COMPSCI 4CR3 Practice Problems

Dev Mody

December 2024

Contents

1 Question 1 **2**

2 Question 2 **2**

3 Question 3 **3**

4 Question 4 **3**

5 Question 5 **4**

6 Question 6 **4**

7 Question 7 **4**

8 Question 8 **5**

9 Question 9 **5**

10 Question 10 **5**

11 Question 11 **6**

12 Question 12 **6**

13 Question 13 **6**

14 Question 14 **7**

15 Question 15 **7**

16 Question 16 **8**

17 Question 17 **8**

18 Question 18 **9**

19 Question 19 **9**

19.1 a) 9

19.2 b) 9

20 Question 20 **10**

20.1 a) $h(x) = 17, k_E = 25$ 10

20.2 b) $h(x) = 2, k_E = 13$ 10

1 Question 1

Let $s_5 = a_5 s_5 + \cdots + a_1 s_1$ represent a linear feedback shift register with coefficients $a_5 a_4 a_3 a_2 a_1 = 11001$ and initial seed set to $s_1 s_2 s_3 s_4 s_5 = 01011$. Compute the next five bits generated by this LFSR.

From my LFSR implementation in Python, I set the coefficients to the array $(1, 1, 0, 0, 1)$ and the initial seed to the array $(0, 1, 0, 1, 1)$. When I passed these values into the `__init__` function, I asked the LFSR to generate the next five bits by asking to return `generate_key_stream(10)`. From this, I got the next 5 bits of the key to be $(1, 0, 0, 1, 0) = 10010$.

2 Question 2

As discussed in class, a pseudorandom number generator (PRNG) is considered secure if it is computationally hard to distinguish its output from a uniformly random sequence. Explain why a PRNG based on a linear feedback shift register (LFSR) is not secure.

A pseudorandom number generator (PRNG) based on a Linear Feedback Shift Register (LFSR) is not secure because the outputs are deterministic making them predictable. Thus, they can be efficiently distinguished from a uniformly random sequence. Let's discuss what aspects of the LFSR lead to this conclusion. First of all, LFSRs are linear since the feedback function is an XOR of the previous state. A massive contributor to the output of an LFSR is from the initial seed and the feedback coefficients. Known plaintext attacks exist on the LFSR which show how if someone has access to some portion of the plaintext, they can use the fact that the plaintext was XORed with the key to get the ciphertext. Since the ciphertext and plaintext are known, the attacker could simply XOR the known bits to get the known key bits. From here, the attacker's goal would be to obtain the initial seed and the feedback coefficients. Since LFSRs are linear, the output streams could be modelled as a System of Linear Equations which could be solved by Gaussian Elimination. Another big disadvantage is that given the degree of the LFSR n, the attacker now knows the period in which outputs repeat which would be at most $2^n - 1$.

3 Question 3

Let $g : \mathbb{Z}_p \to \mathbb{Z}_p \times \mathbb{Z}_p$ be a PRNG defined by $g(x) = (x^3 \mod p, x^5 \mod p)$. Is this PRNG secure? Explain. (You must investigate a sequence of pairs $\{g(x_i)\}$ can be distinguished from a sequence $\{(a_i, b_i)\}$ where a_i, b_i are uniformly random numbers.)

To determine if the PRNG is secure, we must determine if different values of $g(x_i)$ can be distinguished from a sequence $\{(a_i, b_i)\}$ where a_i and b_i are uniformly random numbers. First of all, we immediately can find out that the two values in the pair of the PRNG are dependent to each other. That is, the pair $(x^3 \mod p, x^5 \mod p)$ are dependent on x. If an attacker has access to $x^3 \mod p$, all they need to do is simply express $x^5 \mod p \equiv (x \cdot x^4) \mod p \equiv (x \cdot x \cdot x^3) \mod p$. Since fast algorithms to compute modular exponentations, this attack isn't the most computationally expensive for large modulus values.

4 Question 4

Which of the following statements are true regarding AES?

1. It is a bit-oriented cipher. Since each of the MixColumn operations occur on Extensive Galois Fields $GF(2^m)$ and byte-wise XORS are performed for adding the round keys before and after the augmentation, this is definitely false.

2. It has fast software implementations. Since AES has more of a software-friendly design, straightforward and fast implementations exist which leverage parallelism. True.

3. The Diffusion Layer is not invertible. First of all, since decryption is possible by inverting all the steps of encryption, this is immediately false. In addition, the diffusion layers are SubBytes, ShiftRows and MixColumns. ByteSubstitution can be reversed because two matrices can be defined, one for the encryption process and the other for the decryption. ShiftRows can also be inverted because whichever rows are shifted to the right for encryption can be shifted to the left for decryption purposes. Finally, MixColumns can be inverted because an invertible matrix can be defined for the computation where the inverse is used for the decryption. As a result, the diffusion layer is invertible. Thus, this statement is false.

4. The number of rounds depends on the key size. Since 10 rounds are assigned to 128 bit key sizes, 12 rounds are assigned to 192 bit key sizes and 14 are assigned to 256 bit key sizes, this statement is true.

5. The arithmetic in AES, such as multiplication, is integer arithmetic. The arithmetic in AES is done in Extensive Fields $GF(2^m)$ instead of pure integer arithmetic. Thus, this statement is false.

6. There are no known subexponential attacks against AES. True. There are no known subexponential attacks against AES. The best-known attack on AES (related to brute-force attacks) is a key search that requires exhaustive checking of all possible keys, which takes exponential time (in the worst case).

5 Question 5

Let $u \in \mathbb{Z}_p^n$ be a fixed vector of length n with entries in $\mathbb{Z}_p$. Define a hash function $f_u : \mathbb{Z}_p^n \to \mathbb{Z}_p$ by $f_u(v) = u_1v_1 + \cdots + u_nv_n \mod p$. Is this hash function collision resistant? Explain.

To determine if f_u is collision-resistant, as in Strong Collision Resistant, we must determine if it is computationally infeasible to find two values $x_1, x_2 \in \mathbb{Z}_p^n$ such that $f_u(x_1) = f_u(x_2)$. Since u is a fixed value, it is possible to find an input vector x_1 such that $f_u(x_1) \equiv 0 \mod p$ as the dot product could add up to be a multiple of p. Since p has not been specified to be a large prime, I think it is safe to assume that p could be small and that finding such a vector is not difficult. Also, it is possible to choose vector $x_2 = 0$ such that $f_u(x_2) = 0$ as well. As a result, since we found two different x_1 and x_2 such that $f_u(x_1) = f_u(x_2)$, we have shown that f_u is not Strong Collision Resistant.

6 Question 6

Let h_1 and h_2 be collision-resistant hash functions. Let $g(x) = h_1(h_2(x))$. Is g collision resistant? Explain.

If h_1 and h_2 are collision-resistant it is computationally infeasible to find x_1 and x_2 such that $h_1(x_1) = h_1(x_2)$ and $h_2(x_1) = h_2(x_2)$. For $g(x) = h_1(h_2(x))$, since h_2 is collision-resistant, we know that the hash outputs of h_2 are unique. Since h_1 is also collision-resistant, none of the hash values will be the same for these unique values. As a result, this is equivalent to saying that for any two different $x_1 \neq x_2$, $g(x_1) = g(x_2) \equiv h_1(h_2(x_1)) = h_1(h_2(x_2))$. As a result, g is collision resistant.

7 Question 7

Suppose $h : \{0,1\}^* \to \{0,1\}^n$ is preimage resistant with $n = 160$, so the output of h is 20 bytes. We are looking for an input x such that the third 32 bits of $h(x)$ are 11100011100100011001011100110011. In other words, we seek out x such that $h(x) = y$ where $y = ab11100011100100011001011100110011cd$ and $a, b, c, d \in \{0,1\}^32$. How many inputs, in the worst case, would it take to find such an x?

Since h is a preimage resistant hash function, we know that it is computationally infeasible to find such an x from $h(x) = y$ via $x = h^{-1}(y)$. Thus, in order to find such an x resulting in the y from the above form, we must perform an exhaustive search on the inputs to find such a y. In this case, since the third 32 bits of the output are fixed, we need to perform a search on the remaining $160 - 32 = 128$ bits. The number of keys we would have to search through would be 2^128.

8 Question 8

Let p be a large prime such that 7 does not divide $p-1$. Define function $f : \mathbb{Z}_p \to \mathbb{Z}_p$ by $f(x) = x^7 \mod p$. Is f a one-way function? Explain.

To determine if f is a one-way function, we must determine if it is computationally easy to compute $f(x)$ for any x and if it is computationally infeasible to compute $f^{-1}(y)$. Using standard modular exponentiation algorithms like Square-and-Multiply, it is quite easy to compute $f(x) = x^7 \mod p$ for any x. However, finding such an x such that $x^7 \equiv y \mod p$ is equivalent to the Discrete Logarithm Problem. Another important add-on is that 7 does not divide $p-1$. As a result, we know that no subgroup exists of order 7 in $\mathbb{Z}_p$. Thus, this makes the inversion equivalent to the Discrete Logarithm Problem, which we know already is computationally infeasible to solve for large prime p. As a result, we know it is computationally infeasible to solve $x = f^{-1}(y)$. Therefore, f is a one-way function.

9 Question 9

Let $r_0 = 112$ and $r_1 = 86$. Use the Extended Euclidean Algorithm to find s and t such that $sr_o + tr_1 = gcd(r_0, r_1)$.

Using my code in Python, I have determined the two integers s and t such that $112s + 86r = \gcd(112, 86) = 2$ are $s = 10, r = -13$.

10 Question 10

Which of the following is not a generator for $\mathbb{Z}_{953}^{\times}$?

1. 602. Primitive.
2. 746. Primitive.
3. 780. Not a Primitive.
4. 94. Primitive.

11 Question 11

Let G be a finite cyclic group of size N and let $g \in G$ be a generator. Suppose Trudy has access to an oracle that can solve the computational Diffie-Hellman Problem. For any $1 \leq x, y < N$, given g^x and g^y, Trudy can efficiently compute g^{xy} by accessing the oracle. Can Trudy solve the following problem efficiently?: Given g^x and $n \in O(log(N))$, compute $g^{x^n+3x^2+x+5}$.

To see if Trudy can solve the problem efficiently, we can break $g^{x^n+3x^2+x+5}$ into $g^{x^n} \times g^{3x^2} \times g^x \times g^5$. In this case, g^5 is efficient to compute because we're raising generator g to a fixed constant and g^x is already given. Since $n \in O(\log N)$ we know that n is going to be smaller than $\log N$. For these values of n, we can compute g^{x^n} by referring to the oracle for $g^{x \times x \times \cdots \times x}$. The same situation can be performed for computing g^{3x^2}. As a result, it is possible for Trudy to efficiently solve this problem.

12 Question 12

Let $p = 953$ be a prime number. In the Elgamal Signature Scheme, Trudy has decided to forge a signature by writing a brute-force algorithm that tries all possible ephemeral keys. How many trials ephemeral keys does he have to try?

1. Less than 219
2. Less than 384. Right answer.
3. More than 410
4. More than 412

13 Question 13

Let $(n, e) = (493, 205)$ be the public key in the RSA Signature Scheme. Which of the following is a valid message-signature pair?

1. (32, 16)
2. (6, 415). Valid.
3. (53, 83)
4. (112, 45)

I use my code to solve this. We essentially need to loop through all pairs and perform $s^e \mod n = x$.

14 Question 14

Let S = (Gen, Sig, Ver) be a signature scheme that is existentially unforgeable. Recall that in an existential forgery attack, the attacker constructs a new message-signature (m, s) where m has never been previously signed by the legitimate signer. Consider the following new signature scheme S' = (Gen, Sig', Ver') based on S where Sig' and Ver' are defined as follows:

- Sig'(m, k_{pr}): Choose a random $r = \{0,1\}^n$, output = $(r, Sig(k_{pr}, m \oplus r), Sig(k_{pr}, r))$
- $Ver'(m, k_{pub}, (r, s_1, s_2)$: Output "accept" iff $Ver(k_{pub}, m \oplus r, s_1) = Ver(k_{pub}, r, s_2)$

Is S' a secure signature scheme? If so, justify your answer; otherwise give an attack.

Idk how to answer this

15 Question 15

Briefly explain why the Discrete Logarithm Problem (DLP) over the group of points on an elliptic curve is more widely used than the DLP over the cyclic group $\mathbb{Z}_p^\times$.

The reason why people tend to use the DLP over the group of points on an elliptic curve (ECDLP) rather than normal DLP is because of the good one-way characteristics. If the elliptic curve is chosen with care for ECDHP, the best known attack against ECDLP are considered weaker than the best algorithms for solving the DLP and the best factoring algorithms which are used for RSA attacks. This is because ECC implementations of DLP are lightweight in the sense that the same security of systems using RSA or DLP encryption with over 1000 bits can be replicated with 256 bits of ECC. The computational complexity of ECC is cubic in the bit length of the prime used to define it. Thus increasing the bit length of the prime significantly increases the complexity of its algorithms as the domain of generated points increase.

16 Question 16

In the DSA Signature Scheme, the public values are (p, q, α) where α is the generator of a subgroup of $\mathbb{Z}_p^\times$ of order q. Is it safe to fix these public values once and for all, so that everyone in the world uses them? Explain.

We know from the DSA Signature Scheme that p is a large prime modulus, q is a divisor of $p-1$, ensuring a subgroup of order q and α is a generator for subgroup of order q in $\mathbb{Z}_p^\times$. If these values were to be fixed, this system would be open to many forms of attacks because they would have the ability to perform computations on the same set of values, allowing them as much time they need to solve this problem with the attacks of their choice. For instance, precomputation attacks can be performed, the subgroup property could be exploited to perform a DLP attack or a generic DLP attack could be performed. This also increases the chances of the ephemeral key to be reused making an attack on the same ephemeral key be analogous to the attack on the Elgamal Signature Scheme.

17 Question 17

Suppose the public primes for the DSA Signature Scheme are $p = 2089$ and $q = 29$. Which of the following is a public key β for these parameters?

1. 774
2. 1762
3. 1189
4. 512. Right Answer.

I did this using the code on the GitHub Repo. I basically looped through all generators of subgroups with order 29, and all the possible $0 < d < 29$. For each pair (α, d), I computed $\beta \equiv \alpha^d \mod p$. The only β I found was $\beta = 512$.

18 Question 18

Let $h : \{0,1\}^* \to \{0,1\}^{100}$ be a preimage resistant hash function. Suppose you have a machine M that can compute 2^{30} hashes per second. Using M, approximately how long does it take to find a collision with probability at least $1/2$ for h?

1. 35 million years
2. 15 days
3. 1 year
4. 10 years

I used the formula used for the Birthday Paradox in Chapter 11 for this question. There, the probability λ of at least one collision occurring was represented by: $\lambda \approx 2^{(n+1)/2} \times \sqrt{\ln(1/(1-\lambda))}$ where t represented the number of hashed messaged needed for a collision. In this case, since our output length is $n = 100$ and we need a probability of at least 50%, we could plug $n = 100$ and $\lambda = 0.5$ into the formula and solve for t. In this case $t = 1325645834665744.8$. However, we need to convert this number into a time. Since we know that the machine M computes 2^{30} hashes per second, we can divide t by this number to get the number of seconds required. This number came out to be just 14 days. To make sure we have at least a probability of 50% we can say that machine M needs at least 15 days.

19 Question 19

Consider the curve E: $y^2 = x^3 + x + 2$ over $\mathbb{Z}_{11}$

19.1 a)

Is this an elliptic curve?
We must check the condition from the textbook in terms of a and b where $4 \times a^3 + 27 \times b^2 \neq 0 \mod p$. In our case, $a = 1, b = 2$. Since $4 + 27 \times 4 \mod 11 = 112 \mod 11 \neq 0$, this is definitely an Elliptic Curve.

19.2 b)

The order of the group of points on E is 16. What is the order of the point $P = (8, 4) \in E$?
I computed the order via the code I have on the GitHub. It came out to be 3.

20 Question 20

The parameters of a DSA scheme are given by $p = 59, q = 29, \alpha = 3$ and Bob's private key is $d = 23$. Show the process of signing (by Bob) and verification (by Alice) for the following hashed messages $h(x)$ and ephemeral keys k_E.

20.1 a) $h(x) = 17, k_E = 25$

Signing Procedure: Since we have $p = 59, q = 29, \alpha = 3$ and private key $d = 23$, we must first compute $\beta \equiv \alpha^d \mod p$, which gives us $\beta = 45$. Next, we move to the signing procedure where since we're given $k_E = 25$ and $h(x) = 17$, we can compute $r \equiv (\alpha^{k_E} \mod p) \mod q \equiv 22$ and $s \equiv (h(x) + d \times r) \times k_E^{-1} \mod q \equiv 26$. Thus our signed message would be $(17, (22, 26))$.

Verification Procedure: We compute $w \equiv s^{-1} \mod q$, $u_1 \equiv (w \times h_x) \mod q$, $u_2 \equiv (w \times r) \mod q$ and $v \equiv (\alpha^{u_1} \times \beta^{u_2} \mod p) \mod q$. For my calculations, I got $w = 25, u_1 = 19, u_2 = 28, v = 22$. We then compute $v \equiv r \mod q$ and since $22 \mod 29 = 22 = r$, the signature is valid.

20.2 b) $h(x) = 2, k_E = 13$

I followed a similar procedure to Part A with the different $h(x)$ and k_E values. I got $\beta = 45, r = 25, s = 2, w = 15, u_1 = 1, u_2 = 27, v = 25$. In this case since $25 \mod 29 = 25 = r$, the signature is valid.