Cryptology (1)

Terms and Concepts

John York, Blue Ridge Community College

http://www.brcc.edu

Most slides have references or more information in the notes pages. In PowerPoint, select View > Notes Page

Much of the information in this course came from “Understanding Cryptography” by Christoff Parr and Jan Pelzl, Springer-Verlag 2010
Much of the classical cryptography material and most Python scripts came from “Cracking Codes with Python” by Al Sweigart, NoStarch Press 2018
Also helpful, “Cryptography Engineering” by Ferguson, Schneier, and Kohno, Wiley Publishing, 2010

Getting Started--Definitions

- Cryptology
  - Study of secret codes
  - Includes cryptography and cryptanalysis
- Cryptography--Writing or creating secret codes
- Cryptanalysis--Analyzing or breaking secret codes
- Plaintext--the message we want to keep secret, before it is encrypted
- Ciphertext--the message, after it is encrypted
- Key—the secret used by an encryption algorithm to encrypt or decrypt a message

There is debate about the separate definitions of cryptology, cryptography, and cryptanalysis. Some say the terms are interchangeable. https://crypto.stackexchange.com/questions/57818/is-there-any-difference-between-cryptography-and-cryptology

The other definitions are widely accepted.

Basic Cryptology Rules

- Kerckhoffs's Principle
  - **Security of encryption should only depend on the secrecy of the key**
- Corollary
  - If security depends on the secrecy of the algorithm
    - It is not encryption
    - It is a Capture The Flag (CTF) puzzle
- Schneier's Law
  - "Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break. It's not even hard. What is hard is creating an algorithm that no one else can break, even after years of analysis. And the only way to prove that is to subject the algorithm to years of analysis by the best cryptographers around."

https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle

https://www.schneier.com/blog/archives/2011/04/schneiers_law.html

A system that tries to gain secrecy by hiding the algorithm or method is also called "obfuscation" or "security though obscurity." A good cryptanalyst can reverse engineer a system that relies on the secrecy of the algorithm, given enough time.

An algorithm that has had years of examination by the world cryptanalyst community looking for weaknesses is probably secure. If an algorithm is invented by one or two people, even talented cryptographers, it is likely that an undiscovered subtle weakness exists. Once the algorithm is available or has been reverse engineered, those subtle weaknesses will be quickly discovered.

WEP—a Cryptography Failure (1)

- Wired Equivalent Privacy (WEP)
  - An early protocol designed to encrypt wireless traffic
  - Short key length, 40 or 104 bits
  - Some limitations due to the hardware of the time, BUT…
- WEP used proven stream encryption RC4, but…
  - RC4 spec's clearly state that the same key should never be reused
  - Small key space (24 bit Initialization Vector) caused key to be reused in about 5 hours on a busy Access Point (AP)
  - Statistical analysis could break WEP (recover the key) by listening to busy AP
- WEP did poor job of preventing attacker from inserting packets
  - Allowed attack programs (AirCrack) to break WEP quickly by generating traffic

http://www.dummies.com/programming/networking/understanding-wep-weaknesses/

https://security.blogoverflow.com/2013/08/wifi-security-history-of-insecurities-in-wep-wpa-and-wpa2/

http://dl.aircrack-ng.org/breakingwepandwpa.pdf

https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy

In defense of cryptographers, Bruce Schneier states that the committee that designed WEP did not include any cryptographers. He also lists the many problems of WEP "Cryptography Engineering" by Ferguson, Schneier, and Kohno, Wiley Publishing, 2010, pp 323-324

WEP—a Cryptography Failure (2)

- WEP was an IEEE standard ratified in 1997
  - Passive attack published in 2001 needed 4-6 million packets to recover key
  - Improved passive attack in 2004 needed 700,000 packets
  - By 2007 a passive attack needed 35-40,000 packets
  - Active attacks published in 2004 could recover key in seconds
- Lessons Learned
  - Cryptography is hard
  - Use algorithms and protocols that have been examined by many people for a long time
  - Upgrading crypto algorithms and protocols over time is necessary

Once the research community discovered that WEP had weaknesses, they improved their attacks at an amazing rate. The most serious WEP failure was that it allowed attackers to insert almost any crafted packet into the communication stream. These specially crafted packets were designed to make WEP divulge its key.

Encryption vs. Encoding

- Encryption hides content by using a secret key
- Encoding changes content into a different format, no key required
- Example: base64 encoding
  - Text based systems (email) have difficulty with binary data (pictures, etc.)
  - Base64 converts binary data into text symbols
  - Base64 symbol set is [A-Za-z0-9+/] for 64 symbols
  - 3 bytes of data converts to 4 symbols, add = to end to fit byte boundary
  - “base64 is not encryption” in text converts to:
  - “YmFzZTY0IGlzIG5vdCBlbmNyeXB0aW9u” in base64
- No key needed to decode base64—included in Linux and PowerShell

In Linux

```
john@svgs-ubuntu18:~$ echo -n "base64 is not encryption" | base64
YmFzZTY0IGlzIG5vdCBlbmNyeXB0aW9u
john@svgs-ubuntu18:~$ echo -n "YmFzZTY0IGlzIG5vdCBlbmNyeXB0aW9u" | base64 -d
base64 is not encryption
```

In Windows, the certificate utility, certutil, works for base64. It adds extra text normally present in certificates, but that can be dealt with.

```
C:\Users\John>@echo base64 is not encryption > text.txt
C:\Users\John>type text.txt
base64 is not encryption
C:\Users\John>certutil -encode text.txt base64out.txt
<snip>
C:\Users\John>type base64out.txt
-----BEGIN CERTIFICATE-----
YmFzZTY0IGlzIG5vdCBlbmNyeXB0aW9uIA0K
-----END CERTIFICATE-----
C:\Users\John>certutil -decode base64out.txt base64in.txt
<snip>
C:\Users\John>type base64in.txt
base64 is not encryption
```

Encryption vs. Obfuscation

- Malware authors use algorithms to disguise their malware
  - Attempts to evade Intrusion Detection and Anti-Virus
  - Makes it more difficult for defenders to reverse engineer the malware
  - Advertising often uses obfuscation and thus looks like malware
- Malware normally attacks any computer it comes across
  - Therefore it does not normally use a key. If it does use a key, the key is publicly accessible
- Techniques/algorithms are as varied as the malware authors are imaginative.
- Nonetheless, if all computers can decode the malware, reverse engineers can usually decode it as well

https://www.sans.org/reading-room/whitepapers/engineering/pdf-obfuscation-primer-34005

https://www.sans.org/reading-room/whitepapers/malicious/obfuscation-polymorphism-interpreted-code-37602

In rare cases (state sponsored malware is one) the malware can be so specifically targeted that the configuration of the target computer effectively becomes a key. See Zero Day by Kim Zetter, about Flame and Duqu. If the attacker has a lot of specific information about the target (OS type, software installed, hardware configuration and serial numbers, MAC address, etc.) the software can be written so that it runs only on that computer. Malware researchers will have to duplicate the configuration to be able to examine the malware.

On the other hand, malware that is designed to run on many or all computers will have to have simpler obfuscation and is susceptible to reverse engineering.

Cryptography—major areas

- Symmetric Encryption
  - Same key used for encryption and decryption
- Asymmetric Encryption
  - One key encrypts, another key decrypts
- Protocols
  - Exchanging keys and traffic in a secure manner

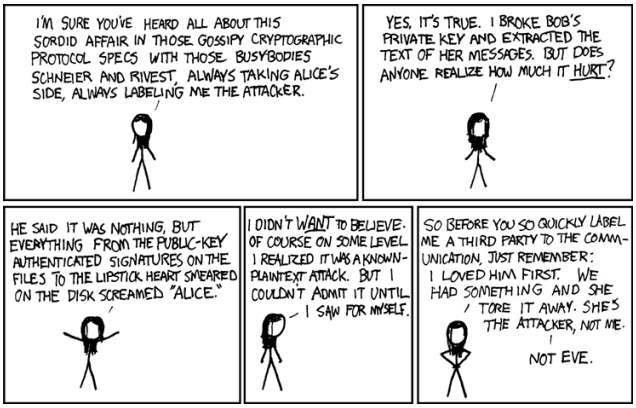
A fundamental difference in types of encryption algorithms is that between symmetric and asymmetric encryption.

Symmetric encryption has been around for thousands of years. The Caesar cipher is an example. The basic property of symmetric encryption or a symmetric cipher is that the same key is used for both encryption and decryption.

Asymmetric encryption is new, with the first publications occurring around 1976. It relies on mathematical functions, and uses a pair two keys. If a message is encrypted with one of the keys, it can only be decrypted with the other key in the pair.

The other major area of work in cryptography is the development of protocols so that keys and encrypted data can be exchanged in a secure manner.

Diversion: Alice, Bob, Eve, Mallory, et al.



- Common Crypto example
- Alice and Bob attempt secure communication
- Eve is an Eavesdropper
- Mallory does MITM attacks
- Others as needed
- https://xkcd.com/177/
- https://en.wikipedia.org/wiki/Alice_and_Bob

Cryptography examples almost always include Alice and Bob as the two people trying to exchange data securely. Often the attacker is Eve (for eavesdropper) but there are many other names for attackers in use, depending on their method of attack.

Symmetric Encryption

- Same key used both for encryption and decryption
- Advantage
  - FAST
- Disadvantage
  - Both sides (Alice and Bob) have to have the same key
  - How do they agree on a key without Eve getting the key as well?
- Agreeing on a key, securely
  - Out of channel—Alice gives the key to Bob in person, or something similar
  - Key exchange protocols
  - Use other forms of encryption to transmit the key

There are two key things to remember about symmetric encryption, speed and key exchange.

Symmetric encryption can be implemented directly in hardware, or in low level code like assembly language. It consists of simple operations that can be done quickly. Therefore it is very fast.

However, the key has to be given to the other party so the message can be decrypted. If the attacker can intercept the key, the attacker can also decrypt the message, no matter how good the encryption is. This is called the key exchange problem.

Asymmetric Encryption

- Also known as public key encryption
- One key used for encryption, one key for decryption
- Advantage
  - Done properly, it allows for secure key exchange
- Disadvantage
  - Slow, ~1,000 times slower than symmetric encryption
  - Each encryption/decryption cycle requires taking a large (1024 bits or more) number to a large exponent
- Usually, use Asymmetric encryption for key exchange then switch to symmetric to encrypt data

Asymmetric encryption involves complex mathematical operations, generally taking very large numbers to very large exponents. These operations are slow, so asymmetric (or public key) encryption is slow.

Asymmetric encryption has the advantage that it can transfer data securely without having to pass the key to the other party. Assume Alice creates a public/private key pair and posts the public key in a trusted location that Bob can access. If Bob encrypts data with Alice's public key, it can only be decrypted with Alice's private key. If Alice has kept her private key secret, then the message can be transmitted securely.