Cryptography for the Blockchain #1: Introduction and RSA

Ihab McShea

Cryptographic systems are of particular interests to blockchains, and I will start a little journey with you, from a **research** point of view, rather than a development one, to explore and get to know more about the blockchain cryptography-wise, to be more specific: I want to introduce you to zkSNARKS but I want to first introduce you to fundamental concepts of cryptography, and perhaps a little bit about blockchains, if you are not already familiar with such concepts. Cryptographic systems we consider in this article are divided into two categories; privacy systems and authentication systems. The first of which is going to be described in detail in the next section, the latter is going to be introduced later on as we go further.

Privacy System

A privacy system is one that involves three parties: a transmitter, a receiver, and an intruder (or eavesdropper). The transmitter generates a plain-text message or unciphered message **P**, the transmitter operates on P, with an invertible transformation S_k to produce a cyphertext $C = S_k(P)$. The key is transmitted to the receiver through a secure path, and since the legitimate receiver already knows the key, she can decipher C by operating with the inverse operation S_k^{-1}, to obtain $S_k^{-1}(C) = S_k^{-1}(S_K(P)) = P$, i.e: the original plaintext message. We can't send the original message through the secure channel for capacity and delay related reasons.

A **Cryptographic System** is a single parameter family $\{S_k\}_{k\in|k|}$ where $S_k : \{P\} \rightarrow \{C\}$, a mapping from the plaintext message {P} to the cryptograms {C} , and if {P} = {C} then we denote them both by {M}.

The main goal of designing the cryptosystem $\{S_k\}$ is to make deciphering and enciphering inexpensive, but also ensure that cryptanalysis is too complex to be reasonable, or economical. To achieve that we need to consider certain approaches, two of which are:

1) A system which relies on complicated and expensive computations which could succumb to an attack of unlimited resources. That is a **computationally secure system.**
2) A system that resists any type of cryptanalytic operation. **Unconditionally secury system**

Unconditional security results from the existence of multiple meaningful solutions to a cryptogram, while a computationally secure system has succificient information to uniquely identify the plaintext and key. It security resides only in the computational effort, or simply put: the computational complexity.

A **computationally infeasible task** is one whose cost is memory or runtime cost is finite but impossibly large, we will explore examples of computationally infeasible tasks, some of which we are going to call NP-problems or sometimes even NP-complete problems, but for now, a popular example would be a variation of the Knapsack problem, and that is the subset sum problem, which is one of the most important problems in cryptography and complexity theory equally, for now let's just say that it is this; given a set of integers, is there a non-empty subset that sum up to zero? For example consider the following set {-5,-4,9,2,1,5,3}, which, to answer the previous question; yes, there exists a subset that sums up to zero, and it's quite obvious, the subset {-5,-4,9} sums up to zero. The problem is NP-complete, and if you don't what that is, it is simply a decision problem [simply, a binary problem that could be answered with either yes -1- or no -0-], more on that later.

Before we continue, we should introduce some important terminology, and first things first, what are the different types of ciphers? First, we have the **stream ciphers** which process the plain text in small chunks (characters or bits), usually producing a pseudo-random sequence of bits that are added together modulo 2 to bits of the original plain text.

The second type is **block ciphers**, which act in purely combinatorial way on large blocks of text, in such a way that small chngess in the input block produces a major change in the output.

0.1 Attacks on Cryptographic Systems

Since encryption is very vital to protect "messages" from predator attacks, cryptosystems are an attractive target for attackers, and in this section we will explore briefly a good number of those attacks.

The first of which is the **Ciphertext-only attack**, which is the least in danger for users, and the most difficult for cryptanalysts to use against users of the system; all the attacker knows is the cyphertext, and maybe she might suspect some important information that are included in the actual plain text message. A smart attacker might by the use of several pieces of a ciphertext, look for certain trends or statistical data, which might lead her to a solution.

A **known plaintext** attack is more dangerous, since the attacker knows both the cipher-text the entire original plain-text message, the main goal of such an attack is to find the key that was used to encrypt the original plaintext message into its corresponding cryptogram, and obviously once the key is found, the attacker is able to decrypt all messages that had been encrypted using the cracked key.

A **Chosen ciphertext** attack is one in which the attacker knows the decryption algorithm, or has access to the decryption device, and is trying to decrypt texts of her own choice to discover the key. Asymmetric encryption systems are the most vulnerable to such an attack. For example, the RSA algorithm is vulnerable to this attack; an attacker would select a certain part of the plain text, encrypt it using the victim's public-key, then try

decrypt the resulting cryptogram to get the original plaintext.

And finally -for now-, symmetric cryptosystems are ones in which the sender and receiver share in secret, the same key, or two keys, each can produce the other easily. An asymmetric cryptosystem is one in which the sender and receiver have different keys, and it is computationally infeasible to produce one from the other. For now, we have been introduced to enough concepts that will make the next section much easier to grasp. RSA is an encryption/decryption algorithm that was presented by R.L. Rivest, A. Shamir, and L. Adleman, whose main purpose was that publicly revealing an encryption key does not reveal the private decryption key.

To encrypt a message M using a public encryption key (e,n) we need to first represent the message as an integer between 0 and (n-1), i.e: break a long message into a series o blocks, and represent the block as integer, then encrypt the message by raising each block to a public power e modulo n. To decrypt the message, we will have to raise it to a power d, that is the exponential inverse of e. For starters, n should be the product of two large prime numbers p and q, because since n is public, we should make factorization of n into p and q hard. Also, d, the private decryption key should be a large prime number which is relatively prime to $\phi(n) = (p-1) \cdot (q-1)$. The integer e, again, the public encryption key, is computed from p,q and d, by a variation of GCD, observe the following process:

Example.

Suppose p = 47, q=59, n= $p \cdot q$ =2773, and d= 157, again check the PDF for a more visually pleasant representation of this example. We have to first compute e, and we do so by computing the GCD of $\phi(n)$ and d, with the addition of two variables a_i , b_i

First, let $x_0 = \phi(n)$ and $x_1 = p$

and to better understand the role of the a's and b's, consider $x_i = a_i x_0 + b_i x_1$, and to compute each $x_i = x_{i-2} mod\, x_{i-1}$

$x_0 = 2773 \quad a_0 = 1 \quad b_0 = 0$
$x_1 = 157 \quad a_1 = 0 \quad b_1 = 1$
$x_2 = 156 \quad a_2 = 1 \quad b_2 = -16$
$x_2 = 1 \quad a_3 = -1 \quad b_3 = 17$

Therefore $e = b_3 = 17$.

Now, let's encypher LA MARKAZ, the space was put deliberately, First, we need to encode the name by representing each block as an integer, and since we only have letters and spaces we will represent the space as 00, and each letter as its order in the alphabet, e.g: A= 01.. Z=26, and so on. Since our n has 4 digits, let's suppose a block is two letters (or four digits), the encoding hence is:

1201 0013 0118 1101 2600

By raising each block e modulo n we get the encrypted version: (e.g: $(1201)^{17} mod\, 2773 = 2526$)

2526 0219 1239 1919 0499

which is utter nonsense.

The inverse? Well, the receiver already has the private decryption key d, and we can observe that $(2526)^{157}$ mod 2773 = 1201, and so on, which technically "proves" our system both encrypts and decrypts messages.