Homomorphic Polynomial Public Key for quantum-secure encapsulation

Carsi, Ana
4740908
ana.carsi_gonzalez@stud.uni-heidelberg.de

***Abstract*—In this report a review on the innovative asymmetric cryptographic method designed by Randy Kuang et al. for post-quantum computing 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. The auxiliary project code can be found in this link.**

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 [1] 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, Randy Kuang [3] introduced Multivariate Polynomial Public Key cryptography (MPPK) in 2022 [4], 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 by means of 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 [3], 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 and its application to digital signature (DS). Finally, its corresponding efficiency and trustability 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 ciphertext and plaintext 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 the general Poylnomial Public Key (PPK) (Kuang 2021) and a specific form of homomorphic encryption. It uses multivariate polynomials whose coefficients lie on finite fields. Plus, the encryption method is proven to be partially homomorphic under addition and multiplication.

Although MPPK and HPPK use multivariate polynomials in their construction core, their security does not rely on the general difficulty of solving multivariate polynomial systems [5] but on the NP-completeness problem of the Modular Diophantine Equation [6].

C. HPPK KEM

HPPK was originally designed for Key Encapsulation Mechanism [3], 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.

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 $u_1, ..., u_m$ are randomly chosen noise variables from $\mathbb{F}_p$. It's worth noting that x is not necessarily the secret itself, but rather a variable that will be used to encode the secret.

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.

Note: The bit length of S_1 and S_2 is typically defined as

$$\ell = 2\log_2 p + 8,$$

which is slightly more than twice the bit length of p. This extra padding ("+8") provides additional security.

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 II-C4. Equivalently,

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

Afterwards, we name

$$\begin{aligned} P' &= P(x, u_1, ..., u_m) - P_0 \\ Q' &= Q(x, u_1, ..., u_m) - Q_0 \end{aligned}$$

with P_0 and Q_0 the constant terms.

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

$$\begin{aligned} \overline{p}_{ij} &= R_1 \cdot p_{ij} \pmod{S_1} \\ \overline{q}_{ij} &= R_2 \cdot q_{ij} \pmod{S_2} \end{aligned}$$

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

3) Decapsulation

The decryption of the ciphertext takes place through symmetric homomorphic decryption (reversing the mathematical operation, which has homomorphic properties) followed by the division of polynomials.

1) Symmetric homomorphic decryption:

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

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 $\mathbb{F}_p$.

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 II-C1.

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 II-C2 can be defined with the permutation operator:

$$\hat{E}(R, S) = R \cdot a \pmod S \tag{1}$$

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

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

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

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

Input: Security parameter λ, polynomial degree n, number of polynomials m, prime p
Output: Private key SK, Public key PK
for $i = 0$ **to** λ **do**
  $f_i \leftarrow \text{Random}() \bmod p$; // generate $f_i()$
  $h_i \leftarrow \text{Random}() \bmod p$; // generate $h_i()$
end
for $i = 0$ **to** n **do**
  for $j = 0$ **to** $m - 1$ **do**
    $c_{ij} \leftarrow \text{Random}() \bmod p$; // generate $\beta(i, 1, \ldots, m)$
  end
end
for $i = 0$ **to** $n + \lambda$ **do**
  for $j = 0$ **to** $m - 1$ **do**
    for $s = 0$ **to** $i - 1$ **do**
      $P_{ij} \leftarrow f_s c_{(i-s)j}$;
      $Q_{ij} \leftarrow h_s c_{(i-s)j}$;
    end
  end
end
$\ell \leftarrow 2 \log_2 p + 8$; // Make the hidden field 8 bits larger than doubled prime field
$S_1 \leftarrow \text{Random}(\ell)$; // Generate the hidden ring $\mathbb{Z}/S_1\mathbb{Z}$
$R_1 \leftarrow \text{Random}(\ell) \bmod S_1$;
while $\gcd(R_1, S_1) \neq 1$ **do**
  $R_1 \leftarrow \text{Random}(\ell) \bmod S_1$;
end
$S_2 \leftarrow \text{Random}(\ell)$; // Generate the hidden ring $\mathbb{Z}/S_2\mathbb{Z}$
$R_2 \leftarrow \text{Random}(\ell) \bmod S_2$;
while $\gcd(R_2, S_2) \neq 1$ **do**
  $R_2 \leftarrow \text{Random}(\ell) \bmod S_2$;
end
for $i = 0$ **to** $n + \lambda$ **do**
  for $j = 0$ **to** $m - 1$ **do**
    for $s = 0$ **to** $i - 1$ **do**
      $P_{ij} \leftarrow R_1 \cdot P_{ij} \bmod S_1$;
      $Q_{ij} \leftarrow R_2 \cdot Q_{ij} \bmod S_2$;
    end
  end
end
SK $\leftarrow (f[], h[], R_1, S_1, R_2, S_2)$;
PK $\leftarrow (P[n + \lambda + 1][m], Q[n + \lambda + 1][m])$;
return *SK, PK*

Algorithm 1: Key generation of HPPK KEM

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

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. This means 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 (Section IV) 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 Gaussian elimination and Generalized Riemann Hypothesis attacks. This was not the case of its predecessor Deterministic Polynomial Public Key (DPPK) (see Appendix).

Application of Galois Theory in HPPK

The encryption operator mentioned in 1 has homomorphic properties and operates within the algebraic structure defined by the finite field $\mathbb{F}_p$, where p is prime. In $\mathbb{F}_p$, the only automorphism is the identity mapping, as every element a satisfies $a^p = a$ (Fermat's Little Theorem).

The encryption operator in HPPK is designed to allow for the decryption of encrypted data, ensuring that the original information can be recovered accurately. While it may not be an automorphism in the strict mathematical sense (as the only automorphism in $\mathbb{F}_p$ is the identity), it preserves the algebraic properties that are crucial for the cryptosystem's functionality.

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. For example, 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.

Now, while $\mathbb{F}_p$, where HPPK operates, has a simpler structure compared to extension fields $\mathbb{F}_{p^n}$ and doesn't directly utilize the rich automorphism structure of extension fields, it still benefits from the algebraic properties that contribute to the scheme's security.

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.

IV. HPPK DS

HPPK was then reframed for digital signature (DS) in [7] 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.

Kuang et al. introduced a modified Barrett reduction algorithm that aims to constrain outputs to [0, n) with high probability. They achieve this by increasing the bit length of the Barrett parameter R, introducing a parameter $\delta = k - \lceil \log_2 n \rceil$, where k is R's bit length. This adjustment reduces the likelihood of results falling outside the desired range.

The HPPK variant also incorporates additional processing steps to handle cases where the result exceeds n, though these instances become increasingly rare as δ increases. Key differences in this modified algorithm include the use of a larger R value ($k\lceil \log_2 n \rceil + 32$) to minimize the probability of $z \geq n$, and a final conditional subtraction to ensure $z < n$, which is rarely needed when k is sufficiently large. By providing more consistent results within the range [0, n), this algorithm enhances the overall efficiency and reliability in modular arithmetic operations in HPPK.

Input: x, n, $R = 2^k$, $\mu = \lfloor R^2/n \rfloor$, where $k \geq \lceil \log_2 n \rceil + 32$
Output: $z \equiv x \pmod{n}$
$q_1 \leftarrow \lfloor x/R \rfloor$;
$q_2 \leftarrow \mu \cdot q_1$;
$q_3 \leftarrow \lfloor q_2/R \rfloor$;
$r_1 \leftarrow x \bmod R$;
$r_2 \leftarrow (q_3 \cdot n) \bmod R$;
$z \leftarrow r_1 - r_2$;
if $z < 0$ **then**
  $z \leftarrow z + R$;
end
if $z \geq n$ **then**
  $z \leftarrow z - n$;
end
return z

Algorithm 2: HPPK Variant of Barrett Reduction

Verification Process

The verifier, upon receiving the signature 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

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

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

with $a_{ij} = \lfloor \frac{RP_{ij}}{S_1} \rfloor$ and $b_{ij} = \lfloor \frac{RQ_{ij}}{S_2} \rfloor$ where $R = 2^k$ is the Barrett parameter as shown in the algorithm.
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

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

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

where φ is the Euler's totient function.

The large key space enabled by the modular multiplication mentioned in 2 reinforces the security of the cryptosystem, as it dramatically increases the number of potential keys that an attacker would need to consider. Additionally, HPPK provides the following entropy values for different NIST security levels: Level I: 144 bits, Level III: 208 bits, Level V: 272 bits, which are equivalent to the difficulty of breaking 128, 192 and 256-bit AES [3].

Efficiency

The HPPK DS scheme maintains efficiency with compact outputs. For example, at NIST security level I:

- Public Key (PK) size: 372 bytes
- Private Key (SK) size: 96 bytes
- Signature size: 128 bytes

Moreover, HPPK's performance with other cryptographic schemes, including AES-based systems like Kyber can be analysed from the benchmark released in [7]. One can see

Scheme	Key Generation	Encapsulation	Decapsulation
HPPK KEM	12,665	25,963	63,365
Kyber	72,403	95,466	117,406
McEliece	152,424,455	108,741	45,122,734

TABLE I: Performance comparison of post-quantum schemes at Security Level I (clock cycles)

HPPK outperforms Kyber, designed to be competitive with AES in terms of efficiency. It is however crucial to note tha these benchmarks are for specific security levels and configurations, and performance may vary at different security

levels or with different parameter choices. Furthermore, Kuang and Perepechaenko mention the need of further work on optimizing HPPK and benchmarking its performance.

Private key recovery attack

The recovery 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 [8]. 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 ($|S_1|^2 = |S_2|^2 = 2|p|^2$).

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. Furthermore, the efficiency analysis has proven the need of further efforts on benchmarking.

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 multiple applications.

Appendix

Efficiency of Homomorphic Encryption

The efficiency of fully homomorphic encryption has been the big question following its invention [1]. 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 [2] . 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. This 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 [9] 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 [8] 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 II-C1).

References

[1] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-First Annual ACM Symposium on Theory*

of Computing, ser. STOC '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 169–178. [Online]. Available: https://doi.org/10.1145/1536414.1536440

[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, 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

[4] 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

[5] R. Kuang, M. Perepechaenko, R. Toth, and Y. Kuang, "Performance comparison of quantum-safe multivariate polynomial public key encapsulation algorithm," *EURASIP Journal on Information Security*, vol. 2024, p. 23, 2024.

[6] R. Kuang and M. Perepechaenko, "A novel homomorphic polynomial public key encapsulation algorithm," *F1000Research*, vol. 12, p. 1347, 2023, version 1; peer review: 1 approved, 1 approved with reservations.

[7] R. Kuang, M. Perepechaenko, D. Lou, and B. Tank, "Benchmark performance of homomorphic polynomial public key cryptography for key encapsulation and digital signature schemes," *Cryptology ePrint Archive*, 2024, paper 2024/019. [Online]. Available: https://eprint.iacr.org/2024/019

[8] 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.

[9] 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.