Project 3

Honya Elfayoumy
helfayoumy3@gatech.edu

1 TASK 2

Since a lot of people like to use common passwords, a suggestion I could implement to improve password security would be to require multifactor authentication. Many applications use this nowadays to increase security measures for their users. It helps protect the user's account by requiring the user to verify their identity through another method instead of just using a password. Passwords are generally "easier" to hack whereas multifactor is more complex to hack. An example of a common multifactor authentication is verification code via SMS. Other methods that can be implemented are biometric scanners (such as fingerprint). Georgia Tech uses Duo to send a one-time token that allows for a second method of verification - this would also be a viable method.

2 TASK 3

Proof of work is a consensus algorithm that is used in kernelcoin blockchain. An alternative to the consensus mechanism is proof of capacity. This system requires more hard drive space typically since it generates plots from datasets stored on the hard drive. As the number of plots increases, there is a higher chance for the algorithm to find the next block in the chain (Hayes, 2021). Whereas with proof of work, the transactions are secured by validating each block every second. Proof of work requires powerful hardware such as CPUs and GPUs (Mahler, 2019).

Strengths (compared to proof of work)

Greener option

More energy efficient

No special hardware needed

Low entry barrier

Weaknesses (compared to proof of work)

Not as popular

Can be infected by malware

3 TASK 4

To implement a function, we want to make sure the digital signature matches the public key of the person. The private key can be validated by checking the public key used to encrypt it. To validate the digital signature, a function can be created to calculate the hash of the data being received and another to decrypt the digital signature using the public key. The two values can then be compared and if they match then the signature is valid. Else, we will know that a different key was used to sign or the data was altered. On top of that, a function to check if the sender has enough coins to send would be needed (How does a public key verify a signature, 2014).

4 TASK 5

It is known that n = p * q and it is assumed that p and q are prime numbers. In this function, p <= q. To start off, I compute the $\sqrt{n}$ and make sure it is an odd number. This is a starting point for finding the value for p. Then, I check through all odd numbers to find the value for p. Once I find the value for p, I can find q by doing n / p. Once I find p and q, I can find the private key by following the steps of RSA as follows:

d * e = 1 (mod phi(p * q))

d = inv(e) (mod phi(p * q))

5 TASK 6

In this task, Linux's RNG system lacks randomness due to the way it's implemented. To make a public key, you need to use p, n, q values made by the system. Sometimes, it can assign the same p value. If this happens, across multiple keys then they share a common denominator. As a result, this information can be used to derive the private key. The public key used in this task is vulnerable because n1 and n2 share a common factor other than 1. Given that scenario, it is easy to find the common denominator. The steps taken to derive the private key in this task was to find the common factor for n1 and n2. If the greatest common factor is not 1, then the private key is discovered (Heninger, Durumeric, Wustrow, Halderman, 2012).

TASK 7

The RSA broadcast attack (AKA Hastad's broadcast attack) is able to determine the original message despite not knowing the private keys. The vulnerability is due to the keys being encrypted using the same small exponent.

According to the RSA broadcast attack, if a message is encrypted by different public keys (n1, n2, n3), it would result in cypher texts (c1, c2, c3). These messages are encrypted with the same exponent, 3. It would look like this:

C1 = $m^3 mod\ n_1$

C2 = $m^3 mod\ n_2$

C3 = $m^3 mod\ n_3$

I solve for m^3 for these equations (Alrasheed, F, n.d.).

m^3 can be found using the Chinese Remainder Theorem (Chinese Remainder Theorem, 2021).

The steps taken were as follows:

1) $m^3 = (C1 * a + C2 * b + C3 * c) \% * (N1 * N2 * N3)$
    a) a % N1 = 1, a % N2 = 0, a % N3 = 0
    b) b % N1 = 0, b % N2 = 1, b % N3 = 0
    c) c % N1 = 0, c % N2 = 0, c % N3 = 1

    Using the rules for a, b, and c with the equation for m^3 will help me find the decrypted message. 0 denotes that it is divisible.

2) I used the Extended Euclidean algorithm to find the GCD. This allowed me to find the correct modular inverses for the below functions:
    a) x[0] = modular inverse (N2 * N3 % N1)
    b) x[1] = modular inverse (N1 * N3 % N2)
    c) x[2] = modular inverse (N1 * N2 % N3)
3) a, b, c are then found by doing the following, and we're multiplying it by Cn ahead of time.

    a) a = x[0] * C1 * N1

    b) b = x[1] * C2 * N2

    c) c = x[2] * C3 * N3

4) Then we take those and take the sum of them all, and modulus them against the product of the N's.
    a) m^3 = (a + b + c) % (N1 * N2 * N3)
5) $\sqrt[3]{m^3}$ is computed using a modified binary search to find the decrypted message (m).

6 REFERENCES

1) Alrasheed, A., & F. (n.d.). RSA Attacks. Retrieved April 04, 2021.
2) Chinese remainder theorem. (2021, March 16). Retrieved April 04, 2021, from https://en.wikipedia.org/wiki/Chinese_remainder_theorem
3) Hayes, A. (2021, March 31). Proof of capacity (cryptocurrency). Retrieved April 06, 2021, from https://www.investopedia.com/terms/p/proof-capacity-cryptocurrency.asp
4) Heninger, N., Durumeric, Z., Wustrow, E., & Halderman, A. (2012, August). Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. Retrieved April 04, 2021, from https://factorable.net/weakkeys12.extended.pdf
5) How does a public key verify a signature? (2014). Retrieved April 04, 2021, from https://stackoverflow.com/questions/18257185/how-does-a-public-key-verify-a-signature
6) Mahler, T. (2019, February 06). Finding consensus 4/4: Alternative consensus mechanisms. Retrieved April 06, 2021, from https://medium.com/blockwhat/finding-consensus-4-4-alternative-consensus-mechanisms-3b8817bd6d57