

列印成品

2024年4月9日 上午 11:06

Introduction to Cryptography, Spring 2024

Homework 4

Due: 4/19/2024 (Friday)

Notes:

- (1) For Part A, submit a “hardcopy” right after the class on the due day.
- (2) TAs will run plagiarism check on your submitted programs. Write your own code and do not copy from others or anywhere.

Part A: Written Problems

1. Compute the period of the linear congruential generator $X_{n+1} = 3X_n + 2 \pmod{23}$ with various initial value X_0
2. Compute the 16 output bits of the LFSR $B_2X^2 + B_0X^4$ with initial values $B_3B_2B_1B_0 = 1010$ and 1011 . What are their periods?
3. Suppose you have an entropy source that produces independent bits, where bit 1 is generated with probability $0.5+p$ and bit 0 is generated with probability $0.5-p$, where $0 < p < 0.5$. Consider the conditioning algorithm that examines the output bit stream as a sequence of non-overlapping pairs. Discard all 00 and 11 pairs. Replace each 01 pair with 0 and each 10 pair with 1.
 - a. What is the probability of occurrences of each pair in the original sequence?
 - b. What is the distribution of occurrences of bits 0 and 1 in the modified sequence?
 - c. What is the expected number of input bits in order to generate an output bit
4. Consider the RSA encryption system. Let $n = 29 \times 43 = 1247$ and $e = 17$.
 - a. What is the private key d ?
 - b. What is the plaintext of ciphertext $C=1123$?
5. Consider RSA encryption with $n=136127$. Assume that Alice has key pair $PU_A = (17, n)$ and $PR_A = (79663, n)$. Alice knows that Bob uses the same n to set up his key pair and $PU_B = (31, n)$. Alice intercepts a ciphertext $C=3761$ which is sent to Bob by Carol. Show that Alice can decrypt C without factoring n .

Part B: Programming Problem

1. This programming problem is to practice RSA encoding and decoding using Crypto++. We only deal with one-block operation without padding, that is, plaintext and ciphertext are both less than

the RSA modulus n . You need to check whether the message length (in bits) is strictly shorter than modulus n 's length.

2. Data format

- For encryption: the input is a line:

enc 64 B14022EEF719F1BB 11 Alice

where **enc** indicates encryption, **64** (decimal) is the modulus length,

B14022EEF719F1BB (Hex) is the modulus n , **11** (Hex) is the encryption exponent e , and **Alice** (ASCII) is the message. Note that the message, consisting of all symbols after the 4th parameter till the end of line, may contain spaces, such as, **Alice is my friend**.

The ASCII message is treated as an integer, for example, Hi (ASCII) = 4869 (Hex) = 18537 (decimal). The output is a line of the ciphertext in Hex, such as,

73DC304C7BF6A0FD and has [64/4] hex symbols, that is, adding leading zeros for small ciphertext values, such as **000A3B9F2359BBE3**.

- For decryption: the input is a line:

dec 64 9D001E6473DFACF9 16282B21A7866BF5 154C638CD3615216

which indicates decryption, modulus length (decimal), the modulus (Hex), the decryption exponent (Hex) and the ciphertext (Hex). The output is a line of the plaintext in ASCII, such as, **Secret**.

3. Submission

- Submit your program to the online judge system before 12:01pm, 4/19 (Friday)
- Your program reads in multiple lines of the above data format from stdin and outputs the results in separate lines of the specified format to stdout.

4. On-site test

- Computer room EC324, 5:30-9:30pm, 4/19 (Fri)
- Due to computer room constraint, TA's will ask you to sign up your preferred time slot in advance. There are three time slots: 17:30-19:30pm, 18:30-20:30pm and 19:30-21:30pm. You need to finish the test within two hours.

5. If you want to generate some RSA keys for practice, try the following program segment:

```
// random number generator
AutoSeededRandomPool rng;
InvertibleRSAFunction parameters;

// Generate RSA keys with key_length bits
int key_length = 256;
parameters.GenerateRandomWithKeySize(rng, key_length);

const Integer& n = parameters.GetModulus();
const Integer& p = parameters.GetPrime1();
const Integer& q = parameters.GetPrime2();
const Integer& d = parameters.GetPrivateExponent();
const Integer& e = parameters.GetPublicExponent();
```

1. $X_{n+1} = 3X_n + 2 \pmod{23}$

$x_0 = 0 \Rightarrow \text{generated sequence : } 0, 2, 8, 3, 11, 12, 15, 1, 5, 17, 7, 0 \dots$

1. $x_{n+1} = 5x_n \bmod 2^5$
- $x_0 = 0 \Rightarrow$ generated sequence: 0, 2, 8, 3, 11, 12, 15, 1, 5, 17, 7, 0...
- \Rightarrow period = 11
- $\Rightarrow x_0 = 0, 1, 2, 3, 5, 7, 8, 11, 12, 15, 17,$ period = 11
- $x_0 = 6 \Rightarrow$ generated sequence: 6, 20, 16, 4, 14, 21, 19, 13, 18, 10, 9, 6,...
- $\Rightarrow x_0 = 4, 6, 9, 10, 13, 14, 16, 18, 19, 20, 21,$ period = 11
- $x_0 = 22 \Rightarrow$ period = 1

2. $B_3 \ B_2 \ B_1 \ B_0$ feedback $B_2 \oplus B_0$

1	0	1	0
0	1	0	1
0	0	1	0
0	0	0	1
0	1	0	0
1	0	1	0

\Rightarrow output sequence: 010100 010100 ...

1	0	1	1
1	1	0	1
0	1	1	0
1	0	1	1

\Rightarrow output sequence: 110 110 ...

3. Let B_i be the random variable for the i th output bit of the entropy source, $i \geq 0$

$$(a) \Pr[B_{2i} = b, B_{2i+1} = b'] = \Pr[B_{2i} = b] \cdot \Pr[B_{2i+1} = b']$$

for $i \geq 0$, $b, b' \in \{0, 1\}$ since B_{2i}, B_{2i+1} are independent

$$\text{Thus, } \Pr[B_{2i} = 0, B_{2i+1} = 0] = (\frac{1}{2} - p)(\frac{1}{2} - p)$$

$$\Pr[B_{2i} = 1, B_{2i+1} = 1] = \Pr[B_{2i} = 1, B_{2i+1} = 0] = (\frac{1}{2} + p)(\frac{1}{2} - p)$$

Thus, $\Pr[B_{2i}=0, B_{2i+1}=0] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$

$$\Pr[B_{2i}=0, B_{2i+1}=1] = \Pr[B_{2i}=1, B_{2i+1}=0] = \left(\frac{1}{2} + p\right) \left(\frac{1}{2} - p\right)$$

$$\Pr[B_{2i}=1, B_{2i+1}=1] = \left(\frac{1}{2} + p\right) \cdot \left(\frac{1}{2} + p\right)$$

(b) Let G be a generated bit.

$$\begin{aligned}\Pr[G=0] &= \Pr[B_{2i}=0, B_{2i+1}=1 \mid B_{2i}, B_{2i+1} \text{ is not discarded}] \\ &= \left(\frac{1}{2} + p\right) \left(\frac{1}{2} - p\right) / \left(1 - \left(\frac{1}{2} + p\right)^2 - \left(\frac{1}{2} - p\right)^2\right) \\ &= \frac{1}{2}\end{aligned}$$

Thus, the output distribution is $\Pr[G=0] = \frac{1}{2}$, $\Pr[G=1] = \frac{1}{2}$

(c) The prob. of a bit being generated is

$$\Pr[B_{2i}=0, B_{2i+1}=1, \text{ or }, B_{2i}=1, B_{2i+1}=0] = \frac{1}{2} - 2p^2.$$

Let X be the number of Bernoulli experiments until the first success.

X is a geometric distribution with success prob. $= \frac{1}{2} - 2p^2$.

$$\text{Thus, } E(X) = 1 / (\frac{1}{2} - 2p^2).$$

The expected number of entropy source bits for generating a truly random bit is $2X \frac{1}{\frac{1}{2}-2p^2} = \frac{4}{1-4p^2}$

4. $d = 761$

$$M = 1123^{761} \bmod 1247 = 1104$$

5. $(7 \times 79553) - 1 = 1352400 = k \cdot \phi(n)$

Equivalent PR' $B = (1134271, n)$

$$M = 3761^{1134271} \bmod 136127$$

$$= 33745$$

check: $(2^{79553})^{31} \bmod 126127 = 3761$

$$= 33745$$

$$\text{check: } (33745)^{\leftarrow 31} \bmod 136129 = 3761$$