Multivariate public key cryptography with polynomial composition

Emile HAUTEFEUILLE
emile.hautefeuille@polytechnique.edu
École Polytechnique, France

2021

Abstract

This paper presents a new public key cryptography scheme using multivariate polynomials in a finite field. Each multivariate polynomial from the public key is obtained by secretly and repeatedly composing quadratic polynomials (of a single variable) with linear combinations of the input variables. Decoding a message involves recursively computing the roots of these quadratic polynomials, and finally solving some linear systems. The main drawback of this scheme is the length of the public key.

1 Introduction

Since the publication of the Diffie–Hellman key exchange [1] and the RSA cryptosystem [2], public key cryptography has been a major field in cryptography with thousands of applications.

A new threat has appeared with the emergence of quantum computing, leading to the development of post-quantum cryptography, aiming to replace current standards (RSA and Elliptic-curve protocols). Many directions have been explored: lattice-based cryptography, code-based cryptography, supersingular isogeny cryptography, multivariate cryptography...

The scheme presented in this article is part of multivariate cryptography: it is based on the supposedly hard computational problem of finding the roots of a system of multivariate polynomials over a finite field.

Famous schemes at the origin of multivariate cryptography are due to Jacques Patarin, including Hidden Field Equations [3] and Unbalanced Oil and Vinegar (UOV) [4] (with Louis Goubin). Most of the multivariate public key schemes, like UOV, are based on the same idea: the secret key consists of affine transformations and quadratic multivariate polynomials with a trapdoor, hidden by composing both types of functions. The resulting public key is a set of quadratic multivariate polynomials.

This paper proposes a new public key scheme using a different approach. This time, the multivariate polynomials from the public key have a degree higher than 2. This means that the public key generated by this protocol is quite long. As a consequence, this new scheme does not claim to overpass the best known protocols in multivariate cryptography.

A python implementation of the scheme can be found here:
github.com/mi10e3/pkc-quadratic-composition

2 Preliminaries

2.1 Public-key cryptography

A public key cryptography protocol generates two keys: a secret key and a public key. The secret key must be kept private, while the public key can be shared publicly. Everyone can use the public key to encrypt a message. Only the owner of the secret key must be able to reverse the operation and to decrypt the message.

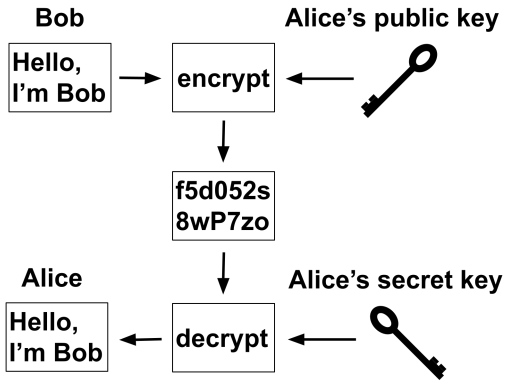

Figure 1: Example of communication using public key cryptography: only Alice can decrypt the message sent by Bob thanks to her secret key.

2.2 Context

For what follows, let's p be a prime number. We will place ourselves in $\mathbb{F}_p$, the finite field of p elements.

The message we want to encrypt will be represented by i elements of $\mathbb{F}_p$: $(x_1, ..., x_i)$, with $i \geq 2$.

The encrypted message will be represented by j elements of $\mathbb{F}_p$: $(y_1, ..., y_j)$, with $j \geq i$.

We will use the ring of polynomials in i variables $(x_1, ..., x_i)$. However, this scheme also involves polynomials of a single variable. For the sake of clarity, the number of variables in the polynomials will always be specified.

In this paper, a *linear combination* of $(x_1, ..., x_i)$ refers to a function of the form:

$$(\mathbb{F}_p)^i \to \mathbb{F}_p$$

$$(x_1, ..., x_i) \mapsto \lambda_0 + \sum_{u=1}^{i} \lambda_u x_u \qquad \text{with } \lambda_0, \lambda_1, ..., \lambda_i \text{ fixed in } (\mathbb{F}_p)^{i+1}$$

(We allow ourselves to add a constant coefficient).

2.3 Computing square roots in $\mathbb{F}_p$

The main idea of this scheme lies in the fact that it is easy to compute a square root modulo p. First, checking if $a \in \mathbb{F}_p$ admits a square root (when there exists $x \in \mathbb{F}_p$ with $x^2 = a$) can be done quickly with Euler's criterion:

$$a^{\frac{p-1}{2}} = \begin{cases} 1 \pmod p \text{ when } a \text{ admits a square root} \\ -1 \pmod p \text{ otherwise} \end{cases}$$

This test can be done in $O(log(p))$ modular multiplications with fast exponentiation. When a square root $x = \sqrt{a}$ exists, its opposite $-x$ is also a square root of a.

The computation of a square root in $\mathbb{F}_p$ is also quick:

- If $p = 3 \pmod 4$, $\sqrt{a} = \pm a^{\frac{p+1}{4}}$ (again $O(log(p))$ modular multiplications with fast exponentiation)
- If $p = 1 \pmod 4$, the Tonelli–Shanks algorithm [5] will compute a square root of a in $O(log(p)+l^2)$ modular multiplications [6], where l is defined by $p - 1 = 2^l u$, with u odd.

For general values of p, l is negligible compared to $log(p)$, and the Tonelli–Shanks algorithm works in $O(log(p))$ modular multiplications.

2.4 Solving quadratic equations in $\mathbb{F}_p$

To solve a quadratic equation in $\mathbb{F}_p$: $ax^2 + bx + c = 0$, we do as if we were in $\mathbb{R}$: we compute the discriminant: $\Delta = b^2 - 4ac$. If it admits a square root, we compute the 2 solutions with the well known formula: $x = (-b \pm \sqrt{\Delta})/2a$, otherwise the equation has no solutions. The computational complexity lies in the square root, which involves $O(log(p))$ modular multiplications (as mentioned in section 2.3).

2.5 Probability of existence of a solution for quadratic equations

Let's consider a quadratic polynomial $f(x) = ax^2 + bx + c$, with $x, a, b, c \in \mathbb{F}_p$ ($a \neq 0$).

With y fixed in $\mathbb{F}_p$, the equation (of unknown x) $f(x) = y$ has:

- 1 solution when $\Delta = 0 \Leftrightarrow b^2 - 4a(c - y) = 0 \Leftrightarrow y = c - \frac{b^2}{4a}$: it happens only for one value of y
- 2 solutions for n different values of y
- 0 solutions for $p - n - 1$ different values of y

As there are p different values for x, $1 + 2n = p \Leftrightarrow n = \frac{p-1}{2}$. The probability for a random quadratic equation in $\mathbb{F}_p$ to have solutions is almost equal to $1/2$.

3 The scheme

3.1 Generate the secret key

The secret key is obtained by randomly choosing a series of quadratic polynomials (of one variable) and a series of linear combinations of $(x_1, ..., x_i)$ in $\mathbb{F}_p$.

More precisely, we can consider that the secret key consists of a set of j lines:

The kth line ($1 \leq k \leq j$) consists of m quadratic polynomials (of one variable): $P_k^1, ..., P_k^m$ and one linear combination of $(x_1, ..., x_i)$: A_k.

- The quadratic polynomial (of a single variable) P_k^q ($1 \leq q \leq m$) is defined as:

$$\begin{aligned} P_k^q &: \mathbb{F}_p \to \mathbb{F}_p \\ & x \mapsto \alpha_k^q x^2 + \beta_k^q x + \gamma_k^q \end{aligned}$$

- The linear combination A_k is defined as:

$$\begin{aligned} A_k &: (\mathbb{F}_p)^i \to \mathbb{F}_p \\ & (x_1, ..., x_i) \mapsto \lambda_k^0 + \sum_{u=1}^{i} \lambda_k^u x_u \end{aligned}$$

Thus, the secret key is created by randomly choosing (in $\mathbb{F}_p$) the following values:

For $k \in \{1, 2, ..., j\}$:

- $\alpha_k^q, \beta_k^q, \gamma_k^q$ (for $q \in \{1, 2, ..., m\}$)
- $\lambda_k^0, \lambda_k^1, ..., \lambda_k^i$

3.2 Generate the public key

The public key consists of j multivariate polynomials (of i variables) on $\mathbb{F}_p$. The kth ($1 \leq k \leq j$) multivariate polynomial is computed as the composition of the quadratic polynomials (of a single variable) and the linear combination of the kth line from the secret key.

More precisely, the public key is defined by j multivariate polynomials: $M_1, ..., M_j$.

For $k \in \{1, 2, ..., j\}$, $M_k : (\mathbb{F}_p)^i \to \mathbb{F}_p$ is computed by $M_k = P_k^1 \circ P_k^2 \circ ... \circ P_k^m \circ A_k$.

The main idea is that the multivariate polynomials of the public key are given in their expanded form:

$$M_k(x_1, ..., x_i) = \sum_{u_1+...+u_i \leq 2^m} \eta_k^{(u_1,...,u_i)} \times x_1^{u_1} \times ... \times x_i^{u_i}$$

The secret key contains the same multivariate polynomials, but in a factored form: a composition of simple quadratic polynomials with a linear combination of $(x_1, ..., x_i)$.

3.3 Encrypt a message

Let's consider a message $(x_1, ..., x_i) \in (\mathbb{F}_p)^i$ and a public key $M_1, ..., M_j$.

For each $k \in \{1, 2, ..., j\}$, we compute:

$y_k = M_k(x_1, ..., x_i)$

The encrypted message is $(y_1, ..., y_j) \in (\mathbb{F}_p)^j$.

3.4 Decrypt a message

Let's consider an encrypted message $(y_1, ..., y_j) \in (\mathbb{F}_p)^j$ and a secret key corresponding to the public key used for encryption: $((P_k^q, \text{ for } q \in \{1, 2, ..., m\}), A_k)$ for $k \in \{1, 2, ..., j\}$.

For $k \in \{1, 2, ..., j\}$:

$$\begin{aligned} & y_k = M_k(x_1, ..., x_i) \\ \Leftrightarrow\ & y_k = P_k^1 \circ ... \circ P_k^m \circ A_k(x_1, ..., x_i) \\ \Leftrightarrow\ & y_k = P_k^1(X) \text{ with } X = P_k^2 \circ ... \circ P_k^m \circ A_k(x_1, ..., x_i) \end{aligned}$$

Since $y_k = P_k^1(X)$ is simply a quadratic equation of X, we can easily compute its 2 solutions (see section 2.3 and 2.4). If the equation does not admit solutions, it means there is no possible message $(x_1, ..., x_i)$ whose encryption gives $(y_1, ..., y_j)$. If we name X_1 and X_2 the 2 solutions of the previous equation, it becomes:

$$P_k^2(P_k^3 \circ ... \circ P_k^m \circ A_k(x_1, ..., x_i)) = \begin{cases} X_1 \\ X_2 \end{cases}$$

In each case (X_1 or X_2), we can repeat the operation. We can visualize it with a tree (see Figure 2).

Note that some branches of the tree may stop, because the corresponding quadratic equation has no solutions. However, if $(y_1, ..., y_j)$ has been obtained by encrypting a message, we can be sure that the tree will go down to at least 2 solutions of $y_k = P_k^1 \circ ... \circ P_k^m(Z)$ (with unknown $Z \in \mathbb{F}_p$).

Let $Z_k^1, Z_k^2, ..., Z_k^{t_k}$ be the solutions of the previous equation we found by iterating the process described above. This process that we can call 'recursive inverse composition' captures all possible solutions.

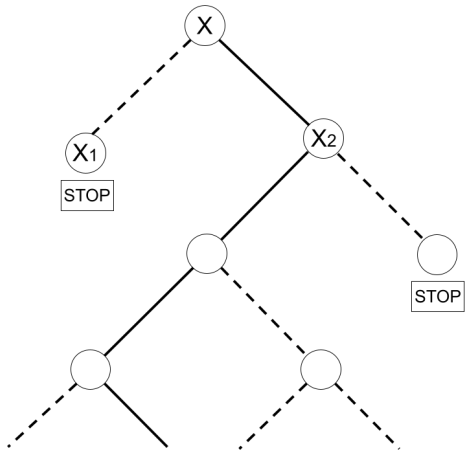

Figure 2: A tree representing the 'recursive inverse composition' of $m = 4$ quadratic polynomials (of a single variable).

Thus, we have: $$A_k(x_1, ..., x_i) = \begin{cases} Z_k^1 \\ ... \\ Z_k^{t_k} \end{cases} \Leftrightarrow \lambda_k^0 + \sum_{u=1}^{i} \lambda_k^u x_u = \begin{cases} Z_k^1 \\ ... \\ Z_k^{t_k} \end{cases}$$

This being true for each k ($1 \leq k \leq j$), we need to check for all the possible configurations between all the Z. Checking a configuration means:

- Selecting, for each k ($1 \leq k \leq j$), a Z in $\{Z_k^1, ..., Z_k^{t_k}\}$.
- Solving the resulting linear system (of j lines and i unknowns). There may not be a solution.

Solving all the linear systems will result in finding all the possible messages $(x_1, ..., x_i)$ whose encryption gives $(y_1, ..., y_j)$.

4 Practical aspects

4.1 Parameters of the scheme

This protocol has 4 parameters:

- p: the characteristic of the finite field in which the the calculations are done.
- i: the number of variables representing the message to encrypt.
- j: the number of variables representing the encrypted message.
- m: the number of quadratic polynomials (of a single variable) to compose for each line of the secret key.

4.2 Size of the keys

The size of the secret key is quite small, it consists of $j \times (3m + i + 1)$ elements of $\mathbb{F}_p$, represented by integers between 0 and $p - 1$ (for each of the j lines, there are m quadratic polynomials (of a single variable), defined by 3 numbers, and one linear combination of i variables, plus a constant coefficient).

The main drawback of this protocol is the length of the public key: $j \times \binom{2^m+i}{i}$ integers between 0 and $p - 1$ (each of the j lines produces a multivariate polynomial of i variables, with degree 2^m, because of the m quadratic compositions, resulting in $\binom{2^m+i}{i}$ monomials).

4.3 Complexity of encryption

The encryption of $(x_1, ..., x_i)$ consists of the evaluation of j multivariate polynomials in $\mathbb{F}_p$. Each polynomial has i variables and a degree of 2^m: it contains $\binom{2^m+i}{i}$ monomials. A classical approach for a fast evaluation is:

- Evaluate first all the monomials without considering the coefficients of the polynomials (compute $X^3Y^2Z^5$, not $134.X^3Y^2Z^5$).
- To do so, start from the monomials of low degree: degrees 0 and 1 need no multiplications.
- Compute the monomials (without coefficient) of degree $k+1$ from those of degree k (stored in memory), as it only involves one multiplication: to compute $X^3Y^2Z^5$, multiply X by $X^2Y^2Z^5$ (already calculated).

This process requires $\binom{2^m+i}{i} - i - 1$ multiplications. If we now consider the polynomial's coefficients, we get to a total of $2 \times \binom{2^m+i}{i} - i - 2$ multiplications.

Finally, $\binom{2^m+i}{i} - 1$ additions are needed to get the result of the evaluation.

As a result, the encryption of $(x_1, ..., x_i)$ can be done in $j \times (2 \times \binom{2^m+i}{i} - i - 2)$ modular multiplications and $j \times (\binom{2^m+i}{i} - 1)$ modular additions.

4.4 Complexity of decryption

The number of operations needed to decrypt $(y_1, ..., y_j)$ depends on the number of solutions of:

$$P_k^1 \circ ... \circ P_k^m(X) = y_k \qquad \text{(for each } k \in \{1, 2, ..., j\})$$

Given $k \in \{1, 2, ..., j\}$, at the step $d \in \{1, 2, ..., m\}$ of the 'recursive inverse composition', we know that:

$$P_k^d \circ ... \circ P_k^m(X) \in \Gamma_k^d \qquad \text{with } \Gamma_k^d \text{ a subset of } \mathbb{F}_p$$

(initially $d = 1$ and $\Gamma_k^1 = \{y_k\}$ for instance).

If $(y_1, ..., y_j)$ was obtained by encrypting $(x_1, ..., x_i)$, we know for sure that one value $\tau_k^d \in \Gamma_k^d$ leads to the decrypted message (precisely, τ_k^d corresponds to $X = P_k^{d+1} \circ ... \circ P_k^m \circ A_k(x_1, ..., x_i)$). For all the other values $\tau' \in \Gamma_k^d \backslash \{\tau_k^d\}$, there is a probability $1/2$ that $P_k^d(X) = \tau'$ admits solutions (see section 2.5). When solutions are found, they come as a distinct pair of values, with probability $1 - \frac{2}{p+1} \approx 1$ (there is the rare case of a root of multiplicity 2, causing the $\frac{2}{p+1}$ term).

Thus, the 'recursive inverse composition' of $P_k^1 \circ ... \circ P_k^m$ can be seen as probabilistic binary tree (see Figure 2):

- Initially, the tree contains only one node, its root.
- At each step $d \in \{1, 2, ..., m\}$, each of the leaf nodes with maximal depth (in other words, with depth $d-1$) has a probability $1/2$ to expand into 2 additional edges, except for one leaf node, chosen randomly, where this probability is 1.

We denote by T_d ($d \in \{1, 2, ..., m\}$) the random variable counting the number of nodes at depth d. The description above leads to:

- $T_1 = 2$ (The root node expands with probability 1).
- $T_{d+1} = 2 + \sum_{u=1}^{T_d - 1} (2X_d^u + 0.(1 - X_d^u))$, with the X_d^u being independent random variables following the Bernoulli distribution of parameter $1/2$.

In the last expression, the +2 comes from the node (present at depth d) which expands with probability 1. For the other $T_d - 1$ nodes, the probability of expansion into 2 additional edges is $1/2$, the other issue being not to expand.

As a consequence:

$$\mathbb{E}(T_{d+1}) = 2 + \mathbb{E}(\sum_{u=1}^{T_d - 1} 2X_d^u) = \mathbb{E}(T_d) + 1$$

Leading to $\mathbb{E}(T_d) = d + 1$.

This gives the average number of nodes with depth $\leq m - 1$:

$$\mathbb{E}(1 + T_1 + T_2 + ... + T_{m-1}) = 1 + 2 + 3 + ... + m = m(m+1)/2 = O(m^2)$$

(the initial 1 stands for the root node, with depth 0).

During the decryption, each of these nodes represents the resolution of a quadratic equation in $\mathbb{F}_p$. As mentioned in section 2.4, $O(log(p))$ modular multiplications are needed to solve a quadratic equation in $\mathbb{F}_p$. As a result, the 'recursive inverse decomposition' of all the j lines requires on average $O(jm^2log(p))$ modular multiplications.

At the end of the construction of the probabilistic tree, each leaf node with maximal depth (in other words, with depth m) represents a possible value for $A_k(x_1, ..., x_i)$ obtained during the decryption. Thus, the number of possible values found for $A_k(x_1, ..., x_i)$ is on average equal to $\mathbb{E}(T_m) = m + 1$.

This being true for each linear combination A_k ($k \in \{1, 2, ..., j\}$), the decryption of the message involves on average trying to solve $(m+1)^j$ linear system in $\mathbb{F}_p$. These systems have i variables and j lines, requiring $O(ji^2)$ multiplications with Gaussian elimination. As a result, solving the linear systems requires on average $O(ji^2m^j)$ multiplications.

To conclude, decrypting $(x_1, ..., x_i)$ from $(y_1, ..., y_j)$ involves on average $O(j[m^2log(p) + i^2m^j])$ modular multiplications.

4.5 Implementation suggestions

Because of the length of the public key, we cannot afford too many compositions: a reasonable value for m is probably between 2 and 10.

i should share the same order of magnitude, for the same reason.

j should be greater than i to reduce the risk of collisions, $j \approx i + 2$ seems reasonable (note that there is only a linear dependency between j and the size of the keys).

Choosing a big value for p seems natural to compensate for the small number of variables representing the message (simply to prevent brute force attacks). Note that this protocol can work very well with $p \approx 10^{1000}$ (see the github implementation).

5 Security

The security of this protocol lies in the difficulty of finding the roots of the system of multivariate polynomials used to encrypt messages. However, the multivariate polynomials from this scheme are not fully random, as they are obtained with quadratic compositions. The most promising attack is probably to try to compute back the secret key from the public key.

For $k \in \{1, 2, ..., j\}$, we need to find:

- $\alpha_k^q, \beta_k^q, \gamma_k^q$ (for $q \in \{1, 2, ..., m\}$)
- $\lambda_k^0, \lambda_k^1, ..., \lambda_k^i$

such that:

$$(\alpha_k^1 X^2 + \beta_k^1 X + \gamma_k^1) \circ ... \circ (\alpha_k^m X^2 + \beta_k^m X + \gamma_k^m) \circ (\lambda_k^0 + \sum_{u=1}^{i} \lambda_k^u x_u) \quad = \quad \underbrace{M_k(x_1, ..., x_i)}_{\text{part of the public key}}$$

By developing the left expression, we get a multivariate polynomial of $x_1, ..., x_i$. Each coefficient is a multivariate polynomial of:

- $\alpha_k^q, \beta_k^q, \gamma_k^q$ (for $q \in \{1, 2, ..., m\}$)
- $\lambda_k^0, \lambda_k^1, ..., \lambda_k^i$

By identifying these coefficients with the ones of $M_k(x_1, ..., x_i)$, publicly known, we obtain another system of $\binom{2^m+i}{i}$ multivariate polynomials with $3m + i + 1$ unknowns. An attacker would have to solve such a system for every line of the key (j times in total).

Further work needs to be conducted to understand how this protocol behave against some classical attacks (Gröbner basis for instance).

6 Conclusion

A new public key protocol in multivariate cryptography has been introduced. It is based on the ease of solving quadratic equations (of a single variable) modulo p. Its main drawback is the length of its public key. The choice of the right parameters in order to guarantee security remains unclear. The open source implementation can be used for further investigation on the security of the multivariate polynomial systems it generates. Finally, it would be interesting to assess to what extent this idea of quadratic composition can be used alongside another existing multivariate cryptosystem, to reinforce its security.

7 Acknowledgements

I am grateful to Olivier Blazy (LIX, École Polytechnique) and Fabienne Robinson (École Polytechnique) for their advice.

8 References

[1] W. Diffie and M. Hellman, "New directions in cryptography," in IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, November 1976, doi: 10.1109/TIT.1976.1055638

[2] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 2 (Feb. 1978), 120–126.

[3] Patarin J. (1996) Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer U. (eds) Advances in Cryptology — EUROCRYPT '96. EUROCRYPT 1996. Lecture Notes in Computer Science, vol 1070. Springer, Berlin, Heidelberg.

[4] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In Eurocrypt'99, LNCS, volume 1592, pages 206–222. Springer, 1999.

[5] Tonelli, A.: Bemerkung ûber die Aufl ösung quadratischer Congruenzen. Gottinger Nachrichten (1891) 344–346

[6] Volker Diekert; Manfred Kufleitner; Gerhard Rosenberger; Ulrich Hertrampf (24 May 2016). Discrete Algebraic Methods: Arithmetic, Cryptography, Automata and Groups. De Gruyter. pp. 163–165. ISBN 978-3-11-041632-9