Mathematical Models for Encrypted Internet Communication

Ryan Jensen

November 10, 2014

Abstract

Any encryption method employed for public use on the Internet must maintain its integrity with respect to 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. We will show these axioms rule out all symmetric encryption systems as potential candidates for Internet use, but advanced number theory and public key cryptosystems systems can provide the initial steps needed 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 analyze its implications on a private Internet, digital authenticity, and use 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 at arbitrarily large scales) 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 with the computers as the vertices and the connections between particular computers as the edges. This graph will rarely be *complete* on any network, meaning we must travel through central hubs to communicate with *any* other computer. 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 is passed through their computer on its way to the intended destination. We will refer to this common type of Internet attack as a *man-in-the-middle attack*.

Encoding schemes are a necessary step toward encryption since most encryption algorithms rely on mathematical 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 each 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 upper-case 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 allowing for a more expansive set of glyphs and languages.

Historically encryption is introduced with a mention of the *Caesar Cipher* or the *shift cipher*; This is a basic form of encryption that takes each encoded character and adds a specified amount to the value then wraps values too high for our encoding back to the beginning (modulus); we could also conceptualize this operation as a character shift to the right for encryption and to the left for decryption. 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, meaning 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 that is standardized yet somehow supposed to be secret. This 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. Simultaneously the financial and commercial sector were transitioning to computerized systems and opposed such suppression for the sake of data security; they would eventually win the debate. 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 in 1977, just a year after Diffie and Hellman conceptualized the public key cryptosystem. The algorithms creators Ron **R**ivest, Adi **S**hamir, and Leonard **A**dleman made concrete what Diffie and Hellman had conceptualized 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 a simple asymmetric encryption scheme. 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 pair, who is waiting with a private key capable of decrypting the received message. 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 published publicly). 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 and/or private key to a message and/or cipher-text as a function, the correspondence would be as follows:

	Message M 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 out 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; later 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 such that $n_i \in \{1, ..., n\}$, that is, the number of relatively prime numbers 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. In order to show that the functions we defined are correct, we must prove that P and S are in fact inverses of each other or that, $S(P(M)) = M$ and $P(S(M) = M$.

4 Mathematical Proof of Correctness

The following background information will be necessary to prove that P and S are inverses of each other for all possible encoded messages M we could create.

Theorem 4.1. Fermat's Little Theorem *Let p be a prime number, and let a be any number with $a \not\equiv 0 \pmod{p}$. Then $a^{p-1} \equiv 1 \pmod{p}$.*

Theorem 4.2. (Euler's Formula) *If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$*

Theorem 4.3. Chinese Remainder Theorem *If $n_1, n_2, ..., n_k$ are pairwise relatively prime and $n = n_1 n_2 ... n_k$, then for any integers $a_1, a_2, ..., a_k$, the set of simultaneous equations*

$$x \equiv a_i \pmod{n_i}$$

for $i = 1, 2, ..., k$, has a unique solution modulo n for the unknown x.

Corollary 4.4. *If $n_1, n_2, ..., n_k$ are pairwise relatively prime and $n = n_1 n_2 ... n_k$, then for all integers x and a,*

$$x \equiv a \pmod{n_i}$$

for $i = 1, 2, ..., k$ if and only if

$$x \equiv a \pmod{n}$$

.

Now we have all of the necessary components to formally prove the inverse nature of P and S over the set of all messages M.

Definition 4.1. *For all $M \in \mathbb{Z}$, for all positive, prime integers p, q, and for $n = pq$. If $e \in \mathbb{N}$ and $\gcd(e, \phi(n)) = 1$ and $d = e^{-1} \pmod{\phi(n)}$ then $M^{ed} \equiv M \pmod{n}$.*

Proof. Let $M \in \mathbb{Z}$, let p, q be positive, prime integers. Define $n = pq$ and compute $\phi(n) = (p - 1)(q - 1)$. Next pick $e \in \mathbb{N}$ so that e is relatively prime to $\phi(n)$ or that $\gcd(e, \phi(n)) = 1$. By the linear congruence theorem the equation $ed \equiv 1 \pmod{\phi(n)}$ has exactly $g = \gcd(e, \phi(n))$ solutions if $g \mid 1$; thus $d = e^{-1} \pmod{\phi(n)}$ exists and is unique

since $g = 1$ and $1 \mid 1$.
Consider the congruence $M^{ed} \pmod{n}$, since $ed \equiv 1 \pmod{\phi(n)} \implies \phi(n) \mid ed - 1 \implies ed = \phi(n)z + 1$ for some $z \in \mathbb{Z}$ then $M^{ed} \pmod{n} = M^{\phi(n)z+1} \pmod{n}$. Now we will break the proof into three exhaustive cases and prove each on individually:

Case 1: Suppose $n \mid M$. Then $M = nx$ for some $x \in \mathbb{Z}$ and $M \equiv nx \equiv 0 \pmod{n}$, then $M^{ed} \equiv (nx)^{ed} \equiv 0^{ed} \equiv 0 \pmod{n}$. Thus $M^{ed} \equiv M \equiv 0 \pmod{n}$. $\square$

Case 2: Suppose $n \nmid M$ and $\gcd(n, M) \neq 1$. Since $n = pq$ with p, q prime then $p \mid M$ or $q \mid M$. Without lose of generality suppose that $p|M$. By theorem 4.4 if $n = pq$ and $\gcd(p, q) = 1$ then if $M^{ed} \equiv M \pmod{p}$ and $M^{ed} \equiv M \pmod{q}$ implies $M^{ed} \equiv M \pmod{n}$. As in case 1, since $p \mid M$ then $M^{ed} \equiv M \equiv 0 \pmod{p}$. Now it suffices to show that $M^{ed} \equiv M \pmod{q}$. By our choice of e, d then $M^{ed} \equiv M^{\phi(n)z+1} \equiv M^{z(p-1)(q-1)+1} \pmod{q}$ since $\phi(n) = (p-1)(q-1)$. By the laws of exponents $\equiv M^{z(p-1)(q-1)+1} \equiv M \cdot (M^{(q-1)})^{z(p-q)} \pmod{q}$. By theorem 4.1 $M^{(q-1)} \equiv 1 \pmod{q}$ implies $M \cdot (M^{(q-1)})^{z(p-q)} \equiv M \cdot (1)^{z(p-q)} \equiv M \pmod{q}$. Since $M^{ed} \equiv M \pmod{p}$ and $M^{ed} \equiv M \pmod{q}$ then $M^{ed} \equiv M \pmod{n}$ by theorem 4.4. The proof if $q \mid M$ follows the same form with q and p's place switched. $\square$

Case 3: Suppose $\gcd(n, M) = 1$. By theorem 4.2 if $\gcd(n, M) = 1$ then $M^{\phi(n)} \equiv 1 \pmod{n}$. Thus $M^{ed} \equiv M^{\phi(n)z+1} \equiv M \cdot (M^{\phi(n)})^z \equiv M \cdot (1)^z \equiv M \pmod{n}$. $\square$

Now we have shown that for all $M \in \mathbb{Z}$ $M^{ed} \equiv M \pmod{n}$ by proving it for three different cases which are exhaustive over the integers. $\square$

Using definition 4.1, we can prove the correctness of the RSA cryptosystem by showing that $S(P(M))$ is equivalent to $M^{ed} \pmod{n}$ and correctly applying definition 4.1. To reiterate $P(M) = M^e \equiv C \pmod{n}$ and $S(C) = C^d \equiv M \pmod{n}$. Composing the functions gives $S(P(M)) = (M^e \pmod{n})^d \pmod{n}$. Since the base of both modulus functions equals n then $(M^e \pmod{n})^d \equiv M^{ed} \pmod{n}$. By the specification of RSA, $n = pq$ and $e \in \mathbb{N}$ with $\gcd(e, \phi(n)) = 1$ and $d = e^{-1} \pmod{\phi(n)}$ so now we can apply definition 4.1 and state that $M^{ed} \equiv M \pmod{n}$ for all $M \in \mathbb{Z}$.

5 Conclusion

In practice the RSA algorithm for encryption and decryption is quite slow when compared to most symmetric encryption algorithms that perform computations computers are better equip to handle. This fact doesn't make the RSA cryptosystem any less revolutionary or important; it just means that the RSA algorithm is only a piece of the *full* solution. Technologies looking to provide encryption on a public network that can be expanded to any combination of endpoints, typically use the RSA cryptosystem to establish what is often described as a *handshake*. One can conceptualize the handshake as the asymmetric portion of the encrypted correspondence which includes: connecting to the other endpoint, exchanging and authenticating the public keys with the other endpoint, then establishing a symmetric secret key often known as the *session key.* The point that the session key is established, the handshake is officially over and an *encrypted session* has now begun which utilizes the speed of symmetric encryption using the shared session key. This further explains the use of the work cryptosystem; the RSA cryptosystem provides the best of both asymmetric and symmetric encryption by allowing a medium to safely authenticate and transfer the session key over the public network; once the session is established, the session key is used by both endpoints for encryption and decryption until the session is terminated. As long as a private key is kept safe, the key can be reused for as many sessions as desired, whereas the symmetric session key should be created at random for each encrypted session with a new endpoint.

In addition, the asymmetric properties of the RSA cryptosystem allow it to excel at allowing for digital authentication. Two measures of authentication of a message are confirming its authorship and confirming that it is untampered by a man-in-the-middle attack. Without these measures, a clever man-in-the-middle could send messages impersonating the endpoint we believe we are communicating with or change some of the contents of a message they intercepted before relaying it the rest of the way to the desired recipient. Both of these attacks are common ways to commit fraudulent money transfers or steal someones personal records from a large company. Remember that conceptually, the private key function S can be applied to a message just the same as the public key function P; the difference in the two is that for a given message M,anyone with P can create $P(M)$ while only the owner of the key S can create the cipher-text $S(M)$. The cipher-text from applying a private key to a message *before it is sent* is called a digit signature. It is uniquely tied to the particular message M and the particular key S so without both of those elements it is impossible to forge. The digital signature's cipher-text can always be undone using the published public key P which allows the recipient of the message/signature pair $(M, S(M))$ to verify that the message was authored by the owner of the key S and untampered by a man-in-the-middle. Both M and $S(M)$ can still be altered by a man-in-the-middle however, it is unfeasible that the change they made to one, the other, or both will match up correctly after the intended recipient applies the public key to the signature and compares the two.

Going forward the arms race between prime factorization algorithms and encryption with larger and larger primes rages on. The larger the prime, the slower encryption and decryption will be, but it also increases the time it takes to reverse the trap-door function between multiplication and factorization. The issue with this particular trap-door function is that it gets weaker and weaker as the prime numbers we use get larger since prime factorization algorithms becomes more efficient as numbers get larger, while multiplication gets less efficient as numbers get larger. If we view the security of RSA as an arms race, eventually as computing power increases into the future, factorization is going to win out over multiplication and RSA will no longer be secure. Right now, the future of the RSA algorithm is elliptical curve cryptography. Using the properties of an elliptical curve in the form $y^2 = x^3 + ax + b$ we can create a trap-door function that remains constant with respect to the size of the numbers used. This is not the only benefit; one can gain an equal level of encryption as the original RSA algorithm using much smaller numbers. An 228-bit key in elliptical curve cryptography gives the security roughly equivalent to a 2380-bit key in RSA. So although the original RSA cryptosystem published by Ron Rivest, Adi Shamir, and Len Adleman may dissappear from public use not too far into the future and be replaced by elliptical curve cryptography, the concepts of the public key cryptosystem published by Whitfield Diffie and Martin Hellman in 1976 won't be going anywhere anytime soon as long as anyone wants to use the Internet for encrypted communication.

References

[1] Cormen, Thomas H. "Number-Theoretic Algorithms." *Introduction to Algorithms.* 3rd ed. Cambridge, Mass.: MIT, 2009. Print.

[2] Morrow, Jim. "The RSA Algorithm." University of Washington, 3 June 2009. Web. 3 Nov. 2014. <`https://www.math.washington.edu/~morrow/336_09/papers/Yevgeny.pdf`>.

[3] Shapley, D., and G. B. Kolata. ”Cryptology: Scientists Puzzle Over Threat to Open Research, Publication.” *Science* (1977): 1345-349. JSTOR. Web. 3 Nov. 2014. <`http://0-www.jstor.org.wncln.wncln.org/stable/1744699`>.

[4] Silverman, Joseph H. ”Powers Modulo M and Successive Squaring.” *A Friendly Introduction to Number Theory*. 3rd ed. Upper Saddle River, New Jersey: Pearson Prentice Hall, 2006. Print.

[5] Sullivan, Nick. ”A (relatively Easy to Understand) Primer on Elliptic Curve Cryptography.” *Ars Technica*. 24 Oct. 2013. Web. 10 Nov. 2014. <`http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/3/`>.