Winter School, PQC 2016, Fukuoka

Multivariate Public Key Cryptography

Jintai Ding

University of Cincinnati

Feb. 22 2016

Outline

Outline

What is a MPKC?

- Multivariate Public Key Cryptosystems
  - *Cryptosystems, whose public keys are a set of multivariate polynomials*

What is a MPKC?

- Multivariate Public Key Cryptosystems
  - *Cryptosystems, whose public keys are a set of multivariate polynomials*
- The public key is given as:

$$G(x_1, ..., x_n) = (G_1(x_1, ..., x_n), ..., G_m(x_1, ..., x_n)).$$

Here the $G_i(x_1, ..., x_n)$ are multivariate polynomials over a finite field.

Encryption

- Any plaintext $M = (x'_1, ..., x'_n)$ has the ciphertext:

$$G(M) = G(x'_1, ..., x'_n) = (y'_1, ..., y'_m).$$

Encryption

- Any plaintext $M = (x_1', ..., x_n')$ has the ciphertext:

$$G(M) = G(x_1', ..., x_n') = (y_1', ..., y_m').$$

- To decrypt the ciphertext $(y_1', ..., y_n')$, one needs to know a secret (**the secret key**), so that one can invert the map: G^{-1} to find the plaintext $(x_1', ..., x_n')$.

$$M = (x_1', ..., x_n') = G^{-1}(y_1', ..., y_m').$$

Toy example

- We use the finite field $k = GF[2]/(x^2 + x + 1)$ with 2^2 elements.

Toy example

- We use the finite field $k = GF[2]/(x^2 + x + 1)$ with 2^2 elements.
- We denote the elements of the field by the set $\{0, 1, 2, 3\}$ to simplify the notation.
  Here 0 represent the 0 in k, 1 for 1, 2 for x, and 3 for $1 + x$.
  In this case, $1 + 3 = 2$ and $2 \cdot 3 = 1$. $2 \cdot 2 = 3$ and $3 \cdot 3 = 2$.

A toy example

$$
\begin{aligned}
G_0(x_1, x_2, x_3) &= 1 + x_2 + 2x_0x_2 + 3x_1^2 + 3x_1x_2 + x_2^2 \\
G_1(x_1, x_2, x_3) &= 1 + 3x_0 + 2x_1 + x_2 + x_0^2 + x_0x_1 + 3x_0x_2 + x_1^2 \\
G_2(x_1, x_2, x_3) &= 3x_2 + x_0^2 + 3x_1^2 + x_1x_2 + 3x_2^2
\end{aligned}
$$

A toy example

- $$\begin{aligned} G_0(x_1, x_2, x_3) = & \quad 1 + x_2 + 2x_0x_2 + 3x_1^2 + 3x_1x_2 + x_2^2 \\ G_1(x_1, x_2, x_3) = & \quad 1 + 3x_0 + 2x_1 + x_2 + x_0^2 + x_0x_1 + 3x_0x_2 + x_1^2 \\ G_2(x_1, x_2, x_3) = & \quad 3x_2 + x_0^2 + 3x_1^2 + x_1x_2 + 3x_2^2 \end{aligned}$$

- For example, if the plaintext is: $x_0 = 1$, $x_1 = 2$, $x_2 = 3$, then we can plug into G_1, G_2 and G_3 to get the ciphertext $y_0 = 0$, $y_1 = 0$, $y_2 = 1$.

A toy example

$$
\begin{aligned}
G_0(x_1, x_2, x_3) &= 1 + x_2 + 2x_0x_2 + 3x_1^2 + 3x_1x_2 + x_2^2 \\
G_1(x_1, x_2, x_3) &= 1 + 3x_0 + 2x_1 + x_2 + x_0^2 + x_0x_1 + 3x_0x_2 + x_1^2 \\
G_2(x_1, x_2, x_3) &= 3x_2 + x_0^2 + 3x_1^2 + x_1x_2 + 3x_2^2
\end{aligned}
$$

- For example, if the plaintext is: $x_0 = 1$, $x_1 = 2$, $x_2 = 3$, then we can plug into G_1, G_2 and G_3 to get the ciphertext $y_0 = 0$, $y_1 = 0$, $y_2 = 1$.
- This is a bijective map and we can invert it easily. This example is based on the Matsumoto-Imai cryptosystem.

Signature

- To sign the document hash value $(y'_1, ..., y'_m)$, one needs to know (**the secret key**), so that one can invert the public key map: G^{-1} to find the signature $(x'_1, ..., x'_n)$.

$$S = (x'_1, ..., x'_n) = G^{-1}(y'_1, ..., y'_m).$$

Signature

- To sign the document hash value $(y'_1, ..., y'_m)$, one needs to know (**the secret key**), so that one can invert the public key map: G^{-1} to find the signature $(x'_1, ..., x'_n)$.

$$S = (x'_1, ..., x'_n) = G^{-1}(y'_1, ..., y'_m).$$

- Given the pair:$((x'_1, ..., x'_n)(y'_1, ..., y'_m))$, anyone can verify the validity of the signature by checking if the following equality holds:

$$G(x'_1, ..., x'_n) = (y'_1, ..., y'_m).$$

Theoretical Foundation

- Direct attack is to solve the set of equations:

$$G(M) = G(x_1, ..., x_n) = (y'_1, ..., y'_m).$$

Theoretical Foundation

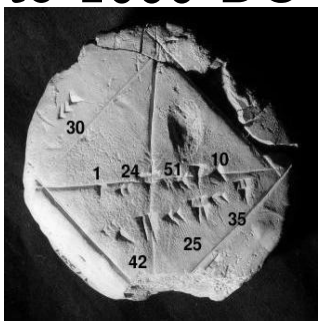
- Direct attack is to solve the set of equations:

$$G(M) = G(x_1, ..., x_n) = (y'_1, ..., y'_m).$$

- *- Solving a set of n randomly chosen equations (nonlinear) with n variables is NP-complete, though this does not necessarily ensure the security of the systems.*

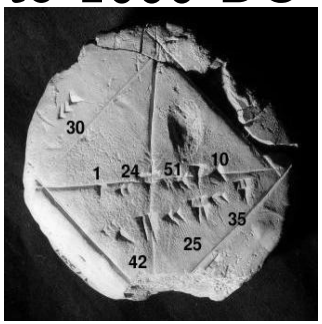
A quick historic overview

- Single variable quadratic equation – Babylonian around 1800 to 1600 BC



A quick historic overview

- Single variable quadratic equation – Babylonian around 1800 to 1600 BC

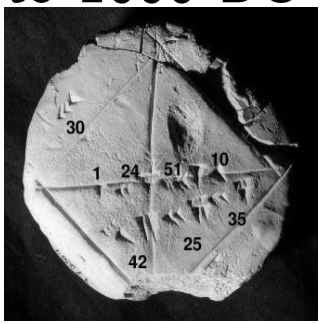


- Cubic and quartic equation – around 1500

Tartaglia  Cardano 

A quick historic overview

- Single variable quadratic equation – Babylonian around 1800 to 1600 BC



- Cubic and quartic equation – around 1500

Tartaglia  Cardano 

- Multivariate system– 1964-1965
  Buchberger : Gröobner Basis
  Hironaka: Standard basis

The hardness of the problem

- Single variable case – Galois's work.

Newton method – continuous system
Berlekamp's algorithm – finite field and low degree

The hardness of the problem

- Single variable case – Galois's work.

  Newton method – continuous system
  Berlekamp's algorithm – finite field and low degree
- Multivariate case: NP- hardness of the generic systems.
  Numerical solvers – continuous systems
  Finite field case

Quadratic Constructions

- *1) Efficiency considerations lead to mainly quadratic constructions.*

$$G_l(x_1, ..x_n) = \sum_{i,j} \alpha_{lij} x_i x_j + \sum_i \beta_{li} x_i + \gamma_l.$$

Quadratic Constructions

- *1) Efficiency considerations lead to mainly quadratic constructions.*

$$G_l(x_1, ..x_n) = \sum_{i,j} \alpha_{lij} x_i x_j + \sum_i \beta_{li} x_i + \gamma_l.$$

- *2) Mathematical structure consideration: Any set of high degree polynomial equations can be reduced to a set of quadratic equations.*

$$x_1 x_2 x_3 = 5,$$

is equivalent to

$$\begin{aligned} x_1 x_2 - y &= 0 \\ y x_3 &= 5. \end{aligned}$$

The view from the history of Mathematics(Diffie in Paris)

- RSA – Number Theory – the 18th century mathematics

The view from the history of Mathematics(Diffie in Paris)

- RSA – Number Theory – the 18th century mathematics
- ECC – Theory of Elliptic Curves – the 19th century mathematics

The view from the history of Mathematics(Diffie in Paris)

- RSA – Number Theory – the 18th century mathematics
- ECC – Theory of Elliptic Curves – the 19th century mathematics
- Multivariate Public key cryptosystem – Algebraic Geometry – the 20th century mathematics
  Algebraic Geometry – Theory of Polynomial Rings

Early works

- Early attempts by Diffie, Fell, Tsujii, Matsumoto, Imai, Ong, Schnorr, Shamir etc

Early works

- Early attempts by Diffie, Fell, Tsujii, Matsumoto, Imai, Ong, Schnorr, Shamir etc
- Fast development in the late 1990s – Patarin'work as catalyst.

Outline

Multivariate Signature schemes

- **Public key**:
  $G(x_1, \ldots, x_n) = (g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$.

Multivariate Signature schemes

- **Public key**:
  $G(x_1, \ldots, x_n) = (g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$.
- **Private key**: a way to compute G^{-1}.

Multivariate Signature schemes

- **Public key**:
  $G(x_1, \ldots, x_n) = (g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$.
- **Private key**: a way to compute G^{-1}.

- **Signing a hash of a document**:

Multivariate Signature schemes

- **Public key**:
  $G(x_1, \ldots, x_n) = (g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$.
- **Private key**: a way to compute G^{-1}.

- **Signing a hash of a document**:
  $(x_1, \ldots, x_n) \in G^{-1}(y_1, \ldots, y_m)$.

Multivariate Signature schemes

- **Public key**:
  $G(x_1, \ldots, x_n) = (g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$.
- **Private key**: a way to compute G^{-1}.

- **Signing a hash of a document**:
  $(x_1, \ldots, x_n) \in G^{-1}(y_1, \ldots, y_m)$.
- **Verifying**: $(y_1, \ldots, y_m) \stackrel{?}{=} G(x_1, \ldots, x_n)$.

  k, a small finite field.

A toy example over GF(3)

$$
\begin{aligned}
G_1(x_1, x_2, x_3) &= 1 + x_3 + x_1x_2 + x_3^2 \\
G_2(x_1, x_2, x_3) &= 2 + x_1 + 2x_2x_3 + x_2 \\
G_3(x_1, x_2, x_3) &= 1 + x_2 + x_1x_3 + x_1^2
\end{aligned}
$$

Hash:

$(y_1, y_2, y_3) = (0, 1, 1).$

A toy example over GF(3)

$$
\begin{aligned}
G_1(x_1, x_2, x_3) &= 1 + x_3 + x_1x_2 + x_3^2 \\
G_2(x_1, x_2, x_3) &= 2 + x_1 + 2x_2x_3 + x_2 \\
G_3(x_1, x_2, x_3) &= 1 + x_2 + x_1x_3 + x_1^2
\end{aligned}
$$

Hash:

$(y_1, y_2, y_3) = (0, 1, 1).$

A signature: $(x_1, x_2, x_3) = (2, 0, 1)$

A toy example over GF(3)

$$
\begin{aligned}
G_1(x_1, x_2, x_3) &= 1 + x_3 + x_1x_2 + x_3^2 \\
G_2(x_1, x_2, x_3) &= 2 + x_1 + 2x_2x_3 + x_2 \\
G_3(x_1, x_2, x_3) &= 1 + x_2 + x_1x_3 + x_1^2
\end{aligned}
$$

Hash:

$(y_1, y_2, y_3) = (0, 1, 1).$

A signature: $(x_1, x_2, x_3) = (2, 0, 1)$

$$
\begin{aligned}
G_1(2, 0, 1) &= 1 + 1 + 2 \times 0 + 1 = 0 \\
G_2(2, 0, 1) &= 2 + 2 + 2 \times 0 \times 1 + 0 = 1 \\
G_3(2, 0, 1) &= 1 + 0 + 2 \times 1 + 1 = 1
\end{aligned}
$$

Security: polynomial solving.

- Signature for $(y_1, y_2, y_3) = (0, 0, 0)$?

Security: polynomial solving.

- Signature for $(y_1, y_2, y_3) = (0, 0, 0)$?

$$\begin{aligned} G_1(x_1, x_2, x_3) &= 1 + x_3 + x_1 x_2 + x_3^2 = 0 \\ G_2(x_1, x_2, x_3) &= 2 + x_1 + 2x_2 x_3 + x_2 = 0 \\ G_3(x_1, x_2, x_3) &= 1 + x_2 + x_1 x_3 + x_1^2 = 0 \end{aligned}$$

Security: polynomial solving.

- Signature for $(y_1, y_2, y_3) = (0, 0, 0)$?

$$\begin{aligned} G_1(x_1, x_2, x_3) &= 1 + x_3 + x_1 x_2 + x_3^2 = 0 \\ G_2(x_1, x_2, x_3) &= 2 + x_1 + 2x_2 x_3 + x_2 = 0 \\ G_3(x_1, x_2, x_3) &= 1 + x_2 + x_1 x_3 + x_1^2 = 0 \end{aligned}$$

- Direct attack: difficulty of solving a set of nonlinear polynomial equations over a finite field.

How to construct G?

- A scheme by Kipnis, Patarin and Goubin 1999. (Eurocrypt 1999)

How to construct G?

- A scheme by Kipnis, Patarin and Goubin 1999. (Eurocrypt 1999)
- $G = F \circ L$.
  F: nonlinear, easy to compute F^{-1}.
  L: invertible linear, to hide the structure of F.

Unbalanced Oil-vinegar (uov) schemes

- $F = (f_1(x_1, .., x_o, x'_1, ..., x'_v), \cdots, f_o(x_1, .., x_o, x'_1, ..., x'_v))$.

Unbalanced Oil-vinegar (uov) schemes

- $F = (f_1(x_1, .., x_o, x'_1, ..., x'_v), \cdots, f_o(x_1, .., x_o, x'_1, ..., x'_v))$.
-

$$f_l(x_1, ., x_o, x'_1, ., x'_v) = \sum a_{lij} x_i x'_j + \sum b_{lij} x'_i x'_j + \sum c_{li} x_i + \sum d_{li} x'_i + e_l$$

Oil variables: $x_1, ..., x_o$.

Vinegar variables: $x'_1, ..., x'_v$.

How to invert F?

$$f_l(x_1, .., x_o, \underbrace{x'_1, .., x'_v}_{\textbf{fix the values}}) =$$

$$\sum a_{lij} x_i x'_j + \sum b_{lij} x'_i x'_j + \sum c_{li} x_i + \sum d_{li} x'_i + e_l.$$

How to invert F?

$$f_l(x_1, .., x_o, x'_1, .., x'_v) =$$
$$\sum a_{lij} x_i x'_j + \sum b_{lij} x'_i x'_j + \sum c_{li} x_i + \sum d_{li} x'_i + e_l.$$

How to invert F?

$$f_l(x_1, .., x_o, x'_1, .., x'_v) =$$
$$\sum a_{lij} x_i x'_j + \sum b_{lij} x'_i x'_j + \sum c_{li} x_i + \sum d_{li} x'_i + e_l.$$

- F: linear in Oil variables: $x_1, .., x_o$.

$\Longrightarrow$ F: easy to invert.

Security analysis

- $v \leq o$ and $v >> o$ not secure

Security analysis

- $v \leq o$ and $v >> o$ not secure
- $v = 2o, 3o$

Security analysis

- $v \leq o$ and $v >> o$ not secure
- $v = 2o, 3o$
- Direct attacks does not work.

Security analysis

- $v \leq o$ and $v >> o$ not secure
- $v = 2o, 3o$
- Direct attacks does not work.
- The mathematical problem to find equivalent secret keys — find the common null subspace spaces of a set of quadratic forms.

Security analysis

- $v \leq o$ and $v >> o$ not secure
- $v = 2o, 3o$
- Direct attacks does not work.
- The mathematical problem to find equivalent secret keys — find the common null subspace spaces of a set of quadratic forms.
- The problem above can also be transformed into solving a set of quadratic equations.

Rainbow – Ding, Schmidtc –2005

- Make F "small" without reducing security.

Rainbow – Ding, Schmidtc –2005

- Make F "small" without reducing security.

$$G = \underbrace{L_1}_{\textbf{Hide the separation}} \circ F \circ \underbrace{L_2}_{\textbf{Hide } L_1 \circ F}.$$

$$F = (F_1, F_2).$$

Rainbow

- Rainbow(18,12,12) over GF(2^8).

  $F_1 : o_1 = 12, v_1 = 18$. 12 OV polynomials:

$$F_1 = (f_1(x_1, ..., x_{30}), ..., f_{12}(x_1, ..., x_{30})).$$

$$\underbrace{x_1, ..., x_{18}}_{Vinegar}, \underbrace{x_{19}, ..., x_{30}}_{Oil}$$

  $F_2 : o_2 = 12, v_2 = 12 + 18 = 30$. 12 OV polynomials:

$$F_2 = (f_{31}(x_1, ..., x_{42}), ..., f_{42}(x_1, ..., x_{42})).$$

$$\underbrace{x_1, ..x_{18}, x_{19}..., x_{30}}_{Vinegar}, \underbrace{x_{31}, ..., x_{42}}_{Oil}$$

Rainbow

- Rainbow(18,12,12)

Rainbow

- Rainbow(18,12,12)
- | **Signature 400 bits** | **Hash 336 bits** |

Implementations

- IC for Rainbow: 804 cycles
  A joint work of Cincinnati and Bochum.(ASAP 2008)

Implementations

- IC for Rainbow: 804 cycles
  A joint work of Cincinnati and Bochum.(ASAP 2008)

- FPGA implementation by the research group of Professor Paar at Bochum (CHES 2009)
  Beat ECC in area and speed.

Side channel attack on Rainbow

- Natural Side channel attack resistance.

Side channel attack on Rainbow

- Natural Side channel attack resistance.
- Further optimizations.

Side channel attack on Rainbow

- Natural Side channel attack resistance.
- Further optimizations.
- Real implementations — works done in Taiwan by Yang, Cheng.

Security

Security

- UOV: not broken since 1999.
- Rainbow – MinRank problem
  MinRank problem – find the (non zero) matrix of the minimum rank in the space spanned by a set of matrices.

Outline

Notation

- k is a small finite field with $|k| = q$

Notation

- k is a small finite field with $|k| = q$
- $\bar{K} = k[x]/(g(x))$, a degree n extension of k and $g(x)$ irreducible of degree n..

Notation

- k is a small finite field with $|k| = q$
- $\bar{K} = k[x]/(g(x))$, a degree n extension of k and $g(x)$ irreducible of degree n..
- The standard k-linear invertible map $\phi : \bar{K} \longrightarrow k^n$, and $\phi^{-1} : k^n \longrightarrow \bar{K}$.

The idea of "BIG" field

- Proposed in 1988 by Matsumoto-Imai.

The idea of "BIG" field

- Proposed in 1988 by Matsumoto-Imai.
- Build up a map F over $\bar{K}$:

$$\bar{F} = L_1 \circ \phi \circ F \circ \phi^{-1} \circ L_2.$$

where the L_i are randomly chosen invertible affine maps over k^n

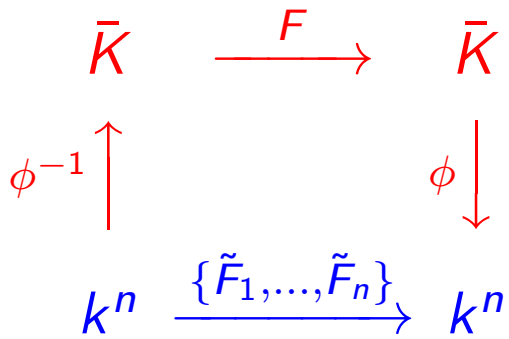
The idea of "BIG" field

- Proposed in 1988 by Matsumoto-Imai.
- Build up a map F over $\bar{K}$:

$$\bar{F} = L_1 \circ \phi \circ F \circ \phi^{-1} \circ L_2.$$

  where the L_i are randomly chosen invertible affine maps over k^n
- The L_i are used to "hide" $\bar{F}$.

Hidden Field Public Key Cryptosystems



Encryption

- The MI construction:

$$F : X \longmapsto X^{q^\theta+1}.$$

Encryption

- The MI construction:

$$F : X \longmapsto X^{q^\theta+1}.$$

- Let $\tilde{F}(x_1, \ldots, x_n) = \phi \circ F \circ \phi^{-1}(x_1, \ldots, x_n) = (\tilde{F}_1, \ldots, \tilde{F}_n)$.

Encryption

- The MI construction:

$$F : X \longmapsto X^{q^\theta+1}.$$

- Let $\tilde{F}(x_1, \ldots, x_n) = \phi \circ F \circ \phi^{-1}(x_1, \ldots, x_n) = (\tilde{F}_1, \ldots, \tilde{F}_n)$.
- The $\tilde{F}_i = \tilde{F}_i(x_1, \ldots, x_n)$ are quadratic polynomials in n variables. Why quadratic?

$$X^{q^\theta+1} = X^{q^\theta} \times X.$$

Decryption

- The condition: $\gcd\left(q^{\theta}+1, q^{n}-1\right)=1$, ensures the invertibility of the map for purposes of decryption. It requires that k must be of characteristic 2.

Decryption

- The condition: $\gcd(q^\theta + 1, q^n - 1) = 1$, ensures the invertibility of the map for purposes of decryption. It requires that k must be of characteristic 2.
- $F^{-1}(X) = X^t$ such that:

$$t \times (q^\theta + 1) \equiv 1 \pmod{q^n - 1}.$$

Decryption

- The condition: $\gcd(q^\theta + 1, q^n - 1) = 1$, ensures the invertibility of the map for purposes of decryption. It requires that k must be of characteristic 2.
- $F^{-1}(X) = X^t$ such that:

$$t \times (q^\theta + 1) \equiv 1 \pmod{q^n - 1}.$$

- The public key includes the field structure of k, θ and $\bar{F} = (\bar{F}_1, .., \bar{F}_n)$. The secret keys are L_1 and L_2.

Decryption

- The condition: $\gcd(q^\theta + 1, q^n - 1) = 1$, ensures the invertibility of the map for purposes of decryption. It requires that k must be of characteristic 2.
- $F^{-1}(X) = X^t$ such that:

$$t \times (q^\theta + 1) \equiv 1 \pmod{q^n - 1}.$$

- The public key includes the field structure of k, θ and $\bar{F} = (\bar{F}_1, .., \bar{F}_n)$. The secret keys are L_1 and L_2.
- The first toy example is produced by setting $n = 3$ and $\theta = 2$.

Decryption

- The condition: $\gcd(q^\theta + 1, q^n - 1) = 1$, ensures the invertibility of the map for purposes of decryption. It requires that k must be of characteristic 2.
- $F^{-1}(X) = X^t$ such that:

$$t \times (q^\theta + 1) \equiv 1 \pmod{q^n - 1}.$$

- The public key includes the field structure of k, θ and $\bar{F} = (\bar{F}_1, .., \bar{F}_n)$. The secret keys are L_1 and L_2.
- The first toy example is produced by setting $n = 3$ and $\theta = 2$.
- This scheme was defeated by linearization equation method by Patarin 1995.

HFE by Patarin etc

- The only difference from MI is that F is replaced by a new map given by:

$$F(X) = \sum_{i,j=0}^{q^i+q^j \leq D} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{q^i \leq D} b_i X^{q^i} + c.$$

- Berlekamp-Massey algorithm to decrypt.
  The complexity $O(D^\omega)$.

HFE by Patarin etc

- The only difference from MI is that F is replaced by a new map given by:

$$F(X) = \sum_{i,j=0}^{q^i+q^j \leq D} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{q^i \leq D} b_i X^{q^i} + c.$$

- Berlekamp-Massey algorithm to decrypt.
  The complexity $O(D^\omega)$.
- Patarin presented two challenges.

Direct Algebraic Attack

Use efficient Gröbner basis (algebraic) algorithms to solve the system of equations:

$$\begin{aligned}
\bar{F}_1(x_1, \ldots, x_n) &= y_1 \\
\bar{F}_2(x_1, \ldots, x_n) &= y_2 \\
&\vdots \\
\bar{F}_n(x_1, \ldots, x_n) &= y_n
\end{aligned}$$

Direct Algebraic Attack

Use efficient Gröbner basis (algebraic) algorithms to solve the system of equations:

$$\begin{aligned}
\bar{F}_1(x_1, \ldots, x_n) &= y_1 \\
\bar{F}_2(x_1, \ldots, x_n) &= y_2 \\
&\vdots \\
\bar{F}_n(x_1, \ldots, x_n) &= y_n
\end{aligned}$$

Direct Algebraic Attack

Algorithm terminates significantly quicker on HFE systems than on random systems. How does the restriction on the degree D of P affect the complexity of algebraic solvers?

- Faugere and Joux broke Challenge 1 with 80 variables and claim Dreg is roughly $log_q(D)$.
- Kipnis-Shamir Minrank attack.
- Granboulan, Joux, Stern (Crypto 2006): If $q = 2$, complexity is quasi-polynomial.

Internal Perturbation

- (Internal) Perturbation was introduced at PKC 2004 as a general method to improve the security of multivariate public key cryptosystems.

Internal Perturbation

- (Internal) Perturbation was introduced at PKC 2004 as a general method to improve the security of multivariate public key cryptosystems.
- Construction – small-scale "noise" is added to the system in a controlled way so as to not fundamentally alter the main structure, but yet substantially increase the "entropy."

Internal Perturbation

- Let r be a small integer and

$$z_1(x_1, \ldots, x_n) = \sum_{j=1}^{n} \alpha_{j1} x_j + \beta_1$$

$$\vdots$$

$$z_r(x_1, \ldots, x_n) = \sum_{j=1}^{n} \alpha_{jr} x_j + \beta_r$$

be a set of randomly chosen affine linear functions in the x_i over k^n such that the $z_j - \beta_j$ are linearly independent.

Internal Perturbation

- Let r be a small integer and

$$z_1(x_1, \ldots, x_n) = \sum_{j=1}^{n} \alpha_{j1} x_j + \beta_1$$

$$\vdots$$

$$z_r(x_1, \ldots, x_n) = \sum_{j=1}^{n} \alpha_{jr} x_j + \beta_r$$

  be a set of randomly chosen affine linear functions in the x_i over k^n such that the $z_j - \beta_j$ are linearly independent.
- We can use these linear functions to create quadratic "perturbation" in HFE (including MI) systems.

IP of MI

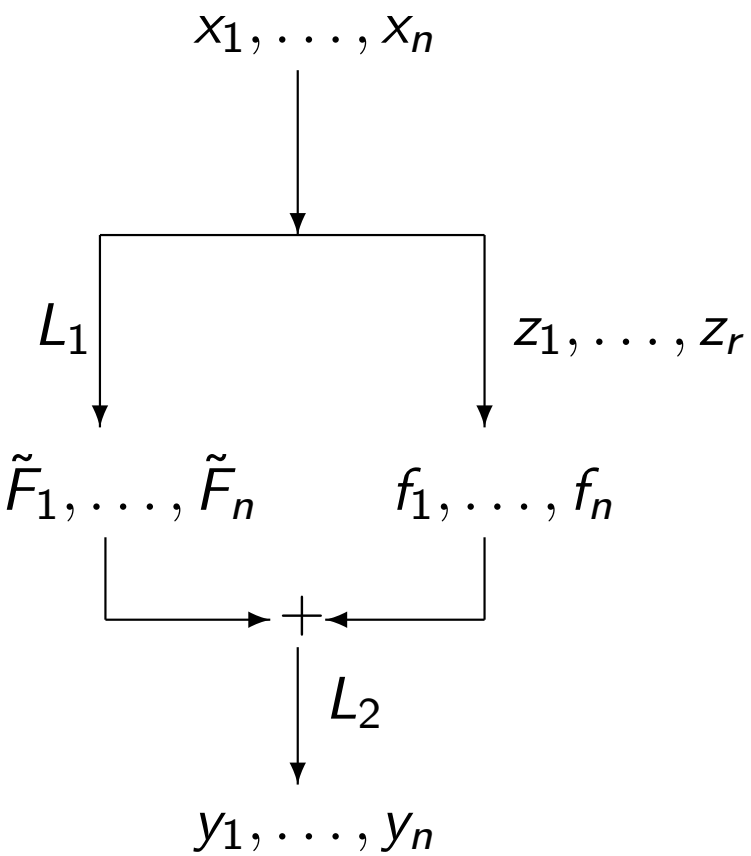


Figure: Structure of Perturbation of the Matsumoto-Imai System.

Decryption

- We need to a search of size of q^r, therefore slower.

Decryption

- We need to a search of size of q^r, therefore slower.
- We need to use Plus Method, **Adding random polynomial**, to help it to resist differential attacks.

Decryption

- We need to a search of size of q^r, therefore slower.
- We need to use Plus Method, **Adding random polynomial**, to help it to resist differential attacks.
- Despite the cost of the search, it is still efficient.

Standing schemes

- PMI+

Standing schemes

- PMI+
- IPHFE+

Standing schemes

- PMI+
- IPHFE+
- HFE Systems of odd characteristics (theoretical support from the view of degree of regularity)

HFEv$^-$ - Key Generation

- finite field $\mathbb{F}$, extension field $\mathbb{E}$ of degree n
- isomorphism $\phi^{-1} : \mathbb{F}^n \to \mathbb{E}$, $\phi(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i \cdot X^{i-1}$
- central map $\mathcal{F} : \mathbb{E} \to \mathbb{E}$,

$$\mathcal{F}(X) = \sum_{0 \le i \le j}^{q^i + q^j \le D} \alpha_{ij} X^{q^i + q^j} + \sum_{i=0}^{q^i \le D} \beta_i(v_1, \ldots, v_v) \cdot X^{q^i} + \gamma(v_1, \ldots, v_v)$$

  where β_i is a linear map from $\mathbb{F}^v$ to $\mathbb{E}$ and γ is quadratic
- public key: $\mathcal{P} = \mathcal{S} \circ \phi \circ \mathcal{F} \circ \phi^{-1} \circ \mathcal{T}$ with two affine (or linear) maps $\mathcal{S} : \mathbb{F}^n \to \mathbb{F}^{n-a}$ and $\mathcal{T} : \mathbb{F}^{n+v} \to \mathbb{F}^{n+v}$ of maximal rank
- private key: $\mathcal{S}$, $\mathcal{F}$, $\mathcal{T}$, ϕ

Signature Generation

Given: message $\mathbf{h} \in \mathbb{F}^{n-a}$

1. Compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{h}) \in \mathbb{F}^n$ and $X = \phi(\mathbf{x}) \in \mathbb{E}$
2. Choose random values for the vinegar variables $v_1, \ldots, v_v$
   Solve $\mathcal{F}_{v_1,\ldots,v_v}(Y) = X$ over $\mathbb{E}$ via Berlekamp's algorithm
3. Compute $\mathbf{y} = \phi^{-1}(Y) \in \mathbb{F}^n$ and $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y}||v_1||\ldots||v_v)$

The signature of the message $\mathbf{h}$ is $\mathbf{z} \in \mathbb{F}^{n+v}$.

QUARTZ

- standardized by Courtois, Patarin in 2002
- HFEv$^-$ with $\mathbb{F} = \mathrm{GF}(2)$, $n = 103$, $D = 129$, $a = 3$ and $v = 4$
  $\Rightarrow \mathbb{E} = \mathrm{GF}(2)^{103} = \mathrm{GF}(2)[x]/(x^{103} + x^9 + 1]$

$$\mathcal{F}(X) = \sum_{0 \le i \le j}^{2^i + 2^j \le 129} \alpha_{ij} X^{2^i + 2^j} + \sum_{i=0}^{2^i \le 129} \beta_i(v_1, \ldots, v_4) \cdot X^{2^i} + \gamma(v_1, \ldots, v_4)$$

- public key: quadratic map $\mathcal{P} : \mathbb{F}^{107} \to \mathbb{F}^{100}$
- To avoid birthday attacks, the signature generation step is performed four times (for $\mathbf{h}$, $\mathcal{H}(\mathbf{h}|00)$, $\mathcal{H}(\mathbf{h}|01)$ and $\mathcal{H}(\mathbf{h}|11)$)
  $\Rightarrow$ signature length: $(n - a) + 4 \cdot (a + v) = 128$ bit

Main attacks

- MinRank Attack
  $\mathrm{Rank}(Q) = r + a + v$
  $\Rightarrow \mathrm{Compl}_{\mathrm{MinRank}} \approx 2^{n\cdot(r+a+v)} \cdot (n-a)^3$

- Direct attack
  Recent breakthrough (result by Ding and Yang)
  $$\mathrm{d_{reg}} \leq \begin{cases} \frac{(q-1)\cdot(r-1+a+v)}{2} + 2 & q \text{ even and } r+a \text{ odd}, \\ \frac{(q-1)\cdot(r+a+v)}{2} + 2 & \text{otherwise.} \end{cases},$$
  with $r = \lfloor \log_q(D-1) \rfloor + 1$.

Efficiency

- Signature generation time $\approx$ 10 seconds
- Bottleneck: Inversion of the univariate polynomial equation

$$\mathcal{F}_{(v_1,\ldots,v_v)}(Y) = X \tag{1}$$

  of degree D over the extension field $\mathbb{E}$ by Berlekamps algorithm: Complexity $\mathcal{O}(D^3 + n \cdot D^2)$
- equation (1) solvable with probability $\approx \frac{1}{e}$
- we have to solve (1) for 4 different values of $X \Rightarrow$ we have to perform Berlekamp's algorithm about 11 times

Research Questions

- Is the upper bound on the degree of regularity given by Ding and Yang reasonably tight?
- Can we decrease the degree D of the central $HFEv-$ polynomial to speed up the scheme?

How should we choose D?

- $D \in \{2, 3\}$ would lead to central maps of rank 2 (Matsumoto-Imai case)
- For $D \in \{5, 7\}$ one can get central maps of rank 2 by linear transformation

$\Rightarrow D \in \{9, 17\}$ (central maps of rank 4 and 6 respectively)

Experiments

- Experiments with $HFEv-$ schemes with low degree central maps ($D \in \{9, 17\}$)
- Implementation of $HFEv-$ in MAGMA code
- Fixing of $a + v$ variables to create determined systems
- Adding field equations
- Systems were solved with F_4 integrated in MAGMA

Experiments (2)

$D = 9$

number of equations		20	25	30	32
$a = v = 4$	theoretical degree of regularity ≤ 7				
	(n,D,a,v)	(24,9,4,4)	(29,9,4,4)	(34,9,4,4)	(36,9,4,4)
	d_{reg}	5	6	6	6
	time (s)	2.7	244	31,537	102,321
$a = v = 5$	theoretical degree of regularity ≤ 8				
	(n,D,a,v)	(25,9,5,5)	(30,9,5,5)	(35,9,5,5)	(37,9,5,5)
	d_{reg}	5	6	6	7
	time (s)	2.8	255	32,481	ooM
for comparison: random system					
	d_{reg}	5	6	6	7
	time (s)	3.5	310	32,533	ooM

Experiments (3)

$D = 17$

number of equations		20	25	30	32
$a = v = 3$	theoretical degree of regularity ≤ 7				
	(n,D,a,v)	(23,17,3,3)	(28,17,3,3)	(33,17,3,3)	(35,17,3,3)
	d_{reg}	5	6	6	6
	time (s)	2.4	245	28,768	87,726
$a = v = 4$	theoretical degree of regularity ≤ 8				
	(n,D,a,v)	(24,17,4,4)	(29,17,4,4)	(34,17,4,4)	(36,17,4,4)
	d_{reg}	5	6	6	7
	time (s)	2.4	248	31,911	ooM
for comparison: random system					
	d_{reg}	5	6	6	7
	time (s)	3.5	310	32,533	ooM

Results

- The theoretical result about the degree of regularity is relatively tight
  (for $a = v = 3$ we can reach the upper bound both for $D = 9$ and $D = 17$)
- For the parameter sets $(D, a, v) = (9, 5, 5)$ and $(D, a, v) = (17, 4, 4)$ and $n \geq 32$ we have $d_{\text{reg}} \geq 7$
  $\Rightarrow$ For $n = 90 + a$ we get

$$\begin{aligned} \text{Complexity}_{\text{direct attack}} &\geq 3 \cdot \binom{n-a+2}{2} \cdot \binom{n-a+d_{\text{reg}}}{d_{\text{reg}}}^2 \\ &= 3 \cdot \binom{92}{2} \cdot \binom{97}{7}^2 \geq 2^{81} \end{aligned}$$

New Designs - Gui - Asiacrypt 2015 - Petzoldt, Chen, Yang, Tao, Ding

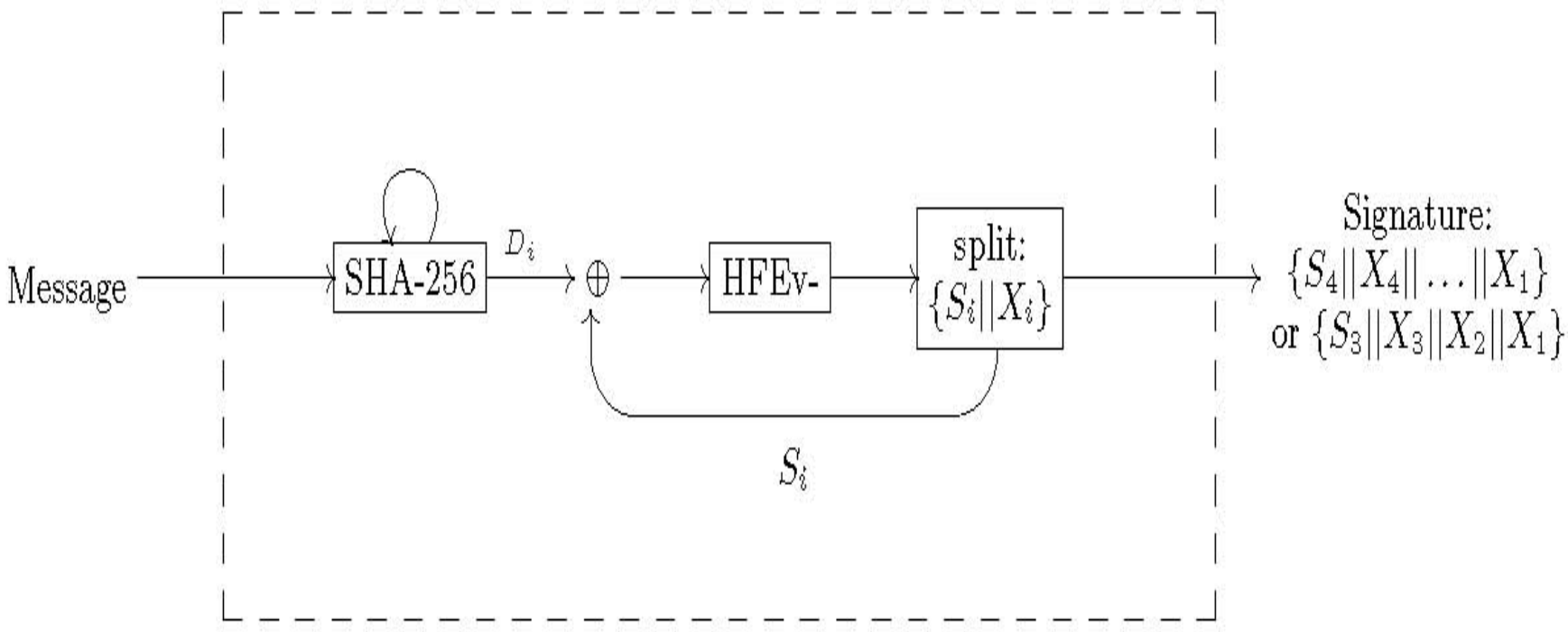
We propose three versions of Gui

- Gui-95 with $(n, D, a, v) = (95, 9, 5, 5)$ providing a security level of 80 bit
- Gui-94 with $(n, D, a, v) = (94, 17, 4, 4)$ providing a security level of 80 bit
  and
- Gui-127 with $(n, D, a, v) = (127, 9, 4, 6)$ providing a security level of 123 bit

Avoiding birthday attacks

- Input size of HFEv- maps is short (in our case 90 - 123 bit)
  $\Rightarrow$ Possibility of birthday attacks
- Solution:
  - Sign k different hash values of the message m.
  - Combine the k outputs to a single signature of size $(n - a) + k \cdot (a + v)$ bit.
- In the case of Gui we set
  - $k = 3$ for Gui-95,
  - $k = 4$ for Gui-94 and Gui-127.

Gui-95


Message
SHA-256
D_i
$\oplus$
HFEv-
split:
$\{S_i||X_i\}$
S_i
Signature:
$\{S_4||X_4||\ldots||X_1\}$
or $\{S_3||X_3||X_2||X_1\}$

Parameters and Key Sizes

scheme	security level (bit)	input size (bit)	signature size (bit)	public key size (Bytes)	private key size (Bytes)
Gui-95	80	90	120	60,600	3,053
Gui-94	80	90	122	58,212	2,943
Gui-127	123	123	163	142,576	5,350
QUARTZ	80	100	128	75,514	3,774
RSA-1024	80	1024	1024	128	128
RSA-2048	112	2048	2048	256	256
ECDSA P160	80	160	320	40	60
ECDSA P192	96	192	384	48	72
ECDSA P256	128	256	512	64	96

Comparison

scheme	security level (bit)	signing time (k-cycles)	verifying time (k-cycles)
Gui-95	80	1,479 / 1,186	325 / 230
Gui-94	80	4,945 / 5,421	357 / 253
Gui-127	123	1,966 / 1,249	707 / 427
QUARTZ	80	167,485 / 168,266	375 / 235
RSA-1024	80	2,080 / 2,115	74 / 64
RSA-2048	112	8,834 / 5,347	138 / 76
ECDSA P160	80	1,283 / 1,115	1,448 / 1,269
ECDSA P192	96	1,513 / 1,273	1, 715 / 1,567
ECDSA P256	128	1,830 / 1,488	2,111 / 1,920

time on AMD Opteron 6212, 2.5 GHz / Intel Xeon E5-2620, 2.0 GHz

Why this name?

Gui

- Chinese pottery from Longshan period
- more than 4000 years old
- 3 legs: one in front, 2 in the back

- front leg : HFE
- back legs: Minus + Vinegar

Key Points

- Proposal of a new multivariate signature scheme Gui
- Use of low degree HFEv- polynomials ($D \in \{9, 17\}$)

$\Rightarrow$ very short signatures (120 bit)
$\Rightarrow$ 150 times faster than QUARTZ
$\Rightarrow$ Efficiency comparable to standard schemes (RSA, ECDSA)

Outline

Key Points

- The Main Defect for insecurity for most of these MPKQ is that some Quadratic Forms associated with their central maps are of Low Rank.
- Direct algebraic attack is easy to handle in general (odd Char) Ding, Tao, Diene etc

Idea of the Simple Matrix Schem for Encrypyion

Main Idea

Create some Matrices having high rank and use some Simple Matrix Multiplication to get a Multivariate Publick Key Scheme that we denote in short by the ABC cryptosystem.

Construction of the SM Cryptosystem

- Let $k = F_q$ be a finite field with q elements and p be the characteristic of k.
- Let n, m be a integer, where $n = s^2, m = 2n$.
- The plaintext will be represented by $(x_1, x_2, \cdots, x_n) \in k^n$.
- The ciphertext will be represented by $(y_1, y_2, \cdots, y_m) \in k^m$.

Construction of the SM Cryptosystem

- Let $\mathcal{L}_1 : k^n \to k^n$ and $\mathcal{L}_2 : k^m \to k^m$ be 2 affine transformations,
  $\mathcal{L}_1(x) = L_1 x + u$ and $\mathcal{L}_2(y) = L_2 y + \nu$
  where L_1 and L_2 are respectively an $n \times n$ and $m \times m$ matrix with entries in k, $x = (x_1, x_2, \cdots, x_n)^t$,
  $u = (u_1, u_2, \cdots, u_n)^t$, $y = (y_1, y_2, \cdots, y_m)^t$,
  $\nu = (v_1, v_2, \cdots, v_m)^t$

Construction of the SM Cryptosystem

- Let

$$A = \begin{pmatrix} x_1 & x_2 & \cdots & x_s \\ x_{s+1} & x_{s+2} & \cdots & x_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(s-1)s+1} & x_{(s-1)s+2} & \cdots & x_{s^2} \end{pmatrix},$$

$$B = \begin{pmatrix} b_1 & b_2 & \cdots & b_s \\ b_{s+1} & b_{s+2} & \cdots & b_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ b_{(s-1)s+1} & b_{(s-1)s+2} & \cdots & b_{s^2} \end{pmatrix} \quad \text{and}$$

$$C = \begin{pmatrix} c_1 & c_2 & \cdots & c_s \\ c_{s+1} & c_{s+2} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{(s-1)s+1} & c_{(s-1)s+2} & \cdots & c_{s^2} \end{pmatrix}$$

Construction of the SM Cryptosystem

- **Central map**

  A, B, and C defined above are 3 $s \times s$ matrices with $x_i \in k$ $(i = 1, 2, \cdots, n)$, b_i and c_i $(i = 1, 2, \cdots, n)$ are random linear combinaisons of elements taken from the set $\{x_1, x_2, \cdots, x_n\}$.
  Let $E_1 = AB$, $E_2 = AC$,
  we denote by $f_{(i-1)s+j} \in k[x_1, x_2, \cdots, x_n]$
  the (i, j) element of E_1 $(i, j = 1, 2, \cdots, s)$.
  $f_{s^2+(i-1)s+j} \in k[x_1, x_2, \cdots, x_n]$
  the (i, j) element of E_2 $(i, j = 1, 2, \cdots, s)$.
  We define then

$$\mathcal{F}(x_1, \cdots, x_n) = (f_1(x_1, \cdots, x_n), f_2(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n)).$$

Construction of the SM Cryptosystem

- The public key:

$$\bar{\mathcal{F}} = \mathcal{L}_2 \circ \mathcal{F} \circ \mathcal{L}_1 = (\bar{f}_1, \bar{f}_2, \cdots, \bar{f}_m),$$

- The secret key is made of the following two parts:

  The invertible affine transformations $\mathcal{L}_1, \mathcal{L}_2$.

  The matrices B, C.

Construction of the SM Cryptosystem

- **Decryption**
  Applying $\bar{\mathcal{F}}^{-1} = \mathcal{L}_1^{-1} \circ \mathcal{F}^{-1} \circ \mathcal{L}_2^{-1}$.
- How to invert the central map:
  Since $E_1 = AB, \quad E_2 = AC$ and assume A is an $s \times s$ nonsingular matrix, we consider the following cases:
  (i) If E_1 is invertible, then $BE_1^{-1}E_2 = C$. We have n linear equations with n unknowns $x_i, i = 1, 2, \cdots, n$.
  (ii) If E_2 is invertible, but E_1 is not invertible, then $CE_2^{-1}E_1 = B$. We also have n linear equations with n unknowns $x_i, i = 1, 2, \cdots, n$.
  (iii) If both E_1 and E_2 are not invertible, then $A^{-1}E_1 = B, A^{-1}E_2 = C$. We interpret the elements of A^{-1} as the new variables, then we have m linear equations with m unknowns.

Construction of the SM Cryptosystem

- Decryption failure
  If A is a singular matrix, we may decrypt failure. The probability of A is invertible is $(1-\frac{1}{q})(1-\frac{1}{q^2})\cdots(1-\frac{1}{q^n})$. Therefore, the probability of decryption failure is $1-(1-\frac{1}{q})(1-\frac{1}{q^2})\cdots(1-\frac{1}{q^n}) \approx \frac{1}{q}$.

Construction of the SM Cryptosystem

- An example
  We let $k = GF(q)$ be a finite field of $q = 127$ elements and $n = 64$. In this case, the plaintext consist of the message $(x_1, x_2, \cdots, x_{64}) \in k^{64}$. The public map is $\bar{F} : k^{64} \to k^{128}$ and the central map is $F : k^{64} \to k^{128}$.
- The public key consists of 128 quadratic polynomials with 64 variables. The number of coefficients for the public key polynomials is $128 \times 64 \times 65/2 = 266,240$, or about $2MB$ of storage.
- The private key consists of two matrices B, C and two affine linear transformations $\mathcal{L}_1, \mathcal{L}_2$. The total size is about $162.5KB$.
- The size of document is $8n = 8 \times 64 = 512bits$. The total size of the ciphertext is $1024bits$.

Security Analysis

- Rank attack:
  For the rank attacks, we have that the MinRank is 16 and the complexity of MinRank attack against our scheme is lager than 2^{160}.
- Algebraic attack:
  For $k = GF(3)$, we obtain the following results with a direct attack using MAGAMA(2.12-16) on a 1.80GHz Intel(R) Atom(TM) CPU

n	9	16	25
time(s)	0.016	3.494	17588.380
memory(MB)	3.4	8.1	1111.7
degree of regularity	4	5	6

We can notice that the degree of regularity increases with n which tells us that the time and memory complexity are exponential.

Construction of the SM Cryptosystem

- Efficiency
  The decrytion is very efficient: only linera algebra opeations.

Improved Conctructions

- Rectangular construction
- Degree three construction using random quadratic polynomials
- Remove the decryptin fauilure
  Ding, Petzolt, Wang

Outline

Key Attack Methods

To be ready for practical applications, we need a solid understanding of the attack complexities with both theoretical and experimental support.
The key attack methods are:

- Direct algebraic attack

Key Attack Methods

To be ready for practical applications, we need a solid understanding of the attack complexities with both theoretical and experimental support.
The key attack methods are:

- Direct algebraic attack
- MinRank attack – which is also reduced to polynomial solving problem.

Key Attack Methods

To be ready for practical applications, we need a solid understanding of the attack complexities with both theoretical and experimental support.
The key attack methods are:

- Direct algebraic attack
- MinRank attack – which is also reduced to polynomial solving problem.
- Differential analysis

Direct Algebraic Attack

Use efficient Gröbner basis (algebraic) algorithms (GB, F_4, XL, Mutant XL) to solve the system of equations:

$$\begin{aligned} p_1(x_1, \ldots, x_n) &= y_1 \\ p_2(x_1, \ldots, x_n) &= y_2 \\ &\vdots \\ p_n(x_1, \ldots, x_n) &= y_n \end{aligned}$$

Direct Algebraic Attack

Use efficient Gröbner basis (algebraic) algorithms (GB, F_4, XL, Mutant XL) to solve the system of equations:

$$\begin{aligned} p_1(x_1, \ldots, x_n) &= y_1 \\ p_2(x_1, \ldots, x_n) &= y_2 \\ &\vdots \\ p_n(x_1, \ldots, x_n) &= y_n \end{aligned}$$

Sometime algorithm terminates significantly quicker for the MPKC systems than on random systems.
Why?

Degree of Regularity

Degree of Regularity: Lowest degree at which non-trivial "degree falls" occur.

$$\deg\left(\sum_i g_i p_i\right) < \max\{\deg(g_i) + \deg(p_i)\}$$

Trivial degree falls:

$$p_i^{q-1} p_i = p_i^q = p_i, \quad p_j p_i - p_i p_j = 0$$

Implication of Degree of Regularity

- **Gröbner basis algorithms terminate shortly after this degree is reached.**

Implication of Degree of Regularity

- **Gröbner basis algorithms terminate shortly after this degree is reached.**
- *At the degree of regularity, in general,* **Mutants** *are produced, which accelerate the solving process.*
  We need more precise mathematical concepts like, degree of regularity, mutants etc to understand solidly how algorithm works.

Degree of Regularity of Leading Terms

Let p_i^h be the highest degree part of p_i considered as an element of the truncated polynomial ring

$$p_i^h \in \frac{\mathbb{F}[x_1, \ldots, x_n]}{\langle x_1^q, \ldots, x_n^q \rangle}$$

Degree of Regularity of Leading Terms

Let p_i^h be the highest degree part of p_i considered as an element of the truncated polynomial ring

$$p_i^h \in \frac{\mathbb{F}[x_1, \ldots, x_n]}{\langle x_1^q, \ldots, x_n^q \rangle}$$

Degree of Regularity of $p_1^h, \ldots, p_n^h$ is first degree at which non-trivial relations occur.

$$\deg\left(\sum_i f_i p_i^h\right) = 0$$

Trivial relations: $(p_i^h)^{q-1} p_i^h = 0, \quad p_j^h p_i^h - p_i^h p_j^h = 0$

Degree of Regularity of Leading Terms

Let p_i^h be the highest degree part of p_i considered as an element of the truncated polynomial ring

$$p_i^h \in \frac{\mathbb{F}[x_1, \ldots, x_n]}{\langle x_1^q, \ldots, x_n^q \rangle}$$

Degree of Regularity of $p_1^h, \ldots, p_n^h$ is first degree at which non-trivial relations occur.

$$\deg\left(\sum_i f_i p_i^h\right) = 0$$

Trivial relations: $(p_i^h)^{q-1} p_i^h = 0, \quad p_j^h p_i^h - p_i^h p_j^h = 0$
Then

$$D_{\mathrm{reg}}(p_1, \ldots, p_n) = D_{\mathrm{reg}}(p_1^h, \ldots, p_n^h)$$

Bounds on Degree of Regularity

- Recently, we found a global upper bound on the degree of regularity (in the sense of DG) of an HFE system.

Bounds on Degree of Regularity

- Recently, we found a global upper bound on the degree of regularity (in the sense of DG) of an HFE system.
- **Main Theorem.**
  The degree of regularity of the system defined by P is bounded by

$$\frac{\mathrm{Rank}(P_0)(q-1)}{2} + 2 \leq \frac{(q-1)(\lfloor \log_q(D-1) \rfloor + 1)}{2} + 2$$

  if $\mathrm{Rank}(P_0) > 1$. Here $\mathrm{Rank}(P_0)$ is the rank of the quadratic form P_0.
  This explains why odd characteristics is good idea and why $q = 2$ is different

Bounds on Degree of Regularity

- Recently, we found a global upper bound on the degree of regularity (in the sense of DG) of an HFE system.
- **Main Theorem.**
  The degree of regularity of the system defined by P is bounded by

$$\frac{\mathrm{Rank}(P_0)(q-1)}{2} + 2 \leq \frac{(q-1)(\lfloor \log_q(D-1) \rfloor + 1)}{2} + 2$$

  if $\mathrm{Rank}(P_0) > 1$. Here $\mathrm{Rank}(P_0)$ is the rank of the quadratic form P_0.
  This explains why odd characteristics is good idea and why $q = 2$ is different
- These are universal bounds that require no additional assumption.

Bounds on Degree of Regularity for other systems

- HFE- (Ding, Kleijung)

Bounds on Degree of Regularity for other systems

- HFE- (Ding, Kleijung)
- HFEv- (Ding, Yang)

Bounds on Degree of Regularity for other systems

- HFE- (Ding, Kleijung)
- HFEv- (Ding, Yang)
- Precise bound for Square systems. (Ding)

Bounds on Degree of Regularity for other systems

- HFE- (Ding, Kleijung)
- HFEv- (Ding, Yang)
- Precise bound for Square systems. (Ding)
- Lower bounds for general case?

Bounds on Degree of Regularity for other systems

- HFE- (Ding, Kleijung)
- HFEv- (Ding, Yang)
- Precise bound for Square systems. (Ding)
- Lower bounds for general case?
- MinRank is also closely related.

MPKCs

- MPKCs has a very solid foundation in terms of both designs and security analysis.

MPKCs

- MPKCs has a very solid foundation in terms of both designs and security analysis.
- Efficient, simple and easy to implement; but large key size

MPKCs

- MPKCs has a very solid foundation in terms of both designs and security analysis.
- Efficient, simple and easy to implement; but large key size
- Quantum computer attack

Acknowledgment

Many thanks for the organizer

Thank you and questions?