CNS Homework 1

B10202012 劉仲楷

Handwriting

1 CIA

- Confidentiality ensures that the information is accessible only to those who are authorized to view it. The message application "line" transfers the user's message through encryption to violate the eavesdroppers.
- Integrity ensures that the information is complete and unaltered, especially during the transmission. The Certificate Authorities (CAs) contain digital signatures that verify the authenticity, ensuring attackers have not altered the data.
- Availability ensures the information is accessible or the service is usable to those who are needed. The proof of work slows down the denial of service attack and protects the availability.

2 Hash Function

- One-wayness: given y and the hash function H, it is computationally infeasible to find x such that $H(x) = y$. Password hashing relies on this property. For the situation that the hash is leaked, the password is secure since it is computationally infeasible to find x.
- Weak collision resistance: given x_1 and the hash function H, it is computationally infeasible to find another x_2 such that $H(x_1) = H(x_2)$ but $x_1 \neq x_2$. The integrity check depends on this property. If the verification hash message is authenticated then the receiver can check the hash and confirm the integrity. However, if another $x_2 \neq x_1$ is found, the attacker could alter the message to x_2 and circumvent this protection.
- Strong collision resistance: given the hash function H, it is computationally infeasible to find any pairs of x_1, x_2 such that $H(x_1) = H(x_2)$ but $x_1 \neq x_2$. The commitment scheme relies on this property since if not one could easily use the hash $H(x_1) = H(x_2)$ in the commit phase and choose the one of x_1 or x_2 depending on the wanted result in the reveal phase.

3 Asymmetric Cryptography

3.1 An introduction to elliptic curves in cryptography

a)

- Point addition
  Suppose the point of the line and the curve intersect at R, the product of addition is point $-R$. We can simply write the formula of the curve and the line to get the intersecting point and derive the formula.

  $$\begin{cases} y^2 &= x^3 + ax + b \\ y &= \lambda x + d \end{cases} \Rightarrow y^2 = \lambda^2 + 2\lambda d + d^2 \cdot \Rightarrow x^3 - \lambda^2 x^2 + (a - 2\lambda d)x + (b - d^2) = 0$$

  Substituting (x_P, y_P) and (x_Q, y_Q), we can get

  $$\lambda = \frac{y_Q - y_P}{x_Q - x_P}.$$

  By the sum of roots, we can get

  $$x_P + x_Q + x_{-R} = \lambda^2 \Rightarrow x_R = x_{-R} = \lambda^2 - x_P - x_Q$$
  $$y_R = -y_{-R} = -(\lambda(x_{-R} - x_P) + y_P) = \lambda(x_P - x_R) - y_P.$$

  To conclude,

  $$\begin{cases} x_R &= \lambda^2 - x_P - x_Q \\ y_R &= \lambda(x_P - x_R) - y_P \end{cases}$$

- Point doubling
  By the same method, but first, we should determine the tangent line.

  $$2ydy = (3x^2 + a)dx \Rightarrow \lambda = \frac{dy}{dx}(x_P, y_P) = \frac{3x_P^2 + a}{2y_P}.$$

  The rest are the same as the addition.

b) $n \times P = a_0 2^0 P + a_1 2^1 P + \cdots + a_{\lceil \log_2 n \rceil - 1} 2^{\lceil \log_2 n \rceil - 1} P$, where $a_i \in \{0, 1\}$. Therefore, by doubling P $\lceil \log_2 n \rceil - 1$ times and adding together takes $\sim 2 \log_2 n$ operation, i.e. $O(\log n)$ additions.

c) If this is the case, then there must be one point in P or Q in which the line is tangent to the curve, says P. This is a kind of degeneracy on P, therefore $-R = P$ and $R = -P$. Thus, $P + Q = -P$. It is obvious to see that it satisfies the group laws since it is just a degenerate form.

d) [1] The discriminant $\Delta \neq 0$ means that the graph shows a cusp or node or isolated point, which may have multiple roots. More precisely,

- In the case of three roots, which exists a cusp, the map

  $$(x, y) \mapsto \frac{x}{y}, \infty \mapsto 0$$

  is a group isomorphism between two groups, regarded as an additive group. By using this result, suppose G is the base point, $P = d \times G$ and the prime number p, we can easily recover d by

  $$g = G.x \times G.y^{-1} \mod p, \quad y = P.x \times P.y^{-1} \mod p$$
  $$d = g^{-1} \times y \mod p$$

- On the other hand, in the case that exists two roots, we can express the curve as
$$y^2 = x^2(x + a),$$
where $a \neq 0$ is another root. We have a node if $a > 0$, or otherwise an acnode (isolated point on $(0, 0)$). Consider the map
$$(x, y) \mapsto \frac{y + \sqrt{a}x}{y - \sqrt{a}x}, \infty \mapsto 1,$$
gives an isomorphism. The rest is similar to the previous case.

3.2 Exercises from the slides

a) It suffices to show that $c^d \equiv m \mod N$.
$$\begin{aligned} c^d &\equiv (m^e \mod N)^d && \mod N \\ &\equiv m^{ed} && \mod N \end{aligned}$$
Since $ed \equiv 1 \mod \phi(n)$, $ed = 1 + k\phi(n)$ where $k \in \{0, 1, \cdots\}$. By Euler's theorem, $a^{\phi(n)} \equiv 1 \mod N$, where a and n are co-primes. Therefore,
$$\begin{aligned} c^d &\equiv m^1 \cdot m^{k\phi(n)} && \mod N \\ &\equiv (m \mod N)(m^{\phi(n)} \mod N)^k && \mod N \\ &\equiv m && \mod N \end{aligned}$$

Elliptic Curve Digital Signature Algorithm (ECDSA)

a)
$$\begin{aligned} P' &= u_1 \times G + u_2 \times Q \\ &\equiv (H(m)s^{-1} \mod n) \times G + (rs^{-1} \mod n)(d \times G) \\ &\equiv (H(m)s^{-1}) \times G + (rs^{-1})(d \times G) && \mod n \\ &\equiv (H(m) + rd)s^{-1} \times G && \mod n \\ &\equiv (H(m) + rd)(k^{-1}(H(m) + rd))^{-1} \times G && \mod n \\ (\text{Since } s \neq 0) &\equiv (H(m) + rd)k(H(m) + rd)^{-1} \times G && \mod n \\ &\equiv k \times G && \mod n \\ &\equiv P && \mod n \end{aligned}$$
Since $r \equiv x \mod n$, then $P \equiv P' \mod n$ implies $r \equiv x' \mod n$.
Thus, Pr[Verify(m, Sign(m)) =accept] = 1.

b) Suppose two different messages m_1, m_2 are signed with the same nounce k and private key d, then the public keys are (r, s_1), (r, s_2).
$$\begin{aligned} & \begin{cases} s_1 = k(H(m_1) + rd) \mod n \\ s_2 = k(H(m_2) + rd) \mod n \end{cases} \\ \Rightarrow \quad & s_2(H(m_1) + rd) = s_1(H(m_2 + rd)) && \mod n \\ \Rightarrow \quad & d = (s_2H(m_1) - s_1H(m_2))(r(s_1 - s_2))^{-1} && \mod n \end{aligned}$$
Consequently, the private key d is recovered.

4 Applications of PRG[1]

4.1 Locked boxes

a) Denote the set containing the pair (x_0, x_1) as X and the set containing r as R. We can achieve $|X| = |\{0,1\}^n|^2 = 2^{2n}$ while $|R| = |\{0,1\}^{3n}| = 2^{3n}$. Then the probability that the situation for choosing $r = G(x_0) \oplus G(x_1)$ is bounded by

$$\frac{|X|}{|R|} = 2^{-n}.$$

b) The construction is shown in table 1. The key is D decides b' based on d exclusive nor d' Given that Bob wins the hiding game with probability $1/2 + \mu$, i.e.,

$$\Pr_{HG}[d = d'] = \frac{1}{2}\Pr[Exp_{HG}(0) = 0] + \frac{1}{2}\Pr[Exp_{HG}(1) = 1] = \frac{1}{2} + \mu.$$

The probability of D winning the PRG game is

$$\Pr_{PRG}[b = b'] = \frac{1}{2}\Pr[Exp_{PRG}(0) = 0] + \frac{1}{2}\Pr[Exp_{PRG}(1) = 1] \tag{1}$$

For convenience, we denote the hiding game as HG. The situation in the first term is competing against a truly random number, e.g., tossing a coin. There is no better strategy than a wild guess. Therefore, the probability is

$$\Pr[Exp_{PRG}(0) = 0] = \frac{1}{2}$$

On the other hand, the situation in the second term is competing against a pseudorandom generator. D guesses 1 only when $d = d'$.

$$\Pr[Exp_{PRG}(1) = 1] = \frac{1}{2}\Pr[Exp_{HG}(0) = 0] + \frac{1}{2}\Pr[Exp_{HG}(1) = 1] = \frac{1}{2} + \mu$$

Finally, plug into eq. 1 and we get

$$\Pr_{PRG}[b = b'] = \frac{1}{2}\left(\frac{1}{2}\right) + \frac{1}{2}\left(\frac{1}{2} + \mu\right) = \frac{1}{2} + \frac{\mu}{2}$$

4.2 An open problem

a) By checking whether $G(y) = x$, it is clear to see that $L \in \mathbf{NP}$ (it is of course in polynomial time by the definition of PRG (def 1 item 1)). Suppose $\mathbf{P} = \mathbf{NP}$, by definition, there exists an algorithm A such that $y \in L$ if and only if $A(y) = 1$. On the other hand, the definition of PRG states that every D wins the PRG game with probability $\frac{1}{2} + \nu(n)$ where ν is negligible. However, by exploiting the algorithm A, we can always find y with probability 1 in polynomial time, which is obviously nonnegligible. This leads to a contradiction and therefore $\mathbf{P} \neq \mathbf{NP}$.

[1] I looked up the textbook for question a). The rest is discovered on my own without discussing with anyone else. (TA says needs to be clarified)

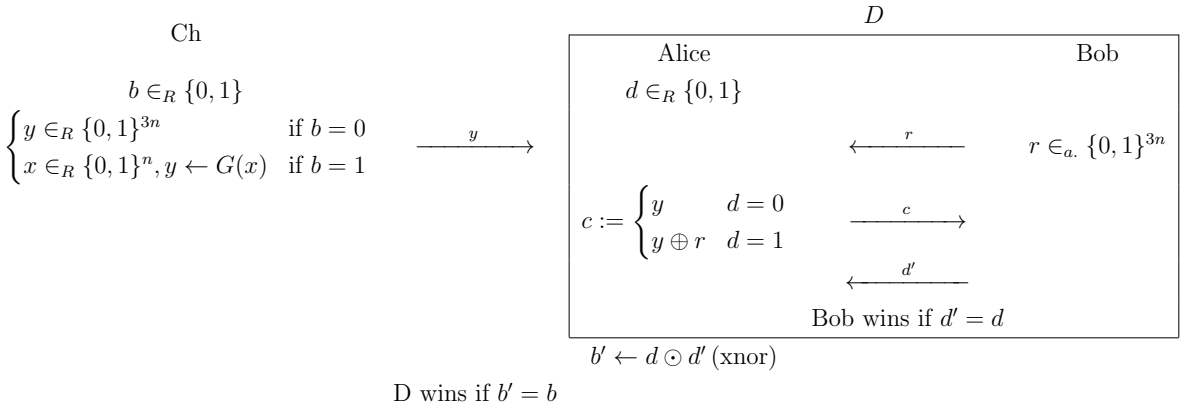

Table 1: Constructing a PPT distinguished D of PRG game using hiding game

Capture The Flag

The write-ups are mainly about what the vulnerability is and the logic of capturing the flag. More implementation details are in `readme.txt`.

5 Simple Crypto

a) `CNS{51MP13_M47H_70_8r34K_4FF1N3_a3e4aa86dc09bf988d0596c4e83a6af2}`
The passphrase is encrypted by Affine cipher, $e = ax + b \mod m$ where a, b, m are parameters. Loop through m; for every m, a, b is determined by the given two characters. Print the result if it is printable. Determine the passphrase by human laboring.

b) `CNS{8rU73_F0rC3_4r3_P0551813_70_8r34K_Z16Z46_50b012399a6eb9004fa2ae62d276cec2}`
The hint is "hidden in the fence" so it is rail fence cipher. I use the package secretpy for decryption, which can be installed by `pip`. The cipher in the package is known as ZigZag. Print the result and determine the passphrase by human laboring.

c)

d) `CNS{07P_4CH13V35_P3rF3C7_53CUr17Y_UND3r_455UMP710N5_b7dd544750014b538024ebe801788feb}`
First, Use the base 64 decoder to decode the encrypted passphrase. The server would return the secret and the clues that the one-time pad key is 6 bytes and the secret is printable. Therefore, I find the potential key for 6 bytes individually which satisfies all characters using the key to decrypt is printable. After that, I search for the real key in the key space of the potential key by brute force. It can be sped up if some bytes of the key are found.

6 Simple RSA

The code `code6.ipynb` is written in Google Colab. The first three flags rely on an online tool, RsaCtfTool. The requirements and instructions for installation are written in the GitHub pages if needed to run on the local side. Or, simply use Colab to run.

a) `CNS{345Y_F4C70r1Z4710N}`
The code utilizes the website factordb to find the prime numbers.

b) `CNS{7W0_C1PH3r73X7_W111_8r34K_r54?!_79f17b6d9c07fb154c6dcdb92ab047a7}`
Due to giving two different e but using same n, we can exploit a `same_n_huge_e` attack.

c) `CNS{CH1N353_r3M41ND3r_r3P347_8r04DC457_4774CK_edc38122a288aa9922617b5a6a08ac61}`
By observing the primes using are not the same every time and the same small e, we can access the server e times and get different pairs of n and c. Finally, exploit Håstad's broadcast attack.

d) `CNS{816_e_4ND_W13N3r_4774CK_1N_7H3_NU75H311_0f2146bda49df4206f344cd29b0503fa}`
Due to the large e, we can exploit Wiener's attack.

7 POA

a) `CNS{p4dd1n9_0r4cl3_4tt4ck_15_315_3v1l}`
Observing that `ValueError` will be raised when there exists padding errors. Therefore, we can exploit the padding oracle attack. It is noted that POA is "highly likely" to succeed (since it may accidentally solve the last two padding which is valid). Thus, if it fails, run again.

b) `CNS{f0r61n6_r3qu3575_4r3_5up3r_345y}`
The vulnerability is that the block before the sender information is not checked. By exploiting the bit-flipping attack, we can generate the message and bit-flip to change the sender to TA. However, the `UnicodeDecodeError` would be encountered. Observing the source code to discover that the 16th and 17th bytes can be altered, i.e. `'; '`. The two bytes can only be XORed with `\x00` to `\x7f` ($0 \sim 127$) to get printable bytes. Loop through the two bytes to find one pair that can decoded properly.

8 CNS Store

First, proof of work is needed in the login stage. Simply brute forcing calculates some hash and finds if there is one that meets the requirement.

a) `CNS{S1mp!e_H45h_C011!510n_@7tacK!}`
The vulnerability is that the shelf would be filled by 10 only the first time. However,

the same product name can be purchased only one time. On the other hand, the product on the shelf is determined by the hash of SHA1. Thus, if a collision is found, then we can buy the different product name (preimage) twice with an amount of 10, and the amount on the shelf becomes -10. After playing the lottery, we can earn $10 \cdot 5$ each time. In conclusion, we buy two different products with the same hash value in the first two rounds and play the lottery in three rounds. Then we have $100 - 10 \cdot 5 \cdot 2 + 10 \cdot 5 \cdot 3 = 150$. The cash we have during the process is ≥ 0. Finally, we can buy the flag. The SHA1 collision can be found in the Google announcement. Since the `CNS2024` is needed in the product name, we simply suffix it after the two preimages.

b) `CNS{H@ppy_B!r7hDaY_t0_Y0U!}`
The concept is the same as above, except the "collision" needs to be found in the last four bytes of SHA256. Apply a birthday paradox attack in which the preimage contains `CNS2024` and the given key.

c) `CNS{https://www.youtube.com/watch?v=YFJowRNfMGI}`
The vulnerability is that it only check the last 5 characters of the given identity. Therefore, we can assign `admin` after the original identity using the length extension attack. I use the package HashTool. The installation details are in the GitHub.

References

[1] Lawrence C. Washington. *Elliptic Curves: Number Theory and Cryptography*. 2nd ed., pp. 59–64.