct: systems without structure)
  - Gröbner bases, XL, Joux-Vitse algorithms

Security of MQ cryptosystems

- If the MQ problem can be solved, MQ cryptosystems can be broken
- not the right direction of reduction, does not say much about the security. . .
- General MQ system solvers provide nevertheless crude upper security bound
- Generic algebraic system solvers
  - Gröbner bases solvers - F4/F5 algorithms
  - XL algorithms
  - Joux-Vitse algorithm
- Probabilistic algorithms
  - Lokshtanov et al.

Security of MQ cryptosystems

- If the MQ problem can be solved, MQ cryptosystems can be broken
- not the right direction of reduction, does not say much about the security...
- General MQ system solvers provide nevertheless crude upper security bound
- Generic algebraic system solvers
  - Gröbner bases solvers - F4/F5 algorithms
  - XL algorithms
  - Joux-Vitse algorithm
- Probabilistic algorithms
  - Lokshtanov et al.

Security of MQ cryptosystems

- If the MQ problem can be solved, MQ cryptosystems can be broken
- not the right direction of reduction, does not say much about the security...
- General MQ system solvers provide nevertheless crude upper security bound
- Generic algebraic system solvers
  - Gröbner bases solvers - F4/F5 algorithms
  - XL algorithms
  - Joux-Vitse algorithm
- Probabilistic algorithms
  - Lokshtanov et al.

Security of MQ cryptosystems

- If the MQ problem can be solved, MQ cryptosystems can be broken
- not the right direction of reduction, does not say much about the security...
- General MQ system solvers provide nevertheless crude upper security bound
- Generic algebraic system solvers
  - Gröbner bases solvers - F4/F5 algorithms
  - XL algorithms
  - Joux-Vitse algorithm
- Probabilistic algorithms
  - Lokshtanov et al.

Security of MQ cryptosystems

- If the MQ problem can be solved, MQ cryptosystems can be broken
- not the right direction of reduction, does not say much about the security...
- General MQ system solvers provide nevertheless crude upper security bound
- Generic algebraic system solvers
  - Gröbner bases solvers - F4/F5 algorithms
  - XL algorithms
  - Joux-Vitse algorithm
- Probabilistic algorithms
  - Lokshtanov et al.

General principle of algebraic system solvers

- We want to solve

$$\begin{cases} p_1(x_1, \ldots, x_n) = & 0 \\ \quad \ldots & \\ p_m(x_1, \ldots, x_n) = & 0 \end{cases}$$

  over the field $\mathbb{F}_q$,
- For simplicity, suppose there is a unique solution $(u_1, u_2, \ldots, u_n)$.
- In $\mathbb{F}_q[x_1, \ldots, x_n]/\langle x_1^q - x_1, \ldots, x_n^q - x_n\rangle$ this means that $(x_1 - u_1, x_2 - u_2, \ldots, x_n - u_n)$ generates the same space (the same ideal) as the polynomials in the above system
- $\Rightarrow (x_1 - u_1, x_2 - u_2, \ldots, x_n - u_n)$ is a **basis** of the ideal
- $\Rightarrow$ There exist polynomials $h_i, i \in \{1, \ldots, n\}$ such that $x_j - u_j = \sum_{i=1}^{n} h_i p_i$

In a nutshell, the goal of an algebraic solver is to find a "nice" basis of the given ideal

- **by finding the right linear combinations** $\sum_{i=1}^{n} h_i p_i$
- main tool is **linear algebra**

General principle of algebraic system solvers

- We want to solve

$$\begin{cases} p_1(x_1,\ldots,x_n) = & 0 \\ \quad\ldots & \\ p_m(x_1,\ldots,x_n) = & 0 \end{cases}$$

  over the field $\mathbb{F}_q$,
- For simplicity, suppose there is a unique solution $(u_1, u_2, \ldots, u_n)$.
- In $\mathbb{F}_q[x_1,\ldots,x_n]/\langle x_1^q - x_1, \ldots, x_n^q - x_n\rangle$ this means that $(x_1 - u_1, x_2 - u_2, \ldots, x_n - u_n)$ generates the same space (the same ideal) as the polynomials in the above system
- $\Rightarrow (x_1 - u_1, x_2 - u_2, \ldots, x_n - u_n)$ is a **basis** of the ideal
- $\Rightarrow$ There exist polynomials $h_i, i \in \{1,\ldots,n\}$ such that $x_j - u_j = \sum_{i=1}^{n} h_i p_i$

In a nutshell, the goal of an algebraic solver is to find a "nice" basis of the given ideal

- **by finding the right linear combinations** $\sum_{i=1}^{n} h_i p_i$
- main tool is **linear algebra**

General principle of algebraic system solvers

- We want to solve

$$\begin{cases} p_1(x_1, \dots, x_n) = & 0 \\ \quad \dots & \\ p_m(x_1, \dots, x_n) = & 0 \end{cases}$$

  over the field $\mathbb{F}_q$,
- For simplicity, suppose there is a unique solution $(u_1, u_2, \dots, u_n)$.
- In $\mathbb{F}_q[x_1, \dots, x_n]/\langle x_1^q - x_1, \dots, x_n^q - x_n \rangle$ this means that $(x_1 - u_1, x_2 - u_2, \dots, x_n - u_n)$ generates the same space (the same ideal) as the polynomials in the above system
- $\Rightarrow (x_1 - u_1, x_2 - u_2, \dots, x_n - u_n)$ is a **basis** of the ideal
- $\Rightarrow$ There exist polynomials $h_i, i \in \{1, \dots, n\}$ such that $x_j - u_j = \sum_{i=1}^{n} h_i p_i$

In a nutshell, the goal of an algebraic solver is to find a "nice" basis of the given ideal

- **by finding the right linear combinations** $\sum_{i=1}^{n} h_i p_i$
- main tool is **linear algebra**

General principle of algebraic system solvers

- We want to solve

$$\begin{cases} p_1(x_1,\ldots,x_n) = & 0 \\ \quad \ldots & \\ p_m(x_1,\ldots,x_n) = & 0 \end{cases}$$

  over the field $\mathbb{F}_q$,
- For simplicity, suppose there is a unique solution $(u_1, u_2, \ldots, u_n)$.
- In $\mathbb{F}_q[x_1,\ldots,x_n]/\langle x_1^q - x_1, \ldots, x_n^q - x_n\rangle$ this means that $(x_1 - u_1, x_2 - u_2, \ldots, x_n - u_n)$ generates the same space (the same ideal) as the polynomials in the above system
- $\Rightarrow (x_1 - u_1, x_2 - u_2, \ldots, x_n - u_n)$ is a **basis** of the ideal
- $\Rightarrow$ There exist polynomials $h_i, i \in \{1,\ldots,n\}$ such that $x_j - u_j = \sum_{i=1}^{n} h_i p_i$

In a nutshell, the goal of an algebraic solver is to find a "nice" basis of the given ideal

- **by finding the right linear combinations** $\sum_{i=1}^{n} h_i p_i$
- main tool is **linear algebra**

General principle of algebraic system solvers

- We want to solve

$$\begin{cases} p_1(x_1, \ldots, x_n) = & 0 \\ \quad \ldots & \\ p_m(x_1, \ldots, x_n) = & 0 \end{cases}$$

  over the field $\mathbb{F}_q$,
- For simplicity, suppose there is a unique solution $(u_1, u_2, \ldots, u_n)$.
- In $\mathbb{F}_q[x_1, \ldots, x_n]/\langle x_1^q - x_1, \ldots, x_n^q - x_n\rangle$ this means that $(x_1 - u_1, x_2 - u_2, \ldots, x_n - u_n)$ generates the same space (the same ideal) as the polynomials in the above system
- $\Rightarrow (x_1 - u_1, x_2 - u_2, \ldots, x_n - u_n)$ is a **basis** of the ideal
- $\Rightarrow$ There exist polynomials $h_i, i \in \{1, \ldots, n\}$ such that $x_j - u_j = \sum_{i=1}^n h_i p_i$

In a nutshell, the goal of an algebraic solver is to find a "nice" basis of the given ideal

- **by finding the right linear combinations $\sum_{i=1}^n h_i p_i$**
- main tool is **linear algebra**

General principle of algebraic system solvers

- We want to solve

$$\begin{cases} p_1(x_1, \ldots, x_n) = & 0 \\ \quad \ldots & \\ p_m(x_1, \ldots, x_n) = & 0 \end{cases}$$

  over the field $\mathbb{F}_q$,
- For simplicity, suppose there is a unique solution $(u_1, u_2, \ldots, u_n)$.
- In $\mathbb{F}_q[x_1, \ldots, x_n]/\langle x_1^q - x_1, \ldots, x_n^q - x_n\rangle$ this means that $(x_1 - u_1, x_2 - u_2, \ldots, x_n - u_n)$ generates the same space (the same ideal) as the polynomials in the above system
- $\Rightarrow$ $(x_1 - u_1, x_2 - u_2, \ldots, x_n - u_n)$ is a **basis** of the ideal
- $\Rightarrow$ There exist polynomials $h_i, i \in \{1, \ldots, n\}$ such that $x_j - u_j = \sum_{i=1}^{n} h_i p_i$

In a nutshell, the goal of an algebraic solver is to find a "nice" basis of the given ideal

- **by finding the right linear combinations** $\sum_{i=1}^{n} h_i p_i$
- main tool is **linear algebra**

- How to find all these linear combinations?
- Form a **(Macaulay) matrix** with coefficients equal to the coefficients of the polynomials
- rows correspond to $mon \cdot p_i$, for all possible monomials mon in $x_1, \ldots, x_n$ up to degree $D-2$
- columns correspond to all possible monomials up to degree D

$$\begin{array}{c}
 \\ p_1 \\ p_2 \\ x_1p_1 \\ x_1p_2 \\ \ldots \\ x_8p_1
\end{array}
\begin{array}{c}
\begin{array}{cccccc} x_6x_7x_8 & \quad & \quad & x_2 & x_1 & 1 \end{array} \\
\begin{pmatrix}
1 & & & 1 & 0 & 0 \\
0 & & & 1 & 1 & 0 \\
1 & & \ldots & 1 & 0 & 1 \\
0 & & & 1 & 0 & 0 \\
\ldots & & & \ldots & & \\
0 & & & 0 & 1 & 0
\end{pmatrix}
\end{array}$$

- Try to Gauss-reduce the matrix
- If it does not reduce to the “nice” form, increase D
- The complexity is determined by the size of the matrix and the lowest D that works

General principle of algebraic system solvers

- How to find all these linear combinations?
- Form a **(Macaulay) matrix** with coefficients equal to the coefficients of the polynomials
- rows correspond to $mon \cdot p_i$, for all possible monomials mon in $x_1, \ldots, x_n$ up to degree $D-2$
- columns correspond to all possible monomials up to degree D

$$
\begin{array}{c}
\\
p_1 \\ p_2 \\ x_1 p_1 \\ x_1 p_2 \\ \ldots \\ x_8 p_1
\end{array}
\begin{array}{c}
\begin{array}{cccccc}
x_6x_7x_8 & \quad & \quad & x_2 & x_1 & 1
\end{array} \\
\begin{pmatrix}
1 & & & 1 & 0 & 0 \\
0 & & & 1 & 1 & 0 \\
1 & & \ldots & 1 & 0 & 1 \\
0 & & & 1 & 0 & 0 \\
\ldots & & & \ldots & & \\
0 & & & 0 & 1 & 0
\end{pmatrix}
\end{array}
$$

- Try to Gauss-reduce the matrix
- If it does not reduce to the "nice" form, increase D
- The complexity is determined by the size of the matrix and the lowest D that works

General principle of algebraic system solvers

- How to find all these linear combinations?
- Form a **(Macaulay) matrix** with coefficients equal to the coefficients of the polynomials
- rows correspond to $mon \cdot p_i$, for all possible monomials mon in $x_1, \ldots, x_n$ up to degree $D-2$
- columns correspond to all possible monomials up to degree D

$$
\begin{array}{c c}
 & \begin{array}{cccccc} x_6x_7x_8 & & & x_2 & x_1 & 1 \end{array} \\
\begin{array}{c} p_1 \\ p_2 \\ x_1p_1 \\ x_1p_2 \\ \ldots \\ x_8p_1 \end{array} &
\begin{pmatrix}
1 & & & 1 & 0 & 0 \\
0 & & & 1 & 1 & 0 \\
1 & & \ldots & 1 & 0 & 1 \\
0 & & & 1 & 0 & 0 \\
\ldots & & & \ldots & & \\
0 & & & 0 & 1 & 0
\end{pmatrix}
\end{array}
$$

- Try to Gauss-reduce the matrix
- If it does not reduce to the “nice” form, increase D
- The complexity is determined by the size of the matrix and the lowest D that works

General principle of algebraic system solvers

- How to find all these linear combinations?
- Form a **(Macaulay) matrix** with coefficients equal to the coefficients of the polynomials
- rows correspond to $mon \cdot p_i$, for all possible monomials mon in $x_1, \ldots, x_n$ up to degree $D-2$
- columns correspond to all possible monomials up to degree D

$$
\begin{array}{c}
\\ p_1 \\ p_2 \\ x_1 p_1 \\ x_1 p_2 \\ \ldots \\ x_8 p_1
\end{array}
\begin{array}{c}
\begin{array}{cccccc} x_6x_7x_8 & \quad & \quad & x_2 & x_1 & 1 \end{array} \\
\begin{pmatrix}
1 & & & 1 & 0 & 0 \\
0 & & & 1 & 1 & 0 \\
1 & & \ldots & 1 & 0 & 1 \\
0 & & & 1 & 0 & 0 \\
\ldots & & & \ldots & & \\
0 & & & 0 & 1 & 0
\end{pmatrix}
\end{array}
$$

- Try to Gauss-reduce the matrix
- If it does not reduce to the "nice" form, increase D
- The complexity is determined by the size of the matrix and the lowest D that works

General principle of algebraic system solvers

- How to find all these linear combinations?
- Form a **(Macaulay) matrix** with coefficients equal to the coefficients of the polynomials
- rows correspond to $mon \cdot p_i$, for all possible monomials mon in $x_1, \ldots, x_n$ up to degree $D-2$
- columns correspond to all possible monomials up to degree D

$$
\begin{array}{c} \\ p_1 \\ p_2 \\ x_1p_1 \\ x_1p_2 \\ \ldots \\ x_8p_1 \end{array}
\begin{array}{c}
\begin{array}{cccccc} x_6x_7x_8 & & & x_2 & x_1 & 1 \end{array} \\
\begin{pmatrix}
1 & & & 1 & 0 & 0 \\
0 & & & 1 & 1 & 0 \\
1 & & \ldots & 1 & 0 & 1 \\
0 & & & 1 & 0 & 0 \\
\ldots & & & \ldots & & \\
0 & & & 0 & 1 & 0
\end{pmatrix}
\end{array}
$$

- Try to Gauss-reduce the matrix
- If it does not reduce to the "nice" form, increase D
- The complexity is determined by the size of the matrix and the lowest D that works

General principle of algebraic system solvers

- Of course, the best algorithms are more sophisticated...
- Some techniques include:
  - don't start over from scratch, but reuse some useful results from the previous interaction
  - don't add rows that are linearly dependent
  - estimate in advance d_{reg}, and Gauss eliminate only Macaulay matrix of this degree
    - benefits: no operations are performed twice + sparse linear algebra can be used
  - choose the best ordering of monomials
  - enumerate (brute-force) a few variables, and solve all systems of fewer variables
- State of the art
  - Gröbner bases solvers - F4/F5 algorithm, XL algorithm (big fields)
  - Joux-Vitse algorithm ($\mathbb{F}_2$) - we zoom into this one

General principle of algebraic system solvers

- Of course, the best algorithms are more sophisticated. . .
- Some techniques include:
  - don't start over from scratch, but reuse some useful results from the previous interaction
  - don't add rows that are linearly dependent
  - estimate in advance d_{reg}, and Gauss eliminate only Macaulay matrix of this degree
    - benefits: no operations are performed twice + sparse linear algebra can be used
  - choose the best ordering of monomials
  - enumerate (brute-force) a few variables, and solve all systems of fewer variables
- State of the art
  - Gröbner bases solvers - F4/F5 algorithm, XL algorithm (big fields)
  - Joux-Vitse algorithm ($\mathbb{F}_2$) - we zoom into this one

General principle of algebraic system solvers

- Of course, the best algorithms are more sophisticated. . .
- Some techniques include:
  - don't start over from scratch, but reuse some useful results from the previous interaction
  - don't add rows that are linearly dependent
  - estimate in advance d_{reg}, and Gauss eliminate only Macaulay matrix of this degree
    - benefits: no operations are performed twice + sparse linear algebra can be used
  - choose the best ordering of monomials
  - enumerate (brute-force) a few variables, and solve all systems of fewer variables
- State of the art
  - Gröbner bases solvers - F4/F5 algorithm, XL algorithm (big fields)
  - Joux-Vitse algorithm ($\mathbb{F}_2$) - we zoom into this one

General principle of algebraic system solvers

- Of course, the best algorithms are more sophisticated...
- Some techniques include:
  - don't start over from scratch, but reuse some useful results from the previous interaction
  - don't add rows that are linearly dependent
  - estimate in advance d_{reg}, and Gauss eliminate only Macaulay matrix of this degree
    - benefits: no operations are performed twice + sparse linear algebra can be used
  - choose the best ordering of monomials
  - enumerate (brute-force) a few variables, and solve all systems of fewer variables
- State of the art
  - Gröbner bases solvers - F4/F5 algorithm, XL algorithm (big fields)
  - Joux-Vitse algorithm ($\mathbb{F}_2$) - we zoom into this one

General principle of algebraic system solvers

- Of course, the best algorithms are more sophisticated...
- Some techniques include:
  - don't start over from scratch, but reuse some useful results from the previous interaction
  - don't add rows that are linearly dependent
  - estimate in advance d_{reg}, and Gauss eliminate only Macaulay matrix of this degree
    - benefits: no operations are performed twice + sparse linear algebra can be used
  - choose the best ordering of monomials
  - enumerate (brute-force) a few variables, and solve all systems of fewer variables
- State of the art
  - Gröbner bases solvers - F4/F5 algorithm, XL algorithm (big fields)
  - Joux-Vitse algorithm ($\mathbb{F}_2$) - we zoom into this one

General principle of algebraic system solvers

- Of course, the best algorithms are more sophisticated. . .
- Some techniques include:
  - don't start over from scratch, but reuse some useful results from the previous interaction
  - don't add rows that are linearly dependent
  - estimate in advance d_{reg}, and Gauss eliminate only Macaulay matrix of this degree
    - benefits: no operations are performed twice + sparse linear algebra can be used
  - choose the best ordering of monomials
  - enumerate (brute-force) a few variables, and solve all systems of fewer variables
- State of the art
  - Gröbner bases solvers - F4/F5 algorithm, XL algorithm (big fields)
  - Joux-Vitse algorithm ($\mathbb{F}_2$) - we zoom into this one

General principle of algebraic system solvers

- Of course, the best algorithms are more sophisticated...
- Some techniques include:
  - don't start over from scratch, but reuse some useful results from the previous interaction
  - don't add rows that are linearly dependent
  - estimate in advance d_{reg}, and Gauss eliminate only Macaulay matrix of this degree
    - benefits: no operations are performed twice + sparse linear algebra can be used
  - choose the best ordering of monomials
  - enumerate (brute-force) a few variables, and solve all systems of fewer variables
- State of the art
  - Gröbner bases solvers - F4/F5 algorithm, XL algorithm (big fields)
  - Joux-Vitse algorithm ($\mathbb{F}_2$) - we zoom into this one

General principle of algebraic system solvers

- Of course, the best algorithms are more sophisticated...
- Some techniques include:
  - don't start over from scratch, but reuse some useful results from the previous interaction
  - don't add rows that are linearly dependent
  - estimate in advance d_{reg}, and Gauss eliminate only Macaulay matrix of this degree
    - benefits: no operations are performed twice + sparse linear algebra can be used
  - choose the best ordering of monomials
  - enumerate (brute-force) a few variables, and solve all systems of fewer variables
- State of the art
  - Gröbner bases solvers - F4/F5 algorithm, XL algorithm (big fields)
  - Joux-Vitse algorithm ($\mathbb{F}_2$) - we zoom into this one

Gröbner bases algorithms

- First studied by Bruno Buchberger in the '60-es
- later improved by Faugère et al. (F4, F5) in '04
- looks for a nice basis of the ideal generated by the polynomial system
- Considers a specific monomial ordering - best for grevlex ordering
- Complexity for semi-regular systems ("random looking")
- F5 does not generate "rows" that are linearly dependent ("no reduction to zero")

$$\mathcal{O}\left(\binom{n+d_{\mathrm{reg}}-1}{d_{\mathrm{reg}}}^{\omega}\right)$$

- Hybrid F5 algorithm [Bettale, Faugère, and Perret '09] - fix k variables for some optimal k

$$\min_k q^k \mathcal{O}\left(\binom{n-k+d_{\mathrm{reg}}-1}{d_{\mathrm{reg}}}^{\omega}\right)$$

Gröbner bases algorithms

- First studied by Bruno Buchberger in the '60-es
- later improved by Faugère et al. (F4, F5) in '04
- looks for a nice basis of the ideal generated by the polynomial system
- Considers a specific monomial ordering - best for grevlex ordering
- Complexity for semi-regular systems ("random looking")
- F5 does not generate "rows" that are linearly dependent ("no reduction to zero")

$$\mathcal{O}\left(\binom{n+d_{\mathrm{reg}}-1}{d_{\mathrm{reg}}}^{\omega}\right)$$

- Hybrid F5 algorithm [Bettale, Faugère, and Perret '09] - fix k variables for some optimal k

$$\min_{k} q^{k}\mathcal{O}\left(\binom{n-k+d_{\mathrm{reg}}-1}{d_{\mathrm{reg}}}^{\omega}\right)$$

Gröbner bases algorithms

- First studied by Bruno Buchberger in the '60-es
- later improved by Faugère et al. (F4, F5) in '04
- looks for a nice basis of the ideal generated by the polynomial system
- Considers a specific monomial ordering - best for grevlex ordering
- Complexity for semi-regular systems ("random looking")
- F5 does not generate "rows" that are linearly dependent ("no reduction to zero")

$$\mathcal{O}\left(\binom{n+d_{\mathrm{reg}}-1}{d_{\mathrm{reg}}}^{\omega}\right)$$

- Hybrid F5 algorithm [Bettale, Faugère, and Perret '09] - fix k variables for some optimal k

$$\min_k q^k \mathcal{O}\left(\binom{n-k+d_{\mathrm{reg}}-1}{d_{\mathrm{reg}}}^{\omega}\right)$$

The XL algorithm

- Proposed by Courtois et al. '00
- Several variants - FXL (fixing variables), MutantXL
- Basically also a Gröbner basis algorithm
  - Took years to establish the equivalence
- . . . but much simpler presentation and analysis
- Main steps of the algorithm:
  1. **eXtend** - form Macaulay matrix of dergee D
  2. **Linearize** - Apply Gaussian Elimination on the extended system to generate a univariate polynomial p (the ordering should be such that all terms in one variable (ex. x_1) are eliminated last)
  3. **Solve** - Use Berlekamps algorithm to find roots of the polynomial p
  4. **Repeat** - Substitute the solution of p into the system and continue with the simplified system
- Complexity:

$$3 \cdot \binom{n + d_{\mathrm{XL}}}{d_{\mathrm{XL}}}^2 \cdot \binom{n}{d}$$

The XL algorithm

- Proposed by Courtois et al. '00
- Several variants - FXL (fixing variables), MutantXL
- Basically also a Gröbner basis algorithm
  - Took years to establish the equivalence
- ... but much simpler presentation and analysis
- Main steps of the algorithm:
  1. **eXtend** - form Macaulay matrix of dergee D
  2. **Linearize** - Apply Gaussian Elimination on the extended system to generate a univariate polynomial p (the ordering should be such that all terms in one variable (ex. x_1) are eliminated last)
  3. **Solve** - Use Berlekamps algorithm to find roots of the polynomial p
  4. **Repeat** - Substitute the solution of p into the system and continue with the simplified system
- Complexity:

$$3 \cdot \binom{n + d_{\mathrm{XL}}}{d_{\mathrm{XL}}}^2 \cdot \binom{n}{d}$$

The XL algorithm

- Proposed by Courtois et al. '00
- Several variants - FXL (fixing variables), MutantXL
- Basically also a Gröbner basis algorithm
  - Took years to establish the equivalence
- ... but much simpler presentation and analysis
- Main steps of the algorithm:
  1. **eXtend** - form Macaulay matrix of dergee D
  2. **Linearize** - Apply Gaussian Elimination on the extended system to generate a univariate polynomial p (the ordering should be such that all terms in one variable (ex. x_1) are eliminated last)
  3. **Solve** - Use Berlekamps algorithm to find roots of the polynomial p
  4. **Repeat** - Substitute the solution of p into the system and continue with the simplified system
- Complexity:

$$3 \cdot \binom{n + d_{\mathrm{XL}}}{d_{\mathrm{XL}}}^2 \cdot \binom{n}{d}$$

The XL algorithm

- Proposed by Courtois et al. '00
- Several variants - FXL (fixing variables), MutantXL
- Basically also a Gröbner basis algorithm
  - Took years to establish the equivalence
- ... but much simpler presentation and analysis
- Main steps of the algorithm:
  1. **eXtend** - form Macaulay matrix of dergee D
  2. **Linearize** - Apply Gaussian Elimination on the extended system to generate a univariate polynomial p (the ordering should be such that all terms in one variable (ex. x_1) are eliminated last)
  3. **Solve** - Use Berlekamps algorithm to find roots of the polynomial p
  4. **Repeat** - Substitute the solution of p into the system and continue with the simplified system
- Complexity:

$$3 \cdot \binom{n + d_{\mathrm{XL}}}{d_{\mathrm{XL}}}^2 \cdot \binom{n}{d}$$

The Joux-Vitse algorithm

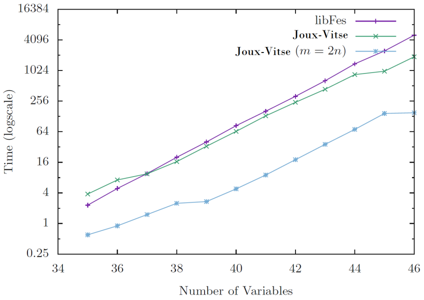
- Proposed by Joux and Vitse in 2017
- Very similar to FXL but with a very clever approach to fixing
- Significantly improves over other algorithms in the practical regime
- Assymptotically it is actually the same as other Gröbner basis algorithms
- Currently the best approach for small fields
- For $\mathbb{F}_2$ beats enumeration at $n = 37$, other algorithms around $n = 200$

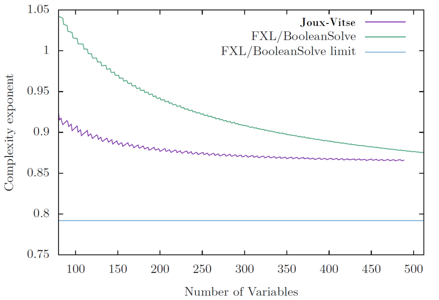
The Joux-Vitse algorithm

- Proposed by Joux and Vitse in 2017
- Very similar to FXL but with a very clever approach to fixing
- Significantly improves over other algorithms in the practical regime
- Assymptotically it is actually the same as other Gröbner basis algorithms
- Currently the best approach for small fields
- For $\mathbb{F}_2$ beats enumeration at $n = 37$, other algorithms around $n = 200$

The Joux-Vitse algorithm

- Proposed by Joux and Vitse in 2017
- Very similar to FXL but with a very clever approach to fixing
- Significantly improves over other algorithms in the practical regime
- Assymptotically it is actually the same as other Gröbner basis algorithms
- Currently the best approach for small fields
- For $\mathbb{F}_2$ beats enumeration at $n = 37$, other algorithms around $n = 200$





An example

Consider the following system:

$$
\begin{aligned}
x_1x_2 + x_1x_3 + x_2x_3 + x_1 + x_3 &= 0 \\
x_1x_3 + x_2x_3 + x_3x_4 + x_2 + x_3 + x_4 &= 0 \\
x_2x_4 + x_3x_4 + x_1 + x_3 + 1 &= 0 \\
x_1x_2 + x_1x_4 + x_2x_3 + x_3 + x_4 + 1 &= 0 \\
x_2x_3 + x_3x_4 + x_1 + x_3 + x_4 &= 0
\end{aligned}
$$

The Macaulay matrix of degree 2 (for lexicographic ordering) is

$$
\begin{array}{c}
\\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5
\end{array}
\begin{array}{c}
\begin{array}{ccccccccccc}
x_1x_2 & x_1x_3 & x_1x_4 & x_1 & x_2x_3 & x_2x_4 & x_2 & x_3x_4 & x_3 & x_4 & 1
\end{array} \\
\begin{pmatrix}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\end{array}
\sim
\begin{pmatrix}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}
$$

Last equation is: $x_2x_3 + x_2x_4 + x_4 + 1 = 0$ - we removed one variable (but this is not enough)

An example

Consider the following system:

$$\begin{aligned}
x_1x_2 + x_1x_3 + x_2x_3 + x_1 + x_3 &= 0\\
x_1x_3 + x_2x_3 + x_3x_4 + x_2 + x_3 + x_4 &= 0\\
x_2x_4 + x_3x_4 + x_1 + x_3 + 1 &= 0\\
x_1x_2 + x_1x_4 + x_2x_3 + x_3 + x_4 + 1 &= 0\\
x_2x_3 + x_3x_4 + x_1 + x_3 + x_4 &= 0
\end{aligned}$$

The Macaulay matrix of degree 2 (for lexicographic ordering) is

$$\begin{array}{cc}
 & \begin{array}{ccccccccccc} x_1x_2 & x_1x_3 & x_1x_4 & x_1 & x_2x_3 & x_2x_4 & x_2 & x_3x_4 & x_3 & x_4 & 1 \end{array}\\
\begin{array}{c} p_1\\ p_2\\ p_3\\ p_4\\ p_5 \end{array} & \begin{pmatrix}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1\\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0\\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1\\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1\\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\end{array}
\sim
\begin{pmatrix}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1\\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0\\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1\\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}$$

Last equation is: $x_2x_3 + x_2x_4 + x_4 + 1 = 0$ - we removed one variable (but this is not enough)

An example

- The above matrix had $\binom{4}{2} + 4 + 1 = 11$ columns and 5 rows
- If we make the degree 3 Macaulay matrix we will have $\binom{4}{3} + \binom{4}{2} + 4 + 1 = 15$ columns and $4 \cdot 5 + 5 = 25$ rows
- Gauss elimination will certainly give us a solution since we have an overdetermined system
- Downside - we needed to make a bigger matrix
  - For example for $n = m = 20$ we get 1351 columns and 400 rows - already a huge matrix, but unfortunatelly can't be echelonized to give us a unique solution.
  - We need an even bigger matrix of degree 4
- **In general, we need a big enough degree D to get more (independent) rows than columns**

An example

- The above matrix had $\binom{4}{2} + 4 + 1 = 11$ columns and 5 rows
- If we make the degree 3 Macaulay matrix we will have $\binom{4}{3} + \binom{4}{2} + 4 + 1 = 15$ columns and $4 \cdot 5 + 5 = 25$ rows
- Gauss elimination will certainly give us a solution since we have an overdetermined system
- Downside - we needed to make a bigger matrix
  - For example for $n = m = 20$ we get 1351 columns and 400 rows - already a huge matrix, but unfortunatelly can't be echelonized to give us a unique solution.
  - We need an even bigger matrix of degree 4
- **In general, we need a big enough degree D to get more (independent) rows than columns**

Towards analysis of algebraic solvers

- Let $T_D = \binom{n+D}{D}$ - the number of monomials of degree at most D
- $N_D = T_D$ - columns in Macaulay matrix
- $R_D = mT_{D-2}$ - rows in Macaulay matrix
- Previous example suggests we need D such that: $R_D \geqslant N_D$
- Sort of ... What if some rows are linearly dependent?
  - Are there always such dependencies?
  - How to find them and count them?
- These dependencies/relations are called "syzygies"
- In general they are hard to find for a given system, unless there is no hidden structure, i.e. the system is **semi-regular**
- Semi-regular overdetermined systems - no other syzigies but the trivial ones $f_i f_j - f_j f_i = 0$ exist
  - We need to remove the syzigies
  - We can Gauss-reduce the Macaulay matrix as soon as the dimension of the row space is larger than N_D

Towards analysis of algebraic solvers

- Let $T_D = \binom{n+D}{D}$ - the number of monomials of degree at most D
- $N_D = T_D$ - columns in Macaulay matrix
- $R_D = mT_{D-2}$ - rows in Macaulay matrix
- Previous example suggests we need D such that: $R_D \geqslant N_D$
- Sort of ... What if some rows are linearly dependent?
  - Are there always such dependencies?
  - How to find them and count them?
- These dependencies/relations are called "syzygies"
- In general they are hard to find for a given system, unless there is no hidden structure, i.e. the system is **semi-regular**
- Semi-regular overdetermined systems - no other syzigies but the trivial ones $f_i f_j - f_j f_i = 0$ exist
  - We need to remove the syzigies
  - We can Gauss-reduce the Macaulay matrix as soon as the dimension of the row space is larger than N_D

Towards analysis of algebraic solvers

- Let $T_D = \binom{n+D}{D}$ - the number of monomials of degree at most D
- $N_D = T_D$ - columns in Macaulay matrix
- $R_D = mT_{D-2}$ - rows in Macaulay matrix
- Previous example suggests we need D such that: $R_D \geqslant N_D$
- Sort of ... What if some rows are linearly dependent?
  - Are there always such dependencies?
  - How to find them and count them?
- These dependencies/relations are called "syzygies"
- In general they are hard to find for a given system, unless there is no hidden structure, i.e. the system is **semi-regular**
- Semi-regular overdetermined systems - no other syzigies but the trivial ones $f_i f_j - f_j f_i = 0$ exist
  - We need to remove the syzigies
  - We can Gauss-reduce the Macaulay matrix as soon as the dimension of the row space is larger than N_D

Towards analysis of algebraic solvers

- Let $T_D = \binom{n+D}{D}$ - the number of monomials of degree at most D
- $N_D = T_D$ - columns in Macaulay matrix
- $R_D = mT_{D-2}$ - rows in Macaulay matrix
- Previous example suggests we need D such that: $R_D \geqslant N_D$
- Sort of ... What if some rows are linearly dependent?
  - Are there always such dependencies?
  - How to find them and count them?
- These dependencies/relations are called "syzygies"
- In general they are hard to find for a given system, unless there is no hidden structure, i.e. the system is **semi-regular**
- Semi-regular overdetermined systems - no other syzigies but the trivial ones $f_i f_j - f_j f_i = 0$ exist
  - We need to remove the syzigies
  - We can Gauss-reduce the Macaulay matrix as soon as the dimension of the row space is larger than N_D

Towards analysis of algebraic solvers

We can use **generating functions** to analyze this.

- Let $[t^D]$ denote the coefficient in front of t^D
- $T_D = [t^D]\frac{1}{(1-t)^{n+1}}$ - the number of monomials of degree at most D
- $N_D = T_D$ - columns in Macaulay matrix
- $I_D = [t^D]\frac{1-(1-t^2)^m}{(1-t)^{n+1}}$ - independent rows in Macaulay matrix
- Now, condition for full rank N_D of degree D Macaulay matrix becomes: $[t^D](T_D - I_D) < 0$, i.e.

$$[t^D]\frac{(1-t^2)^m}{(1-t)^{n+1}} < 0$$

- Hence, we need to calculate the smallest D that satisfies this condition and use degree D Macaulay matrix to solve the system

Towards analysis of algebraic solvers

We can use **generating functions** to analyze this.

- Let $[t^D]$ denote the coefficient in front of t^D
- $T_D = [t^D]\frac{1}{(1-t)^{n+1}}$ - the number of monomials of degree at most D
- $N_D = T_D$ - columns in Macaulay matrix
- $I_D = [t^D]\frac{1-(1-t^2)^m}{(1-t)^{n+1}}$ - independent rows in Macaulay matrix
- Now, condition for full rank N_D of degree D Macaulay matrix becomes: $[t^D](T_D - I_D) < 0$, i.e.

$$[t^D]\frac{(1-t^2)^m}{(1-t)^{n+1}} < 0$$

- Hence, we need to calculate the smallest D that satisfies this condition and use degree D Macaulay matrix to solve the system

Back to our example

The Macaulay matrix of degree 2 (for lexicographic ordering) is

$$
\begin{array}{c}
\\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5
\end{array}
\begin{array}{c}
\begin{array}{ccccccccccc}
\mathbf{x_1x_2} & \mathbf{x_1x_3} & x_1x_4 & x_1 & \mathbf{x_2x_3} & x_2x_4 & x_2 & x_3x_4 & x_3 & x_4 & 1
\end{array} \\
\begin{pmatrix}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\end{array}
$$

- Call the columns $\mathbf{x_1x_2}$, $\mathbf{x_1x_3}$ and $\mathbf{x_2x_3}$ - matrix M'
- If we remove M', the rest is bilinear in x_1, x_2, x_3 and x_4
- If we fix x_4 we obtain a linear system in x_1, x_2, x_3
- Hence, if we find at least 3 vectors in the kernel of the matrix M' we can use these
  1. to trasform the Macaulay matrix to one that has M' removed and has at least 3 rows
  2. to enumerate over all values for x_4
  3. to solve a linear system in x_1, x_2, x_3

These are basically the steps of the Joux-Vitse algorithm!

Back to our example

The Macaulay matrix of degree 2 (for lexicographic ordering) is

$$\begin{array}{cc} & \begin{array}{ccccccccccc} \mathbf{x_1x_2} & \mathbf{x_1x_3} & x_1x_4 & x_1 & \mathbf{x_2x_3} & x_2x_4 & x_2 & x_3x_4 & x_3 & x_4 & 1 \end{array} \\ \begin{array}{c} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} & \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

- Call the columns $\mathbf{x_1x_2}$, $\mathbf{x_1x_3}$ and $\mathbf{x_2x_3}$ - matrix M'
- If we remove M', the rest is bilinear in x_1, x_2, x_3 and x_4
- If we fix x_4 we obtain a linear system in x_1, x_2, x_3
- Hence, if we find at least 3 vectors in the kernel of the matrix M' we can use these
  1. to trasform the Macaulay matrix to one that has M' removed and has at least 3 rows
  2. to enumerate over all values for x_4
  3. to solve a linear system in x_1, x_2, x_3

These are basically the steps of the Joux-Vitse algorithm!

Back to our example

The Macaulay matrix of degree 2 (for lexicographic ordering) is

$$
\begin{array}{c c}
 & \begin{array}{ccccccccccc} \mathbf{x_1x_2} & \mathbf{x_1x_3} & x_1x_4 & x_1 & \mathbf{x_2x_3} & x_2x_4 & x_2 & x_3x_4 & x_3 & x_4 & 1 \end{array} \\
\begin{array}{c} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} &
\begin{pmatrix}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\end{array}
$$

- Call the columns $\mathbf{x_1x_2}$, $\mathbf{x_1x_3}$ and $\mathbf{x_2x_3}$ - matrix M'
- If we remove M', the rest is bilinear in x_1, x_2, x_3 and x_4
- If we fix x_4 we obtain a linear system in x_1, x_2, x_3
- Hence, if we find at least 3 vectors in the kernel of the matrix M' we can use these
  1. to trasform the Macaulay matrix to one that has M' removed and has at least 3 rows
  2. to enumerate over all values for x_4
  3. to solve a linear system in x_1, x_2, x_3

These are basically the steps of the Joux-Vitse algorithm!

The Joux-Vitse algorithm - informal description

For appropriately chosen degree D Macaulay matrix $\mathcal{M}$:

1. Take M' to be the matrix of columns of $\mathcal{M}$ that correspond to monomials of $deg > 1$ in the first k variables
2. Find k independent vectors in the kernel of M'
3. Multiply these vectors by $\mathcal{M}$ to obtain a matrix $\mathcal{M}'$
4. For each possible value of the last $n - k$ variables form a linear system from $\mathcal{M}'$. If it has a solution, output it as the solution to the given system