Project 2: Cryptanalysis of integer based homomorphic cryptosystems

Computer Algebra

Zirui Yan r0916941

May 2023

Task 1

(a)If we split m up to its individual bits and encrypting these with the mothod of $\mathbb{Z}_N$, the addition and multiplication can only be satisfied for each bit. So we should use following method to encrypt messages in $\mathbb{Z}_N$:

Key Generation

- The secret key p $\in]N^{\eta-1}, N^{\eta}[$ is kept private by the user. And p should be coprime to N.

- The evaluation key is given by $x_0 = pq_0$ where q_0 is a uniformly random multiplier in $\mathbb{Z} \cap]0, N^{\gamma}/p[$.

Encryption To encrypt a message $m \in \mathbb{Z}_N$, sample a uniformly random noise term $r \in \mathbb{Z} \cap]-N^{\rho}, N^{\rho}[$ and a uniformly random multiplier $q \in \mathbb{Z} \cap [0, 2^{\gamma}/p[$, and return the ciphertext c=pq+Nr+m

Decrytion When p is odd, to decrypt a ciphertext c, compute the unique $c' \in]-p/2, p/2[$ such that $c' = c \bmod p$, and return the message $m = c' \bmod N$. When p is even, to decrypt a ciphertext c, compute the unique $c' \in]-p/2, p/2+1[$ such that $c' = c \bmod p$, and return the message $m = c' \bmod N$.

When we decrypt a ciphertext c, we first compute $c' = c \bmod p$, then compute the message $m = c' \bmod N$. In the first step, we cancel the pq term and keep $Nr + m$ unmodified. So p should larger than the absolute value of $Nr + m$. Such boundaries for p, r can make sure p is larger than the absolute value of $Nr + m$.

Disscussion of restriction on the private key p: if p is not coprime to N, then $c \bmod N$ leaks some information because in this case, $pq \bmod N$ is not uniformly random anymore. If p is coprime to N, $c \bmod N = pq + m \bmod N$. Since pq keeps its randomness, no information leaks.

(b) The following discussion is based on not considering addition and multiplication:

First, N is an integer larger or equal to 2. But bigger N can make the cryptosystem more secure at the expense of increased computation complexity.

If we only consider encryption and decryption without addition and multiplication operations on ciphertext, then we

need to make sure p/2 is larger than the absolute value of $Nr + m$, which will be explained later.

$$p/2 > max\{Nr + m\}$$
$$\frac{1}{2}N^{\eta-1} > N^{\rho+1} + N$$

And we have $\frac{1}{2}N^{\eta-1} > N^{\eta-2}$ and $N^{\rho+1} + N < N^{\rho+2}$. So $\underline{\eta \geq \rho + 4}$ since $N \geq 2$. (But most of time $\eta \geq \rho + 3$ is totally enough, especially when $N >> 2$.) In addition to this, η and ρ should be bigger enough to enable secret key p and random noise r having more value can be chosen, which guarantee the cryptosystem is secure.

We need p/2 is larger than the absolute value of $Nr + m$ because of our way of encryption and decryption. The idea of decryption is to cancel pq term of $c = pq + Nr + m$ first, then get plaintext from $c' = Nr + m$. Since random noise term $r \in \mathbb{Z} \cap]-N^{\rho}, N^{\rho}[$, it is necessary to have $c' \in]-p/2, p/2+1[$ in order to ensure $c' = Nr + m$. Otherwise, $c' = Nr + m = kp$ for some k.

$pq, x_0 \in \mathbb{Z} \cap]0, N^{\gamma}[$, which means that γ decides the maximum value of pq and x_0. As long as interval $\mathbb{Z} \cap]0, N^{\gamma}/p[$ contains 1, pq and x_0 are greater or equal to p, then they will not affect decryption, addition and multiplication. So $N^{\gamma} > p$ should always be satisfied, which gives us the restriction for γ: $\underline{\gamma \geq \eta - 1}$.

(c) Given 2 ciphertext c_1 and c_2 which are encrypted from m_1 and m_2 separately:$c_1 = pq_1 + Nr_1 + m_1$, $c_2 = pq_2 + Nr_2 + m_2$

We can compute the encryption c_3 of the sum of the 2 messages and the encryption c_4 of the product of the 2 messages by: $c_3 = c_1 + c_2 \ mod \ x_0$, $c_4 = c_1c_2 \ mod \ x_0$ => $c_3 = p(q_1 + q_2 - Cq_0) + N(r_1 + r_2) + (m_1 + m + 2)$, $c_4 = p(pq_1p_2 + Nq_1r_2 + Nq_2r_1 + q_1m_2 + q_2m_1 - Cq_0) + N(Nr_1r_2 + r_1m_2 + r_2m_1) + m_1m_2$

After one addition, noise is equal to the sum of previous noise. For one multiplication, since $Nr_1r_2 \gg r_1m_2 + r_2m_1$ if N is large enough, we can ignore the term r_1m_2 and the term r_2m_1. Therefore, the noise is approximately equal to the product of the previous noise multiplied by N after one multiplication.

As discussed in task1.(a), decryption will fail if p/2 is less than the absolute value of $Nr + m$. No matter how many multiplication we do, $r \gg m$, so we only need to make sure Nr is smaller enough than p. And it is easy to prove noise r of $\prod_{i=1}^{k} c_i \ mod \ x_0$ is approximately equal to $N^{k-1} \prod_{i=1}^{k} r_i$.

$p \in]N^{\eta-1}, N^{\eta}[$ and noise r of $\prod_{i=1}^{k} c_i \ mod \ x_0$ is approximately equal to $N^{k-1} \prod_{i=1}^{k} r_i$, so $\left|N^k \prod_{i=1}^{k} r_i\right| < \frac{1}{2}N^{\eta-1}$. Since $r \in \mathbb{Z} \cap]-N^{\rho}, N^{\rho}[$, $N^{(\rho+1)k} < \frac{1}{2}N^{\eta-1}$. And we have $\frac{1}{2}N^{\eta-1} > N^{\eta-2}$, so $k < \frac{\eta-2}{\rho+1}$ can make sure decryption not fail. In practice, it is almost impossible for all $|r_i|$ and p to have extreme value at the same time when N and k are large and $\frac{1}{2}N^{\eta-1} >> N^{\eta-2}$ when N is large, so $\underline{k \leq \frac{\eta-2}{\rho+1} \text{ is a conservative estimation}}$.

Task 2

(f) The table below is the numbers of successful decryption out of 1000 experiments with $\gamma = 14, \rho = 2, N = 13$ and $\eta = 5$ to 14.

η	$\frac{\eta-2}{\rho+1}$	k=2	k=3	k=4	k=5	k=6
9	7/3	1000	814	98	75	86
10	8/3	1000	999	152	101	90
11	3	1000	1000	412	77	72
12	10/3	1000	1000	928	107	71
13	11/3	1000	1000	1000	247	71
14	4	1000	1000	1000	617	100

The table below is the numbers of successful decryption out of 1000 experiments with $\gamma = 2000, \rho = 20, N = 100$ and different η.

η	$\frac{\eta-2}{\rho+1}$	k=2	k=3	k=4	k=5	k=6
63	2.90	1000	751	12	3	7
64	2.95	1000	1000	14	12	10
65	3	1000	1000	5	13	12
80	3.71	1000	1000	11	8	7

My bound behaves as I thought. The line eta=13 also illustrate my bound is a conservative estimation, especially when N and k become large. $k \leq \frac{\eta-2}{\rho+1} - 1$ is enough when N is large enough.

Task 3

(a) Van Dijk et al. [2] and Galbraith et al. [2] pointed out that the SDA algorithm and the OL attack for solving ACD problems have similar performances. So we can arbitrarily choose one of it. The core idea of SDA algorithm is to construct a lattice whose shortest vector is $(q_1, q_2, ..., q_t)$. The OL attack given by [1] is the second OL attack. The core idea of the second OL attack is to search the vector orthogonal to $(1, -\frac{r_1}{x^\rho}, ..., -\frac{r_n}{x^\rho})$. Once it found out, we have $\sum_{i=1}^{t} u_i q_i = 0$ and $\sum_{i=1}^{t} u_i r_i = v_0$.

(b) We can compute r_i's by c_i's and p. If r_i's are in the interval $]-2^\rho, 2^\rho]$, it is likely the p is correct.

(c) Not given η and ρ, it is hard to construct the lattice which can give us the right result with high probability and we cannot use the necessary condition for the algorithm to succeed anymore.

(d) I did at least 10 experiments for each parameter-combination with LLL reduction algorithm. "Succeed" means that attack succeeds every time. "Fail" means at least one attack fails.

η	ρ	γ	$\frac{\gamma-\eta}{\eta-\rho}$	t=10	t=20	t=30	t=50	t=60	t=100
100	20	2000	23.75	fail	fail	succeed	succeed	succeed	succeed
100	80	1000	45	fail	fail	fail	fail	succeed	succeed

$\frac{\gamma-\eta}{\eta-\rho}$ is a necessary condition given in [1]. Since it ignores some constant terms, the actual minimal number is higher than that.

If I implement it with BKZ_6, the running time is much more longer than the time with LLL. And it is impossible to use BKZ when t=100, because the running time error and stability problems of BKZ. During my experiments, LLL and BKZ succeed and fail at the same time. But when they fail, sometimes the numbers are the same, but sometimes not. So LLL and BKZ sometimes don't give the same reduction.

η	ρ	γ	$\frac{\gamma-\eta}{\eta-\rho}$	average running time with t=50
100	20	1000	11.25	2.613
100	20	1500	17.5	5.406
100	20	2000	23.75	8.703
100	40	1000	15	2.613
100	60	1000	22.5	2.664
200	20	1000	4.44	2.343
400	20	1000	1.58	1.709

Runtime grows as γ grows, slightly grows as ρ grows, but declines as η grows. This is because γ and ρ affects the value of c_i, which decides the time for LLL with n and t.

Task 4

(a) We can divide the decryption into 3 steps: $c' = c \ mod \ p_3$, $c' = \frac{x}{y} \ mod p_3$ and $m = \frac{x}{y} \ mod \ p_1$.

We need $c' = c \ mod \ p_3$ because after adition and multiplication, we've added a multiple of x_0. And we can get rid of it by using $mod \ p_3$.

$c' = \frac{x}{y} \ mod p_3$ gives back x,y which we've gotten during encrytion. We need to use the algorithm in slide 20 again here. the x,y during decryption must be the x,y during encryption. Denote L the lattice used in encryption and L' the lattice used in decryption. L' contains all vectors (a,b) s.t. $a - bc' = kp_3$, so L' contains (x,y) we want. When $\lambda_1(L) \leq \sqrt{2}vol(L)^{1/2} \leq \sqrt{\frac{2}{2\pi e}}vol(L')^{1/2} \approx \lambda_2(L')$, the shortest vector in L must be the shortest vector in L' (since it exists in L' and shorter than the λ_2). So $2\pi e \cdot vol(L) \leq vol(L')$ => $\underline{2\pi eN \leq p_3}$ => $2\pi e \cdot 2^{2\lambda} \leq 2^{\alpha\lambda-1}$ => $\underline{\alpha \geq \frac{2\lambda+6}{\lambda}}$ (This is a relatively tight bound. $\alpha \geq \frac{2\lambda+5}{\lambda}$ should also be enough in most cases.) And the decryption might fail when $x = 0$, but it is almost impossible for x to be 0 in practice.

Then obviously $m = \frac{x}{y} \ mod \ p_1$ can give us the plaintext because of CRT and property of mod.

On the other hand, we need to be able to find the shortest vector in L', which gives us $2\pi N \leq p_3$. But this restriction is redundant since we've already have the restriction above.

(b) Suppose $c_1 = \frac{x_1}{y_1} - k_1 p_3$ and $c_2 = \frac{x_2}{y_2} - k_2 p_3$, then the encryption of the sum of the two messages $c_3 = \frac{x_1}{y_1} + \frac{x_2}{y_2} - (k_1 + k_2)p_3 - k_3' N p_3$, the encryption of the product of the two messages $c_4 = \frac{x_1 x_2}{y_1 y_2} - k_1 \frac{x_2}{y_2} p_3 - k_2 \frac{x_1}{y_1} p_3 + k_1 k_2 p_3^2 - k_4' N p_3$.

We can conclude that $\sum_{i=1}^{k} c_i \ mod x_0 = \sum_{i=1}^{k} \frac{x_i}{y_i} + P_3(p_3)p_3$ and $\Pi_{i=1}^{k} c_i \ mod \ x_0 = \Pi_{i=1}^{k} \frac{x_i}{y_i} + P_4(p_3)p_3$ for some polynomials $P_3(\cdot)$, $P_4(\cdot)$. $c' = c \ mod \ p_3$ can cancel out $P_3(p_3)p_3$ and $P_4(p_3)p_3$. But it needs $\sum_{i=1}^{k} \frac{x_i}{y_i}$ and $\Pi_{i=1}^{k} \frac{x_i}{y_i}$ to satisfy $\|(x, y)\| \leq \sqrt{N}$ otherwise it cannot find the (x,y) we need.

But it is impossible, which can be shown by some simple calculation. So I prefer DGHV. And my implementation verifies what I thought.

Task 6

(a) On one hand, evaluation key $x_0 = p_1p_2p_3$. Since we've known p_3, we can compute p_1p_2. On the other hand, $c' = \frac{x}{y} \ mod \ p_3$, then we get x, y. Since $m = \frac{x}{y} \ mod \ p_3$, $x - my = kp_1$ for some k. Then, $gcd(kp_1, p_1p_2) = p_1$ unless k is a multiple of p_2. In most cases, we get p_1 and p_2 easily from a pair of plaintext/ciphertext.

(b) $c_i = k_ip_3 + \frac{x_i}{y_I}$ for some k_i

(c) No. If we treat $\frac{x}{y}$ as a number, we are not able to find a ρ to limit the size of it and it can be very large. $\frac{x}{y}$ is not the part can be neglect.

(d) This is an algorithm which only use one ciphertext c: First, we have $c = \frac{x}{y} mod p_3$ => $yc = kp_3 + x$ for some k. Multiply p_1p_2 on both sides, we get $yp_1p_2c = kx_0 + xp_1p_2$ => $yp_1p_2c = xp_1p_2 \ mod \ x_0$. Since $c < p_3 < x_0$, we can use a 2-d lattice reduction to compute yp_1p_2, xp_1p_2. p_3 is a large prime number, so $gcd(xp_1p_2, x_0) = p_1p_2$ with very high probability. Then $p_3 = x_0/p_1p_2$.

But when using 2-d lattice reduction, we need $\|(xp_1p_2, yp_1p_2)\| < \sqrt{x_0}$ to make sure the shortest vector (xp_1p_2, yp_1p_2). This gives the restriction $\alpha \geq \frac{3\lambda+3}{\lambda}$. And this algorithm spends most time on doing a 2-d LLL reduction, so the complexity should be $O(log^3 B) \approx O(\alpha^3\lambda^3)$

In my implementation, I only use the first element of cs. And this algorithm fails when $\alpha = 2, 3$. The tableau below shows the running time with $\lambda = 256, \alpha = 4, 5, 6$.

	experiment 1	experiment 2	experiment 3	experiment 4	experiment 5	average running time
$\lambda = 256 \quad \alpha = 4$	28.906	78.891	47.890	62.781	59.641	55.622
$\lambda = 256 \quad \alpha = 5$	44.860	65.891	43.953	67.547	49.281	54.306
$\lambda = 256 \quad \alpha = 6$	322.312	278.857	308.031	351.265	359.283	323.95

Task 7

(a) **Step1**: set $c_2 = 1$. Do $c_2 = 2 * c_2$ until $D(c_2) = 1$. **Step2**: let $c_1 = c_2/2$ (c_2 be a large number, so c_1 must be even) and c_3 be the nearest even number of $(c_1 + c_2)/2$. If $D(c_3) = 1$, then $c_2 = c_3$. If not, $c_1 = c_3$. Repeat this step until $c_1 + 2 = c_2$. **Step3**: $c_4 = (c1 + c2)/2$. If $c_4 = 1$, then $p = 2c_2 - 1$. If not, $p = 2c_1 + 1$. **The math behind**: in step1, we find an even number c_2 between p/2 and p because $D(c) = c \ mod \ 2 = 0$ when $c < p/2$ and $D(c) = (c - p) \ mod \ 2 = c \ mod \ 2 - p \ mod \ 2 = 1$ when $c < p/2$. Then find p by bisection.

(b) My attack succeeds on all given parameter set.

(c) Decryption oracle can give attacker some information, even if the input ciphertext c can never be generated by its

encryption function. So we can check if c is within the boundary and if $c' \in \mathbb{Z} \cap]-2^\rho, 2^\rho + 1[$. If not, give random output.

References

[1] Galbraith, S. D., Gebregiyorgis, S. W., and Murphy, S. Algorithms for the approximate common divisor problem. *LMS Journal of Computation and Mathematics 19*, A (2016), 58–72.

[2] Van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. Fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29* (2010), Springer, pp. 24–43.