$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Public Key Schemes

Christopher Wolf

Katholieke Universiteit Leuven
Dept. Electrical Engineering — ESAT/COSIC

`Christopher.Wolf@esat.kuleuven.ac.be`
`chris@Christopher-Wolf.de`

`hack.lu` 2005

October 14, 2005, Luxembourg

Outline

Introduction

$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

Basic Classes

Practical Examples

Conclusions

Introduction

Motivation

- Public Key Cryptography is necessary for using the Internet in a secure manner, *e.g.*, eCommerce applications, private communication, secure communication
- At present, mostly RSA (factoring), and ECC (discrete logarithm) are used for public key cryptography
- Both factoring and discrete logarithm are insecure under the assumption that quantum computer exist (algorithm of Shor), hence new schemes are needed

Introduction
$\mathcal{MQ}$-Schemes

- $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic schemes are known since the 1980s — since then, some effort has been made to understand their security
- These schemes allow fast encryption and signature verification — often also fast decryption and signature generation
- With the current constructions, it is possible to achieve signatures as short as 128 bits
- All in all, they are a worthwhile research topic but could be used in practice now

KATHOLIEKE UNIVERSITEIT LEUVEN

Outline

Introduction

$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

Basic Classes

Practical Examples

Conclusions

$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

Graphical Overview

$$\begin{tikzcd}
\text{input } x \arrow[d] \\
x = (x_1, \ldots, x_n) \arrow[d, "\text{private: } S"'] \arrow[dddd, bend left, "\textbf{public}\text{: } (p_1, \ldots, p_m)"] \\
x' \arrow[d, "\text{private: } \mathcal{P}'"'] \\
y' \arrow[d, "\text{private: } T"'] \\
\text{output } y
\end{tikzcd}$$

$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

Overview

- Public key schemes based on problem of solving $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic polynomial equations and the Isomorphism of Polynomials problem
- Use polynomials over small finite fields $\mathbb{F}$, *e.g.*, GF(2), GF(128), or GF(256) — hence very suitable for 8-bit microprocessors
- Some schemes use extension fields $\mathbb{E}$ with dimension n over $\mathbb{F}$, constructed by an irreducible polynomial $i(t)$
- Secret key $(S, \mathcal{P}', T) \in \mathrm{AGL}_n(\mathbb{F}) \times \mathcal{MQ}_m(\mathbb{F}^n) \times \mathrm{AGL}_m(\mathbb{F})$
- Public key $\mathcal{P} \in \mathcal{MQ}_m(\mathbb{F}^n)$ with $\mathcal{P} = T \circ \mathcal{P}' \circ S$

$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

Public Key

Public Key equations:

$$p_i(x_1,\ldots,x_n) := \sum_{1\le j\le k\le n} \gamma_{i,j,k} x_j x_k + \sum_{j=1}^{n} \beta_{i,j} x_j + \alpha_i$$

for $1 \le i \le m$ and coefficients $\alpha_i, \beta_{i,j}, \gamma_{i,j,k} \in \mathbb{F}$

Notation: $\mathcal{P}(x) := (p_1(x_1,\ldots,x_n),\ldots,p_m(x_1,\ldots,x_n))$

Encryption: Compute $y \in \mathbb{F}^m$ for given $x \in \mathbb{F}^n$ by evaluating $y = \mathcal{P}(x)$

Signature verification: Given pair $(x,y) \in \mathbb{F}^n \times \mathbb{F}^m$. Check if equation $y \overset{?}{=} \mathcal{P}(x)$ hold

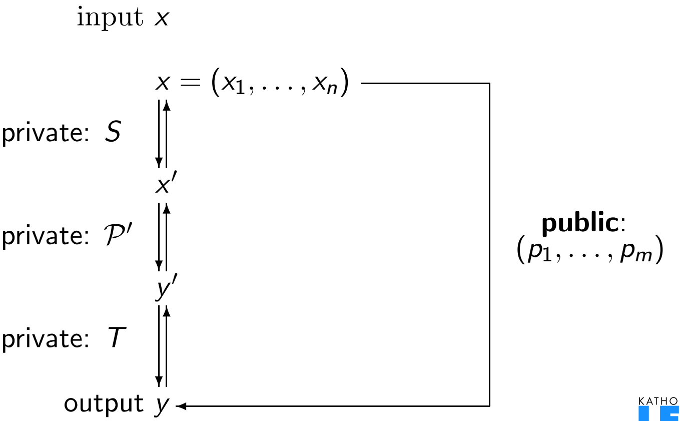
$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

Private Key Computations

- **Inversion of affine transformations:**
Let $S(x) = M_S x + v_s$ for $M_S \in \mathbb{F}^{n \times n}, v_s \in \mathbb{F}^n$. We require M_S being invertible and compute $S^{-1}(x') = M_S^{-1}(x' - v_s)$.
Similar, we can invert $T(y') = y$ for given y
- **Inversion of $\mathcal{P}$':** *Different for each scheme, see later*
- **Signature generation:** invert each step for given $x \in \mathbb{F}^m$, publish the corresponding x as signature of y
- **Decryption:** unique inversion of $\mathcal{P}'$ may not be possible, hence, some redundancy $H := h(x)$ is used with $h(\cdot)$ a cryptographically secure hash function to pick the correct cleartext $x \in \mathbb{F}^n$ for given ciphertext $y \in \mathbb{F}^m$

$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

Graphical Overview (2)



input x

$x = (x_1, \ldots, x_n)$

private: S

x'

private: $\mathcal{P}'$

y'

private: T

output y

public:
$(p_1, \ldots, p_m)$

Outline

Introduction

$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

Basic Classes

Practical Examples

Conclusions

Basic Classes

Unbalance Oil and Vinegar scheme (UOV)

Unbalanced Oil and Vinegar scheme. Uses vinegar and oil variables (v and o). We have $n = v + o$ and need $v = 2o \ldots 3o$ for a secure scheme. Moreover, we have $o = m$.
Central polynomials have the form

$$p'_i(x'_1, \ldots, x'_n) := \sum_{j=1}^{v} \sum_{k=1}^{n} \gamma'_{i,j,k} x'_j x'_k + \sum_{j=1}^{n} \beta'_{i,j} x'_j + \alpha'_i$$

for $1 \leq i \leq m$ and coefficients $\alpha'_i, \beta'_{i,j}, \gamma'_{i,j,k} \in \mathbb{F}$.
Note: These equations become linear if values are assigned to the vinegar variables $x'_1, \ldots, x'_v$

Basic Classes

Unbalance Oil and Vinegar scheme (UOV)

Inversion:

- Assign random values $f_1, \dots, f_v \in \mathbb{F}$ to the vinegar variables
- Partly evaluate the equations in these variables, *i.e.*, compute the polynomials $p'_i(x'_1, \dots, x'_n)|_{x'_1 \leftarrow f_1, \dots, x_v \leftarrow f'_v}$
- The new system $\mathcal{P}'$ of equations is now linear in the o oil variables $x'_{v+1}, \dots, x'_n$
- Solve $\mathcal{P}(x'_{v+1}, \dots, x'_n)$, *e.g.*, using Gaussian elimination
- If $\mathcal{P}(x'_{v+1}, \dots, x'_n) = y'$ does not have a solution for the given $y' \in \mathbb{F}^m$, go back to Step 1 and try another random assignment

Basic Classes

STS — Overview

System $\mathcal{P}'$:

$$\begin{array}{ll}
\text{Step } 1 & \left\{ \begin{array}{rlll}
p'_1 & (x'_1, \ldots, x'_r) & & \\
 & \vdots & & \\
p'_r & (x'_1, \ldots, x'_r) & &
\end{array} \right. \\
\vdots & \\
\text{Step } l & \left\{ \begin{array}{rlll}
p'_{(l-1)r+1} & (x'_1, \ldots, x'_r, & \ldots, & x'_{(l-1)r+1}, \ldots, x'_{lr}) \\
 & & \vdots & \\
p'_{lr} & (x'_1, \ldots, x'_r, & \ldots, & x'_{(l-1)r+1}, \ldots, x'_{lr})
\end{array} \right.
\end{array}$$

Basic Classes

Stepwise Triangular System (STS) — Outline

The scheme is called Stepwise Triangular Schemes (STS), due to the structure of its internal equations which come in *layers* or *steps*.
Inversion: For each layer l, try all possible q^r values of new variables. If there are several possibilities, try all of them for the next step. Hence, the scheme becomes particularly efficient if each layer is a bijection in the new variables.
The scheme from the previous slide is a regular stepwise triangular scheme as all layers have the same number of variables. Using two L-tuples $(n_1, \ldots, n_L), (m_1, \ldots, m_L) \in \mathbb{N}^L$ with $n_1 + \ldots + n_L = n$ and $m_1 + \ldots + m_L$ instead, we obtain the general STS class.

KATHOLIEKE UNIVERSITEIT
LEUVEN

Basic Classes

Matsumoto Imai Scheme A (MIA)

For this scheme (Matsumoto Imai Scheme A), the central equations are over an extension field $\mathbb{E}$ of degree n. They have the form

$$P(X') := X'^{q^\lambda+1}$$

for $q := |\mathbb{F}|$ and some $\lambda \in \mathbb{N}$.
Between $\mathbb{F}^n$ and $\mathbb{E}$, we use a coefficient-wise bijection, *i.e.*, let the vector $a \in \mathbb{F}^n$ be $(a_1, \ldots, a_n)$ with $a_i \in \mathbb{F}$, and let the element $b \in \mathbb{E}$ have the form $b_{n-1}t^{n-1} + \cdots + b_1 t + b_0$ with $b_i \in \mathbb{F}$ and $i(t)$ the defining polynomial of $\mathbb{E}$.
Then we have the bijection $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$ as

$$\phi(a) := b \text{ where } b_{i-1} := a_i \text{ for } 1 \leq i \leq n$$

Basic Classes

Matsumoto Imai Scheme A (MIA)

As X'^{q^λ} is a linear equation over $\mathbb{F}$ for any $\lambda \in \mathbb{N}$ and so is X', their product leads to quadratic equations over $\mathbb{F}$.
To obtain a bijection, we also need $\gcd(q^n - 1, q^\lambda + 1) = 1$. This way, we can compute some $h \in \mathbb{N}$ such that $(q^\lambda + 1).h \equiv 1 \pmod{q^n - 1}$.
In particular, we now have the following equation in $\mathbb{E}$:

$$Y'^h = P(X')^h = X^{(q^\lambda+1).h} = X'$$

and can hence compute efficiently X' for given $Y' \in \mathbb{E}$.

Basic Classes

Hidden Field Equations (HFE)

Hidden Field Equations. Use a similar idea as MIA, but exploit a different idea for the trapdoor.
As for MIA, the central equations are over an extension field $\mathbb{E}$ of degree n. They have the form

$$P'(X') := \sum_{\substack{0\le i,j\le d\\ q^i+q^j\le d}} C'_{i,j} X'^{q^i+q^j} + \sum_{\substack{0\le k\le d\\ q^k\le d}} B'_k X'^{q^k} + A'$$

$$\text{where} \left\{ \begin{array}{ll} C'_{i,j}X^{q^i+q^j} & \text{for } C'_{i,j} \in \mathbb{E} \text{ are the quadratic terms,} \\ B'_k X^{q^k} & \text{for } B'_k \in \mathbb{E} \text{ are the linear terms, and} \\ A' & \text{for } A' \in \mathbb{E} \text{ is the constant term} \end{array} \right.$$

for $i, j \in \mathbb{N}$ and some degree $d \in \mathbb{N}$.

Basic Classes

Hidden Field Equations (HFE)

In contrast to MIA, we do not have a bijection anymore for $P'(X')$ but any polynomial with the above coefficients. However, for the degree d in the range of $d = 129 \ldots 513$ it is possible to obtain both secure and also fast schemes as we can solve the equation $P'(X') = Y'$ for given $Y' \in \mathbb{E}$ and unknown X', *i.e.*, for the inversion step.
As for MIA, we use the bijection $\phi : \mathbb{F}^n \to \mathbb{E}$ as

$$\phi(a) := b \text{ where } b_{i-1} := a_i \text{ for } 1 \leq i \leq n$$

to map elements from the vector space $\mathbb{F}^n$ to the extension field $\mathbb{E}$ and vice versa.

Modifiers

Motivation

Except for UOV, all basic classes are broken. To derive secure schemes, there is a need for “modifiers”.
In all cases, the basic trapdoors from above are used but with a slight “twist” on one of there properties. If well chosen, these modifiers strengthen the resulting scheme against all known attacks. If chosen badly, they actually weaken it.

Modifiers
Motivation

So far, the following generic modifiers are known:

- Adding Equations: "+"
- Removing Equations: "-"
- Public Key in a Subfield: "/"
- Splitting the Trapdoor into Branches: "$\perp$"
- Fixing Variables: "f"
- Internal perturbating equations: "i"

Modifiers

Plus: "+"

Basic idea: pick $a \in \mathbb{N}$ random equations

$$p'_{n+i}(x'_1, \ldots, x'_n) := \sum_{j=1}^{v} \sum_{k=1}^{n} \gamma'_{n+i,j,k} x'_j x'_k + \sum_{j=1}^{n} \beta'_{n+i,j} x'_j + \alpha'_{n+i}$$

with $1 \leq i \leq a$ and coefficients in $\mathbb{F}$.
These random equations are then mixed with the normal trapdoor equations by the new affine transformation $T \in \mathrm{AGL}_{n+a}(\mathbb{F})$.
The size of the public key increases by $O(an^2)$, as the new equations need more space.

Modifiers

Plus: "$+$"

For signature generation, we have a probability of q^{-a} that these equations are satisfied as we did not embed any special trapdoor into them. Hence, signature generation will be slowed down by this factor.

For decryption, these extra equations are always satisfied for the correct choice of x'. Hence, they do not slow down the decryption process.

All in all, the modification looks insecure. At least, all variations of the above basic trapdoors with this modifier have been broken. Its use is therefore not recommended.

Modifiers

Minus: "-"

In contrast to "$+$", we now modify the public key by removing $r \in \mathbb{N}$ of its equations.

This way, decryption is slowed down by q^{-r}. Signature generation on the other hand is not affected as we can always substitute random values for the missing polynomials. For signature verification, these values are not checked anyway.

Quite strong modification for the MIA and HFE trapdoor. For r high enough, all known attacks are countered.

Modifiers

Subfield "/"

The aim is to have all coefficients in the public key in a smaller set, *e.g.*, a subfield $\tilde{\mathbb{F}}$.

To achieve this, all coefficients in the affine transformations S, T but also the central equations $\mathcal{P}'$ must be in this subfield $\tilde{\mathbb{F}}$.

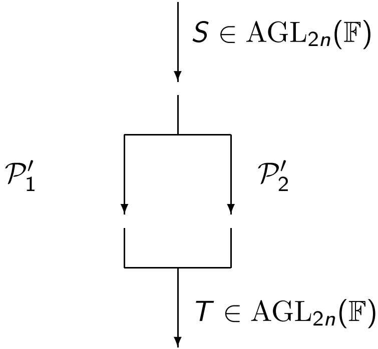
As a result, the public key becomes $\frac{\log|\tilde{\mathbb{F}}|}{\log|\mathbb{F}|}$ smaller.

All basic trapdoors using this modifier have been broken so far.

Hence, its use is strongly discouraged.

Modifiers

Branches: “⊥”



$S \in \mathrm{AGL}_{2n}(\mathbb{F})$

$\mathcal{P}'_1$

$\mathcal{P}'_2$

$T \in \mathrm{AGL}_{2n}(\mathbb{F})$

Modifiers

Branches: "$\perp$"

Instead of having one trapdoor, this modification uses $B \in \mathbb{N}$ branches of the same trapdoor. We show an easy example with two trapdoors $\mathcal{P}'_1, \mathcal{P}'_2 \in \mathcal{MQ}_n(\mathbb{F}^n)$.
This modification gives a nice speed up. However, there exists an algorithm which retrieves these branches in $O(n^6)$. The use of this modification is therefore strongly discouraged.

Modifiers

Fixing: "f"

Basic idea: take the public key and partly evaluate it. This means that some variables $x_{n-f+1}, \dots, x_n$ will be assigned random values and the public key will be changed accordingly.

For decryption, this is no problem as these constraints are satisfied by construction. However, for signature generation, we are now slowed down by $O(q^f)$ as we only have a probability of q^{-f} that these random values are met by a random signature.

Its security has not been studied deeply. However, we get more equations than variables now. This makes some attacks more efficient. Hence, this modification was rejected quite early.

Modifiers

Internal perturbation: "i"

Basic idea: every private key polynomial $p'_i(x'_1, \ldots, x'_n)$ for $1 \leq i \leq n$ is "perturbated" by another polynomial $\phi'_i(x_1, \ldots, x_w)$ for $w \in \mathbb{N}$ with $w \ll n$ and $1 \leq i \leq n$. These polynomials ϕ'_i have the same form as the original p'_i but are in far less input variables. As there is no trapdoor for the perturbation polynomials, we have an additional workload proportional to q^w. In contrast to the "f" and the "+" modification, we still have 1 solution on average to find a signature for a given message. However, decryption is slowed down by q^w.

Initially introduced with the MIA trapdoor — but proved insecure here. May be more interesting with the HFE trapdoor (no attack known so far).

Outline

Introduction

$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

Basic Classes

Practical Examples

Conclusions

Practical Examples

Random Number Generation

Exploit the fact that $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic equations are difficult to solve, namely $\mathcal{NP}$-complete. Secure parameters would be

Seed	Parameter	$\mathcal{MQ}$-System	Evaluation
[bit]		[kByte]	[ms]
259	$q = 128,\ m = n = 37$	23	< 1
469	$q = 128,\ m = n = 67$	134	< 1

The evaluation time has been computed for a PC. However, we only used “full” $\mathcal{MQ}$-systems for this proposal. Using specially designed equations, we could both reduce the number of variables and the size of these systems.

Practical Examples

Electronic Stamps

Need fast signature generation, fast signature verification. Large public keys are no problem as we would not change the public keys too often. Main concern: high through-put of messages and low signature expansion.

Hash [bit]	Parameter	Priv. Key [kByte]	Pub. Key [kByte]	Sign [ms]	Verify [ms]	Expansion [bit]
160	$q = 128$ $n = 67$ $r = 11$	7.8	112.3	< 1	< 1	237

KATHOLIEKE UNIVERSITEIT LEUVEN

Practical Examples

Electronic Stamps

The above parameters have been taken from Quartz. Note the low signature expansion rate. However, signature generation time now goes up to 5 seconds (extrapolation from Quartz).

Message [bit]	Parameter	Pub. Key [kByte]	Sign [ms]	Verify [ms]	Expansion [bit]
173	$q = 2$ $n = 173$ $r = 10$	310.2	5,000	< 5	10

Practical Examples

Quartz Signature Scheme

Quartz has been suggested as a signature scheme in the European NESSIE project. Below we summarize the parameter for its secure variation Quartz-7m.

Parameter	Priv. Key [kByte]	Pub. Key [kByte]	Sign [ms]	Verify [ms]	Signature [bit]
$q = 2$ $n = 107$ $r = 7$	3	71	10,000	< 1	128

KATHOLIEKE UNIVERSITEIT LEUVEN

Outline

Introduction

$\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Cryptography

Basic Classes

Practical Examples

Conclusions

Conclusions

- There are only 4 basic trapdoors for multivariate schemes known so far — and 3 of them are broken in their basic version
- However, they may be combined with several modifiers
- We know that some of the modified versions (*e.g.*, HFE- instead of HFE, MIA- instead of MIA) are much stronger. Hence, a more systematic study of this topic may be useful

Conclusions

- A general characteristic of $\mathcal{MQ}$-systems is that they allow several "tweaks" and trades, *e.g.*,
  - key size vs. generation time
  - signature size vs. generation time
  - key size vs. message expansion
- $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic signature schemes have been investigated since 2 decades by now. The theory is developed, we are now ready to do the step from concepts to products

KATHOLIEKE UNIVERSITEIT LEUVEN

Thank you for your attention!

Questions?