Spring 2024 Cryptography and Network Security

Homework 1

Release Date: 2024/03/05

Due Date: 2024/04/01, 23:59

TA's Email: `cns@csie.ntu.edu.tw`

Instructions

- This homework set is worth 110 points, including 10 bonus points.
- **Submission Guide:** Please submit all your codes and report to NTU COOL. Please refer to the homework instructions slides for information.
- You may encounter new concepts that haven't been taught in class, and thus you're encouraged to discuss with your classmates, search online, ask TAs, etc. However, **you must write your own answer and code**. Violation of this policy leads to serious consequences.
- You may need to write programs in the Capture The Flag (CTF) problems. Since you can use any programming language you prefer, we will use a pseudo extension code**.ext** (e.g., code.py, code.c) when referring to the file name in the problem descriptions.
- In each of the Capture The Flag (CTF) problems, you need to find out a flag, which is in `CNS{...}` format, to prove that you have succeeded in solving the problem.
- Besides the flag, you also need to submit the code you used and a short write-up in the report to get full points. The code should be named **code{problem_number}.ext**. For example, code3.py.
- In some CTF problems, your solution may involve human-laboring or online tools. These are allowed as long as you get the flag not by cheating or plagiarism. If your code does not directly output the flag (e.g. it requires the user to do manual filtering on some messages), please specify the execution process of your code in **readme.txt** file.
- In some CTF problems, you need to connect to a given service to get the flag. These services only allow connections from 140.112.0.0/16, 140.118.0.0/16, and 140.122.0.0/16.

Handwriting

1. CIA (10%)

Please explain three major security requirements: confidentiality, integrity, and availability. For each security requirement, please give an example in the real world.

2. Hash Function (10%)

Please explain three properties of a cryptographic hash function: one-wayness, weak collision resistance, and strong collision resistance. For each property, please give an example applied in the real world.

3. Asymmetric Cryptography (20%)

We had introduced RSA and ECDSA in the lecture on asymmetric cryptography, the latter depends on elliptic curves, however, we did not dig too deep into what an elliptic curve is, in the first part, we will introduce the point operations on an elliptic curve, and discuss the discrete log problem on elliptic curves. In the second part, we will look into the correctness of those two schemes.

3.1. An introduction to elliptic curves in cryptography

Given an elliptic curve $y^2 = x^3 + ax + b$ over the field $\mathbb{F}_p$, the set of integral points together with a point at "infinity" denoted by O, which serves as the identity element, with a binary operator "+" forms a group, where the binary operator is defined as follows:

Given a point $P = (x_P, y_P)$ on the curve, we define the point $-P$ as the point $(x_P, -y_P)$, it is clear that this point is also on the curve. Now given two points P and Q with $Q \neq -P$, we consider the curve $y^2 = x^3 + ax + b$ over $\mathbb{R}^2$ and consider the line through P and Q, and if $P = Q$, we consider the line tangent to the curve at P instead, this line would intersect the curve at another point, which we would define as $-(P+Q)$, and if $Q = -P$, $P + Q$ is the point at "infinity".

a) (4%) Derive the formula for point addition and point doubling.

b) (2%) Show that one can compute $n \times P := \underbrace{P + \cdots + P}_{n \text{ times}}$ with $O(\log n)$ additions.

c) (2%) Notice that even if $P \neq Q$ and $P \neq -Q$, it is not necessary that the line passing P and Q to intersect the elliptic curve at another point, discuss how one would define addition in this case. Note that the definition must satisfy the group laws.

One may verify that the set of integral points on the elliptic curve and a point at "infinity" together with the addition defined above form a finite abelian group. Remark: Using Hasse's theorem, we know that there are $(q+1) - 2\sqrt{p}$ to $(q+1) + 2\sqrt{p}$ many points in the group.

The discrete log problem on elliptic curves is defined as follows: Given a point P, and a point $Q := cP$, where c is unknown, on elliptic curve $y^2 = x^3 + ax + b$ over the field $\mathbb{F}_p$, find c. Most schemes involving elliptic curves rely on the hardness of this problem.

d) (5%) An elliptic curve is singular if its discriminant $\Delta := -16(4a^3 + 27b^2)$ over the base field $\mathbb{F}_p$ is zero, are discrete log problems on singular elliptic curves easier to solve? Why or why not?

Remark: Discrete log problem could be defined on any group, that is, given a group G and its binary operator $\circ$, a generator g, and an element h, find an integer x such that $g^x := \underbrace{g \circ g \circ \cdots \circ g}_{x \text{ times}} = h$, however, not all discrete log problems are equal, for example, discrete log problem on additive group of $\mathbb{F}_p$ is trivial, and it is harder on the multiplicative group of $\mathbb{F}_p$, and much harder on elliptic curve groups as of now.

3.2. Exercises from the slides

Textbook RSA RSA is the most popular public key encryption scheme, the scheme is as follows:

Suppose Bob wants to send a message m to Alice, first, Alice chooses two random primes, p and q, and let $N = pq$, next Alice generates her public key e randomly such that e and $\phi(N)$ are coprime, and let $d = e^{-1} \mod \phi(N)$ be her private key. Then Alice would send over (e, N) to Bob, and Bob would send the ciphertext $c := m^e \mod N$ to Alice, and Alice could decrypt the ciphertext by calculating $c^d \mod N$.

a) (3%) Show that RSA is correct.

Elliptic Curve Digital Signature Algorithm (ECDSA) ECDSA is a digital signature scheme which uses elliptic curves, the scheme is as follows

Suppose Alice wants to sign a message, and Bob needs to verify it, first, they agree on the same elliptic curve E, base point G, its order n and a hash function H, and Alice generates her private key d randomly in $[1, n-1]$ and let $Q = d \times G$ be her public key.

SIGN(m)
Generate a cryptographically secure random nounce $k \leftarrow [1, n-1]$ $P = (x, y) = k \times G$ $r = x \mod n$ $s = k^{-1}(H(m) + rd) \mod n$ Pick another k and retry if $r = 0$ or $s = 0$ **Output:** (r, s)

VERIFY$(m, (r, s))$

Verify that $Q \neq O$ is a point on the curve, and $n \times Q = O$.
$u_1 = H(m)s^{-1} \mod n$
$u_2 = rs^{-1} \mod n$
$P' = (x', y') = u_1 \times G + u_2 \times Q$
if $x' = r$ **then**
  Accept
else
  Reject

a) (2%) Show that the scheme is correct.

b) (2%) Show that if two different messages is signed with the same nounce k, then it is possible to recover the private key.

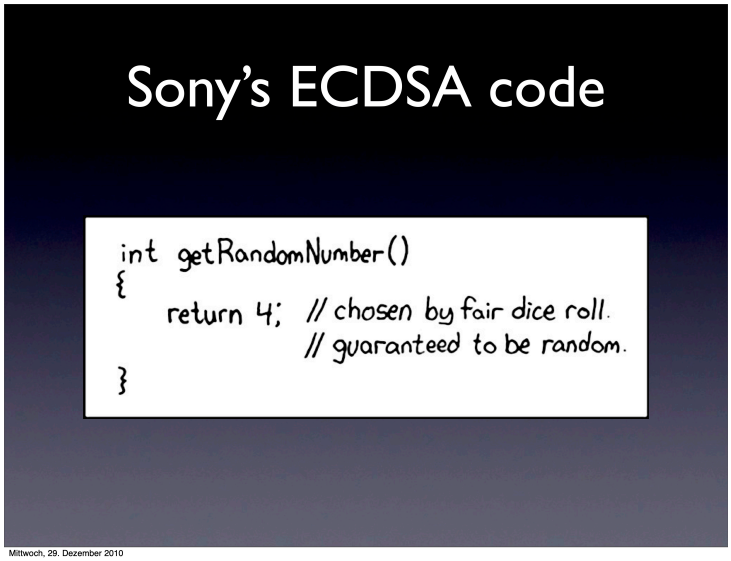

Figure 1: An real life example [1]

4. Applications of PRG

Convention

- $[n] = \{1, 2, \ldots, n\}$.
- $|X|$ denotes the number of elements in the set X.
- $x_1, \ldots, x_n \in_R X$ means for $i \in [n]$, x_i is independently and uniformly sampled from a set X.
- $y \leftarrow A(x)$ sets y to the output of running an algorithm A with input x.
- $x := y$ means assigning y to x.
- For $x, y \in \{0,1\}^n$, $x \oplus y$ denotes the bit-wise exclusive-or of x and y.

Recall that in the lectures we introduced the concept of a pseudorandom generator (PRG), and showed how we can build a semantically secure stream cipher from any PRG. However, many more primitives can be built from a PRG, including block cipher, one-time signature, and commitment. Furthermore, PRG is not only sufficient for these primitives, PRG is necessary for them because we can extract a PRG from all of these symmetric cryptographic primitives. In this problem, we illustrate the power of PRG by showing two applications of it.

First, we define PRG.

Definition 1. A function $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ is a PRG if it meets the following requirements.

- G is efficiently computable; more concretely, G is computable by a polynomial time function.
- G is extending; that is, $m(n) > n$ for every input length n.
- G is pseudorandom; that is, for every probabilistic polynomial time (PPT) distinguisher D, D wins Game 1 with probability $\frac{1}{2} + \nu(n)$, where ν is negligible.

Ch		D
$b \in_R \{0,1\}$		
$\begin{cases} y \in_R \{0,1\}^{m(n)} \\ x \in_R \{0,1\}^n, y \leftarrow G(x) \end{cases}$	$\begin{matrix} \text{if } b = 0 \\ \text{if } b = 1 \end{matrix}$	
	$\xrightarrow{\quad y \quad}$	
		$b' \leftarrow D(y)$
D wins if $b' = b$.		

Table 1: PRG game

We can interpret the pseudorandomness requirement in the following way: every efficient (randomized) algorithm cannot distinguish between truly random y and $G(x)$ better than tossing a coin.

[1] Figure obtained from https://fahrplan.events.ccc.de/congress/2010/Fahrplan/attachments/1780_27c3_console_hacking_2010.pdf

Note that for each n, there is a security game, in which D wins with probability $1/2 + \epsilon(n)$. To show that G is pseudorandom, ϵ has to be a negligible function. (Please refer to the slides or 2.3 of Boneh and Shoup for the definition and properties of negligible functions.) If you are not familiar with security reductions, in addition to the slides and Boneh and Shoup, we recommend this lecture note by Pass and Shelat, The Joy of Cryptography by Rosulek, or this handout by Falor, Vafa, and Xiao. Their definition of PRG might be different from our definition, so you should be careful if you want to adapt their proof.

4.1. Locked boxes

In the lectures, we learned how to flip a coin by telephone with the help of commitments, a cryptographic analog of a locked box. In this problem, we will see how to commit to a bit using a PRG.

Suppose Alice wants to commit to some bit d, and reveal the bit to Bob later. The commitment scheme consists of two stages.

Construction 2. $G : \{0,1\}^n \to \{0,1\}^{3n}$ is a PRG.

- Commit phase: Alice commits to $d \in \{0,1\}$ by performing the interaction in Table 2.

Alice		Bob
$x \in_R \{0,1\}^n$	$\xleftarrow{\quad r \quad}$	$r \in_R \{0,1\}^{3n}$
$y \leftarrow G(x)$		
$c := \begin{cases} y & d = 0 \\ y \oplus r & d = 1 \end{cases}$	$\xrightarrow{\quad c \quad}$	

Table 2: Commit phase

- Reveal phase: Alice sends x to Bob. Bob opens d by computing $G(x) \oplus c$.

In the coin-flipping protocol in real life, Alice gives a locked box to Bob, in which she places a sheet of paper that says the result of her coin toss, and sends the key to Bob after Bob announces his result. This way, when Bob opens the box, he knows that Alice has locked the result in the box, and cannot change her mind after she gives the box to Bob.

Following this analogy we see that Alice is not allowed to change her mind after the commit phase and let Bob open to another $d' \neq d$. This is called the **binding** requirement of commitment schemes. In the following, we will show that the scheme is binding, in the sense that if Bob follows the protocol, Alice can open to both 0 and 1 with negligible probability.

a) (4%) Show that given Bob's choice of randomness $r \in_R \{0,1\}^{3n}$, the probability that Alice can open to both 0 and 1 is at most 2^{-n}.

Hint: When can Alice open to both 0 *and* 1*? It happens only when there are some* r, x_0, x_1 *with* $G(x_0) = G(x_1) \oplus r$*. But for each pair of* G*'s seeds* (x_0, x_1)*, there is at most one* r *such that* $G(x_0) = G(x_1) \oplus r$*.*

Moreover, Bob should not be able to look inside the locked box before Alice gives him the key, or Bob can decide the outcome of the coin flip all by himself. Thus, a commitment scheme

Alice		Bob
$d \in_R \{0,1\}$		
$x \in_R \{0,1\}^n$	$\xleftarrow{\quad r \quad}$	
$y \leftarrow G(x)$		
$c := \begin{cases} y & d = 0 \\ y \oplus r & d = 1 \end{cases}$	$\xrightarrow{\quad c \quad}$	
	$\xleftarrow{\quad d' \quad}$	
		Bob wins if $d' = d$

Table 3: Hiding game

has to hide its content, More formally, if every PPT Bob wins the Hiding game (Game 3) with probability $1/2 + \nu(n)$, where ν is negligible, we say Construction 2 is **hiding**.

We show that our scheme is hiding by constructing a security reduction to the pseudorandomness of G. Note that in the Hiding game, Bob is allowed to send arbitrary $r \in \{0,1\}^{3n}$ (not necessarily sampled uniformly randomly) to Alice. One motivation we give Bob the power to deviate from the protocol is that it is difficult for Alice to know whether Bob follows the protocol. Also, in real life, we hope the box does not leak its content even if Bob tries to open the box using lock picks or angle grinders.

b) (8%) Given a PPT Bob that wins Game 3 with probability $1/2 + \mu$, construct a PPT distinguisher D that wins the PRG game (Game 1) against G with probability $1/2 + \mu/2$. Please prove that its winning probability is at least $1/2 + \mu/2$. You do not have to explain why it runs in polynomial time, but please make sure it does.

Hint: D plays the role of Alice when interacting with Bob. How well does Bob do when given $y \leftarrow G(x)$ with $x \in_R \{0,1\}^n$? What about $y \in_R \{0,1\}^{3n}$?

4.2. An open problem

As a second application, we take a look at, $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$, a central open problem in computer science.

Let $\{0,1\}^* = \bigsqcup_{n=0}^{\infty} \{0,1\}^n$ denote the set of all binary strings. A decision problem $L \subseteq \{0,1\}^*$ is a subset of binary strings. For each problem instance $y \in \{0,1\}^*$, which is a string, we have to decide whether the answer to y is yes or no, where the answer is yes if and only if $y \in L$.

We say a problem $L \in \mathbf{P}$ if there is an algorithm A that returns 0 or 1 in at most $t(|y|)$ steps on input y, t is a polynomial, and

$$y \in L \text{ if and only if } A(y) = 1.$$

This means L is a decision problem that can be solved in polynomial time. Examples of problems in $\mathbf{P}$ include graph connectivity, (decision version of) linear programming, and primality testing.

We say a problem $L \in \mathbf{NP}$ if there is an algorithm V that returns 0 or 1 in at most $t(|y|, |x|)$ steps on input y and x, t and s are polynomials, and

$$y \in L \text{ if and only if there is } x \in \bigsqcup_{n=0}^{s(|y|)} \{0,1\}^n \text{ such that } V(y,x) = 1.$$

This means if the answer to y is yes, there is a proof string x that can be efficiently verified by V. On the other hand, if the answer is no, no proof can be used to fool the verifying algorithm V. Examples of problems in **NP** include all the problems in **P**, graph 3-colorability, (decision version of) integer programming, and (decision version of) integer factorization.

The proof that $\mathbf{P} \subseteq \mathbf{NP}$ is straightforward because an efficient solver A can be viewed as an efficient verifier that takes an empty proof string. On the other hand, however, Boolean formula satisfiability, knapsack problem, and many other problems are believed to live in **NP** but not in **P**. We think they are too difficult to solve efficiently in all cases, but none of these are proved to be out of **P**. There is even formal evidence (called barrier results) that many current techniques are not enough to separate the classes **P** and **NP**. (See, for example, Scott Aaronson's survey.) If you can show either $\mathbf{P} = \mathbf{NP}$ or $\mathbf{P} \neq \mathbf{NP}$, you can receive a million dollars from Clay Institute.

Nevertheless, the following shows that if we can prove the existence of a PRG unconditionally, that is, without assumptions like discrete log or factoring is hard, we immediately resolve this open problem.

a) (8%) Show that if $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ is a PRG, then $\mathbf{P} \neq \mathbf{NP}$.

Hint: Consider the decision problem

$$L = G(\{0,1\}^*) = \{G(x) : x \in \{0,1\}^*\}.$$

Show that L is in NP and use a polynomial time decider of L to break the pseudorandomness of G.

Since PRG is necessary for symmetric cryptography (see the reference mentioned before the start of subsection 4.1.), from what we just showed, $\mathbf{P} = \mathbf{NP}$ is a necessary assumption for symmetric cryptography. As a side note, the natural proof barrier to proving $\mathbf{P} \neq \mathbf{NP}$ also has to do with the existence of PRG (secure against certain circuit classes). This can be found in the aforementioned survey by Aaronson.

Capture The Flag

5. Simple Crypto (10%)

Carol, the creator of Super Secure Secret Storage (SSSS), has implemented functionality within the system to store secrets and protect users by allowing them to set a passphrase. Additionally, Carol holds the belief that classical ciphers are cryptographically secure, expressing enough confidence to provide functionality for accessing the database where user passphrases are encrypted using classical ciphers.
On the contrary, Alice, a student who has learned about the vulnerabilities of classical ciphers, is attempting to persuade Carol that these ciphers have weaknesses. Can you help Alice to show that classical ciphers have vulnerabilities that may break the Carol's system and cause the secret leak by obtain the four flags hidden within the secrets of four individuals stored in SSSS? You can access the system by `nc cns.csie.org 44398`

a) (2%) Obtain the FLAG1 from Affine's secret

b) (2%) Obtain the FLAG2 from Bob's secret

c) (3%) Obtain the FLAG3 from Eve's secret

d) (3%) Obtain the FLAG4 from Admin's secret

Hint 1: If you use Python and pwntools, keep an eye on the return value of Python's builtin function `input()`*. Sometimes a trailing newline character may exist with unknown reason.*
Hint 2: Passphrases and Secrets are words that are both human-readable and meaningful.
Hint 3: Read the homework instructions carefully.

6. Simple RSA (10%)

After solving Simple Crypto and show that classical ciphers are indeed insecure, Carol decided to modify the system. Now, instead of using encrypted passphrase, the secret is encrypted by using RSA. Can you still break Carol's system to show that RSA may still be compromised without proper implementation?
You can access the challenge by `nc cns.csie.org 44399`. The challenge source is included in `simple_rsa/server.py`.

a) (2%) Obtain the FLAG1 from Affine's secret

b) (2%) Obtain the FLAG2 from Bob's secret

c) (3%) Obtain the FLAG3 from Eve's secret

d) (3%) Obtain the FLAG4 from Admin's secret

7. POA (15%)

One day, you saw a CNS TA leaving room 204, later you found out that the TA forgot to log out from the new super-secure messaging app POA on a public computer, you can access this via `nc cns.csie.org 1337`. The challenge source is included in `poa/server.py`

a) (8%) It seems like someone left a message for the TA, you wondered if this message has anything to do with the first flag.

b) (7%) Based on the first message, there seems to be another flag, maybe you can pretend to be the TA and ask for the second flag?

8. CNS Store (15%)

Suddenly, you find yourself in a mysterious store. As you look around, you discover that the store sells all kinds of goods, perhaps even some flags. However, you need enough money to purchase the items you desire.
Visit CNS Store by `nc cns.csie.org 9010`. Also refer to the source code in `cns-store/server.py`

a) (5%) flag1: Can you buy the flag in stage 1? It is a bit expensive though.

b) (5%) flag2: Things wouldn't be that easy this time! Try to buy the flag again.

c) (5%) flag3: Login as admin and steal the flag to revenge on the owner that force you to work here. NEETs never do gigs, right?