Galois Theory and Homomorphic Polynomial Public Key

Carsi, Ana

Cybersecurity II

August 23, 2024

Abstract

In this report a review on the innovative asymmetric cryptographic method designed by Randy Kuang et al. [1] for the post-quantum era is provided. Its relation with algebra and the properties of finite fields are explained, with the hope to bring understanding on cryptographic methods' design. Its efficiency and security constraints are analysed as well.

Keywords: Galois Theory, Post-Quantum Computing, Polynomial Public Key, homomorphic encryption, ...

I. Introduction

Cryptography has undergone a major evolution in the last decade, driven by the increasing need of stronger post-quantum encryption mechanisms. One promising direction has been homomorphic encryption, which allows encrypted data processing without exposing its underlying information. In 2009, Gentry [**?**] proposed the first fully homomorphic encryption (FHE) scheme with the help of bootstrapping (see Appendix) to which Brakerski and Vaikuntanathan contributed with progressively simplified methods [2].

In parallel, Kuang et al. introduced multivariate polynomial public key cryptography (MPPK) in 2022 [3], which was based on the NP-complete problem of solving modular Diophantine equations for large primes p. The public key was then derived from the coefficients of the resulting polynomial products, with some exceptions, and includes two noise functions.

Building on MPPK, Key Encapsulation Mechanisms (KEM) were developed to enhance the efficiency of key exchange in public key systems. KEM allowed for smaller key sizes and faster operations, addressing some of the practical limitations of earlier public key methods. Afterwards, Kuang et al. developed Homomorphic Polynomial Public Key Cryptography [1], which combines MPPK and KEM while incorporating homomorphic encryption.

This report aims to review HPPK from the core perspective of Galois Theory. In the first section, the mathematical background and the foundations of the scheme are introduced, to be followed by the definition of the HPPK KEM mechanism. In the final section, its application to digital signatures and its corresponding efficiency in this scope are analysed.

II. Mathematical background

A. Homomorphic Encryption

As mentioned, one key feature of HPPK is homomorphic encryption, which allows computations on ciphertext which after decryption match the result of operations performed on the plaintext. Formally, if E is an encryption function and D is the corresponding decryption function, and $*$ and $\cdot$ are operations in the plaintext and ciphertext domains respectively, then:

$$D(E(m_1) * E(m_2)) = m_1 \cdot m_2$$

In the mathematical sense, this means that encryption and decryption have the homomorphic property, i.e. $f : A \to B$ between $(A, *)$ and $(B, \cdot)$ s.t.

$$f(x * y) = f(x) \cdot f(y)$$

for every pair $x, y \in A$.

This way, Homomorphic Encryption enables encrypted data processing without exposing its underlying information.

B. Polynomial Public Key

Homomorphic Polynomial Public Key (HPPK) is an extension of PPK and a specific form of homomorphic encryption. Here, the multivariate polynomials lie as well on finite fields, and the encryption method is proven to be partially homomorphic under addition and multiplication. Both (and therefore MPPK as well) are based on the difficulty of solving systems of multivariate polynomial equations, which is an NP-hard problem. Specifically, these schemes are built upon the Multivariate Quadratic (MQ) problem, which involves finding solutions to a system of multivariate quadratic equations over a finite field. This problem becomes computationally infeasible as the number of variables and equations increases.

C. HPPK KEM

HPPK was originally designed for Key Encapsulation Mechanism [1], aiming to securely generate and share a secret symmetric key between two parties. KEM consists of three steps: key pair generation, encapsulation (encryption) to create a shared secret key, and decapsulation (decryption) for key recovery.

C.1 Key Pair Generation

In KEM, the sender generates the public-private key pair and sends afterwards the public key to the receiver. Firstly, two polynomials $f(x)$ and $h(x)$ on a shared finite field $\mathbb{F}_p$ with p prime are chosen. Then one constructs the multivariate base polynomial $B(x, u_1, ..., u_m)$, where x is the secret and $u_1, ..., u_m$ are randomly chosen noise variables from $\mathbb{F}_p$.

Consider two hidden rings $\mathbb{Z}_{S_1}$, $\mathbb{Z}_{S_2}$ where S_1, S_2 have bit length twice of p and define R_1 and R_2 with $\gcd(R_1, S_1) = 1$ and $\gcd(R_2, S_2) = 1$, which will be needed for encryption.

One then calculates:

$$\begin{aligned} P(x, u_1, .., u_m) &= B(x, u_1, ..., u_m) f(x) \pmod p \\ &= \vec{x}^T \cdot \mathbf{p} \cdot \vec{u} \\ Q(x, u_1, .., u_m) &= B(x, u_1, ..., u_m) h(x) \pmod p \\ &= \vec{x}^T \cdot \mathbf{q} \cdot \vec{u}. \end{aligned}$$

This matrix expression will be further exploited in C.4. Equivalently,

$$P(x, u_1, ..., u_m) = \sum_{j=1}^{m} \sum_{i=0}^{n+\lambda} p_{ij} x^i u_j \pmod p$$

$$Q(x, u_1, ..., u_m) = \sum_{j=1}^{m} \sum_{i=0}^{n+\lambda} q_{ij} x^i u_j \pmod p.$$

Afterwards, we name

$$P' = P(x, u_1, ..., u_m) - P_0$$
$$Q' = Q(x, u_1, ..., u_m) - Q_0$$

with P_0 and Q_0 the constant terms.

C.2 Encapsulation

The receiver uses the sender's public key to encapsulate a randomly generated symmetric key. This process produces a ciphertext that contains the symmetric key encrypted with the public key. Using the secrets S_1 and S_2 and the secret encryption key R_1 and R_2 over the ring, the coefficients of P and Q are encrypted onto $\overline{\mathcal{P}}$ and $\overline{\mathcal{Q}}$ with

$$\overline{p}_{ij} = R_1 \cdot p_{ij} \pmod{S_1}$$

$$\overline{q}_{ij} = R_2 \cdot q_{ij} \pmod{S_2}$$

The resulting ciphertext is $\{\overline{\mathcal{P}}, \overline{\mathcal{Q}}\}$.

C.3 Decapsulation

The decryption of the ciphertext takes place through symmetric homomorphic decryption followed by the division of polynomials.

1. Symmetric homomorphic decryption:

$$\overline{P} = (R_1^{-1} \cdot \overline{\mathcal{P}} \pmod{S_1}) \pmod p$$
$$\overline{Q} = (R_2^{-1} \cdot \overline{\mathcal{Q}} \pmod{S_2}) \pmod p$$

2. Radical solution:

$$\frac{f(x)}{h(x)} = \frac{\overline{P} + f_0 B_0}{\overline{Q} + h_0 B_0} = k \pmod p$$

Note: The process of encapsulation involves therefore "lifting" the polynomial coefficients to two bigger rings S_1 and S_2, and decapsulating or decrypting involves projecting them back to the finite field $\mathbb{F}_p$.

C.4 Resulting Key Pair

- The public key is:

$$\{\mathbf{P}[n+\lambda+1][m]\ ,\ \mathbf{Q}[n+\lambda+1][m]\}$$

with $\vec{x}^T \cdot \mathbf{P} \cdot \vec{u} = \mathcal{P}(x, u_1, ..., u_m)$ and $\vec{x}^T \cdot \mathbf{Q} \cdot \vec{u} = \mathcal{Q}(x, u_1, ..., u_m)$, with $\mathcal{P}$ and $\mathcal{Q}$ the polynomials resulting after the encryption of the coefficients.

- The private key is:

$$\{f[\lambda+1],\ h[\lambda+1],\ R_1,\ S_1,\ R_2,\ S_2\}$$

Here, $\mathbf{P}[]$ and $f[]$ denote the coefficients of the polynomials of greatest degree, which can be seen as a matrix through the dot product in C.1.

III. Algebraic reasoning behind the method

In this section we will justify the construction of HPPK through the mathematical foundations of algebra. Firstly, one can prove that it is indeed a homomorphic encryption scheme, since the operation used in the encryption process C.2 can be defined with the permutation operator:

$$\hat{E}(R, S) = R \cdot a \pmod{S}$$

for a coprime pair of L-bits R and S, where a represents a polynomial.Let

$$\hat{E}(R, S)a = R \cdot a \pmod{S} = a'$$

$$\hat{E}(R, S)b = R \cdot b \pmod{S} = b'$$

be the encryption of polynomials a and b. We see that $\hat{E}(R, S)$ preserves the addition and multiplication

$$\begin{aligned}\hat{E}(R, S)(a+b) &= R \cdot (a+b) \pmod{S} \\ &= \hat{E}(R, S)a + \hat{E}(R, S)b\end{aligned}$$

$$\begin{aligned}\hat{E}(R, S)ca &= R \cdot (ca) \pmod{S} \\ &= ca' \pmod{S} \\ &= c\hat{E}(R, S)a\end{aligned}$$

and therefore it is a homomorphism. This justifies its construction and allows secure computation over encrypted data.

Furthermore, the election of a finite field $\mathbb{F}_p$ provides the following benefits:

1. **Consistency:** the operations of encryption, decryption and arithmetical operations are guaranteed to be well defined.

2. **Prevention of Overflow:** finite fields are inherently bounded, which eliminates the risk of overflow during computations. This ensures the stability of the cryptographic process.

3. **Invertibility:** The property of $\mathbb{F}_p$ being a field guarantees that every non-zero element has a multiplicative inverse, which ensures that encrypted data can be decrypted accurately.

4. **Efficiency through the Chinese Remainder Theorem:** The Chinese Remainder Theorem (CRT) enables the reconstruction of numbers from their remainders modulo coprime integers. Therefore, CRT allows for parallelization and optimization of computation which can be performed more efficiently within smaller modular arithmetic settings. This not only accelerates cryptographic operations but also enhances their security by distributing the computation across multiple smaller fields.

   Additionally, the use of Barrett's reduction algorithm further improves efficiency by minimizing the computational overhead associated with modular reduction.

5. **Security against Algebraic Attacks:** The system is designed to be resilient against algebraic attacks, such as those based on Gaussian elimination and Generalized Riemann Hypothesis. This resilience ensures that the cryptographic scheme remains robust even against advanced mathematical techniques.

Application of Galois Theory in HPPK

The encryption operator, while it can exhibit homomorphic properties, it operates as an automorphism within the algebraic structure defined by the finite field $\mathbb{F}_{2^n}$, ensuring that the operations required for encryption and decryption are well-defined within this field.

Although the encryption operator in HPPK is not necessarily bijective, it is designed to allow for the decryption of encrypted data, meaning that the original information can be recovered accurately.

Relevance of Finite Fields in HPPK

The Galois group associated with a finite field extension $\mathbb{F}_{p^n}$ governs the field's automorphisms—permutations of the field elements that preserve the field's algebraic structure. These automorphisms are valuable in algebraic coding theory and cryptography, as they provide a rich structure for constructing secure encryption schemes. The non-commutative nature of operations in certain Galois groups can add complexity to cryptographic algorithms, which enhances security by making it more difficult for attackers to reverse-engineer the encryption process.

In summary, the application of finite fields and the algebraic properties derived from Galois theory—such as modular arithmetic, non-commutativity, and the ability to define secure permutations—contributes to the robustness of the HPPK scheme. It is as well interesting to note that in the quantum realm, the non-commutativity of the Galois Permutation Group directly corresponds to the uncertainty principle in quantum mechanics.

IV. HPPK DS

HPPK was then reframed for digital signature (DS) in [4] introducing an extended Barrett reduction algorithm, which transforms modular multiplications over hidden rings into divisions in the verification equation. This non-linear embedding of signatures into public polynomial coefficients overcomes vulnerabilities associated with linear relationships in the earlier MPPK, resulting in exponential complexity for both private key recovery and forged signature attacks.

Signing

The process begins with the signer generating the hash code $x \leftarrow HASH(M)$ by means of a chosen cryptographic hash function. Then the private key to sign x is used:

$$F = R_2^{-1} * [\alpha f(x) \pmod p] \pmod{S_2}$$

$$H = R_1^{-1} * [\alpha h(x) \pmod p] \pmod{S_1}$$

where α is a randomly chosen integer from $\mathbb{F}_p$. The HPPK verification equation then results

$$\vec{x}^T \cdot (f'\mathbf{q}) \cdot \vec{u} = \vec{x}^T \cdot (h'\mathbf{p}) \cdot \vec{u} \pmod p$$

with $f' = \alpha f(x) \pmod p$ and $h' = \alpha h(x) \pmod p$. Applying the encryption operators $\hat{E}(R, S) = R \cdot a \pmod S$ we get:

$$\begin{aligned} &(\vec{x}^T \cdot (F\mathbf{Q} \pmod{S_2} \cdot \vec{u}) \pmod p \\ =\ &(\vec{x}^T \cdot (H\mathbf{P} \pmod{S_1} \cdot \vec{u}) \pmod p. \end{aligned}$$

Naming $V = F\mathbf{Q}$ and $U = H\mathbf{P}$ we get the expression:

$$\begin{aligned} &V(x, u_1, ..., u_m) \pmod p \\ =\ &U(x, u_1, ..., u_m) \pmod p \end{aligned}$$

Finally, since S_1 and S_2 are unknown for the verifier, the variables must be calculated using the Barrett reduction algorithm.

Barret Reduction algorithm

The Barrett Reduction algorithm is an efficient method for reducing large integers modulo a smaller integer, which is utilized to manage the reduction of polynomial coefficients after encryption. The algorithm operates as follows:

Given a large integer x, modulus S, and a precomputed value $\mu = \left\lfloor \frac{2^{2k}}{S} \right\rfloor$, where k is the bit-length of S:

$$q_1 = \left\lfloor \frac{x}{2^k} \right\rfloor$$

$$q_2 = \left\lfloor \frac{\mu \cdot q_1}{2^k} \right\rfloor$$

$$r = x - q_2 \cdot S$$

The result r represents the remainder when x is divided by S, which is used to reduce the coefficients of the polynomials efficiently.

Algorithm 1 barrett_reduce

```
function BARRETT_REDUCE(x, mod, mu)
    q1 = x >> 32
    r1 = x & 0xFFFFFFFF
    q2 = ((mu · q1) >> 32)· mod
    r2 = mu· r1
    q3 = r2 >> 32
    r3 = r1 - q3 · mod
    remainder = r3 + ((r3 >> 63) & mod)
    if remainder < 0 then
        remainder+ = mod
    end if
    return remainder
end function
```

Figure 2: *Barret Reduction algorithm*

Verification Process

The verifier, upon receiving the signature $\text{Sig} = \{F, H\}$, computes the corresponding polynomials V and U using the public key. By comparing these computed values with the received signature, the verifier can ascertain the validity of the signature:

1. $x = HASH(M)$ and randomly choose $u_1, ..., u_m \in \mathbb{F}_p$.
2. Evaluate $U_{ij}(H)$ and $V_{ij}(F)$ for $i = 0, ..., n + \lambda$, $j = 1, ..., m$ in

$$U_{ij}(H) = H(x)p'_{ij} - s_1 \lfloor \frac{H(x)b_{ij}}{R} \rfloor \pmod{p}$$

$$U_{ij}(H) = F(x)q'_{ij} - s_2 \lfloor \frac{F(x)a_{ij}}{R} \rfloor \pmod{p}$$

3. Evaluate $V(F, x, u_1, ..., u_m)$ and $U(H, x, u_1, ..., u_m)$ to check if they are equal.

The security of this scheme is guaranteed because the private key components $f(x)$ and $h(x)$, along with the values R_1 and R_2, remain unknown to the verifier, thereby making it infeasible to forge a valid signature.

V. Security: Advantages and Challenges

Entropy

Moreover, the operation $((R \cdot b) \bmod S)$ serves as a fundamental arithmetic permutation in the cryptographic scheme. When S is kept confidential and has a known bit length L, the number of possible permutations is given by

$$\varphi(S) \cdot 2^L \tag{1}$$

where φ is the Euler's totient function.

The large key space enabled by the modular multiplication mentioned in 1 reinforces the security of the cryptosystem, as it dramatically increases the number of potential keys that an attacker would need to consider. Additionally, the quantum key space in this setting holds an entropy $E = \log_2(2^n!)$, representing the vast number of possible configurations after retrieving the key operators, assuming an equally likely distribution. This high entropy is a measure of the system's unpredictability and resilience against brute-force attacks.

Discovering the symmetric encryption key involves knowing R_1 and S_1 and R_2 and S_2, achieved through random guessing with a complexity of $O(S_1^2)$ and $O(S_2^2)$ respectively. For each guessed S_1, the attacker must brute-force R_1 and test if R_1 is coprime with S_1 (respectively for S_2).

Private key recovery attack

This attack has a classical computational complexity of $O(p(S_1/p * S_2/p))$, where p is a prime value security parameter, and S_1 and S_2 are hidden ring values. The attack involves Brute-force searching for values to find S_1 and S_2, using intercepted signature values and public key elements to recreate certain expressions, and exploiting the structure of the verification equation to forge signatures IV. An improved version of the attack has a classical computational complexity of $O(p^3)$ when the parameters are optimally chosen ($|S_1|^2 = |S_2|^2 = 2|p|^2$).

The threat level of these attacks depends on the chosen parameters. With proper parameter selection, the HPPK digital signature scheme can maintain a high level of security against these attacks.

VI. Conclusion

In this paper we have explored the mathematical foundations and cryptographic applications of HPPK and conclude it offers a robust

framework for secure data encryption. The scheme's inherent homomorphism supports secure arithmetic operations on encrypted data, making it particularly well-suited for complex cryptographic tasks, such as key encapsulation and digital signatures.

However, it is important to acknowledge the computational trade-offs associated with HPPK. While the scheme provides strong security guarantees, its efficiency on classical computers may not be optimal due to the complexity of the underlying algebraic operations. The computational overhead introduced by finite field arithmetic and Barrett reduction can be significant, making HPPK less practical for applications that require real-time processing or limited computational resources.

In conclusion, while HPPK may not be the most efficient choice for classical computing environments, its design principles position it as a forward-looking solution that anticipates the demands of post-quantum cryptography. Future research and advancements in quantum computing could further optimize HPPK, enhancing its practical utility across various applications.

Appendix

Efficiency of Homomorphic Encryption

The efficiency of fully homomorphic encryption has been the big question following its invention [**?**]. During Gentry's construction of the first plausible FHE Scheme, the technique of bootstrapping was introduced to manage noise growth in ciphertexts. As homomorphic operations are performed, noise in the ciphertext increases, eventually making it undecipherable. Bootstrapping "refreshes" a ciphertext by homomorphically evaluating the decryption function on it, using an encrypted secret key given in the public key. This process produces a new ciphertext with less noise, allowing for further computations.

To compute its efficiency, one can calculate:

$$p = \frac{time_E}{time_D}$$

where $time_E$ stands for the time cost of a computation on the encrypted data (performed homomorphically) and $time_D$ is the corresponding on decrypted data. p is known as the *per-gate overhead*. In early FHE schemes as the Gentry-Halevi implementation, a timing of approximately 30 minutes per simple bit operation was reported [**?**]. Therefore bootstrapping was considered, although useful, a major bottleneck.

Afterwards, researches Zvika Brakerski and Vinod Vaikuntanathan improved the efficiency of FHE schemes, introducing Learning With Errors (LWE) as a conceptually simpler method in comparison to Gentry's original ideal lattices. It involves trying to solve a system of linear equations with some added noise, which is believed to be computationally hard. In LWE-based encryption: a) the secret key is a random vector, b) encryptions are noisy linear combinations of this vector, c) decryption involves solving these noisy equations.

This scheme could evaluate circuits (sequence of operations, like addition and multiplication, that can be performed on encrypted data) of predetermined length without using Gentry's bootstrapping technique.

Deterministic Polynomial Public Key

DPPK (Deterministic Polynomial Public Key) is a public key scheme proposed by Kuang in 2021 [5] for key exchange purposes consisting of two solvable univariate polynomials $f(x)$ and $h(x)$ multiplying with a randomly chosen polynomial $B(x)$ (in contrast to MPPK and HPPK, where $B(x)$ is multivariate. The public key is constructed similarly through polynomial multiplications over a finite field, excluding their constant terms.

However, while secure against key recovery from the public key, it was found to be vulnerable to secret recovery using deterministic factoring techniques. Evdokimov [6] determined that DPPK is susceptible to a secret recovery attack from the ciphertext under the Generalized Riemann Hypothesis (GRH). This lead to the private key being vulnerable to forgery in subexponential time through Gaussian elimination, which was solved by the inclusion of noise variables in MPPK and HPPK (see section C.1).

References

[1] R. Kuang, M. Perepechaenko, M. Sayed, and D. Lou, "Homomorphic polynomial public key cryptography for quantum-secure digital signature," Cryptology ePrint Archive, Paper 2023/1768, 2023. [Online]. Available: https://eprint.iacr.org/2023/1768

[2] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," Cryptology ePrint Archive, Paper 2011/344, 2011. [Online]. Available: https://eprint.iacr.org/2011/344

[3] R. Kuang, M. Perepechaenko, and M. Barbeau, "A new post-quantum multivariate polynomial public key encapsulation algorithm," *Quantum Information Processing*, vol. 21, no. 11, pp. 1–22, 2022. [Online]. Available: https://doi.org/10.1007/s11128-022-03712-5

[4] R. Kuang, M. Perepechaenko, M. Sayed, and D. Lou, "Homomorphic polynomial public key cryptography for quantum-secure digital signature," *ArXiv*, vol. abs/2311.08967, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265212806

[5] R. Kuang, "A deterministic polynomial public key algorithm over a prime galois field gf(p)," in *2021 2nd Asia Conference on Computers and Communications (ACCC)*, 2021, pp. 79–88.

[6] S. Evdokimov, "Factorization of polynomials over finite fields in subexponential time under grh," in *Algorithmic Number Theory*, L. M. Adleman and M.-D. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 209–219.