Practical Fully Homomorphic Encryption for Integers over Polynomial Quotient Rings

[1]Alexander Zhirov, [1]Olga Zhirova, [2]Sergey F. Krendelev
[1]*Department of Information Technology Novosibirsk State University Novosibirsk, Russia*
[2]*Lab of Modern Computer Technilogies Novosibirsk State University Novosibirsk, Russia*

***Abstract*—In this article we describe a simplified version of Polly Cracker-style fully homomorphic encryption scheme. The main feature of our scheme is an ability to define a strict upper bound of ciphertext size when performing calculations on it for both addition and multiplication. Combined with homomorphic properties of Polly Cracker it's able to reach high calculation performance without degrading in time. Another important aspect is utilization of large finite rings for calculations in untrusted environment, which prevents most of known attacks on Polly Cracker family.**

***Index Terms*—fully homomorphic cryptography, finite integer rings, polynomials, polly cracker.**

I. Introduction

In our days, development of practically efficient fully homomorphic encryption (FHE) schemes is the one of the most important problems in cryptography. An encryption scheme is called *fully homomorphic* if it's able to evaluate an arbitrary function $f(x_1, \dots, x_n)$ over ciphertexts. In this case decrypted value must match to a calculation result of the same function f over plaintexts.

Wide spread of cloud computing is a serious driving factor for development of such kind of ciphers to protect confidential data in an untrusted environment. There are two general scenarios which pose slightly different requirements for encryption.

The first is protection of information stored in cloud databases. Traditional encryption doesn't fit really well for this case, because it doesn't allow to perform any computations (i.e. statistical) on data directly in the cloud. For this case we need an FHE resistant to *known plaintext attacks* additionally with relatively small ciphertext size. At the same time, we usually don't perform complex calculations on data inside database, so we can lower requirements for computation performance.

Second scenario is outsourcing heavy computations (i.e. physical modeling, video rendering and so on) to a cloud. To make this cost effective we must require high computation performance over encrypted data, but we may lower security requirements for certain scenarios such as *chosen plain-text or ciphertext attack.*

This research was performed in Novosibirsk State University under support of the Ministry of education and science of Russia (contract no. 02.G25.31.0054).

Initial breakthrough in this domain was a research done by Craig Gentry for his PhD [1] in 2009. He proved theoretical possibility of such encryption though his original scheme was absolutely impractical due to heavy requirements on memory and computational resources. His work had a significant impact on cryptographic community and in following years many derived encryption schemes were presented to improve it's performance. Unfortunately, all currently known encryption schemes based on Gentry's ideas are still not efficient enough to be incorporated into a real world applications and databases.

In this paper we consider an encryption scheme, which may be useful in both described scenarios and is based on Polly Cracker ideas, allowing to reach high calculation performance.

II. Prior works and our contribution

In early 1990s several works were presented, which introduced family of public-key cryptosystems currently known as Polly Cracker. Fellows and Koblitz proposed a public-key encryption based on a secret ideal in polynomial ring [2]. In this work a set of multivariate polynomials $f_1, \dots, f_{m-1} \in F[x_1, \dots x_n]$ over a field F represent an ideal $\mathcal{I}$ and are considered as a public key. These polynomials must have a common root x^*, which is a private key. Ciphertext for a message $m \in P/\mathcal{I}$ is computed as follows:

$$c = \sum_i h_i f_i + m.$$

To decrypt is it's enough to evaluate c at the secret root.

About the same time B. Barkee et al. published a paper [3] presenting a cryptosystem based on similar ideas. In their scheme, private key consists of Gröbner basis G defining an ideal $\mathcal{I}$ in polynomial ring. Public key included a set of arbitrary samples $f_1, \dots, f_{m-1} \in \mathcal{I}$.

In the same paper they demonstrated weakness on such kind of encryption to attacks employing Buchberger's algorithm [4] to recover Gröbner basis. They posed a long-standing question, if it's possible to use Gröbner basis for cryptographic purposes.

These works were followed by several attempts to invent a secure Polly Cracker-style cryptosystem, but almost all of them were broken by now. A very good survey of these schemes and their weaknesses is presented in [5].

In 2011 Albrecht et al. published a paper [6] where homomorphic properties of Barkee's scheme [3] and it's IND-BCPA security were discussed. They considered two modifications: noiseless FHE and noisy somewhat homomorphic encryption (SHE), conceptually similar Gentry's SHE. They prove that noiseless scheme reaches only bounded m-IND-CPA security, which means that it might be broken if an adversary is able to encrypt $m(\lambda)$ plaintexts, where λ is a security parameter and m — a polynomial function.

In this paper we present modification of noiseless FHE [6] which resists Gröbner basis recovery attacks. We turn ciphertext domain into polynomial quotient ring, which allows to define strict upper bound of ciphertext size independently of evaluated circuit depth. This makes our cryptosystem practically applicable to secure computations in, for example, cloud databases.

Additionally, we introduce notion of *public* and *private* rings. A private ring is an integer quotient ring by prime modulus. A public ring contains the private as a subring and is an integer quotient ring by composite modulus. Given only with public ring adversary can't apply Buchberger's algorithm to recover a secret key.

The rest of the paper is organized as follows. In section III we establish notation, describe our cryptosystem and provide initialization, encryption, decryption and calculation algorithms. Then, in section IV we consider it from mathematical standpoint and prove it's correctness and homomorphic properties. Next, in section V we provide a brief security review. In section VI we discuss a modification of our scheme with even better level of security at the cost of worse calculation performance. We also discuss implementation-related issues in VII practical aspects of the scheme in section VIII. Then we present benchmarking results in IX. Finally, in section X we discuss main achieved results and possible development directions for future research.

III. Encryption scheme construction

In the rest of the paper we will use following notations:

- $\mathbb{Z}_n$ — integer ring modulo n.
- $\mathbb{Z}_n[x]$ — univariate polynomial ring over $\mathbb{Z}_n$.
- $P_{n,d} \subset \mathbb{Z}_n[x]$ — a subset of $\mathbb{Z}_n[x]$ containing all polynomials of degree less or equal to d.
- $I_{n,d} \subset P_{n,d}$ — a subset of $P_{nm,d}$ containing only irreducible monic polynomials.
- $\mathcal{I}_w$ — an ideal in $\mathbb{Z}_n[x]$ generated by a polynomial $w(x)$.
- $\mathbb{Z}_n[x]/\mathcal{I}_w$ — quotient ring modulo ideal $\mathcal{I}_w$.
- $\mathbf{P}_b$ — a set of prime numbers with a length of binary representation b.
- $x \leftarrow_\$ X$ — sampling a random element x from a set X with uniform distribution.
- $\mathrm{LC}(c(x))$ — coefficient of a highest non-zero term of a polynomial $c(x)$.

As a *server* we will denote an untrusted party, which is given with ciphertexts and public cryptosystem properties to perform encrypted computations over them.

An integer quotient ring which holds plaintext is called *private*. An integer quotient ring which holds coefficients of ciphertext polynomials is called *public*.

A. The encryption concept overview

Hereinafter we assume that λ is a general cryptosystem security parameter and all other parameters implicitly depend on it. We fix two of them: $d = d(\lambda) \in \mathbb{Z}$ defines secret polynomial degree and $b = b(\lambda) \in \mathbb{Z}$ specifies private ring modulus binary length. Let's establish a secret key:

$$u(x) \leftarrow_\$ P_{n,d}$$
$$n \leftarrow_\$ \mathbf{P}_b.$$
$$m \leftarrow_\$ \mathbf{P}_b.$$

To encrypt an integer number $\alpha \in \mathbb{Z}_n$, where $n \in \mathbf{P}_b$, we need to generate random $s_1(x) \in P_{n,d}$, $s_2(x) \in P_{m,d}$, $\beta \in \mathbb{Z}_m$. Then we compute two ciphertexts:

$$c_1(x) = (s_1(x)u(x) + \alpha) \in \mathbb{Z}_n[x]$$
$$c_2(x) = (s_2(x)u(x) + \beta) \in \mathbb{Z}_m[x].$$

$c_1(x)$ is a "real" ciphertext and $c_2(x)$ a "fake". By Chinese Remainder Theorem we can construct a polynomial $c(x) \in \mathbb{Z}_{nm}[x]$ such that $c(x) = c_1(x) \mod n = c_2(x) \mod m$. This also implies two following statements:

$$(c(x) - \alpha) \not\equiv u(x)s'(x) \mod nm \quad (1)$$
$$[c(x) \mod u(x)] \not\equiv \alpha \mod nm \quad (2)$$

Being mathematically trivial, (1) illustrates, that secret polynomial $u(x)$ can't be recovered by finding GCD of two ciphertexts of zero, and (2) implies, that knowing $u(x)$ doesn't decrypt original plaintext over $\mathbb{Z}_{nm}$. More detailed security analysis will be provided in section V.

In practice we don't care about particular value of β, so we can simplify encryption procedure and build $c(x)$ by following formula:

$$c(x) = s(x)u(x) + n \cdot r(x) + \alpha.$$

When $r(x) \not\equiv 0$ statements (1) and (2) will remain true for $c(x)$.

B. Cryptosystem initialization

Before initializing cryptosystem we must specify parameters $d = \mathrm{d}(\lambda)$ and $b = \mathrm{b}(\lambda)$. d defines degree of secret polynomial and b is number of bits in binary representation of numbers n and m. The larger values are chosen, the higher level of security is reached.

We select two prime numbers n, m of length b, which define private ring $\mathbb{Z}_n$ and public ring $\mathbb{Z}_{nm}$. Since we need to hide private ring from an adversary we pass to a server product nm only. Thus n becomes a part of secret key.

We sample a random monic irreducible polynomial $u(x) \in I_{n,d}$ of degree d, which becomes a second part of secret key. We also construct $w(x) \in \mathbb{Z}_{nm}[x]$ to be

$$w(x) = u(x) \cdot v(x) \mod n,$$

Input: b — size of prime modulus n in bits, d — degree of secret polynomial $u(x)$.
Output: $\langle P_{pub}, P_{key} \rangle$ — initialized cryptosystem.

```
function CS_INIT($b, d$)
    $n \leftarrow_\$ \mathbf{P}_b$
    $m \leftarrow_\$ \mathbf{P}_b$
    $u(x) \leftarrow_\$ I_{n,d}$
    $v(x) \leftarrow_\$ I_{n,d+1}$
    $w(x) \leftarrow_\$ P_{nm,2d}$
    $w(x) \leftarrow u(x) \cdot v(x) + w(x) \cdot n$
    $P_{pub} \leftarrow \langle n \cdot m, w(x) \rangle$
    $P_{key} \leftarrow \langle n, u(x), d \rangle$
    return $\langle P_{pub}, P_{key} \rangle$
end function
```

Figure 1. Cryptosystem initialization algorithm

having $v(x) \in I_{n,d+1}$.

At this point we can define cryptosystem public properties P_{pub} and a secret key P_{key}:

$$P_{pub} = \langle nm, w(x) \rangle$$
$$P_{key} = \langle n, u(x) \rangle$$

Algorithm 1 provides formalized description of cryptosystem initialization process.

C. Encryption and decryption

Let $\alpha \in \mathbb{Z}_n$ be a plaintext. Then we map it to a ciphertext polynomial $c(x) \in \mathbb{Z}_{nm}[x]$ computed by following formula:

$$c(x) = s(x) \cdot u(x) + n \cdot r(x) + \alpha \mod nm. \quad (3)$$

Here $s(x)$ is randomly sampled from $I_{nm,d}$ and $r(x) \in \mathbb{Z}_{nm}[x]$ is a random non-zero polynomial of degree $2d-1$. It's important to use different $s(x)$ for each encryption to obtain properties of probabilistic encryption. This is also critical for encryption security level.

Decryption is as easy as computing remainder of division of $c(x)$ by n and then $u(x)$:

$$\alpha = [(c(x) \mod n) \mod u(x)]$$

Since $u(x)$ is monic and 1 is reversible in any ring $\mathbb{Z}_n$, α is determined unambiguously.

Algorithms 2 and 3 provide formal description for this procedures.

D. Calculation over ciphertexts

The encryption scheme supports homomorphic addition and multiplication over ciphertexts.

Polynomials produced by the algorithm 2 may be considered as elements of quotient ring $\mathbb{Z}_{nm}[x]/\mathcal{I}_w$. Thus addition and multiplication of ciphertexts must be performed according to rules of multiplication and addition of polynomials in such ring. Due to nature of the quotient ring degree of minimal class member is below $\deg(w(x))$. This implies upper limit of ciphertext size.

Input: $\langle P_{pub}, P_{key} \rangle$ — initialized cryptosystem, α — plaintext.
Output: $c(x)$ — ciphertext.

```
function CS_ENCRYPT($P_{pub}, P_{key}, \alpha$)
    $s(x) \leftarrow_\$ I_{n,d}$
    $r(x) \leftarrow_\$ P_{nm,d}$
    $c(x) \leftarrow s(x) \cdot u(x) + n \cdot r(x) + \alpha \mod nm$
    return $c(x)$
end function
```

Figure 2. Encryption algorithm

Input: $\langle P_{pub}, P_{key} \rangle$ — initialized cryptosystem, $c(x)$ — ciphertext.
Output: α — plaintext.

```
function CS_DECRYPT($P_{pub}, P_{key}, c(x)$)
    $c'(x) \leftarrow c(x) \mod n$
    $p(x) \leftarrow c'(x) \mod u(x)$
    if $\deg(p(x)) \neq 0$ then
        return Error: invalid ciphertext.
    end if
    $\alpha \leftarrow p(x) \mod x$.
    return $\alpha$.
end function
```

Figure 3. Decryption algorithm

Informally this may be treated as follows. Polynomials of a form $ax^l \cdot w(x)$ are ciphertexts of zero in terms of our encryption. Assume that we've got ciphertext $c(x)$ as a result of some calculations. Let $d' = \deg(c(x))$ be greater than $\deg(w(x)) = 2d + 1$. Due to homomorphic properties of our cryptosystem ciphertext

$$c'(x) = c(x) - \mathrm{LC}(c(x))x^{d'-(2d+1)}w(x)$$

is an encryption of the same number as $c(x)$. By construction, $\deg(c'(x)) < \deg(c(x))$. Repeating this operation we can reduce degree of $c(x)$ at least to $2d$ while preserving decryption correctness. Described procedure is essentially equivalent to finding remainder of division of $c(x)$ by $w(x)$, thus it's equivalent to operations over the ring $\mathbb{Z}_{nm}[x]/\mathcal{I}_w$.

Formal correctness of this view and homomorphic encryption properties are proved in section IV.

IV. Encryption correctness

Correctness of FHE scheme must be proved in two meanings. First, it must be able to encrypt any plaintext with any valid secret key and then correctly decrypt it. Second, for each claimed homomorphic operation $\circ$ and ciphertexts $c_1(x), c_2(x)$ for numbers α, β respectively decryption of $c_1(x) \circ c_2(x)$ must match to $\alpha \circ \beta$.

Lemma 1. *Given polynomials $u(x) \in I_{n,d}$ and $n \cdot r(x) \in P_{nm,2d}$, following statements are true:*

1) *there exist polynomials $k(x) \in P_{nm,d}$, $l(x) \in P_{nm,d-1}$ such that $n \cdot r(x) = k(x)u(x) + nl(x)$;*
2) *$k(x), l(x)$ are unique.*

Proof: Given an arbitrary $r(x)$ and monic $u(x)$, Euclidean division algorithm unique quotient and remainder:

$$\begin{aligned} r(x) &= k'(x)u(x) + l(x) \\ \deg[l'(x)] &< \deg[u(x)] \\ \Rightarrow n \cdot r(x) &= [n \cdot k'(x)]u(x) + nl(x) = \\ &= k(x)u(x) + nl(x) \end{aligned}$$

Thus, $k(x)$ and $l(x)$ exist and unique for any given $u(x)$ and $n \cdot r(x)$. ■

Theorem 1. *Let $d > 0$ be an integer, $u(x) \in I_{n,d}$, $v(x) \in I_{n,d+1}$, $w'(x) \in P_{nm,2d}$, $w(x) = u(x)v(x) + n \cdot w'(x)$.*
Consider

$$R = \{c(x) \in \mathbb{Z}_{nm}[x]/\mathcal{I}_w \mid c(x) = s(x)u(x) + n \cdot r(x) + \alpha\},$$

where:
1) *$s(x) \in P_{nm,d}$ is an arbitrary monic polynomial,*
2) *$r(x) \in P_{nm,2d}$,*
3) *$\alpha \in \mathbb{Z}_{nm}$.*

Then R defines a subring in $\mathbb{Z}_{nm}[x]/\mathcal{I}_w$.

Proof: Let

$$R' = \{c(x) \in \mathbb{Z}_{nm}[x] \mid c(x) = s(x)u(x) + n \cdot r(x) + \alpha\}$$

be a subset of $\mathbb{Z}_{nm}[x]$ body, where:
1) $s(x) \in \mathbb{Z}_{nm}[x]$ is an arbitrary monic polynomial,
2) $r(x) \in P_{nm,2d}$,
3) $\alpha \in \mathbb{Z}_{nm}$.

Let's prove that R' defines subring in $\mathbb{Z}_{nm}[x]$. Obviously, R' contains both 0 and 1:

$$1 = 0 \cdot u(x) + n \cdot 0 + 1 \in R' \quad (4)$$
$$1 = 0 \cdot u(x) + n \cdot 0 + 0 \in R' \quad (5)$$

Consider two samples $c_\alpha(x), c_\beta(x) \in R'$:

$$\begin{aligned} c_\alpha(x) &= s_\alpha(x)u(x) + nr_\alpha(x) + \alpha \\ c_\beta(x) &= s_\beta(x)u(x) + nr_\beta(x) + \beta \end{aligned}$$

R' is closed under addition:

$$\begin{aligned} c_{\alpha+\beta}(x) &= c_\alpha(x) + c_\beta(x) = \\ &= [s_\alpha(x) + s_\beta(x)] \cdot u(x) + n[r_\alpha(x) + r_\beta(x)] + \\ &\quad + [\alpha + \beta] \in R' \quad (6) \end{aligned}$$

Note that given $c_i(x) \in R'$ might have several representation that satisfy conditions on R'. Though there always exists unique representation of $c_i(x)$ having $\deg[r_i(x)] < \deg[u(x)]$ due to lemma 1. We will refer it as *normalized form*.

Now consider product

$$\begin{aligned} c_{\alpha\beta}(x) &= c_\alpha(x) \cdot c_\beta(x) = \\ &= [s_\alpha(x)u(x) + nr_\alpha(x) + \alpha] \cdot [s_\beta(x)u(x) + nr_\beta(x) + \beta] = \\ &= [s_\alpha(x)s_\beta(x)u(x) + ns_\alpha(x)r_\beta(x) + ns_\beta(x)r_\alpha(x) + \\ &\quad + s_\alpha(x)\beta + s_\beta(x)\alpha] \cdot u(x) + n[r_\alpha(x)\beta + r_\beta(x)\alpha] + \alpha\beta + \\ &\quad + n^2 r_\alpha(x)r_\beta(x) = \\ &= s'_{\alpha\beta}(x)u(x) + nr'_{\alpha\beta}(x)\alpha\beta + \underbrace{n^2 r_\alpha(x)r_\beta(x)}_{\overset{\text{lem. 1}}{=} s''_{\alpha\beta}(x)u(x) + n \cdot r''_{\alpha\beta}(x)} = \\ &= [s'_{\alpha\beta}(x) + s''_{\alpha\beta}(x)]u(x) + n[r'_{\alpha\beta}(x) + r''_{\alpha\beta}(x)] + \alpha\beta \quad (7) \end{aligned}$$

By lemma 1, (7) satisfies requirements for being normalized form of $c_{\alpha\cdot\beta}(x)$, thus $c_{\alpha\cdot\beta}(x) \in R'$. Statements (4), (5), (6), (7) show that R' passes subring test, thus R' is subring in $\mathbb{Z}_{nm}[x]$.

Now consider natural ring homomorphism $\omega : \mathbb{Z}_{nm}[x] \to \mathbb{Z}_{nm}[x]/\mathcal{I}_w$. Since $R = \omega(R')$ and R' is a subring in $\mathbb{Z}_{nm}[x]$, thus R is a subring in $\mathbb{Z}_{nm}[x]/\mathcal{I}_w$ by the First isomorphism theorem.

■

Theorem 2. *Using the notion of theorem 1, following statements are true:*
1) *Mapping $\mathcal{D} : R \to \mathbb{Z}_n$*

$$\mathcal{D}(c(x)) = [(c(x) \mod u(x)) \mod n]$$

is a ring epimorphism (surjective homomorphism).

2) *there exists injective mapping $\mathcal{E} : \mathbb{Z}_n \to R$ such that for any $\alpha \in \mathbb{Z}_n$ following equation is correct:*

$$\alpha = \mathcal{D}(\mathcal{E}(\alpha))$$

Proof: $\mathcal{D}$ can be represented as a composition of two mappings defined by natural homomorphisms:

$$\begin{aligned} \phi &: R \to R/\mathcal{I}_u \\ \psi &: R/\mathcal{I}_u \to R/\langle n \rangle. \end{aligned}$$

Here as $R/\langle n \rangle$ we denote a quotient ring generated by ideal $\langle n \rangle$, where n is considered as polynomial of zero degree. By definition, $R/\langle n \rangle \subseteq \mathbb{Z}_n$. As composition of two homomorphisms is a homomorphism and $\mathcal{D} \equiv \psi \circ \phi$, thus $\mathcal{D}$ is homomorphism from R to $\mathbb{Z}_n$.

Let's prove that $\mathcal{D}$ is an epimorphism. For any $\alpha \in \mathbb{Z}_n$ there exists $c(x) \in R$ which normalized form is equals to:

$$c(x) = s(x)u(x) + nr(x) + \alpha.$$

By definition of $\mathcal{D}$, $\mathcal{D}(c(x)) = \alpha$. Thus $\mathcal{D}$ is an epimorphism.

By construction, any mapping $\mathcal{E}(\alpha) = s(x)u(x) + nr(x) + \alpha$ for given $u(x)$ and n and arbitrary $s(x)$ and $r(x)$ satisfies requirement $\alpha = \mathcal{D}(\mathcal{E}(\alpha))$ and any $\alpha \in \mathbb{Z}_n$. ■

From this theorems two corollaries follow, which directly prove encryption correctness.

Corollary 1 (On correctness of encryption and decryption)**.** *Let $\mathcal{E}_{P_{key}}$ and $\mathcal{D}_{P_{key}}$ be the mappings defined by algorithms 2 and 3 respectively. Then following statements are true:*

1) *For any pair of plaintext and secret key $\langle \alpha, P_{key} \rangle$ there exists an appropriate ciphertext $c(x)$:*

$$c(x) = \mathcal{E}_{P_{key}}(\alpha)$$

2) *For any pair of ciphertext and secret key $\langle c(x), P_{key} \rangle$ there exists one and only one matching plaintext $\alpha = \mathcal{D}_{P_{key}}(c(x))$ such that*

$$c(x) = \mathcal{E}_{P_{key}}(\alpha)$$

Proof: By construction $\mathcal{E}_{P_{key}} \equiv \mathcal{E}$ and $\mathcal{D}_{P_{key}} \equiv \mathcal{D}$, where mappings $\mathcal{E}$ and $\mathcal{D}$ are from theorem 2. Considering this, the corollary immediately follows from theorem 2. ■

Corollary 2 (On correctness of computations over ciphertexts)**.** *Assume that addition and multiplication of ciphertexts are performed according to rules of polynomial addition and multiplication in a ring $\mathbb{Z}_{nm}[x]/\mathcal{I}_w$. Then for any cryptosystem defined by $\langle P_{pub}, P_{key} \rangle$ and any plaintexts $\alpha, \beta \in \mathbb{Z}_n$ following statements are correct:*

$$\alpha + \beta = \mathcal{D}_{P_{key}}[\mathcal{E}_{P_{key}}(\alpha) + \mathcal{E}_{P_{key}}(\beta)] \quad (8)$$

$$\alpha \cdot \beta = \mathcal{D}_{P_{key}}[\mathcal{E}_{P_{key}}(\alpha) \cdot \mathcal{E}_{P_{key}}(\beta)] \quad (9)$$

Proof: By construction $\mathcal{E}_{P_{key}} \equiv \mathcal{E}$ and $\mathcal{D}_{P_{key}} \equiv \mathcal{D}$, where mappings $\mathcal{E}$ and $\mathcal{D}$ are from theorem 2. Theorem 1 guarantees that decryption algorithm will be able to handle correctly ciphertexts after addition or multiplication. In turn, homomorphic properties proven in theorem 2 imply correctness of statements (8) and (9). ■

Based on corollaries 1 and 2 we can state that provided algorithms might be used for encryption, yet it's security still must be evaluated. We do it in the next section.

V. ENCRYPTION SECURITY EVALUATION

A. Preliminaries

In this section we will consider the cryptosystem security against most typical attacks. As it was mentioned in introduction this encryption might be considered as a member of Polly Cracker family, in particular as a modification of noiseless encryption described in [6]. Since Polly Cracker-style schemes are believed to be generally insecure, we must also evaluate resistance of our encryption against most typical attacks proposed for this cryptosystem family.

Hereinafter we will assume that during cryptosystem construction prime numbers n and m were chosen large enough to make factoring a product of them unacceptable for an adversary due to high costs. By the time of writing it is sufficient to have $\log_2(n) = \log_2(m) = 1024$ to make factoring merely impossible.

B. Ciphertext-only attack

First of all, we must consider a case when an adversary has access only to a set of ciphertexts $c_1(x), \ldots, c_k(x)$ and cryptosystem public parameters $P_{pub} = \langle N, w(x) \rangle$. The most straightforward attack involves solving system of non-linear equations over $\mathbb{Z}_{nm}$ for the unknown $u(x)$:

$$\begin{aligned} c_1(x) &= s_1(x) \cdot u(x) + nr_1(x) + \alpha_1 \\ &\ldots \\ c_k(x) &= s_k(x) \cdot u(x) + nr_k(x) + \alpha_k \\ w(x) &= v(x) \cdot u(x) + nr'(x) \\ N &= n \cdot m \end{aligned}$$

where $u(x), s_i(x), r_i(x), r'(x), \alpha_i, n, m$ are unknown $\forall i = 1, \ldots, k$.

Typically such equation systems are solved in three steps: ring $\mathbb{Z}_{nm}$ is decomposed into a direct product of prime fields $\mathbb{Z}_n \times \mathbb{Z}_m$, equations are solved over each of prime fields and by the Chinese Remainder Theorem solution over $\mathbb{Z}_{nm}$ is reconstructed. However, this method won't work for our case, because we assume factoring nm to decompose ring $\mathbb{Z}_{nm}$ to be a hard problem. On the other hand, solving polynomial equations of an arbitrary degree right over the ring is as hard as integer factorization, as proved for Rabin's public key cryptosystem [7].

Another approach is brute-force attack on $u(x)$. Given with a polynomial $u'(x)$ attacker can perform following test:

$$r_0^i + r_1^i x + \ldots + r_{d-1}^i x^{d-1} = c_i(x) \mod u'(x)$$

$$\forall i = 1, \ldots, k : \; GCD(r_1^i, \ldots, r_{d-1}^i) \overset{?}{\neq} 1 \quad (10)$$

If inequality (10) is satisfied, then given $u'(x)$ is either secret $u(x)$ itself or random multiplier $s(x)$ used for encrypting this particular ciphertext. Probability of false positive is not higher than probability of using the same random $s(x)$ for encrypting all given ciphertext, which is negligible for any $k > 1$.

Nevertheless, complexity of this kind of attack is estimated as $O(2^{\lceil \log_2(n) \rceil \cdot d}/d)$, where $d = \deg[u(x)]$, because about a fraction of $1/d$ of polynomials of degree d is irreducible over finite field [8, p. 142]. It means, that brute-force complexity is harder, than factoring nm, which is assumed to be infeasible.

It seems to be more practical to investigate polynomial $w(x)$ which is part of P_{pub}. As it was mentioned it is effectively a ciphertext of zero, so known plaintext attack might be performed. This scenario is considered in section V-C.

C. Known plaintext attack

In this case an adversary has access to a set of pairs of ciphertexts and corresponding plaintexts: $\langle \alpha_1, c'_1(x) \rangle, \ldots, \langle \alpha_l, c'_l(x) \rangle$. We note that due to homomorphic encryption properties he might construct ciphertext for any desired plaintext. This essentially implies that, for any homomorphic encryption which allows addition operation, CPA and KPA attacks are equivalent and attacker is limited only in number of samples he has access to.

Consider ciphertext structure (3). An attacker is interested in revealing values of $u(x)$ and n. Thus an encrypted plaintext α acts like an unwanted noise for him, which he can easily remove in KPA scenario:

$$c_i(x) = c'_i(x) - \alpha_i = s_i(x)u(x) + nr_i(x), \forall i = 1, \dots, l$$
$$\mathcal{D}_{P_{key}}(c_i(x)) = 0$$

Nevertheless $nr(x)$ term is still present in $c_i(x)$, making it impossible to use GCD algorithm to reveal $u(x)$. In turn, without being able to remove term $s_i(x)u(x)$ from $c_i(x)$, an adversary can't extract value of n by applying GCD to coefficients of $nr(x)$.

Let's consider another hypothetical attack targeted on finding roots of $u(x)$ over $\mathbb{Z}_{nm}$. Assume that adversary tries different elements of $\mathbb{Z}_{nm}$ according to some strategy. For each such element x^* he performs following test:

$$\text{GCD}(c_1(x^*), \dots, c_l(x^*), nm) \neq 1 \tag{11}$$

If there exists such $x^* \in \mathbb{Z}_{nm}$, which passes test (11), attacker will also recover $n = \text{GCD}(c_1(x^*), nm)$. This information is enough to decrypt any ciphertext $c(x)$:

$$\alpha = c(x^*) \mod n$$

This illustrates that requirement for $u(x)$ to be irreducible over $\mathbb{Z}_n$ is substantial, because it makes impossible such kind of attack.

D. Typical attacks on Polly Cracker cryptosystems

For the sake of completeness we must consider proposed cryptosystem as member of Polly Cracker system and verify that it's secure against already known attacks. In [5], there is a good review of those. They may be classified into several groups: attacks based on Buchberger's algorithm [4] or it's modifications, sparse polynomial based attacks and other algebraic attacks based on solving equation systems.

The first group uses Buchberger's algorithm to reconstruct Gröbner basis of a secret ideal, for example, the Fantomas [3] and 2-nomial attacks [9]. These attacks could be barely applied to our cryptosystem, because Gröbner basis theory for finite integer rings isn't currently developed and there is no suitable modification of Buchberger's algorithm. Typical for this purpose method of ring decomposition in conjunction with Chinese Remainder Theorem couldn't be applied here due to hardness of factoring public ring modulus.

The second group exploits sparseness of polynomials used for encryption which is obviously not our case. We use univariate polynomials and can afford using dense ones. Examples for these groups are intelligent linear algebra [10, Ch. 5, §6], differential [11] and zero-evaluation attacks [5, p. 7-8].

Finally, most of other attacks reduce to solving various equation systems over a finite ring, which is $\mathbb{Z}_{nm}$ in our case. As it was already mentioned, the problem of root finding over a finite integer ring is computationally hard in case of composite modulus. This allows us to state that our cryptosystem is secure against typical Polly Cracker attacks.

VI. Growing ciphertext variant

As it was shown in previous section, security of our cryptosystem mostly relies on hardness of factoring public ring modulus nm. It turns out, that we can trade some additional level of security for calculations performance. The idea behind this is using $\mathbb{Z}$ as public ring instead of $\mathbb{Z}_{nm}$. Thus we transfer only $P'_{pub} = \langle w(x) \rangle$ to a server, keeping nm in secret and all encrypted calculations are performed over $\mathbb{Z}[x]/\mathcal{I}_w$.

Corollary 3. *Statements from corollaries 1 and 2 are also correct for cryptosystem defined by pair of $\langle P_{key}, P'_{pub} \rangle$.*

Proof: This directly follows from existence of natural homomorphism from $\mathbb{Z}$ to $\mathbb{Z}_n$. ■

From security point of view, hiding product of nm gives us two advantages: any possible attacks based on factoring nm become impossible, since an attacker doesn't know the product; when brute-forcing $u(x)$ an adversary has no strict upper bound for key coefficients, making search harder. Thus, brute-force attack described in section V-B is still theoretically possible, though it becomes even more sophisticated.

The only drawback of required scheme is ciphertext growth during encrypted calculations. Let $p = \lceil log_2(n) \rceil$ be a number of bits, occupied by single coefficient and $d = \deg[c(x)]$. Denote $\text{size}[c(x)]$ a number of bit occupied by $c(x)$. Hereinafter in this section, when adding or multiplying polynomials, we always mean operations performed modulo $w(x)$.

When adding two polynomials, each coefficient might grow by 1 bit, resulting up to d bits growth for a whole polynomial.

$$\text{size}[c(x) + c(x)] \leq \text{size}[c(x)] + d. \tag{12}$$

Similar, after multiplication each new coefficient is a sum of up to d products of original coefficients. Product of two numbers might take up to twice more space then original numbers, resulting following estimation:

$$\text{size}[c(x) \cdot c(x)] \leq \text{size}[c(x)] \cdot 2 + d^2 \tag{13}$$

From (12), (13) we can conclude, that despite of better security this variant is recommended for use only if intended computations are short enough to afford ciphertext growth.

VII. Implementation-related issues

Any cryptosystem intended for practical usage requires not only theoretical proof of security, but also careful implementation. In our work we actively use many non-trivial entities, such as irreducible polynomials and large prime numbers. Thus, implementation brings up some additional mathematical questions, which are critical for actual level of security provided by it. In our case two major issues are generation of random prime numbers and random irreducible polynomials. Additionally, there is a class of software engineering problems, such as efficient memory management and polynomial arithmetics, which are quite important for good performance of encrypted calculations. We admit importance of this questions, though they are out of this paper's scope.

A. Random prime numbers

Our encryption definitely falls into a category of cryptosystems, which rely on integer factorization hardness. That's why we require high-quality random prime numbers for key generation. Current industry-wide standard is usage of pseudoprimes, which are prime with high probability. Though throughout this paper we always consider numbers n and m to be prime, this fact isn't critical for correctness of encryption and calculations over ciphertexts, which means it's safe to use pseudoprimes in actual encryption implementation.

Another implementation issue is type of prime number to use. Though there is a debate [12] about strong primes advantages, we prefer to use them instead of arbitrary large pseudoprimes, since there are fast enough approaches [13] to generate them.

Finally, we note that recent research [14] shows that there are efficient algorithms, that are able to factorize product of two primes with small difference. For example, [15, sec. 4.1.2 and C.3] requires at least 100 bit difference between them. Though this analysis was performed for RSA cryptosystem, it also applies to our work.

B. Random irreducible polynomials

Since we need to construct random irreducible polynomials, most of existing deterministic algorithms aren't suitable for us. Thus we consider taking random polynomials and testing them for irreducibility. There is estimation [8, p. 142] of number of irreducible polynomials of given degree d over finite field $\mathbb{Z}_n$:

$$\frac{n^d}{d} - \frac{n(n^{d/2}-1)}{(n-1)d} \leq |I_{n,d}| \leq \frac{n^d - n}{d}$$

which means that a fraction $1/d$ of polynomials of degree d is irreducible.

In our encryption, we need polynomials irreducible over $\mathbb{Z}_n$. Using this estimation we can find an optimal trade-off between encryption security and space efficiency, depending of value of d. Assume that n is 1024-bit number. Then we'd have $|I_{n,d}| \approx 2^{1023 \cdot d}/d$ irreducible polynomials.

This means even $d = 3$ is enough to make it merely impossible to generate same polynomial twice. I. e. if one would encrypt a petabyte of 64-bit integers, chance that one irreducible polynomial would have been used twice is be about 2^{-3020}. At the same time, size of ciphertext for a single integer would be 1024 bytes, 128 times larger than plain text.

Finally, it's critical for encryption performance to have efficient irreducibility test for our approach. According to [16] Ben-Or test is the most effective one, especially for low-degree polynomials like we are proposing.

VIII. Practical aspects of the scheme

Since we claim our scheme to be practical, we must consider more precisely how it could be applied to a real-world computations. We have already mentioned, that we must use private ring with modulo being about 1024-bit number to reach required level of security. The only practical difficulty is mismatch between private ring $\mathbb{Z}_n$ and integer rings which computer operates, typically $\mathbb{Z}_{2^{32}}$ or $\mathbb{Z}_{2^{64}}$, which we denote as $\mathbb{Z}_{comp}$. Indeed, there is no homomorphism between $\mathbb{Z}_n$ and $\mathbb{Z}_{comp}$, thus we must take care of possible calculation result mismatches.

Fortunately, $\mathbb{Z}_n$ is much larger than $\mathbb{Z}_{comp}$. Therefore, decryption correctness issues would occur only after integer overflow in $\mathbb{Z}_n$. In practice most of calculations are organized to prevent integer overflow in $\mathbb{Z}_{comp}$. For such cases our cryptosystem is able to evaluate formulas of an arbitrary length.

If a specific type of calculations relies on integer overflows, we can construct a cryptosystem to tolerate at least l overflows for any predefined $l > 0$. Denote $p = \lfloor \log_2(n) \rfloor$ — guaranteed number of significant bits, which may be taken by a number from private ring before overflow might occur, $q = \lfloor \log_2(|\mathbb{Z}_{comp}|) \rfloor$ — number of bits in native computer data type we need to encrypt.

Consider operations on two integers a_1, a_2 of k_1, k_2 significant bits respectively. Following statements describe growth of bit number taken by operation result:

$$\begin{aligned} \max(k_1, k_2) + 1 &\geq \log_2(a_1 + a_2) \\ k_1 + k_2 &\geq \log_2(a_1 \cdot a_2). \end{aligned}$$

In a worst case, result size increases by one bit after each addition and doubles after multiplication. Therefore, to tolerate at least l_{add} addition and l_{mul} multiplication overflows we must choose

$$\begin{aligned} p \geq l_{\text{add}} + q &\Rightarrow n \geq 2^{l_{\text{add}}+q} \\ p \geq q \cdot 2^{l_{\text{mul}}} &\Rightarrow n \geq 2^{q \cdot 2^{l_{\text{mul}}}}. \end{aligned}$$

For example, in case of 32-bit integers and 1024-bit private ring we can tolerate at least 91 addition and 4 multiplication sequential overflows. We expect that this is quite enough for most kinds of calculations, especially in database.

There is one important observation from this example. It's reasonable to choose private ring modulus n of $2^x + 1$ significant meaningful binary digits. This would permit one additional multiplication overflow at cost of only 1 bit increase of private ring. Therefore, if implementation allows us to store ciphertext polynomial coefficients aligned to a single bit, we could benefit from using 1025-bit private ring and 2050-bit public ring instead of 1024 and 2048-bit ones.

IX. Performance

To test actual performance of our encryption scheme we've implemented a small C++ library, on top of mathematical primitives from NTL with GMP backend.

We have tested performance with following parameters: $\deg[u(x)] = 1, 3, 5, 10$ and size of prime numbers n, m equals 512 and 1024 bits, 8 different configurations in total.

We had following reasoning behind selected values of $\deg[u(x)]$: 1 as the least theoretically possible value, thus it must have the highest performance among others; 3 is optimal value from security standpoint, discussed in section VII-B; 5 and 10 were selected to evaluate overall performance loss

Table I
ENCRYPTED ADDITION OPERATION PERFORMANCE

$\lceil\log_2(n)\rceil$	$\deg[u(x)]$	Additions/sec	Overhead
512	1	1138317	1100
	3	472131	2600
	5	253055	4900
	10	115210	10700
1024	1	942608	1300
	3	295707	4200
	5	131879	9300
	10	68498	18000

Table II
ENCRYPTED MULTIPLICATION OPERATION PERFORMANCE

$\lceil\log_2(n)\rceil$	$\deg[u(x)]$	Multiplications/sec	Overhead
512	1	74074	8800
	3	16844	38600
	5	7367	88300
	10	2360	275600
1024	1	29423	22100
	3	5811	111900
	5	2504	259700
	10	790	823300

Table III
ENCRYPTION AND DECRYPTION SPEED

$\lceil\log_2(n)\rceil$	$\deg[u(x)]$	Encryptions/sec	Decryptions/sec
512	1	95080	551298
	3	13037	188971
	5	4860	99548
	10	1370	35886
1024	1	69334	290573
	3	6376	83119
	5	2240	40399
	10	581	13532

trend. As of bit sizes, 512 and 1024 are the most interesting values, since currently factoring 1024-bit number is recognized to be infeasible for most of organizations in the world and 2048-bit is considered as "future-proof" for several upcoming decades.

All presented results were obtained on consumer-grade laptop with Intel Core i5 M460 2.53 GHz CPU, under Ubuntu Linux 13.10, using NTL 6.0.0 and GMP 5.1.2 compiled by GCC 4.8.1.

Encrypted calculation performance was measured for point-wise addition and multiplication of two random vectors of length 400. As a reference, the same test was performed for vectors of 64-bit long integers. Tables I and II hold results of these tests. "Overhead" column shows slowdown factor comparing to unencrypted calculations. Addition demonstrates expected performance hit of about 10^3–10^4 times, caused by more expensive long arithmetic involved and significant size of operands (128 times larger than regular 64-bit integer). Multiplication suffers from slowdown by 10^4–10^5 times because more computationally expensive polynomial multiplication algorithm and (for our implementation) frequent memory re-allocations inside NTL. Thus, we'd expect significantly better results in case of implementation, which doesn't actively use dynamic memory.

Finally, in table III we see expected encryption performance losses for higher degrees of $u(x)$ caused higher complexity of finding irreducible polynomials. Decryption performs better both because it's computationally simpler and doesn't require searching irreducible polynomials.

X. DISCUSSION AND FUTURE WORK

In this paper we have proposed a cryptosystem which may be considered as a candidate for a practical FHE scheme applicable for real-world usage. Certainly our modifications, called to resist known attacks require more thorough analysis for actual level of security reached.

We highlight the method of restricting ciphertext growth as the main practical result of this work. The idea of exploiting ciphertext structure and a valid encryption of zero to reduce ciphertext length in cryptosystem homomorphic to subtraction might be applied to other encryption schemes. This is the main field of future research, which should bring us closer to practical homomorphic encryption.

Our cryptosystem has also some useful properties, which might become important in practical usage. For example, variable substitution might be exploited as on-the-fly key switching mechanism without re-encryption. If we consider ciphertext $c(x) = s(x)u(x) + nr(x) + \alpha$ and variable substitution $x = f(y)$, then $c(f(y))$ can be decrypted with key $P_{key} = \langle n, u(f(y))\rangle$, though the old key $\langle n, u(x)\rangle$ won't fit anymore. In practice this may become a real obstacle for an adversary. By the time he will recover the old key it won't be useful.

REFERENCES

[1] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009. [Online]. Available: http://crypto.stanford.edu/craig/

[2] M. Fellows and N. Koblitz, "Combinatorial cryptosystems galore!" *Contemporary Mathematics*, vol. 168, p. 51, 1994.

[3] B. Barkee, D. D. C. Can, J. Ecks, T. Moriarty, and R. F. Ree, "Why You Cannot Even Hope to use Gröbner Bases in Public Key Cryptography: An Open Letter to a Scientist Who Failed and a Challenge to Those Who Have Not Yet Failed," *Journal of Symbolic Computation*, vol. 18, no. 6, pp. 497–501, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0747717184710613

[4] B. Buchberger, "A theoretical basis for the reduction of polynomials to canonical forms," *SIGSAM Bull.*, vol. 10, no. 3, pp. 19–29, 1976. [Online]. Available: http://doi.acm.org/10.1145/1088216.1088219

[5] F. L. dit Vehel, M. G. Marinari, L. Perret, and C. Traverso, "A survey on polly cracker system," *Gröbner bases, coding and cryptography*, pp. 263–283.

[6] M. R. Albrecht, P. Farshim, J.-C. Faugère, and L. Perret, "Polly cracker, revisited," in *Advances in Cryptology–ASIACRYPT 2011*. Springer, 2011, pp. 179–196.

[7] M. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," 1979. [Online]. Available: http://dl.acm.org/citation.cfm?id=889813

[8] R. Lidl and H. Niederreiter, “Finite Fields: Encyclopedia of Mathematics and Its Applications.” *... & Mathematics with Applications*, p. 1983, 1997.

[9] R. Steinwandt, W. Geiselmann, and R. Endsuleit, “Attacking a polynomial-based cryptosystem: Polly Cracker,” *International Journal of Information Security*, vol. 1, no. 3, pp. 143–148, 2002.

[10] N. Koblitz, “Algebraic aspects of cryptography. Volume 3 of Algorithms and computation in mathematics,” 1998.

[11] D. Hofheinz and R. Steinwandt, “A “differential” attack on Polly Cracker,” in *Proc. of ISIT*, vol. 2002, 2002, p. 211.

[12] R. Rivest, “Are’strong’primes needed for RSA,” *In The 1997 RSA Laboratories Seminar Series, ...*, 1997. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.5241

[13] M. Wiener, “Safe Prime Generation with a Combined Sieve.” *IACR Cryptology ePrint Archive*, vol. 3, pp. 1–2, 2003. [Online]. Available: http://eprint.iacr.org/2003/186.pdf

[14] B. D. Weger, “Cryptanalysis of RSA with small prime difference,” *Applicable Algebra in Engineering, Communication ...*, pp. 1–11, 2002. [Online]. Available: http://www.springerlink.com/index/L19T64VJL37TMXN2.pdf

[15] ANSI, “X9. 31: 1998: Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA),” *American National Standards Institute*, 1998.

[16] S. Gao and D. Panario, “Tests and constructions of irreducible polynomials over finite fields,” in *Foundations of Computational Mathematics*. Springer, 1997, pp. 346–361.