301.3

An Introduction to Cryptography

SEC301.3 | Introduction to Cyber Security

SANS

An Introduction to Cryptography

Na Vagebqhpgvba gb Pelcgbtencul

This page intentionally left blank.

Module 8: Intro to Crypto

- Definition of Cryptography
- Fundamental Terminology
- Branches of Secret Writing
- History of Cryptography

COURSE ROADMAP

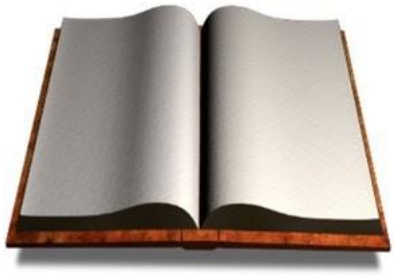
- **Module 8: Intro to Crypto**
  - Lab 3.1: Crypto by Hand
- Module 9: Building Blocks of Modern Crypto
  - Lab 3.2: Visual Crypto
- Module 10: Data Encrypting Protocols

Module 8: Intro to Crypto

In this module, we begin with our definition of cryptography and fundamental terminology. We introduce the "branches of secret writing," which we then refer to throughout much of the upcoming lecture. Finally, we look at some historical examples of cryptography and examples of each.

Recommended Reading

- *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* by Simon Singh
  - https://amzn.to/2Gnnr2C
  - ISBN: 978-0385495325
  - Extremely good, simple explanations on cryptography



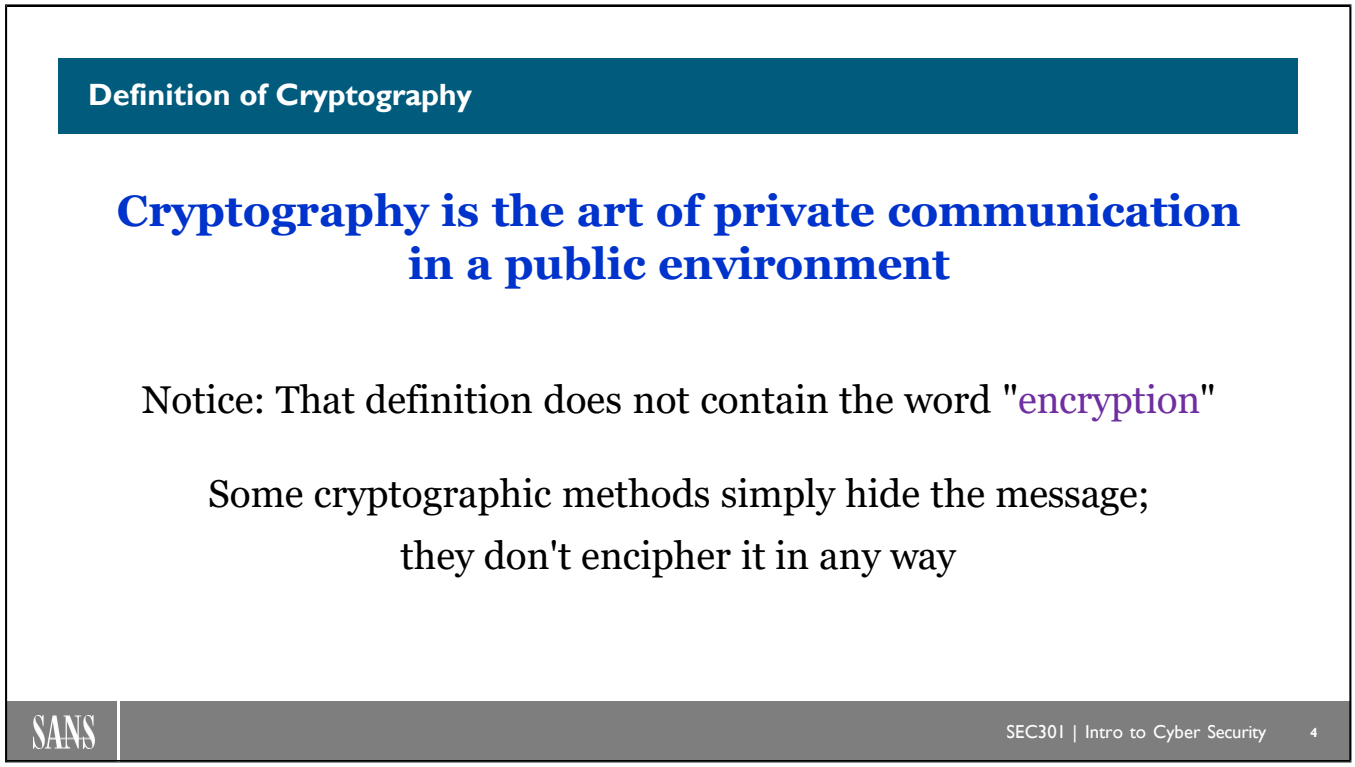
- Also recommended:
  - http://www.braingle.com/brainteasers/codes

Recommended Reading

If you need a book that offers good explanations of cryptography that are sufficiently detailed, yet easy to understand, and don't include the math… this book is highly recommended. This qualifies as the best book on cryptography the author of this course has ever read and one of the best technical books of any kind.

This website also offers some excellent explanations:

http://www.braingle.com/brainteasers/codes/index.php



Definition of Cryptography

Cryptography defined: "**The art of private communication in a public environment**."

If you go on the internet, you can find many similar definitions, but they all come down to the same thing. Anything that enables us to communicate in a way that we know what we are saying, and our friends know what we are saying, but our adversaries don't. Or to say it another way, anything that allows for private communication in a public environment.

You might notice that the word "encryption" does not appear in that definition. This is proper. Some forms of "private communication" deal with hiding the meaning of the message through encryption. Some methods deal with hiding the fact that the message exists at all but not with encrypting that message (called Steganography). As you will see, you can combine the two for added security.

Fundamental Terminology (1)

- Plaintext/Ciphertext
  - Plaintext =human readable
    - Also; Cleartext
  - Ciphertext = non-human readable
- Encryption
  - Transform readable to nonreadable
  - Also; Encipherment
- Decryption
  - Transform nonreadable back to original readable
  - Also; Decipherment

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.5 (GNU/Linux)

hQIOAOuHn1ue4n32EAf/UEF6JLrap1OBMdKMvb+Dz9GvoijUixH+gbcpi9qGa+43
vC3ktMwo7OWqPyJseVRSPBOv6dOwy65KrzrHwhOHO/CKEk2O5STAwzj6C3USgDfZ
6E+Gc4iumM1725JNahJzcL5ED33LFdZ6uoEjgqggxG1dFwvwksRHA4+VU9Bcd5eL
T9aRVbkXNxXkQn2FWhWuhPQFNWLwIVrDd9TPtDvpRT16YiB1AM9ks3HlYZHL7mfR
Hk9yfy1nGXdhi06EDvvTvd/Lq1xsFjKh6y/pG6NxABGdT6VoeWGVtQGqwpbOZGgq
xoSYkWm8MmAkkqYXZLraSEzyxxxu4cQzvzz3vrpN3AgAhObP2eUFU29EJAQpdKJW
fKAhohPVpd6+ETnzL53VLg1IJJdNG1pIziO9alNnYmDSnt2EwAELqTUl3jPiGYt5
cvSUBe3ER4/CkjvYXOVaO7ezHmCAkQpB2ILV8OwI74DQn7tNKf2gJnwzkYAF7yyf
XFG1J8oaLpRV499mN7lNfo+ZV2HrR9xti+jUPFv+H+ROt4fMmAU5I95UksQFe/A9
YUdSBAEqKkW9zLDgpWS2oxJymGufBdhzxpw7uJlzrwsHIYIt7PSeJG4VO+xJqHvO
1qHXSukK648F1OImmVUM9csPOcvfOMZeAgh4i+HYQvFF/kGHp6ogevD4pVhztbzd
F9JhAbJSeOvZKZFPhzjgX+mCgvzVRniSdDg7wc3+YKNei2zQrmTsii06JyhQV2OI
tAqTk572zdZbrCtSgcthrN/uxbJSNnw4X9IZbWtFOUr3lr676II8Q112ttO3IVCe
fF/pZA==
=sPWf
-----END PGP MESSAGE-----
```


Fundamental Terminology (1)

There is some fundamental terminology that you must know to understand the future discussion—or indeed, cryptography itself. The terms follow:

- **Plaintext:** What you are reading now—text in a human-readable form.
- **Ciphertext:** The plaintext transformed into a non-human-readable (or computer readable) form. You will see several examples in this course.
- **Encryption:** The process of turning plaintext into ciphertext. Transform readable to nonreadable
- **Decryption:** The process of turning ciphertext back into plaintext. Transform nonreadable back to the original human-readable form.


```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.5 (GNU/Linux)

hQIOAOuHn1ue4n32EAf/UEF6JLrap1OBMdKMvb+Dz9GvoijUixH+gbcpi9qGa+43
vC3ktMwo7OWqPyJseVRSPBOv6dOwy65KrzrHwhOHO/CKEk2O5STAwzj6C3USgDfZ
6E+Gc4iumM1725JNahJzcL5ED33LFdZ6uoEjgqggxG1dFwvwksRHA4+VU9Bcd5eL
T9aRVbkXNxXkQn2FWhWuhPQFNWLwIVrDd9TPtDvpRT16YiB1AM9ks3HlYZHL7mfR
Hk9yfy1nGXdhi06EDvvTvd/Lq1xsFjKh6y/pG6NxABGdT6VoeWGVtQGqwpbOZGgq
xoSYkWm8MmAkkqYXZLraSEzyxxxu4cQzvzz3vrpN3AgAhObP2eUFU29EJAQpdKJW
fKAhohPVpd6+ETnzL53VLg1IJJdNG1pIziO9alNnYmDSnt2EwAELqTUl3jPiGYt5
cvSUBe3ER4/CkjvYXOVaO7ezHmCAkQpB2ILV8OwI74DQn7tNKf2gJnwzkYAF7yyf
XFG1J8oaLpRV499mN7lNfo+ZV2HrR9xti+jUPFv+H+ROt4fMmAU5I95UksQFe/A9
YUdSBAEqKkW9zLDgpWS2oxJymGufBdhzxpw7uJlzrwsHIYIt7PSeJG4VO+xJqHvO
1qHXSukK648F1OImmVUM9csPOcvfOMZeAgh4i+HYQvFF/kGHp6ogevD4pVhztbzd
F9JhAbJSeOvZKZFPhzjgX+mCgvzVRniSdDg7wc3+YKNei2zQrmTsii06JyhQV2OI
tAqTk572zdZbrCtSgcthrN/uxbJSNnw4X9IZbWtFOUr3lr676II8Q112ttO3IVCe
fF/pZA==
=sPWf
-----END PGP MESSAGE-----
```

Fundamental Terminology (2)

- Algorithm:
  - The public knowledge set of rules behind cryptography
    - Modern cryptosystems utilize math for this
  - Two modern Algorithms we will focus on tod ay:
    - **Data Encryption Standard (DES) / Triple DES**
    - **Advanced Encryption Standard (AES)**
    - There are many others—these are the most common
- Cryptanalysis:
  - The art and science of breaking cryptography
  - Also called Cryptanalytic Attack

Fundamental Terminology (2)
Continuing with the fundamental terminology:

Algorithm: The public knowledge set of rules behind cryptography. As you will see, older methods of encryption used fairly simple methods. Modern ciphers utilize highly complex mathematical functions to accomplish the encryption of data.

During the discussion here, we will focus on the two most common modern algorithms: Data Encryption Standard (DES) and Advanced Encryption Standard (AES). Note that there are many, many other encryption algorithms around.

Cryptanalysis, or Cryptanalytic Attack: The art and science of breaking cryptography. Throughout history, the folks wanting to hide their communications have gotten better at doing so (then, the people wanting to read those messages come up with a better way to break it), so the crypto folks get better at hiding, and the attackers get better at breaking, and so on. The cycle has repeated itself for many years and shows no sign of slowing down.

Fundamental Terminology (3)

- Key: A numeric value of a given length (expressed in bits)
  - The secret that must be protected
  - The changing part of the algorithm
- Keyspace: The range of values that can be used to construct a key
  - The total of all possible combination of 1's and 0's given a specific number of binary bits
  - Add a bit, the number of combinations doubles

For example: The Data Encryption Standard (DES) algorithm uses a 56-bit Key. The keyspace with that number of bits is 72 followed by 15 zeros or 72,000,000,000,000,000 or 72 quadrillion (more on this later…)

Fundamental Terminology (3)
Key: A numeric value of a given length (expressed in bits). The changing part of the encryption algorithm that you must protect.

Keyspace: The range of values that can be used to construct a key given a particular key length. For example, the Data Encryption Standard (DES) encryption algorithm uses a key of 56 bits. Given that length, all possible combinations of 56 bits are roughly 72 quadrillion combinations. That is the size of the keyspace for DES.

If you have a hard time understanding (or picturing) 72 quadrillion, see:
https://numbermatics.com/n/72000000000000000/

Services of Cryptography

- **Confidentiality:**
  - "I encrypt my email, so you can't read it."
  - Perfectly valid use of crypto
  - But not all you can get …
- **Integrity:**
  - Ensuring the message sent is the message received
- **Authentication:**
  - Verify the identity of a person or system
- **Nonrepudiation:**
  - The person that sent a message cannot deny doing so

NOTE: Cryptography does not address Availability.

Except, if you don't have the key, it becomes Anti-Availability! Think Ransomware!

Cryptography assists with meeting all four other goals.

Services of Cryptography

When we talk about the services of cryptography, it is important to note what the process does and does NOT provide.

The most common reason people employ cryptography is for **confidentiality**. "I'm going to encrypt my email, so you can't read it." This is, of course, a perfectly valid reason for using cryptographic methods. However, this is not all you can get from the use of crypto.

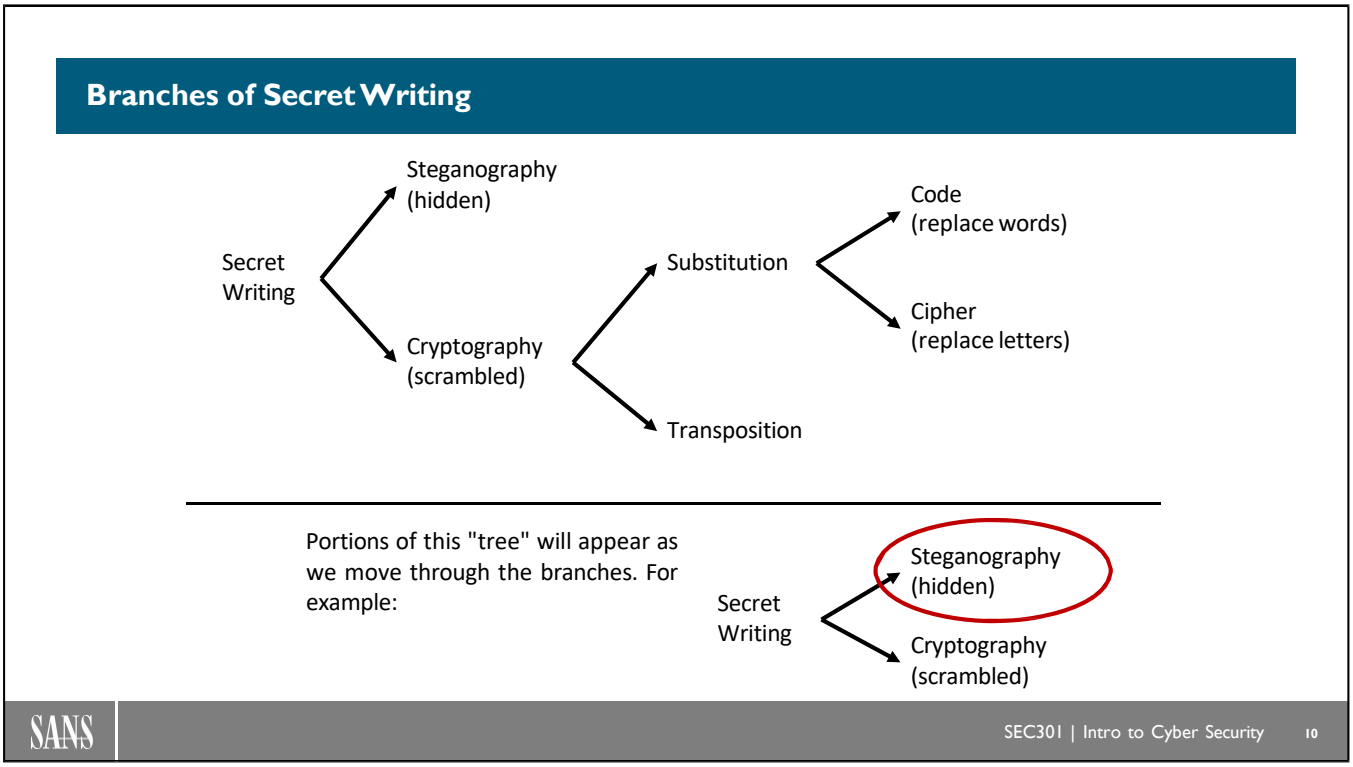
You can also obtain **integrity**. In other words, you can ensure the message sent and the message received are identical. You can take that assurance to the point of mathematically provable fact.

Virtually every **authentication** system utilizes cryptographic techniques in one way, shape, or form, and many utilize multiple methods.

Finally, with proper implementation, you can obtain something called **nonrepudiation**. We don't have all of the pieces and parts in place just yet to understand how we accomplish this, but we will soon. For now, simply know that cryptographic systems can make nonrepudiation possible, meaning the sender cannot deny sending a document and the sender cannot say the document you received is different from the one they sent. Details to come …

It is important to note that cryptography does not directly address availability. For example, if I encrypt a file, you might not be able to read the file, but you may be able to delete the file. Cryptography does not address that. In fact, the only way that cryptography (sort of) addresses availability is if you were to lose your key. Then you would be unable to access your data. In this way, cryptography has the potential to be Anti-Availability!

Any good information system provides Confidentiality, Integrity, Availability, Authenticity, and Accountability. Cryptography helps us to address all of these *except* availability. Encryption provides the confidentiality, while Digital signatures help us to address integrity, authenticity, and accountability.



Branches of Secret Writing

Here you see the "branches of secret writing." Over the next several slides, we move through each of these branches and explain what they mean. As we do so, you can see the branch we are currently discussing show up on the slide someplace (similar to the bottom of this slide).

In other words, we use this as something of a roadmap for the next several topics of the course.

SEC301.3 | Introduction to Cyber Security

SANS

Steganography
- Hidden / Concealment Ciphers

Transposition

Substitution

Mono- and Polyalphabetic Ciphers

Codes and Code Books and One-time Pads

History of Cryptography

History of Cryptography
As the slide indicates, we are going to move through hidden and concealment ciphers (steganography). Then we discuss transposition ciphers, followed by substitution ciphers and both mono- and polyalphabetic ciphers. We end this section with an examination of Code Books and One-time Pads.

Modern ciphers (DES, AES, and more) use transposition, substitution, and so on. Actually, they use them very much the same way as the methods you are about to see. It's just that the modern cipher:

1. Does each of them in MUCH more complex ways
2. Uses all of them together one after the other
3. Performs those multiple steps many times

Any of these historical methods you see here, by itself, is trivial to break. It is the added complexity, combination of multiple methods, and repeating those methods several times that give us the strength of modern ciphers.

Why the History Lesson?!?!

- To help you understand modern crypto:
  - If you understand the simple, historical examples
  - Understanding modern crypto is *much* easier
- In crypto, there is little new:
  - There are modern, advanced ways of doing old things
  - Even the "Advanced Encryption Standard" uses methods of encryption from 100s of years BC:
    - It just calls them by new names
    - And does them in much more complex ways

SANS SEC301 | Intro to Cyber Security 12

Why the History Lesson?!?!

We learned quite some time ago that when you need someone to understand cryptography, it helps to start at its beginning. There are many examples of "private communication in a public environment" throughout history. Indeed, the examples date back to the dawn of recorded time. Many of these old methods of encryption were quite ingenious for their time. Today, they are considered simplistic. But it is that simplicity that makes them valuable in explaining cryptography.

If you understand these simple historical examples of cryptography, understanding modern cryptographic methods is much simpler. Why? Because in crypto there is little new. There are significantly more advanced ways of doing very old things.

Even the "Advanced Encryption Standard" (currently considered the strongest publicly available crypto algorithm) uses methods of encryption from 100s of years BC. It calls them by new names. It does them in *much* more complex ways. But if you understand how the old methods work, you understand how they are being applied in the modern world.

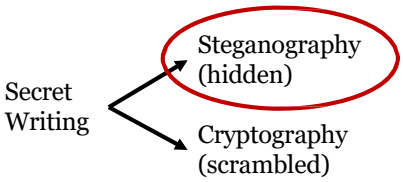
Bear with us. First, we will show you the old, easy to understand methods of cryptography. Later, when we get to the modern algorithms (Advanced Encryption Standard in particular), we will show you how it uses these same functions.

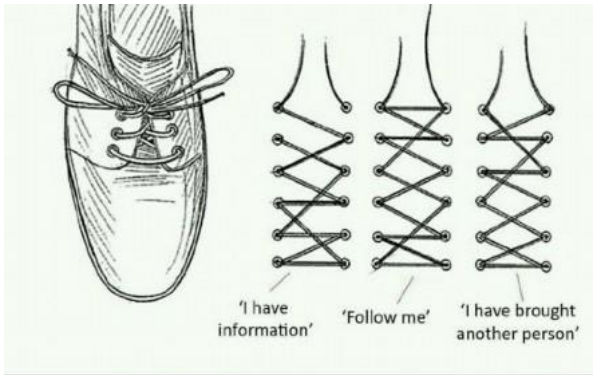
Jeremiah Denton POW in Vietnam – Blinks Torture in Morse Code – Link in the notes...

Hidden Ciphers: Origins of Steganography

- The shaven head
- Invisible ink
- Pin pricks
- Shoelaces
- Sir John Trevanion
- Steganography

Slides to follow





During the Cold War, CIA agents would secretly communicate with each other based on how their shoelaces were tied.

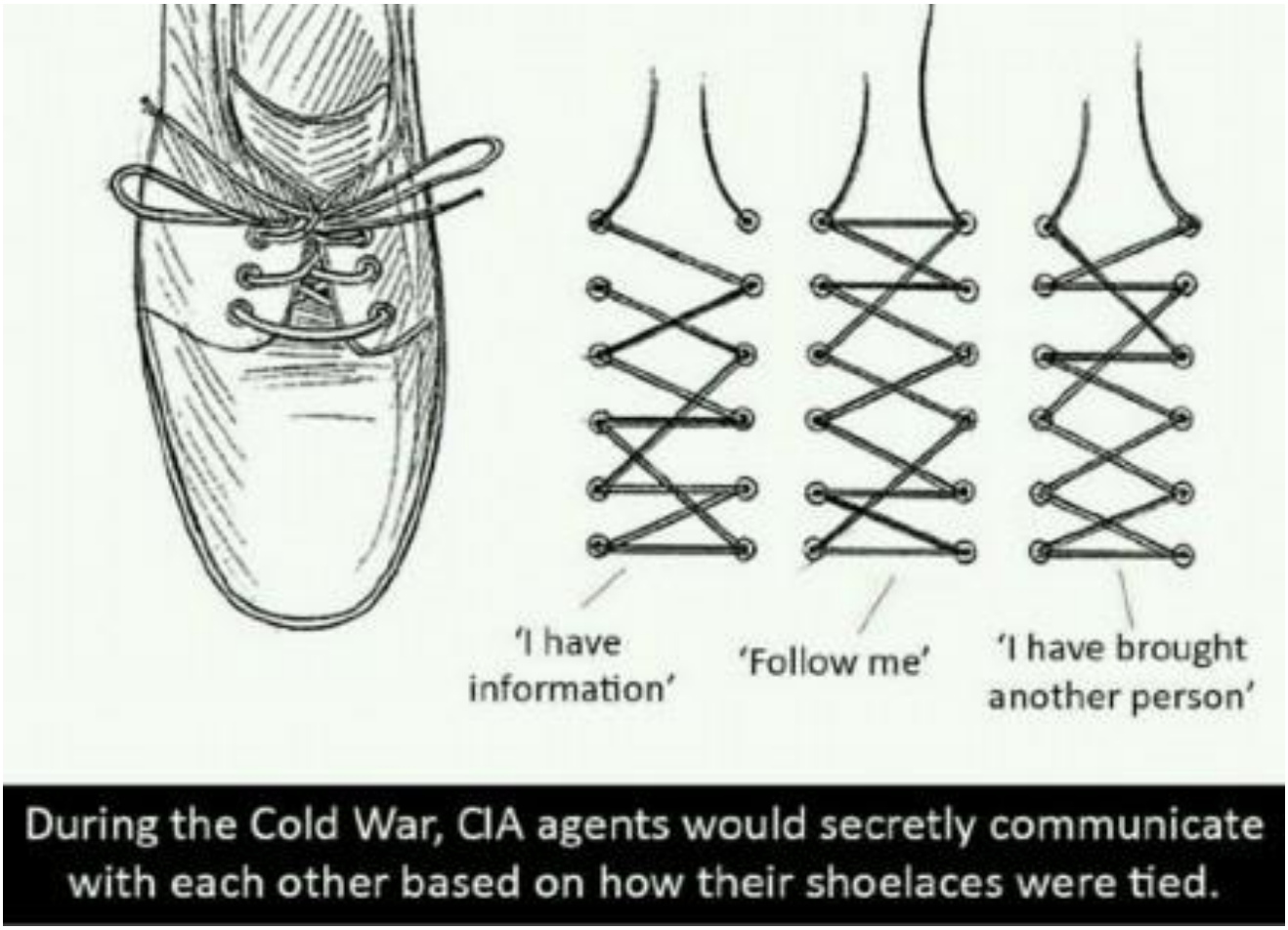
Hidden Ciphers: Origins of Steganography
Examples of humans secreting information from other humans goes back as far as recorded time. One of the earliest examples dates to several hundred years BC. People would take a slave and shave their head, tattoo a message to their scalp, and wait for the hair to grow back. Then they would send the slave to the other side, shave the head, and read the message.

There are also several examples of invisible ink, both manufactured and naturally occurring. For example, take 100% pure lemon juice, a sharp stick, and a piece of paper; dip the stick in the lemon juice and write a message. When it dries, you cannot see the message. Take the paper to a candle and heat it slowly: the carbon in the lemon juice turns gray and you can read the message. (Heating the paper with an iron works as well.) There are other naturally occurring invisible inks as well.

During World War II in Great Britain, Nazi sympathizers communicated in secret by a simple method. They'd take a newspaper and a needle and poke a hole above certain letters in the newspaper. Once the newspaper was folded, nobody could see there was a message hidden there. One would then drop the newspaper in a predetermined garbage can where their compatriot could pick it up. Once the newspaper was unfolded and held up to the light, the light shined through the holes and the message would be revealed.

These last two examples, Sir John Trevanion's Letter and modern steganography, will be covered on slides of their own.

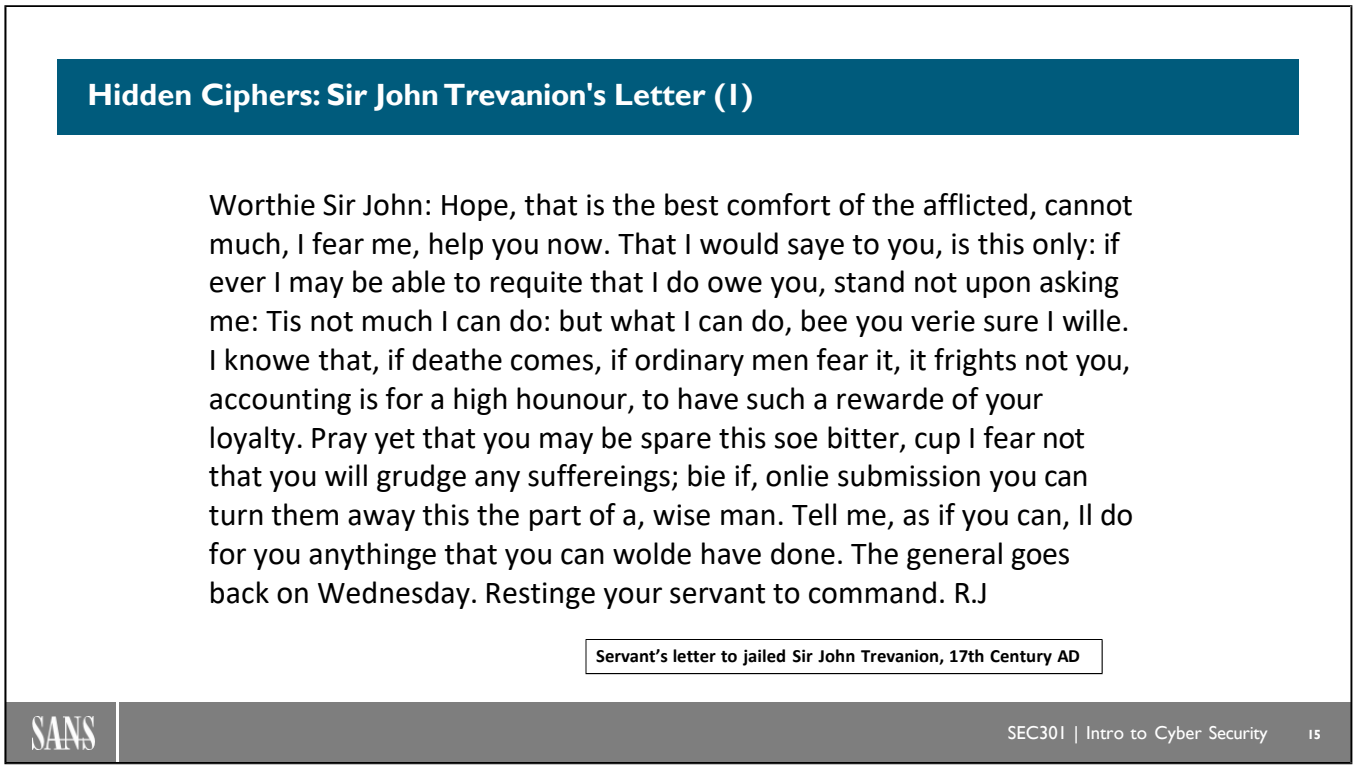
Stringing together a secret message …



Another simply amazing example of steganography is Jeremiah Denton. He was a POW in North Vietnam. Forced to film a propaganda video, he blinks very precisely. Specifically, he blinks Morse code for the letters "torture".

You can see a video of Admiral Denton's video here:

https://www.youtube.com/watch?v=ioC_F8FvviM

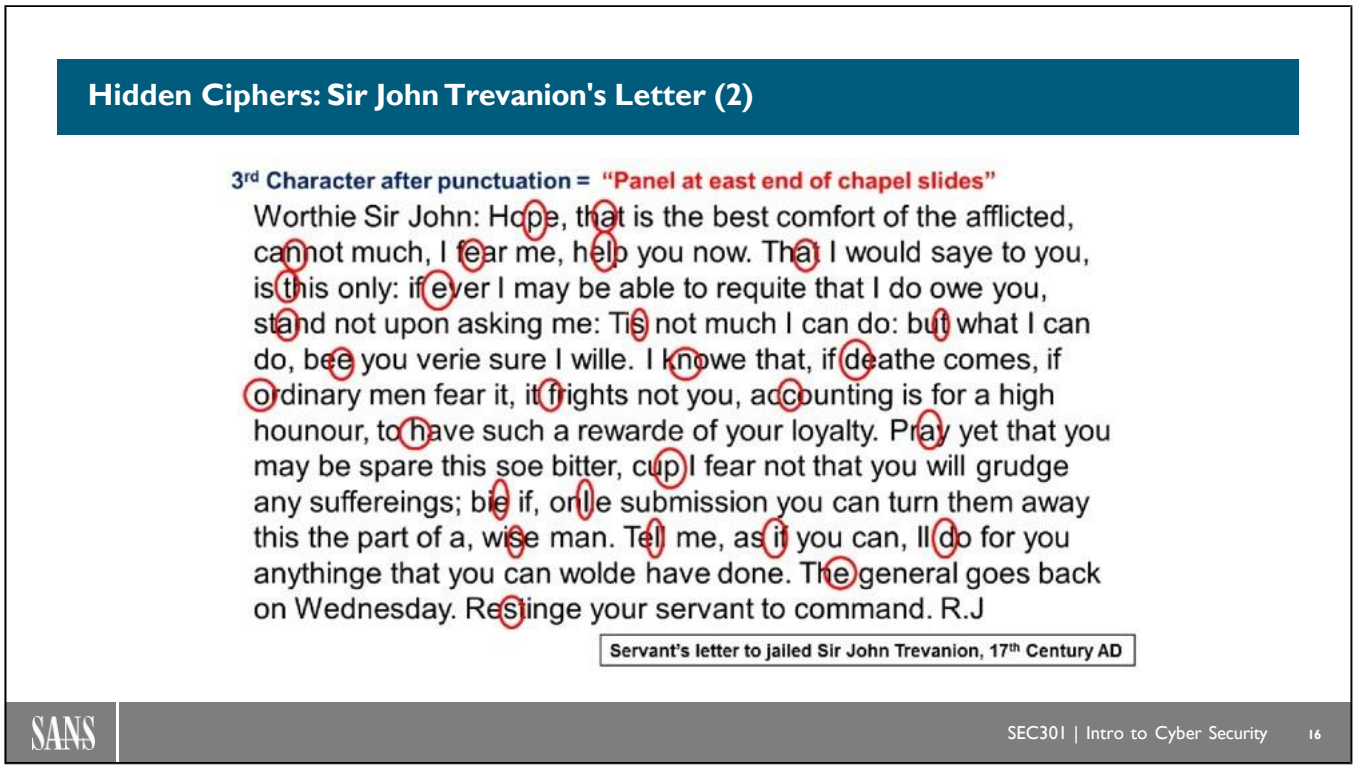


Hidden Ciphers: Sir John Trevanion's Letter (1)
A famous (and ingenious) example of steganographic communication comes from the 17th century. Sir John Trevanion had been imprisoned and sentenced to death. While in prison, one of his servants sent him the letter you see above. If you begin reading through it, you see that it just sort of rambles on and does not really say too much of importance. Therefore, Sir John's guards didn't see a problem with giving it to him.

Now take a closer look. You might notice that throughout the letter, there is punctuation that can be described only as "strange." And that is the trick that Sir John was looking for.

The answer to the puzzle is on the next page when you are ready.

HINT: Look at the third character after any punctuation mark.



Hidden Ciphers: Sir John Trevanion's Letter (2)
The Trevanion Cipher, as it has become known, is not actually a true cipher at all. It is a form of steganography.

The trick here is that Sir John is looking for the third character after any punctuation mark. Look at the first few sentences:

Worthie Sir John: Hope, that is the best comfort of the afflicted, cannot much, I fear me, help you now.

So, you have a colon, with a third character after it of P. Then, a comma and a third character of A, and so on.

If you go through the entire letter extracting those third characters after punctuation, you arrive at the secret message: "**Panel at east end of chapel slides**."

Worthie Sir John told his captors that before they put him to death that he would like an hour of quiet reflection in the chapel. They allowed him to go into the chapel alone. Instead of doing quiet reflection, he slid the panel and crawled out through a tunnel. He lived for several more years.

A Modern Hidden Cipher: Steganography (1)

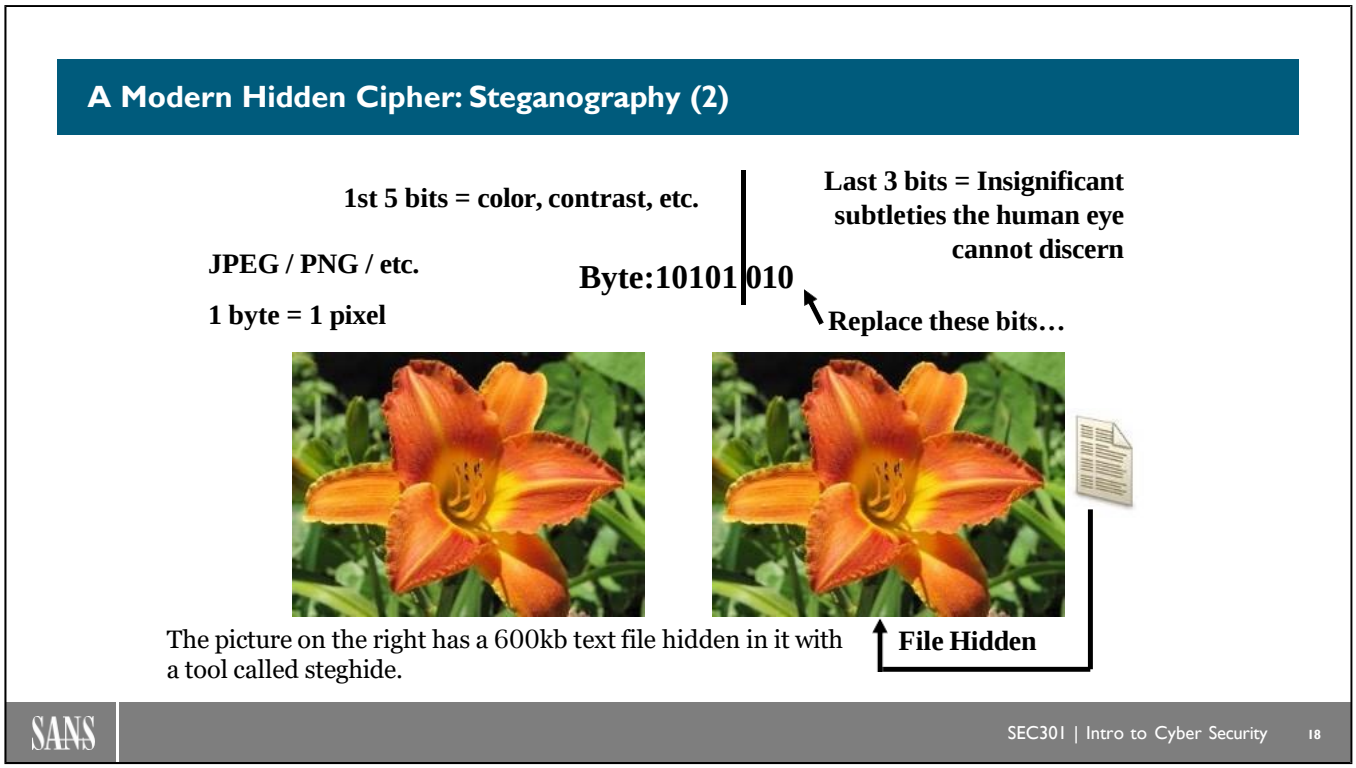
- Technically, all hidden ciphers = Steganography
  - Greek "steganos" = "covered"
- Today, it has a different meaning and usage:
  - Hiding a message inside a picture, movie, or similar type of file
  - Example follows …

A Modern Hidden Cipher: Steganography (1)
All the "ciphers" we have discussed to this point are examples of steganography—and history is full of examples.

The term actually comes from the Greek words "steganos," which means "covered," and "graphy," which means "writing." So, steganography literally translates to "covered writing."

Although history is full of examples of covered writing, in the modern world, steganography is accomplished in ways the ancients could never have imagined. There are many methods for performing steganography with computer files and far more pieces of software to assist in the matter.

Here, we look at one popular method as an example.



A Modern Hidden Cipher: Steganography (2)
One popular method of hiding messages involves the use of pictures. Again, there are many ways of using graphics for this, but we will concentrate on a method that works with JPG files.

When you look at how a JPG is constructed, each byte of the file defines one pixel. The first 5 bits define the color, contrast, and such. The last 3 bits define insignificant subtleties the human eye cannot discern. It is those 3 bits we will use here.

The picture on the left is just a JPG image. The picture on the right is the same image but has a 60kb text file hidden inside it. We used the tool called "steghide" to accomplish this.

The human eye is unable to discern a difference in these two pictures because the text file has only overwritten one of the last 3 bits of each byte, where the insignificant subtleties are stored. A cryptographic hash (which we will discuss shortly) could tell us *that* the images are different but not *how they are different*. And to perform a comparison of that nature would require that you have both the before and after images as you do here.

In testing, if you embed a large enough file that it overwrites all three of the last bits of most of the bytes, there is indeed a slight degradation of the image. But without the before and after, it would be hard to tell.

Reference
Steghide is available from http://steghide.sourceforge.net/

A Modern Hidden Cipher: Steganography (3)

- If the message is first encrypted, it cannot be discovered:
  - Rumor is that organized crime and terrorists use this
  - Causes problems for forensics investigators
- They are also used for digital watermarking to make it difficult to copy a file undetected

A Modern Hidden Cipher: Steganography (3)
As stated, we used steghide to embed the text file inside the JPG file on the previous slide. When we ran that software, it prompted us for a passphrase. This typically means (and in the case of steghide **does** mean) the text file will be encrypted before it is embedded. Usually, the passphrase becomes the encryption key—so choose a good one.

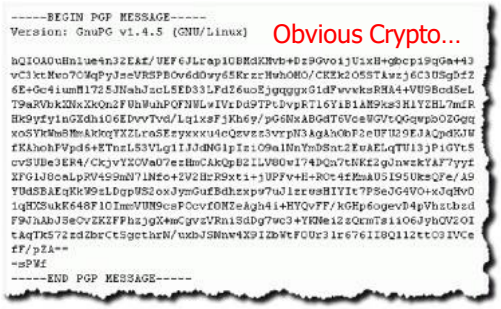
With this particular method of steganography, if you first encrypt the message and then embed it into a JPG, nobody yet has figured out how to tell there is a message there.

Rumor has it that this method of communication is used today, and for several years now, by both organized crime and terrorists. Because they are first encrypting and then embedding, we cannot prove it is happening. It is widely believed to be true among the law enforcement community.

Incidentally, steganography is not just for evil purposes; it can be used for good as well. For example, it would be possible to use steganographic technique to digitally watermark the image on the previous slide. We could then prove plagiarism if someone copies the image. Even if they edited the image, a portion of the watermark will survive.

Steganography Versus Encryption

- Encryption (our next topic) provides:
  - **"Confidentiality of communication"**
  - NOT "Secrecy of communication"
  - Nobody else knows what you are saying, but they can tell you are talking privately
  - Humans and computers can spot cyphertext
- Steganography provides:
  - **"Secrecy of communication"**
  - Nobody knows the parties are even talking
  - Combined with crypto, you can get the best of both



Just a pretty picture...

Steganography Versus Encryption

There is a fundamental difference between encryption and steganography. Encryption provides for "confidentiality of communication." It does *not* provide for "secrecy of communication." In other words, when you use encryption by itself, other people cannot tell what you are saying. But they *can* tell you are talking privately. Humans and computers can spot cyphertext easily.

By contrast, steganography attempts to obtain secrecy of communication. So, nobody even realizes two parties are talking. Further, when you combine steganography with encryption, you can get the best of both.

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.5 (GNU/Linux)

hQIOAOuHn1ue4n32EAf/UEF6JLrap1OBMdKMvb+Dz9GvoijUixH+gbcpi9qGa+43
vC3ktMwo7OWqPyJseVRSPBOv6dOwy65KrzrHwhOHO/CKEk2O5STAwzj6C3USgDfZ
6E+Gc4iumM1725JNahJzcL5ED33LFdZ6uoEjgqggxG1dFwvwksRHA4+VU9Bcd5eL
T9aRVbkXNxXkQn2FWhWuhPQFNWLwIVrDd9TPtDvpRT16YiB1AM9ks3HlYZHL7mfR
Hk9yfy1nGXdhiO6EDvvTvd/Lq1xsFjKh6y/pG6NxABGdT6VoeWGVtQGqwpbOZGgq
xoSYkWm8MmAkkqYXZLraSEzyxxxu4cQzvzz3vrpN3AgAhObP2eUFU29EJAQpdKJW
fKAhohPVpd6+ETnzL53VLg1IJJdNGlpIziO9alNnYmDSnt2EwAELqTU13jPiGYt5
cvSUBe3ER4/CkjvYXOVaO7ezHmCAkQpB2ILV8OwI74DQn7tNKf2gJnwzkYAF7yyf
XFG1J8oaLpRV499mN7lNfo+ZV2HrR9xti+jUPFv+H+ROt4fMmAU5I95UksQFe/A9
YUdSBAEqKkW9zLDgpWS2oxJymGufBdhzxpw7uJlzrwsHIYIt7PSeJG4VO+xJqHvO
1qHXSukK648F1OImmVUM9csPOcvfOMZeAgh4i+HYQvFF/kGHp6ogevD4pVhztbzd
F9JhAbJSeOvZKZFPhzjgX+mCgvzVRniSdDg7wc3+YKNei2zQrmTsiiO6JyhQV2OI
tAqTk572zdZbrCtSgcthrN/uxbJSNnw4X9IZbWtFOUr3lr676II8Q112ttO3IVCe
fF/pZA==
=sPWf
-----END PGP MESSAGE-----

Moving on to Encryption

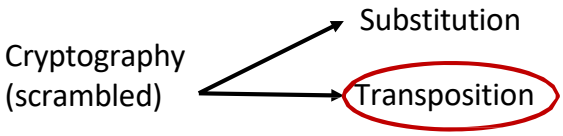
We have a good understanding of hidden ciphers and steganography

Now let's move into our discussion of encryption methods

This page intentionally left blank.

Transposition/Permutation/Obfuscation (1)

- Transposition is simply reordering the letters of a message:
  - Think of a simple anagram—a rearranging of letters
  - For example, take the word "cinema" and rearrange the letters to create "iceman":
    - In crypto, the new version of the letters are not always readable words
  - You *transpose* the order of the letters
  - You *permute* the order of the letters
  - You *obfuscate* the order of the letters



Transposition/Permutation/Obfuscation (1)

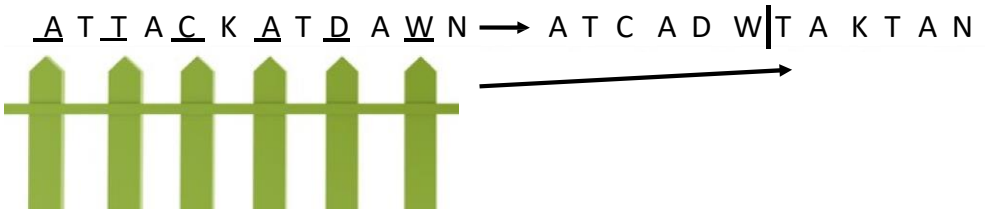
Transposition ciphers do pretty much what the name implies. They transpose the order of letters (or of groups of letters or of whole words in some cases).

If it helps, think of a simple anagram. For example, you might change the order of the letters in the word "cinema" to create "iceman." The difference in cryptographic systems is that the result is not always a readable word, as you see above with iceman. We may use some formula to turn the word "cinema" into "emicna," for example.

Note that you will also see this method of encipherment called *obfuscation* and *permutation*. The terms are interchangeable.

Transposition/Permutation/Obfuscation (2)

- An example is the "Rail Fence" cipher:
  - Every other letter "strikes a rail" and stops
  - Every other letter "falls between the rails," forming two lines of letters
  - The top line goes in front of the bottom line
  - "Attack at dawn" would look like this:

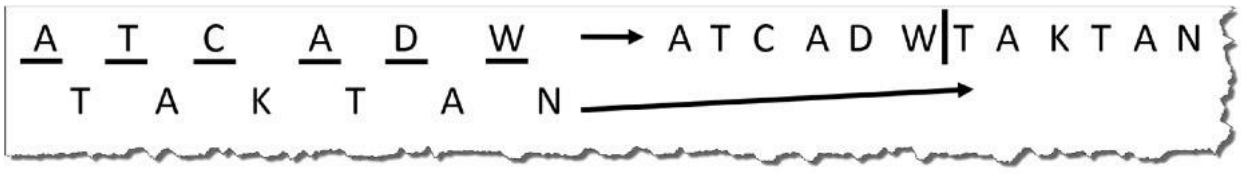


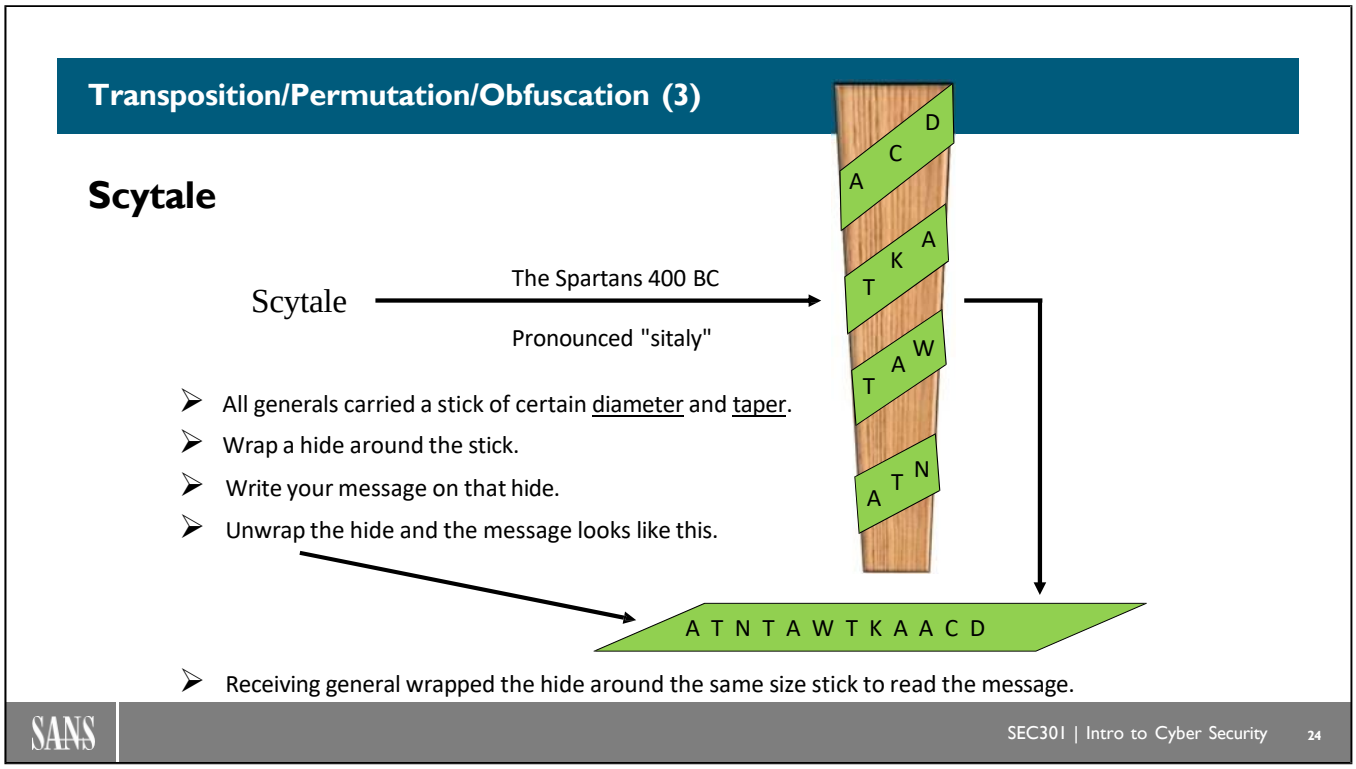
Transposition/Permutation/Obfuscation (2)

One of the best ways to explain transposition or permutation is the "Rail Fence" cipher … a really old example that explains the concept extremely well.

With this cipher, you have a Rail Fence. (You might want to think of a more modern picket fence.) You throw the letters of your message against the fence. Every other letter strikes a rail of the fence and stops. Every other letter passes between the rails and keeps going. You then write the top line, followed by the bottom line. You draw a line between the two strings to show the demarcation.

Given a secret message of "ATTACK AT DAWN," the end result looks something like this:





Transposition/Permutation/Obfuscation (3)

Another well-known transposition or permutation cipher comes from the Spartans around 400 BC. It is called the Scytale (pronounced "sitaly"), which means staff in ancient Greek.

All generals carried a stick of a certain diameter and taper. They would wrap a hide around the stick and write their message down the strips as you see above. (Again, our secret is ATTACK AT DAWN.) When you unwrapped the hide, you had the letters in the order they occur on the strips.

Without the proper diameter staff, you would have a hard time re-aligning the letters to read the message. Of course, part of the security of the Scytale cipher is the fact that only a tiny percentage of the population could read anyway.

Transposition/Permutation/Obfuscation (4)

Column Shift

1	2	3	4
A	C	T	W
T	K		N
T		D	01
A	A	A	10

3	2	4	1
T	C	W	A
	K	N	T
D		01	T
A	A	10	A

Mix up the 1234 in the first box above using some formula. For example, it might become 3241, with the result that "ATTACK AT DAWN" becomes "T DACK AWN0110ATTA" (The 01 and 10 are just padding to fill the block.)

Transposition/Permutation/Obfuscation (4)

Column Shift

The last example of a permutation cipher we look at is the Column Shift cipher. It works exactly as the name implies it does.

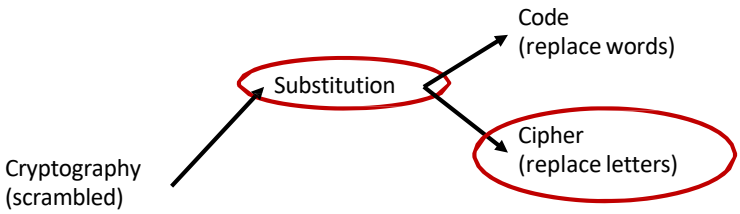
You build a block of text boxes in columns. Again, our secret is ATTACK AT DAWN. When we write that message into the block, moving down the first column and then wrapping to the top of the second column and so on, we wind up with a block that looks like the one to the left in the slide. Notice that our message does not completely fill the block, so we add some padding characters: In this case, 01 and 10.

We then employ some type of formula to rearrange the order of the columns. In this example, on the left, the columns are numbered 1, 2, 3, 4. Some highly complex formula gives us a new order of 3, 2, 4, 1. When we reorder (or shift) the columns into the new order, we get the block on the right.

The end result is that ATTACK AT DAWN is permuted into T DACK AWN0110ATTA. Once again, without knowing the formula to put the columns back in order, it is difficult to arrive back at the original plaintext.

Substitution Ciphers

- Substitution ciphers replace the letters of a message using some formula:
  - Some replace letters with other letters
  - Some replace letters with numbers
- Two broad categories:
  - Monoalphabetic
  - Polyalphabetic



Substitution Ciphers
Where a transposition cipher changes the order of the letters in a message, a substitution cipher replaces the letters using one of several formulas.

A simple example comes from literature. In the book "2001: A Space Odyssey," Arthur C. Clarke came up with the name of the computer in that story by using an anagram of letters. When that book was written, THE computer company was IBM. To come up with the computer named HAL, he simply took the letters before IBM in the alphabet: H before I, A before B, and L before M.

Substitution ciphers generally fall into one of two broad categories:

Monoalphabetic: You replace the plaintext alphabet with one ciphertext alphabet.
Polyalphabetic: You replace the plaintext alphabet with many ciphertext alphabets.

Now, look at examples of each.

Monoalphabetic Ciphers

- Simple substitution Ciphers:
  - Exchange a plaintext alphabet (A B C)
  - For an alternative ciphertext alphabet (E M Z)

  - In the example above, an A in the plaintext becomes an E in ciphertext
  - Plaintext B becomes a ciphertext M
  - C in plaintext becomes Z in ciphertext

  - So the word CAB would encrypt to ZEM in this case
- Examples follow

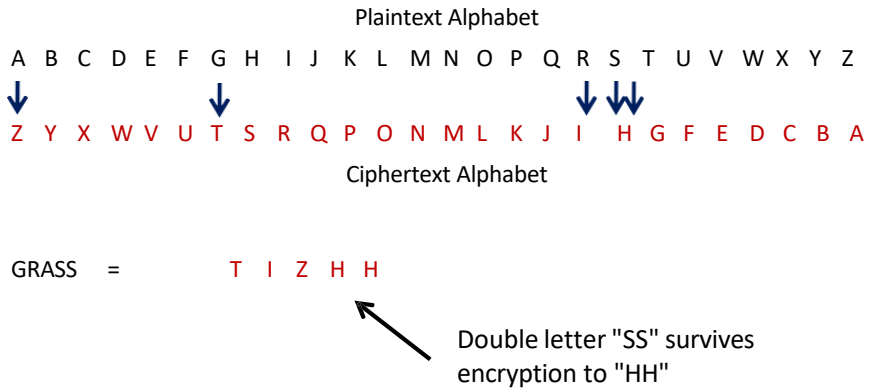
Monoalphabetic Ciphers

A monoalphabetic cipher works by the simple expediency of replacing the plaintext alphabet with a ciphertext alphabet. Any number of methods for creating the ciphertext alphabet are employed. We will look at some of the better-known methods.

In this slide, we show the beginning of the plaintext alphabet ABC. Below it is the beginning of a ciphertext alphabet, in this case, EMZ. Here, any A in our plaintext would become an E in the ciphertext. A plaintext B becomes a ciphertext M. And a C in plaintext is substituted with a Z in the ciphertext.

Monoalphabetic Ciphers—Atbash: 600 to 500 BC

- Simple Substitution cipher using a backward alphabet



Monoalphabetic Ciphers—Atbash: 600 to 500 BC

One of the best known monoalphabetic ciphers is called the Atbash and was used from 600 to 500 BC. To create the ciphertext alphabet, they simply turned the alphabet backward, so instead of beginning ABC, it begins ZYX.

To encrypt a string such as GRASS:

G becomes T

R becomes I
A becomes Z
S becomes H
S becomes H

So, GRASS becomes TIZHH; notice that the double SS survives encryption to HH. This will be important shortly.

Monoalphabetic Ciphers: Caesar Cipher (or C3)

- Another simple substitution cipher, this time shifting the alphabet by three characters

Plaintext Alphabet

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

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

Ciphertext Alphabet

It rotates the alphabet: "Rotational Substitution Cipher"
Also sometimes called a ROT3, as in Rotate 3 positions

Monoalphabetic Ciphers: Caesar Cipher (or C3)
Probably the best known monoalphabetic cipher is the Caesar Cipher, or C3. In this case, they simply shifted (or rotated) the alphabet by three positions to create the ciphertext alphabet. Because they rotate the alphabet, this is known as a Rotational Substitution Cipher.

The ciphertext alphabet now starts with DEF and ends with ABC.

So, to encrypt GRASS:

G becomes
R becomes
A becomes
S becomes
S becomes

So, with a Caesar Cipher, GRASS becomes _____

Would the double SS still survive encryptions? Y / N

To answer the questions on the previous page …

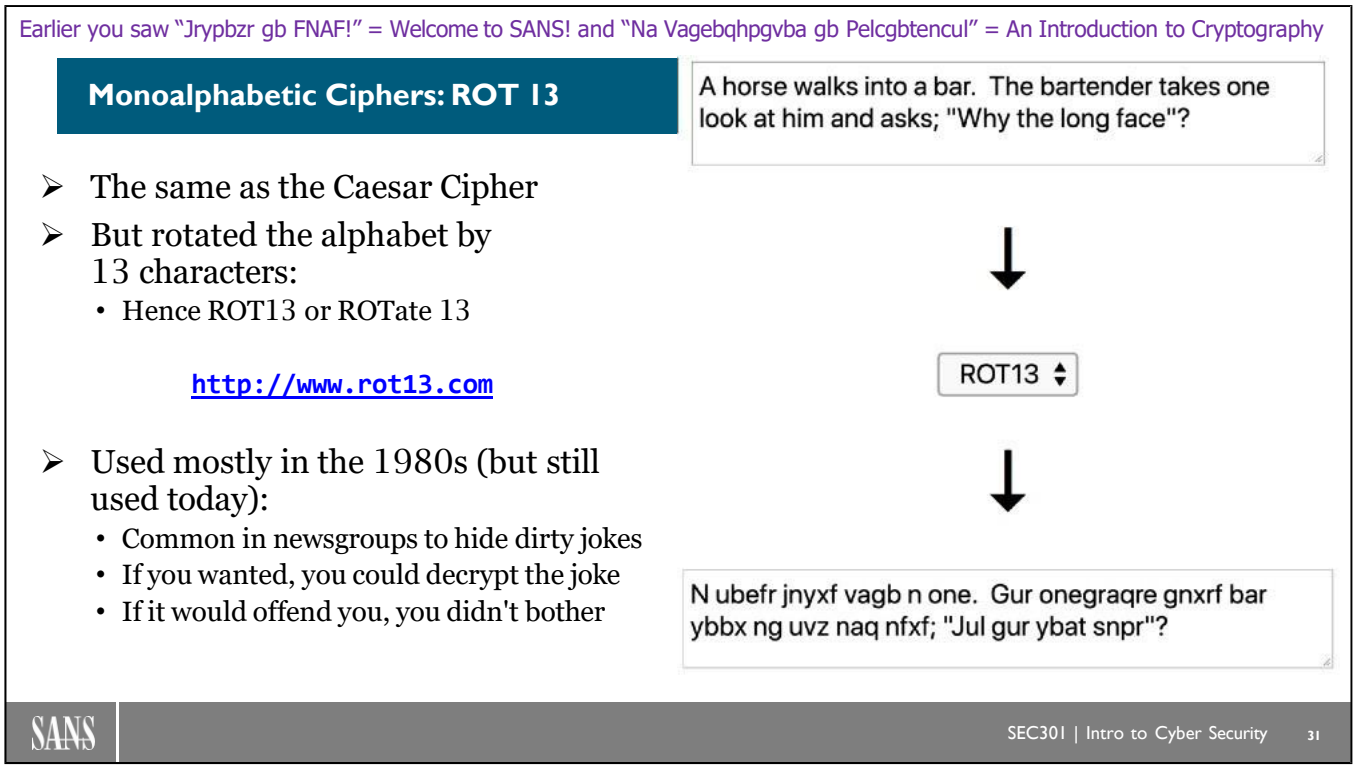
So, to encrypt GRASS:

G becomes J

R becomes U
A becomes D
S becomes V
S becomes V

So, with a Caesar Cipher, GRASS becomes JUDVV.

Would the double SS still survive encryptions? YES



Monoalphabetic Ciphers: ROT 13

There is one more well-known rotational substitution cipher. Where the Caesar Cipher rotates by 3 positions and is referred to as ROT3, the ROT13 cipher rotates the alphabet by 13 positions.

This cipher was most common in the 1980s, but is still used today. By tradition, in newsgroups, if you are going to tell an offensive joke, you will encrypt it with ROT13. Everyone in that community just knows this. So, on the receiving end, if the person wants to see the joke, they know to decrypt with ROT13. If they choose not to be offended, they simply won't bother.

There are several websites that allow you to post plaintext and get the ROT13 encrypted text. You can then post the encrypted text and get the original plaintext back. One such site is:

http://www.rot13.com

Clearly, you would not use a site of that nature for any truly sensitive information, but for jokes, it seems to be okay.

Monoalphabetic Cipher: Attacks

- Very susceptible to Frequency Analysis or Cyphertext Only attacks:
  - In U.S. English:
    - Most common letters E=12.7%, T=9.1%, A=8.2%, Q=0.09%, Z=0.07%, and so on
    - Digraphs: TH=1.52% and UR=0.02%
    - Double letters such as LL=0.577%, SS=0.405% , EE=0.378%,
      - UU and HH are only 0.001%
      - Many letters almost never occur as double letters (KK, JJ, QQ, etc.)
    - These patterns "survive" encryption
  - Simply examine ciphertext for these patterns
- Susceptible to a known plaintext attack:
  - Arabs defeated the Greeks:
    - Every message started with "In the name of God"—Decipher those 18 characters and you discern a great deal of the cipher alphabet

Monoalphabetic Ciphers: Attacks

When used by themselves, monoalphabetic ciphers are highly susceptible to attack, regardless of the method used to create the ciphertext alphabet. In fact, history has several examples of these ciphers being broken.

One such attack is called a **Frequency Analysis** attack, also called a **Cyphertext Only** attack. Meaning the adversary has nothing to work with but your cyphertext. They perform frequency analysis on that cyphertext to attempt to discern patterns.

To perform this attack, you must know the language of the original plaintext. (We will use U.S. English.)

In each language, letters have a frequency of occurrence. For example, in modern U.S. English, the letter E is most common, occurring about 12.7% of the time. T is around 9.1%, A is 8.2%, and this continues to the least common of Q at 0.09% and Z at 0.07%. In addition, there are digraphs, or letters, which often occur together. TH is most common at 1.52%, and UR is least at 0.02%. Of course, there are also frequencies of double letters such as SS, EE, TT (the three most common). And now you realize why we pointed out how double letters survived encryption.

So, to perform a frequency analysis attack, we begin by looking for patterns in your ciphertext. Perhaps we find the letter Z is occurring about 12.7% of the time and there is a high frequency of ZZs and so on. We now believe you are replacing E's with Z's.

Next, we see that Q occurs approximately 9.1% of the time, and that QB happens approximately 1.52%. There is also a high number of QQs. Hmmm—we think you are replacing T's with Q's. And perhaps, given the 1.52%, the QB is replacing TH, so let's look further to see if you are replacing H's with B's.

You get the idea. We continue this frequency analysis, figuring out as much as possible about your ciphertext alphabet. If we also know the frequency of certain one-, two-, and three-letter words, this is also extremely helpful. (And yes, those frequencies are available on the internet.) Eventually, the process of elimination starts kicking in as well. When we know all but eight or ten of the letters, it starts being easier to figure out those remaining.

Eventually, we discern the entire cipher alphabet and can read your mail. Incidentally, because we began this process with nothing but your ciphertext and looked for patterns, this is also an example of a **Ciphertext-Only** attack.

A monoalphabetic cipher is also susceptible to a **Known Plaintext Attack**. A simple example of this comes again from hundreds of years BC. In this example, the Arabs and the Greeks were having a war. The Arabs knew that, by tradition, the Greeks always began every message with the words, "In the name of God." With that knowledge, they simply looked at how those 18 characters were encrypted. This told them enough about the monoalphabetic cipher that they deciphered the entire message. As a result, the Greeks lost the war.

A modern-day example of the known plaintext attack would be if you know every email message a person sends out contains "Microsoft Outlook 2013" from bit X to bit Y of the message. If the messages always end with a signature block and the paragraph that says, "If this is sensitive and you get it by mistake, please disregard … ," all these could be examples of known plaintext. Therefore, the attack is *theoretically* possible today. However, given the extreme amount of randomness that modern ciphers put into ciphertext, it would realistically take millions of years or longer to actually pull the attack off.

Polyalphabetic Ciphers

- Multiple cipher alphabets are used to encrypt ("poly" is Greek for "many")
- Resistant to Frequency Analysis:
  - Each letter of the message is encrypted using a different cipher alphabet
  - Double letters, digraphs, and more do not survive encryption
- Most famous invented by Blaise de Vigenère in the 16th century:
  - Known as the Vigenère cipher
  - Also called (in French) le chiffre indéchiffrable, which translates to "the undecipherable cipher" in English
  - The state of the art for encryption for hundreds of years
  - Example follows

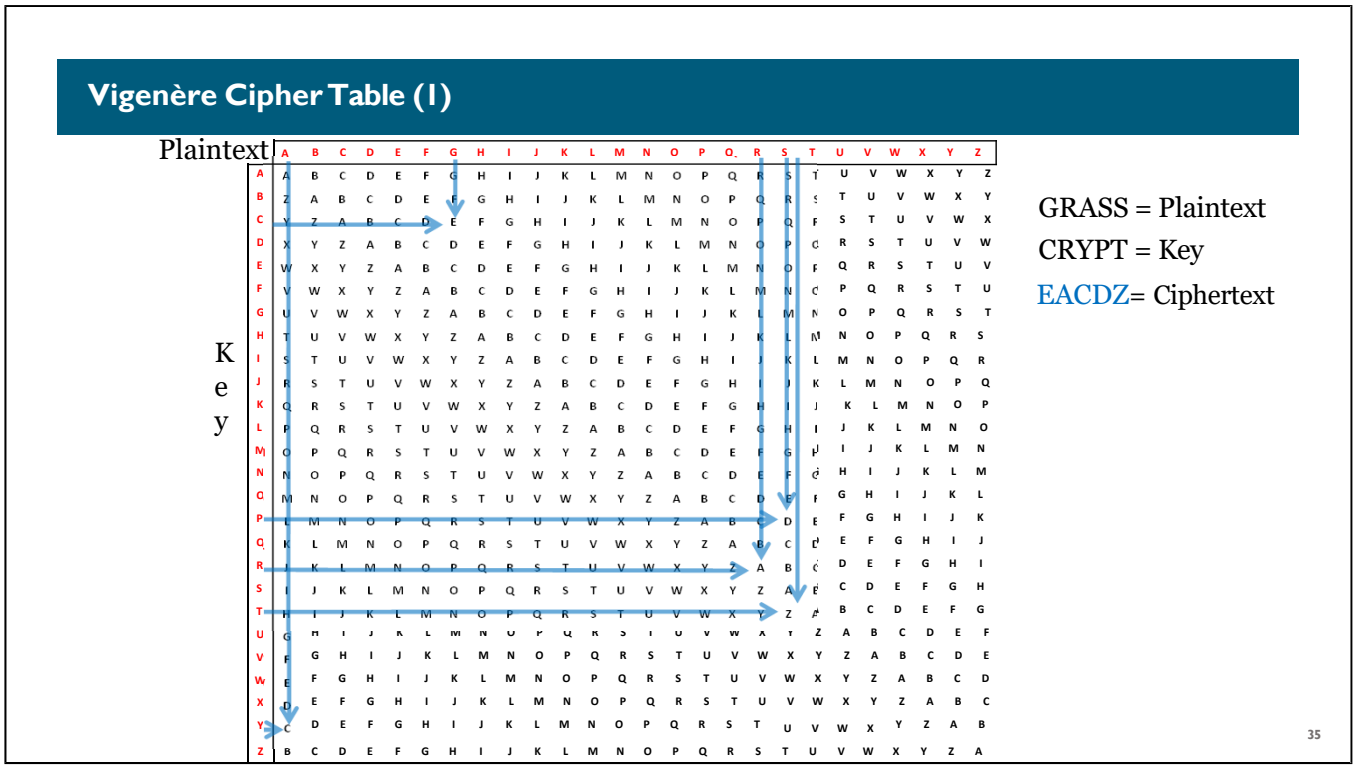
Polyalphabetic Ciphers

The word "mono" is from ancient Greek and means "one." Likewise, the word "poly" is also from ancient Greek and means "many." It is, therefore, straightforward to figure out the difference between a monoalphabetic and polyalphabetic cipher.

Instead of replacing the plaintext alphabet with a single alphabet, polyalphabetic ciphers use multiple cipher alphabets. There is some debate as to who actually invented the notion of polyalphabetic ciphers. But there is little doubt which one is best known. Blaise de Vigenère in the 16th Century invented what became known as the Vigenère Cipher. In fact, it was often called "le chiffre indéchiffrable" in French, which translates to "the undecipherable cipher" in English.

This became the state of the art in "cipher technology" for many years to come. A more complex version is still used by the U.S. military to this day.

Let's look at an example.



Vigenère Cipher Table (1)
The Vigenère Cipher uses a table like the one you see here. (It is also sometimes called a Vigenère Square.) Across the top is the plaintext alphabet. Down the left side is the "key column." Inside the table, you find the alphabet on the first line, the alphabet rotated by one on the second line, rotated by two on the third line, and so on. At the bottom of the table, you find the alphabet rotated by 25 positions.

Note that you will find examples of a Vigenère table "right-rotated" as you see here. (The second line starts with Z, the third with Y.) You will find examples "left-rotated" as well. In a left-rotated table, the second line inside the table begins with B, the third with C, and so on. It does not matter which way you build the table, so long as both parties build it the same way.

To use the Vigenère Cipher to encrypt data, this is the process you follow. Let's say you need to encrypt the word GRASS, and you have chosen a key of CRYPT. You go across the top (plaintext) line to the letter G. Go down the key column to the letter C. Find where those two intersect inside the table. That is your first letter of ciphertext.

Repeat this process for each letter you need to encrypt.

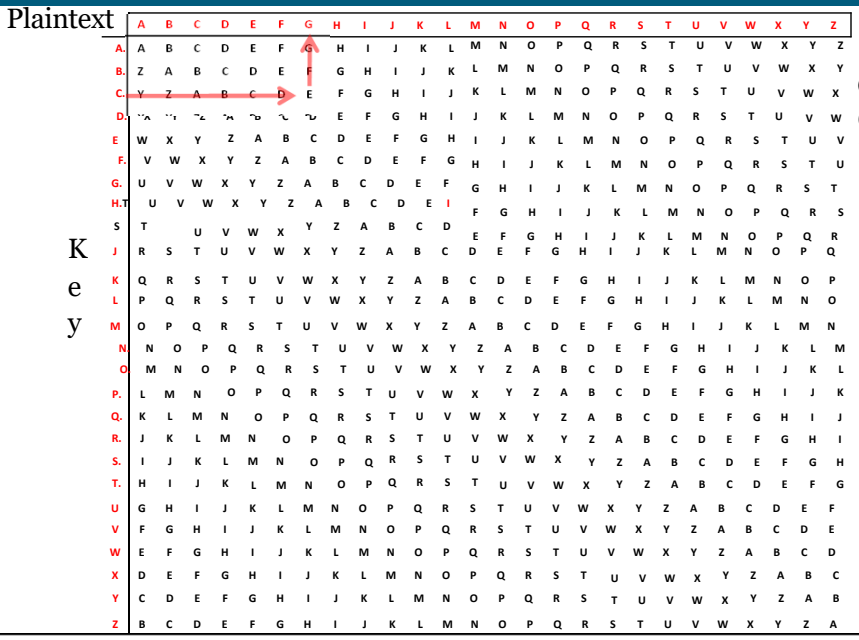
PT	**Key**	**CT**
G	C	E
R	R	A
A	Y	C
S	P	D
S	T	Z

So, given a table built this way and a key of CRYPT, the word GRASS encrypts to EACDZ.

Note: Unlike the monoalphabetic ciphers we looked at, with a polyalphabetic cipher, the double SS does not survive encryption. If it ever does happen that a double letter survives the encryption process (in other words, had SS encrypted to DD), it is pure coincidence and means nothing to an attacker.

We will see more of the Vigenère table when we look at modern ciphers.

Vigenère Cipher Table (2)



GRASS = Plaintext
CRYPT = Key
EACDZ = Ciphertext

To decrypt: Take the first letter of the key (C) above. Move down the left side to that letter. Move over on that line to the first letter of ciphertext. In this example, it is the letter E, the seventh character over. The letter at the top of that line is the plaintext – G in this case.

37

Vigenère Cipher Table (2)

Decrypting ciphertext on a Vigenère table is easy enough, provided you have the key and the ciphertext.

Again, the key is CRYPT. The ciphertext is EACDZ. Find the first letter of the key in the "key column." Follow that line from the key to the letter inside the table that is the ciphertext. When you find the ciphertext letter, follow that line to the top plaintext line. That letter is your plaintext.

Vigenère Cipher

- The Vigenère cipher was originally used with a repeating key:
  - LEMONLEMONLEMONLEMONLEMON
  - This was eventually broken with an advanced frequency analysis attack
- Later used with a "Running Key":
  - Simply means the message and the key are the same length
  - Common to use religious texts: the Koran, the Bible, and more
  - Give someone the ciphertext and tell them the correct passage: The letters of that passage are the key for whatever length the message runs
  - Also, could use a specific page # of a predetermined book
  - See http://www.braingle.com/brainteasers/codes/runningkey.php
  - And http://en.wikipedia.org/wiki/Running_key_cipher

Vigenère Cipher
Initial implementations of the Vigenère Cipher utilized a repeating key. For example, if you chose a key of LEMON, you would just repeat that key (LEMONLEMONLEMONLEMON …) until you had all your text encrypted. It took many years, but, eventually, someone figured out an advanced statistical analysis attack and defeated the repeating key method.

It then became common to use a "running key." This means the key must be the same length as the plaintext—your message contains 1,000 letters, so does your key.

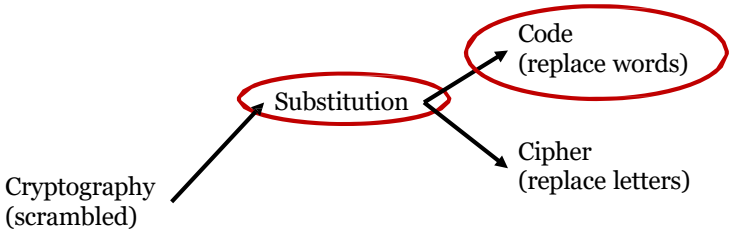
It was a fairly common practice to use religious texts for this purpose. The people wanting to communicate usually used the same religious book (Koran, Bible, and such) and those texts generally contained the same letters. So, for example, you might receive a slip of paper containing what looks like gobbledygook. The person handing you the slip of paper would say something like, "I've always found PSALM 24 to be very enlightening." You now know to turn to that point in your Bible to find the decryption key.

It was also somewhat common for diplomats to carry a special book. They would receive a message that might begin "912 VHDX DADHDFGR …" This told them to turn to page 912 of their special book to find the key for decrypting the message.

These links have good explanations of running key:
http://www.braingle.com/brainteasers/codes/runningkey.php
http://en.wikipedia.org/wiki/Running_key_cipher

Substitution Ciphers: Codes and Code Books

- Where ciphers replace letters, codes replace words
- Code books usually contain the plaintext-to-code in the front, and the matching code-to-plaintext in the back:
  - See the following example



Substitution Ciphers: Codes and Code Books
We now move from replacing individual letters (usually referred to as ciphers) to replacing words (usually referred to as codes). Do note that the terms cipher and code are interchangeable here and you may hear these referred to both ways.

The most common implementation of a Code Book follows:

The front of the book contains the plaintext-to-code. The back of the book contains a matching set of codes-to-plaintext pairs.

An example follows.

Substitution Cipher: Code Book Example

(Encrypt from the front half of the book, arranged by type and then alpha-numeric)					
air support	ZCO	0800	QTD	cover fire	NLG
ground support	BQL	0900	UMW	fire for effect	FRP

Message = Air support / cover fire / 0800

Transmitted via radio in clear
Zulu **C**harlie **O**scar **N**ovember **L**ima **G**olf **Q**uebec **T**ango **D**elta

(Decrypt from back half of the book, letter groups arranged alphabetically)					
BQL	ground support	NLG	cover fire	UMW	0900
FRP	fire for effect	QTD	0800	ZCO	air support

Substitution Cipher: Code Book Example
This is a classic example of a substitution cipher Code Book. This method of secreting information has been used by militaries around the world for many years (there are examples dating to the U.S. Revolutionary War).

Important point: For the communication method described here to work, both parties ***must*** have an identical copy of the Code Book. Each book is typically good for a limited amount of time. When the book expires, both parties destroy that version and move to the next version. This has to be coordinated very carefully.

The front of the book contains the plaintext-to-code pairs. It might look like:

(Encrypt from the front half of the book – arranged by type then alpha-numeric)					
air support	ZCO	0800	QTD	cover fire	NLG
ground support	BQL	0900	UMW	fire for effect	FRP

Let's say we need to pass the message that we want "air support" for "cover fire" at "0800" hours.

Air support = ZCO
Cover fire = NLG
0800 = QTD

So, the encoded message is **ZCO NLG QTD**. Over the (unencrypted) radio, you would pass the encoded message phonetically by saying:

Zulu **C**harlie **O**scar **N**ovember **L**ima **G**olf **Q**uebec **T**ango **D**elta

The person on the other end writes down the letters of the encoded message. They turn to the back of their exact copy of the Code Book and see:

(Decrypt from back half of the book – letter groups arranged alphabetically)					
BQL	ground support	NLG	cover fire	UMW	0900
FRP	fire for effect	QTD	0800	ZCO	air support

They find the "letter groups" arranged in alphabetical order and can easily look them up. They see that ZCO means the other end needs "air support," NLG means they need "cover fire," and QTD means they need it at "0800" hours.

This is not the simplest way of passing messages. However, if all automated means are down or not working for some reason, it does work.

With a One-time Pad, the "Pad" is also the encryption key

Substitution Cipher: One-time Pad or OTP

- Gilbert Vernam in 1917, also called Vernam Cipher
- The only unbreakable cipher: IF implemented correctly:
  - Used once and never again
  - Pad/Key must be as long as the message (running key)
  - Pad/Key must be protected at all times by both parties
  - Pad/Key must be as random as possible
- Before an OTP example, we need a quick introduction to XOR

Substitution Cipher: One-time Pad or OTP

In 1917, Gilbert Vernam invented the Vernam Cipher, better known as the One-time Pad. In many security books, you will read that this is "the only unbreakable cipher" or "the only provably secure cipher." Although this can be true, it remains so only if several facts remain true as well. (The books rarely point these out.)

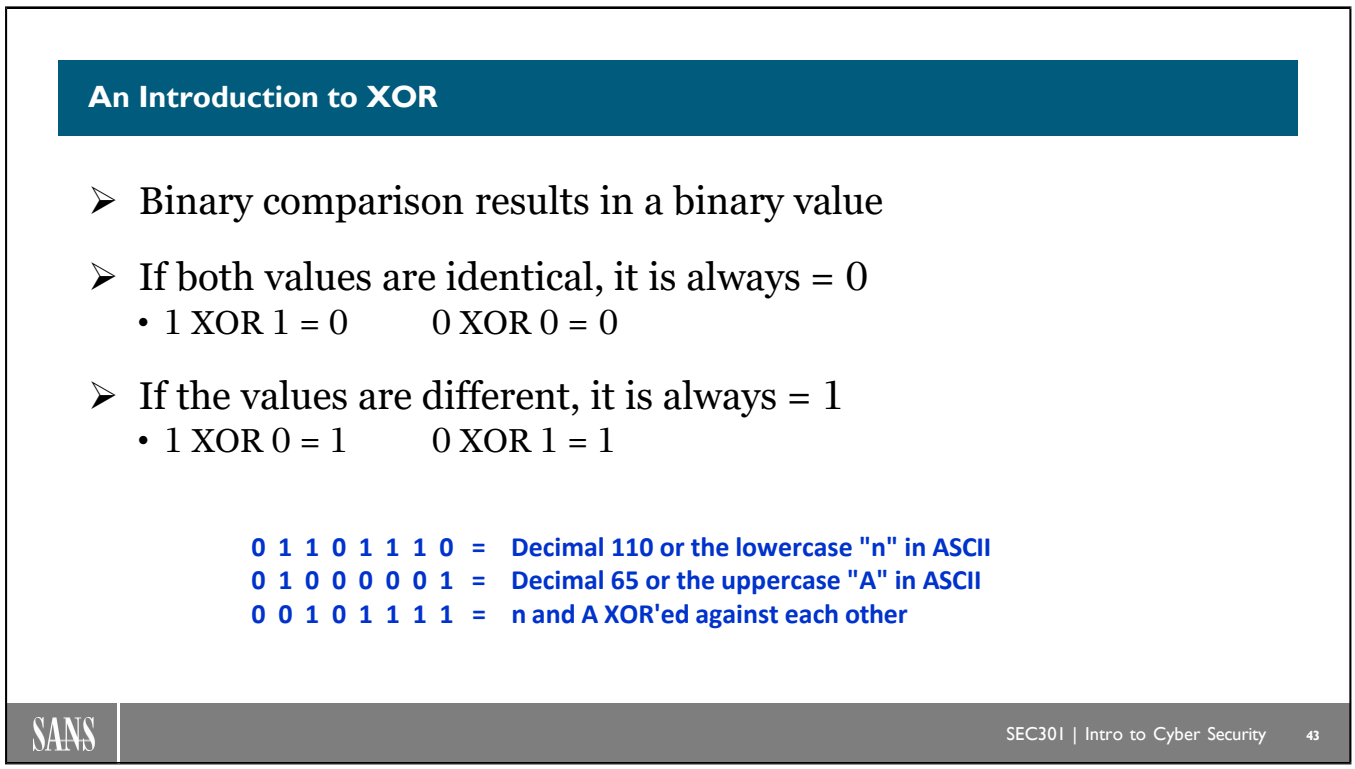
For a One-time Pad to remain secure:

- You use it once and ***never*** again. Even just twice and it can be broken.
- The pad must be as long as the message (a running key).
- Both parties must protect the pad at all times. If anyone else has it, they can read your message.
- The pad must be as random as possible. Obtaining good randomness is not as simple as many might imagine.

So long as ***each*** of those points remains true, then this cipher can indeed be highly secure.

Before we look at an example, we have to understand an operation called XOR, because this cipher makes use of that function. (Some denigrate XOR as "the poor man's cipher" … but every modern crypto algorithm makes heavy use of XOR.)

First, we need to make sure you understand a few of the fundamental workings of a computer.



An Introduction to XOR

XOR performs a simple comparison of two binary bits and produces a result based on the comparison. If the bits are identical (two 1s or two 0s), the result is always a 0. If the bits are different (a 0 & 1 or 1 & 0), the result is always a 1.

To understand why this would matter, you need to understand a fundamental premise of how a computer operates. In reality, a computer can only do *one thing*. It can add. It does addition extremely well and remarkably fast, but that is the only thing it knows how to do.

Let's say, for example, that the computer you are using is putting characters on the screen. The computer sees only strings of binary 1s and 0s, and it knows that each of them has a place value. It adds those place values together to come to a decimal number. It then looks in an ASCII chart for that number to determine which letter that number represents.

From here, you have two choices. You can simply accept our word that the following is true. Or you can skip down to the more technical explanation. (That explanation requires that you understand binary well.)

Option 1: Two characters and an XOR result.

01101110 is a decimal 110 and the computer knows that is a lowercase "n."
01000001 is a decimal 65 and the computer knows that is the uppercase "A."
00101111 is the XOR result (two identical results in 0, two different results in 1).

Option 2: The Technical Explanation

You need to remember binary math and the place values to understand this explanation. Remember that the place values of binary are 1, 2, 4, 8, 16, 32, 64, and 128. So, the binary byte:

128 64 32 16 8 4 2 1
1 0 1 0 1 0 1 0 = 128 + 0 + 32 + 0 + 8 + 0 + 2 + 0 = 170 in decimal

Let's say that a computer needs to place a lowercase "n" on your screen. It sees the binary string 01101110. If you add the place values for the 1s together that is:

0 + 64 + 32 + 0 + 8 + 4 + 2 + 0 = 110

When you look decimal 110 up in an ASCII chart, you will find that means lowercase "n."

Now if the computer needs to display "N," it would see binary 01001110, which equates to 0 + 64 + 0 + 0 + 8 + 4 + 2 + 0 = 78. In an ASCII chart, that means an uppercase "N." Now we can take that knowledge into an explanation of XOR.

XOR is simply a process by which you compare two binary bits and arrive at a third bit.

If the bits you're comparing are identical (two 1s or two 0s), you always get a 0 result.
If the bits are different (a 1 and a 0, or a 0 and a 1), you always get a 1 result.

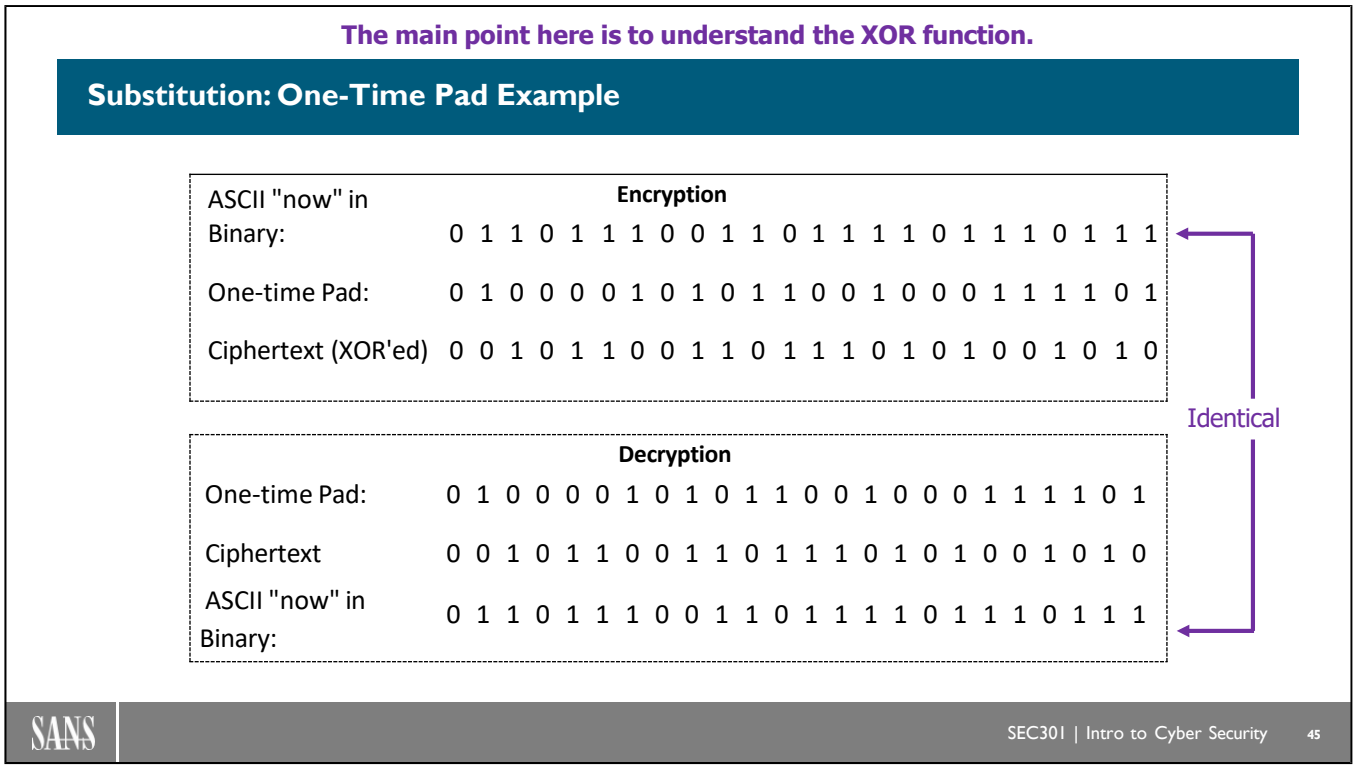
To help understand the process, let's return to our preceding examples of lowercase "n" and uppercase "N" in binary. Here is how those letters would XOR against each other. Remember the two simple rules above:

n = 0 1 1 0 1 1 1 0
N = 0 1 0 0 1 1 1 0
XOR = 0 0 1 0 0 0 0 0

If we perform the exact same operation with an uppercase N and an uppercase X (decimal 88, or 01011000), it looks like this.

N = 0 1 0 0 1 1 1 0
X = 0 1 0 1 1 0 0 0
XOR = 0 0 0 1 0 1 1 0

Cryptographic algorithms often make use of this type of "bit switching" as *one of* their functions to increase randomness and so on. It happens the One-time Pad uses this function exclusively.



Substitution: One-time Pad Example
Now to explain the One-time Pad:

First, as with the Code Book, both communicating parties must have an identical copy of the pad (the pad amounts to the cryptographic key). As stated earlier, the pad must be random, protected, used only once, and at least as long as the message. (If it is longer, you don't utilize the leftover bits of the pad.)

Here, you can see the word "now" in binary on the top line in the box labeled Encryption. The second line is the One-time Pad. Perform the XOR operation and you arrive at the third line (the ciphertext).

You send that line to the person you are communicating with. As stated, they already have a copy of the pad.

Moving to the bottom box labeled Decryption, the recipient XOR's the ciphertext against the pad and obtains the original plaintext.

Lab Time

- Lab 3.1: Crypto by Hand
- Objectives:
  - Encrypt using permutation via the Rail Fence Cipher
  - Encrypt using substitution via the Caesar Cipher
  - Decrypt using substitution via the ROT13 Cipher
  - Perform triple encrypting via Vigenère Cipher:
    - Triple encryption, such as Triple DES, is an upcoming topic
  - Estimated completion time: 30 minutes

An enlarged Vigenère Cipher Table is below this slide in your notes – it will help at the end of this lab!

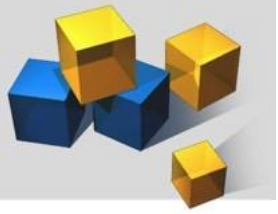
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
C	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
D	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
E	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L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

Module 9: Building Blocks of Modern Crypto

- Strength of Cryptosystems and the Work Factor
- Message Digests/One-Way Hashes
- Symmetric and Asymmetric Keys
- Generic Cryptographic Process Examples
- Digital Certificates & PKI

COURSE ROADMAP

- Module 8: Intro to Crypto
  - Lab 3.1: Crypto by Hand
- **Module 9: Building Blocks of Modern Crypto**
  - Lab 3.2: Visual Crypto
- Module 10: Data Encrypting Protocols



Module 9: Building Blocks of Modern Crypto

This page intentionally left blank.

With Brute Force – You usually guess the right key about 50% of the way through all possible guesses.

Strength of Cryptosystems: Keyspace Matters

- Long key = larger keyspace = more guesses to brute force:
  - Each additional bit doubles the number of guesses to brute force the key
  - Brute force attack will always work—*Eventually*
    - it is a question of how long it will take

Brute force is like finding a needle in a haystack…

Or more precisely, a specific needle in a needle stack!

SANS SEC301 | Intro to Cyber Security 48

Strength of Cryptosystems: Keyspace Matters

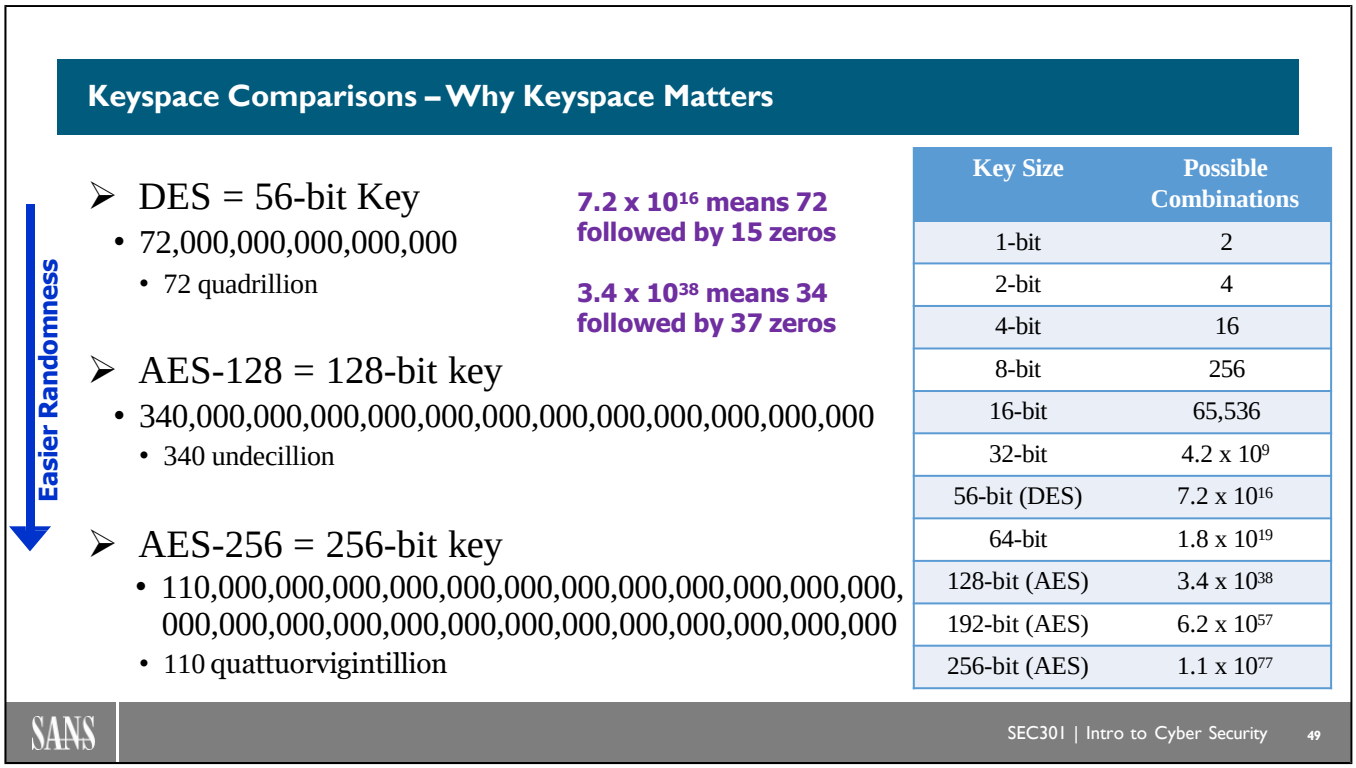
In cryptography, the term Brute Force Attack simply means that you guess every possible combination of a key. Eventually, one of them will work. This is the one attack against crypto that is ***always*** guaranteed to work. It is a question of how long it will take. (Brute force is covered in greater detail in a later module.)

When you look at the strength of a cryptographic system, one of the first things you want to know is the length of the key. A longer key means more possible combinations, meaning more (many more) guesses to defeat the system via a brute force attack.

For every binary bit that you add to the length of a key, the number of possible combinations (and therefore the number of necessary guesses) doubles. Consider the case of DES, for example. That algorithm uses a 56-bit key. That equates to roughly 72 quadrillion possible keys (or combinations or guesses). If you add 1 bit to make it 57 bits, that number doubles. Add another 1 to make it 58 bits and the number doubles again, and so on.

So, if you look at the number of guesses necessary against a 56-bit key versus the number necessary against a 128-bit key, the number is vastly greater. If you go to the 256-bit key, the number of combinations is truly ginormous.

What do these numbers look like? Turn to the next page for some examples…



Keyspace Comparisons – Why Keyspace Matters

Earlier, we defined Keyspace as "the range of values that can be used to construct a key". We gave one of the examples you see above. The DES encryption algorithm uses a 56-bit key. This means its keyspace is 7.2 x 10^{16} (72 followed by 15 zeros) or 72,000,000,000,000,000, called 72 quadrillion.

Another common algorithm we discuss quite a bit is the Advanced Encryption Standard or AES. It has three different key lengths you can choose from: 128-bit, 192-bit, and 256-bit. Most implementations use either 128-bit or 256-bit.

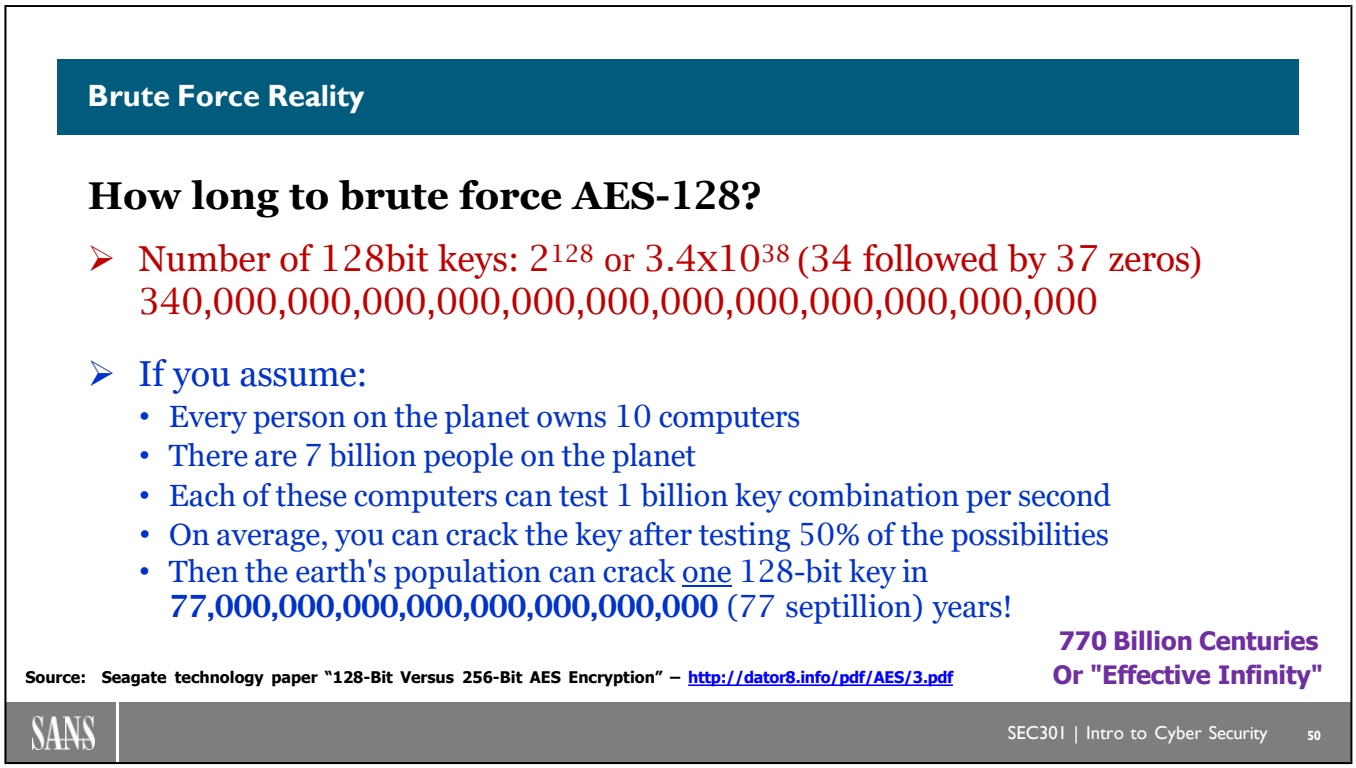
Here we see the keyspace sizes for those two key lengths:

AES with 128-bit key is 3.4 x 10^{38} or 340,000,000,000,000,000,000,000,000,000,000,000,000, or 340 undecillion.

AES with 256-bit key has a keyspace of 1.1 x 10^{77} or 110,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000, or 110 quattuorvigintillion. I found that word on the internet, so it has to be true! ☺

But what do these massive numbers mean and why do we care? Let's go to the next page to find out.

Reference: http://www.thealmightyguru.com/Pointless/BigNumbers.html



Brute Force Reality

There is an outstanding explanation of the time required to brute force a key of a given length in a Seagate technology paper on Full-Disk Encryption titled "128-Bit Versus 256-Bit AES Encryption." The paper is available at the link below.

As you can see in the slide above, even if every person on the planet had 10 computers and each of those 70 billion computers made a billion guesses per second, it would still take 77 septillion years to brute force a single 128-bit key. To brute force, a second key would require another 77 septillion years, and so on.

77 septillion years is equivalent to 770 billion centuries. Or to put it another way, scientists tell us that the universe is 13.8 billion years old. So 77 septillion is more than 5 quadrillion times the age of the universe.

Now apply those same assumptions to AES-256-bit key…

Reference:

http://dator8.info/pdf/AES/3.pdf

The point of the paper found at this link is this: Why should they use 256-bit key, since 128-bit encryption provides for protection to the point of effective infinity? Why would you ever need something like 512-bit encryption? In a little while we will discuss some of the things you have to keep in mind for encryption over long periods of time. For now, our common key lengths are from 128 bit to 256 bit and are completely sufficient.

Strength of Cryptosystems: Randomness Is Critical

- Everybody focuses on keyspace to the exclusion of other factors:
  - It is important but NOT the whole story
- You must have highly Random key:
  - Computers cannot generate a truly random value
  - They use a Pseudo-Random Number Generator
    - **PRNG = Pseudo-Random Number Generator**
  - Predictability leads to a "Weak Keys Attack"
- More Randomness is critical!

Strength of Cryptosystems: Randomness Is Critical
In addition to being long, the key must be as random as possible. However, computers cannot generate a truly random value. They can come close, but it is not truly random. Therefore, computers utilize a Pseudo-Random Number Generator (PRNG) to generate a "random enough" value for a cryptographic key.

NOTE: You will see the abbreviation PRNG throughout our upcoming discussions.

If the key is not random enough—if it is predictable—this leads to an attack known as a *weak key attack*. The attacker examines the formula used by your PRNG and determines the pattern. When the pattern is known, the next key can be predicted.

As much randomness as possible in generating crypto keys is critical!

Following are examples.

Strength of Cryptosystems: Birthday Attack (1)

- Brute Force and the Birthday Attack:
  - This is normally discussed with hashes
  - But can also speed a brute force attack:
    - Brute Force = Simply guess every possible combination; eventually one will work
  - What if … we could find a way to predict the most likely keys a PRNG will generate?
  - Then, guess the most likely first; this will speed the brute force attack

Strength of Cryptosystems: Birthday Attack (1)

There is an interesting attack in cryptography known as the Birthday Attack. Let's take a look at how this attack works because it is a randomness issue that leads to the attack.

> NOTE: Most discussions of the Birthday Attack discuss it with cryptographic hashes, and it does apply there as well. In this case, we will explain how it can be used to accelerate a brute force attack.

An important question to ask when looking at attacks is, "What if … ?"

So, what if we could find a way to predict the most likely keys a PRNG will generate? If we can do that, and then guess those combinations first, this would clearly speed up the brute force attack.

But how can you accomplish this?

Strength of Cryptosystems: Birthday Attack (2)

- Birthday Probability Theory:
  - Take 23 people born in the United States:
    - 50% probability: Two will have the same birthday
    - With 47 people, it becomes a virtual certainty
  - Most common birth dates are late September:
    - 9 months following the holidays (Christmas in particular)
  - In other words, certain birth dates are more likely to occur than others

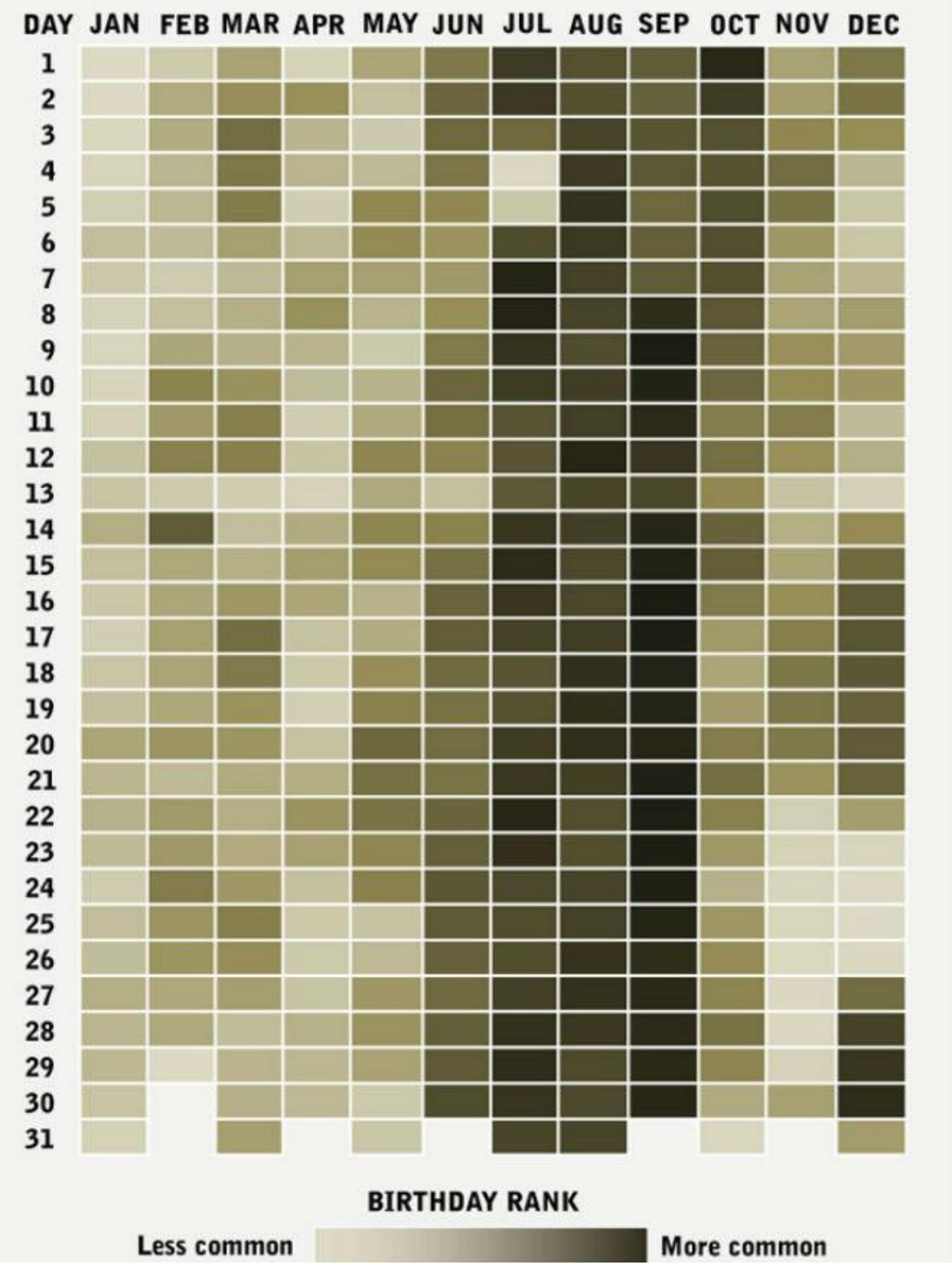
Strength of Cryptosystems: Birthday Attack (2)

We need to begin by understanding the Birthday Probability Theory, the gist of which states that if you take 23 people born in the United States, there is a 50% probability that two of them will have the same birth date (month and day). If you raise the number to 47 people, it becomes a virtual certainty that two of them will share a birth date.

There is a great deal of math that goes into this theory (also called the Birthday Paradox). But beyond the math, there is some simple fact. The fact is that the most common birthdays in the United States are in late September. (September 16 to October 1 are especially common.) It just so happens that these dates are 9 months after the holidays, Christmas in particular.

But the point is that there are certain days on the calendar that are more likely to occur as birth dates than others. We can apply that to our PRNG in cryptography.

This chart shows birth date commonality. It was created by Dan Lin of the CDC (Centers for Disease Control and Prevention). The darker the shading, the more common the birth date.



http://www.todaysparent.com/wp-content/uploads/2014/09/heatmapbirthdays1.jpg

Strength of Cryptosystems: Birthday Attack (3)

- Apply that theory to a PRNG:
  - It is *Pseudo* random, not truly random
  - It will not generate a key of:
    - All zeros (00000000000000000000)
    - All ones (11111111111111111111)
    - Patterns (10101010101010101010)
    - And so on
  - Meaning certain keys are less likely to occur
  - Automatically meaning *other* keys are more likely to occur:
    - Just like birth dates on a calendar
  - Look at a PRNG: Figure out the keys it is most likely to generate
  - Guess those combinations first
  - Accelerates the brute force attack

Strength of Cryptosystems: Birthday Attack (3)

So now let's apply the concept behind the Birthday Probability Theory to a PRNG in cryptography.

A Pseudo-Random Number Generator will not generate a 40-bit key in a pattern like the following:

0000000000000000000000000000000000000000
1111111111111111111111111111111111111111
0101010101010101010101010101010101010101

In fact, a good PRNG is specifically designed to avoid patterns. That is why they are *Pseudo Random*.

A true random number generator generates those types of patterns. Just like in the lottery, numbers 1, 2, 3, 4, and 5 are just as likely to win as any other combination. (But who plays those?) That is true randomness.

So, if we apply the Birthday Paradox concept to a PRNG and figure out which keys it is less likely to generate, the combinations that are left are automatically more likely to be created. If we can apply this concept to determine the keys most likely to occur and guess those first in our brute force attack, we can accelerate the attack a great deal.

Find examples of Crypto Snake Oil with this Google search: site:www.schneier.com "The Doghouse:"

Strength of Cryptosystems: Strong Math & Peer Reviewed Algorithms

- Strong Algorithm: Only good peer review of algorithms
  - Weak math results in a mathematical attack
  - If mathematical attack is successful, the system is truly broken
    - You can decrypt without possessing the key

- No "Snake Oil" allowed: No proprietary algorithms
  - A substance with no real medicinal value sold as a remedy for all diseases
  - A product, policy claim, or idea of little real worth or value that is promoted as the solution to a problem

- We need Peer Reviewed Algorithms
  - DES has been peer-reviewed since 1975
  - AES has been peer-reviewed since 1998

We also want a good "Avalanche Effect"

Small change in plaintext = huge change in ciphertext

Strength of Cryptosystems: Strong Math & Peer Reviewed Algorithms

Another critical aspect in determining the strength of a cryptosystem is the algorithm. In modern cryptography, the algorithm is a highly complex mathematical formula. If there is a flaw in that math, this leads to a "mathematical attack." The result is that you can decrypt data without the key.

Be aware of "snake oil." In other words, if a vendor tells you they have a "proprietary encryption algorithm," do not buy the product. In every case we know of, when that proprietary algorithm is released to the public for peer review, it is broken quickly, usually within approximately two weeks. You can find many examples of cryptography snake oil by searching for site:www.schneier.com "The Doghouse:". In his Cryptogram Newsletter, Bruce Schneier calls them out frequently.

By contrast, the DES algorithm has been public domain since March 17, 1975. Every mathematician in the world has tried to find a flaw in its math and none have (so far). That is good peer review and gives us a high degree of confidence in that algorithm. (Yes, the key can be brute forced just like any cryptosystem, but the math appears solid.)

Incidentally, the AES algorithm we will also discuss later has been in the public domain since August 20, 1998. So far, no mathematician has reported any flaws in its math either.

A good crypto algorithm should also provide a strong Avalanche Effect, meaning that any tiny change in the plaintext input (even a single binary bit) will result in a completely different ciphertext output. This indicates strong randomness functions on the part of the algorithm. It also prevents an attack where the attacker knows the contents of one message and can easily discern the contents of similar messages.

The Work Factor (1)

- If you encrypt something and anyone in the world can decrypt it, then everyone in the world can decrypt it!
- It is not a question of can they, but of how quickly can they?
  - If they cannot read your email for 20,000 years, will you still care?
  - 20,000 years is not a large work factor
  - In an earlier slide, we saw a 77 septillion year work factor

The Work Factor (1)

All this discussion of key lengths, randomness, strength of algorithms, and so on has lead us to the discussion of Work Factor. If you truly want to understand cryptography, you need to understand this concept.

If you encrypt something and anyone in the world can decrypt and read it, then ***everyone*** in the world can decrypt and read it. It is not a question of can they, but of how quickly can they?

If you encrypt your email today and attackers can't decrypt and read it for 20,000 years, will you still care?

And by the way, we are not using ridiculous numbers here. You have already seen that brute forcing modern cryptography would take "effective infinity".

1 GFLOP = 1 GigaFLOP or 1 Billion Floating Point Operations Per Second

The Work Factor (2)

- To calculate: Include all factors we discussed (key length, randomness, algorithm strength, Avalanche Effect):
  - Calculate how quickly processing power increases
  - Moore's Law: Processing speed doubles roughly every 2 years
    - DES work factor = 20K years in 1975, now brute forced in hours

iPhone 5, 2012
76.8 GFLOPs

iPhone 6, 2014
115.2 GFLOPs

iPhone 7, 2016
499.2 GFLOPS

iPhone X, 2017
600.0 GFLOPS

Cray2 mainframe
1985
1.9 GFLOPs
($16M then or $32M in 2010 dollars)

SANS SEC301 | Intro to Cyber Security 59

The Work Factor (2)

When calculating the work factor, you take into account everything we have talked about to this point. The key length, the randomness, the algorithm strength, the Avalanche Effect, and so on. But again, this is not the whole story.

You also have to take into account Moore's law. This principle is named for Gordon E. Moore, co-founder of Intel Corporation. In a 1965 paper, he explained that the number of transistors in a dense integrated circuit doubles approximately every two years. The upshot of Moore's law is that the speed of computers doubles approximately every two years.

Think of it this way. In computer speak, 1 GFLOP is 1 Billion Floating Point Operations per Second, a standard measurement of computing speed. When the Cray 2 mainframe was released in 1985, it could perform at a speed of 1.9 GFLOPS. By contrast, the graphics processor of the iPhone 5 released in 2012 performs at 76.8 GFLOPS. The iPhone 8 (and iPhone 10) that came out in 2017 does 600.0 GFLOPS.

Now think about what that speed difference means when someone is brute forcing passwords or crypto keys. Every two years, the number of guesses they can make per second doubles. It means that the DES algorithm's 56-bit key that would take more than 20,000 years to brute force in 1975 can now be brute forced in hours.

Note: These processing speeds are according to the website https://gflops.surge.sh/

The Work Factor (3)

- An "infinite work factor" does not exist:
  - Consider how long it must be kept a secret:
    - "Attack at dawn" needs only 24 hours protection
    - Other secrets need to be protected much longer
  - Long work factors are hard to achieve:
    - 100 years of protection required
    - Encrypt with the best there is today (AES 256)
    - 20 years from now, decrypt with the old and re-encrypt with the new
    - 20 years later, do that again … and so on
  - AND: <u>At no time can you transmit the data</u>, or you lose protection before the 100-year point

The Work Factor (3)

Here is the bottom line. There is no such thing as an Infinite Work Factor. You need to determine an Appropriate Work Factor.

The classic example: The secret "Attack at dawn" needs to be protected for only approximately 24 hours. After you launch the attack, the other side kind of knows what you were planning. It is no longer a secret and therefore no longer needs to be protected.

Extremely long work factors are hard to achieve, in large part because of the speed at which computing power is increasing. Think about a piece of data that you *must* protect for 100 years. Here is what you would have to do.

Encrypt it today with the best crypto available, probably AES 256-bit key. Then 20 or 25 years from now, AES will be considered weak and something new and better will be out. Decrypt with the old; re-encrypt with the new. Twenty years later, the cycle repeats, and you decrypt with the old and re-encrypt with the new. Continue this process for the next 100 years (protecting the keys as you go, of course).

AND … at no time can you transmit that data. If, for example, you transmit it today in the AES 256-bit encrypted form and someone captures it in that form, then 25 years from now, they will be able to break it back to plaintext. You did not obtain your 100 years of protection.

What does this mean to you? (We will use a U.S.-specific example, but similar cases would apply in other countries.)

Have you ever transmitted your Social Security number (or other government identification code for yourself) across the internet in an encrypted form? (If you have ever filed your taxes electronically, then the answer to that question is "yes, you have.")

If so, then sometime during your long lifetime, your SSN will be subject to compromise because an appropriate work factor was not obtained.

Message Digest: One-Way Hash

- Proper name = Message Digest: One-way hash is more common slang
  - Not encryption, but utilized by cryptosystems
  - Mathematical algorithm runs against 1s and 0s of a file
  - The file is not modified in *any* way
  - Generates a fixed length hash from that file

- The hash is always the same length regardless of the file size:
  - That is, MD5 = 128-bit hash – SHA-1 = 160-bit hash – SHA-256 = 256-bit
- Provides no insight as to the input
- Must have a good Avalanche Effect

Message Digest: One-Way Hash

Continuing with our discussion of the building blocks of modern crypto, we are now going to discuss three different types of algorithms. Specifically, we will talk about one-way hashes, symmetric algorithms, and asymmetric algorithms.

First, consider the one-way hash, which is actually properly called a Message Digest. However, if you use the second term, most people will not know what you are talking about. We will call it a one-way hash, or just "hash" for short.

Hash functions are not actually encryption. There is no crypto key, for example. Cryptosystems utilize hash functions, as you will see.

Hashing software uses a mathematical algorithm that runs against the 1s and 0s of a file. As it runs, the input file is not modified in *any* way.

The operation results in a fixed-length string, commonly referred to as a hash. It is important to note that the output hash is *always* the exact same length from a given algorithm, regardless of the size of the input. For example, the MD5 algorithm will always generate 128 bits, no matter how large or small the input is.

It is also good to know that the hash provides no insight whatsoever as to what the input looked like or how large it might have been.

For example, you could go to the link: http://www.xorbin.com/tools/md5-hash-calculator

Paste the title of this slide into that page "Message Digest: One-Way Hash" and click Calculate. The output is:

4df98231549beb91208800b53449d0f1

If we change the capital D to a lowercase d, the output looks like this:

3297dc5d579bec4efa50ada046d3dc32

Notice you can tell at a glance they are completely different (an Avalanche Effect).

Furthermore, if we use the xorbin.com site to hash **all** the text on the previous page, the resulting hash is:

Adacdd931c512f0624066e34faf56c51

Still 128 bits, even though the input is significantly longer.

One-Way Hash Example

Nothing embedded

```
$ md5 IMG_1132.JPG
MD5 (IMG_1132.JPG) = 82d6748bc011b537984b043e3515d7b3
$
```

Note: This tells us ***that*** the images are different,
but not ***how*** the images are different.

With file embedded

```
$ md5 embed_IMG_1132.JPG
MD5 (embed_IMG_1132.JPG) = 54b2e36b3ed876fe8220e8be1191a51f
$
```

SANS | SEC301 | Intro to Cyber Security 64

One-Way Hash Example

If we return to the steganography example from earlier, we see two pictures. The top picture has nothing embedded in it. The bottom picture has a file steganographically hidden.

At a command prompt in the Mac OS, you can simply type the command md5 <filename> to generate an MD5 hash of that file. That is what we have done here with the two images. Also note that the hash for each file is radically different.

```
$ md5 IMG_1132.JPG
MD5 (IMG_1132.JPG) = 82d6748bc011b537984b043e3515d7b3
$
```

```
$ md5 embed_IMG_1132.JPG
MD5 (embed_IMG_1132.JPG) = 54b2e36b3ed876fe8220e8be1191a51f
$
```

But notice, the hash tells us *that* the pictures are different, not *how* they are different. Any minor change to the bottom picture would result in a different hash. In this case, the difference happens to be that the second image has a file embedded in it. But the hash does not actually tell us that. If someone edited even a single binary bit of a single pixel, it would also generate a different hash.

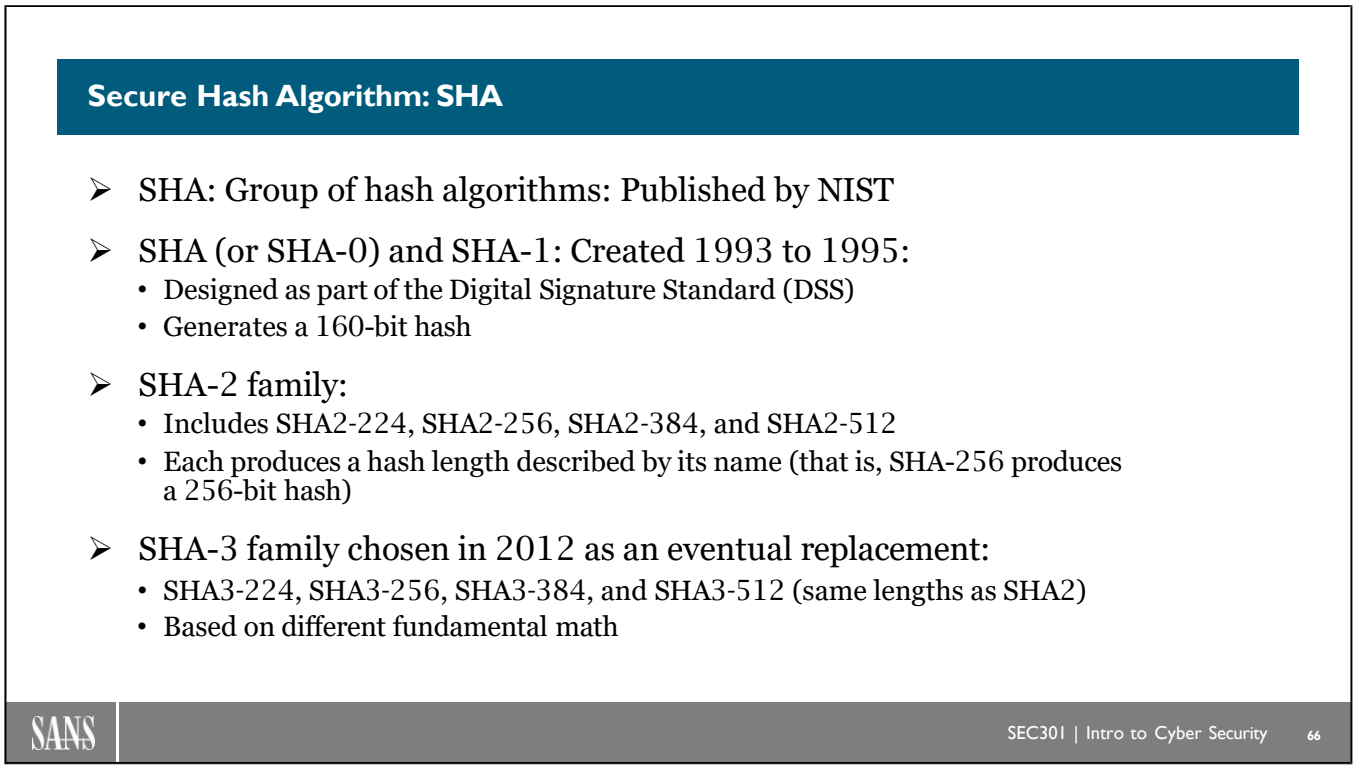
Message Digest Version x: MDx

- The "MD Family" of hash algorithms created by Ron Rivest:
  - All generate 128-bit hash regardless of input size
  - All are public domain
- MD2:
  - Very slow
- MD4:
  - Much faster
- MD5:
  - Newer version of MD4
  - Much more complex, harder to break
  - Most common

Message Digest Version x: MDx
First, consider the MD family of hashing algorithms. All these were invented by Ron Rivest (the R of RSA Security Company).

This flavor of hashing algorithm has gone through a few iterations over the years. Today, the one that is most common, and therefore most important to focus on, is **MD5**.

Note that all the "MD Family" of hashing algorithms always generates a **128-bit hash**.



Secure Hash Algorithm: SHA

The U.S. National Institute of Standards and Technology (NIST) publishes the Secure Hash Algorithm, or SHA group, of hashing algorithms.

SHA-0 is considered flawed and no longer used.

SHA-1 is extremely common today. This algorithm always generates a 160-bit hash.

The "SHA-2" family, as it is sometimes called, is a collection of algorithms. The names of these algorithms tell you the length of the hash. (That is, SHA-256 generates a 256-bit hash.) As you can see, there are four different lengths of hashes available in this grouping.

In 2012, NIST approved the SHA-3 family as the eventual replacement for SHA-2. It is too early to tell if SHA-3 will ever become important or widely used. The hash lengths available in SHA-3 are the same as the lengths available in SHA-2. You may ask why NIST went to the trouble of approving the SHA-3 family when the hash lengths are the same? The answer is not easy to find, but MD5, SHA-1, and all SHA-2 algorithms are based on the same fundamental math. SHA-3 algorithms are based on different fundamental math. By having the SHA-3 family approved now, if anything happens to invalidate the fundamental math of the other algorithms, then SHA-3 is ready to go.

Message Digests: Problems

- Collision:
  - A one-way hash should not generate the same hash from two documents
  - When that does happen, it is called a "collision"
  - International researchers found these in MD5, and RIPEMD in 2005
  - Google researchers found them with SHA-1 in 2017
- Preimage *attack:*
  - Effectively a predictable collision
  - Very rare, but if it happens, the algorithm is truly broken

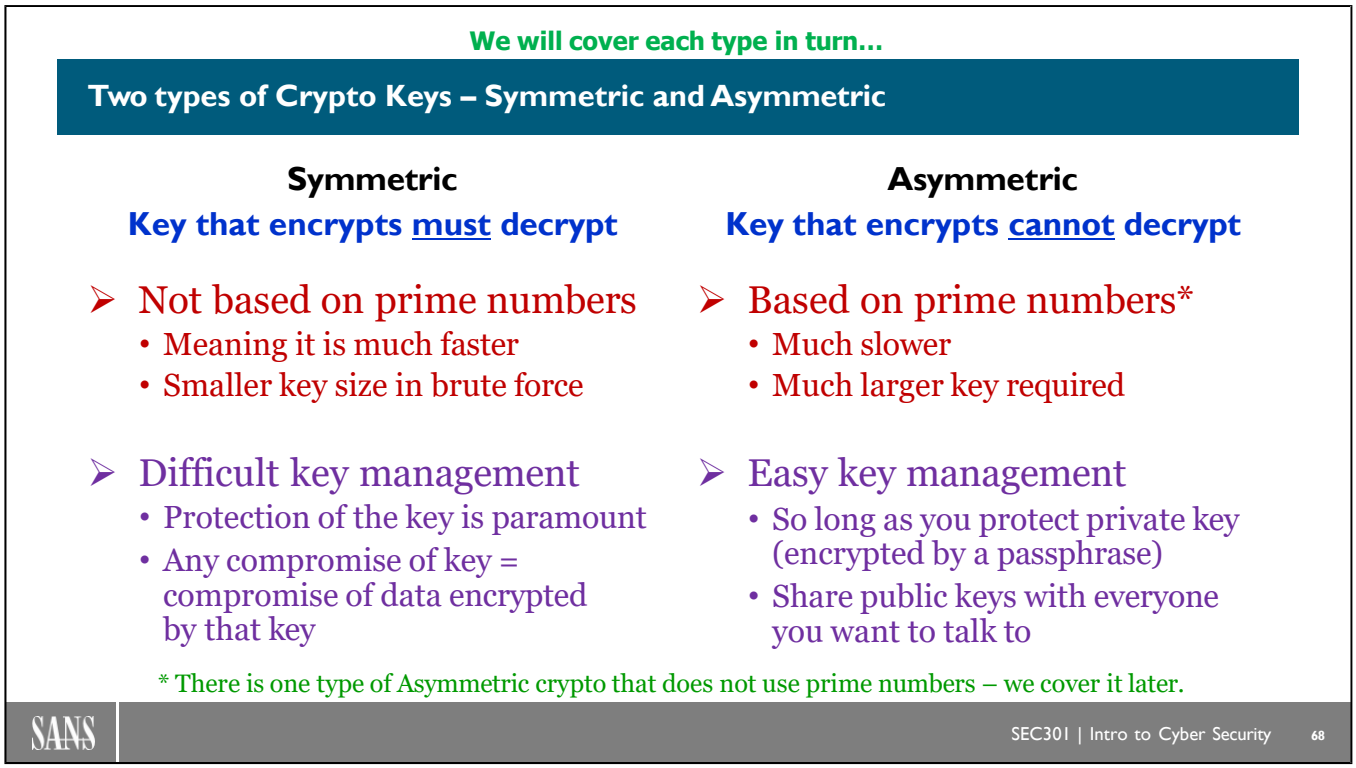
Message Digests: Problems
Several years ago, some international researchers published a paper in which they proved that they had found "collisions" with MD5 and RIPEMD. In 2017, Google researchers announced they had discovered collisions with SHA-1. Although this is certainly significant, it is not as devastating (or surprising) as many would have you believe.

First, it is true that they did indeed find collisions, which means that two different documents generated the exact same hash. This is not surprising if you think about it.

Let's take an MD5 128-bit hash as an example. How many combinations are there of 128 bits? 340 undecillion. (340,000,000,000,000,000,000,000,000,000,000,000,000). How many documents are there in the world? Some number greater than that!" Therefore, there will, of course, be collisions.

Let's simplify this equation: If there are 100 possible combinations of a hash and there are 101 documents in existence, then two of those documents **must** generate the same hash.

It is also important to note that although they did find collisions, they did not find *predictable collisions*. In other words, they could not tell in advance which two documents would generate the same hash and cause the collision. If a hashing algorithm causes predictable collisions, it is truly broken and should not be used going forward.



Two types of Crypto Keys – Symmetric and Asymmetric
As the slide suggests, there are two types of cryptographic keys. They are called Symmetric and Asymmetric.

Which key encrypts / decrypts
The first and most crucial difference between the two types is which keys are used to encrypt and decrypt.

- With Symmetric cryptography, the key that is used to encrypt must be used to decrypt.
- With Asymmetric, the key that is used to encrypt cannot decrypt – you use a different key.

When you look at a cryptographic system and need to know if it is Symmetric or Asymmetric, this is what you look at. If the key that encrypts also decrypts, it is symmetric. If you have to use a different key to decode, it is Asymmetric.

Symmetric is much faster than Asymmetric
The next significant difference between the two types of cryptography is the type of math that drives them.

- All Asymmetric algorithms except one are based on prime numbers (we cover the exception).
- Symmetric cryptosystems do not rely on primes.

This fact is vital for a couple of reasons. For some complex reasons we won't attempt to explain here, prime number math is a great deal slower than non-prime number math. This means

asymmetric cryptography's prime based math is slower than Symmetric's non-prime math. Depending on several variables, it can be between 100 and 10,000 times slower to use asymmetric systems than symmetric systems.

Asymmetric requires a MUCH longer key…
The next problem with the prime number based Asymmetric is you have to have a much longer asymmetric key to obtain the same level of security as a symmetric system provides (the same work factor). To understand why we need to return to our discussion of brute force.

- To brute force Symmetric cryptography, you must guess every number.
- To brute force Asymmetric cryptography, you only have to guess the prime numbers.

You should recall that if you have a key length of 56 bits, then your keyspace is 72 quadrillion possible combinations or keys. But if we are going to brute force the key and only have to guess the prime numbers, then how many of those 72 quadrillion possibilities are primes? The answer to this question is an amazingly tricky math problem, but suffice it to say the answer comes out to approximately 30% of all possibilities. (Note - that percentage will change with different keyspace sizes.)

The end result is that if you want the same work factor as a 256-bit Symmetric key, your Asymmetric key would have to be 15,360 bits long. Only then would you force the same number of guesses in a brute force of an Asymmetric algorithm.

Symmetric key management is hard – Asymmetric key management is easy
At this point, you may be wondering why on earth you would want to use an Asymmetric algorithm when it has all these drawbacks. There are two reasons we like Asymmetric.

- Symmetric key management is extremely difficult.
- Asymmetric key management is very simple by comparison.

As we will explain as we cover each type of key in more detail, key management is a critical issue. In fact, the difficulty of Symmetric cryptography's key management is what kept it from wide-spread adoption in the commercial sector for decades. The reason we like Asymmetric is that it simplifies Symmetric key's management. As you will see in the pages coming up, we can use Asymmetric cryptography to distribute Symmetric key to our users securely. This is another of those really vital points.

Symmetric Key Cryptography Introduction

- Commonly used for many years
- The key that encrypts **must** decrypt:
  - Vigenère cipher: Both sides require an identical key
  - ROT13: Both sides rotate the alphabet by 13 positions

Key

123 X _789_ + 456

Cryptovariable

Both sides must use 789 as their key in this example.

Remember…

The algorithm 123X ___+456 is public domain for peer review.

Only the key 789 is private.

Protection of the key is paramount!

Symmetric Key Cryptography Introduction

As just stated, with Symmetric key cryptography, the key that encrypts must be used to decrypt. We have actually shown several examples of this type of cryptography already. Think back to our earlier conversations. The Vigenère Cipher, the One-time-Pad, and the Code Book examples are all examples of symmetric key cryptography. Both communicating parties must possess the key (the book, the pad, the running key, etc.).

Even the ROT13 cipher is an example of a symmetric key. Both communicating parties must rotate the alphabet by the same amount (13 positions in that case).

A modern crypto algorithm is a large, very complex math formula. As a greatly simplified example, it might look like:

123 x _____ + 456

This is very complex math here. Notice it is not all there; part of it is missing. To use the algorithm, we have to complete the algorithm by filling in the "cryptovariable" with a cryptographic key. A key is a string of binary 1's and 0's which represent a numeric value. So for example, if we insert 789, the math now looks like this:

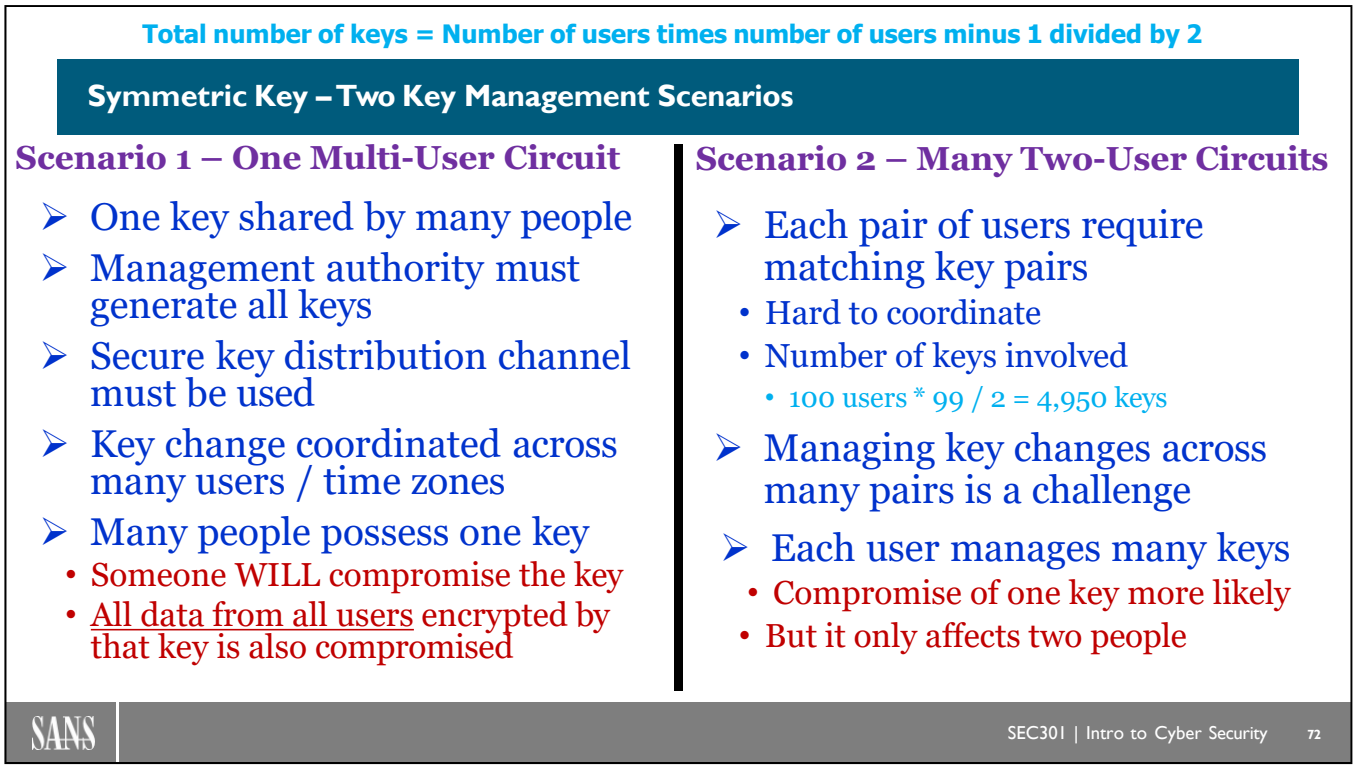
123 x 789 + 456

nd the ciphertext to the recipient.

On the receiving end, they must exactly recreate the algorithm to decrypt the information. If, for example, the recipient inserted a key of 987 instead of 789, it won't work.

Go back to the example of encrypting and decrypting with a Vigenère Cipher. If we encrypt something using a key of CRYPT, as in our earlier example, and then you attempt to decrypt with a key of SOUND, it is not going to decrypt correctly—you will get garbled output. You must use the exact same key on each end of the communication; therefore, Vigenère is an example of symmetric cryptography.

In the case of the Vigenère Cipher example above, the output of trying to decrypt with the wrong key is garbled text. With a modern cryptosystem, you might get garbled output, or you might simply get an error. In either case, the point is that it won't work that way.



Symmetric Key – Two Key Management Scenarios

An earlier slide mentioned that Symmetric key cryptography has difficult key management. This is true for a variety of reasons. To understand those reasons, let's look at two scenarios. In both situations, assume there are 100 users we need to support.

Scenario 1: Large Multi-User Circuit

To encrypt in this case, all users must possess an identical key. This means that a management authority of some kind must use their Pseudo Random Number Generator (PRNG) to generate reasonably random keys. The management authority must then come up with a secure key distribution channel. In the past, we used armed couriers and sometimes still do. Today, however, there are secure electronic key distribution methods available.

The next issue is that all users on the circuit need to change their keys regularly. The more data encrypted by a single key, the more likely an adversary will discern patterns and break the key in a ciphertext-only attack. This key change must be coordinated across all 100 users so that everyone does it simultaneously. Any delay by any user can cause data loss. This coordination becomes especially tricky when dealing with users in multiple time-zones.

The biggest problem with this scenario is the likelihood & damage of a key compromise. When one copy of a key is possessed by 100 users, it is only a matter of time, and one of them will make a mistake. Because the adversary now possesses the entire formula (remember, the

algorithm is public domain for peer review—only the key is unknown by others), they can decrypt any data encrypted by that key. Meaning, that compromise impacts all 100 users in our scenario.

Scenario 2: Many Two-User Circuits

In this case, each pair of users must have a matching key pair. This can be difficult to manage, especially across many people (such as our scenario of 100 users). The formula to determine the total number of keys required is the number of users times the number of users minus one divided by two. In other words:

100 * 99 / 2 = 4,950 total keys required to support 100 people.

In addition to the simple volume of keys, you also have to manage key changes across that many users. Granted, we are not trying to get 100 people to all change keys simultaneously as in the last scenario, but you do have to coordinate each pair. That means you have to coordinate 50 times instead of just once.

Finally, in the first scenario, each user is only managing one key at a time, which is reasonably straight-forward, but any compromise of the key is very damaging. In our second scenario, each user is managing many keys per day, which is much more complicated—making it more likely that someone will compromise a key. However, any such compromise only impacts two people. While any compromise is bad, in this scenario, it is not as damning.

Symmetric Key General Functions (1)

- Block cipher: The majority of symmetric algorithms:
  - Encrypts a block of text at a time
  - DES = 64-bit blocks (8 bytes)
  - AES = 128-bit blocks (16 bytes)
  - More efficient for bulk encryption
  - Must pad data to be an exact multiple of the block size

SANS | SEC301 | Intro to Cyber Security 74

Symmetric Key: General Functions (1)

The majority of symmetric algorithms operate as a block cipher, meaning they encrypt a block of text at a time. The most common block size today is 64 bits. (Though some algorithms do use other block sizes, as you will see.)

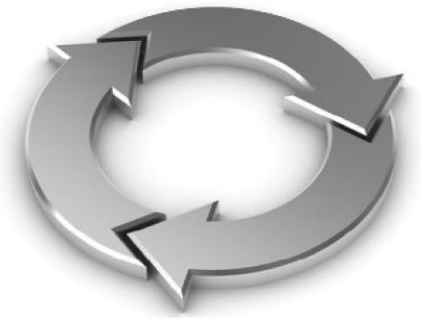
Examples of block ciphers that you have seen in this lecture follow:

- **The Column Shift cipher:** That example encrypted in 4-byte (32-bit) chunks
- **The Rail Fence cipher:** Which encoded the whole string as a block
- **The Code Book:** Which was encoding a word at a time

In all cases, a block cipher must pad the end of the data to be an exact multiple of the block size. For example, if the block size is 64 bits (8 bytes) and the last block of text to be encrypted is only 48 bits (6 bytes), then the cipher must pad the last 2 bytes onto the end of the plaintext before encrypting.

Symmetric Key General Functions (2)

- Takes various encryption functions ...
  - Rotational substitution
  - Permutation
  - XOR
  - Etc.
- Repeats all of them several times
  - Called the "number of rounds" or "rounds of computation":
    - The number of times a cryptosystem runs data through its process
    - Example: AES-256 uses four basic encryption functions and repeats them up to 14 times

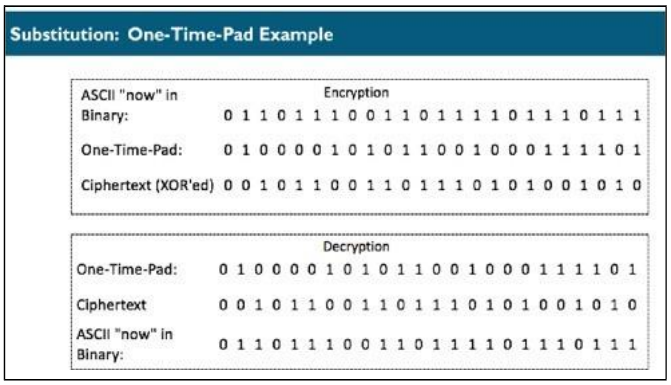


Symmetric Key: General Functions (2)

Symmetric block ciphers often use a variety of encryption functions such as rotational substitution, permutation, and so on. See the earlier History of Cryptography section for examples. A block cipher then repeats all of those functions some number of times. This is called the "number of rounds" or "rounds of computation." For example, the AES-256 algorithm uses four basic encryption functions and repeats them 14 times.

Symmetric Key General Functions (3)

- Stream cipher—less common:
  - Encrypts 1-bit of data at a time
  - Efficient for smaller transactions (ATM transactions and so on)
  - Requires more computational power, best when implemented in hardware
  - One-time Pad is an example:



Symmetric Key: General Functions (3)

There are fewer symmetric stream cipher algorithms, but some are commonly used. For example, the RC4 algorithm is a stream cipher that is quite common.

With a stream cipher, you encrypt one bit of data at a time. Again, you already fully understand this concept. Simply go back to the One-time Pad. That is a perfect example of a stream cipher.

Deprecate: It is still available, but not recommended. Something better is available.

DES and Triple DES

- IBM's Lucifer chosen as Data Encryption Standard (DES): 1975
  - 56-bit key, 64-bit block, 16 rounds of computation
  - Due to Moore's Law: By late 1990s, could be brute forced too quickly
- Triple DES (encrypt w/key 1, encrypt w/key2, encrypt w/key3)
  - 168-bit key (56x3), 48 rounds (16x3), 64-bit block (does not change)
- If you use Triple DES today, **accelerate the move to AES!**
  - On July 11, 2017, NIST deprecated Triple DES
  - Will no longer be allowed in TLS, IPSec, and others
  - You should not be able to obtain a certificate that supports Triple DES

DES and Triple DES

In the early 1970s, the U.S. government put out the call to have all the best encryption algorithms of the day evaluated. It would pick one to become the Data Encryption Standard. In 1975, an algorithm from IBM known as Lucifer was selected and published as this new standard. (The formal document declaring the DES standard was not published until 1975.)

Interestingly, the Lucifer cipher utilized a 128-bit key, but the key length was shortened to 56 bits. This caused some of the brute force problems we will discuss in a few minutes.

The DES algorithm is reasonably fast. It has a 56-bit key, operates on 64-bit blocks of text, and performs 16 rounds of computation before spitting out the ciphertext.

In 1998, it was reported that DES could be brute forced in three days with a purpose-built computer costing $250,000. Today, a computer that can brute force DES in under one day (hours) costs less than $5,000.

When the reports that DES could be brute forced so quickly came out, we moved to Double DES. In other words, we would encrypt the information with DES using one key. Then we would encrypt it again using a second key. We thought this would be a lot better. Unfortunately, it wasn't. Ron Rivest proved in a paper that Double DES is not significantly better than single DES. It was, in fact, like changing **single** DES from a 56-bit key to a 57-bit key. Some improvement, but not enough. Mr. Rivest described a "meet-in-the-middle" attack in which you essentially approach the problem from both ends at once and solve the problem almost as quickly as if it were single DES.

A footnote on the timeline, since it sometimes causes confusion: March 17, 1975, DES was published in the Federal Register, meaning it was published for peer review at that time. However, DES was not formally adopted as a standard until November 1976, and formally published as such in January 1977.

In the same paper, he also proved that if you do it a third time, it is 2^{128} better. In other words, a significantly longer work factor. 2^{128} is a really big number!

This became known as Triple DES and is still in use today. With Triple DES (or 3DES as you will sometimes see it), you have an effective key length of 168 bits (56 times 3).

On July 11, 2017, the U.S. National Institute for Standards and Technology (NIST) formally began the process of "deprecating" Triple DES. To begin, they are shortening the amount of data you should encrypt with a single set of keys. They also republished the standards for Transport Layer Security (TLS) and IPsec, disallowing the use of Triple DES. They are considering the same for other protocols.

In computing, when you "*deprecate*" something, it means that it is still there and will still work, but something new and better is available. Deprecation is a strong suggestion that you stop using the old and move to the new.


dep·re·cate
/ˈdeprəˌkāt/

verb
verb: **deprecate**; 3rd person present: **deprecates**; past tense: **deprecated**; past participle: **deprecated**; gerund or present participle: **deprecating**

1. express disapproval of.
   "he sniffed in a deprecating way"
   - (chiefly of a software feature) be usable but regarded as obsolete and best avoided, typically due to having been superseded.
     "this feature is deprecated and will be removed in later versions"

Important note: If possible, you should use the AES algorithm because it is more secure. If for some reason that is not possible, and you must use DES, make sure you use Triple DES. DO NOT use single DES. It can be broken too quickly via a brute force attack.

Reference for deprecation

https://csrc.nist.gov/News/2017/Update-to-Current-Use-and-Deprecation-of-TDEA

Advanced Encryption Standard: AES

- Symmetric block cipher
- Originally called Rijndael:
  - Chosen by NIST as AES in October 2000 after 3-year selection process

- Varying key size and number of rounds – always 128 bit block size:
  - AES 128 = 128-bit key, 10 rounds
  - AES 192 = 192-bit key, 12 rounds
  - AES 256 = 256-bit key, 14 rounds
- Widely implemented in a variety of cryptosystems

Advanced Encryption Standard: AES
In the late 1990s, reports started surfacing that DES could be brute forced too quickly. So, the National Institute of Standards and Technology (NIST) again put out the call for encryption algorithms. One would be chosen as the new Advanced Encryption Standard. In October 2000, they chose the Rijndael algorithm to become AES.

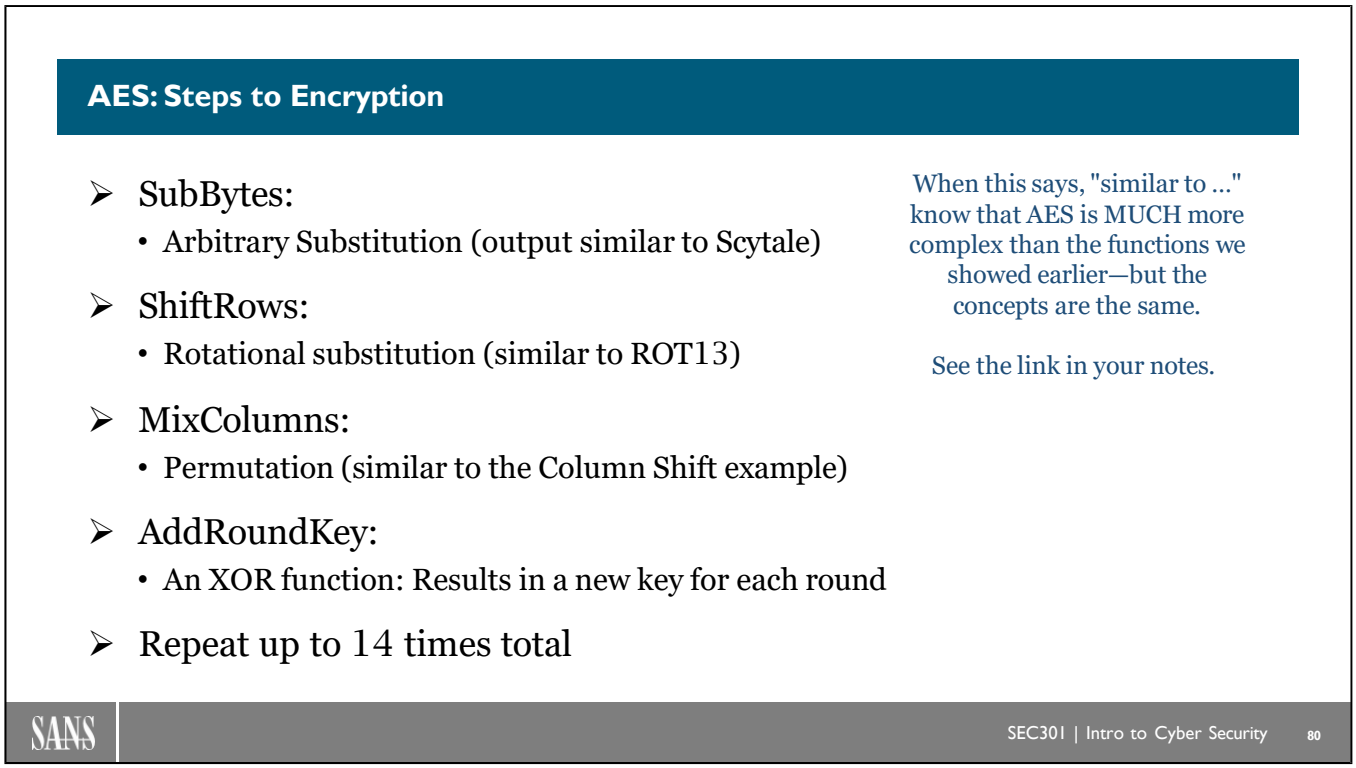
It seems that a new system comes out every day using AES. It is common and becoming more so. It is widely considered to be extremely secure (assuming it is correctly implemented).

As you can see here, AES always uses a 128 bit block size. It has three key lengths to choose from: 128 bit, 192 bit, and 256 bit. When you choose those key lengths, you are also choosing the number of rounds of computation: 10, 12, and 14, respectively.

AES Selection Timeline:

- January 2, 1997 – NIST announces AES development effort
- August 20, 1998 – NIST announces 15 candidates – requests peer review on all 15
- April 15, 1999 – NIST announces 5 finalist - MARS, RC6, Rijndael, Serpent, and Twofish
- October 2, 2000 – NIST announces Rijndael will become the AES

Reference:
https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development



AES: Steps to Encryption
Believe it or not, at this point in our conversation, you have all the tools in your mind to understand the gist of how AES works. When AES encrypts data, it runs four separate operations on the data to produce ciphertext.

The functions follow:

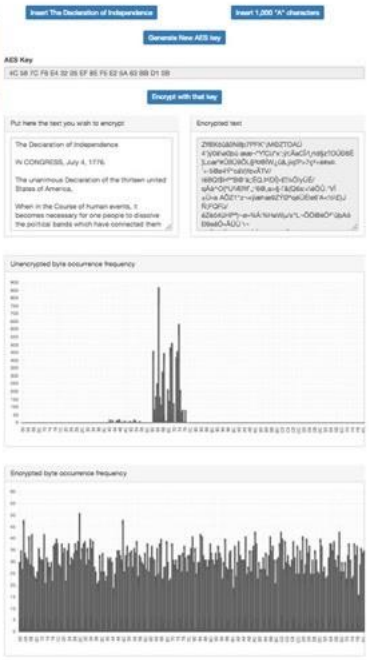
- **SubBytes:** Arbitrary substitution that generates an output similar to what you saw earlier with the Scytale cipher
- **ShiftRows:** Rotational substitution similar to ROT13 or Caesar Cipher
- **MixColumns:** Permutation similar the Column Shift example
- **AddRoundKey:** A deceptively (controversially) simple XOR operation to randomize the key **for each round**

It combines those four functions and repeats them up to 14 times. Just please remember that AES uses ***much*** more complex actual functions than the simple examples provided here. The examples shown (Scytale, ROT13, Column Shift, and such) are conceptually similar. They are not exactly the same, however.

If you would like to understand AES better, there is a great stick figure drawing explanation at http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

Lab Time

- LAB 3.2: Visual Crypto
- Objectives:
  - See plaintext turn into ciphertext
  - See ciphertext turn back into the original plaintext
  - See the randomness using histograms
  - See what happens if the key is wrong
  - See what happens if the ciphertext changes
  - Estimated completion time: 25 minutes



Lab Time

LAB 3.2: Visual Crypto

Asymmetric Cryptography (1)

- Two mathematically linked keys:
  - One called the "Public Key"—One called the "Private Key"
    - Give the public key to anyone—Keep the private key to yourself (passphrase protected)
  - Possession of one key does not allow you to discern the other key
  - The key that encrypts **cannot** decrypt:
    - Just backward of symmetric—and the most fundamental difference
  - Anything encrypted by one key can ONLY be decrypted by the mathematically linked key

- Allows for four processes:
  - Encrypt to send
  - Decrypt to read
  - Create digital signature
  - Verify digital signature

Asymmetric Cryptography (1)

Now we move on to asymmetric algorithms. As stated, with symmetric systems, the key that encrypts must decrypt. With asymmetric, the key that encrypts cannot decrypt; it requires a different key. Let's understand how this could work.

When you are installing software that utilizes this type of key, part of the installation process is the creation of a pair of mathematically linked keys. One is called the *public key* and one is called the *private key*.

Important points to understand about these keys:

- If you have one, it is not feasibly possible to discern the other. For example, anyone can (and will) have your public key. That does not give them any insight into your private key.
- You distribute your public key to everyone you want to communicate with.
- You do not distribute your private key to anyone. You keep it private (to yourself).
- Anything encrypted by one key can be decrypted ***only*** by the mathematically associated key:
  - If I encrypt with your public key, you can decrypt only with your private key.
  - Likewise, if I encrypt with my private key, you can decrypt only with my public key.

Yes, this is confusing the first time you see it. Stay tuned for several examples that will help make it more clear.

Asymmetric Cryptography (3)

- Before using this type of crypto, there MUST be a key exchange:
  - If Alice and Bob are going to talk, then …
  - Alice must give Bob her public key
  - Bob must give Alice his public key:
    - How they accomplish this does not matter
    - Anybody can have Alice's and Bob's public key
    - So, a secure distribution channel is not necessary
  - Alice has: Her private and public key, Bob's public key
  - Bob has: His private and public key, Alice's public key

SANS SEC301 | Intro to Cyber Security 83

Asymmetric Cryptography (3)

Understanding this type of cryptosystem requires an understanding of its key exchange requirements. Before using asymmetric key systems, there ***must*** be a key exchange. For example:

- If Alice and Bob are going to talk, they must exchange each other's public keys.
- Alice must give Bob her public key.
- Bob must give Alice his public key.

In the end, how they accomplish this does not matter. Any number of methods are available. Perhaps they exchange a thumb drive or an old floppy disk. Perhaps they each have their public key stored on a server somewhere and their respective software retrieves the public key of the other. The simplest method: If you digitally sign an email, your public key is an attachment to that email. So perhaps they simply sent each other a signed email message.

Because anybody can have their public key and still not discern their private key, how they exchange the public key is not critically important. A secure key distribution channel is not required for an asymmetric key as it is for a symmetric key.

The bottom line: Before Alice and Bob can communicate, the following has to be in place:

- Alice needs her private key and Bob's public key.
- Bob needs his private key and Alice's public key.

Asymmetric Cryptography (2)

	Four Processes	Who is doing it?	Which keys do they have?	What is the goal?	Which Key!
1	Encrypt to send	Sender	• Sender Private • Sender Public • Recipient Public	Only Recipient can read	Key associated with Recipient **-- Recipient Public --**
2	Decrypt to Read	Recipient	• Recipient Private • Recipient Public • Sender Public	Only Recipient can read	Key only Recipient has **-- Recipient Private --**
3	Create Signature	Sender	• Sender Private • Sender Public • Recipient Public	Prove it came from Sender	Key only Sender has **-- Sender Private --**
4	Verify Signature	Recipient	• Recipient Private • Recipient Public • Sender Public	Prove it came from Sender	Key associated with Sender **-- Sender Public --**

- **Answer these questions logically and you will always get the correct answers.**
- **Answer these questions correctly and you will always arrive at the correct key.**

Asymmetric Cryptography (2)

When using asymmetric key crypto, there are four possible processes you might want to accomplish. There are also four keys. One of the most common mistakes people make with these systems is using the wrong key for the wrong purpose.

Following are the four processes:

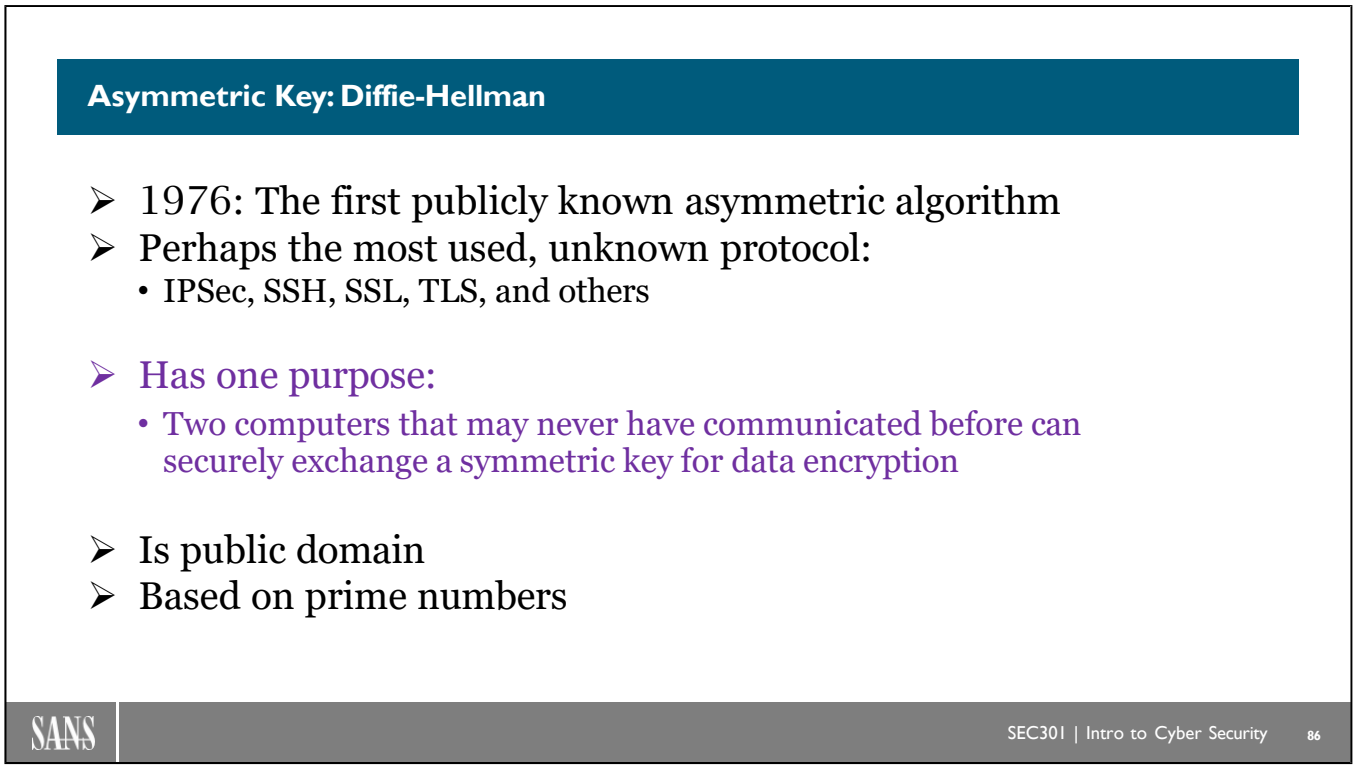
- **Encrypt to send:** Use the recipient public key.
- **Decrypt to read:** Use the recipient private key.
- **Generate a digital signature:** Use the sender private key.
- **Verify a digital signature:** Use the sender public key.

If you follow the purpose of these operations through and understand the goal of each, the correct key for the correct process becomes obvious. Let's start by examining the first two.

The logic process depicted on the slide (and larger on the next page) is the exact process the author goes through each time he has to ensure he selects the correct key. The process works 100% of the time if you answer each question logically.

	Four Processes	Who is doing it?	Which keys do they have?	What is the goal?	Which Key!
1	Encrypt to send	Sender	• Sender Private • Sender Public • Recipient Public	Only Recipient can read	Key associated with Recipient **-- Recipient Public --**
2	Decrypt to Read	Recipient	• Recipient Private • Recipient Public • Sender Public	Only Recipient can read	Key only Recipient has **-- Recipient Private --**
3	Create Signature	Sender	• Sender Private • Sender Public • Recipient Public	Prove it came from Sender	Key only Sender has **-- Sender Private --**
4	Verify Signature	Recipient	• Recipient Private • Recipient Public • Sender Public	Prove it came from Sender	Key associated with Sender **-- Sender Public --**

- **Answer these questions logically and you will always get the correct answers.**
- **Answer these questions correctly and you will always arrive at the correct key.**

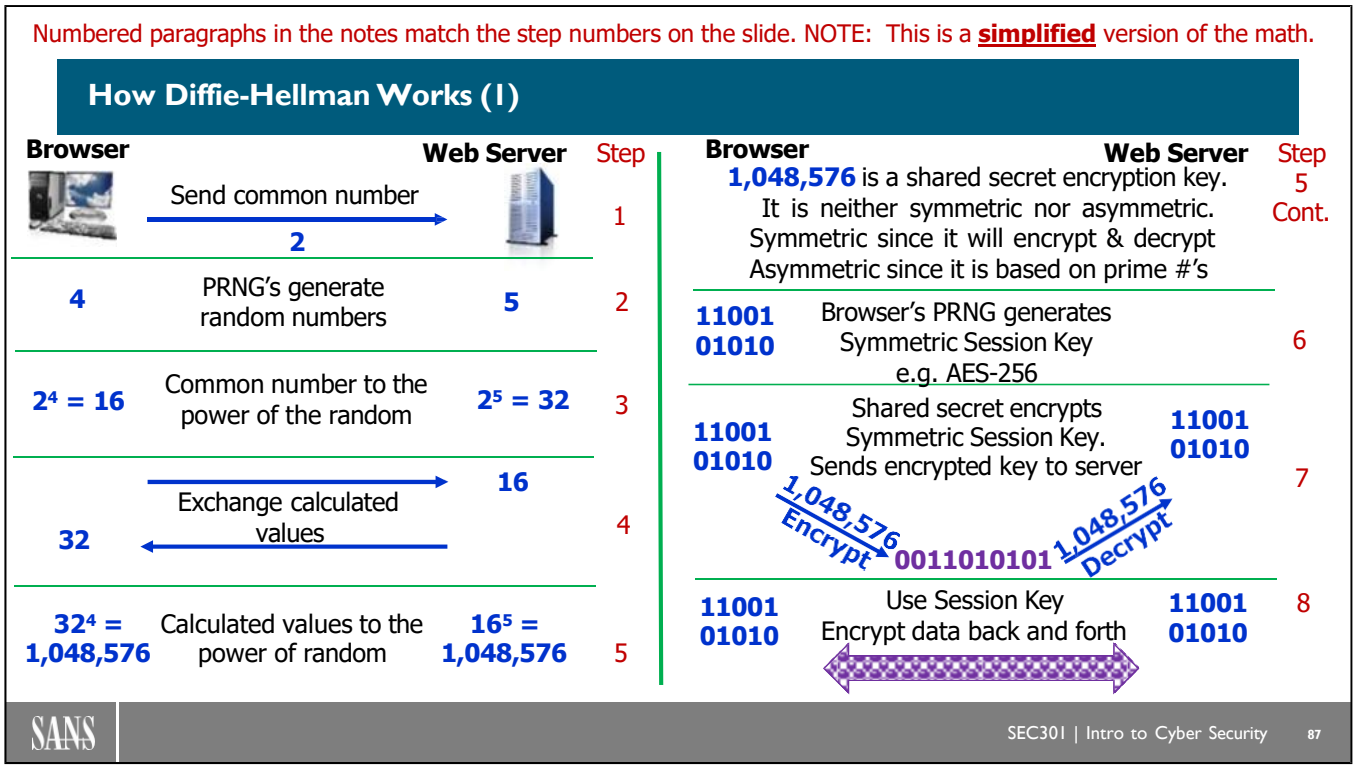


Asymmetric Key: Diffie-Hellman
The first publicly available asymmetric algorithm was released in 1976 and is known as Diffie-Hellman. (There is some compelling evidence that secret government programs in the United States, Great Britain, and other countries made the discovery of asymmetric crypto prior to the advent of Diffie-Hellman's.)

This is one of those things that does only one thing and does it extremely well. It allows two computers, which may never have communicated in the past, to securely exchange a symmetric key. They then use that symmetric key to exchange data.

Diffie-Hellman is perhaps the most common, unheard of protocol in existence. You almost certainly use Diffie-Hellman every day, probably several times a day. Even if you have never heard of it before.

There are many encryption methods that use the Diffie-Hellman key exchange protocol. Some of them include IPsec-based VPNs, SSH (Secure Shell), and eCommerce or online banking using HTTPS in your browser.



How Diffie-Hellman Works (1)

Diffie-Hellman utilizes a multistep process. Because it is so commonly used (and so widely misunderstood as to how it actually functions), we will go through those steps here. As we move through the steps, remember that the two computers in question may have never communicated before.

In this example, we will use a web browser and a web server setting up encryption for an HTTPS session. Remember that SSH, IPSec, and several other functions also utilize Diffie Hellman.

1. To begin, the web browser's PRNG generates a number and sends it to the server. For the math that follows, both sides must know this common number. Note that the math will only work correctly if this is a prime number. It so happens that 2 is a prime number, so it will work for a simple explanation.
2. Each side of the communication then uses their PRNG to generate random numbers. For our simplified example, we will use 4 on the browser's side and 5 on the server's side.
3. Each side then takes the common number to the power of the random number they just generated. For example, on the browser's side, the computer takes 2 to the power of 4 and calculates a new value of 16. On the server's side, the computer takes 2 to the power of 5 and generates a value of 32.
4. Each side sends these newly calculated values to the other side. The browser sends 16 to the server. The server sends 32 to the browser.

5. Each side takes the calculated value to the power of their individual random values from earlier in the process. The browser takes 32 that it received from the server to the power of 4 and generates 1,048,576. The server takes 16 that it got from the browser to the power of its random 5 and also generates 1,048,576.
   The number 1,048,576 is a shared secret encryption key. The key is neither symmetric nor asymmetric since it has properties of both. It is symmetric in that one value will be used to both encrypt and decrypt. It is asymmetric in that it is based on prime numbers and is, therefore, very slow. While it is technically feasible to encrypt your data using this value, it is never done that way. It would be much too slow! Instead, we will use it to encrypt an encryption key—usually no longer than 256 binary bits, which will be quick regardless of the encryption method used.
6. Next, the browser's PRNG generates a symmetric session key. It is truly symmetric in nature—perhaps AES-256-bit key for example. The term session key means that this key will encrypt this one web page, and that is all it will ever be used for.
7. On the browser's side, the shared secret generated in step 5 is then used to encrypt the symmetric session key. The encrypted form of the symmetric session key transmits to the server. There, the server decrypts the symmetric session key using its copy of the shared secret from step 5.
8. The browser and server proceed to utilize the symmetric session key to both encrypt and decrypt traffic to each other.

Please note this example of the math is **GREATLY SIMPLIFIED**. There is a great deal of complexity removed here, such as a ton of modulus arithmetic, and so on. This is just the essence of the process.

There is a truly fantastic explanation of Diffie-Hellman in this YouTube video:

https://www.youtube.com/watch?v=3QnD2c4Xovk

Numbered paragraphs in the notes match the step numbers on the slide. NOTE: This is a **simplified** version of the math.

How Diffie-Hellman Works (1)

Browser		Web Server	Step
	Send common number → 2		1
4	PRNG's generates random numbers	5	2
$2^4 = 16$	Common number to the power of the random	$2^5 = 32$	3
32 ←	Exchange calculated values	→ 16	4
$32^4 =$ 1,048,576	Calculated values to the power of random	$16^5 =$ 1,048,576	5

Browser		Web Server	Step
1,048,576 is a shared secret encryption key. It is neither symmetric nor asymmetric. Symmetric since it will encrypt & decrypt Asymmetric since it is based on prime #'s			5 Cont.
11001 01010	Browser's PRNG generates Symmetric Session Key e.g. AES-256		6
11001 01010	Shared secret encrypts Symmetric Session Key. Sends encrypted key to server 1,048,567 Encrypt → 0011010101 → 1,048,567 Decrypt	11001 01010	7
11001 01010	Use Session Key Encrypt data back and forth ↔	11001 01010	8

NOTE: **the common number (2 in the example) must be a prime number**

How Diffie-Hellman Works (2)

- How can this be secure if the common number (2) and the calculated numbers (16 & 32) are passed in the clear???
  - The attacker would have to predict the PRNG generated numbers (4&5 in the example)
  - In the last slide, we used 2, 4, and 5, but those are actually 4,096-bit numbers
  - Which are 1,234-digit decimal values
  - So the example numbers of 2, 4, and 5 we used really look like this:

466268828538646053042085021399969458248158094283464507456188077039233539653378069177978315920917436325490720325139300096623804
4
099119741300664689390534647276431986578942063560456025484254776907591920593431858118225015949145521338951686761225358762616300
0
688159014859189648246402005007456996453549665662371001100099799770678996392093381961778299830658820588847256136861711573911770
3
822208980651103526477920843458064088796419594332775397144126044726851328594292997186747653972695496449783990943735101564594527
1
703646793409368355755261167967751933957231455501835655402716265961929398875350445198471258548412781789901160494809760411442828
5
667352051945512109808722372086954703515155359707291538452872947979960812170372464248179139308283451915922880394145815643585112
7
388470675284309827096931638921172009587294741689250817480319696215834997531395810996884538781982040599043221935072545674940092
4
601806064443932419048885362283985378201699232815004232648837067142773040113385451088357842444187420570671806673187406549320124
4
030721282725397996842831925434725815308833195655123811049118172484701942715381179190848094143194875333631518496620153673870923
6
822251608505718201177620955500276173123611657489975485230600726049028152871711784621373678922578953481849999041543009724534119
5
074965975943224908219780396284925081748225358762616300068815901485918964824640200500745699645354966566237100110009979977067899
6
392093381961778299830658820588864248179139308283451915922880394145815643585112738847067528430982709693163892117200958729474168
9
2508174803196962158349964727643198657894206356045602548 42 – which is remarkably difficult to predict or brute force!

SANS SEC301 | Intro to Cyber Security 90

How Diffie-Hellman Works (2)

Probably the most common question we get when we first explain Diffie-Hellman is, “How can this be secure if the common number and both calculated numbers are shared in the clear for all to see?” It is a very good question and we are glad you asked! Here is the answer…

The answer is that in order to decrypt the symmetric session key, an attacker would have to know the common number (2 in our example) which of course they do know since it was shared in plaintext. The attacker would also need one of the private PRNG generated numbers (4 on the browser side, 5 on the server side in the example). The strength of Diffie-Hellman boils down to how hard it would be to predict or brute force one of those numbers. In our example, we used the numbers of 2, 4, and 5 to keep things simple. In real-world implementation, those three numbers are usually each 4,096 binary bits which is 1,234 digits in length. An example of such a number is shown on the slide.

Predicting or brute forcing such a number is virtually impossible today. Of course, there are other factors you would have to take into consideration. For example, if the numbers (2, 4, and 5) are not random enough, prediction becomes much simpler.

Source of the random number above:

https://stattrek.com/statistics/random-number-generator.aspx

Asymmetric Key: RSA

- Named for creators: Ron **R**ivest, Adi **S**hamir, Leonard **A**dleman:
  - Created in 1978 at MIT
- Probably the most common asymmetric algorithm for encryption and digital signatures:
  - Considered a de facto standard worldwide
- Security comes from "the difficulty of factoring large integers into its two prime factors."
- Can be used for:
  - Encryption
  - Digital Signatures
  - Secure Key Exchange

Asymmetric Key: RSA

The RSA algorithm is probably the most commonly used asymmetric algorithm today. It was invented by Rivest, Shamir, and Adleman: The three letters of the RSA algorithm and of the company RSA (now owned by EMC). The algorithm is based on the difficulty of factoring large prime numbers.

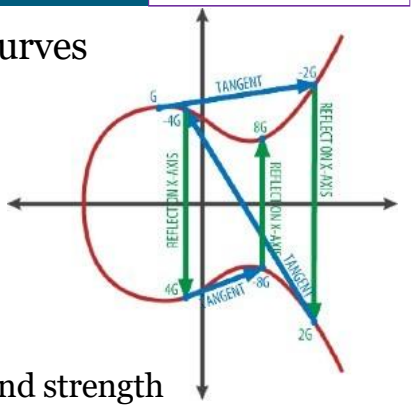
Created in 1978, the RSA algorithm can perform secure key exchange like Diffie-Hellman. Unlike Diffie-Hellman, it can also be used for encryption and digital signatures. Initially patented, it has been public domain since September 2000. Since then, it has become so common that it is now considered a de facto worldwide standard.

Note: The definition quoted above comes from
http://searchsecurity.techtarget.com/definition/RSA

Asymmetric Key: Elliptical Curve Crypto: ECC (1)

We will not explain this diagram! The point is, it is complex stuff.

- Based on the difficulty of factoring Elliptical Curves
- Provides:
  - Encryption
  - Digital Signature
  - Key exchange
- Very efficient:
  - An exception to the rule on asymmetric key length and strength
    - In rough numbers, 128-bit symmetric = 160-bit ECC, 256 bit symmetric = 512 bit ECC
  - Therefore, implemented in smart cards, PDAs, phones, and other devices with limited storage/processing capability



Asymmetric Key: Elliptical Curve Crypto: ECC (1)
The Elliptical Curve Cryptosystem (ECC) is the exception to the rule regarding asymmetric algorithms using prime numbers. Instead, it is based on the difficulty of factoring points on an elliptical curve (complex math stuff).

The result: Because it is *not* based on prime numbers, the earlier discussions about asymmetric being slower and requiring much larger keys do not apply to ECC. It is fast and efficient, even in less powerful hardware. The key does not need to be nearly so long as with something like RSA or Diffie-Hellman (DH).

	AES	RSA	ECC
Equivalent Security	256-bit	15,360-bit	512-bit

The high-efficiency and smaller key size make ECC popular. It is especially making headway in smart card implementations, smartphones, and other devices with limited storage and processing capability.

Incidentally, there is also Elliptical Curve Diffie-Hellman (ECDH) key exchange. It is also growing rapidly in popularity.

Asymmetric Key: Elliptical Curve Crypto: ECC (2)

- Patents on 5 curves held by Certicom of Ontario, Canada:
  - There is some debate about the patents
- NSA licensed Certicom's ECC implementation for $25M:
  - NSA's Commercial National Security Algorithm Suite specifies:
    - AES 256 bit
    - Elliptical Curve Digital Signature Algorithm (ECDSA) signatures
    - Elliptical Curve Diffie-Hellman (ECDH) key exchange
      - Both require the use of curve P-384
    - SHA-384
  - With proper combinations of the preceding, you can encrypt up to U.S. Top Secret information:
    - Appears to be a strong endorsement of AES, SHA-384+, and ECC by the NSA

Asymmetric Key: Elliptical Curve Crypto: ECC (2)

The company Certicom of Ontario, Canada, holds a patent on ECC. There is some debate as to the validity of those patents and/or as to exactly what they cover. But it appears the patents are solid. The bottom line: ECC adoption is growing but would be growing more rapidly without the patents and resulting lawsuits.

Interestingly, the NSA licensed Certicom's ECC implementation for $25 million. In a 180-degree turnaround, the National Security Agency now allows for the use of public domain algorithms to encrypt U.S. classified information up to Top Secret. NSA has never even considered this in the past. This seems to be a strong endorsement of AES, ECDH, ECDSA (Elliptical Curve Digital Signature Algorithm), and SHA-384.

You can learn more about this at the link below; however, as of Sep 14, 2018, the site has an expired certificate so most browsers will not allow you to view the site. ;~)

https://www.iad.gov/iad/programs/iad-initiatives/cnsa-suite.cfm

First, for some complex reasons we won't attempt to explain here, prime number math is a great deal slower than nonprime number math. This means asymmetric cryptography is slower than symmetric (which does not rely on prime numbers). As stated earlier, depending on a number of variables, it can be between 100 and 10,000 times slower to use asymmetric systems than symmetric systems.

The second problem with the prime number issue is you have to have a much longer asymmetric key to obtain the same level of security as a symmetric system provides (the same work factor).

We need to return to our discussion of brute force. For the purpose of explanation here, we use the 56-bit key length of the DES algorithm. (Yes; DES is symmetric, but we need a number of bits to work with and 56 will do the trick.)

If we look at a 56-bit key, there are more than 72 quadrillion combinations. (The exact number is below.) So, if we want to brute force that key, that is the number of guesses it would take to go through every possibility. However, how many of those 72 quadrillion combinations are prime numbers? If we are brute forcing a key and have to guess only the prime numbers, then the total number of guesses required is much lower.

Let's use some shorter numbers for easier explanation. If you have:

- 25 total combinations of key, only 9 of them are prime numbers.
- 100 combinations of key, only 25 of them are prime numbers.
- 1,000 combinations of key, only 168 of them are prime numbers

And so on. To brute force a symmetric system with 1,000 possible keys, you would have to make up to 1,000 guesses. With a prime number-based asymmetric system with the same number of bits, you would only have to make up to 168 guesses.

Now let's return to 56 bits. Up to the number 10,000,000,000 (10 billion), mathematicians can count the number of primes. After that number, they use a formula to calculate the number of primes they think there are. Here are the actual numbers.

72,057,594,037,927,900: The number of combinations of 56 bits—72 quadrillion.
279,238,341,033,925: The number of prime numbers in 10 quadrillion—279.2 trillion.

Granted, that is still a lot of guesses required, but it is also a lot fewer. As the number of bits increase, the gap between these two numbers also increases. The end result is:

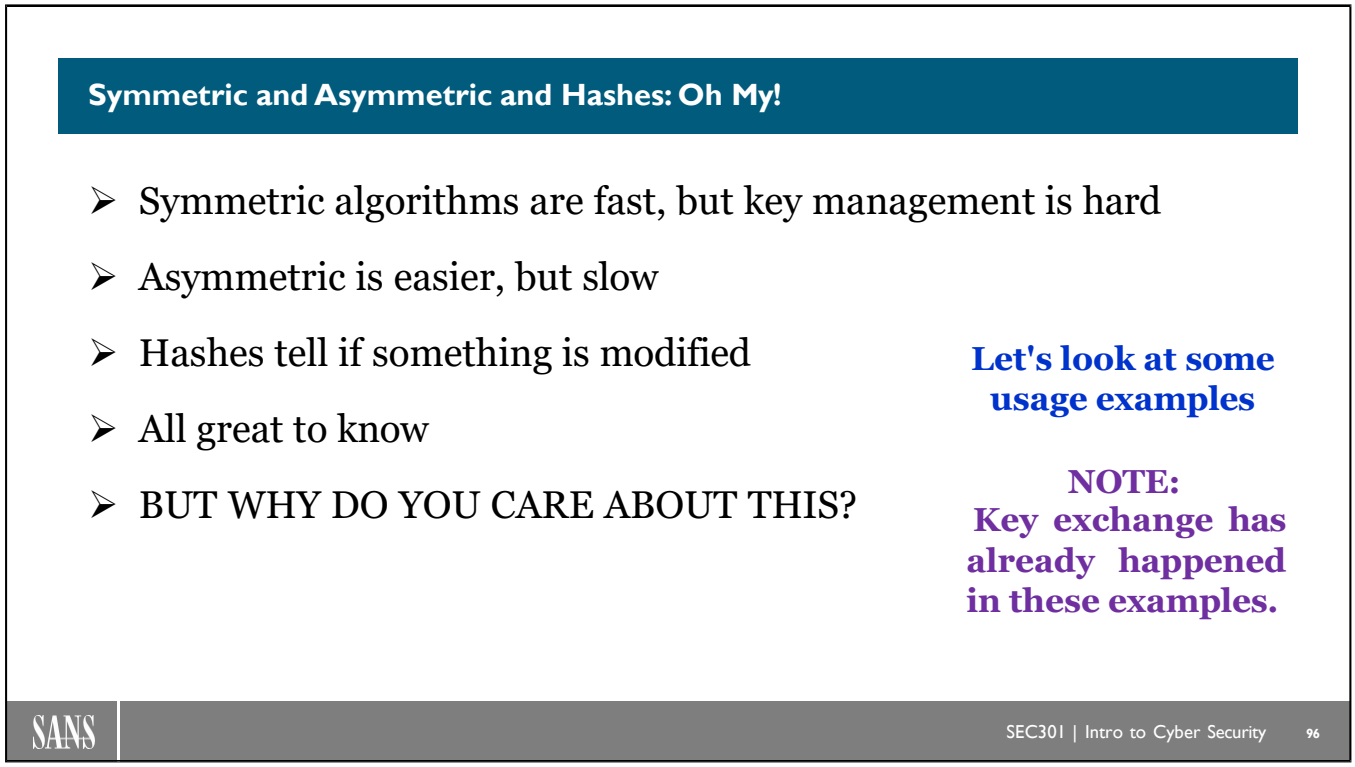
> 256-bit symmetric key is equivalent to
> 15,360-bit asymmetric key

That is how much bigger a prime number-based asymmetric key has to be to provide the same level of protection as the AES 256-bit algorithm.

(**Author's note:** Try as I might, I could not find a reference giving the exact number of prime numbers in 72,057,594,037,927,936 and I am not enough of a mathematician to calculate it myself. The 279 trillion figure above is for 10 quadrillion. This comes from the following link.)

Reference

https://primes.utm.edu/howmany.html



Symmetric and Asymmetric and Hashes: Oh My!
Okay. We have looked at a lot of information about symmetric algorithms. We have looked at asymmetric algorithms in great detail. We have even examined one-way hashes.

If you are like most students, right about now, you have one burning question in your mind:

WHY DO I CARE?

Let's look at some examples to find out the answer to that question.

Digital Signature (1)

- A generic term to describe a process to:
  - Verify who sent the document
  - Verify the document received is the exact same document that was sent
- The detailed steps to generate and verify a signature vary with different implementations
- An example follows

Digital Signature (1)
There are several slides that follow showing various cryptographic functions in real-world scenarios. Specifically, each of these slides is based on sending email using Microsoft Outlook.

We touched briefly on digital signatures earlier. Let's add some detail to understand them fully.

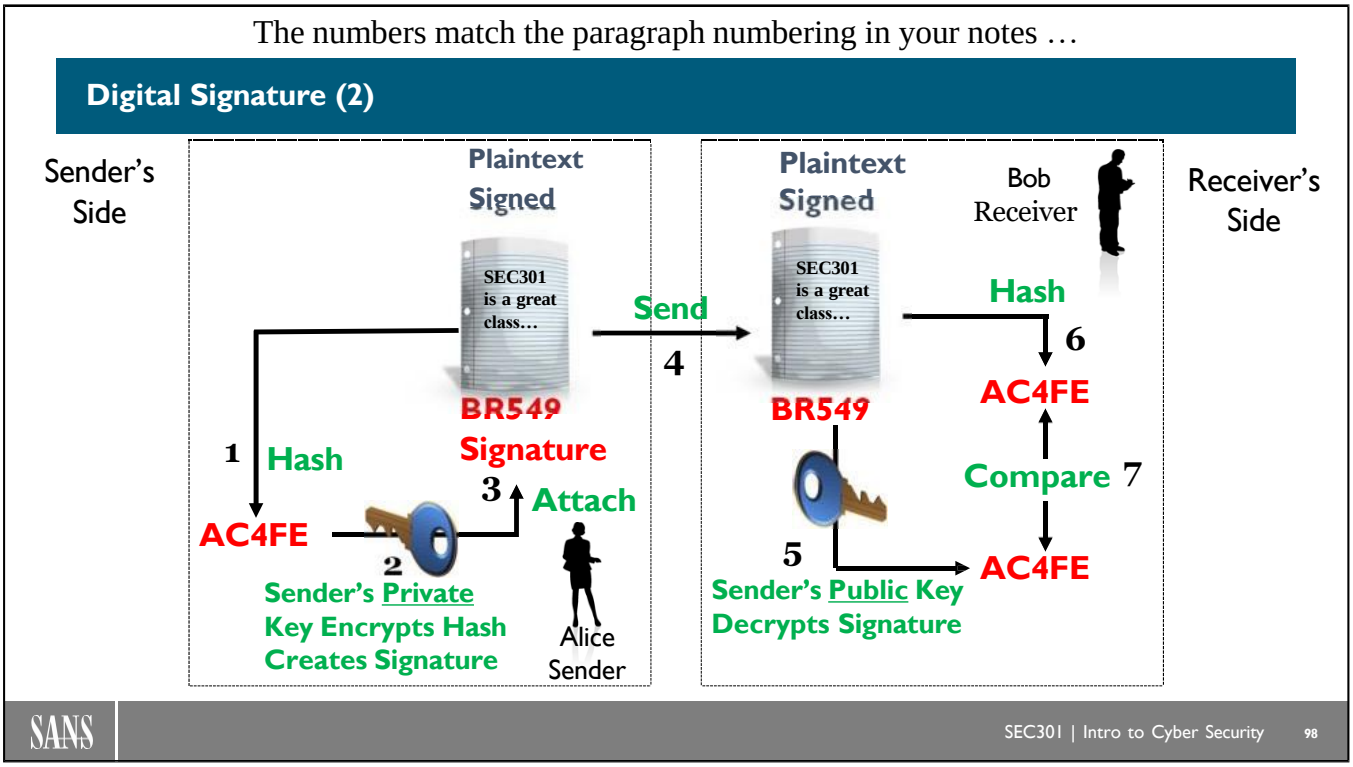
Digital Signature is a generic term to describe a process. The purpose of that process is twofold:

1. Verify exactly ***who sent*** you a message or document.
2. Verify the message or document you received is ***exactly identical*** to the one that was sent.

Let's go through the detailed steps necessary to accomplish this on both the sender and receiver end of the communication.

Note: In all scenarios that follow, there is an assumption that Alice and Bob have, at some time in the past, exchanged their PUBLIC keys with each other using some mechanism. There is also an assumption that each is certain the public key in their possession really does belong to the other.

- Alice has her private key and Bob's public key.
- Bob has his private key and Alice's public key.



Digital Signature (2)

Once again, Alice is going to talk to Bob. At this time, they do not care about confidentiality. They are aiming only to attain integrity. Specifically, they want to prove that Alice sent the message and they want to prove the message Bob receives is the same as the one Alice sends. It will help to understand this example if you know that a digital signature is actually an encrypted hash of the message or file being sent. The hash of the message is encrypted with the sender's private key.

Alice opens Outlook and types a message. She clicks a little button in Outlook that says Sign and then clicks the button that says Send. The process that follows happens behind the scenes.

1. The software generates a one-way hash of the plaintext file.
2. The hash is encrypted with Alice's (the sender's) *private* key. This creates the actual digital signature.
3. The digital signature (encrypted hash) is attached to the message.
4. The message and attached signature transmit to Bob.
5. When Bob opens the message, the software detaches the signature and decrypts it. Because it was encrypted with the sender's private key, only the sender's public key can decrypt it.
6. The software generates a second one-way hash of the plaintext message.
7. The new hash and the one retrieved by decrypting the signature are compared.

If those hashes match, then Bob knows two facts.

- **Fact one:** Because Alice's public key was capable of decrypting the signature, and *only* Alice possesses the mathematically associated private key, then *only* Alice could have generated that signature. To say it another way, because Alice's public key was able to decrypt the signature, it authenticates to Bob that only Alice could have sent the message.

- **Fact two:** Because the hash from the signature and the new hash match in step 7, the message Bob received, and the message Alice sent are exactly identical. In fact, every binary bit is the same.

Digital Signatures and Nonrepudiation

- A proper digital signature results in nonrepudiation:
  - The sender cannot repudiate or deny having sent the document
  - This can stand up in court, just as a pen-and-ink signature can
- To prove nonrepudiation, the receiver must prove **two** f acts:
  - The person he thinks sent the document did in fact send it
  - He received exactly the same document the sender sent
- If the sender's public key can decrypt the signature, this a uthenticates that it came from the sender:
  - See Certificate Authority discussion later for how to obtain proof
- If the hash in the signature and the hash generated by the receiver match, then what was received is what was sent
- Properly completed digital signatures may be legally recognized in 68 countries

Digital Signature and Nonrepudiation

Earlier, you saw the term nonrepudiation. At that time, we did not have enough explanations in place to make that term plain. Now we do.

We need to begin by knowing the definition of *repudiate*, which is *to deny the truth or validity of.* Now keep the digital signature example from the last slide firmly in mind.

Let's say that the message Alice sent Bob said she would pay him 10 million dollars. Six months later, Alice repudiates that message, saying that what she sent actually said she would pay Bob only $10. As you might expect, Bob is less than pleased with this development!

If Bob has a properly completed digital signature, he can take the message (and Alice) into court and try to prove nonrepudiation. To do so, he must convince a judge of two facts:

- **Fact one:** Alice sent the message. Nobody else could have.
- **Fact two:** The message he received is identical to the message Alice sent.

If Bob can convince the judge of *both* facts, then the judge can bring the gavel down and say, "Alice, pay up." Alice can no longer repudiate having sent the message Bob received because he has proven that she did. If repudiate means to deny, then nonrepudiation means the inability to deny.

In the United States, Section 15 U.S. Code § 7001 and the Uniform Electronic Transactions Act give digital signatures the same weight under the law as a pen-and-ink signature. Several other countries have laws saying something similar. Perhaps the best list of those is at: http://en.wikipedia.org/wiki/Digital_signatures_and_law

Hybrid Cryptography

- Extremely common cryptographic process
- Proper implementation gives you:
  - The speed of symmetric key
  - Ease of use of asymmetric key
  - Security of sending ciphertext
  - Protection of the symmetric key in transmission

Hybrid Cryptography

Digital signatures provide for nonrepudiation. They do not provide for confidentiality. So how can we obtain the confidentiality piece of the equation? A common answer to that question today is called *hybrid cryptography*. The name comes from the fact that we are going to implement both symmetric and asymmetric cryptography at the same time.

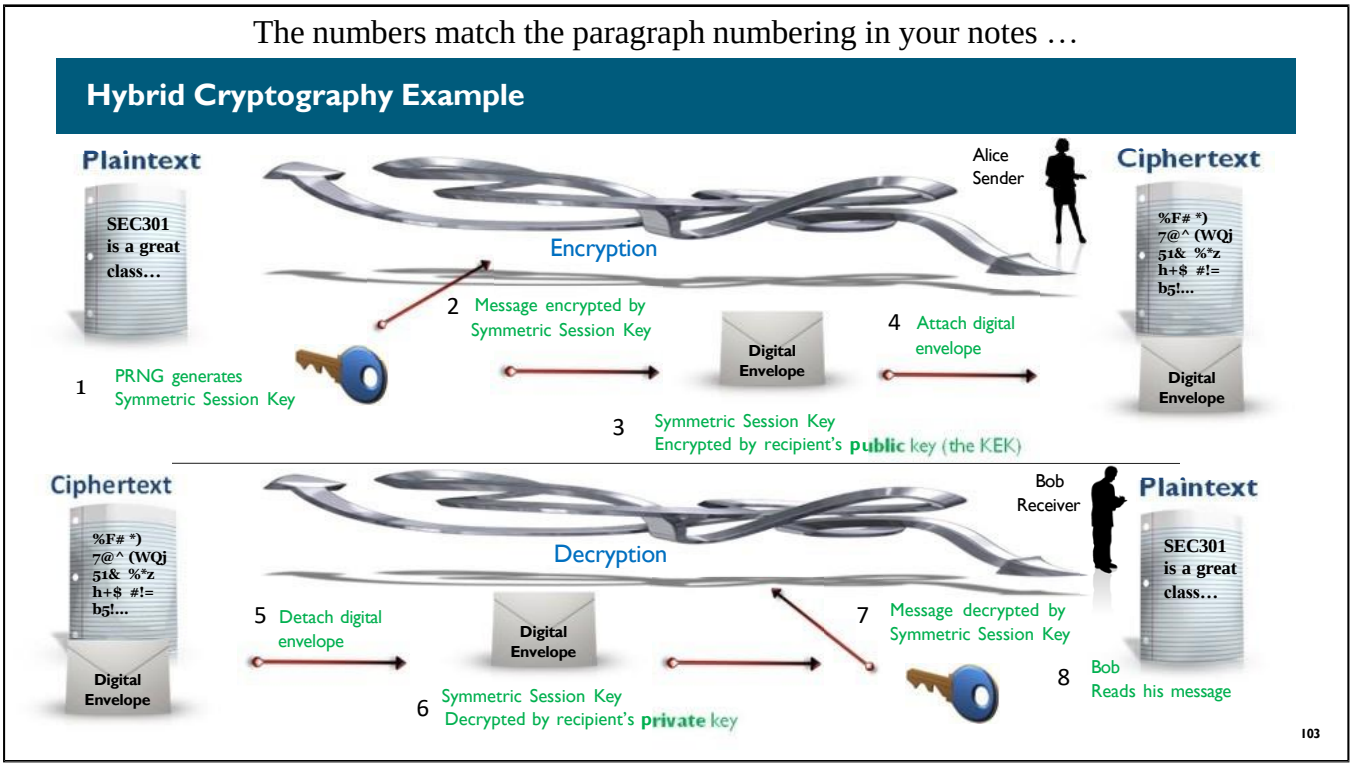
If we do this, we can get the best of both worlds:

- The speed of symmetric key for encryption.
- The ease of use of asymmetric key.
- The security of sending ciphertext.
- BUT we must protect the symmetric key throughout the process.

There are some terms we need to understand. We have already discussed *PRNG* thoroughly.

- **KEK (Key Encryption Key):** When you use a crypto key to encrypt a crypto key, so it can be securely transmitted.
- **Digital Envelope:** The encrypted form of the symmetric key (encrypted by the KEK above).
- **Session Key:** The key is generated and used for one purpose and one purpose only. In this example, it is to encrypt a single email message. The term also applies to a key used to encrypt a single web page, and so on.

The numbers match the paragraph numbering in your notes …



Hybrid Cryptography Example

In this example, Alice and Bob are concerned with confidentiality but not with integrity. Again, Alice opens Outlook and types an email message addressed to Bob. This time, she clicks the button that says Encrypt, followed by the button that says Send. This is what happens in the background:

1. A PRNG generates a symmetric session key. (Let's say it is an AES 256-bit key.)
2. The symmetric session key encrypts the plaintext message, generating the ciphertext version.
3. The symmetric session key is encrypted by the recipient's (Bob's) public key. This creates the digital envelope, which is just the encrypted form of the symmetric session key.
4. The digital envelope is attached to the ciphertext message.

The package transmits to Bob. When Bob opens the message, this is what happens in the background:

5. The digital envelope is detached from the ciphertext message.
6. The digital envelope was encrypted with Bob's public key, so his private key decrypts it. This gets the symmetric session key back in usable plaintext form.
7. The symmetric session key decrypts the ciphertext, giving Bob the plaintext version of the message.
8. Bob reads the message.

Properly implemented, Alice and Bob have obtained the best of both worlds: Symmetric and asymmetric. The speed of symmetric, the ease of use of asymmetric, the symmetric key was protected for transmission, and so on. This is extremely common in today's world. Virtually all email encryption works this way. Whole disk encryption systems work this way. Many other crypto implementations use hybrid encryption.

You also see here the most common BAD implementation of cryptography on this slide. It is common for people to say, "I want really good security, so I'll use an AES 256-bit key" for the symmetric session key. They then choose a 1,024-bit asymmetric key to encrypt the digital envelope. This becomes the weakest link in the chain. Remember that asymmetric keys have to be longer to provide the same level of security. If you use a 1,024-bit asymmetric key, that is equivalent to an 80-bit symmetric key. If you truly want/need the security of AES 256-bit encryption, then the asymmetric key (the KEK) would need to be 15,360 bits long. We have never seen hybrid encryption implemented with an asymmetric key nearly that long.

Author's Note: This is not intended to say that an implementation with a 1,024-bit asymmetric key will be broken in minutes or hours or even days. But it WILL be broken more quickly than the 256-bit symmetric key. According to https://www.keylength.com/en/4/

- 1,024-bit asymmetric is equivalent to 80-bit symmetric
- 2,048-bit asymmetric is equivalent to 112-bit symmetric
- 3,072-bit asymmetric is equivalent to 128-bit symmetric
- 7,680-bit asymmetric is equivalent to 192-bit symmetric
- 15,360-bit asymmetric is equivalent to 256-bit symmetric

(The equivalents above do not apply to Elliptical Curve Asymmetric Cryptography discussed later.)

Also note that this is not a failing of hybrid encryption. It is a failing of those doing the implementation to fully understand all the issues. Without proper implementation, the best security mechanism in the world will do you little or no good.

Signcryption

- Hybrid cryptography is great:
  - But what if we want even more?
- How can we get the following all at once?
  - Nonrepudiation of a digital signature
  - Confidentiality of hybrid cryptography
  - Protection of the symmetric key
- We want it ALL!
  - And Signcryption is the answer

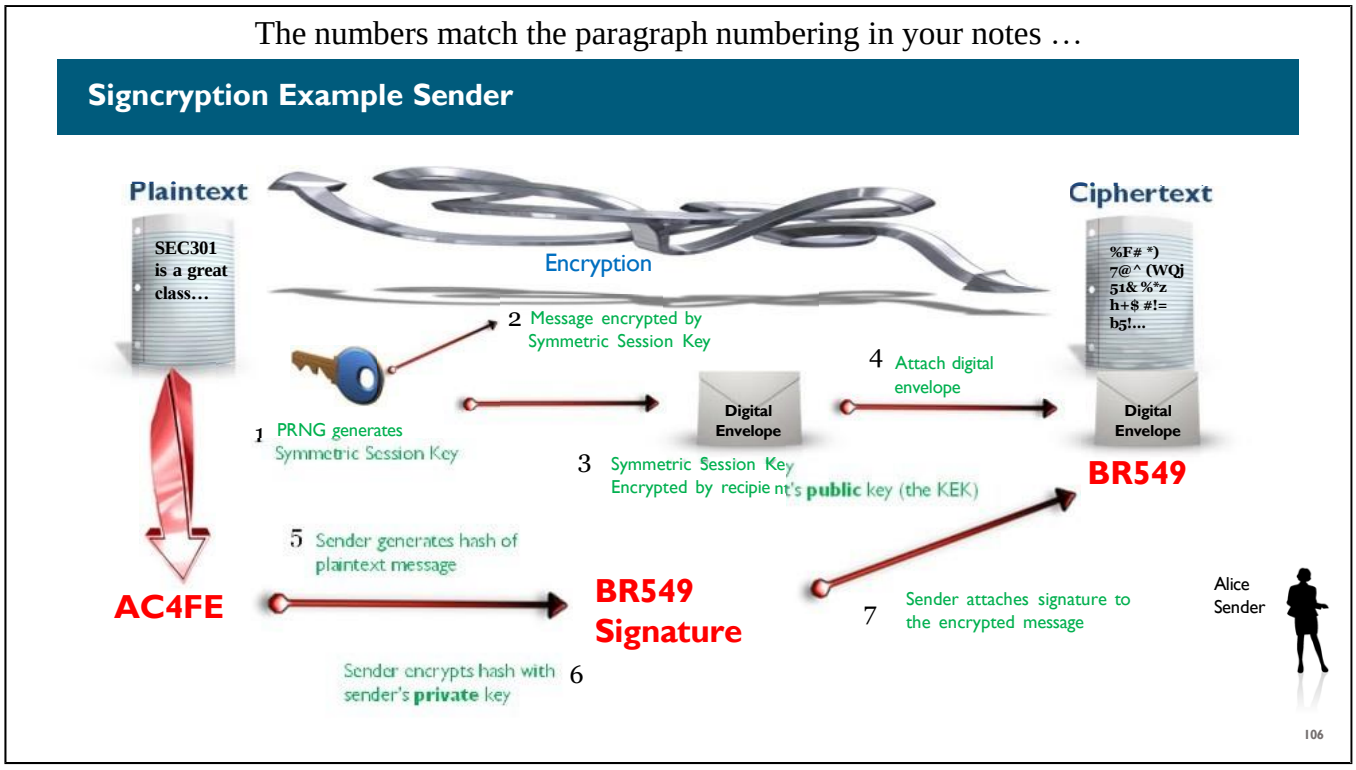
Signcryption
Hybrid encryption is great, but what if we are greedy and want even more? What if we want:

- The nonrepudiation of the digital signature
- The confidentiality of hybrid cryptography
- The speed of symmetric key
- The ease of use of asymmetric key
- Protection of the symmetric key during transmission

We want it ALL!

Signcryption is the answer.

The numbers match the paragraph numbering in your notes …



Signcryption Example Sender

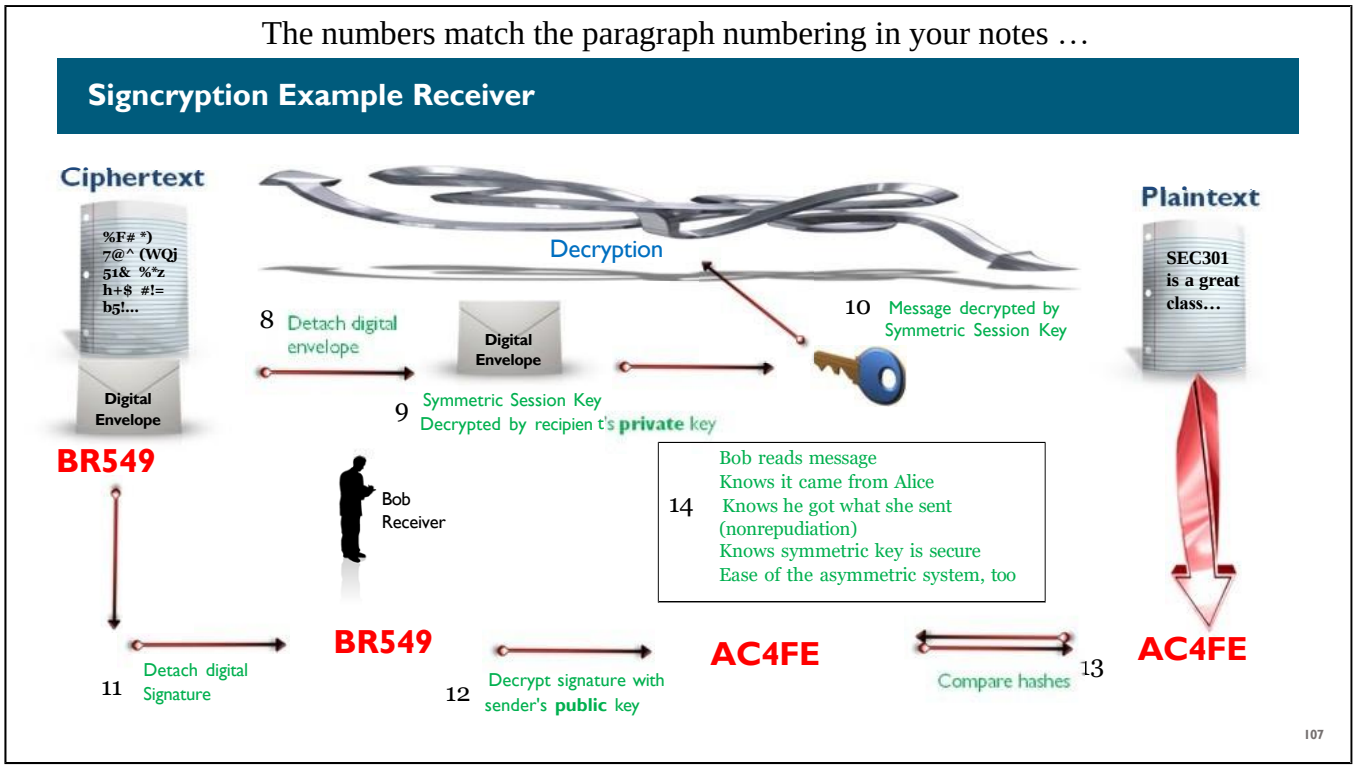
There are a few extra steps involved in signcryption, so this will take two slides to explain; but, we will continue numbering the steps onto the next slide. The last step here is 7; the first step on the next slide will be 8.

When Alice and Bob want both confidentiality and integrity (nonrepudiation), this is how they can obtain it. Once again, Alice opens Outlook and types a message addressed to Bob. But this time, she clicks both the button that says Sign and the button that says Encrypt. Then, she clicks Send, and this is what happens in the background:

1. The PRNG generates a symmetric session key.
2. The symmetric session key encrypts the plaintext, generating the ciphertext.
3. The symmetric session key is encrypted with the recipient's (Bob's) public key to create the digital envelope.
4. The digital envelope is attached to the ciphertext message.
   (Up to this point, the process is identical to the top one-half of the hybrid encryption slide.)
5. A one-way hash algorithm generates a hash of the plaintext message.
6. The hash is encrypted with the sender's (Alice's) private key to create the digital signature.
7. The digital signature is attached to the ciphertext message.

The combined package of the ciphertext message, the digital envelope, and the digital signature are transmitted to Bob …

The numbers match the paragraph numbering in your notes …



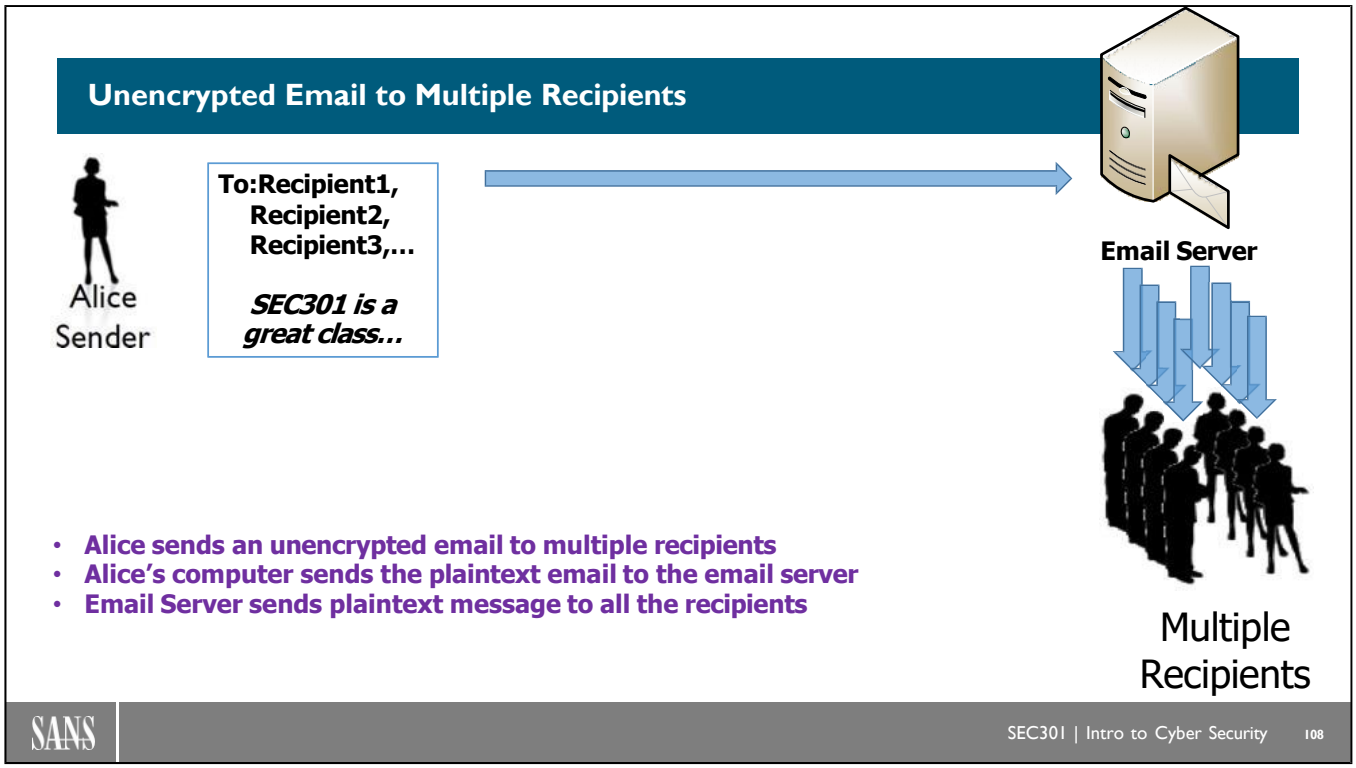
Signcryption Example Receiver

… and Bob receives the package containing the ciphertext message, the digital envelope, and the digital signature.

When Bob goes to open the message, this is what happens in the background:

8. The software detaches the digital envelope.
9. The digital envelope was encrypted with Bob's public key, so his private key decrypts it. This returns the symmetric session key to a usable state.
10. The symmetric session key decrypts the ciphertext, regenerating the original plaintext.
11. The digital signature is detached by the software.
12. The digital signature was encrypted by the sender's (Alice's) private key, so only the sender's public key can decrypt it. The digital signature is decrypted, returning the original hash (created in step 5 on the last slide) to plaintext.
13. Bob's software generates a second hash using the same algorithm. The new hash and the original hash are compared. If they match, the message opens on Bob's screen without errors.
14. Bob reads the message. He knows it came from Alice and that he got exactly what she sent (nonrepudiation). He knows the symmetric key was protected throughout the process. And he appreciates the ease of use of the asymmetric key as well.

In reality, Outlook implements this process, so it is transparent to the user. Bob may not actually know any of that. But now you know how it works and knowing is one-half the battle.

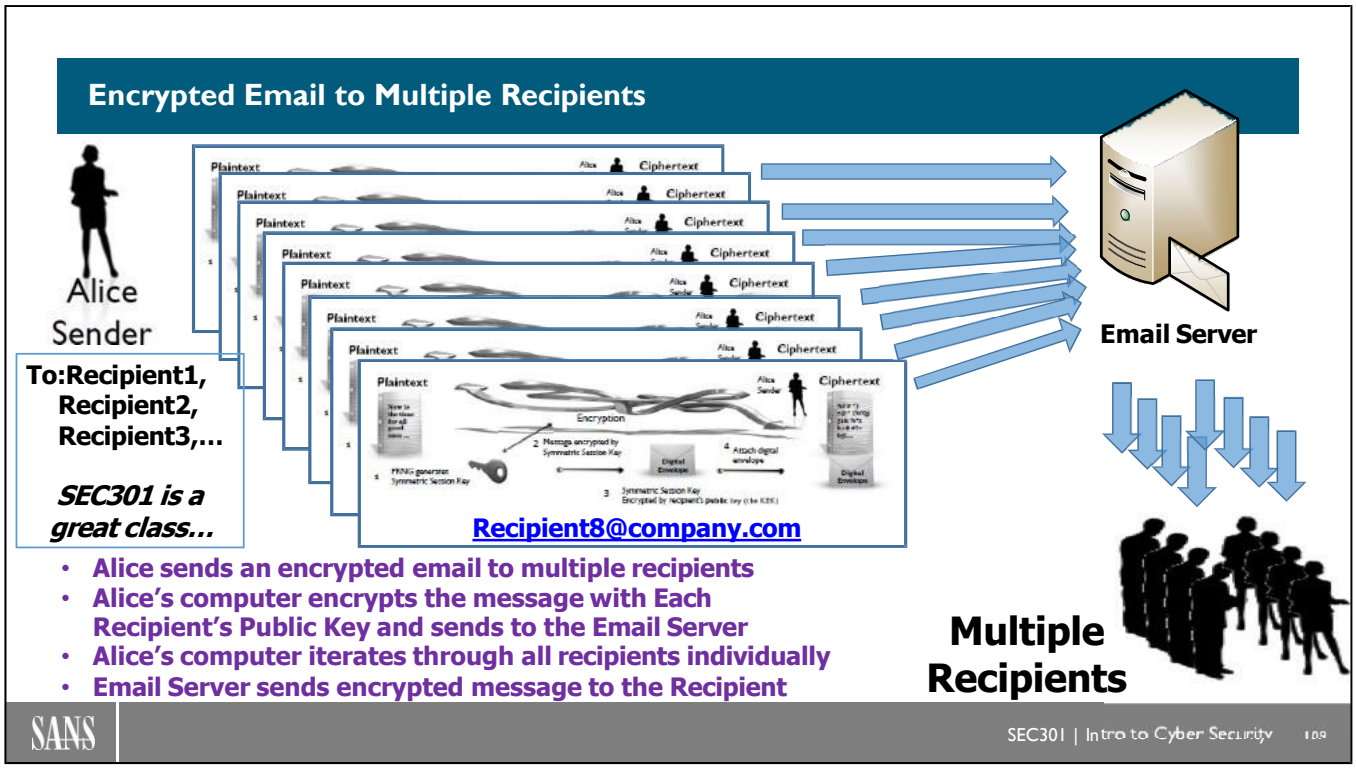


Unencrypted Email to Multiple Recipients

A common question from those who have used email encryption is; "Why is it so slow when I send encrypted email, especially to multiple people?"

To answer this question, let's first look at how an unencrypted email transmits to multiple recipients. Alice types her email and puts multiple recipients in the "To:" address line. When she clicks on send, her computer connects to the email server and the email transmits to that server. The email server distributes the email to all recipients.

Note that in this case, the email transmits from Alice's computer to the mail server one time and the mail server distributes it to all recipients. This is important to remember when we move to our next example.



Encrypted Email to Multiple Recipients

When Alice sends encrypted email to multiple recipients, the process has to change. The reason for this change is one of the most important facts to always remember about cryptography:

Encryption and decryption can only occur on the system(s) where the keys reside.

In this case, the email server does not have the encryption/decryption keys. Those reside on Alice's computer and on each recipient's computer. Therefore,

- Alice's computer must encrypt the message (along with any attachments) with recipient1's key, send it to the server and the server sends the email to recipient1.
- Alice's computer must encrypt the message (along with any attachments) with recipient1's key, send it to the server and the server sends the email to recipient2.
- Alice's computer must encrypt the message (along with any attachments) with recipient1's key, send it to the server and the server sends the email to recipient3
- This continues until all recipients have received the email encrypted via their own key.

Of course, this all depends on Alice already having the Public keys of all recipients in her computer's configuration. The key exchange had to happen before any of this could occur.

Digital Certificates (X.509)

- A digital certificate (or just certificate) is an electronic document:
  - Analogies include passport, driver's license, and so on
  - They are issued by a Certificate Authority (CA)
- They always have two parts:
  - The private portion contains the private key
  - The public portion contains (among other things)
    - Your public key
    - Your name
    - A validity period (from date X to date Y)
    - Who issued it (the Certificate Authority)
    - Digital signature that can prove the certificate's validity

Digital Certificates (X.509)

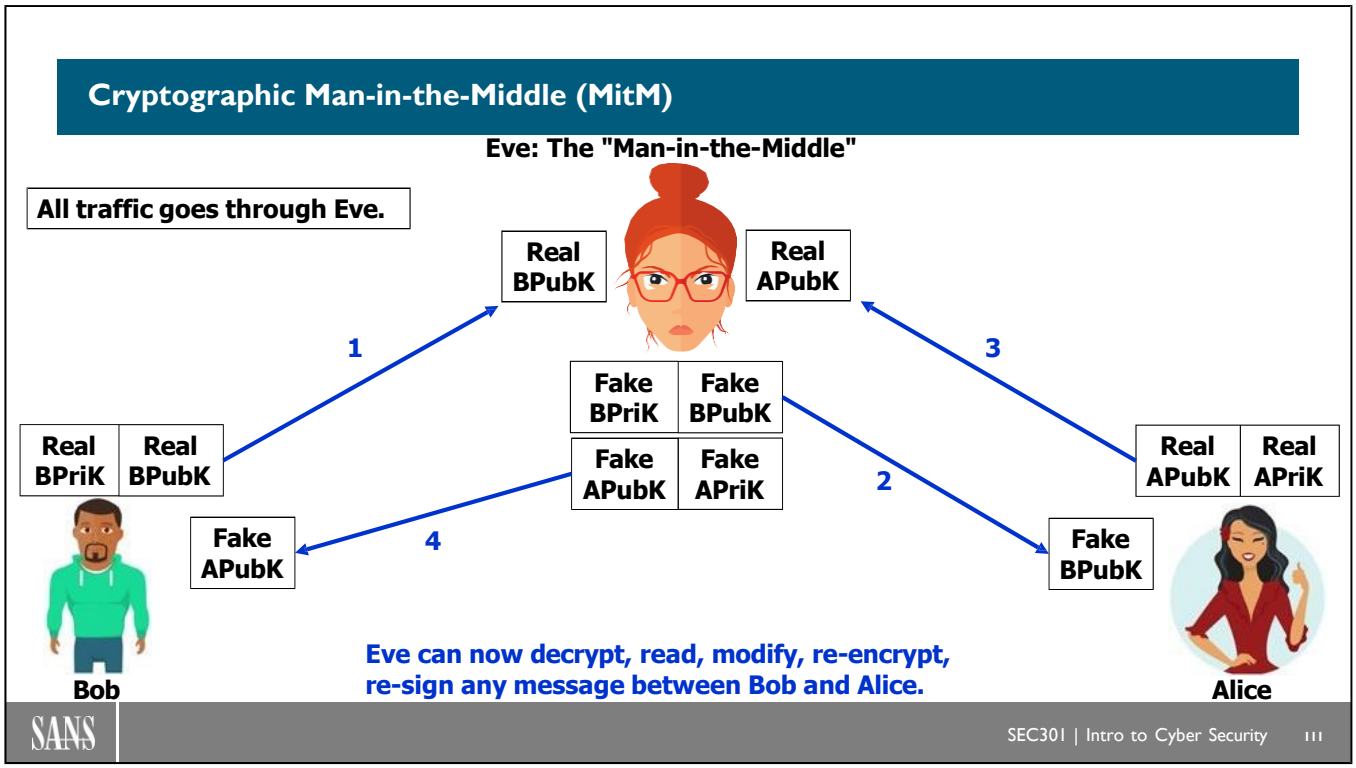
Perhaps the best way to describe a digital certificate is to call it an "electronic document." There are several analogies comparing it to a passport or a driver's license and other similar documents.

It is an electronic document containing all the following:

- Your public key
- Your name
- A validity period: The certificate is good from date X to date Y
- Who issued it: Known as the Certificate Authority
- The digital signature of the entire certificate: Allows the Certificate Authority to verify the certificate is valid

(There are more fields in a certificate. You will see the full list shortly.)

Note that in conversation, it is common for people to simply call these "certificates" instead of the more correct "digital certificates."

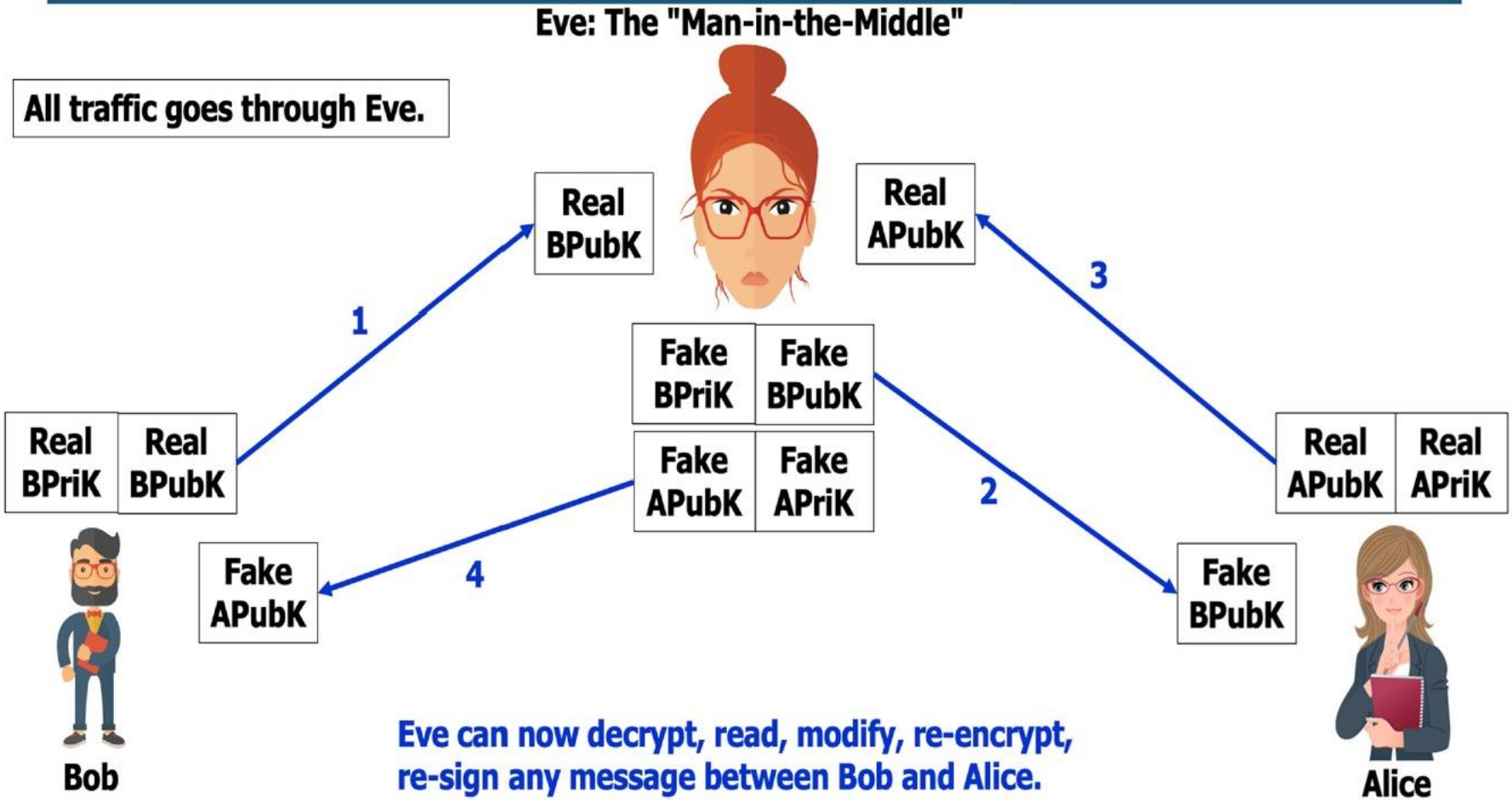


Cryptographic Man-in-the-Middle (MitM)
The Cryptographic Man-in-the-Middle attack occurs when someone has created a situation in which all your data goes through the attacker (Eve).

- When Bob sends his Real Public Key (Real BPubK) to Alice, Eve intercepts it and sends a Fake of Bob's public key (Fake BPubK) on to Alice.
- When Alice sends her Real Public Key (Real APubK) to Bob, Eve intercepts it and sends a Fake of Alice's public key (Fake APubK) on to Bob.
- Notice: Eve has the mathematically associated private key for each of the public keys forwarded to Alice and Bob. Specifically, Eve possesses Fake BPriK and Fake APriK since she created those key-pairs. Of course, she also has Real BPubK and Real APubK since she just intercepted them.

Eve now has all the keys necessary to really mess with Bob and Alice. Say, for example, Alice sends an encrypted and signed message to Bob. Eve intercepts the message. It was encrypted with Bob's Fake Public key (since that is the public key Alice thinks belongs to Bob), and Eve has the Fake Private key to decrypt and read the message. Indeed, Eve can even modify the message if she likes. Then, she can re-encrypt the message with Bob's Real Public Key and re-sign it with Fake APriK and send the message on to him. If Eve is careful, this attack is virtually undetectable.

Cryptographic Man-in-the-Middle (MitM)



Public Key Infrastructure (PKI)

- A framework that provides for the creation, distribution, and management of digital certificates (and therefore keys):
  - ISO Standard: Does not specify protocols to use
  - Is therefore a framework for technology, not itself a technology
- Note: It is a Framework; it is NOT a protocol:
  - Therefore, open to interpretation of implementation
  - Results in competing "standards"
- Ideally transparent to the end user:
  - The end user simply says "encrypt and sign this message"
  - The PKI software takes care of key exchange and other back-end processes

Public Key Infrastructure (PKI)
PKI is a framework that provides for the creation, distribution, and management of digital certificates (and therefore the keys they contain).

The term *framework* is important here. Take the IP protocol. As a protocol, it tells a device such as a router precisely where to find the destination IP address, how it is formatted, and more. By contrast, if IP were a framework, it would simply say, "There needs to be a destination addressing capability. However you accomplish that is fine, so long as it is there."

In other words, as a framework instead of a protocol, PKI is open to widely varying interpretations. It is extremely rare for two vendors' systems to be interoperable.

This fact, perhaps more than any other, has slowed the adoption of PKI.

With proper implementation, PKI is transparent to the end-user. The user simply says, "encrypt and sign this message," and the PKI software takes care of any necessary key exchanges and so on. The end-user does not even have to know that cryptographic keys exist. Indeed, in some implementations, the user does not even have to say, "encrypt and sign this message," since even that can be automated.

Module 10: Data Encrypting Protocols

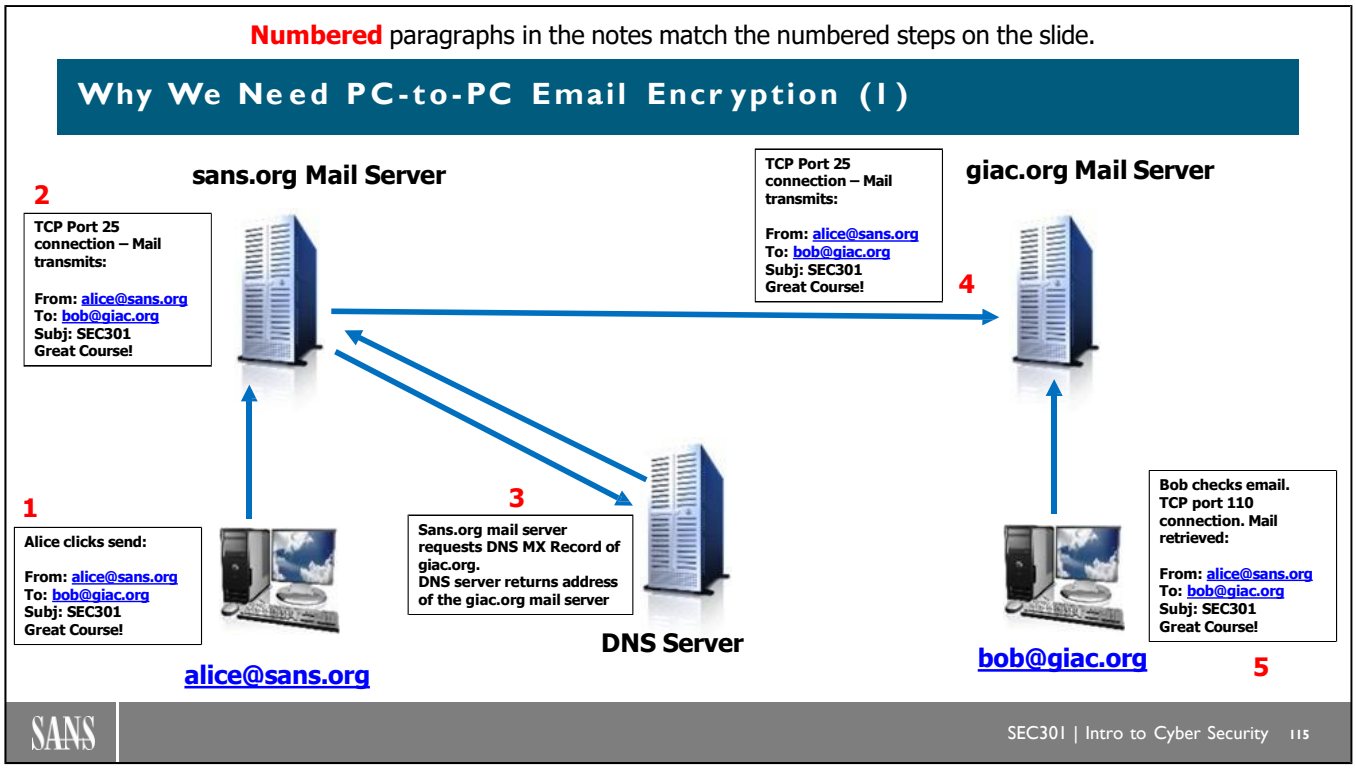
- S/MIME
- PGP
- SSH
- SSL and TLS
- HTTPS
- FTPS, SFTP, and SCP
- IPSec

COURSE ROADMAP

- Module 8: Intro to Crypto
  - Lab 3.1: Crypto by Hand
- Module 9: Building Blocks of Modern Crypto
  - Lab 3.2: Visual Crypto
- **Module 10: Data Encrypting Protocols**

Module 10: Data Encrypting Protocols

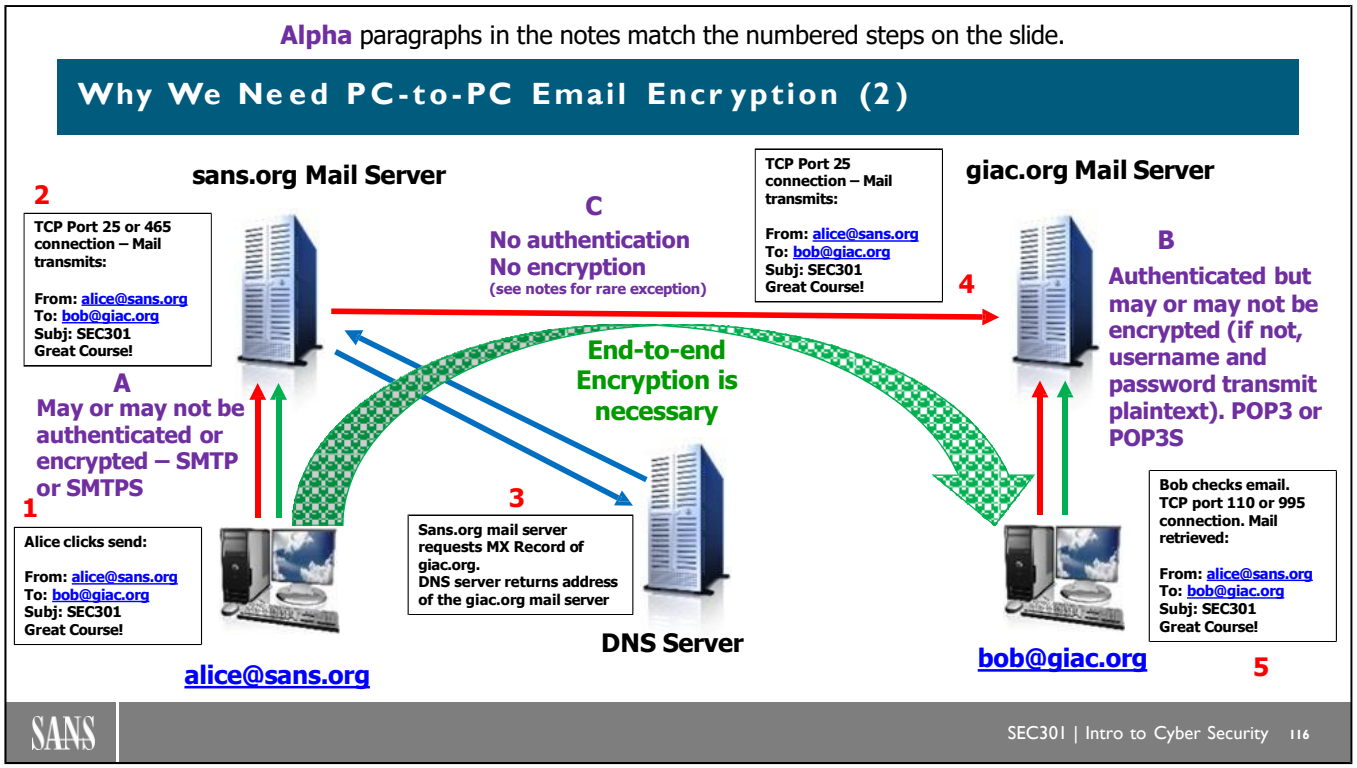
This page intentionally left blank.



Why We Need PC-to-PC Email Encryption (1)

Above, we see the basics of how email transfers across the internet from one internet domain to another. In this example, Alice at SANS.org sends an email to Bob at GIAC.org. Note there can be some variations on the above. For example, there are a few different protocols we can use in the final step that would change the port number used. Here, we have a generic example.

1. In the first step, Alice has typed an email. It is from her address (alice@sans.org) to Bob's address (bob@giac.org). Alice clicks Send in her email program. (Note that Alice's email program has the IP address of the sans.org email server in its configuration.)
2. Alice's computer connects to the sans.org email server with a TCP Port 25 connection. TCP Port 25 is for the service "Simple Mail Transfer Protocol." The email transfers to the mail server.
3. The sans.org email server looks at the "To:" address of bob@giac.org and sends a query to its DNS server asking for the MX record of giac.org (whatever comes after the @ in the email address). In DNS, the MX record for each domain returns the IP address of that domain's mail server. Therefore, the DNS server returns the IP address of the giac.org mail server in this case.
4. Once the sans.org mail server has the IP address of the giac.org mail server, it establishes a TCP Port 25 connection with that mail server, and the email transfers.
5. Once the email is on the giac.org mail server, it sits there patiently until Bob checks his email. When Bob does check email, his email software (which has the IP address of the giac.org mail server in its configuration) connects to the giac.org mail server on TCP Port 110 and retrieves the email message. (TCP Port 110 is Post Office Protocol 3, or POP3.)



Why We Need PC-to-PC Email Encryption (2)

Now that we understand how email transmits, we need to know when authentication or encryption is applied.

A. When the email transmits between the sending PC and (in this case) the sans.org mail server, there may or may not be authentication and encryption. When the Simple Mail Transfer Protocol (SMTP) first came out years ago, this connection had no authentication or encryption. Today, many organizations implement a newer version of SMTP that offers password authentication and SSL/TLS-based encryption (which we will discuss shortly).

B. Likewise, when the POP3 protocol first came out, there was no encryption. The protocol required password authentication, but the username and password passed across the wire in plaintext. (This was also true of Internet Message Access Protocol [IMAP] that came along later.) Again, today there are new versions of both POP3 and IMAP that encrypt the data, including the password, using SSL/TLS encryption.

C. The server-to-server connection is almost always unauthenticated and unencrypted. Think it through and you will realize it has to be this way. Authentication would require every mail server to know a username/password on every other mail server in the world. Encryption would need every mail server to have the encryption keys of every mail server in the world. Both would be utterly impossible to implement. (Note: In some rare and limited instances, you may find an authenticated and encrypted server-to-server communication. It is a relatively new idea, but partner companies can exchange cryptographic certificates and set their servers up to authenticate and encrypt to each other. But even with that in place, the security only happens between those two servers.)

S/MIME (Secure Multipurpose Internet Mail Extensions)

- MIME: Transforms all email and attachments to 100% ASCII text for transition

- S/MIME: Lets MIME transform to ASCII
  - Then encrypts the entire block of ASCII using hybrid cryptography
  - Meaning that both email text and attachments are encrypted
- S/MIME is built into Outlook and most other popular email clients
  - Requires the use of digital certificates to contain the keys

S/MIME (Secure Multipurpose Internet Mail Extensions)

MIME, or Multipurpose Internet Mail Extensions, is a utility that is used in all internet email. Its purpose is actually quite simple. All email transmits as 100% ASCII text—including non-ASCII attachments. MIME is the protocol that translates those attachments into ASCII text when sending, and back into attachments on the receiving end.

For example, let's say you have typed a paragraph in an email, and that email also has a graphic attached in JPG format. JPG is *not* a purely ASCII format, as discussed earlier in the class. When you send email, MIME takes the paragraph you typed and the JPG image and translates them to a very long string of ASCII text (the image does indeed become an extremely large amount of ASCII). The message then transmits as ASCII text, and, on the receiving end, MIME translates the long string of ASCII back into the original message and attachment.

S/MIME simply allows MIME to do its thing and translate everything into ASCII, then it uses hybrid cryptography (described earlier in the course) to encrypt the entire block of ASCII. Of course, on the receiving end, S/MIME decrypts everything and passes it back to MIME for translation back to its original form.

S/MIME is implemented in Microsoft Outlook as well as in most other popular email clients. It almost always requires digital certificates to contain and manage the keys (the author cannot find any exceptions to this).

GNU stands for "GNU Not Unix" and means open-source, free software.

Email Encryption: PGP/GPG

- Released by Phil Zimmerman in 1991:
  - Encrypts and digitally signs email and files/folders and whole disk
  - Originally utilized PGP keys (and still can)
  - Now supports certificates:
    - Not true X.509 certificates; they have additional fields
- First significant use of both Hybrid crypto and Signcryption
- Now available in both freeware (GNU PGP or GPG) and commercial PGP. Now owned by Symantec:
  - GPG is significantly harder to use than S/MIME or commercial PGP
- OSI Layer 7: Only encrypts data

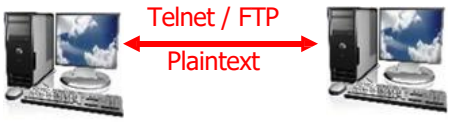
Email Encryption: PGP/GPG

PGP was originally released by Phil Zimmerman in 1991. It was the first important email encryption capability. Originally, you utilized PGP keys when using this software. Those keys were stored on your hard drive in a "public key ring" and a "private key ring," which were just files on your system containing the keys. Although the capability is still there to use PGP keys, the commercial implementation now uses certificates more commonly.

There is the commercial PGP, now owned by Symantec. There is also the public domain version under GNU licensing (commonly called GPG). Using the freeware version requires a little more technical knowledge on the part of the user, but it is usable.

Remote Access and File Transfer: Secure Shell (SSH)

- Provides the same functionality as Telnet, FTP, and others:
  - Remote command line configuration
  - File transfer



- The difference: SSH encrypts all the traffic:
  - An encrypted connection between the client and server
  - Utilizes Diffie-Hellman for secure key exchange
- Uses TCP Port 22

- Available in OpenSSH and commercial implementations:
  - http://www.openssh.org/

Remote Access and File Transfer: Secure Shell (SSH)

If you have ever used telnet, you know what it looks and feels like to use Secure Shell (SSH). In other words, it is a command-line prompt on a computer somewhere else on the network. You enter commands on that distant computer and have them execute there.

The difference is that telnet transmits all your commands, their output, your username/password, and everything else across the network in cleartext. Secure Shell encrypts all that information.

As you will see in a moment, SSH also provides other functionality, such as file transfer.

Secure Transfer of Data: SSL and TLS

- Secure Socket Layer (SSL) and Transport Layer Security (TLS):
  - SSL (developed by Netscape) is in version 3 and always will be:
    - Netscape gave it to the IETF for v4, which changed the name to TLS
- Operates at OSI Layer 4 (transport):
  - Can be used with *any* TCP protocol:
    - HTTPS (port 443) is most common
      - There is also FTPS (ports 989 for data and 990 for control), SMTP-SSL (port 465), LDAP-SSL (port 636), and POP3-ssl (port 995)
  - Secure key exchange normally handled by Diffie-Hellman, Elliptical Curve Diffie-Hellman, or RSA

Port numbers are NEVER assigned to transport layer protocols – only application layer protocols.

Secure Transfer of Data: SSL and TLS

Secure Socket Layer (SSL) and Transport Layer Security (TLS) are extremely closely related. In fact, TLS is simply a later version of SSL.

SSL was developed by Netscape, who maintained it until version 3. When it turned maintenance over to the Internet Engineering Task Force (IETF), it improved on SSL, and the IETF renamed it TLS. So, TLS version 1 is actually SSL version 4 (though that is not what it is called).

Because SSL/TLS operate at the transport Layer (OSI Layer 4), you can actually encrypt any TCP-based protocol. Historically, the most common has been HTTP (making it HTTPS), but there is also an FTPS for encrypted file transfer and several others.

One of the most common misconceptions in IT today is that SSL and TLS run on TCP Port 443. That is actually incorrect. SSL and TLS are transport layer protocols; port numbers are assigned only to application layer protocols. HTTPS uses port 443. But if you look at this slide, you can see that FTPS, TelnetS, and several others have their own port numbers assigned to them.

Secure Web: HTTPS

- HTTPS (HTTP Secure): HTTP—SSL or TLS encrypted:
  - Encrypts all data between browser and web server (both directions)
  - Requires configuring server; no editing of HTML is required
  - Key exchange: Diffie-Hellman is most common, but supports RSA
  - An absolute standard

- An OSI Layer 7 application: Only encrypts data:
  - Usually operates on TCP Port 443

Secure Web: HTTPS

Speaking of HTTPS … this is the HTTP protocol encrypted using either SSL or TLS. It does use TCP Port 443.

This is what you use when you open a browser and enter a URL (web address) that looks something like the following:

https://www.sec301.com/

The information in front of the ":" in that address tells the browser to connect to the web server on port 443. Assuming the web server is properly configured, you will encrypt your traffic to and from that server.

Secure File Transfer: FTPS, SFTP, and SCP

- File Transfer Protocol (FTP):
  - Cleartext authentication and transfer of data
- FTPS:
  - SSL/TLS encrypted FTP
  - Ports 989 for data and 990 for control

If the acronym **ends** with "S," it uses SSL/TLS.

If the acronym **begins** with "S," it uses SSH (port 22).

- SFTP:
  - a.k.a. SSH File Transfer Protocol (or Secure File Transfer Protocol)
  - TCP Port 22, just like all SSH
- SCP:
  - File transfer command available in SSH (also on TCP Port 22)
  - So, it is another file transfer option when working with SSH

Secure File Transfer: FTPS, SFTP, and SCP

Just to be aware, several options are available for secure file transfer. If you talk to anyone using any one of these, they will swear that their solution is great, and the others are bad. In reality, they each have their good points and bad points. Assuming proper implementation, any one of them is capable.

IP Security (IPSec)

- Family of protocols originally developed as part of IPv6
- Two main protocols:
  - Authentication Header (AH—rare today) (Protocol 51):
    - Provides authentication of origin, replay protection, integrity
  - Encapsulating Security Payload (ESP—common) (Protocol 50):
    - Provides everything Authentication Header provides
    - Plus, it encrypts traffic
  - Note: These can be used together and often are today

IP Security (IPSec)

IPSec was originally created as part of the IP version 6 protocols. When the adoption of IPv6 started taking longer than anticipated, they tweaked IPSec so that it would work in IP version 4. We have used it to build encrypted virtual private network (VPN) tunnels ever since.

One of the most common ways to enable tunneling is to use a protocol specifically designed to provide the security of IP networks. The native IP protocol specification does not provide for any real security in its transmissions. Previous efforts have tried to add security at the Application Layer, but what is needed is a way to put security at the network layer, so it is transparent to any higher-level protocol. The IP Security Protocol, or *IPSec*, was designed to work directly with the network layer to provide more robust and reliable security features. IPSec protects IP packets by defining a method of specifying the traffic to protect, how to protect the traffic, and to whom the traffic is sent. IPSec can protect traffic between hosts, between network security gateways like routers or firewalls, or between hosts and security gateways. IPSec consists of two different protocols:

The *Authentication Header*, or *AH*, provides proof that a packet came from the computer indicated by the packet's source address. It also contains data integrity protection, which ensures that the data in the packet has not been altered during the transmission. Finally, the AH provides protection against replay attacks, meaning that an attacker cannot capture the packets and then resend them on the network at a later time. When combined, all these features provide for a much greater level of protection than ordinary IP traffic. The only thing the AH does not provide is data confidentiality; it does not encrypt the traffic.

The *Encapsulating Security Payload*, or *ESP*, supplies data confidentiality. The ESP takes the AH one step further by also providing encryption to protect the data in transit from being viewed or being altered.

Virtual Private Networks

- Connecting two private networks over a public medium, such as the internet

- Advantages:
  - Transparent
  - Low-cost alternative to leased lines
    - Though these should still be encrypted!
  - Can be used for internal communications
- Does not protect you from what is coming through the tunnel

Virtual Private Networks

IPSec gives organizations a way to take advantage of their internet connectivity to connect disparate networks and mitigate many of their security concerns at the same time. The way in which they accomplish this is called a virtual private network, or VPN. As its name implies, a VPN is virtual: It appears to the two end networks to be a dedicated link between the two, but in reality, it shares network space with the rest of the internet.

A VPN is private in that it uses encryption and authentication technologies to keep information secret from prying eyes on the internet. It is, however, a true network: Two remote computing nodes connected by a shared line.

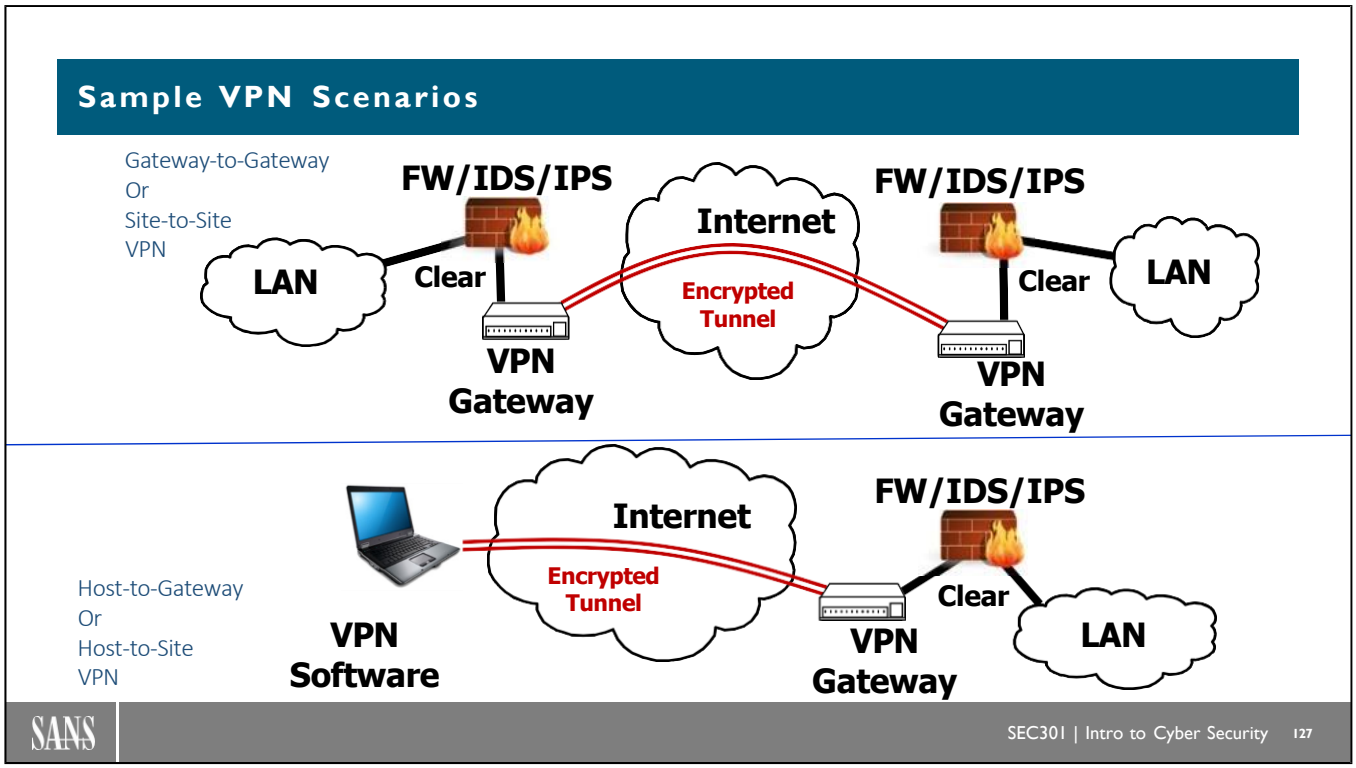
A VPN's biggest advantage is that it provides transparent network security to any application that travels through the VPN tunnel. After the VPN connection is established, any application (email, web browsers, ftp transfers, and IM sessions) can use the tunnel as if the computer is directly connected to a local network. No additional work or programming needs to be done to these applications. VPN is also a great alternative to connecting networks through leased lines. Traditional leased lines can take months to set up and cost thousands of dollars a month to maintain. For a fraction of that cost, a company can use the internet connection they already have and connect to a remote office or business partner in minutes.

A VPN is not just for connecting your network to some external network. Some organizations use VPN within their own internal networks to ensure confidentiality and integrity of highly sensitive information between parts of the network.

There is one thing to remember about VPN. After you put your communications inside the VPN tunnel, it is shielded from sniffing and eavesdropping. This includes the port and protocol analysis typically performed by your firewalls or intrusion detection devices.

A VPN does not protect you from malicious traffic coming through the tunnel. In this case, you might want to consider moving (or adding) your firewall and IDS sensor after the VPN connection is decrypted and before it is put out on your network.

A VPN is an important step in bringing network communications up to speed for security confidentiality, integrity, and availability.



Sample VPN Configurations

VPN has two primary uses. The first is to connect two networks together securely over an insecure middle network. The second is to connect a single user to a network securely, again over an insecure network. Let's examine each application separately.

The top picture in the slide is a diagram of two Local Area Networks (or LANs), one on each end of the chain. These two networks might be two remote offices of one company, or they might be networks in separate companies that, for some reason, need to communicate with each other. To communicate securely over the internet, the owners of these networks decide to use VPN. First, they put a VPN gateway in front of each network. These gateways encrypt the network traffic that comes out of the local network, which is normally transmitted in the clear without encryption. When the traffic reaches the remote network, the VPN gateway on that side decrypts the traffic and sends it to its destination.

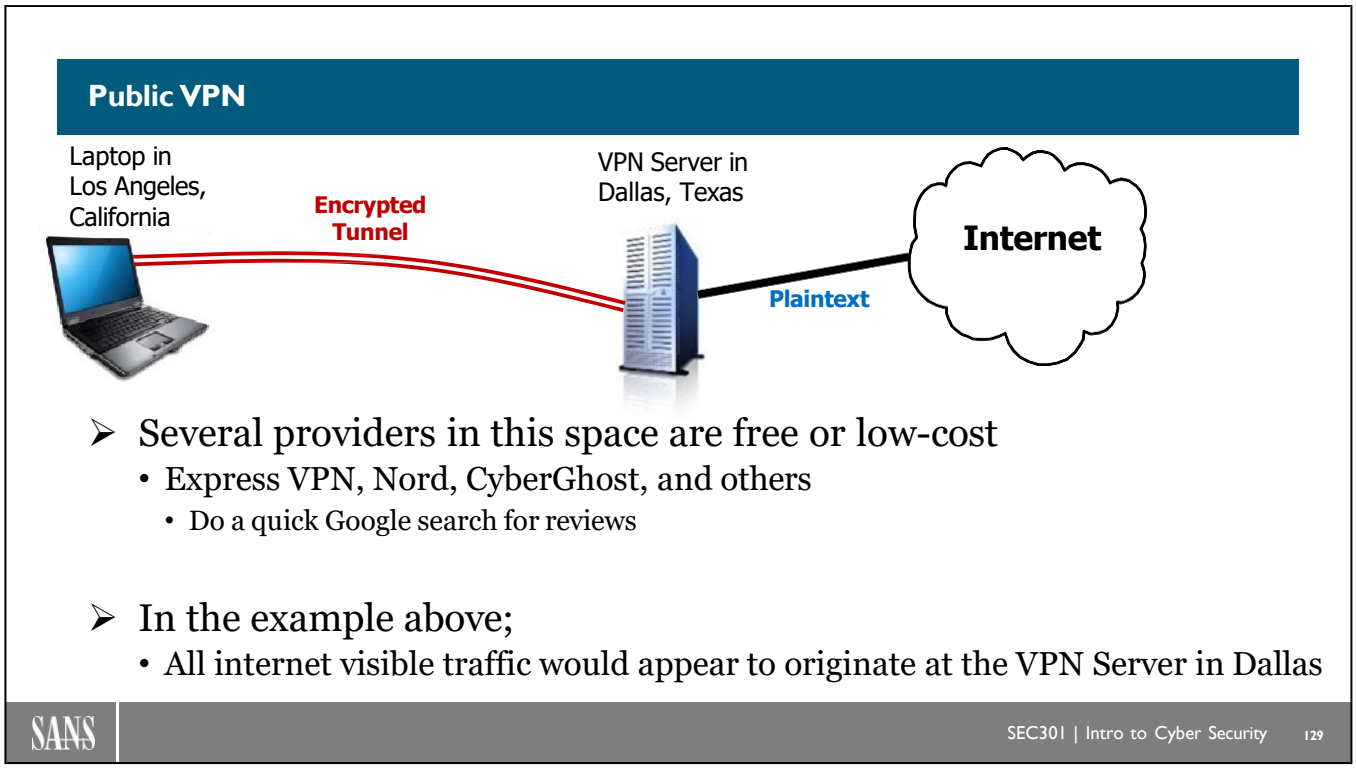
An essential element in this use of VPN is the use of a firewall between the local networks and the internet. As we have discussed previously, a firewall is mandatory whenever connecting to the internet. The firewall often has the VPN server software built in, making it even easier to implement VPN over the internet. Where you place the firewall in relation to the VPN server is a matter of choice, but here are a couple of points to consider when making your decision. If you place the VPN server behind the firewall (between the firewall and the internal network), you will effectively "blind" your firewall as well as any other security mechanisms, such as an IDS or an IPS. With IPSec VPN using Encapsulating Security Payload in Tunnel Mode, the IP header, the transport header (such as TCP or UDP headers), and the data are all encrypted.

If you pass the encrypted form of the packet through a firewall or other security device, everything they might try to make a decision on is encrypted and can't be seen.

This is why the diagram shows the traffic encryption point occurring where it does. The firewall, IDS, and IPS technologies see cleartext traffic and can analyze it.

The second diagram shows a user's laptop connecting to the corporate network over the internet. This scenario typically applies to telecommuting or traveling situations. Business people on the road need to access their corporate networks to work. The cheapest way to do this is for the traveler to connect to the internet through a local ISP, or perhaps a hotel's internet connection, and use the internet to connect to the corporate network. When doing this, of course, the traffic must be protected via encryption as it traverses the hotel network and the internet.

In this scenario, the user has VPN client software built in to their computer. The client software has the same function as the VPN gateway in the top diagram; it encrypts all traffic leaving the computer and sends it to the VPN gateway on the corporate network. When VPN traffic returns to the computer, it decrypts the transmission and passes it into the computer for processing. Again, a firewall is used to protect the corporate network from the perils of the internet and is placed such that it is seeing the traffic in cleartext.



Public VPN

Another type of VPN is sometimes referred to as Public VPN. With this type of VPN, you utilize a service and can connect to one of their servers. You establish an encrypted tunnel from your device to the VPN server. At that point, your traffic is decrypted and sent out to the internet.

Above, you see a laptop in Los Angeles, California with a VPN tunnel to Dallas, Texas. All traffic from the laptop to the server in Dallas is encrypted. Traffic from the Dallas server to the internet is unencrypted. Note: If you are using one of the encrypting protocols discussed in this module, such as HTTPS, then that encryption would remain on the data once it passes the Dallas server and on out to the internet.

In this scenario, all traffic from the laptop will be seen on the internet as coming from the Dallas server. In fact, if you have the setup depicted in the diagram and go into Google and ask it to show "restaurants near me", it will show you eating establishments in the Dallas, Texas area even though you are physically in Los Angeles.

There are several free and low-cost products in this space. A quick Google search for "VPN reviews" will retrieve a ton of information. Some of the reviews give preference to any service located outside the United States, because that means that the U.S. National Security Agency does not have backdoors into the software (who knows if they really do or not). You will need to determine what features matter most to you. The author currently uses Express VPN, which is headquartered out of the British Virgin Islands and has very strong privacy protection. Honestly, that had little to do with selecting that product. My concerns are security and speed. I'm not trying to hide my activity from the U.S. government. I'm trying to hide my activity from hackers on a hotel network. You need to decide which features really matter to you when evaluating

products.

SEC301.3 | Introduction to Cyber Security

SANS

An Introduction to Cryptography End Day 3

This page intentionally left blank.

This page intentionally left blank.

Appendix

Appendix
Review Questions

This page intentionally left blank.

SANS

Intro to Crypto Review Question

This page intentionally left blank.

Review Questions (1)

- What is information in a human-readable form called (choose two)?
  A. Plaintext
  B. Ciphertext
  C. Cleartext
  D. Encrypted
- What is the process of turning plaintext into ciphertext called (choose two)?
  A. Decipherment
  B. Encipherment
  C. Decryption
  D. Encryption
  E. Obfuscation

This page intentionally left blank.

Intro to Crypto

Review Questions (2)

- What is the art and science of breaking cryptography called (choose two)?
  A. Cryptjacking
  B. Cryptanalytic Attack
  C. Cryptanalysis
  D. Cryptographic Testing
- What does "The range of values that can be used to construct a key" define?
  A. Cryptanalysis
  B. Algorithm
  C. Keyspace
  D. Encryption

This page intentionally left blank.

- The Scytale Cipher used a stick of a certain diameter. What was the other important characteristic of this stick?
  A. It had to be made of ash
  B. It had to be a very specific length
  C. It has a specific taper
  D. Only the diameter mattered

- What are two of the services of encryption (choose two)?
  A. Decipherment
  B. Confidentiality
  C. Availability
  D. Integrity
  E. Distortion

This page intentionally left blank.

- What is another name for hidden ciphers?
  A. Sir John Trevanion's Letter
  B. Data Encryption Standard (DES)
  C. Invisible Ink
  D. Steganography

- There is secrecy of communication and confidentiality of communication. Which does encryption provide?
  A. Secrecy
  B. Confidentiality
  C. Neither
  D. Both

This page intentionally left blank.

Review Questions (5)

- What is a cipher that reorders the letters of a message called (choose two)?
  A. Transposition
  B. Steganography
  C. Deobfuscation
  D. Permutation

- What is a cipher that replaces letters of a message with new letters called?
  A. Cider Cipher
  B. Replacement Cipher
  C. Substitution Cipher
  D. Monoalphabetic

This page intentionally left blank.

Intro to Crypto

Review Questions (6)

- A Caesar Cipher is an example of which two of the following (choose two)?
  A. Permutation Cipher
  B. Polyalphabetic Cipher
  C. Monoalphabetic Cipher
  D. Rotational Substitution Cipher

- What is a cipher that uses more than one replacement alphabet called?
  A. Permutation Cipher
  B. Polyalphabetic Cipher
  C. Monoalphabetic Cipher
  D. Rotational Substitution Cipher

This page intentionally left blank.

- When the cryptographic key and the message are the same length, the key is called a _____ key?
  A. Long
  B. Running
  C. Message
  D. This has no special name

- A cipher replaces letters. What does a code replace?
  A. Letters
  B. Messages
  C. Paragraphs
  D. Words and/or phrases

This page intentionally left blank.

SANS

Building Blocks of Modern Crypto
Review Question

This page intentionally left blank.

Review Questions (1)

- When you add a bit to the length of a key, what happens to the keyspace?
  A. It increases by 50%
  B. It triples
  C. It doubles
  D. It remains the same

- Computers cannot create truly random values. Therefore, the randomization function they use is called a
  A. Pretty Random Number Generator (PRNG)
  B. Mostly Random Number Generator (MRNG)
  C. Pseudo-Random Number Generator (PRNG)
  D. Very Random Number Generator (VRNG)

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (2)

- What is the name of the attack that attempts to guess every possible cryptographic key?
  A. Birthday Attack
  B. Weak Key Attack
  C. Brute Force
  D. That attack is impossible!

- What is the name of an attack based on the probability that certain keys are more likely to occur than other keys?
  A. Birthday Attack
  B. Weak Key Attack
  C. Brute Force
  D. That attack is impossible!

This page intentionally left blank.

Review Questions (3)

- In cryptography, the term "snake oil" is often used to describe what?
  A. Proprietary Encryption Algorithms
  B. Peer-Reviewed Algorithms
  C. That is a nonsense term
  D. Weak Key Attack

- In cryptography, what is it called when a small change in the input results in a huge change to the output?
  A. Modification effect
  B. Avalanche effect
  C. Undesirable
  D. Birthday effect

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (4)

- What is the term for the amount of time it would take to break a cryptographic system?
  A. To infinity—and beyond!
  B. The strength factor
  C. The avalanche effect
  D. The work factor

- Which type of algorithm always gives a fixed length output?
  A. One-Way Hash Algorithm
  B. Symmetric Algorithm
  C. Asymmetric Algorithm
  D. No algorithms do that

This page intentionally left blank.

Review Questions (5)

- When hashing a document, if a word is added to the document, what will the output be? (choose two)
  - A. Longer
  - B. Shorter
  - C. The same length
  - D. Widely Divergent
- When the key that is used to encrypt must be used to decrypt, what kind of algorithm are you working with?
  - A. One-Way Hash Algorithm
  - B. Symmetric Algorithm
  - C. Asymmetric Algorithm
  - D. No algorithms do that

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (6)

- Which type of algorithm is almost always faster?
  A. Signcryption
  B. Hybrid Cryptography
  C. Asymmetric
  D. Symmetric

- When the key that is used to encrypt cannot be used to decrypt, what kind of algorithm are you working with?
  A. One-Way Hash Algorithm
  B. Symmetric Algorithm
  C. Asymmetric Algorithm
  D. No algorithms do that

This page intentionally left blank.

Review Questions (7)

- To encrypt a message before sending, which key would you use?
  A. Recipient Public Key
  B. Recipient Private Key
  C. Sender Public Key
  D. Sender Private Key

- To verify a digital signature, which key would you use?
  A. Recipient Public Key
  B. Recipient Private Key
  C. Sender Public Key
  D. Sender Private Key

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (8)

- What is the longest hash length of the SHA-2 family?
  A. 224 bits
  B. 256 bits
  C. 384 bits
  D. 512 bits
  E. 1,024 bits

- What is an algorithm that encrypts a specific number of bits of text (for example, 64 bits) at a time called?
  A. Stream cipher
  B. No algorithms do that
  C. Block cipher
  D. All algorithms do that

This page intentionally left blank.

Review Questions (9)

- When an algorithm repeats its encryption functions multiple times, what do you call it?
  A. Stream cipher
  B. Block cipher
  C. Rounds of encryption
  D. Multi-Crypts

- What is the more modern and well-known name of the Lucifer algorithm?
  A. Advanced Encryption Standard (AES)
  B. The RC4 Encryption Algorithm
  C. Data Encryption Standard (DES)
  D. Rijndael

This page intentionally left blank.

Review Questions (10)

- What do we call the most common implementation of multiple encryptions today?
  A. DES
  B. AES
  C. Triple DES
  D. Triple AES

- AES has a function called AddRoundKey. What is the basic functionality behind AddRoundKey?
  A. Block Cipher
  B. Stream Cipher
  C. Permutation
  D. XOR
  E. Substitution

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (11)

- What is the name of an algorithm that is only able to do secure key exchange?
  A. Diffie-Hellman
  B. RSA
  C. Rivest–Shamir–Adleman
  D. Triple DES

- To prove nonrepudiation, which two facts must you prove? (Choose two)
  A. When the document was sent
  B. Who sent the document
  C. Who received the document
  D. That the document was not tampered with (the document received is identical to the document sent)

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (12)

- Which asymmetric encryption system is <u>not</u> based on prime numbers?
  A. Advanced Encryption Standard (AES)
  B. Diffie-Hellman
  C. Elliptical Curve Cryptography (ECC)
  D. RSA

- Which attack against crypto is always guaranteed to work?
  A. Brute Force Attack
  B. Statistical Analysis Attack
  C. Birthday Attack
  D. If implemented properly, no attack will succeed

This page intentionally left blank.

Review Questions (13)

- What is it called when the attacker has nothing but encrypted text and attempts to discern patterns?
  A. Impossible
  B. Brute force attack
  C. Weak key attack
  D. Ciphertext-only attack

- What is it called when a flaw is found in a crypto algorithm?
  A. Brute force attack
  B. Ciphertext attack
  C. Mathematical attack
  D. Chosen plaintext attack

This page intentionally left blank.

SANS

Data Encrypting Protocols Review Question

This page intentionally left blank.

Data Encrypting Protocols

Review Questions (1)

- What type of data traffic does S/MIME protect?
  A. Web traffic
  B. Email traffic
  C. File transfer traffic
  D. Multipurpose Internet Message Exchange traffic

- How does Secure Shell (SSH) protect traffic?
  A. By translating it to ASCII before transmission
  B. It doesn't
  C. By using Hexadecimal Binary Encoding
  D. All traffic is end-to-end encrypted

This page intentionally left blank.

Data Encrypting Protocols

Review Questions (2)

- How does Transport Layer Security (TLS) differ from Secure Socket Layer (SSL)?
  A. They are completely different
  B. TLS is a newer version of SSL
  C. TLS operates at OSI Layer 7, whereas SSL operates at OSI Layer 4
  D. TLS encrypts the entire packet whereas SSL encrypts only the data

- What is the default port number for HTTPS?
  A. UDP Port 443
  B. UDP Port 80
  C. TCP Port 80
  D. TCP Port 443

This page intentionally left blank.

Data Encrypting Protocols

Review Questions (3)

- We talked about three encrypted file transfer protocols (FTPS, SFTP, and SCP). Which two use TCP Port 22? (choose two)
  - A. FTPS
  - B. SFTP
  - C. SCP
- What is the most significant difference between IPSec's Authentication Header and Encapsulating Security Payload protocols?
  - A. Authentication Header encrypts traffic
  - B. Encapsulating Security Payload encrypts traffic
  - C. This is two names for the same thing
  - D. Authentication Header decrypts traffic
  - E. Encapsulating Security Payload does not encrypt traffic

This page intentionally left blank.

Appendix

SANS

Appendix
Answers to Review Questions

This page intentionally left blank.

SANS

Intro to Crypto Review Question Answers

This page intentionally left blank.

Intro to Crypto

Review Questions (1)

- What is information that is in a human readable form called (choose two)?
  - **A. Plaintext**
  - B. Ciphertext
  - **C. Cleartext**
  - D. Encrypted
- What is the process of turning plaintext into ciphertext called (chose two)?
  - A. Decipherment
  - **B. Encipherment**
  - C. Decryption
  - **D. Encryption**
  - E. Obfuscation

This page intentionally left blank.

- What is the art and science of breaking cryptography called (choose two)?
  - A. Cryptjacking
  - **B. Cryptanalytic Attack**
  - **C. Cryptanalysis**
  - D. Cryptographic Testing
- What does "The range of values that can be used to construct a key" define?
  - A. Cryptanalysis
  - B. Algorithm
  - **C. Keyspace**
  - D. Encryption

This page intentionally left blank.

Intro to Crypto

Review Questions (3)

- The Scytale Cipher used a stick of a certain diameter.
  What was the other important characteristic of this stick?
  A. It had to be made of ash
  B. It had to be a very specific length
  C. <u>It has a specific taper</u>
  D. Only the diameter mattered

- What are two of the services of encryption (choose two)?
  A. Decipherment
  B. <u>Confidentiality</u>
  C. Availability
  D. <u>Integrity</u>
  E. Distortion

This page intentionally left blank.

Intro to Crypto

Review Questions (4)

- What is another name for hidden ciphers?
  A. Sir John Trevanion's Letter
  B. Data Encryption Standard (DES)
  C. Invisible Ink
  D. Steganography

- There is secrecy of communication and confidentiality of communication. Which does encryption provide?
  A. Secrecy
  B. Confidentiality
  C. Neither
  D. Both

This page intentionally left blank.

Intro to Crypto

Review Questions (5)

- What is a cipher that reorders the letters of a message called (choose two)?
  - **A. Transposition**
  - B. Steganography
  - C. Deobfuscation
  - **D. Permutation**
- What is a cipher that replaces letters of a message with new letters called?
  - A. Cider Cipher
  - B. Replacement Cipher
  - **C. Substitution Cipher**
  - D. Monoalphabetic

This page intentionally left blank.

Intro to Crypto

Review Questions (6)

- A Caesar Cipher is an example of which two of the following (choose two)?
  A. Permutation Cipher
  B. Polyalphabetic Cipher
  C. Monoalphabetic Cipher
  D. Rotational Substitution Cipher

- What is a cipher that uses more than one replacement alphabet called?
  A. Permutation Cipher
  B. Polyalphabetic Cipher
  C. Monoalphabetic Cipher
  D. Rotational Substitution Cipher

This page intentionally left blank.

Intro to Crypto

Review Questions (7)

- When the cryptographic key and the message are the same length, the key is called a _____ key?
  - A. Long
  - **B. Running**
  - C. Message
  - D. This has no special name

- A cipher replaces letters. What does a code replace?
  - A. Letters
  - B. Messages
  - C. Paragraphs
  - **D. Words and/or phrases**

This page intentionally left blank.

SANS

Building Blocks of Modern Crypto: Review Question Answers

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (1)

- When you add a bit to the length of a key, what happens to the keyspace?
  A. It increases by 50%
  B. It triples
  C. It doubles
  D. It remains the same

- Computers cannot create truly random values. Therefore, the randomization function they use is called a
  A. Pretty Random Number Generator (PRNG)
  B. Mostly Random Number Generator (MRNG)
  C. Pseudo-Random Number Generator (PRNG)
  D. Very Random Number Generator (VRNG)

This page intentionally left blank.

Review Questions (2)

- What is the name of the attack that attempts to guess every possible cryptographic key?
  A. Birthday Attack
  B. Weak Key Attack
  C. Brute Force
  D. That attack is impossible!

- What is the name of an attack based on the probability that certain keys are more likely to occur than other keys?
  A. Birthday Attack
  B. Weak Key Attack
  C. Brute Force
  D. That attack is impossible!

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (3)

- In cryptography, what is the term "snake oil" often used to describe?
  - **A. Proprietary Encryption Algorithms**
  - B. Peer-Reviewed Algorithms
  - C. That is a nonsense term
  - D. Weak Key Attack

- In cryptography, what is it called when a small change in the input results in a huge change to the output?
  - A. The modification effect
  - **B. An avalanche effect**
  - C. Undesirable
  - D. The birthday effect

This page intentionally left blank.

- What is the term for the amount of time it would take to break a cryptographic system?
  A. To infinity—and beyond!
  B. The strength factor
  C. The avalanche effect
  D. <u>The work factor</u>

- Which type of algorithm always gives a fixed length output?
  A. <u>One-Way Hash Algorithm</u>
  B. Symmetric Algorithm
  C. Asymmetric Algorithm
  D. No algorithms do that

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (5)

- When hashing a document, if a word is added to the document, what will the output be? (choose two)
  - A. Longer
  - B. Shorter
  - **C. <u>The same length</u>**
  - **D. <u>Widely Divergent</u>**

- When the key that is used to encrypt must be used to decrypt, what kind of algorithm are you working with?
  - A. One-Way Hash Algorithm
  - **B. <u>Symmetric Algorithm</u>**
  - C. Asymmetric Algorithm
  - D. No algorithms do that

This page intentionally left blank.

Review Questions (6)

- Which type of algorithm is almost always faster?
  A. Signcryption
  B. Hybrid Cryptography
  C. Asymmetric
  D. Symmetric

- When the key that is used to encrypt cannot be used to decrypt, what kind of algorithm are you working with?
  A. One-Way Hash Algorithm
  B. Symmetric Algorithm
  C. Asymmetric Algorithm
  D. No algorithms do that

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (7)

- To encrypt a message before sending, which key would you use?
  - **A. Recipient Public Key**
  - B. Recipient Private Key
  - C. Sender Public Key
  - D. Sender Private Key

- To verify a digital signature, which key would you use?
  - A. Recipient Public Key
  - B. Recipient Private Key
  - **C. Sender Public Key**
  - D. Sender Private Key

This page intentionally left blank.

- What is the longest hash length of the SHA-2 family?
  A. 224 bits
  B. 256 bits
  C. 384 bits
  D. 512 bits
  E. 1,024 bits

- What is an algorithm that encrypts a specific number of bits of text (for example, 64 bits) at a time called?
  A. Stream cipher
  B. No algorithms do that
  C. Block cipher
  D. All algorithms do that

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (9)

- When an algorithm repeats its encryption functions multiple times, what do you call it?
  A. Stream cipher
  B. Block cipher
  C. Rounds of encryption
  D. Multi-Crypts

- What is the more modern and well-known name of the Lucifer algorithm?
  A. Advanced Encryption Standard (AES)
  B. The RC4 Encryption Algorithm
  C. Data Encryption Standard (DES)
  D. Rijndael

This page intentionally left blank.

- What do we call the most common implementation of multiple encryptions today?
  A. DES
  B. AES
  C. Triple DES
  D. Triple AES

- AES has a function called AddRoundKey. What is the basic functionality behind AddRoundKey?
  A. Block Cipher
  B. Stream Cipher
  C. Permutation
  D. XOR
  E. Substitution

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (11)

- What is the name of an algorithm that is only able to do secure key exchange?
  - **A. Diffie-Hellman**
  - B. RSA
  - C. Rivest–Shamir–Adleman
  - D. Triple DES

- To prove nonrepudiation, which two facts must you prove? (Choose two)
  - A. When the document was sent
  - **B. Who sent the document**
  - C. Who received the document
  - **D. That the document was not tampered with (the document received is identical to the document sent)**

This page intentionally left blank.

- Which asymmetric encryption system is <u>not</u> based on prime numbers?
  A. Advanced Encryption Standard (AES)
  B. Diffie-Hellman
  C. <u>Elliptical Curve Cryptography (ECC)</u>
  D. RSA

- Which attack against crypto is always guaranteed to work?
  A. <u>Brute Force Attack</u>
  B. Statistical Analysis Attack
  C. Birthday Attack
  D. If implemented properly, no attack will succeed

This page intentionally left blank.

Building Blocks of Modern Crypto

Review Questions (13)

- What is it called when the attacker has nothing but encrypted text and attempts to discern patterns?
  A. Impossible
  B. Brute force attack
  C. Weak key attack
  D. Ciphertext only attack

- What is it called when a flaw is found in a crypto algorithm?
  A. Brute force attack
  B. Ciphertext attack
  C. Mathematical attack
  D. Chosen plaintext attack

This page intentionally left blank.

SANS

Data Encrypting Protocols Review Question Answers

This page intentionally left blank.

Data Encrypting Protocols

Review Questions (1)

- What type of data traffic does S/MIME protect?
  - A. Web traffic
  - **B. <u>Email traffic</u>**
  - C. File transfer traffic
  - D. Multipurpose Internet Message Exchange traffic
- How does Secure Shell (SSH) protect traffic?
  - A. By translating it to ASCII before transmission
  - B. It doesn't
  - C. By using Hexadecimal Binary Encoding
  - **D. <u>All traffic is end-to-end encrypted</u>**

This page intentionally left blank.

Data Encrypting Protocols

Review Questions (2)

- How does Transport Layer Security (TLS) differ from Secure Socket Layer (SSL)?
  A. They are completely different
  B. TLS is a newer version of SSL
  C. TLS operates at OSI Layer 7, whereas SSL operates at OSI Layer 4
  D. TLS encrypts the entire packet whereas SSL encrypts only the data

- What is the default port number for HTTPS?
  A. UDP Port 443
  B. UDP Port 80
  C. TCP Port 80
  D. TCP Port 443

This page intentionally left blank.

Data Encrypting Protocols

Review Questions (3)

- We talked about three encrypted file transfer protocols (FTPS, SFTP, and SCP). Which two use TCP Port 22? (choose two)
  - A. FTPS
  - **B. SFTP**
  - **C. SCP**

- What is the most significant difference between IPSec's Authentication Header and Encapsulating Security Payload protocols?
  - A. Authentication Header encrypts traffic
  - **B. Encapsulating Security Payload encrypts traffic**
  - C. This is two names for the same thing
  - D. Authentication Header decrypts traffic
  - E. Encapsulating Security Payload does not encrypt traffic

This page intentionally left blank.

Many of the slide graphics in this course are provided through a royalty-free license with PresentationPro. http://www.presentationpro.com/