Homework 1 Solution

Due on Monday, Oct. 16 , 2023, 11:59 PM (via Canvas)

Problem 1

1 Group Properties

For a group $\mathbb{G}$, if $g, h \in \mathbb{G}$ and $g^n = h$, we call g the n-th root of h, if ω_1 is the x-th root of h and ω_2 is the y-th root of h.

1. In case $\gcd(x, y) = 1$, what is the xy-th root of h?
2. How about the case $\gcd(x, y) = d \neq 1$?

Solution:

1. (a) Since ω_1 and ω_2 are the x-th and y-th roots of h, respectively, we can deduce that: $\omega_1^x = \omega_2^y = h$. Our aim is to find Ω such that: $\Omega^{xy} = h$. Since x and y are coprime, the Generalized Euclidean algorithm will guarantee integers a and $b \in \mathbb{Z}$ such that: $ax + by = 1$. We will claim that $\Omega = \omega_2^a \omega_1^b$ is the xy-th root of h, and here is the proof:

$$(\omega_2^a \omega_1^b)^{xy} = (\omega_2^x)^{ay} (\omega_1^y)^{bx} = h^{ax} \times h^{by} = h^{ax+by} = h^1 = h \tag{1}$$

   So the xy-th root of h is $\Omega = \omega_2^a \omega_1^b$.

   (b) following a similar approach to the previus section, one can easily verify that the result is $\Omega = (\omega_2^a \omega_1^b)^{\frac{1}{d}}$.

Problem 2

2 RSA algorithm

You are setting up a RSA cipher woth modulus 5021131, You may use the fact that 5021131 = 1907(2633). (You may want to use a computer to do most of the computation on this problem.)

1. Is 5 a valid encryption key? If so, find the corresponding decryption key. If not, explain why not.

2. Is 7 a valid encryption key? If so, find the corresponding decryption key. If not, explain why not.

3. The encryption key e = 3 is valid for the modulus 5021131 and corresponds to the decryption key d = 3344395. If you are setting up a public key with this information, what is all of the information that you would post publicly so that a friend could send you an encrypted message? What is all of the information relevant to this cipher that you would keep secret?

4. If your plaintext is represented by x = 884204, use the encryption key e = 3 to encrypt x.

5. You have received the ciphertext message y = 384. Use the decryption key d = 3344395 to decrypt the message.

Solution:

1. Yes. We need to check if 5 is relatively prime to the Euler's totient function $\phi(n)$. To calculate $\phi(n)$, we can use the formula $\phi(n) = (p-1) * (q-1)$, where p and q are the prime factors of the modulus. In this case, $p = 1907$ and $q = 2633$, so we have: $\phi(n) = (1907-1) * (2633-1) = 1906 * 2632 = 5014592$. Calculating the $gcd(5, 2632) = 1$, $gcd(5, 1906) = 1$, we conclude that 5 is a valid encryption key for the RSA cipher.

2. No, based on the explanation from the previous part, the greatest common divisor of 7 and 2632 is 7. Since $gcd(7, 2632) \neq 1$, the modular multiplicative inverse can not be found. This indicates that 7 is not a valid encryption key for the RSA cipher with a modulus of 5021131.

3. In order to encrypt a message, we need to use the formula $c(m) = m^e \mod n$, so the public key $\{e, n\}$ must be published for friends. However, it is important not to publish the private key $\{d, n\}$, which is needed to decrypt the message using the formula $m(c) = c^d \mod n$. Also, we note that the values of p, q should be kept private.

4. Using $c(m) = m^e \mod n$ we have:
$c(m) = 884204^3 \mod 5021131 = 4491607$

5. Using $m(c) = c^d \mod n$ we have:
$m(c) = 384^{3344395} \mod 5021131 = 387243$

Problem 3

3 Merkle Tree

a) Consider a Merkle tree with 4 leaf nodes labeled A, B, C and D. Assume the hash function used to calculate the hash values for the leaf nodes is SHA-256. Calculate the root hash of the Merkle Tree. (Check the tree in Figure 1. Also, You may want to use an online tools or any developed code in computer to do the computation on this problem.)

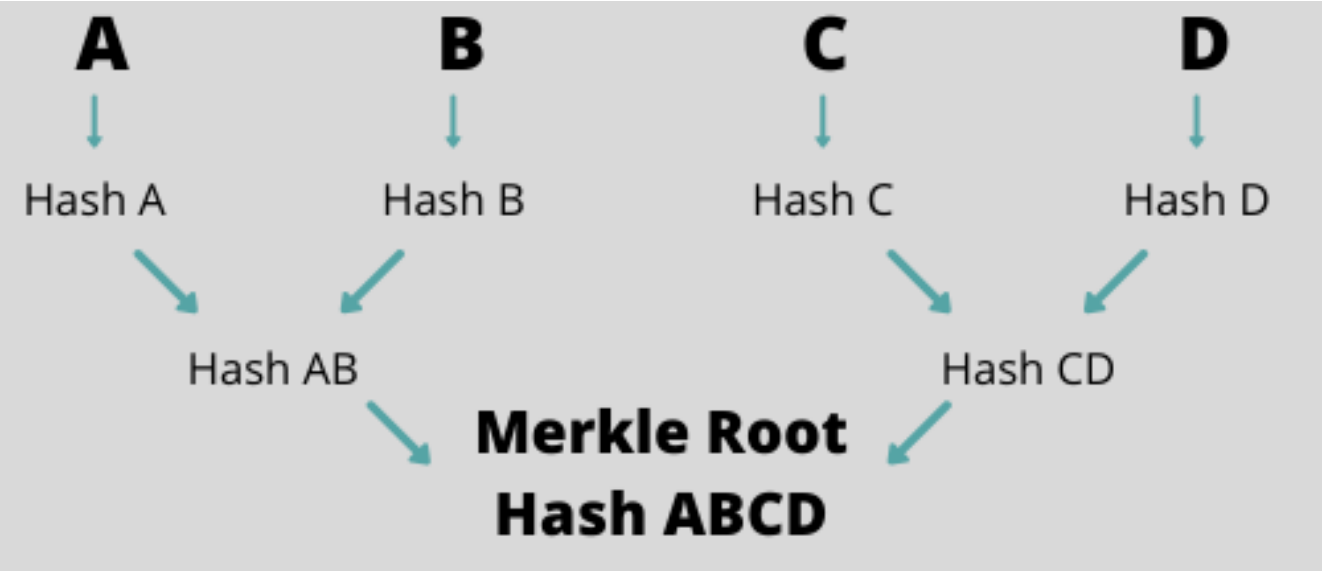

Figure 1: Merkle Root Hash ABCD

b) Alice can use a binary Merkle tree to commit to a tuple of elements $S = (T_1, \cdots, T_n)$ so that later she can prove to Bob that some T_i is in S, using an inclusion proof containing at most $[\log_2 n]$ hash values. The binding commitment to S is a single hash value. In this question your goal is to explain how to do the same using a k-ary tree, that is, where every non-leaf node has up to k children. The hash value for every non-leaf node is computed as the hash of the concatenation of the values of all its children.

b-1. Suppose $S = (T_1, \cdots, T_9)$. Explain how Alice computes a commitment to S using a ternary Merkle tree (i.e., $k = 3$). How does Alice later prove to Bob that T_4 is in S?

b-2. Suppose S contains n elements. What is the length of the proof that proves that some T_i is in S, as a function of n and k?

b-3. For large n, if we want to minimize the proof size, what is the best value for k?

Solution:

a)

Step 1: Compute the hash values for the leaf nodes

- Hash("A") = 559aead08264d5795d3909718cdd05abd49572e84fe55590eef31a88a08fdffd

- Hash("B") = df7e70e5021544f4834bbee64a9e3789febc4be81470df629cad6ddb03320a5c

- Hash("C") = 6b23c0d5f35d1b11f9b683f0b0a617355deb11277d91ae091d399c655b87940d

- Hash("D") = 3f39d5c348e5b79d06e842c114e6cc571583bbf44e4b0ebfda1a01ec05745d43

Step 2: Calculate the intermediate hash values for the internal nodes by pairing up adjacent leaf nodes in a bottom-up manner and computing the hash value of their concatenation:

- Hash(AB) = SHA256(Hash("A") || Hash("B"))

⇒ Hash(AB) = b30ab174f7459cdd40a3acdf15d0c9444fec2adcfb9d579aa154c084885edd0a

- Hash(CD) = SHA256(Hash("C") || Hash("D"))

⇒ Hash(CD) = 26b5aabe804fe5d533c663dea833e8078188376ce5ca2b5c3371d09ef6b0657b

Step 3: Therefore, the root hash of the Merkle Tree is

- Hash(ABCD) = SHA256(Hash(AB) || Hash(CD))

⇒ Hash(ABCD) = 50a504831bd50fee3581d287168a85a8dcdd6aa777ffd0fe35e37290268a0153

b)

b-1) To prove to Bob that T_4 is in S, Alice provides an inclusion proof containing at most $2 \times \lceil \log_3 9 \rceil = 4$ hash values:
1. She provides the hash of the siblings of T_4, which are $Hash(T_5)$ and $Hash(T_6)$.
2. She also provides $Hash(Hash(T_1)||Hash(T_2)||Hash(T_3))$ and $Hash(Hash(T_7)||Hash(T_8)||Hash(T_9))$.

Bob can easily find $Hash(T_4)$ because T_4 is available.

Using $Hash(T_5)$ and $Hash(T_6)$, $Hash(Hash(T_4)||Hash(T_5)||Hash(T_6))$ can be calculated. Moreover, Alice has provided $Hash(Hash(T_1)||Hash(T_2)||Hash(T_3))$ and $Hash(Hash(T_7)||Hash(T_8)||Hash(T_9))$. Thus the root hash of this 3-ary Merkle Tree can be found. If the final root hash matches the commitment, Bob knows that T_4 is indeed part of the original tuple S. Refer to Figure 2 where proof of T_3 is discussed (Procedures are the same).

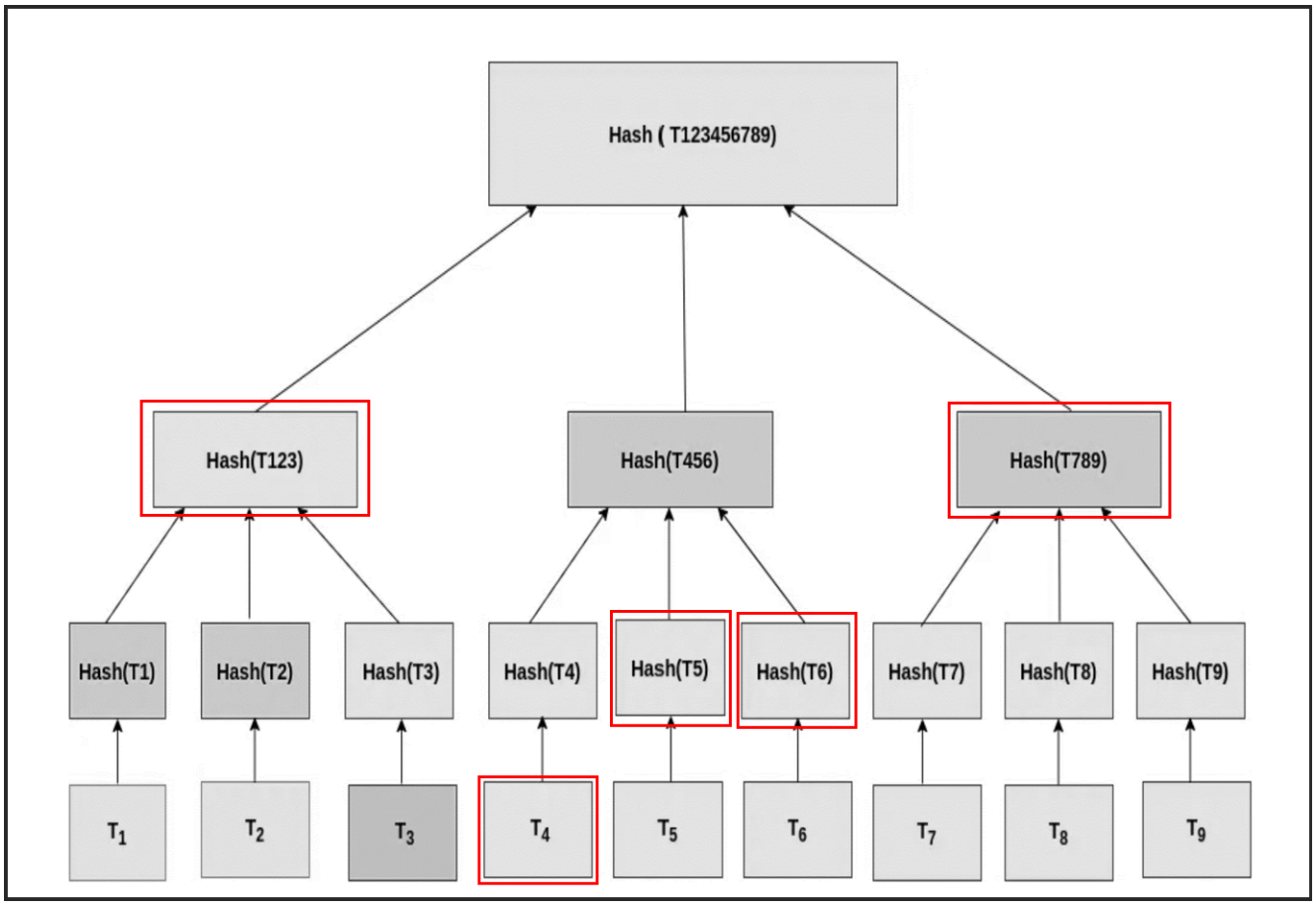

Figure 2: A 3-ary Merkle tree with Merkle proof highlighted in yellow

b-2)

The length of the proof is $(k-1) \times \lceil \log_k n \rceil$, where k is the number of children per non-leaf node, and n is the number of elements in the tuple. To achieve the hash parent in any layer of the K-ary Merkle Tree, we have one leaf of the Tree and the rest $k-1$ are unknown to us. Thus, $k-1$ other leaves are needed. Also, $\lceil \log_k n \rceil$ layers of tree are necessary to achieve the root. Thus, the length of the proof would be $(k-1) \times \lceil \log_k n \rceil$.

In this case, for a ternary tree ($k = 3$), the length of the proof is $2 \times \lceil \log_3 n \rceil$.

b-3)

We need to find the suitable value k^* for which proof size $(k-1) \times \lceil \log_k n \rceil$ is minimized (For a fixed arbitrary n). $k = 1$ is clearly not acceptable. Now, define $f(k) = (k-1) \times \lceil \log_k n \rceil$. We have

$$\frac{d}{dk} f(k) = \frac{ln(n)(kln(k) - k + 1)}{kln^2(k)} \tag{2}$$

We note that for all $k > 1, k \in \mathbb{N}$, we have $kln(k) - k + 1 > 0$. Therefore, $\frac{d}{dk} f(k) > 0$. This means that $f(k)$ is ascending. So k^* must be the lowest possible value which is $k^* = 2$.

Problem 4

4 Digital Signature Algorithms

Suppose Bob wants to sign a message m and Alice wants to verify the signed message. Initially, they must agree on the curve parameters (CURVE, G, n). Bob creates a key pair, consisting of a private key integer d_A, randomly selected in the interval $[1, n-1]$; and a public key curve point $Q_A = d_A \times G$.

Parameter	Description
G	elliptic curve base point, a point on the curve that generates a subgroup of large prime order n
n	integer order of G, means that $n \times G = O$, where O is the identity element.
d_A	the private key which is randomly selected
Q_A	the public key $d_A \times G$ which is calculated by elliptic curve
m	the message to send
$\times$	Used to denote elliptic curve point multiplication by a scalar

For Bob to sign a message m, she follows these steps:

1. Calculate $e = \text{HASH}(m)$.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order n.
3. Select a cryptographically secure random integer k from $[1, n-1]$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$, go back to step 3.
6. Calculate $s = k^{-1}(z + rd_A) \bmod n$. If $s = 0$, go back to step 3.
7. The signature is the pair (r, s). (And $(r, -s \bmod n)$ is also a valid signature.)

a) Having a signature pair and the message m, how can Alice verify if it is a correct signature from Bob?

b) Given two signatures (r, s) and (r, s'), employing the same unknown k for different known messages m and m', is it possible for an attacker to find private key (d_A)? (If yes, formulate the method of the attacker.)

Solution:

a)

Bob follows these steps:

1. Verify that r and s are integers in $[1, n-1]$. If not, the signature is invalid.
2. Calculate $e = \text{HASH}(m)$, where HASH is the same function used in the signature generation.
3. Let z be the L_n leftmost bits of e.
4. Calculate $u_1 = zs^{-1} \bmod n$ and $u_2 = rs^{-1} \bmod n$.
5. Calculate the curve point $(x'_1, y'_1) = u_1 \times G + u_2 \times Q_A$. If $(x'_1, y'_1) = O$ then the signature is invalid.
6. The signature is valid if $r \equiv x'_1 \pmod n$, invalid otherwise.

To see how this the algorithm works, denote C as the curve point computed in step 5 of verification,

$C = u_1 \times G + u_2 \times Q_A$

From the definition of the public key as $Q_A = d_A \times G$,

$C = u_1 \times G + u_2 d_A \times G$

It implies that,

$C = (u_1 + u_2 d_A) \times G$

Expanding the definition of u_1 and u_2 from verification step 4,

$C = (zs^{-1} + rd_A s^{-1}) \times G$

Collecting the common term s^{-1},

$C = (z + rd_A)s^{-1} \times G$

Expanding the definition of s from signature step 6,

$C = (z + rd_A)(z + rd_A)^{-1}(k^{-1})^{-1} \times G$

Since the inverse of an inverse is the original element, and the product of an element's inverse and the element is the identity, we are left with

$C = k \times G$

From the definition of r, this is verification step 6.

This shows only that a correctly signed message will verify correctly.

b)

It is also crucial to select different k for different signatures. Otherwise, the equation in step 6 can be solved for d_A, the private key: given two signatures (r, s) and (r, s'), employing the same unknown k for different known messages m and m', an attacker can calculate z and z',

Since $s - s' = k^{-1}(z - z')$(mod n) the attacker can find $k = \frac{z-z'}{s-s'}$.

Since $s = k^{-1}(z + rd_A)$(mod n), the attacker can now calculate the private key $d_A = \frac{sk-z}{r}$.

Problem 5

5 ECDSA Algorithm

We know that given two points $P = (X_P, Y_P)$ and $Q = (X_Q, Y_Q)$ on an elliptic curve $E(a, b)$, to compute the point $R = P + Q(X_R, Y_R)$, we first draw a straight line through P and Q. Next, we find the third intersection of this line with the elliptic curve and then, R is equal to the mirror reflection this intersection about the x-axis. Assume that Δ denotes the slope of the line connecting two distinct points P and Q. More precisely, we have $\Delta = \frac{Y_Q - Y_P}{X_Q - X_P}$. Prove that we have if $P \neq Q$

$$X_R = \Delta^2 - X_P - X_Q$$
$$Y_R = -Y_P + \Delta(X_P - X_R)$$

Drive the relationship for $P = Q$.

Solution:

We have 2 points $P = (X_P, Y_P)$ and $Q = (X_Q, Y_Q)$. We wish to compute $R = (X_R, Y_R) = P + Q$ As stated in the question, $P + Q$ is equal to the mirror reflection of the third intersection of the line with the elliptic curve, let's call it R' about the x-axis. In other words, if P, Q, and R' are the three intersections of the straight line with the curve, then

$$P + Q = -R' \Rightarrow P + Q + R' = O$$

The line passing through P and Q has the form $y = \Delta x + \gamma$, where Δ is the slope of the line and it's given as

$$\Delta = \frac{Y_Q - Y_P}{X_Q - X_P}$$

As we know the elliptic curve $E(a, b)$ is described as below

$$E(a, b) : y^2 = x^3 + ax + b$$

For the intersection of the curve and the line passing through P and Q, we must have

$$(\Delta x + \gamma)^2 = x^3 + ax + b$$

For there to be three points of intersection between the straight line and the elliptic curve, the cubic form in the above equation must have three roots. We already know two of these roots, since they must be X_P and X_Q, correspond to the points P and Q.

Being a cubic equation, since Equation (3) has at most three roots, the remaining root must be $X_{R'}$, the x-coordinate of the third point R'.

We rewrite the equation as below:

$$(\Delta x + \gamma)^2 = x^3 + ax + b$$
$$x^3 - \Delta^2 x^2 + (a - 2\Delta\gamma)x + (b - \gamma^2) = 0$$

The equation represents a monic polynomial (the coefficient of the highest power of x is 1). Therefore we know that the sum of the roots equals the negative of the coefficient of the second highest power. The second highest power(x^2) has the coefficient $-\Delta^2$. Hence, We obtain the following

$$X_P + X_Q + X_{R'} = \Delta^2$$
$$\Rightarrow \quad X_{R'} = \Delta^2 - X_P - X_Q$$

And we can obtain the expression for $Y_{R'}$ from the line formula:

$$\begin{aligned} Y_{R'} &= \Delta X_{R'} + \gamma \\ &= \Delta X_{R'} + (Y_P - \Delta X_P) \\ &= \Delta(X_{R'} - X_P) + Y_P \end{aligned}$$

Since $R = P + Q$ is the reflection of R' about the x-axis, they have the same x-coordinate ($X_R = X_{R'}$), but their y-coordinates have opposite signs ($Y_R = -Y_{R'}$). So, we have

$$X_R = \Delta^2 - X_P - X_Q$$
$$Y_R = \Delta(X_P - X_R) - Y_P$$

Now, consider the case that $P = Q$. Therefore, we're basically computing $2P$ which requires us to draw a tangent at P and to find the intersection of this tangent with the curve. The reflection of this intersection about the x-axis is then the value of $2P$.
Given the equation of the elliptic curve $E(a, b) : y^2 = x^3 + ax + b$, the slope of the tangent at a point (x, y) is obtained by differentiating both sides of the curve equation

$$2y\frac{dy}{dx} = 3x^2 + a$$

Therefore the slope of the tangent is

$$\Delta = \frac{3x^2 + a}{2y}$$
$$\text{at point P:} \quad \Delta = \frac{3X_P^2 + a}{2Y_P}$$

As before, R' is the point of intersection of the tangent with the elliptic curve.

$$(\Delta x + \gamma)^2 = x^3 + ax + b$$

As before, we can use the property that sum of the roots of the monic polynomial above must equal the negative of the coefficient of the second highest power. Noting two of the three roots have coalesced into X_P , we get

$$X_P + X_P + X_{R'} = \Delta^2$$

$$\Rightarrow \quad X_{R'} = \Delta^2 - 2X_P$$

And for $Y_{R'}$ we have

$$\begin{aligned} Y_{R'} &= \Delta X_{R'} + \gamma \\ &= \Delta X_{R'} + (Y_P - \Delta X_P) \\ &= \Delta(X_{R'} - X_P) + Y_P \end{aligned}$$

Again for R we have

$$X_R = \Delta^2 - 2X_P$$
$$Y_R = \Delta(X_P - X_R) - Y_P$$

where $\Delta = \frac{3X_P^2 + a}{2Y_P}$

Problem 6

6 Hash Functions

We use hash function in many applications. Here, we are going to work with this function in order to learn it better. Answer the questions below:

1. Assume you have a hash function which is 2nd-preimage resistant, will this necessarily imply that it is also preimage resistant? Explain and prove your thoughts. (If not, give a counter-example.)
2. Assume you have a hash function which is collision-resistant, will this necessarily imply that it is also 2nd-preimage resistant? Explain and prove your thoughts. (If not, give a counter-example.)
3. Is the reverse true in part 2? Assume you have a hash function which is 2nd-preimage resistant, will this necessarily imply that it is also collision resistant? Explain and prove your thoughts. (If not, give a counter-example.)
4. We use Sha-256 for hashing and might use less significant bits of that. For example, we can construct our hash function as below:

   $$H(x) := SHA - 256(x) \; mod \; 2^{20}$$

   Examples:

   Sha256("Hi")="3639EFCD08ABB273B1619E82E78C29A7DF02C1051B1820E99 FC395DCAA3326B8"

   Sha256("Good Bye")="570398A0EE87C360188291D4116AE9D70183DE80C385F 006133DC90C077C1C60"

   H("Hi") = "326B8"

   H("Good Bye") = "C1C60"

   a) Using online Hash Calculators or a piece of code (Python is recommended), calculate the hash of your *"University ID"* using various algorithms including SHA-256, SHA-512, md2 and whirlpool.

   b) For your university (id), find an integer or sting (t) where $H(t) = H(id)$. (Using Python is recommended.)

Solution:

1. No, it won't imply,

A function that is second-preimage resistant is not necessarily preimage resistant. Let's provide a counter-example.

For example, let $h(x) : \{0,1\}^m \to \{0,1\}^m,\ h(x) = x$

h(x) is collision and second preimage resistant but not preimage resistant.

2. Yes, it will imply,

- Assume that H is collision resistant.
- Assume that H is not second preimage resistant.
  - Given x , we can efficiently compute $x' \neq x$ such that $H(x) = H(x')$.
  - Choose x , then we can find x' as above.
  - We have found a collision (x, x') efficiently, contradiction.
- It follows that H is second preimage resistant.

3. No, it won't imply.

Consider the hash function $h(x) : \{0,1\}^m \to \{0,1\}^m$, if x has less then 24 bits, then h(x)=0, else h(x)=sha256(x). This is very obviously not collision resistant (choose any 2 words that have less than 4 letters), but, as long as your text is longer, this function is preimage-resistant and 2nd-preimage-resistant (assuming sha256 hasn't been broken yet).

You can consider another better hash function h(x) = 0 if x=1 or x =0 , else h(x) = sha256(x).

4)

a)

Assume ID = "400102114", Hash("400102114") is:

1) Using SHA-256:

"fb42e421b20ae948aae85ce400bac216b257a116722d5cf1f12f21a44909d91f"

2) Using SHA-512: "31d246acee5c14f7cec52f770237781b85560abd4ff860b1c97ae9e3b0fd0bf216ca4bbbaceb13436d04fe316f3d7d9fdd64c1d9dd884324ba82a3bdf30e368e"

3) Using md2:

"8b25741c580b8ccfb683cd5e7f155b0c"

b)

Assume ID = “96101165”, the following code finds a t every time it is run. For example, a possible t for this ID is: t = eCil6fWi

Listing 1: Python Script

```
import random
import string
import hashlib as hl

def get_random_string():
    length = random.randint(1, 10)
    letters = string.ascii_letters + string.digits
    result = ''.join(random.choice(letters) for i in range(length))
    return result

value = hl.sha256(b'96101165').hexdigest()
if len(value) > 5:
    value = value[-5:]
while True:
    random_string = get_random_string()
    random_string_hash = hl.sha256(random_string.encode('UTF-8')).hexdigest()
    if len(random_string_hash) > 5:
        random_string_hash = random_string_hash[-5:]
    if random_string_hash == value:
        break
# Writing ans in a file
fileID = open('x.txt', 'w')
fileID.write('96101165' + '\n')
fileID.write(random_string)
fileID.close()
```

Problem 7

7 Bitcoin in action

1. Explore the blocks generated in the last month by visiting https://explorer.btc.com/btc/blocks and exporting the data in an Excel-formatted file.
    - Calculate the average time interval between consecutive blocks.
    - Determine the average transaction fee per block.
    - Compute the average block size.
    - Identify the mining pool responsible for generating the largest fraction of blocks and calculate the percentage. Discuss any concerns regarding this concentration.
2. Discuss the rationale behind setting the difficulty of ”proof of work” in Bitcoin to achieve an average block time of 10 minutes. Explain the potential outcomes if the Bitcoin protocol were designed for an average block time of 1 minute. Is it related to the safety or liveness of the chain?

Solution:

1.1. After clicking on the export button in the provided link, a CSV file will be downloaded. This file can be processed using any preferred tools. To calculate the average of differences between two consecutive cells in the "Time (UTC)" column, follow these instructions:

- Convert the date to a serial number using the formula $= DATEVALUE(LEFT(C2, 10)) + TIMEVALUE(MID(C2, 12, 8))$.
- Apply this formula to all cells in column C by dragging the corner of the new cell in column L.
- Lastly, use the formula $= AVERAGE(L2 : L4500 - L3 : L4501) * 86400$ to calculate the average difference in seconds.

The average time interval between consecutive blocks from September 17, 2023, to October 17, 2023, is approximately $577.9 seconds$, which is roughly $10 minutes$.

1.2. This part could simply be done by applying formula $= AVERAGE(G2 : G4500)$ where G is the column containing $Fees(BTC)$. At the given interval the average transaction fee per block is $0.2BTC$

1.3. Same as previous part we use $= AVERAGE(F2 : F4500)$ which gives us $1688KB$.

1.4. By utilizing the formula $= INDEX(B2 : B4500, MODE(MATCH(B2 : B4500, B2 : B4500, 0)))$, we have determined that the mining pool "Foundary USA" is responsible for generating the highest proportion of blocks. This particular mining pool appeared a total of 1319 times out of the 4519 blocks, accounting for approximately 29% of the total.
Having one mining pool responsible for 29% of the blocks in Bitcoin raises several concerns related to security, and potential centralization of power. When a single mining pool controls a significant portion of the blocks, it increases the risk of a 51% attack.

2. The rationale behind setting the difficulty of "proof of work" in Bitcoin to achieve an average block time of 10 minutes is to decrease the chance of natural forking, which is due to the network delay among the honest parties.

Problem 8

8 Network Partitioning

1. Imagine a scenario where, for a duration of 5 hours, the Bitcoin network is divided or partitioned. During this time, miners in Asia and Europe work on one fork of the blockchain, while miners in other regions of the world work on a separate fork. When the network is reconnected, what outcomes can be expected? Will these forks resolve or merge over time?
2. How exactly k-*depth confirmation* will decrease the possibility of fraud (a transaction gets eliminated from the longest chain)? Can different clients choose different k?

   Solution:

   1. When the Network gets reconnected, we will have a fork resulting in generation of two parallel chains starting from a root (fork point). According to *Longest Chain Rule*, the one with more blocks will be chosen

and in case both chains have equal blocks, we will wait until the next block is generated. With high probability, this time one will prevail since the probability that these chains grow at the same rate will shrink exponentially. Eventually, we will chose the longest chain.
2. As mentioned earlier in the class, The probability of adversary catching up when he is k blocks behind a specific confirmed block is bounded exponentially by e^{-kc}, for some constant c. Thus, choosing bigger k, decrease the chance of fraud.

To answer the second question: Yes, we have flexibility. Each user with respect to its tolerance and risk can have choose different k's.