Self-loop-based Modification of Smart and Vercauteren's Fully Homomorphic Encryption

Gu Chunsheng
School of Computer Engineering
Jiangsu Teachers University of Technology
Changzhou, China, 213001
guchunsheng@gmail.com

Abstract: This paper modifies the Smart and Vercauteren's fully homomorphic encryption scheme [SV10] by applying self-loop bootstrappable technique. The security of the modified scheme only depends on the hardness of the polynomial coset problem, removing the assumption of the sparse subset sum problem in the original paper.

Keywords: Fully Homomorphic Encryption, Polynomial Coset Problem, Sparse Subset Sum Problem

1. Introduction

By applying self-loop bootstrappable technique, we modify the fully homomorphic encryption scheme in [SV10], which uses the elementary theory of algebraic number fields. The public key in their scheme consists of a prime p and an integer $\alpha \bmod p$. The private key consists of an integer S. To encrypt a bit m, we first select a small random polynomial $R(x)$, then compute the ciphertext $c = (m + 2R(\alpha)) \bmod p$. To decrypt a ciphertext c, we can compute the message bit as follows: $m = (c - \lfloor c \times S / p + 0.5 \rfloor) \bmod 2$.

1.1 Our Contribution

The difference between our scheme and their work is mainly located on the fully homomorphic encryption. We use the self-loop bootstrappable technique, whereas the bootstrappable technique they use is to introduce a new assumption of SSSP. The security of the modified scheme is only based on the hardness of the polynomial coset problem.

1.2 Related work

Rivest, Adleman, and Dertouzos [RAD78] first investigated a privacy homomorphism, which now is called the fully homomorphic encryption (FHE). Many researchers [BGN05, ACG08, SYY99, Yao82] have worked at this open problem. Until 2009, Gentry [Gen09] constructed the first fully homomorphic encryption using ideal lattice. In Gentry's scheme, the public key is approximately n^7 bits, the computation per gate costs $O(n^6)$ operations. Smart and Vercauteren [SV10] presented a fully homomorphic encryption scheme with both relatively small key $O(n^3)$ bits , ciphertext size $O(n^{1.5})$ bits and computation per gate at least $O(n^3)$ operations, which is in some sense a specialization and optimization of Gentry's scheme. Dijk, Gentry, Halevi, and Vaikuntanathan [vDGHV10] proposed a simple fully homomorphic encryption scheme over the integers, whose security depends on the hardness of finding an approximate integer gcd. Stehle and Steinfeld [SS10] improved Gentry's fully homomorphic scheme and obtained to a faster fully homomorphic scheme, with $O(n^{3.5})$ bits complexity per elementary binary addition/multiplication gate, but the hardness assumption of the security of the scheme in [SS10] is stronger than that in [Gen09].

1.3 Outline

Section 2 gives their somewhat homomorphic encryption. Section 3 transforms the somewhat homomorphic encryption into a new fully homomorphic encryption by using self-loop bootstrappable technique. Section 4 concludes this paper and presents an open problem.

2. Somewhat Homomorphic Encryption (SHE)

To conveniently describe, we give their SHE [SV10], but do a minor modification when generating the secret key to obtain our fully homomorphic encryption. We only add the step 8 in the following SHE, all others are from that of [SV10].

2.1 Construction

Key Generating Algorithm (GenKey).

(1) Set the plaintext space to be $m \in \{0,1\}$.

(2) Choose a monic irreducible polynomial $F(x) \in Z[x]$ of degree N.

(3) Repeat:

- $S(x) \leftarrow_R B_{\infty,N}(\eta/2)$.
- $G(x) \leftarrow 1 + 2 \times S(x)$.
- $p \leftarrow res(G(x), F(x))$, where res is the resultant of $G(x), F(x)$.

(4) Until p is prime.

(5) Compute $D(x) = \gcd(G(x), F(x))$ over $F_p[x]$. Assume $\alpha \in F_p$ is the unique root of $D(x)$.

(6) Apply the XGCD-algorithm over $Q[x]$ to obtain $K(x) = \sum_{i=0}^{N-1} z_i x^i \in Z[x]$ such that $K(x) \times G(x) = p \bmod F(x)$.

(7) Set $S = (K(x) \bmod x) \bmod (2p)$.

(8) Compute the hamming weight of S in binary representation $h(S)$. If $h(S) > \beta$, then go to (2), otherwise (9), where $\beta \geq O(\log n)$ is an integer corresponding to s_2 of the fully homomorphic encryption in [SV10].

(9) Output the public key is $pk = (p, \alpha)$, the secret key is $sk = (p, S)$.

Encryption Algorithm (Enc). Given the public key pk and an message bit $m \in \{0,1\}$, choose a small random polynomial $R(x) \leftarrow_R B_{\infty,N}(\mu/2)$. Compute the ciphertext $c = (2R(\alpha) + m) \bmod p$.

Add Operation (Add). Given the public key pk, and the ciphertexts c_1, c_2, evaluate the ciphertext $c = (c_1 + c_2) \bmod p$.

Multiplication Operation (Mul). Given the public key pk and two ciphertexts c_1, c_2, evaluate a new ciphertext $c = (c_1 \times c_2) \bmod p$.

Decryption Algorithm (Dec). Given the secret key sk and a ciphertext c, decipher the message bit $m = (c - \lfloor c \times S / p + 0.5 \rfloor) \bmod 2$.

3. Fully Homomorphic Encryption (FHE)

We now construct an **FHE** by using the **SHE** by applying self-loop bootstrappable technique. We give a new algorithm **Recrypt**, which freshens a ‘dirty’ ciphertext c into a new ciphertext c_{new} with the ‘smaller’ error term and the same plaintext of c. To implement this, we generate the ciphertexts of the secret key and add them to the public key.

KeyGen. Generate pk and sk as before. Assume $\lambda = \lfloor \log(2p) \rfloor$ and the binary representation of S is $S = S_\lambda S_{\lambda-1} \ldots S_0$ Choose small random polynomials $R_i(x) \in_R B_{\infty,N}(\mu/2)$, $i \in \{0,1,\ldots,\lambda\}$, and compute $\vec{S}_i = (2R_i(\alpha) + S_i) \bmod p$.

Output the secret key $sk = (S)$ and the public key $pk = (p, \alpha, \vec{S})$.

Recrypting algorithm (Recrypt(*pk*, *c*)).

(1) Set $h_i = (c \times 2^i)/p$, $i \in \{0,1,\ldots,\lambda\}$, keeping only $k = \lambda + 4$ bits of precision after the binary point for each h_i.

(2) Compute $g = \sum_{i=0}^{\lambda} h_i \times \vec{S}_i$, and $g_{0,-1} = Add(g_0, g_{-1})$.

(3) Evaluate $u = c \bmod 2$.

(4) Output a new ciphertext $c_{new} = Add(u, g_{0,-1})$.

Theorem 3.1. Recrypt algorithm correctly generates a 'fresh' ciphertext c_{new} with the same message of c when $\beta \le s_2$, where the size of s_2 is identical to that in [SV10].

Proof: By the **DEC** algorithm, we know

$$\begin{aligned}
&(c - \lfloor c \times S / p + 0.5 \rfloor) \bmod 2 \\
&= c \bmod 2 + \lfloor c \times S / p + 0.5 \rfloor \bmod 2 \\
&= c \bmod 2 + \lfloor (c/p) \times S + 0.5 \rfloor \bmod 2 \\
&= c \bmod 2 + \left\lfloor (c/p) \times (\textstyle\sum_{i=0}^{\lambda} S_i 2^i) + 0.5 \right\rfloor \bmod 2 \\
&= c \bmod 2 + \left\lfloor \textstyle\sum_{i=0}^{\lambda} (c \times 2^i)/p \times S_i + 0.5 \right\rfloor \bmod 2
\end{aligned}.$$

It is not difficult to verify that **Recrypt** algorithm correctly generates a new ciphertext of m by using the ciphertext arithmetic operations when $\beta \le s_2$, namely, all computations in the Recrypt only replace S with its corresponding ciphertexts $\vec{S}$. Notice that Step (2) in **Recrypt** uses the methods of hamming weights, symmetric polynomials and the 'three-for-two', all of which are explained in [Gen09, vDGHV10].■

4. Conclusion and Open Problem

By using self-loop bootstrappable technique, we modified the fully homomorphic encryption scheme in [SV10], whose security only depends on the hardness of the polynomial coset problem.

An interesting open problems is how to efficiently generate the public key and the secret key.

References

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In Proc. of STOC 1996, pages 99-108, 1996.
[ACG08] C. Aguilar Melchor, G. Castagnos, and G. Gaborit. Lattice-based homomorphic encryption of vector spaces. In IEEE International Symposium on Information Theory, ISIT'2008, pages 1858-1862, 2008.
[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. Lecture Notes in Computer Science, 2005, Volume 3378, pages 325-341, 2005.
[vDGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In Proc. of Eurocrypt, volume 6110 of LNCS, pages 24-43. Springer, 2010.
[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In Proc. of STOC, pages 169-178, 2009.
[GHV10] C. Gentry and S. Halevi and V. Vaikuntanathan. A Simple BGN-type Cryptosystem from LWE. In Proc. of Eurocrypt, volume 6110, pages 506-522, 2010.
[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Proc. of STOC, pages 197-206, 2008.
[LPR10] V. Lyubashevsky and C. Peikert and O. Regev. On Ideal Lattices and Learning with Errors over Rings. In Proc. of Eurocrypt, volume 6110, pages 1–23, 2010.
[Mic07] D. Micciancio Generalized compact knapsaks, cyclic lattices, and efficient one-way functions. Computational Complexity, 16(4):365-411.
[MR07] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussion measures. SIAM Journal Computing, 37(1):267-302, 2007.
[Reg09] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, Journal of the ACM (JACM), v.56 n.6, pages1-40, 2009.
[SS10] D. Stehle and R. Steinfeld. Faster Fully Homomorphic Encryption. Cryptology ePrint Archive: Report 2010/299: http://eprint.iacr.org/2010/299.
[SV10] N. P. Smart and F. Vercauteren Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. Lecture Notes in Computer Science, 2010, Volume 6056/2010, 420-443.
[SYY99] T. Sander, A. Young, and M. Yung. Non-interactive CryptoComputing for NC1. In 40th Annual Symposium on Foundations of Computer Science, pages 554{567. IEEE, 1999.
[RAD78] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, pages 169-180, 1978.
[Yao82] A. C. Yao. Protocols for secure computations (extended abstract). In 23rd Annual Symposium on Foundations of Computer Science (FOCS '82), pages 160-164. IEEE, 1982.