MSIN-4212 Deciphering Bitcoin

Cryptography Assignment

Take-Home Exam

Daniel Felipe Carrillo Vanegas – Cod. 201215750
Javier Peniche – Cod. 201716772

JUNE 2022

Question 1: What is the difference between an encryption scheme and an encryption algorithm? And what is the difference between encryption schemes in symmetric and asymmetric cryptography with regards to the cryptographic keys used?

An encryption scheme offers a template for encryption based on a core algorithm. The template offers three main components for the encryption and decryption. A key generation algorithm, which generates an encryption key, an encryption algorithm which creates a ciphertext based on the message and the key, and a decryption algorithm which produces the message based on the ciphertext and one key.

Otherwise, the encryption algorithm is just one component of the template offered by the encryption scheme. It refers to the encryption algorithm which is the method used to transform data into ciphertext, based on an encryption key.

Regarding the cryptographic keys the symmetric and asymmetric encryption schemes differs in that symmetric scheme uses the same private key for both encryption and decryption process. In contrast, in the asymmetric scheme, two keys are used in the process of decryption and encryption, while plaintext is encrypted with one key, it must then be decrypted with the other key pair.

Question 2: What are the most common ways in which symmetric encryption schemes are used?

- The first common way is the one in which two or more agents are communicating from a distance and want to keep the content of their communication secret.
  It assumes the agents are the only ones who have access to the private key and that it must not be shared.
- The other common way is these in which one agent wants to keep the contents of a message secret across time.
  In this case one agent just want to encrypt contents at rest and store it anywhere. To retrieve it, it must then be decrypted with the same key used to encrypt.

Question 3: Please consider this statement: “Cryptography is concerned with secrecy.” Is it True or False? Explain your answer

Yes, in the case of the cryptographic scheme symmetric it is supported by the Kerckhoff’s principle, which stands that the security of a symmetric encryption scheme can only rely on the secrecy of the private key. So, it is a matter of secrecy of the private key, all other security aspects of the symmetric encryption are assumed that is known by the attackers. In the case of asymmetric scheme, it is concerned with the secrecy of the private key

generated and stored only by the main agent which requires to communicate to many agents.

Question 4: What is a stream cipher? What specifically is a primitive stream cipher? Is RC4 an example of a primitive stream cipher?

A stream cipher is a symmetric encryption scheme. Hence, it consists of:

- The private key
- The encryption algorithm which produces a keystream with the same length of the plaintext with the aid of the private key. The plaintext is then combined with the keystream to produce a ciphertext, typically an XOR operation.
- The decryption algorithm which is the reverse operation of the encryption.

A primitive stream cipher are the stream ciphers that are not created from block ciphers as it is for example the case of the OFB mode or Rijndael.

Yes, RC4 is an example of a primitive stream cipher

Question 5: What are the key distribution and key management problems? How are these addressed by asymmetric cryptography?

The symmetric encryption scheme requires both entities that wish to use encrypted communications to have the same private key. Whenever private keys are not required to be shared publicly, such as the internet, this works perfectly well. However, when one of the entities require to communicate with not just one, but many other entities, which are not necessarily geographically close and the use of a public channel such as the internet is needed, a problem arises, this is known as the **key distribution problem**. There could be some solutions to this problem like couriers delivering the key to the other entities, but it is not a scalable and a sustainable solution since the number of entities grow and the distance from the main entity increases.

Otherwise, having the same private key for all the entities is not secure, so it is necessary to create a private key for each customer. Consequently, the main entity must store the same number of keys as the number of clients it has, secure them at rest, manage its lifecycle, and secure the keys on the other entities' end. This is known as the **key management problem**.

These problems are presented in such one-to-many communications where the symmetric scheme is used. The **asymmetric cryptography** scheme solved this by using two keys, one private and one public, instead of just one private key. The idea is that private key is stored in the main entity and will be used to both encrypt and decrypt and the public key will be

shared to the public which request the communication. If the message is decrypted with whether the private key, then it must be decrypted with the other one. In this case, the main entity just stores and protect one key, the private key, while the other entities received the public key via a public network as the internet and stored it locally without no consequences if it stolen as it is public. The private key on the main entity side assures the secure communication, and this is how asymmetric cryptography addressed the key distribution and key management problems.

Question 6: What two types of computationally hard problems is most of asymmetric cryptography based on? Specifically, what is the computationally hard problem on which Bitcoin's digital signature schemes are based?

The two types of hard computationally hard problems is most of asymmetric cryptography based on are *prime factorization* and *the calculation of discrete algorithms*.

Bitcoin's digital signature schemes are based on the discrete logarithm problem for a particular elliptic curve cyclic group.

Question 7: What are the main two properties required of hash functions in cryptographic applications? Can you give an example for each property of how it is used in practice?

1. Collision-resistance: a hash function H is collision-resistance if it is infeasible to find two values, x and y, such that $x \neq y$, yet $H(x) = H(y)$.

   In software verification: This is the case for a software package which can be downloaded via a web page, and it is provided with a digital signature. It helps to ensure that the downloaded package was not modified or has malicious code put by an attacker. The way it works is that official package is provided with its respective digital signatures as well as the executable generated by the contributors. The public keys should be downloadable from the official contributors' page and with these, the hash can be generated. Assuming a SHA-256 hash and after generating the hash of the executable with the public keys, the hashes can be compared, and the result must be the same. Because the SHA-256 hash function is collision-resistant, the hash generated and the hash in the package downloaded must be the same, otherwise, the package was modified.

2. Hiding: a hash function has this property if for any randomly selected x from a very large range, it is infeasable to find x when only given H(x).

   In password management: platforms which manages user's login and register. Those platforms store in the database a hash value of the password that the user typed in

the register. When the user requests a login, the hash is calculated again with the password the user just sent and compared with the one store at the first time. In this way, the attackers cannot neither calculate nor get the original password of the user.

Question 8: Suppose that you and your team have created new Bitcoin wallet for desktop computers. It looks really fantastic. However, your team does not have a recognized TLS certificate from a certificate authority. How could your team release this wallet using digital signatures, so that you can improve user security in downloading and installing it? Please broadly describe the steps you would take.

Step 1:
Firstly, we need to publish our public keys in an official site, like the official GitHub or our official website.

Step 2:
Then, we need to generate the wallet's executable release, and from this executable we generate the SHA-256 hash as well as the signatures of the contributors.

Step 3:
We need to package the three files, the executable, the hash, and the signatures and make the package publicly downloadable from our official webpage or repository.

Step 4:
Now the user can download the package, as well as the keys. After verifying the keys they are able to generate the SHA-256 hash in order to verify the executable authenticity and integrity.