Travis Peters
Western Washington University
Math 419: Historical Perspectives of Mathematics

What's the Key?

An Exploration of Historical Security Approaches & Public Key Encryption

Travis W. Peters

8/7/2012

Contents

Table of Figures

Introduction

Throughout history and our own lives we have witnessed many cycles. Commonly known examples can be seen in planetary cycles or even life cycles, the common factors being that there are definite outcomes (consistent motion and death, respectively) which we know only come as a result of some preceding event in the solar system or existence of life. Another example is what I refer to as the 'cycle of problem and solution'. This cycle, demonstrated in the writing that follows, has special significance in the area of Public key cryptology. Public key cryptology is a special scheme which exists within the problem domain of encryption. Upon further investigation of the problems associated with encryption we see methods of security created to protect sensitive information that, as we will see later, are ultimately rendered useless due to the sophisticated advances in mathematics and computing used to design solutions to the original problem. I intend to argue that security approaches, which often result directly from developments in mathematics and computing technologies, will inevitably equip attackers with both the tools and knowledge necessary to overcome the same security approaches. In turn, a vicious cycle is created that will eternally prevent the advent of truly dependable security of information.

Early History of Computing

Understanding even a brief history of the development of computing systems will allow us to first see significance in the area of data encryption, then, more importantly, will lay the foundation for exploring public key encryption as a possible approach to information security. Before delving into the definitions of some of the terms above and the analysis of modern security approaches, we will first begin in the 1940s where we see the first general purpose computers come into production. The first computers were large enough to fill entire rooms, consumed large amounts of energy, and often broke down within a few minutes of operation. In addition to these wonderful characteristics, they weren't even easy to use! The technology was so new and complex that interaction with computers was limited to people who understood the cryptic methods of using paper tape or punched cards and a typewriter to tell computers what to do. Even if someone was able to communicate with a computer, it was an entirely different battle

to interpret computer output via cards, lights, and switches to try to comprehend what a computer was saying in response to some sequence of machine instructions. Nevertheless, the sheer computing power of these machines justified their usefulness and issues with usability and reliability were greatly corrected in the following years. Machine code was, in time, replaced with higher level programming languages, computation speed increased, and hardware became more dependable and functional. [1] Even into today we see computers getting smaller, faster, more dependable, cheaper, and more user-friendly.

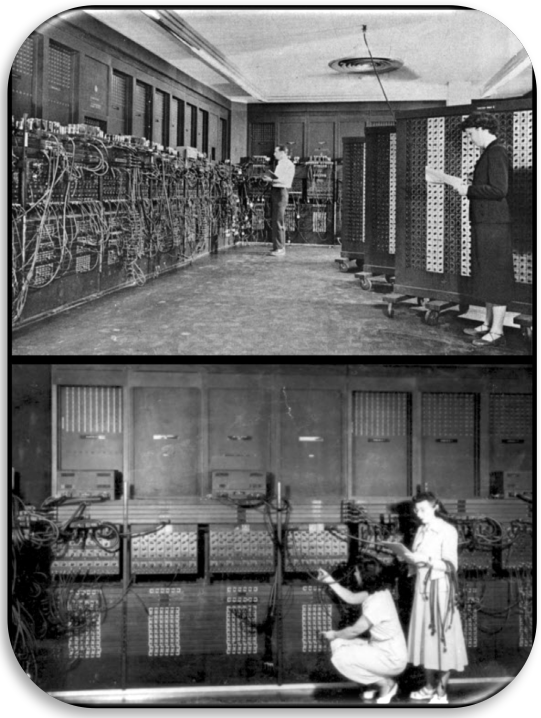

Figure 1: *The ENIAC (1946) was the world's first general-purpose electronic, digital computer. The ENIAC was used for calculations for the creation of the hydrogen bomb. [4]*

The historical development of computers is a fascinating one, but this account so far lacks a major component of the story: the security issues which arose as a result of the new technology. Security was actually an issue from the beginning. At first it was simply the physical computer which needed to be secure and safely kept, specifically from physical damage. However, it wasn't long before reports of disgruntled employees shed light on the havoc that could be caused by tampering with the "software". Churchhouse's *Codes and Ciphers* recalls an early story of a programmer who had "written a payroll program and who noticed that his employers had a habit of sacking their employees at very short notice. He therefore inserted a section of code into the program to check that his name was still on the payroll and, if it wasn't, to delete the entire program." The programmer was terminated after management became aware of the program, and the program was deleted. Interestingly enough, they had to bring that programmer back on, giving him a pay increase, to replace the program. [1] This example, though fairly harmless and solvable, delineates the vulnerability of computers and points to a need for computer security and protection.

The Dawn of a Need for Security

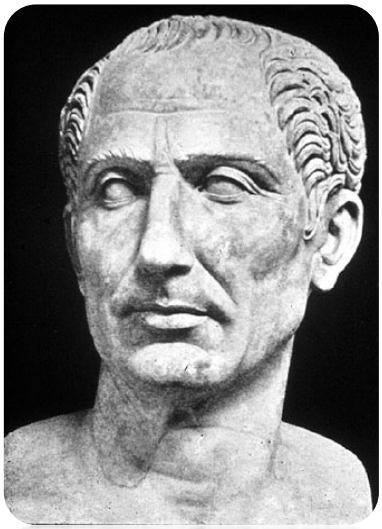

Figure 2: *Julius Caesar*

Security was actually a huge issue long before computers came into the picture. Dating back more than 2000 years ago, there are records of people wanting to send messages which could only be read by the people for whom they were intended. One peculiar solution was found among the ancient Greeks where "they took a slave and shaved his head and scratched the message on it. When his hair had grown they sent him off to deliver the message. The recipient shaved the slave's head and read the message." [1] This might have worked for people with an abundance of time and slaves, but serious issues can arise in this scenario. For instance, if someone who wanted the message was even remotely aware of this method of message passing, it would be as easy as capturing the slave in route of his delivery to compromise the secrecy of the message. The real question then is, "how can information be passed secretly and safely?" This is, in its very essence, the founding idea for *cryptography* which is "the art and science of secret writing" or, more exactly, "the process of providing secure communications over insecure channels." [2].

Julius Caesar (100 BC – 44 BC), a Roman general and ruler in his time, proposed a method that lives on to this day as a primitive yet insightful and educational example of the sorts of creativity that can arise in the field of cryptography. Caesar's cipher, as it is known, is simple: write down the message and move every letter three places forward in the alphabet. Caesar's cipher, while clever, is nothing more than a special case of what is known as a *substitution cipher* which is the shift of any number of characters in the forward (positive) or backward (negative) direction, *modulo* the length of the alphabet in question (for details on modular arithmetic, see

Figure 3: *The Caesar Cipher. The two levels could spin independent of one another.*

Appendix B). Essentially all this means that since there are only 26 letters in our alphabet, if you shift *more* than 26 characters in some direction, then you end up back at the beginning again. Consider the following example of applying *modular arithmetic*: say you want to start at the letter 'a' (consider it to be the letter in the alphabet at position 0) and shift the cipher 28 letters in the positive direction (e.g. a, b, c, d, and so on). After your 26^{th} shift, you've arrived back at the letter 'a'! Then you perform the remaining shifts and we end up at the letter 'c'. So, really you have only shifted the cipher two characters in the positive direction. Without actually performing all 28 shifts, we could have determined which character we would shift to by using modular arithmetic. Have you figured out what idea in math *modular arithmetic* is similar to? Maybe seeing how this would be expressed mathematically will give you a hint:

$$28\ (mod\ 26) = 2.$$

Have you figured it out? Well, if not, I won't hold the secret back from you. Modular arithmetic should be reminiscent of division! Looking back to the equation above, ask yourself, "What is 28 divided by 26?" Surely it is one with a remainder of two! That is precisely what happens with modular arithmetic; you essentially "throw away" the divisor and simply look at the remainder. In our example, this will tell us how far forward (in the "positive" direction of the alphabet) we should shift. Now modular math should be quite clear:

$$0\ (mod\ 26) = 0$$

$$1\ (mod\ 26) = 1$$

$$\vdots$$

$$25\ (mod\ 26) = 25$$

$$26\ (mod\ 26) = 0$$

$$27\ (mod\ 26) = 1$$

and so on and so forth.

Now, let's look at a more involved example: suppose we apply Caesar's cipher to the message "I love math", (which I would have no reason of hiding, but for the sake of an example I will proceed with the illustration.) the recipient would receive a message that says "l oryh pdwk" – note that this string of characters comes as a result of "shifting" the alphabet forward by three letters (e.g. replacing 'i' with 'l', 'l' with 'o', 'o' with 'r', and so forth). While this method does not have a great deal of sophistication, it does require that one knows the *key* in order to decrypt the ciphered text. A *key* in the formal sense is "any set of instructions that establish the method by which messages are to be encrypted and decrypted within a particular cryptosystem." [2]. However, we can think of key more informally as the information that we need to "unlock" the secret message. The key in the case of this last example is to shift every character *backwards* by three characters so that we can determine the original text. The use of the term "set of instruction" is appropriate as we begin to think about more sophisticated means of encrypting information. To expand on this, consider the substitution cipher in general where we could have shifted characters any number of times in any direction. The key for this cipher method then consists of: (1) *the number of letters shifted in the alphabet*, (2) *the direction of the shift in the alphabet, and* (3) *the length of the entire alphabet in use for modular computation* (we cannot necessarily assume that our 26-letter alphabet is used). Since this method was first introduced at least 2000 years ago, it has been well studied and is quite easily solvable today. In fact, with the advances in technology, it is not unreasonable to solve a substitution cipher of this level of sophistication via purely brute force computing and about 200 characters of encrypted text (also known as cipher text). [1, pg. 110]

The Fall of the Enigma

We know, however, that we live in world today where information is at least relatively secure, so we now fast forward in history to where we see significant progress in security methods. The encryption of secret messages grew in sophistication as people sought more secure and efficient methods of protecting their information. In fact, it wasn't long before people were combining the use of multiple alphabets and shorter sequences of cipher text in each respective alphabet to give any potential code

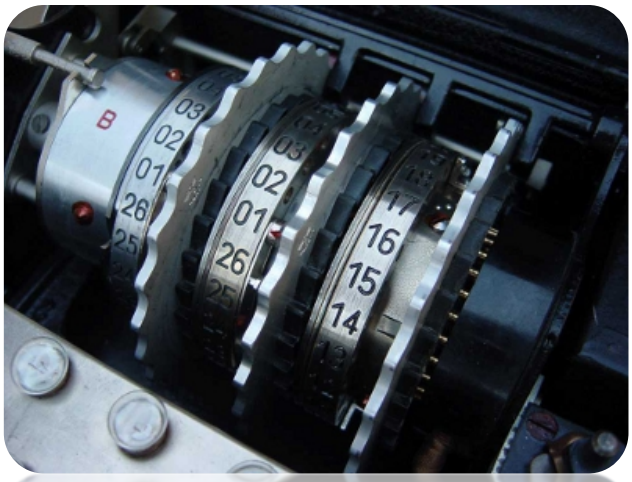

Figure 4: *The heart of an original Enigma machine. Keys pressed would cause one or more rotors to step, constantly changing the substitution alphabet before each character was processed.*

breakers an overwhelming amount of jumbled characters from varying alphabets to sift through. Since they didn't have access to computers at that time, this made these messages very hard or nearly impossible to decipher. However, there were disadvantages to this new level of complexity; as the complexity of the system grows, so does the chance of making errors when attempting to decrypt the message – this includes the person(s) for whom the message was intended.

A well-known example of such a system takes our historical exploration to the battle grounds of World War I (1914-1918) & World War II (1939-1945). The primary means of communication between military allies in this time was via radio transmission; however, the messages could be easily intercepted by enemy forces by simply locating the correct frequency of transmission. It was at the close of World War I that a huge break-through in the encryption landscape was credited to Arthur Scherbius, a German engineer, who designed an encipherment/decipherment machine which came to be known as the *Enigma*. The Enigma, in concept, selected a different alphabet automatically after each individual letter was enciphered. Thus, none of the substitution alphabets would repeat until thousands of characters had been processed. [1, pg. 111] To accomplish this, the Enigma consisted of multiple movable wheels which rotated in different ways after each character was processed. Although this encryption method seemed promising, in 1932 messages that had been encrypted by the Enigma were cracked! As mentioned earlier, as the complexity of an encryption method increases, so does the complication of decrypting any messages that are encrypted in this way. It just so happens that the defeat of the Enigma rested in this flaw. In order for the intended recipient to be able to decrypt messages, information about the *key* had to be sent as part of a message or in plain text. Neither of these are ideal options since the latter is extremely dangerous and the former implies that the key for *that* message must be sent in one of the two ways (thus

creating a cycle that must break down at some point). So, the Germans determined a third option: they devised a scheme in which they would systematically encrypt messages where the day in the year dictated the initial settings of the Enigma. [1]

Initially, the challenge to anyone attempting to intercept messages and decrypt them was that the Enigma enabled so many seemingly random choices for encrypting messages. However, once a scheme is devised, such as the one the Germans thought up, the element of randomness goes out the window and patterns begin to emerge. The details of how the Enigma was "cracked" are rather elaborate and are far beyond the scope of this paper but, to give the reader some idea of the discovery, it was by using facts about *symmetric plain-cipher pairs*, the identification of non-random wheel settings (*ground settings*), and properties of *cipher doublets*, along with access to numerous encrypted messages, that it was possible to determine the *ground settings* of the wheels at the beginning of message encryption, allowing cryptanalysts to reverse the cipher and decrypt a message. Germans attempted to modify the Enigma to make it overcome the discovery of the flaw by Allied forces, but because of the breakthrough "German coded messages never again proved an insurmountable task for the Allies." [2] This is fitting with the theme of this paper in that a clever method used for securing information motivated a lot of attention on the Enigma. The result was that a method to defeat the Enigma was developed and would, from that point on, make that security method or any other similar methods altogether obsolete.

The fact that the integrity of the Enigma was compromised because of the decryption method and the transmission of the necessary information for operators to encrypt/decrypt messages provides a suitable transition towards the real aim of this paper. We have now arrived at historically relevant motivation which calls to question the manner in which we encrypt information as well as the method for protecting the transmission of the *key* to that encrypted information. This lends itself well to investigating public key encryption, which is the method of utilizing public keys (keys that are available to the public) and private keys (keys which are kept secret from non-intended recipients) in order to allow the encryption and decryption of information. The basic idea is that anyone using a public key (of which there

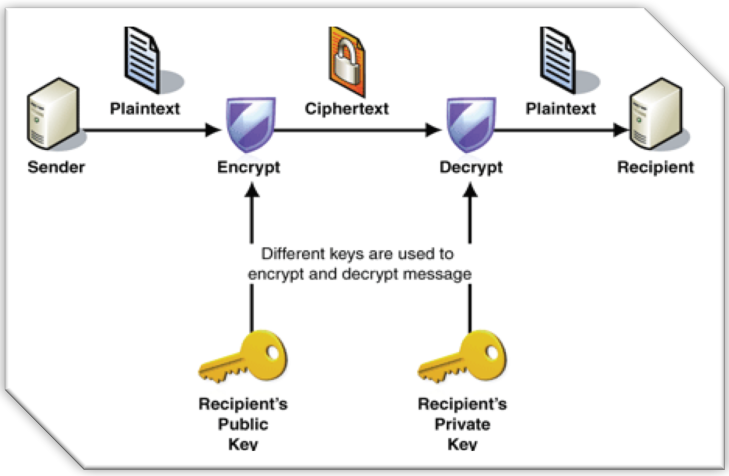

Figure 5: *An illustration of the encryption/decryption process which is dependent on the use of an intended recipient's public and private keys.*

can be many), can encrypt messages and send them to the person who holds the corresponding private key. Only the holder of a private key can decrypt messages that come from the agreeing public key. Therefore, as long as we keep our private key in a safe place, no one else can "unlock" the lock to our secret messages. If this isn't entirely clear at this point, don't worry. The majority of the paper that follows will elaborate on these ideas, but first we establish a few tools which will allow us to discuss the topic further.

Mathematical Problems: Impossible or "Hard"

Before we are able to discuss public key encryption and its use as a popular method of securing and transmitting information, we must first explore some of the underlying problems and mathematics that have made public key encryption altogether possible. We begin this discussion with a man named Alan Turing (1912-1954), born in London, England, who was a British Mathematician and cryptanalyst. Growing up there was nothing particularly special about Turing. In his early childhood, Turing was a bit of a problem child because he didn't care much for teachers that put more emphasis on non-science related teachings. Though recognized as a bright child, Turing is probably most known for his fondness of calculus and for his enjoyment of competitive running. [2] Some years later, Turing was rejected from Trinity College in Cambridge, and turned to King's College (also in Cambridge). It was from King's College where Turing graduated with a master's degree in mathematics "and produced his first, and perhaps greatest, work," his paper '*On Computable Numbers with an Application to the Entscheidungs Problem*'. [3] In this paper (1936), Turing responds to a problem which questions the completeness of logic: whether all mathematical problems could, in principle, be solved. Turing's results concluded that

Figure 6: *Alan Turing*

"there can be no general process for determining whether a given formula **Ц** of the functional calculus **K** is provable, i.e. that there can be no machine which, supplied with any one **Ц** of these formulae, will eventually say whether **Ц** is provable." [10] A footnote in the paper spoke of a theoretical computing machine that would be able to solve any mathematical problem "provided it was given the proper algorithms, or problem-solving equations or instructions." [3] The Turing Machine, as it is known, is quite an interesting idea, but its relevance to this paper is minimal so we will leave it to the interested reader to independently explore any details. A significant take away from this could be that Turing's analysis allowed him to make the above claim and gives rise to Turing being credited with laying the foundation for the creation of the modern computer.

Another significant result, and the idea that we will focus on, is that Turing's paper led to the idea of categorizing problems according to the difficulty of solving some problem. This area of study, known as 'Complexity Theory,' is an important branch of theoretical mathematics and computer science, where one "tries to classify problems that can or cannot be solved with appropriately restricted resources." [11] The notion of *restricted resources* most often refers to finite resources such as memory or computer processors, and what we will see soon is that the real question is "regardless of resources, can problem *x* be solved?" Since this is a large area of study, we will specifically consider the idea of investigating problems that are or are not able to be solved by some set of mathematical set of instructions. The lead up to this point is quite central because we are arriving at this idea that *not all problems are solvable*. If this is in fact true, then it can mean one of two things for us as it specifically relates to encryption:

1) If we are to find some security method for encrypting and transmitting information which depends on a sophisticated mathematical idea, then there exists the possibility that we will be

unable to use this method if we ourselves are unable to guarantee that the foundational mathematics are sound. This means that our methods are confined to be more *trivial*. Or,

2) If we can in some way use mathematical ideas that are *hard* to solve, but nevertheless *solvable*, then we have not guaranteed that our methods are entirely secure.

The preceding argument is a good segue to now discuss a few of the problems that one may encounter in Complexity Theory, and, more specifically, in encryption methods used today.

Factoring Large Primes

One of the chief ideas that today's encryption methods rest in is that of *large* prime numbers and interesting, but non-trivial, properties associated to *number theory* (an area of mathematics). Think about this: the problems that we encountered earlier with historical methods of security had significant issues with how to keep the *key* to our encrypted messages safe (i.e. recall the story of the captured slave and the non-random *ground settings* which led to the fall of the Enigma). What we really want then is a way to make our key impossible (or at least really difficult) to determine, right?

The *fundamental theorem of arithmetic*, also known as the unique factorization theorem, states "(existence) that every integer greater than one is either a prime itself or is the product of prime numbers, and (uniqueness) that, although the order of primes in the second case is arbitrary, the primes themselves are not." [12] For example, we know that 2, 3, 5, 7, 11, 13, 17, 19, 23, and so on, are all prime numbers. Consider the number 1200. Clearly not prime, right? *1200 is clearly divisible by 2 since it is an even number*. That means it must be the product of prime numbers if the fundamental theorem of arithmetic is true; and, as a matter of fact, 1200 is a product of prime numbers:

$$1200 = 2 * 2 * 2 * 2 * 3 * 5 * 5 = 2^4 * 3^1 * 5^2.$$

Why is this important or applicable to methods of encryption? It just so happens that certain algorithms, as we will see soon, are dependent on the idea of computing the product of *very large* prime

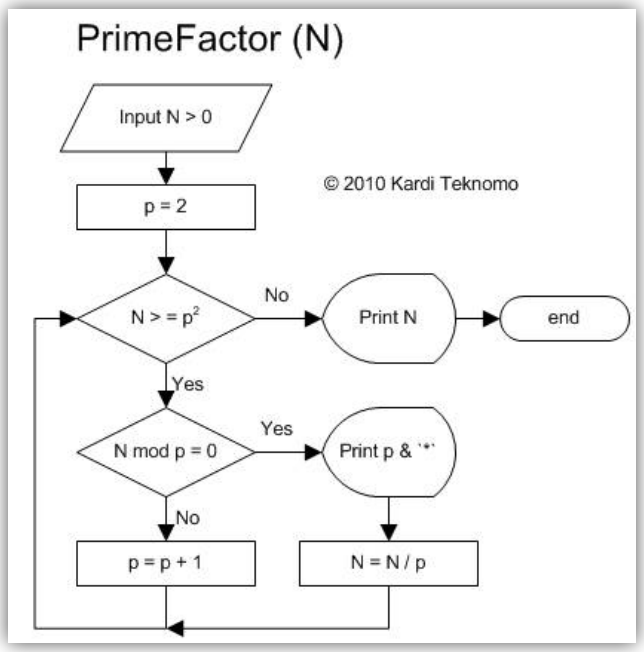

Figure 7: *One example of a factorization algorithm. The number N is the composite of some number of primes.*

numbers, and using the prime number factors as the *public* and *private* keys. As it stands today in the landscape of mathematics, and more specifically number theory, it is no trivial task to determine the prime factors of large numbers. While there are in fact algorithms (we will not go into detail here, but an example of one is shown in the figure 7), which allow for some number *N* to be factored into its primes, for large numbers, "no efficient, non-quantum integer factorization algorithm is known; an effort concluded in 2009 by several researchers factored a 232-digit number (RSA-768), utilizing *hundreds* of machines over a span of *2 years*." [14] (Figure 8 below is a graph of the time needed to decrypt a message when the key is *known*). To give the reader a better idea, when I say large, today's encryption numbers use numbers in the range of 1024-2048 bits. For example, RSA-2048 is an "RSA-number" that has 2,048 bits, which is 617 decimal digits (RSA will be discussed in further detail a bit later). Also, to further delineate for the reader how difficult this problem is, it might be useful to know that currently the largest factored RSA-number is RSA-768 (768 bits) which is only 232 decimal digits and the US is offering $200,000 to anyone able to factor an RSA-number of 2048 bits (RSA-2048) into its prime factors. [8]

Since it is difficult to factor large numbers into their primes, it seems as if using this mathematical idea will ensure a promising level of security from attackers that are trying to decrypt any intercepted messages. Without the key, anyone who wrongfully obtains a message has their work cut out for them if they hope to determine its contents.

At least this is what most people think today. However, in February of 2012, a group of European and American mathematicians and cryptographers discovered an unexpected weakness in encryption systems used by online banks, e-mail, and other internet services. [13] As with the Enigma, the flaw rested in the non-random determination of a key. In similar form today, "The flaw — which involves a small but measurable number of cases — has to do with the way the system generates random numbers." [13] The whole premise of the use of prime numbers as keys, as mentioned earlier, is that the original numbers *must* be kept secret and they must be *randomly* generated primes (so as to rule any deterministic patterns in directories containing large amounts of keys). The group of researchers found that the system fails to generate these numbers randomly in some cases. The following findings are alarming:

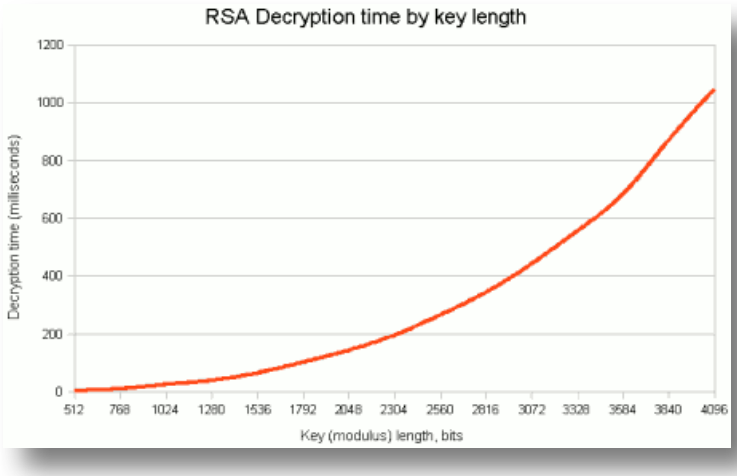

Figure 8: *Graph of decryption time based on a known key of some bit length. It is important here to notice the trade-off: greater security comes at a costly price.*

> *"The researchers examined public databases of 7.1 million public keys used to secure e-mail messages, online banking transactions and other secure data exchanges. The researchers employed the Euclidean algorithm, an efficient way to find the greatest common divisor of two integers, to examine those public key numbers. They were able to produce evidence that a small percentage of those numbers were not truly random, making it possible to determine the underlying numbers, or secret keys, used to generate the public key." [13]*

The researchers ended up finding almost 27,000 different keys that offer *no* security! While this isn't a direct issue with the prime number factorization, this still suggests that the *method* which utilizes prime number properties is not without blemish. This is proof of how our continued study of technology and mathematics results in the exploitation of our methods. Also, one concern that the researchers had was that this issue isn't altogether difficult to notice; people just haven't been looking in the right place or in the right way. Consequently, their fear is that others have already discovered this vulnerability and have chosen not to make it known to others so that they are/were able to take advantage of this weakness in encryption security. The only positive outcome from this, which surely is not a real solution but rather a band aid, is that the researchers rushed to make their findings public so that banks, online stores, and other internet services could make any necessary changes to prevent attackers from taking advantage of *this* flaw.

The RSA Problem

Another principle idea behind our security methods today is that of the *RSA Problem*. The RSA problem is to *efficiently* compute P (a secret prime factor) given only an RSA public key (N, e) – where N is the publicly available product of the secret primes, and e is the public exponent) – and a ciphertext $\mathrm{C} \equiv P^e \pmod{\mathrm{N}}$. The structure of the RSA public key as specified by the RSA algorithm requires that N be a large composite of two large primes, that $2 < e < N$, that e be *coprime* $\varphi(N)$, and that $0 \leq C < N$ where C is chosen *randomly* within that range. [8] Specifics of this will be discussed in greater detail in the next section, but for the time being this is presented to articulate the nature of the problem. Sifting through the mathematical notation and technical definitions, the problem that emerges is to determine precisely how knowing N and e can allow us to recover P, "which will depend on the precise means of RSA random keypair generation in use." [8] This last statement alone sums up the issues presented in the previous section where computers failed to generate *random* keypairs for all keys.

Since no deterministic algorithm has yet been developed to guarantee a solution to this problem, we have again found just motivation for using these mathematical ideas as a means of creating a viable

security method. In the section that follows, methods of encryption and message transmission will be studied in much greater detail. The initial analysis of the methods should bring about a sense of confidence in the derived methods that are used today, but the subsequent review of flaws and weaknesses exist to support the underlying message of this paper: mathematical sophistication and technological advances equip attackers with the information and tools necessary to prevent the engineering of a truly dependable and reliable method of security.

Public Key Encryption

Now that we have established a sound foundation in relevant history and necessary mathematical tools, we can dive deeper into our direct investigation of the idea of public key encryption and some of the common systems which are in play today. Please note that this is by no means an all-inclusive discussion, but rather a few case studies which will allow us to get a sense of the landscape in the realm of security and encryption via the use of public keys. As we dive into examples in the coming text, I would like to introduce my friends Bob and Alice. Bob and Alice are two hypothetical persons that are commonly used in examples and illustrations to help delineate the nuances of concepts in cryptography. Bob and Alice will represent two persons trying to establish a secure channel of communication so that they may transmit information (messages) without fear of some unwanted party compromising their privacy. In order to establish such a channel of communication, we look to the encryption system which involves the use of one or more keys, which only Bob and Alice know, to establish and share a common secret key, allowing the two to securely encrypt and decrypt messages back and forth. Now, the underlying issue, "How are they to let each other know their secret key(s) without other people discovering them too?" [1, pg. 166] While a number of issues can be discussed in light of this question, the scope of this paper narrows in on two such problems: *the authentication problem* and *the signature verification problem*. Details of these problems will be highlighted in the examples that follow.

The Diffie-Hellman Key Exchange

One method that is currently in use for securely sending messages over an insecure channel is credited to the work of Whitfield Diffie and Martin Hellman. Though rumor suggests that the creation of the algorithm took place in years prior, in 1976 Diffie and Hellman proposed a system that suggested that a person should be able to send an encrypted message, but not necessarily be able to decrypt it themselves. This method, which came to be known as the Diffie-Hellman key exchange, allows an individual to select two separate algorithms: one is a known as the public encryption algorithm, and the other as the private decryption algorithm, which is retained by the individual. Now, the public key is, as the name suggests, made public. The user can give the encrypting algorithm to persons wishing to send private messages, with confidence that the user has the sole decryption algorithm (private key). There is however one element where this system can break down, so one must begin this process with special caution. The two algorithms must be selected in such a way that the encryption algorithm is unable (or unlikely) to be able to determine the decryption algorithm. If the ladder scenario fails, then secrecy will most definitely be compromised.

Method & Mathematics

The sole purpose of this high level discussion points to a method of establishing common ground where messages can be encrypted and decrypted by a common key. Let's look a bit deeper at this method by bringing our good friends Bob and Alice in to help us out. If we take a look 'under the hood', the Diffie-Hellman key exchange is as follows:

(1) Bob and Alice both agree on the use of two integers, call them p and q, where p is a large prime and $1 < q < (p-1)$. We do not care if the values of p and q are known to others, but the picking of these numbers must satisfy the noted restrictions.

(2) Now, suppose Bob chooses a *secret* number, x, and Alice chooses a *secret* number, y. Note, the chosen numbers should satisfy $1 < x, y < (p-1)$, and neither should share any common factors

with $(p-1)$. We do care here that the values x and y are kept secret from one another and from anyone else (these are the only numbers that need be *private*).

(3) The reader is expected to have some knowledge of basic number theory for this step (see Appendix B for further details). Before we can establish the common key, Bob must first compute

$$B_x = m^x (mod p)$$

and send the result to Alice. Likewise, Alice must compute

$$A_y = m^y (mod p)$$

and send the result to Bob.

(4) Finally, we can compute the *common key*, which we will call K. As in (3), recall the discussion of number theory in the previous section. What we see in the following expression is that both Bob and Alice have kept their own secret numbers private, but they have given each other a modified value which hides their information and allows them to make a similar computation and arrive at the same value, K:

$$(B_x)^y = \left(A_y\right)^x \equiv m^{xy}\ (mod p) = K.$$

Clearly stated, Alice and Bob compute the first two terms in the above equality, respectively. [1, pg. 167].

Discussion & Review

In reviewing the method, did you catch any possible problems? One fairly obvious issue is the possibility of a third party intercepting computed values that Bob and Alice send each other, B_x and A_y. The third party could replace one of their computed values with his/her own, and then pass the message along to one of them without them knowing what had happened. This third party could then pretend as if they were Bob or Alice, establish the common key, and then be granted the ability to freely encrypt and decrypt any messages. This brings up one of the *major* issues that was briefly mentioned at the start of this section, *the authentication problem*. Bob and/or Alice have no way of knowing whether or not the values they are receiving are actually coming from one another. As we have discovered here, the Diffie-

Hellman key exchange fails to authenticate the sender of the message to the receiver of the message, thus opening to door for secrecy to be compromised.

Another potential issue is the solvability of $B_x = m^x \ (mod\ p)$ for Bob's secret value of x. It isn't altogether unlikely that some third party could obtain all of the components of this expression (excluding x – if the third party had x there would be no need to solve this equation). However, this particular problem is considered to be a 'very difficult problem' given that p is a very large prime, on the order of 10^{200} and is considered to be solvable only by an attacker that is well-equipped with strong computing power, solid understanding of mathematics, and a high level of determination. This is the first example where our investigation of an encryption method has obvious flaws and where it is well known that (more) computing power is a sure bet element in the potential failure of such a scheme.

The RSA Algorithm

The RSA algorithm is another encryption algorithm which will invoke some similar ideas as what we witnessed previously with the Diffie-Hellman key exchange algorithm. The RSA algorithm is titled such after three MIT students, Ron Rivest, Adi Shamir, and Leonard Adleman, who developed the algorithm in 1977. Before moving into the mathematics and further discussion of the RSA algorithm, I would like to begin with clarifying that the method that follows is not so much concerned with the actual process of encrypting or decrypting information, but rather, is used to transport symmetric keys, in order to establish grounds for secure interaction between any and all desired parties. In fact, the RSA algorithm is often coupled with SHA-1 or and MD5 algorithm which actually handles the encryption/decryption of data after the secure connection has been established. These particular encryption methods are beyond the scope of this paper.

Method & Mathematics

The RSA algorithm consists of three primary steps: 1) key generation, 2) encryption, and 3) decryption. The steps, though not altogether mathematically complex, stand on the foundation that the

security of RSA is based on the difficulty of solving the problem of factoring large integers. As with the previous method, we break down the steps of approaching encryption via the RSA algorithm:

Key Generation

(1) Choose two distinct prime numbers, p and q, of similar bit length. Define the modulus for the public & private keys as the product of the selected numbers (e.g. $n = pq$).

(2) Using Euler's totient function, compute $\varphi(n) = (p-1)(q-1)$.

(3) Now, choose an integer, call it x, such that $1 < x < \varphi(n)$ and x and $\varphi(n)$ are coprime.

- Note: x is public (no need to keep this value secret).

(4) Determine d where $d \equiv x^{-1} \ (mod\ \varphi(n))$ or, rather, $de \ \big(mod\ \varphi(n)\big) = 1$.

- Note: d is now our private key exponent (do not share!)

To illustrate the next two steps in the RSA algorithm, we call our friends Bob and Alice back into the picture:

Encryption

Suppose Bob wants to send Alice a message. Assuming that Bob and Alice have each already generated both their public and private keys, Bob can convert his message into some integer, call it m for *the message*. Bob then computes the following and sends the result to Alice:

$$B_x = m^x \ (mod\ n)$$

Does any of this look familiar? Bob sent Alice something very similar to what we saw transmitted in the Diffie-Hellman key exchange.

Decryption

Alice has received the encrypted message from Bob, but now what? Well, as we might have suspected, Alice can recover m simply by computing

$$m = (B_x)^y \ (mod\ n)$$

where, y is Alice's private key (chosen similar to how Bob chose his private key). [8]

Discussion & Proof of Correctness (Fermat's Little Theorem)

After presenting the previous method, the reader was then asked if they noticed any problems with the method, so, to follow suit, are there any foreseeable issues with the RSA algorithm? The answer is "of course!" but the issues may not be as obvious in this case since this method, in many ways, is superior to the previous method we covered. One rather large issue, however, is due to the deterministic nature of the RSA algorithm. The implication of this is that there is no random component in the algorithm (everything is chosen based on strict criteria and well defined functions), so an attacker can perform a plain text attack where they encrypt their own data via the available public key and try to discover information with the returned cipher text. Although some encryption systems safeguard against attacks such as these, it is not unfeasible for attackers to use this method of attack and be successful.

On a more positive note, one of the benefits of this algorithm is the introduction of the ability to support what is known as *digital signatures,* which is accomplished via the development of collision resistant hash functions for signing a hash of the message. [7] This connects us to the other idea which was briefly presented at the beginning of this section: *the signature verification problem*. The issue stated above arises, in part, from the fact that we are unsure of who we are receiving messages from (since anyone has access to the public key(s), anyone is able to encrypt messages and send them to the owner of that public key). So how do we know that a particular person sent a specific message? Well, the RSA algorithm handles this by allowing a person, say, Alice, to use her personal private key to produce a hash value of the message, raising it to the power of her private key and taking the mod n (exactly the same as decrypting messages), and attaches it as a signatures to the message to be sent. [8] To verify the signature, Bob, as the intended recipient, can use the same hash algorithm with Alice's public key. After raising the signature to the power of his private key number and taking the mod n, Bob can compare the actual message hash with the resulting hash and, if they match, he can be confident that the sender was in possession of Alice's private key (hopefully meaning that Alice had possession of her private key).

Before we move on from this discussion, one of the underlying points of this paper is to raise awareness of the issues that follow rigorous mathematics. As a result of clearly defining deterministic algorithms, such as the RSA algorithm, potential attackers have access to the thought process of the enemy. They have the advantage in that they have all of the information presented in proofs and other well defined mathematics, but they get to keep their secrets and exploit the flaws which are freely presented as creators or revolutionists publish their every thought about a given idea. The following proof of correctness of the RSA algorithm using Fermat's Little Theorem is one such example:

Fermat's Little Theorem: If p is prime and p does not divide an integer a, then

$$a^{(p-1)} \equiv 1 \ (mod\ p).$$

Proof:

First, we want to show $(m^e)^d \equiv m \ (mod\ pq), \ \ \forall\ m \in \mathbb{Z},$ when p and q are distinct prime numbers and e and d are positive integers satisfying

$$ed \equiv 1\big(mod\ (p-1)(q-1)\big) \quad \Rightarrow \quad ed - 1 = h(p-1)(q-1)$$

for some nonnegative integer h. In order to verify congruency $mod\ pq$, it suffices to check congruency $mod\ p$ and $mod\ q$ separately (we do this as a result of what is done in the Chinese Remainder Theorem). Now, to verify that

$$m^{ed} \equiv m \ (mod\ p)$$

consider two cases:

$$m^{ed} \equiv 0 \ \equiv m \ (mod\ p)$$

and

$$m^{ed} = m^{(ed-1)} = m^{h(p-1)(q-1)}m = (m^{(p-1)})^{h(q-1)}m \equiv 1^{h(q-1)}m \ \equiv m \ (mod\ p)$$

where we used Fermat's Little Theorem to replace $m^{(p-1)}\ mod\ p$ with 1.

We can proceed in a similar manner to verify the cases for $mod\ q$.

Thus, $(m^e)^d \equiv m\ (mod\ pq), \quad \forall\ m\ \epsilon\ \mathbb{Z}$ □

Conclusion

Encryption Today

Research of encryption in today's world yields some surprising results. SafeNet, a data protection company, published an article that reflects on an industry survey about encryption practices within organizations. SafeNet sheds light on the increase of encryption in organizations today, and makes recommendations based on the survey intended to help companies become more aware of their potential vulnerabilities. In the article, SafeNet presents findings and responses to questions like "How are you currently using encryption today?", "How actively do you rotate your encryption keys?", and "How do you manage cryptographic infrastructure?" [6] Responses to the first question suggest that companies are focusing a great deal on encrypting data for usage on laptops, smart phones, tablets, web applications, and identity-based authentication. This seems to set a great tone, showing that companies are flexible and willing to move with the changes in the technology world. It also affirms that internet based communication and transactions are the areas which we need to truly focus on securing since our society is moving more towards a web-based culture, both in work and recreation.

I am, however, concerned about one of the findings that was highlighted in the article by SafeNet. In response to the second question posed above, only 44% of those surveyed were confident that they knew where their cryptographic keys were stored and nearly 20% had no knowledge whatsoever of the location or use of cryptographic keys in their company. The implication of these findings led those writing from SafeNet to assert, "Too often, the efforts and tactics required to secure cryptographic keys aren't fully understood when security teams first embark on an encryption initiative. As the above results

make clear, lack of visibility, control, and security of cryptographic keys represents a significant vulnerability in many organizations today." [6]

Support claim that technology continues to grow and sustain the "cycle"

After our investigation of some of the more common encryption methods in use today, as well as a brief look into some of the feelings regarding encryption in practice today, it is my ultimate hope that certain connections have been drawn. First and foremost, we cannot deny that as mathematics and computing grow in rigor, power, and complexity, likewise the resources available to potential attackers increase. Clever and increasingly complex mathematics shed light on problems that were once considered 'hard' to solve, and they also make the actions and methods of potential attackers less noticeable and more lethal. Also, it should be noted that the study in this paper exposes the fact that new methods and dependable solutions for information encryption are not progressing as quickly as they once were. Realistically, today our popular security methods consist of modifications to methods developed in the early history of computing up through the 1980s and 1990s. The raw power of computers, however, is increasing exponentially according to Moore's Law, which has accurately described the trend of computer development since the 1960s. [5] In closing, while I do not discourage the study and progression of our understanding of mathematics and computing technology, I do wish that we would inform people more frequently of the implications of our progress. I see the development and growth of our society as an unknowing and altogether unaware antagonist to the successes of the corrupt and vicious; our best weapon moving forward is awareness of the era that we have entered.

Appendix A: Terms & Definitions

- **Cipher System**: A *cipher system* is any method or system that can be used to change the text of a message. The main goal is to make the text unreadable to anyone whom the message is not intended for.
- **Cipher Text**: The encrypted version of the original message.
- **Cryptology**: The science of secure communications.
- **Cryptography**: The study of the design and use of cipher systems including their strengths, weaknesses, and vulnerability to various methods of attack. [1]
- **Encrypt (Encryption)**: To *encrypt* a message or string of bits is the process of applying a cipher system to a message or string of bits.
- **Decrypt (Decryption)**: To *decrypt* a message or string of bits is to reverse the process of encryption, thus transforming the cipher text back to the original plain text.
- **Hash (Function)**: A *hash (function)* projects a value from a set with many (or even an infinite number of) members to a value from a set with a fixed number of (fewer) members. Hash functions are not reversible.
- **Key(s)**: A piece (or pieces) of information that enables a recipient to decrypt cipher text from a cryptographic algorithm or cipher back to its plaintext form.
- **Key Exchange**: Any means by which cryptographic keys are exchanged between parties so that a cryptographic algorithm can be used to secure any transfer of information.
- **Plain Text**: The original message (before encryption).

The following two definitions are related to one another:

- **Private Key**: The key which is kept private – used to decrypt any messages that are encrypted with its corresponding *public key*.
- **Public Key**: The key which is made private – accessible to the general public – used to encrypt messages which can *only* be decrypted by the holder of the *private key*.

- **RSA-number**: The *RSA-numbers* are the set of large, semiprime numbers (numbers with exactly two primes). RSA-*X* refers to any such semiprime number that is made up of *X* bits. [8]

Appendix B: Mathematics

1. Modular Mathematics:

Some examples of modular arithmetic already exist in the section of this paper titled *The Dawn of a Need for Security*. The following are properties and general rules which are well defined within mathematics and are relevant to the level of modular mathematics covered in this paper. For the following rules we define *a, b, c, k, m,* and *n* to be some integers, where $a = a' \pmod m$, $b = b' \pmod m$, and $c = c' \pmod m$:

a. **Equivalence:** $a \equiv b \bmod 0 \rightarrow a = b$

b. **Reflexivity:** $a \equiv a \bmod m$.

c. **Symmetry:** If $a \equiv b \bmod m$, then $b \equiv a \bmod m$.

d. **Transitivity:** If $a \equiv b \bmod m$ and $b \equiv c \bmod m$, then $a \equiv c \bmod m$.

e. $ab \equiv a'b' \pmod m$

f. $a \equiv b \bmod m \rightarrow ka = kb \pmod m$

g. $a \equiv b \bmod m \rightarrow a^n = b^n \pmod m$ [15]

2. Prime Number Theory:

a. Chinese Remainder (Factoring) Theorem: The theorem states that if $\gcd(m_1, m_2) = 1$ then there is a unique solution $0 \leq x < m_1 m_2$ to $x \equiv c_i (mod\ m_i)$ for $i = 1,2$. [7]

b. Euler's totient or phi function, $\varphi(n)$ is an arithmetic function that counts the number of positive integers less than or equal to n that are relatively prime to n. That is, if n is a positive integer, then $\varphi(n)$ is the number of integers k in the range $1 \leq k \leq n$, for which $gcd(n, k) = 1$. The totient function is a multiplicative function, meaning that if two numbers m and n are relatively prime (to each other), then $\varphi(mn) = \varphi(m)\varphi(n)$. [9]

Works Cited

[1] Churchhouse, R. F. *Codes and Ciphers: Julius Caesar, the Enigma, and the Internet*. Cambridge: Cambridge UP, 2002. Print.

[2] Newton, David E. Encyclopedia of Cryptology. Santa Barbara, CA: ABC-CLIO, 1997. Print.

[3] Young, Robyn V. Notable Mathematicians: From Ancient times to the Present. Detroit, MI: Gale, 1998. Print. (pg. 479-482)

[4] "Retro Delight: Gallery of Early Computers (1940s - 1960s)." Retro Delight: Gallery of Early Computers (1940s - 1960s). Pingdom, 11 Dec. 2009. Web. 20 July 2012. <http://royal.pingdom.com/2009/12/11/retro-delight-gallery-of-early-computers-1940s-1960s/>.

[5] "Moore's Law." Wikipedia. Wikimedia Foundation, 21 July 2012. Web. 24 July 2012. <http://en.wikipedia.org/wiki/Moore's_law>.

[6] "The Current State of Encryption and Key Management." SafeNet, 2012. Web. <www.safenet-inc.com>.

[7] Galbraith, Steven D. Mathematics of Public Key Cryptography. Cambridge: Cambridge UP, 2012. Print.

[8] "RSA (algorithm)." Wikipedia. Wikimedia Foundation, 24 July 2012. Web. 24 July 2012. <http://en.wikipedia.org/wiki/RSA_%28algorithm%29>.

[9] "Euler's totient function" Wikipedia. Wikimedia Foundation, 21 July 2012. Web. 24 July 2012. <http://en.wikipedia.org/wiki/Euler%27s_totient_function>.

[10] Turing, Alan M. "On Computable Numbers, with an Application to the Entscheidungs Problem." Thesis. King's College, 1936. Web. <http://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf>.

[11] "Computational Complexity Theory." Wikipedia. Wikimedia Foundation, 08 Jan. 2012. Web. 29 July 2012. <http://en.wikipedia.org/wiki/Computational_complexity_theory>.

[12] "Fundamental Theorem of Arithmetic." *Wikipedia*. Wikimedia Foundation, 30 July 2012. Web. 1 Aug. 2012. <http://en.wikipedia.org/wiki/Fundamental_theorem_of_arithmetic>.

[13] Markoff, John. "Researchers Find a Flaw in a Widely Used Online Encryption Method." *The New York Times*. The New York Times, 15 Feb. 2012. Web. 28 July 2012. <http://www.nytimes.com/2012/02/15/technology/researchers-find-flaw-in-an-online-encryption-

method.html?_r=2>.

[14] "Integer Factorization." *Wikipedia*. Wikimedia Foundation, 30 July 2012. Web. 01 Aug. 2012. <http://en.wikipedia.org/wiki/Integer_factorization>

[15] Weisstein, Eric W. "Congruence." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/Congruence.html