# Cryptography: Attempting to Avoid the Sins of Fathers

Ryan Guzal

GSEC v 1.4b – Option 1
10Jan2003

## Abstract

For many corporate enterprises worldwide, the past few years have seen an almost exponential rise in traffic arriving from, and destined for the Internet. Traffic in the form of web browsing, file transfer, and email screams through network routers at a steady pace, and all the while, most users never give a thought to the security of the information they are passing back and forth. Most users are ultimately concerned with whether or not the email they sent to their sibling arrived on time (if at all), not whether it was intact or unaltered. Likewise, the recipient sibling rarely gives a second thought to the true origin of said email, let alone the integrity of the data it contains. Did sender sibling actually craft the email? If so, is the data that recipient sibling is looking at the actual data that sender sibling sent? As stated, these are questions that rarely cross the mind of the common Internet user. But these are the questions that security administrators everywhere can and must answer on a daily basis.

In this paper, we'll take a look at some of the modern encryption techniques that are used to secure the transfer of data on the Internet. We'll take a look at the latest efforts in the battle to keep our data out of the hands of our enemies (or useless if it is intercepted by them). But before we get to today's standards in the cryptography circle, we'll take a look at a few of our historical harbingers. We'll pay a visit to some folks from the past, and we'll take a look at the encryption methods they used to keep their data secure while potentially passing through the hands of their enemies. The purpose of this paper is to discuss the path that cryptography has taken to this point in history. This paper will attempt to draw a correlation between the historical methods humans have used, and their ultimate failure. Finally, we'll look at the methods we're using today, attempting to use history as a guide to help us 'temper our expectations.' For if there is one thing that history has taught us, it is this: No matter how strong you think the encryption method that you are using is, it is only a matter of time before someone (or something) is smart enough or powerful enough to crack your 'unbreakable' code. It's got to be secure.

## Background

> Cryptography is both the lock and the combination (or key) that can be used to help protect your data. There are a variety of cryptographic methods and keys. Together, the method and the key determine cryptographic security.[1]

The history of cryptography dates back to the earliest recorded instances of man. We learned in grade school of the hieroglyphics that the ancient Egyptians used, and some of us may be aware that among other things, Thomas Jefferson is known as the father of American cryptography.[2] Therefore it's safe to say that if our ancient predecessors were masking the true meaning contained in their correspondence, centuries and even millennia ago, the need for such a science will remain going forward. For as long as cryptography has existed, so too has the need for it. As the saying goes, "Just because you're paranoid doesn't mean somebody isn't out to get you."

## Early Cryptography

The earliest instances of cryptography were rudimentary when compared to the advanced technology and mathematics that are employed in today's cryptographic methods. This is not to say though that ancient cryptographers were deficient in accomplishing their task. Since cryptography is all about successfully passing critical or sensitive information in the presence of adversaries, in order to be successful in encrypting our data we need only to keep our adversaries at bay until the information is no longer confidential. For instance, if an army commander needs to send a message to his troops informing them to attack the enemy on Friday morning at 0700 hours, he need only chose an encryption method that can withstand attack until sometime after 0700 on Friday morning (assuming the commander does not intend to use the same encryption methods on ensuing messages).

Throughout history, cryptographers have continually discovered new methods to disguise, or cipher their data. But as we look back at our crypto history, we notice a few techniques that have survived the test of time.

## Substitution Cipher

One of those early ciphers whose methods of encryption are rudimentary when compared to today's elaborate encryption techniques is known as the Caesar Cipher. Julius Caesar was aware of the critical advantage that information can create, and he sought to ensure that only his allies acquired and maintained such an advantage. To accomplish this, he used a simple substitution cipher.[3] In a substitution cipher, the art of disguising the true meaning of the message is accomplished by substituting letters of the alphabet for the actual letter the message contains. The alphabet is shifted a specific number of places whereby the letter A becomes some letter other than A. Each corresponding letter of the alphabet becomes the letter after that which A has been substituted for. Consider the following:

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | . | . | . | R | S | T | U | V | W | X | Y | Z |
| L | M | N | O | P | Q | R | S | . | . | . | C | D | E | F | G | H | I | J | K |

Text
Ciphertext

Using the above substitution cipher, we could then send the following enciphered message.

OCZGP OZHYEZHY TY ESP CLTY

Using the key -Shift +11- the recipient can then decode the message:

OCZGP OZHYEZHY TY ESP CLTY
DROVE DOWNTOWN IN THE RAIN

It's easy to see that this method of ciphering -also referred to as a rotational or shift cipher- would never be considered effective in today's generation. It is important to remember that at the time the Caesar Cipher was used, many people were unable to read. This simple fact alone greatly increased the security of Caesar's messages.

Today though, this method would be about as effective as spelling words to keep a young person from overhearing the sound those letters spell. While in the presence of little Johnny, mother might ask father, "Do you want some G-U-M?" She knows that Johnny might get overly excited & perhaps throw a fit if mother tells him that he cannot have any 'gum.' Therefore, mother uses this primitive form of code to keep little Johnny –adversary is a little harsh, don't you think- from understanding the meaning of her communication. Although this cipher is not of the substitution variety, it does give us a modern day example of primitive, yet effective, encryption.

The point here is that eventually little Johnny will learn how to spell, rendering this encryption method useless. Though by the time Johnny can read, and thereby decrypt mother's code, he might be mature and disciplined enough to be told no without reacting like a spoiled child. Just as Johnny will soon learn to read, so did most of society, and eventually the Caesar Cipher became ineffective as well.

If the history of cryptography has taught us anything is it that no encryption scheme lasts forever. Whether or not Caesar was aware the one-day his encryption scheme would become a simple game in the Sunday Times (provided he also envisioned the Sunday Times) is irrelevant. What's important to remember is that at one time this encryption scheme was effective and in use. This becomes one of the first instances of humans implementing an encryption method that was eventually rendered virtually useless in its primitive form.

## Transposition Cipher

In the scheme of things, most cryptographic methods need to secure their respective data for much longer than the short amounts of time we've seen in the above examples. However, it is important to note that no matter what encryption method we use today, it is only as secure as the methods (and thereby the technology) that our adversaries are using to break the code we use. Eventually it became apparent that a simple substitution cipher was inadequate. Luckily, cryptographers had been working on something with a little different twist.

What they discovered was a cryptographic method that was later dubbed a transposition cipher. In one flavor of this method, rather than replace the actual letter with another letter from the alphabet, the letters contained in the message are rearranged to form a grid. Using the grid, a transposition map is then built. From this map, the ciphertext and decryption key are derived.[4] Sound cryptic? Consider the following phrase.

NINE THIRTY ON A TUESDAY NITE[5]

We then insert the text into a 3 x 8 grid.

| | | |
|---|---|---|
| N | I | N |
| E | T | H |
| I | R | T |
| Y | O | N |
| A | T | U |
| E | S | D |
| A | Y | N |
| I | T | E |

Then, we recompose the message into ciphertext by arranging the letters from top to bottom, and left to right, starting with the leftmost column. The new, encrypted message looks like this:

NEIY AEAI ITRO TSYT NHTN UDNE

While this method is more effective than a simple substitution cipher, a major weakness of a basic transposition cipher is that the actual text of the message is contained in the ciphertext. Therefore, a transposition cipher is particularly vulnerable to known plaintext attack.

If our adversary has intercepted the above message and expects (or guesses) the message to contain instructions on a time and/or a date, we might be sunk. He or she need only to scan the letters of the message to determine which days of the week can be spelled (if any) and from there, attempt to determine the key that decrypts the entire message. If the cryptanalyst were to study the letters in search of the days to the week, he or she would see that there are zero M's, W's or F's. Thus, the cryptanalyst can eliminate Monday, Wednesday and Friday. Assuming this, he or she can get one step closer to discovering the data contained in the message. Suddenly, the cryptanalyst is well on the way to discovering the day that 'something' happened or will happen.

A truly adept cryptanalyst would take a slightly different approach. He or she might also guess that a day of the week (e.g. Sunday, Monday) is contained within the text of the message and thereby the ciphertext. This guessed keyword is often referred to as a *crib*.[6] Assuming this crib, a quick scan of the message shows that there are two Y's, two A's, but only one D. Since all of the days of the week happen to end in –day, the cryptanalyst need only to determine the numbers of positions used in this transposition scheme to extract the clues that will eventually unlock the whole thing. (Since we assume that our crib ends in –day, we'll refer to our guessed keyword, our crib, as crib-day)

Let's take a look at the message again and see how this little bit of knowledge can be used to determine the key to this ciphertext. Working forward from the letter D, we can quickly determine that there are two possible keys to this cipher.

NEIY **A**E**A**I ITRO TSYT NHTN U**D**NE
➔

The first possibility uses the first A of the ciphertext to make up the second to last letter in our crib-day. We know that if our crib exists in the ciphertext, it must use the letter D we have highlighted at the end of the ciphertext (since the ciphertext contains on one D). Counting forward from the D to the A, we notice that there are seven spaces. We then use this information (number) to determine if we're on the right path to cracking this cipher. From there, we work forward through the message to determine how many spaces are between the first letter A and the letter Y (the last letter in our crib-day). It's fairly obvious that the first letter Y in the ciphertext is a lot more than seven spaces away, so we can safely assume that if our crib-day exists, it will use the second letter Y in the ciphertext and not the first letter Y. Counting forward, we can see that there are ten spaces between the A and the Y. If our cryptanalyst had done their homework, he or she would quickly determine that the first letter A is not part of the crib-day contained in the ciphertext. In the next example, we'll see exactly why these two numbers can quickly be used to eliminate to first A from our crib-day.

NEIY **A**EAI ITRO TS**Y**T NHTN U**D**NE
7 ➔| ➔ 10 ➔| ➔

Since a transposition cipher uses a grid-ed scheme, all adjoining letters of the plaintext must share either the same number of spaces between them (n) or be separated by no more that (n+1) or (n-1), depending on which direction the cryptanalyst is moving through the cipher. At this point, our encryption is still afloat, but unless 9:30 on Tuesday has already passed, there's an increasing chance that we could get discovered.

At this point, the cryptanalyst is very near the solution to our transposition scheme. Armed with the knowledge that if the message does in fact contain the crib-day, it will use the second ciphertext A to do so, our cryptanalyst would then count the spaces between the letter D and the second A (9). Then, moving forward from the second A, the cryptanalyst would count nine spaces to arrive at the letter T. n-1 puts the cryptanalyst at the letter Y, and very near the key to the entire thing.

➔ ⬀9⬂ ➔

NEIY AE**A**I ITRO TS**Y**T NHTN U**D**NE

At this point, it's only a matter of time before the cryptanalyst counts backward eight spaces from the D to arrive at the letter S, eight more to get from S to the letter E, 9 more to get to the U, and eight more to get to the letter T to arrive at Y-A-D-S-E-U-T. Sure enough, the crib-day the cryptanalyst was after does in deed exist within the ciphertext (and thereby the original message).

As illustrated, there are inherent weaknesses in a simple transposition cipher. However, transposition ciphers were rather effective for a period of time. Instances of transposition ciphers exist throughout history, from the American Civil War to World War I and beyond. In most cases, the cryptographers that used these types of ciphers were under the impression that their data was secure. As we can see in the above example, transposition ciphers are not unsolvable by any means. In fact, much like the substitution ciphers that Caesar employed, simple transposition ciphers are common in newspapers and word game books. But once again, some of history's past cryptographers believed that a simple transposition cipher secured their data beyond that which could be cracked. This is another example of humans implementing a system that they believed to be un-crackable.

While other ciphers have existed throughout history, the substitution and transposition ciphers are most commonly used in today's digital world. Other ciphers will not be discussed in this document.

## Cryptography Comes of Age

As we have progressed as a species into the digital age, we've come to realize the shortcomings contained in ancient enciphering methods. While successful for periods of time, those times have changed. Substitution ciphers may be too obvious, and transposition ciphers may be too easy to crack, but they are not completely without merit in today's technically advanced world. While each of these individual ciphers alone does not offer a modern-day cryptographer great amounts of security, when combined, the resulting cipher can be very secure. The resulting combination of substitution and transposition in a cipher is called *diffusion*, a term coined by Claude Elwood Shannon in the 1940's.[7]

It is important to note that prior to the invention of the computer, ciphers in their simpler forms were deemed adequate. A cryptographer had only to outsmart his adversary for his data to be secure. Nowadays, the landscape of cryptography has changed so much that while a good diffusion cipher might be enough to outsmart even the smartest cryptanalyst; it's nowhere near strong enough to stump a computer. Since so much of modern cryptography depends on mathematics, it's only a matter of time before a computer can crack any code a cipher uses. Simply stated, over time ANY cipher that can be created can be deciphered. But just as we saw in the military example early in this paper, and again in the transposition cipher example, the silver lining in the cryptographic cloud is that your data only needs to be secure until it is no longer confidential or sensitive.

As we scroll through crypto history on our way toward to the 21st century, it becomes apparent that no single invention has affected the way we encrypt and decrypt data more than the computer. To administrators everywhere this statement may be answered with a resounding, "DUH!" but it is important to note because as soon as computers were brought into the crypto fold, the face of encryption changed forever.

## Recent Cryptography

As more and more computers began to crop up on the landscape, there began a dramatic shift in the way we saw and performed cryptographic functions. "Anything I can do, a computer can do better" became a common theme in cryptography labs everywhere, and for good reason. There's little debate that a computer can perform mathematical functions faster than a human. This fact is not disputed. The problem with computers as they relate to cryptography however, is not one of speed, but rather one of discipline. Before we get into the limitations a computer brings to the cryptographic table, let's first take a look at some of the undeniable bonuses that computers offer us.

Early on in the computer age, cryptographers began to realize how difficult securing data had become. The invention of data transfer over insecure lines only compounded this problem. They were nearing a crossroads, and the path they chose, to borrow a line from Robert Frost, has made all the difference.

At this crossroads, the cryptographers of the 20th century were faced with a very real challenge. As always, they first had a need to secure their data. Next, they needed to use a method that was incredibly complex, so that it would take a computer a lengthy amount of time to crack the code. Thirdly and most importantly, the true recipient must be able to decrypt the ciphertext quickly. Once again, our cryptographic predecessors had a difficult dilemma on their hands. Just as the Caesar Cipher would become obsolete when the masses learned how to read, so too would any encryption method that relied on the 'all things being equal' theory. Luckily, these men and women took their task to the extreme (or perhaps what may have been considered extreme if only at that time).

In the 1970's, Intel co-founder Gordon Moore stated that the number of transistors a silicon chip contains would double every 18 months without causing an increase in the price to the chip. This idea, commonly referred to as Moore's Law[8] has often been interpreted to mean that the speed of a pc will double every 1.5 years without adding significantly to the overall cost of purchase. Assuming this to be true, -for if Moore was wrong, it was only because 18 months was too long a timeframe- it would seem that there is very little a cryptographer could do to build an encryption scheme that could stand the test of time. Believe it or not, this may actually be a good thing.

## Modern Cryptography

We've taken a look at ancient encryption tactics, thumbed our way through the invention of the computer and scratched the surface on the dramatic impact this one machine has had on cryptography. We've seen examples of substitution and transposition ciphers, and all the while, one thing has held true. In each and every implementation, someone or something has eventually been able to crack the code and discover the key to decrypting the entire ciphertext message. But at the same time, we've stated that if something can be encrypted, it can also be decrypted. Nothing is sacred. So where does cryptography go from here?

Assuming a background in modular mathematics, it is important to note the latest functions and tactics in use in today's Internet / Instant Generation. If we encrypt a message, we want it done right now. If we send it to someone, we want it there yesterday. And when they receive it, we expect it to open without any problems so that the recipient can respond faster than we can call them to ask if they have received the message. In what is probably a shocking amount of cases, if all of these things do not happen, a person would rather send a plaintext message than hassle with encryption. This is the world we live in.

So as cryptographers approach today's date on the crypto timeline, they have a new challenge in front of them. Security may still be number 1, but seamless speed is 1A. While security administrators lose sleep at night worrying about things like authenticity, integrity, non-repudiation and confidentiality, the world sits in waiting for that 'best of both worlds' encryption standard. This is, a standard that can provide all of the services listed above and a heavy dose of speed, to boot.

Unfortunately the news is not all good. Modern encryption techniques have painted modern cryptographers into the proverbial corner. Without the computer, most of the schemes in use today cannot even be computed. Yet, without the computer, none of the schemes in use would even be necessary. Nevertheless, here we are.

**Services**

In order to gain the full benefit of cryptography, we need to understand the four services we touched on above, authenticity, integrity, non-repudiation and confidentiality. Detailed usage of the four cryptographic services is beyond the scope of this document. For our discussion it will suffice to define the purpose of each service in order only to establish its importance.

The four cryptographic services provide both the sender and receiver various benefits. Authenticity provides a mechanism for the sender and receiver to validate that either person is who they say they are. Integrity insures the sender and receiver that the data that was received was the actual data that was sent. This service confirms that the data was unaltered in transit. Non-repudiation provides assurance that the data was sent from the person claiming to be the sender. With non-repudiation services, the sender cannot deny having sent a file, email, etc. Confidentiality provides users with the assurance that only those with the key can decipher data encrypted for/with that key.[9]

**Modern Implementations and Keys**

As always, we've got a cipher and a key that unlocks that cipher. But now, a computer generates these functions. There's no way we could crack the cipher even if we wanted to spend the rest of our natural life trying. In fact, even if we sat in front of a computer that spent all day everyday crunching numbers, we still wouldn't see the solution in our lifetime. So this is good. We can feel relatively safe that our data is secure. But how do we get the information to our allies, and more importantly, how do we get the decryption key to them?

Most of the simple ciphers we discussed early on were those of the shared secret key variety. And much like today's encryption, they suffered from the same concerns surrounding shared secret key cryptography today. That is, the issue of how to distribute the key. If the information is sensitive enough to encrypt, isn't the key just as important (yes! more so even)? If we were to encrypt the key and send it to our allies, how will the recipient decrypt the message containing the very key itself? If the key is secret, should we not take great care in distributing it?

So once again, we're back in that pickle. We want the most secure data on the web, but our users aren't willing to pay the price necessary to ensure that the data remains useless in the event it falls into the hands of our adversaries. Is there an easier way to insure confidentiality? The short answer to that question is yes. Enter the public key / private key (referred to as simply public key) encryption system.

As we saw earlier, within the shared secret key system, the biggest hurdle facing a user (or security administrator) is the question of how to get the decryption key to the person they desire to share secure communications with. In public key cryptography, this hurdle is completely eliminated.

In the public key cryptography system, a user needs only to advertise his or her public key. The algorithm used to cipher data in the public key encryption scheme uses modular mathematics in order to generate two numbers that are modular inverses. It then applies one of the pair of inverses to the public key, the other to the public key.[10] The specifics of modular mathematics are beyond the scope of this document, and will not be discussed in detail. However, in order to build a better foundation for how public key encryption works, inverses will be discussed.

Let's assume that we have a piece of data we wish to send. In this example, the number 10 represents our data. In strictly mathematical terms, if we sought to multiply that message by a whole number, yet retain the same value as our original data; we know that we would multiply that number by 1. This theory is simple enough to most everyone who has taken a mathematics course.

$10 \times 1 = 10 =$ original data

Now, suppose that we sought to multiply our data (10) by two numbers, yet retain the same value as our original data. In order to achieve this, we'd need to multiply our data by two modular inverses (one for the public key, one for the private key) that equal 1. In this example we will use *multiplicative* inverses so as to familiarize the reader with the concept *only* behind public key cryptography's key encryption and decryption.

$10 \times \frac{1}{4} \times 4 = 10 =$ original data

Similarly,

$10 \times 12/13 \times 13/12 = 10 =$ original data

Therefore in order to cipher our data, the sender need only apply one half of a modular inverse (the public key) and recipient apply the other half of the modular inverse pair (the private key) in order to return the data to its original format and value (10).

Obviously, if a there are two keys, each with values so large that it would take a computer centuries upon centuries to crack –by today's hardware standards- this method is very secure. Why then does anyone even bother with using shared secret keys anymore? In part, the answer to that question goes back to User Rule 1A. Speed. On average, a shared secret key encryption is a least 100 times faster that public key cryptography. In fact, in some cases secret key cryptography is as much as 1000 faster.[11] That's enough to make users howl. On the one hand, we've got the faster, secret key encryption method that requires us to somehow get the key to our allies. On the other hand, with public key cryptography, we've got a freely distributed public key and a corresponding private key that we share with no one, yet we're saddled with incredible overhead. This overhead slows our encryption down to the point where users will complain that it's not fast enough. Isn't there a happy medium somewhere within? Again, the short answer to that question is yes.

**Best of Both Worlds Scenario (by today's standards)**

In order to achieve that elusive 'best of both worlds' scenario we seek, there is a simple solution. To sidestep the insecure method of sending a plaintext shared secret key to our ally, we first implement a public key encryption.

In this scenario, we have our safely guarded private key in hand, while our public key is widely distributed. As noted, our encrypted transmissions will suffer degradation in speed when we use public key encryption, but eventually, we will eliminate the usage of this encryption method altogether (except in very specific circumstances described below).

First, we confirm that our ally has our public key. Next, using our public key, our ally would then encrypt the shared secret key that we wish to use in encrypted communications going forward. Next, our ally would encrypt the shared secret key in our public key and send it off to us.

In the scenario described above, the shared secret key that our ally is sending to us is completely safe should it land in the hands of our enemies. When we receive the shared secret key, we need only to decrypt the data with our private key and implement the shared secret key. After we perform these steps, all ensuing transmissions between our ally and us can be encrypted and decrypted using the shared secret key. Voila! We've sidestepped the major difficulty of shared secret key encryption, namely key distribution, and we've avoided a major weakness of public key encryption, that of speed degradation.

Unfortunately, this is not the end of our saga. As we alluded to earlier, there are four cryptographic services. Each of these services provides a very important function to encrypted transmissions. If we only seek to insure confidentiality, integrity and authentication, then the scenario described above would fully suffice for our operations. Unfortunately, if we also need to insure that we also have non-repudiation services on our side, the above scenario will not accomplish our goal.

Although shared secret key encryption has its advantages over public key encryption (namely speed), it also has limitations. The most obvious of these limitations is the inability to provide non-repudiation services.

Shared secret key encryption and public key encryption both have weaknesses. Neither is the be-all end-all of Internet encryption. But as you can see, if we use each encryption scheme toward a very specific goal, we can implement a solution that is both fast and secure…for now

## Summary

The cryptography landscape is more like a mountain than a molehill. From the very beginning, the evolution of encryption has been building like a huge, virtual snowball. With iterations of speed continually being built into the microprocessor, and the price of RAM continuing to drop, the mathematical algorithms that generate the encryption schemes and keys come closer and closer to being cracked. As far back as Julius Caesar, cryptographic methods that were eventually rendered useless were commonly put into use. Today is no different. The cutting-edge crypto algorithms of today will be broken in almost no time in the future. Routines that take months to finish today, will take minutes and possibly seconds to finish in the future. The cycle is never-ending.

The problem with this never-ending cycle of technology, as it relates to cryptography is not that a larger algorithm is bad, but rather that for as long as we use the same encryption scheme, we must continue on the perpetual treadmill of building faster and faster encryption computers to compete against faster encryption encryption-cracking computers.

The obvious fact that is in our favor nowadays is that we realize that no encryption scheme is rock solid. If an encryption cipher can be composed, it can be cracked. Therefore, we must continually assess our encryption standards, lest we become doomed to repeat the mistakes of history's past cryptographers.

## References

[1] Mel, H.X.; Baker, Doris.  Cryptography Decrypted: Boston: Addison Wesley, 2001. 6

[2] CBSNews.com. URL: http://www.cbsnews.com/htdocs/spies/timeline.html (1/13/2003)

[3] Safehack.com. “Caesar Cipher” URL: http://www.safehack.com/Textware/cissp/Crypto/Domain4_Crypto-142.htm (1/13/2003)

[4] New Mexico State University. URL: http://www.math.nmsu.edu/crypto/public_html/Transposition.html (1/13/2003)

[5] Absolute Lyrics. “BareNaked Ladies: Brian Wilson” URL: http://www.absolutelyric.com/a/view/Barenaked Ladies/Brian Wilson/ (1/13/2003)

[6] Atkinson, Julie; HSS August 2002 Newsletter. URL: http://hampsteadscience.ac.uk/HSS_Aug_2002.htm (1/13/2003)

[7] Synaxis.org. “Dave’s Crypto Notes” URL: http://www.synaxis.org/bastion/crypto (1/13/2003)

[8] Webopedia.com. “Moore’s Law” URL: http://www.webopedia.com/TERM/M/Moores_Law.html (1/13/2003)

[9] Bennett, Larry; “Cryptographic Services – A Brief Overview” URL: http://www.sans.org/rr/encryption/crypto_services.php (1/13/2003)

[10] Netscape DevEdge. “Public Key Cryptography” URL: http://developer.netscape.com/docs/manuals/security/pkin/contents.htm (1/13/2003)

[11] Mel, H.X.; Baker, Doris.  Cryptography Decrypted: Boston: Addison Wesley, 2001. 157

Last Updated: July 26th, 2017

# Upcoming Training

Click Here to
{Get CERTIFIED!}

| | | | |
|---|---|---|---|
| SANS Prague 2017 | Prague, Czech Republic | Aug 07, 2017 - Aug 12, 2017 | Live Event |
| SANS Boston 2017 | Boston, MA | Aug 07, 2017 - Aug 12, 2017 | Live Event |
| Community SANS Omaha SEC401* | Omaha, NE | Aug 14, 2017 - Aug 19, 2017 | Community SANS |
| SANS New York City 2017 | New York City, NY | Aug 14, 2017 - Aug 19, 2017 | Live Event |
| SANS Salt Lake City 2017 | Salt Lake City, UT | Aug 14, 2017 - Aug 19, 2017 | Live Event |
| SANS Virginia Beach 2017 | Virginia Beach, VA | Aug 21, 2017 - Sep 01, 2017 | Live Event |
| SANS Adelaide 2017 | Adelaide, Australia | Aug 21, 2017 - Aug 26, 2017 | Live Event |
| Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style | Virginia Beach, VA | Aug 21, 2017 - Aug 26, 2017 | vLive |
| SANS Chicago 2017 | Chicago, IL | Aug 21, 2017 - Aug 26, 2017 | Live Event |
| Community SANS Pasadena SEC401 @ NASA | Pasadena, CA | Aug 23, 2017 - Aug 30, 2017 | Community SANS |
| Mentor Session - SEC401 | Minneapolis, MN | Aug 29, 2017 - Oct 10, 2017 | Mentor |
| SANS San Francisco Fall 2017 | San Francisco, CA | Sep 05, 2017 - Sep 10, 2017 | Live Event |
| SANS Tampa - Clearwater 2017 | Clearwater, FL | Sep 05, 2017 - Sep 10, 2017 | Live Event |
| Mentor Session - SEC401 | Edmonton, AB | Sep 06, 2017 - Oct 18, 2017 | Mentor |
| SANS Network Security 2017 | Las Vegas, NV | Sep 10, 2017 - Sep 17, 2017 | Live Event |
| Community SANS Albany SEC401 | Albany, NY | Sep 11, 2017 - Sep 16, 2017 | Community SANS |
| Mentor Session - SEC401 | Ventura, CA | Sep 11, 2017 - Oct 12, 2017 | Mentor |
| Community SANS Columbia SEC401 | Columbia, MD | Sep 18, 2017 - Sep 23, 2017 | Community SANS |
| Community SANS Dallas SEC401 | Dallas, TX | Sep 18, 2017 - Sep 23, 2017 | Community SANS |
| Community SANS Boise SEC401 | Boise, ID | Sep 25, 2017 - Sep 30, 2017 | Community SANS |
| Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style | Baltimore, MD | Sep 25, 2017 - Sep 30, 2017 | vLive |
| Community SANS New York SEC401 | New York, NY | Sep 25, 2017 - Sep 30, 2017 | Community SANS |
| Rocky Mountain Fall 2017 | Denver, CO | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| SANS London September 2017 | London, United Kingdom | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| SANS Baltimore Fall 2017 | Baltimore, MD | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| SANS Copenhagen 2017 | Copenhagen, Denmark | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| Community SANS Sacramento SEC401 | Sacramento, CA | Oct 02, 2017 - Oct 07, 2017 | Community SANS |
| SANS DFIR Prague 2017 | Prague, Czech Republic | Oct 02, 2017 - Oct 08, 2017 | Live Event |
| Community SANS Charleston SEC401 | Charleston, SC | Oct 02, 2017 - Oct 07, 2017 | Community SANS |
| Mentor Session - SEC401 | Arlington, VA | Oct 04, 2017 - Nov 15, 2017 | Mentor |
| Community SANS Indianapolis SEC401 | Indianapolis, IN | Oct 09, 2017 - Oct 14, 2017 | Community SANS |