On the CCA-1 Security of Somewhat Homomorphic Encryption over the Integers*

Zhenfei Zhang, Thomas Plantard, and Willy Susilo

Centre for Computer and Information Security Research
School of Computer Science & Software Engineering (SCSSE)
University Of Wollongong, Australia
{zz920, thomaspl, wsusilo}@uow.edu.au

Abstract. The notion of fully homomorphic encryption is very important since it enables many important applications, such as the cloud computing scenario. In EUROCRYPT 2010, van Dijk, Gentry, Halevi and Vaikuntanathan proposed an interesting fully homomorphic encryption scheme based on a somewhat homomorphic encryption scheme using integers. In this paper, we demonstrate a very practical CCA-1 attack against this somewhat homomorphic encryption scheme. Given a decryption oracle, we show that within $O(\lambda^2)$ queries, we can recover the secret key successfully, where λ is the security parameter for the system.

Keywords: Fully Homomorphic Encryption, Somewhat Homomorphic Encryption, CCA-1 attack, Approximate GCD.

1 Introduction

Fully homomorphic encryption is a very important notion for cloud computing. It allows the cloud to process users' encrypted data without the need to decrypt them.

Essentially, fully homomorphic encryption schemes enable one to apply homomorphic operations over arbitrary number (n) of given ciphertexts c_1, c_2, ..., c_n without the need to know the corresponding plaintexts m_1, m_2, ..., m_n.

This notion, which was initially named "data homomorphisms", was proposed by Rivest, Shamir and Dertouzos [1] shortly after the introduction of RSA [2]. For many years, schemes that support partial homomorphism have been proposed. Nevertheless, the construction of fully homomorphic encryption had been a long standing open research problem, until the recent Gentry's breakthrough work [3, 4], where a fully homomorphic scheme was proposed.

The initial construction of Gentry's FHE scheme (referred to as Gentry scheme throughout this paper) uses ideal lattices. His work was then refined and optimized by Smart and Vercauteren [5] (referred to as Smart-Vercauteren variant), Stehlé and Steinfeld [6], and Gentry and Halevi [7] (referred to as

* This work is supported by ARC Future Fellowship FT0991397.

Gentry-Halevi variant). Meanwhile, van Dijk et al. proposed a FHE scheme using integers (referred to as vDGHV variant) [8], which is later extended by Coron et al. [9] (referred to as CMNT variant), Coron et al. [10], whose security was re-evaluated by Chen and Nguyen [11]. There is also a third type of FHE variants that are based on coding theory proposed recently [12], whose structure is very close to the construction from ideal lattices.

The essential idea of such a scheme is to construct a somewhat homomorphic encryption (SHE) scheme and then convert it to a fully one using Gentry's bootstrapping technique (see section 2.2) [3, 13]. Therefore, usually a fully homomorphic encryption scheme contains two parts, a somewhat homomorphic encryption scheme whose ability of homomorphic operations is limited, and a bootstrapping technique that breaks such a limitation.

It is known that any FHE scheme that adopts Gentry's bootstrapping technique cannot be CCA-1 secure (see Subsection 2.4 for definitions). Since the bootstrapping technique requires one to publish the encryption of their secret keys, therefore, if there exists a decryption oracle, then the attacker can recover the secret key by incorporating this oracle within k queries, where k is the number of bits of the security key (in the case of vDGHV scheme, the secret key of the squashed decryption algorithm has $O(\lambda^5)$ bits, hence, a CCA-1 attack is successful in $O(\lambda^5)$ queries). As a result, the CCA-1 security cannot be achieved by FHE schemes that use bootstrapping technique.

We should highlight that it is important to investigate a somewhat homomorphic encryption (SHE) scheme by itself, since it has many applications, such as medical, financial and the advertising domains as mentioned in [14]. It is noted that in these applications, only SHE schemes are required.

As far as a SHE is concerned, the CCA-2 security is also not achievable. As a SHE allows certain level of homomorphic operations on ciphertexts, one can modify the CCA-2 challenge ciphertext and submit it to the decryption oracle. Therefore, the attacker can recover the plaintext.

However, whether a SHE can be CCA-1 secure remains an open problem. Indeed, in [15], Loftus et al. showed a CCA-1 attack against Gentry-Halevi SHE scheme and Smart-Vercauteren SHE scheme (we refer to this work as LMSV attack), and proposed a CCA-1 secure SHE scheme (referred to as LMSV variant).

Our Contributions

In this paper, we propose a CCA-1 attack against vDGHV SHE scheme, which is *different* from the LMSV attack. We should highlight that the technique used in the LMSV scheme to stop the LMSV attack *is not* applicable to our case. With a decryption oracle of vDGHV SHE scheme, we can recover the secret key successfully. Also, since the ciphertexts of vDGHV SHE scheme and CMNT SHE scheme share the same structure, our attack can be applied to CMNT SHE scheme as well. Moreover, our CCA-1 attack against vDGHV FHE scheme uses $O(\lambda^2)$ queries, where λ is the security parameter of the system, while a trivial CCA-1 attack uses $O(\lambda^5)$ queries.

2 Preliminaries

2.1 Notations

Let λ be the security parameter of the system, i.e., it takes at least 2^λ operations to break the system. For integers z and d, denote $[z]_d$ for the reduction of z mod d within $(-d/2, d/2]$. For a rational number q, let $\lfloor q \rceil$ be the closest integer to q, $[q]$ the fractional part of q.

2.2 Gentry's Framework

As stated earlier, a fully homomorphic encryption scheme essentially consists of two parts: a somewhat homomorphic encryption scheme and a bootstrapping technique.

The somewhat homomorphic encryptions scheme enables basic additions and multiplications over $\mathbb{F}_2$. Hence, it is arithmetically "complete", because essentially any circuit is derived from additions and multiplications over $\mathbb{F}_2$ [3]. However, in such a scheme, in order to bring some security strength, the ciphertexts contain a random "noise". The size of the noise is limited, to ensure a valid decryption. Nevertheless, it grows in size as the ciphertext is processed to homomorphically evaluate the function on its plaintext. Once the size of the noise in the ciphertext exceeds a certain threshold, then the ciphertext can no longer be decrypted correctly.

The "bootstrapping technique" is to solve such a limitation. If there is a guarantee that the maximum evaluation circuit depth of this somewhat homomorphic scheme is greater than its decryption circuit depth, then one can reduce the noise by evaluating its own decryption circuit, and consequently, convert the somewhat homomorphic scheme to a fully homomorphic scheme.

The general idea of bootstrapping is to "refresh" a ciphertext, namely given a ciphertext c for some plaintext m, compute a ciphertext c' such that the size of the noise in c' is smaller than the size of the noise in c. The algorithm to conduct this ciphertext refreshing operation is called "Recrypt", which enables one to evaluate arbitrarily large circuits.

To enable Recrypt, one publishes an encryption of the (SHE) secret key. The new ciphertext c' encrypts the same message, but it maintains a smaller noise,

$$c' = \textsc{Recrypt}(\textsc{Encrypt}(sk), \textsc{Encrypt}(c)),$$

$$\textsc{Decrypt}(sk, c') = \textsc{Decrypt}(sk, c).$$

2.3 Overview of vDGHV SHE Scheme

In this subsection, we briefly review the vDGHV SHE scheme. We omit the description of CMNT scheme. However, it is important to note that the ciphertexts of those two schemes resemble the same structure (i.e., $c_i = g_i p + 2r_i$), and their decryption algorithms are identical.

Parameters The following are the parameters used in the vDGHV somewhat homomorphic encryption scheme.

- α: the maximum length of the noise r_i, i.e.: $r_i \in [-2^\alpha, 2^\alpha)$;
- β: the length of the secret key p, i.e.: $p \in (2^\beta, 2^{\beta+1})$
- γ: the maximum length of an integer in the public key;
- σ: the number of public keys used in one encryption;
- n: the number of integers in the public key;

	α	β	γ	n
Minimum	λ	λ^2	$\lambda^4 \log \lambda$	$\lambda^4 \log \lambda + \lambda$
Recommended	λ	λ^2	λ^5	$\lambda^5 + \lambda$

Table 1. Parameter Configurations.

In their paper, the author also gave two sets of parameters, one for the minimum requirement of the system, and another for the recommended configuration. We briefly list their configurations in table 1.

The Scheme Now we describe vDGHV SHE scheme. The somewhat homomorphic encryption scheme consists of four algorithms:

KeyGen(λ)

- Generate parameters $\alpha, \beta, \gamma, \sigma, n$ in function of λ;
- Generate a random odd integer $p \in [2^\beta, 2^{\beta+1})$;
- Generate n random integers $\{r_i \in [-2^\alpha, 2^\alpha)\}$;
- Generate n random integers $\{g_i \in [0, 2^{\gamma-\beta})\}$;
- $sk \leftarrow p$;
- $x_0 \leftarrow g_0 p + 2r_0$;
- $x_i \leftarrow g_i p + 2r_i \bmod x_0$, $0 < i \leq n$;
- Reorder $\{x_i\}$ such that x_0 is the smallest;
- $pk \leftarrow \{x_i, \alpha, \sigma\}$.

Encrypt(m, pk)

- Generate a bit sequence $\{s_i\}$, such that $\sum s_i = \sigma$;
- $c \leftarrow m + 2 \times r + \sum_{i=1}^{n-1}(s_i \times x_i) \pmod{x_0}$, where $r \in [-2^\alpha, 2^\alpha)$;

Decrypt(c, sk)

- $m \leftarrow (c \bmod 2) \oplus (\lfloor c/p \rceil \bmod 2)$.

Remark 1 *Essentially, the* DECRYPT *algorithm is simplified from* $[c - \lfloor c/p \rceil]_2$*, where* $\lfloor z \rceil$ *is to find the closest integer to* z*, while* $[z]_p$ *is to find the residue of* $z \bmod p$ *in* $(-p/2, p/2]$*. This is slightly different from* $c \bmod p \bmod 2$*.* $c \bmod p$ *returns an integer in* $[0, p)$*, while* $c - p \times \lfloor c/p \rceil$ *finds an integer in* $(-p/2, p/2]$*.*

EVALUATE(c_1, c_2, ..., c_k, $\mathcal{P}$, pk)

– Return $\mathcal{P}(c_1, c_2, ..., c_k)$.

$\mathcal{P}$ is a k-inputs evaluation polynomial while the depth of its circuit $\mathcal{C}_D$ is lower than the maximum circuit depth allowed by this SHE.

This SHE supports homomorphic additions and multiplications, when $\alpha \ll \beta$. For instance, suppose $c_1 = m_1 + g_1 p + 2r_1$ and $c_2 = m_2 + g_2 p + 2r_2$ for certain g_1, r_1, g_2, r_2, the product of two ciphertexts $c_1 c_2 = m_1 m_2 + 2(r_1 m_2 + r_2 m_1 + 2r_1 r_2) + p(g_1 m_2 + 2g_1 r_2 + g_2 m_1 + 2g_2 r_1 + g_1 g_2 p)$. One can observe that the decryption of $c_1 c_2$ is $m_1 m_2$, as long as $2(r_1 m_2 + r_2 m_1 + 2r_1 r_2) \in (-p/2, p/2]$. Therefore, the above scheme is somewhat homomorphic.

However, the homomorphic circuit depth is limited, i.e., the noise grows after each operation, and eventually the absolute value of the noise will be greater than $p/2$ and a decryption error is then possibly generated.

Suppose we want to evaluate a circuit whose depth is greater than this SHE permits, we break the circuit into several sub-circuits. For each sub-circuit, the resulting noise is less than the threshold $(p/2)$. Then we refresh the resulting ciphtertext using the bootstrapping technique. We refer the readers to the original scheme for more details. In the following, we describe the bootstrapping technique in general.

To bootstrap, firstly, decryption circuit in Remark 1 need to be modified. The original decryption requires at least one division, while the modified one consists of only additions. As we have shown earlier, the noise grows significantly faster in a multiplication than in an addition.

Remark 2 *The squash technique transfers an original secret key of* $O(\lambda^2)$ *bits into a new secret key of* $O(\lambda^5)$ *bits. As a requirement of bootstrapping, one is obliged to publish the encryption of the secret key, which in this case is the new one. Therefore, a trivial CCA-1 attack to the* vDGHV *FHE uses* $O(\lambda^5)$ *queries.*

Then, because the modified decryption circuit depth is relatively low, now it is possible to carry out the decryption circuit homomorphically, through the proposed SHE. In practice, we encrypt ciphertexts, denoted by $Enc(c)$ and the public keys, denoted by $Enc(pk)$. Let $\mathcal{C}_D$ be the decryption circuit, then $\text{EVAL}(\mathcal{C}_D, Enc(c), Enc(pk)) = Enc(m)$. This is because firstly $\mathcal{C}_D(c, pk) = m$ and secondly, $\mathcal{C}_D$ can be carried out homomorphically. Therefore, we obtain a new ciphertext $Enc(m)$.

The new ciphertext, $Enc(m)$ has a refreshed noise level (less than 2^α), which means $Enc(m)$ can be evaluated again. By doing this repeatedly, we can evaluate circuit with any depth homomorphically. Therefore, a fully homomorphic encryption scheme is achieved.

2.4 Security Models

In the following, we describe briefly both CCA-1 and CCA-2 attacks for completeness [16]. The IND-CCA-1/2 security game is defined as follows:

1. The challenger runs KeyGen algorithm and output a secret key sk and a public key pk;
2. The attacker is given two oracles, an encryption oracle and a decryption oracle;
3. The challenger generates $c = \text{Encrypt}(m_b, sk)$, where $b \in \{0, 1\}$;
4. (Only for CCA-2) The attacker is given two oracles again, but it can not query on c;
5. The attacker output b'.

We say that an encryption scheme is CCA-1/2 secure if the advantage of the attacker to win the game $(Pr[b = b'] - 1/2)$ is negligible.

2.5 LMSV CCA-1 Attack

In this subsection we briefly revisit the LMSV attack against Gentry-Halevi SHE and Smart-Vercauteren SHE schemes. For completeness, we show the above SHE schemes first, as well as LMSV SHE scheme, which is resistant against their own CCA-1 attack.

Recall that all three SHE schemes consist of four algorithms, KeyGen, Encrypt, Decrypt and Evaluate. The LMSV attack requires the first three algorithms.

KeyGen(λ)

- Generate parameters n, t, ρ in function of λ;
- Set $f(x) \leftarrow x^n + 1$, n is a power of 2;
- Pick a random $n - 1$ degree polynomial $v(x)$, with coefficients $v_i \in (0, 2^t)$, denote $\vec{v}$ the vector form of coefficients of $v(x)$;
- Generate a matrix V from $\vec{v}$ and check if the Hermite Normal Form (HNF) of V has the correct form (as shown below) and if d is an odd number;

$$V = \begin{vmatrix} v_0 & v_1 & v_2 & \dots & v_n \\ -v_n & v_0 & v_1 & \dots & v_{n-1} \\ -v_{n-1} & -v_n & v_0 & \dots & v_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -v_1 & -v_2 & -v_3 & \dots & v_0 \end{vmatrix}, \quad HNF(V) = \begin{vmatrix} d & 0\,0 \dots 0 \\ [-a]_d & 1\,0 \dots 0 \\ [-a^2]_d & 0\,1 \dots 0 \\ \vdots & \vdots\,\vdots\,\ddots\,\vdots \\ [-a^{n-1}]_d & 0\,0 \dots 1 \end{vmatrix}.$$

- Find the polynomial $w(x)$, such that $w(x) \times v(x) = d \bmod f(x)$;
- $sk \leftarrow w$, where w is one of odd coefficients of $w(x)$;
- $pk \leftarrow \{a, d, \rho\}$.

Encrypt(m, pk)

- Generate a degree $n-1$ polynomial $r(x)$, with coefficients $r_i \in (0, \rho)$;
- $c(x) \leftarrow m + 2 \times r(x)$, where $m \in \{0, 1\}$;
- $c \leftarrow [m + c(a)]_d$.

The three SHE schemes have different decryption algorithms. We firstly introduce DecryptGH and DecryptSV that are used in Gentry-Halevi SHE and Smart-Vercauteren SHE, respectively.

DecryptGH/DecryptSV(c, sk)

- $m \leftarrow [c \times w]_d \bmod 2$.

The decryption algorithm for LMSV SHE uses a ciphertext check procedure to stop the LMSV attack. Here, T is the threshold used for the ciphertext check. T need to be greater than a function of ρ.
DecryptLMSV(c, sk)

- $c(x) \leftarrow c - \lfloor c \times w(x)/d \rceil \times g(x) \bmod f(x)$;
- $c' \leftarrow [c(a)]_d$;
- If $c' \neq c$ or $\|c(x)\|_\infty \geq T$ return $\perp$;
- Else, return $c(x) \bmod 2$

Now we describe the LMSV attack. Decrypt algorithms for Gentry-Halevi SHE and Smart-Vercauteren SHE schemes are $m \leftarrow [c \times w]_d \bmod 2$. This decryption will be valid as long as $[c \times w/d] \leq 1/2$. Therefore, for a certain key set (w, d), the maximum value c' allowed is a fixed integer. The adversary picks several different "ciphertexts", and pass them to the decryption oracle to check if they can be decrypted correctly. Eventually, the attacker will recover the threshold c' which is the maximum integer that can be decrypted correctly. This c' in return gives the attacker w, the secret key.

To stop this attack, Loftus et al. proposed a ciphertext check procedure. The ciphertext that is to be decrypted, will be "disassembled" into the generating polynomial $c(x)$. Recall that $c(x) = 2 \times r(x) + m$, hence, for valid ciphertexts, $\|c(x)\|_\infty$ is bounded by a certain threshold smaller than T, while for invalid "ciphertexts" (i.e., integers picked by attacker), the corresponding $c(x)$ can have arbitrary coefficients. Therefore, in the latter case, an error $\perp$ is generated, and the decryption stops.

3 Our CCA-1 attack

In this section we present our CCA-1 attack. We use vDGHV SHE scheme to demonstrate our attack. However, the following attack can be applied to CNMT with a trivial modification.

The security strength of vDGHV SHE comes from the noise that is added to ciphertexts. If we can somehow reduce the noise in ciphertext, then the scheme will no longer be secure. With the help of a decryption oracle, we can eliminate the noise. Hence, we achieve a CCA-1 attack.

We propose two variants that follow the same idea. The first variant requires several ciphertexts. The main idea is to eliminate the noise and then, to find the GCD of the remaining parts. The second variant requires only *one* ciphertext, and we are able to recover the secret key directly.

We note that our attack recovers the secret key that allows us to decrypt any valid ciphertexts, and does *not* require any access to $\mathcal{O}_D$ at Stage 5 in the CCA attack model. Thus, our attack falls in the category of CCA-1. However, we also note that essentially, our attack is stronger than CCA-1, because instead of solving one challenge, we recover the secret key.

3.1 The Attack

Essentially, in vDGHV SHE scheme, the public keys (x_i-s) can be treated as ciphertexts encrypting 0-s with smaller noise. Therefore, the following algorithms can be applied on public keys with less cost. However, in order to strictly follow the definition of CCA-1 attack, we apply our algorithms on real ciphertexts.

We note that for any correct ciphertext c_i, the following holds that

$$c_i = m_i + 2r'_i + g'_i p$$

for certain $r'_i \in (-p/4, p/4]$ and integer g'_i, since if $|2r'| > p/2$ decryption error will be induced. For convenience, denote $\alpha' = \beta - 1$.Using the recommended parameter configuration, we have $\alpha' = \lambda^2 - 1$.

Now we show our attack against vDGHV SHE scheme. Suppose we have k ciphertexts $c_1, c_2, ..., c_k$ of encrypted 0-s, i.e.: $c_i = g'_i p + 2r'_i$. It holds that $\text{DECRYPT}(c_i, sk) = 0$. The length of r'_i-s is no greater than α'.

Let $\mathcal{O}_D(c)$ be the decryption oracle that returns $\text{DECRYPT}(c, sk)$. The following pieces of pseudo-code describe two variants of our attack.

Algorithm 1 NoiseEli(c)

```
l_p ← 2^β − 2^{α'+1}
r_p ← 2^{β+1} + 2^{α'+1}
while r_p − l_p > 2 do
  s ← ⌊(l_p + r_p)/4⌉ × 2
  if O_D(c + s) = 0 then
    l_p ← s
  end if
  if O_D(c + s) = 1 then
    r_p ← s
  end if
end while
if O_D(c + s + 1) = 1 then
  s ← s + 1
end if
return c' ← c + s
```

Algorithm 1: NoiseEli is to help to eliminate the noise in ciphertext. Instead of generating a ciphertext with no noise, this algorithm generates a ciphertext with a fixed noise.

Algorithm 2: CCA-GCD describes the first variant of our attack, while Algorithm 3 CCA-p describes the second variant of our attack.

Algorithm 2 CCA-GCD$(c_0, c_1, ..., c_k)$

$c'_0 \leftarrow$ NoiseEli(c_0)
$c'_1 \leftarrow$ NoiseEli(c_1)
$c'_2 \leftarrow$ NoiseEli(c_2)
$p' \leftarrow \gcd(c'_2 - c'_1, c'_1 - c'_0)$
$t \leftarrow 3$
while $p' \geq 2^{\beta+1}$ **and** $t \leq k$ **do**
  $c'_t \leftarrow$ NoiseEli(c_t)
  $p' \leftarrow \gcd(c'_t - c'_{t-1}, p')$
  $t \leftarrow t + 1$
end while
return p'

The first algorithm inputs a ciphertext $c = 2r' + g'p$ with any noise r', it outputs a new ciphertexts $c' = g'p + \lfloor p/2 \rfloor$. The new ciphertext contains a constant noise of $\lfloor p/2 \rfloor$.

For any ciphertext $c = 2r + gp$, the Decrypt algorithm will always output 0, as long as $|2r| \leq \lfloor p/2 \rfloor$, and output 1 if $2r > \lfloor p/2 \rfloor$. Denote $l = \lfloor p/2 \rfloor - 2r$. l represents the threshold, such that $\mathcal{O}_D(c + l) = 0$ and $\mathcal{O}_D(c + l + 2) = 1$. Also, we know that $l \in (2^\beta - 2^{\alpha'+1}, 2^{\beta+1} + 2^{\alpha'+1})$. Therefore, we set l_p and r_p to be the lower and upper bound of l. Then we start a while loop to narrow the bound as in Algorithm 1.

Algorithm 3 CCA-p(c)

$a \leftarrow$ NoiseEli(c)
$b \leftarrow$ NoiseEli$(-c)$
return $a + b + 1$

3.2 Correctness

In this subsection we prove the correctness of our attack.

For any *even* integer $s \in (2^\beta - 2^{\alpha'+1}, 2^{\beta+1} + 2^{\alpha'+1})$, decrypting $c + s$ has two possible consequences:

- If $\mathcal{O}_D(c + s) = 1$, we know that the threshold l for this ciphertext is smaller than s, then we move the upper bound r_p to s;

– One the other hand, if $\mathcal{O}_D(c+s)=0$, we know that the threshold l for this ciphertext is greater than s, then we move the lower bound l_p to s;

By the end of the loop, we have $s=l_p=r_p-2$. Also, it holds that $\mathcal{O}_D(c+s)=0$ and $\mathcal{O}_D(c+s+2)=1$. If $\lfloor p/2 \rfloor$ is an even integer, then $s=\lfloor p/2 \rfloor$, and $\mathcal{O}_D(c+s+1)=1$. By contrast, if $\lfloor p/2 \rfloor$ is odd, then $s=\lfloor p/2 \rfloor -1$, and $\mathcal{O}_D(c+s+1)=0$. In this case, we increase s by 1.

Hence, s is the threshold l we were looking for, and $c+s=gp+\lfloor p/2 \rfloor$. Therefore, we successfully generate a fixed noise ciphertext in $\log(2^{\beta}+2^{\alpha'+2})+1$ queries.

The second algorithm is more straightforward. Given $k+1$ outputs of Algorithm 1, we obtain k linear independent noise free ciphertexts. By running a classic GCD algorithm we obtain p'. It holds that either $p=p'$ or $p|p'$.

To have $p=p'$ it requires $\gcd(g_2-g_1, g_3-g_2, ..., g_k-g_{k-1})=1$. The probability of k random integers from $\mathbb{Z}$ to be coprime is $1/\zeta(k)$, where $\zeta(x)$ is the Riemann Zeta function $\sum_{i=1}^{\infty}\frac{1}{i^x}$ (see [17] for more details), while the probability of having k numbers randomly chosen form $(0, 2^{\lambda^5})$ to be coprime is greater than $1/\zeta(k)$.

In practice, 4 random integers has $1/\zeta(4)>92\%$ probability of being coprime, while 7 random integers has $1/\zeta(7)>99\%$ probability of being coprime. Our test (see subsection 3.5) confirmed this result, where on average cases, 3 random integers are coprime.

For the last algorithm, we generate $a=\textsc{NoiseEli}(c)$ and $b=\textsc{NoiseEli}(-c)$. It holds that:

$$a=gp+\lfloor p/2 \rfloor, \qquad b=-gp+\lfloor p/2 \rfloor.$$

Therefore, we obtain $2\times\lfloor p/2 \rfloor$ from $a+b$. Because p is an odd integer, we recover the secret key by $p=a+b+1$.

3.3 Efficiency

We examine the efficiency of our last two algorithms. For original ciphertexts (no homomorphic operations have been evaluated on them), it requires $\log(2^{\beta}+2^{\alpha'+2})+1<\beta+3$ queries to find the fixed noise ciphertext. CCA-GCD algorithm requires a minimum 3 fixed noise ciphertexts. Therefore, in best cases we recover the secret key in $3(\beta+3)$ queries. As $\beta=\lambda^2$, Algorithm 2 recovers the secret key in $O(\lambda^2)$ operations.

Algorithm 3 also works on $O(\lambda^2)$ but with better performance. To be more precise, it uses one ciphertext only, therefore to eliminate the noise requires at most $2(\beta+3)$ queries.

It is true that Algorithm 3 is more efficient that Algorithm 2. The reason that we propose Algorithm 2 is that we observe Algorithm 3 will fail if we modify the decryption circuit. For instance, if $c \bmod p$ returns an integer within $[0, p)$ instead of $(-p/2, p/2]$, then Algorithm 3 will be unsuccessful. However, in this case Algorithm 2 is still valid.

3.4 An Example

In this subsection, we give an example of our CCA-1 attack. In our example, the security parameter λ is 2. Therefore, the noise r'_i and the multiplier g'_i are bounded by 2^2 and 2^{28}, respectively. The secret key p is an odd integer between 2^4 and 2^5. The ciphertext is in form of $c'_i = 2r'_i + g'_i p$. Table 2 lists three ciphtertexts.

	r'_i	p	g'_i	c'_i
c_1	-3	19	343759059	6531422115
c_2	1	19	230194545	4373696357
c_3	2	19	276209466	5247979858

Table 2. Three sample ciphertexts.

The results below indicate that we retrieve the secret key successfully. Table 3 shows that the noise is successfully eliminated within maximum 5 queries for each ciphertext. We recover s_i for each ciphertext such that $c_i + s_i = g_i p + (p-1)/2$.

3.5 Implementation

In this subsection, we show the result of our implementation of our attack with different λ. This implementation is based on the NTL library [18].

The implementation was conducted in a 2.66 GHz CPU. The memory was always sufficient, as it merely required more than 600 Mbs. We started from $\lambda = 2$, and increased λ continuously until it reached 32. For each λ, we fed the program with 100 different seeds, and recorded the average time to find the secret key p, as well as average number of ciphertexts required for Algorithm 2.

The average number of ciphertexts required for Algorithm 2 for different choice of λ is quite stable. Approximately 3.8 ciphertexts are required to recover the secret key p. This implies that on average case the number of integers required to have them being coprime is 2.8.

Fig. 1 shows the timing results of our implementation. The x axis shows the choice of λ, while the y axis indicates the average time (in seconds) for each attack. Statistically, attack CCA-P uses approximately 1.9 times less time in comparison to attack CCA-GCD (this is due to the number of ciphertexts required to be noise-eliminated), and this is consistent with our result.

4 Discussions

4.1 On the Difference between our attack and LMSV attack

We note that the LMSV attack is different from the attack described in this paper. The LMSV attack uses the decryption oracle to find the integer s such that $[w \times s/d] = 1/2$, and eventually recover the secret key, while our attack

NoiseEli(c_1)				
l_p	r_p	s_1	$\mathcal{O}_D(c_1 + s_1)$	action
0	24	12	0	$l_p \leftarrow 12$
12	24	18	1	$r_p \leftarrow 18$
12	18	14	0	$l_p \leftarrow 14$
14	18	16	1	$r_p \leftarrow 16$
14	16	14		end of loop
$\mathcal{O}_D(c_1 + s_1 + 1) = 1 \Longrightarrow s_1 = 15$				

NoiseEli(c_2)				
l_p	r_p	s_2	$\mathcal{O}_D(c_2 + s_2)$	action
0	24	12	1	$r_p \leftarrow 12$
0	12	6	0	$l_p \leftarrow 6$
6	12	8	1	$r_p \leftarrow 8$
6	8	6		end of loop
$\mathcal{O}_D(c_2 + s_2 + 1) = 1 \Longrightarrow s_2 = 7$				

Table 4 and 5 show how to extract p from c_i' in two ways. Both of the two examples uses NoiseEli to find constant noise ciphertext. As displayed in the tables, in GCD-CCA we recover $3p$ instead of p, and we did not further proceed the algorithm, since it is merely an example. This example requires access to the oracle for 14 times for the GCD-CCA variant and 10 times for CCA-p.

NoiseEli(c_3)				
l_p	r_p	s_3	$\mathcal{O}_D(c_3 + s_3)$	action
0	24	12	1	$r_p \leftarrow 12$
0	12	6	1	$r_p \leftarrow 6$
0	6	2	0	$l_p \leftarrow 2$
2	6	4	0	$l_p \leftarrow 4$
4	6	4		end of loop
$\mathcal{O}_D(c_3 + s_3 + 1) = 1 \Longrightarrow s_3 = 5$				

Table 3. Eliminate the noise of three ciphertexts.

CCA-GCD(c_1,c_2)	
$c_1' = c_1 + 15 - c_2 - 7$	$c_2' = c_2 + 7 - c_3 - 5$
$p' = \gcd(c_1', c_2') = 57$	

Table 4. Find p with CCA-GCD.

NoiseEli(c_1)				
l_p	r_p	s_1	$\mathcal{O}_D(c_1 + s_1)$	action
0	24	12	0	$l_p \leftarrow 12$
12	24	18	1	$r_p \leftarrow 18$
12	18	14	0	$l_p \leftarrow 14$
14	18	16	1	$r_p \leftarrow 16$
14	16	14		end of loop
$\mathcal{O}_D(c_1 + s_1 + 1) = 1 \Longrightarrow s_1 = 15$				

NoiseEli($-c_1$)				
l_p	r_p	s_{-1}	$\mathcal{O}_D(c_1 + s_{-1})$	action
0	24	12	1	$r_p \leftarrow 12$
0	12	6	1	$r_p \leftarrow 6$
0	6	2	0	$l_p \leftarrow 2$
2	6	4	1	$r_p \leftarrow 4$
2	4	2		end of loop
$\mathcal{O}_D(-c_1 + s_{-1} + 1) = 1 \Longrightarrow s_{-1} = 3$				

CCA-p(c_1)	
$a \leftarrow c_1 + 15$	$b \leftarrow -c_1 + 3$
$p \leftarrow (a + b) + 1 = 19$	

Table 5. Find p with CCA-p.

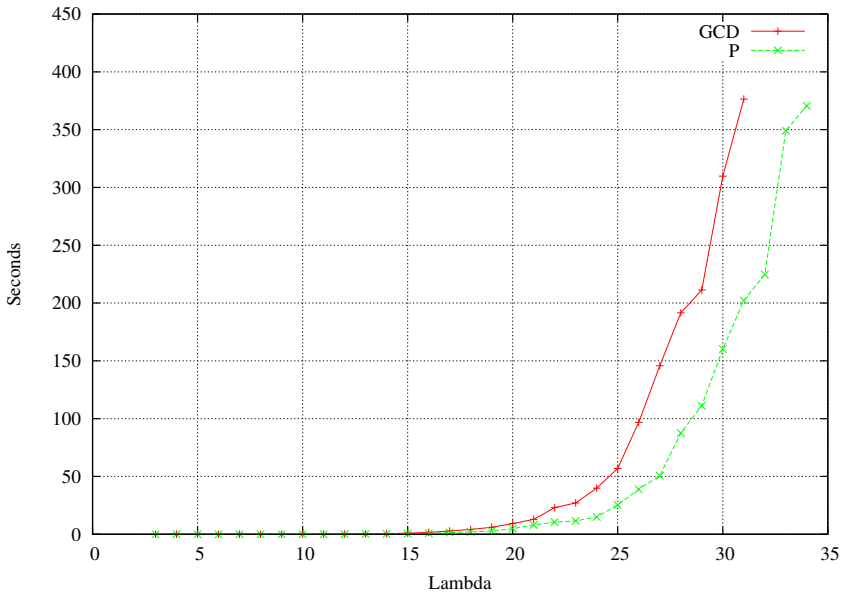

Fig. 1. Average time for recovering p.

aims to manipulate the noise in the ciphertexts. By recovering the noise, our attack will recover the secret key of vDGHV variant.

A proof of such a difference is that our attack can be adapted to recover the noise for LMSV SHE scheme as well. However, this does not help us to recover the secret key or break the CCA-1 security (see subsection 4.2). Another evidence is that using LMSV SHE's solution (i.e., generating $\perp$ for invalid ciphertext) *will not* stop our attack either (see subsection 4.3).

4.2 On Adapting Our Attack to SHE Schemes with Ideal Lattice

We note that our method cannot be adapted to attack SHE schemes that use ideal lattices. In a typical SHE scheme with ideal lattice, a message is encrypted in a similar way:

$$c = m + rB_I + gB_J^{pk},$$

where I and J are two ideal lattices that are co-prime. r and g are some random elements generated during encryption. Lattice I is usually $< 2 >$, which are all even numbers, and B_I is a basis of I. As for lattice J, it generates a good basis B_J^{sk} and a bad basis B_J^{pk}. Then B_J^{pk} is used for encryption, while B_J^{sk} becomes the secret key.

Therefore, even we can somehow eliminate r through our attack, we still need to solve such a problem: given as many $g_i B_J^{pk}$, find B_J^{sk}. This is a GGH

type cryptosystem [19]. As a result, we cannot recover B_J^{sk} directly using our technique.

Take LMSV SHE scheme for instance, for a ciphertext $c = 2r(a) + m \pmod{d}$, our attack recovers $r(x)$. To recover the i-th coefficient of $r(x)$, one passes $c + 2 \times r' \times a^i$ to the decryption oracle, where r' is a random integer picked by the attacker. By observing if the oracle returns m or $\perp$, one increases or decrease r' accordingly. Eventually, the attacker obtains $r' + r_i = T$. As a result, the attacker recovers one coefficient of $r(x)$. By doing this repetitively, one recovers the entire $r(x)$.

However, we note that this does not lead to a CCA-1 attack or a secret key attack. To recover the secret key one still need to solve the following problem: given an ideal lattice in the form of a, d, find a good basis of this lattice.

4.3 On LMSV SHE CCA-1 Approach

In this subsection we consider the existing proposed solution to make vDGHV SHE scheme CCA-1 secure. We note that our attack is successful, since we are able to eliminate the noise. Therefore, if there exist some techniques to disturb the noise elimination, our attack will fail.

Loftus et al. [15] showed a solution to combat their own CCA-1 attack against Gentry-Halevi SHE scheme/Smart-Vercauteren SHE scheme (which we refer to as the LMSV SHE scheme). However, the solution in LMSV SHE scheme is *not applicable* in our case. Their possible solution is to generate some error $\perp$ (or even some random 0-s or 1-s), when the decryption oracle detects that the noise r is very close to $\pm(p-1)/2$. The decryption algorithm sets a bound T, such that when $(p-1)/2 - |r| < T$, it will not proceed decryption. However, essentially it will still leak some information. We can modify our attack to find T and consequently find a fixed noise ciphertext.

We modify our attack as follows: for each round, we query to the oracle multiply times. If the feedbacks are consistent (meaning that the attacker is not confused by random 0-s and 1-s) and not $\perp$, we proceed to the next round. Otherwise, we recover a fixed noise ciphertext with a noise level of T. Hence, our attack will still be successful even after the "patch" suggested by Loftus et al. [15].

5 Conclusion

Fully homomorphic encryption schemes play an important role in the security of many practical applications, such as cloud computing. Although the CCA-1 security for a FHE scheme is not achievable, whether its SHE scheme is CCA-1 secure remains an interesting research question, since a SHE scheme has potential to enable promising applications.

In this paper, we proposed a CCA-1 attack against vDGHV SHE scheme with integers. Unlike other schemes where the ciphertexts are protected by some noise and lattices, the strength of ciphertexts in SHE scheme with integers comes

only from the noise. We demonstrated that we successfully eliminated the noise and recovered the secret key. Hence, we achieve a CCA-1 attack against vDGHV SHE scheme.

References

1. Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, Academic Press (1978) 169–177
2. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2) (1978) 120–126
3. Gentry, C.: Fully homomorphic encryption using ideal lattices. [13] 169–178
4. Gentry, C.: A Fully Homomorphic Encyrption Scheme. PhD thesis, Stanford University (2009)
5. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In Nguyen, P.Q., Pointcheval, D., eds.: Public Key Cryptography. Volume 6056 of Lecture Notes in Computer Science., Springer (2010) 420–443
6. Stehlé, D., Steinfeld, R.: Faster fully homomorphic encryption. In Abe, M., ed.: ASIACRYPT. Volume 6477 of Lecture Notes in Computer Science., Springer (2010) 377–394
7. Gentry, C., Halevi, S.: Implementing gentry's fully-homomorphic encryption scheme. In Paterson, K.G., ed.: EUROCRYPT. Volume 6632 of Lecture Notes in Computer Science., Springer (2011) 129–148
8. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In Gilbert, H., ed.: EUROCRYPT. Volume 6110 of Lecture Notes in Computer Science., Springer (2010) 24–43
9. Coron, J.S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: CRYPTO. (2011) 487–504
10. Coron, J.S., Naccache, D., Tibouchi, M.: Optimization of fully homomorphic encryption. Cryptology ePrint Archive, Report 2011/440 (2011) `http://eprint.iacr.org/`.
11. Chen, Y., Nguyen, P.Q.: Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. Cryptology ePrint Archive, Report 2011/436 (2011) `http://eprint.iacr.org/`.
12. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. [20] 505–524
13. Mitzenmacher, M., ed.: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. In Mitzenmacher, M., ed.: STOC, ACM (2009)
14. Lauter, K., Naehrig, M., Vaikuntanathan, V.: Can homomorphic encryption be practical? IACR Cryptology ePrint Archive **2011** (2011) 405
15. Loftus, J., May, A., Smart, N., Vercauteren, F.: On cca-secure fully homomorphic encryption. Cryptology ePrint Archive, Report 2010/560 (2010) `http://eprint.iacr.org/`.
16. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In Krawczyk, H., ed.: CRYPTO. Volume 1462 of Lecture Notes in Computer Science., Springer (1998) 26–45
17. Nymann, J.E.: On the probability that k positive integers are relatively prime ii. Journal of Number Theory **7**(4) (1975) 406–412

18. Shoup, V.: NTL - A Library for Doing Number Theory. (online) `http://www.shoup.net/ntl/index.html`.
19. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In Jr., B.S.K., ed.: CRYPTO. Volume 1294 of Lecture Notes in Computer Science., Springer (1997) 112–131
20. Rogaway, P., ed.: Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. In Rogaway, P., ed.: CRYPTO. Volume 6841 of Lecture Notes in Computer Science., Springer (2011)