# Mathematical Models for Encrypted Internet Communication

Ryan Jensen

November 9, 2014

**Abstract**

Any encryption methods employed for public use on the Internet must maintain their integrity while respecting the following three axioms: (i) all communications can be intercepted and modified in transit; (ii) all algorithm's details must be publicly available to users and attackers alike; and (iii) no communication may bypass the network to trade information secretly at any point in the correspondence. Although these axioms rule out all symmetric encryption systems as potential candidates, advanced number theory and asymmetric encryption systems can provide the initial steps to start an encrypted session and allow symmetric encryption to take over at that point. More specifically we will be exploring the RSA Cryptosystem and its implications on a private Internet, digital authenticity, and using mathematical models to show how and why it works.

## 1 Introduction

**The goal** of this paper is to ease the reader into the complex subject of encrypted Internet communication. First we will attempt to shed some light on the challenges of achieving a private, encrypted *infrastructure* on a public network (meaning the solution must work on the scale of society, not individuals) and look into a particular solution to the problem at hand–The RSA Cryptosystem. The context of this paper is the modern day Internet, which we will define loosely as a network of interconnected computers which are both utilizing the network (consuming) and sustaining the network (producing). In order to communicate with any other person or computer on the network, our traffic must necessarily be routed from one computer to another until it arrives at the correct destination; think of the network as

a graph, the computers as the vertices and the connections between particular computers as the edges. Although most of the computers a message is relayed between are Internet service providers, commercial companies, or benign users, the troubling fact is that any one of them could be a malicious attacker looking to access and exploit the information that passed through their computer on its way to the intended destination. We will refer to this type of an attack as a *man-in-the-middle attack*.

**Encoding** schemes are a necessary step to encryption since most encryption algorithms rely on math functions that operate only on numbers. Typically a message is made up of a string of characters or *glyphs* so we will need a function that maps a glyph to a particular integer. We would call such a function an *encoding* of a set of characters; to map from an integer back to a glyph is a *decoding* function. A simple example of such an encoding would be to let $A = 01, B = 02, ..., Z = 26$; now we can get from characters to integers interchangeably. Common encoding schemes used on the Internet are *UTC-8* and *UTC-16* which encode $2^8$ and $2^{16}$ different characters respectively.

**Historically** encryption is introduced with a mention of the *Caesar Cipher* or the *shift cipher*; This is a basic form of encryption where you take each encoded character and add a specified amount to the value and wrap values too high back to the beginning (modulus); we could also conceptualize this operation as a character shift. We will define our *secret* or our *key* to be the piece of information required to encrypt or decrypt messages. For the shift cipher the key would be the number of characters we shifted. The shift cipher is a very basic example of a *symmetric* encryption system, which means it uses the same key for encryption and decryption at both *endpoints* (the sender and receiver of the encrypted communication). By contrast, *asymmetric* encryption systems have a key or secret for each endpoint; each key has two components: one for encryption and one for decryption. In total, a encrypted conversation between two endpoints, denoted $A$ and $B$, using an asymmetric system would have four pieces: encryption key from $A$ to $B$, encryption key from $B$ to $A$, decryption key for $A$, and a decryption key for $B$.

**The Internet** is to reiterate, an open network where all communication between two endpoints can be intercepted by way of a man-in-the-middle attack. To achieve encrypted Internet communication we must assume that all messages will be intercepted; there is no room for optimism in this field of study. With this assumption in mind, the use of symmetric encryption requires that the secret key is already established at both endpoints, since any

attempt to transmit the key over the network compromises the key's secrecy. An alternative scenario is to have a non-disclosed encryption algorithm ready at both endpoints so that establishing the key over the network wouldn't compromise the security, since an attacker wouldn't know what to do with the key they intercepted. Both of these scenarios work fine for personal, specialized uses but fail to hold up when we talk about encrypted public infrastructures. As soon as one needs to communicate with someone new at a different endpoint, we either need to have a new secret algorithm or have a secret key established *before* any correspondence over the network begins. It is unfeasible to establish such a system between every combination of endpoints on the Internet, which is exactly why symmetric encryption is not enough to create an encrypted public infrastructure. To achieve this goal, we must look elsewhere.

# 2 The RSA Cryptosystem

**The history** of the RSA Cryptosystem dates back to the mid 1970s; a period when the government was attempting to suppress encryption algorithms from being released due to Cold War tensions, while the transition to computerized systems in the commercial and financial sector opposed such suppression. In 1976, Whitfield Diffie and Martin Hellman coined the term *public key cryptosystem* to describe the idea of releasing enough information so that others can encrypt messages to you, while maintaining the secrecy of the decryption key. In such a system the encryption key can be released to the general public and thus named the *public key*; the decyption key should be kept private so it is called the *private key*. The function to get from the private key to the public key should be a *trap-door* function or a *one-way* function, which means the computation is simple in one direction and difficult in the other. For the basic RSA algorithm the trap-door function is that multiplication of two prime numbers is easy relative to the factoring of a large composite number with only the two prime factors. To reiterate the point, in RSA, the deriving of the public key from the private key should involve the easy computation, multiplication, and the deriving of the private key from the public key involves the difficult computation, factoring.

**The development** and release of the RSA algorithm came just a year after Diffie and Hellman conceptualized the public key cryptosystem in 1977. The algorithms creators Ron **R**ivest, Adi **S**hamir, and Leonard **A**dleman made concrete the work that Diffie and Hellman had started in the year before but not implemented. Each endpoint creates a public key $P$

and a private key $S$; $P$ will published by its owner so that others may encrypt messages to them which can be decoded using the private key $S$. In practice, the system is obviously more complicated but the essence is just that–simple asymmetric encryption. The method works well on the Internet since now each endpoint $A$ and $B$ have their key pairs $(P_A, S_A)$ and $(P_B, S_A)$ respectively, they can send each other their public keys which can be used to encrypt messages back to the owner of the key who has the decryption key ready to decrypt the messages they receive. Without lose of generality, suppose endpoint $A$ would like to send a message to endpoint $B$ through network $N$. Suppose the public keys have already been traded (no encryption necessary since these are public). We will let $M$ be a message that has been encoded using a shared encoding scheme and $C$ will denote our *encrypted cipher-text* which is the encrypted text that can be sent through the public network without its original meaning being understood by a man-in-the-middle attacker. If we conceptualize the application of the public or private key to a message or cipher-text as a function the correspondence would be as follows:

| | |
|---|---|
| | Message being sent from $A$ to $B$ over network $N$ |
| $P_B(M) = C$ | $A$ encrypts message $M$ using $B$'s public key |
| $C : A \to N$ | $A$ sends the encrypted cipher-text $C$ over the network $N$ |
| $C : N \to B$ | $B$ receives cipher-text $C$ from network $N$ |
| $S_B(C) = M$ | $B$ applies their private key to get back $M$ |

Public key cryptosystems are conceptually very simple and satisfy the constraints we defined when working on a public network for widespread communications. It is important to point our that since $P_B(M) = C$ and $S_B(C) = M$, then $S_B(P_B(M)) = M$ and $P_B(S_B(M)) = M$ which implies the keys work as inverses of each other; we will formally prove this to be the case when we prove the correctness of RSA. Now that we have the concept in mind, we can lay out the definitions and mathematical implementation required to work through an example and prove its correctness.

# 3 Mathematical Implementation of RSA

**Implementation** of the RSA cryptosystem is quite simple. First we generate two $s - bit$ prime numbers where $s$ is the length of the binary representation of the prime numbers; we use binary since this process is typically done on computers and the length of the prime

numbers correlates to the level of security; cytologists would typically regard $512-bit$ primes as lower bound of acceptable sizes. We will denote the two $s-bit$ prime numbers $p$ and $q$ respectively. Next we compute $n = p \cdot q$ where $n, p, q$ make up our trap-door function (since computing $n$ from $p, q$ is easy but computing $p, q$ from $n$ alone is not). Now we define a new function *Euler's Phi Function*, $\phi(n)$ which computes the number of relatively prime integers $n_i \in \{1, ..., n\}$, that is, less than or equal to $n$.

**[Note:]** for the remainder of the paper, we will make ***heavy*** use of the *modulus* operator. For $n \in \mathbb{N}, x \in \mathbb{Z}, \quad x \pmod{n} \in \{0, ..., n-1\}$ and can be described conceptually as how much larger $x$ is than a multiple of $n$. For example, $1, 4, 7, ..., 3 * x + 1$ are equivalent ($\equiv$) or congruent to, $1 \pmod 3$. More formally we would say that if $x \equiv y \pmod{n}$ then $x - y = nz$ for some $z \in \mathbb{Z}$ and $y \in \{0, ..., n-1\}$.

**Our public key cryptosystem** works by taking our encoded message to certain exponents modulus $n$. We will have one encryption exponent $e$ which is part of our public key and one decryption exponent $d$ which is part of our private key. The choice of $e$ is constrained only by the fact that it should be relatively prime to $\phi(n)$; to achieve this we need only ensure that $\gcd(e, \phi(n)) = 1$. Common choices for $e$ are $\{3, 5, 17, 257, 65537\}$, but the industry standard, if one exists, would be to choose $e = 65537$ for maximum compatibility. Now we will find $d$ by solving the linear congruence $ed \equiv 1 \pmod{\phi(n)}$. The *Linear Congruence Theorem* states that $ax \equiv c \pmod{m}$ has $g$ solutions if $g \mid c$, where $g = \gcd(a, m)$. So in our case we are guaranteed a single unique solution since $\gcd(e, \phi(n)) = 1$ by our choice of $e$ and $1 \mid 1$ which proves we can find a unique $d$ so that $d = e^{-1} \pmod{\phi(n)}$.

Now that $e, d, n$ are all chosen we can define the public key we publish to be $P = (e, n)$. The encryption function $P(M) = M^e \pmod{n}$, applies the public key to an encoded message $M$ and creates encrypted cipher-text $C$. Now we define the private key kept secret as $S = (d, n)$. The decryption function $S(C) = M^d \pmod{n}$, applies the private key to encrypted cipher-text $C$ and returns an encoded message $M$.