Post-quantum Cryptography
Multivariate Public Key Cryptography

Jintai Ding

Academis Sinica
University of Cincinnati

April 2, 2014

Outline

Outline

PQC

Cryptosystems that have potential to resist the future quantum computer attacks.

- Code-based cryptography

PQC

Cryptosystems that have potential to resist the future quantum computer attacks.

- Code-based cryptography
- Hash-based crytograohy

PQC

Cryptosystems that have potential to resist the future quantum computer attacks.

- Code-based cryptography
- Hash-based crytograohy
- Lattice cryptography

PQC

Cryptosystems that have potential to resist the future quantum computer attacks.

- Code-based cryptography
- Hash-based crytograohy
- Lattice cryptography
- Multivariate cryptography

What is a MPKC?

- Multivariate Public Key Cryptosystems
  - *Cryptosystems with public keys as a set of multivariate functions*

What is a MPKC?

- Multivariate Public Key Cryptosystems
  - *Cryptosystems with public keys as a set of multivariate functions*
- **Public key**: G is a map from k^n to k^m:

$$G(x_1, \ldots, x_n) = (g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n));$$

$$G = L_2 \circ F \circ L_1,$$

over k, a small finite field like GF(2^8)
F: central map and F^{-1} easy to compute.
L_1 and L_2: "locks" on the secret of F.

What is a MPKC?

- Multivariate Public Key Cryptosystems
  - *Cryptosystems with public keys as a set of multivariate functions*
- **Public key**: G is a map from k^n to k^m:

$$G(x_1, \ldots, x_n) = (g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n));$$

$$G = L_2 \circ F \circ L_1,$$

  over k, a small finite field like GF(2^8)
  F: central map and F^{-1} easy to compute.
  L_1 and L_2: "locks" on the secret of F.
- **Private key**: a way to compute G^{-1} via the map decomposition or **factoring**.

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

a MPKC signature system

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

a MPKC signature system

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

$$G^{-1}(y_1, \ldots, y_m) = L_2^{-1} \circ F^{-1} \circ L_1^{-1}(y_1, \ldots, y_m).$$

- **Verifying**: $(y_1, \ldots, y_m) \stackrel{?}{=} G(x_1, \ldots, x_n)$.

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-hard, though this does not necessarily ensure the security of the systems.*

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.*

$$x_1 x_2 x_3 = 1,$$

is equivalent to

$$x_4 = x_1 x_2$$
$$x_4 x_3 = 1.$$

The view from the history of Mathematics

- RSA – Number Theory – the 18th century mathematics

The view from the history of Mathematics

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

The view from the history of Mathematics

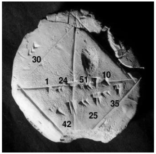
- RSA – Number Theory – the 18th century mathematics
- ECC – Theory of Elliptic Curves – the 19th century mathematics
- Multivariate Public key cryptosystem – Algebraic Geometry – the 20th century mathematics

Algebraic Geometry – Theory of Polynomial Rings

Humans have been trying to solve polynomial equations for thousands of years.

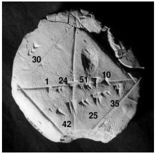
A quick historic overview

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



A quick historic overview

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

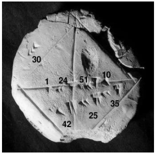


- Cubic and quartic equation – around 1500

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: Normal 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

Historical Development

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

Historical Development

- Early attempts by Diffie, Fell, Imai, Ong, Matsumoto, Schnorr, Shamir, Tsujii, etc
- Fast development in the late 1990s.

Outline

How to construct G?

- The unbalanced Oil-Vinegar scheme by Kipnis, Patarin and Goubin 1999.

How to construct G?

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

How to construct G?

- More efficient construction - Multi-layer UOV – Rainbow by Ding and Schmidt 2005.

How to construct G?

- More efficient construction - Multi-layer UOV – Rainbow by Ding and Schmidt 2005.
- $G = L_2 \circ F \circ L_1$.
  F: Multilayer UOV, easy to compute F^{-1}.
  L_1, L_2: 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

- $F = (f_1(x_1, .., x_o, x'_1, ..., x'_v), \cdots, f_o(x_1, .., x_o, x'_1, ..., x'_v))$.
- Each f_i is an Oil-Vinegar polynomial:

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

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

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

How to invert F?

- Randomly assign values to Vinegar variables:

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

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

How to invert F?

- Randomly assign values to Vinegar variables:

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

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

$$f_l(x_1, .., x_o, x'_1, .., x'_v) =$$

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

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

$\Longrightarrow F$: easy to invert.

The F for Rainbow

- Layer 1:
  Vinegar: $x_1, ., x_{v_1}$
  Oil: $x_{v_1+1}, ., x_{v_1+o_1}$

$$(f_1, ..., f_{o_1})$$

The F for Rainbow

- Layer 1:
  Vinegar: $x_1, ., x_{v_1}$
  Oil: $x_{v_1+1}, ., x_{v_1+o_1}$

$$(f_1, ..., f_{o_1})$$

- Layer 2:
  Vinegar: $x_1, ., x_{v_1}, x_{v_1+1}, ., x_{v_1+o_1}$ Oil: $x_{v_1+o_1+1}, ., x_{v_1+o_1+o_2}$

$$(f_{o_1+1}, ..., f_{o_1+o_2})$$

The F for Rainbow

- Layer 1:
  Vinegar: $x_1, ., x_{v_1}$
  Oil: $x_{v_1+1}, ., x_{v_1+o_1}$

$$(f_1, ..., f_{o_1})$$

- Layer 2:
  Vinegar: $x_1, ., x_{v_1}, x_{v_1+1}, ., x_{v_1+o_1}$ Oil: $x_{v_1+o_1+1}, ., x_{v_1+o_1+o_2}$

$$(f_{o_1+1}, ..., f_{o_1+o_2})$$

- $$F = (f_1, .., f_{o_1}, f_{o_1+1}, ..., f_{o_1+o_2}).$$

The F^{-1} for Rainbow

- Layer 1:
  Assign values to Vinegar: $x_1, ., x_{v_1}$ in

$$(f_1, ..., f_{o_1}) = (y_1, .., y_{o_1}),$$

  solve and find the value of Oil: $x_{v_1+1}, ., x_{v_1+o_1}$

The F^{-1} for Rainbow

- Layer 1:
  Assign values to Vinegar: $x_1, ., x_{v_1}$ in

  $$(f_1, ..., f_{o_1}) = (y_1, .., y_{o_1}),$$

  solve and find the value of Oil: $x_{v_1+1}, ., x_{v_1+o_1}$
- Layer 2:
  Plug in values of
  Vinegar: $x_1, ., x_{v_1}, x_{v_1+1}, ., x_{v_1+o_1}$
  in

  $$(f_{o_1+1}, ..., f_{o_1+o_2}) = (y_{o_1+1}, .., y_{o_1+o_2})$$

  find the values of Oil: $x_{v_1+o_1+1}, ., x_{v_1+o_1+o_2}$

The F^{-1} for Rainbow

- Layer 1:
  Assign values to Vinegar: $x_1, ., x_{v_1}$ in
  $$(f_1, ..., f_{o_1}) = (y_1, .., y_{o_1}),$$
  solve and find the value of Oil: $x_{v_1+1}, ., x_{v_1+o_1}$
- Layer 2:
  Plug in values of
  Vinegar: $x_1, ., x_{v_1}, x_{v_1+1}, ., x_{v_1+o_1}$
  in
  $$(f_{o_1+1}, ..., f_{o_1+o_2}) = (y_{o_1+1}, .., y_{o_1+o_2})$$
  find the values of Oil: $x_{v_1+o_1+1}, ., x_{v_1+o_1+o_2}$
- This givs us $F^{-1}(y_i, .., y_{o_1+o_2}$:
  $(x_1, .., x_{v_1}, ..., x_{o_1+v_1}, ..., x_{o_1+o_2+v_1})$.

Security analysis

1. Systematic theoretical and experimental analysis
   - Direct attack does not work against best existing polynomial solving algorithms
     The cpomplexity bahves just like a random system.
   - Finding keys again becomes a problem of solving polynomial equations
     Here we need to be careful with choice of parameters.
   - MinRank attack on Rainbow:
     Given a set of matrix $M_1, ..M_n$ find a non-trivial $\sum a_i M_i$ with lowest rank.
     MinRank is a hard problem and attack it is reduced to solve multivariate polynomial equations again.
   - Natural Side channel attack resistance.

Security analysis

1. Systematic theoretical and experimental analysis
   - Direct attack does not work against best existing polynomial solving algorithms
     The cpomplexity bahves just like a random system.
   - Finding keys again becomes a problem of solving polynomial equations
     Here we need to be careful with choice of parameters.
   - MinRank attack on Rainbow:
     Given a set of matrix $M_1, ..M_n$ find a non-trivial $\sum a_i M_i$ with lowest rank.
     MinRank is a hard problem and attack it is reduced to solve multivariate polynomial equations again.
   - Natural Side channel attack resistance.
2. No weakness yet being found in the design.

Parameters and Performance

- Rainbow(17,13,13) over GF(2^8): Signature size: 43 bytes, private key: 19.1KB, public key 25.1KB.
- Rainbow(26,16,17) over GF(2^8): Signature size: 59 bytes , private key 45.0KB, public key 59.0KB.
- Rainbow(36,21,22) over GF(2^8): Signature size: 79 bytes, private key 101.5KB, public key 136.1KB.

Parameters and Performance

- High efficiency – solving linear equations.
  IC for Rainbow: 804 cycles. (ASAP 2008)
  FPGA implementation at Bochum (CHES 2009) – Beat ECC in area and speed.
  Faster parallel implementation 200 cycles – (PQC 2011)

Parameters and Performance

- High efficiency – solving linear equations.
  IC for Rainbow: 804 cycles. (ASAP 2008)
  FPGA implementation at Bochum (CHES 2009) – Beat ECC in area and speed.
  Faster parallel implementation 200 cycles – (PQC 2011)
- Relative large public key
  Further optimizations – Petzoldt, Buchmann etc. at TU Darmstadt

Parameters and Performance

- High efficiency – solving linear equations.
  IC for Rainbow: 804 cycles. (ASAP 2008)
  FPGA implementation at Bochum (CHES 2009) – Beat ECC in area and speed.
  Faster parallel implementation 200 cycles – (PQC 2011)
- Relative large public key
  Further optimizations – Petzoldt, Buchmann etc. at TU Darmstadt
- Highly efficient compact signature
  Small devices – RFID, Sensors.

Another choice – HFEV-Minus – Quartz

- The basic design: Hidden field equation system (HFE) with Vinegar variables and Minus modification designed in 1999

  HFE: k^n can be identified as a lrage field $\bar{K} = k[x]/g(x)$, where $g(x)$ an ireeducible polynomial.

  We use a olynomail

$$F(X) = \sum a_{ij} X^{q^i + q^j} + \sum b_i X^{q^i} + C..$$

Another choice – HFEV-Minus – Quartz

- The basic design: Hidden field equation system (HFE) with Vinegar variables and Minus modification designed in 1999

  HFE: k^n can be identified as a lrage field $\bar{K} = k[x]/g(x)$, where $g(x)$ an ireeducible polynomial.

  We use a olynomail

  $$F(X) = \sum a_{ij} X^{q^i + q^j} + \sum b_i X^{q^i} + C..$$

- Very short signature (107 bits) but slow.

Another choice – HFEV-Minus – Quartz

- The basic design: Hidden field equation system (HFE) with Vinegar variables and Minus modification designed in 1999

  HFE: k^n can be identified as a lrage field $\bar{K} = k[x]/g(x)$, where $g(x)$ an ireeducible polynomial.

  We use a olynomail

  $$F(X) = \sum a_{ij} X^{q^i + q^j} + \sum b_i X^{q^i} + C..$$

- Very short signature (107 bits) but slow.
- No weakness yet found.

Another choice – HFEV-Minus – Quartz

- The basic design: Hidden field equation system (HFE) with Vinegar variables and Minus modification designed in 1999

  HFE: k^n can be identified as a lrage field $\bar{K} = k[x]/g(x)$, where $g(x)$ an ireeducible polynomial.

  We use a olynomail

$$F(X) = \sum a_{ij} X^{q^i + q^j} + \sum b_i X^{q^i} + C..$$

- Very short signature (107 bits) but slow.
- No weakness yet found.
- New designs by Ding, Petzoldt, Tao, Yang. very efficient (more than 1000 times faster with a 92 bits signature, or 170bits for post-quantum signature.)

Another choice – HFEV-Minus – Quartz

- The basic design: Hidden field equation system (HFE) with Vinegar variables and Minus modification designed in 1999

  HFE: k^n can be identified as a lrage field $\bar{K} = k[x]/g(x)$, where $g(x)$ an ireeducible polynomial.

  We use a olynomail

$$F(X) = \sum a_{ij} X^{q^i + q^j} + \sum b_i X^{q^i} + C..$$

- Very short signature (107 bits) but slow.
- No weakness yet found.
- New designs by Ding, Petzoldt, Tao, Yang. very efficient (more than 1000 times faster with a 92 bits signature, or 170bits for post-quantum signature.)
- Solid theoretical and experimental security analysis. Degree of regularity, solving degree, degeneration degree

Outline

The basic design

- The public key is given as:

$$G(x_1, ..., x_n) = (G_1(x_1, ..., x_n), ..., G_m(x_1, ..., x_n)) = L_2 \circ F \circ L_1.$$

G_i are multivariate polynomials over a finite field, which are mostly degree 2

The basic design

- The public key is given as:

  $$G(x_1, ..., x_n) = (G_1(x_1, ..., x_n), ..., G_m(x_1, ..., x_n)) = L_2 \circ F \circ L_1.$$

  G_i are multivariate polynomials over a finite field, which are mostly degree 2
- Any plaintext $M = (x'_1, ..., x'_n)$ is encrypted via polynomial evaluation:
  $G(M) = G(x'_1, ..., x'_n) = (y'_1, ..., y'_m)$.

The basic design

- The public key is given as:

  $$G(x_1, ..., x_n) = (G_1(x_1, ..., x_n), ..., G_m(x_1, ..., x_n)) = L_2 \circ F \circ L_1.$$

  G_i are multivariate polynomials over a finite field, which are mostly degree 2
- Any plaintext $M = (x'_1, ..., x'_n)$ is encrypted via polynomial evaluation:
  $G(M) = G(x'_1, ..., x'_n) = (y'_1, ..., y'_m)$.
- To decrypt the ciphertext $(y'_1, ..., y'_n)$, one needs to know a secret (**the secret key**) to compute the inverse map G^{-1} to find the plaintext $(x'_1, ..., x'_n) = G^{-1}(y'_1, .., y'_n)$.

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 $\{\mathit{0}, \mathit{1}, \mathit{2}, \mathit{3}\}$ to simplify the notation.
  Here $\mathit{0}$ represents the 0 in k, $\mathit{1}$ for 1, $\mathit{2}$ for x, and $\mathit{3}$ for $1 + x$.
  In this case, $\mathit{1} + \mathit{3} = \mathit{2}$ and $\mathit{2} * \mathit{3} = \mathit{1}$.

A toy example

-

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

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.

The best designs

- Internal perturbation of HFE and perturbed MI with Plus. Designed by Ding, Schmidt.

The best designs

- Internal perturbation of HFE and perturbed MI with Plus. Designed by Ding, Schmidt.
- But relatively slow and large key size.

The best designs

- Internal perturbation of HFE and perturbed MI with Plus. Designed by Ding, Schmidt.
- But relatively slow and large key size.
- New designs – Simple matrix method by Ding and Tao 2013.

The best designs

- Internal perturbation of HFE and perturbed MI with Plus. Designed by Ding, Schmidt.
- But relatively slow and large key size.
- New designs – Simple matrix method by Ding and Tao 2013.
- The efficiency is now comparable with with the signature scheme.

Outline

Main attacks

- Albegraic attacks: attack a cryptosystem via a problem solving a set of polynomial equations.
  Degree of regularity, degeneration degree, solving degree.

Main attacks

- Albegraic attacks: attack a cryptosystem via a problem solving a set of polynomial equations.
  Degree of regularity, degeneration degree, solving degree.
- MinRank Problem:
  Given a set of matrix $M_1, ..M_n$, find the nonetrivial minimum rank of $a_1 M_1 + a_2 M_2 + ..., a_n M_n$.
  This is again coverted in to a polynomial solving problem.

Main attacks

- Albegraic attacks: attack a cryptosystem via a problem solving a set of polynomial equations.
  Degree of regularity, degeneration degree, solving degree.
- MinRank Problem:
  Given a set of matrix $M_1, ..M_n$, find the nonetrivial minimum rank of $a_1 M_1 + a_2 M_2 + ..., a_n M_n$.
  This is again coverted in to a polynomial solving problem.
- Hidden symmetry: we can handle these problems easily by eliminating those symmetries with mathematical proofs. (D. Smith, R. Perlner)

Algebraic attacks

- Algebraic attacks: attack a cryptosystem via a problem solving a set of polynomial equations.

Algebraic attacks

- Algebraic attacks: attack a cryptosystem via a problem solving a set of polynomial equations.
- Polynomial solving algorithms: F4, Mutant XL, SAT solvers etc

Algebraic attacks

- Algebraic attacks: attack a cryptosystem via a problem solving a set of polynomial equations.
- Polynomial solving algorithms: F4, Mutant XL, SAT solvers etc
- We have a solid understanding of the complexity of those attacks, where our theoretical analysis matches precisely the experimental analysis.
  Degeneration degree, solving degree (degree of regualrity)

Summary

- MPKC provide the best signature designs in terms of computing performance and signature size.

Summary

- MPKC provide the best signature designs in terms of computing performance and signature size.
- The security analysis has solid theoretical support and systematic experimental support.

Summary

- MPKC provide the best signature designs in terms of computing performance and signature size.
- The security analysis has solid theoretical support and systematic experimental support.
- Drawback: relative large key sizes (10s KB) but can be substantially improved with further optimization

Summary

- MPKC provide the best signature designs in terms of computing performance and signature size.
- The security analysis has solid theoretical support and systematic experimental support.
- Drawback: relative large key sizes (10s KB) but can be substantially improved with further optimization
- We have solid but not so efficient encryption schemes. New designs are catching up.

The end

Thank you