Cyber Security Technologies

Session 9 – Cryptography I

Shawn Davis
ITMS 448 – Spring 2016

Slides contain original content from Davis, S., and Lidinsky, W.

Note

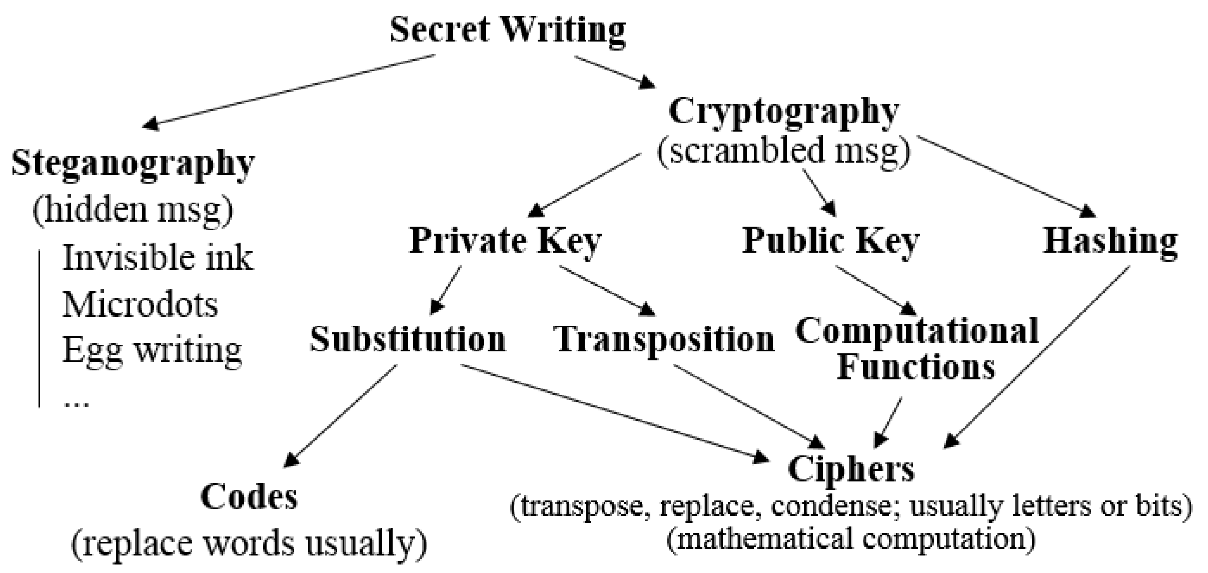
- Logon to RADISH and your Win 8.1 VM
- Open your M: Drive and go to the Tools folder
- Copy SetupCrypTool and paste to your desktop
- Run the installer
- Choose default prompts
- Un-check “Show Readme” and hit “Finish”
- Close the “How to Start” dialog window
- We will use CrypTool at times during the lecture

Note

- We will also use Kali, so make sure that is ready as well with a terminal open
- These slides are available in Blackboard under Week 9
- I recommend pulling them up now to make going through the labs easier in case you need to back up a slide or two at times

Overview

A Taxonomy



Cryptography and **Ciphers** usually refer to the same things, with Cryptography being the science and Ciphers being the encoded entities. The only exception is **Codes.**

Private key ciphers usually use S and T.

Public key systems use CF.

A Notable Code:
Navajo natural language

Some Notable Ciphers:

Caesar (S)
Polyalphabetic (S)
One-time pad (S)
DES (T&S)
IDEA
RSA Pub (CF)
Elliptical (CF)
AES (T&S)
Monoalphabetic (S)
Vigenère (S)
Enigma (T&S)
RSA (T&S)
MD5 (H)
Diffie-Hellman (CF)
SHA-1 (H)

S = substitution
T = transposition
H = hashing
CF = computational functions

Two Part Lecture

- Part I:
  - Cryptography Basics
  - Private Key Crypto
  - Codes
    - Substitution
  - Ciphers
    - Substitution
    - Transposition
- Part II:
  - Public Key Crypto
  - Computational Functions
  - Ciphers
  - Hashing
  - Secure Communications

  - Midterm Answer Review

Part I

Cryptography Basics

Some Terminology

- Secret Writing: The science of hiding ***messages*** or their ***meaning*** from outsiders.
- Steganography: Science of hiding messages so that the ***existence*** of the message is ***unknown*** to outsiders.
- Cryptography: Science of hiding the ***meaning*** of messages that are ***known*** to exist.
  - Writing and reading scrambled (often incorrectly referred to as "coded") messages.
  - Today ***cryptography*** subsumes ***coding***, ***transposing***, ***hashing***, and some forms of authentication.

Some Terminology (Cont.)

- Coding (Substituting): The process of substituting one part of a message with something that is external to the message so as to hide its meaning from outsiders. Parts can be letters, words, phrases, bits...
- Transposing: The process of moving parts of a message from one place to another within the message so as to hide its meaning from outsiders. Parts can be letters, words, phrases, bits...
- Hashing: Creating a fixed-size value called a "hash" using a message as the source to create the hash, and doing it in a way that distributes the possible messages evenly among the possible hash values.

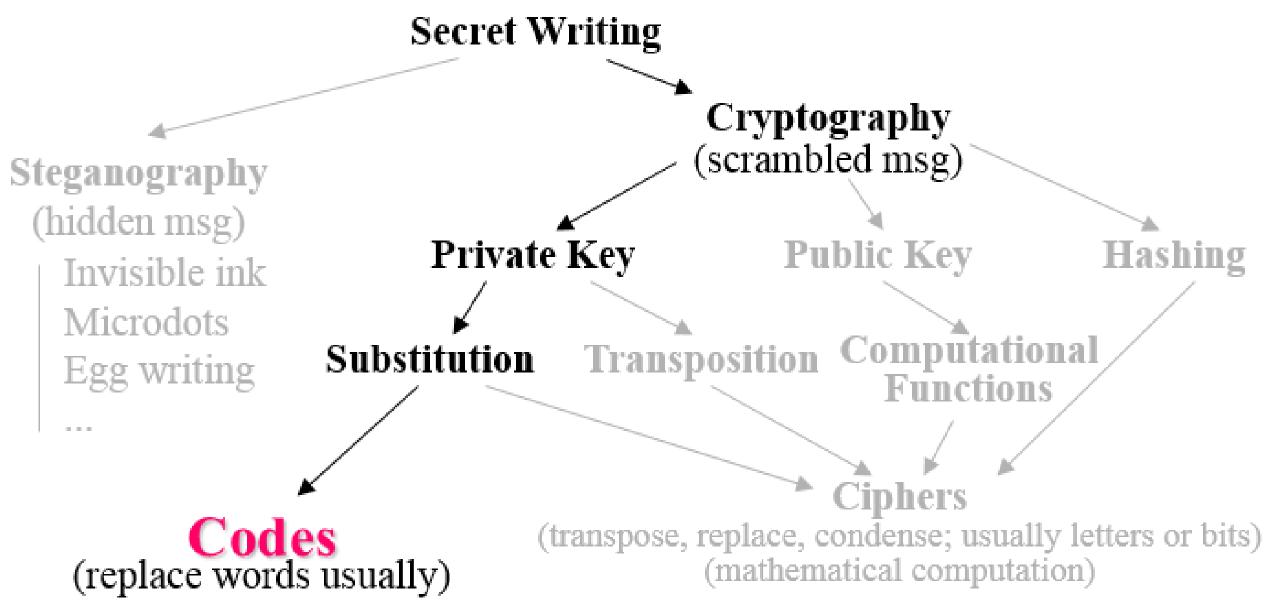
Some More Terminology

- Cryptanalyst: A person or persons that try to decipher encrypted messages. Often synonymous with "attacker" or "threat".
- Encryption: The science of scrambling the contents of the message in such a way that hides its meaning from outsiders.
- Plaintext or Cleartext: Unencrypted message (not necessarily text).
- Ciphertext: Encrypted plaintext.
- Entropy: Measure of uncertainty associated with a random variable

Part II

Codes

A Taxonomy



A Notable Code:
Navajo natural language

S = substitution
T = transposition
H = hashing
CF = computational functions

Some Notable Ciphers:

Caesar (S)	Monoalphabetic (S)
Polyalphabetic (S)	Vigenère (S)
One-time pad (S)	Enigma (T&S)
DES (T&S)	RSA (T&S)
IDEA	MD5 (H)
RSA Pub (CF)	Diffie-Hellman (CF)
Elliptical (CF)	SHA-1 (H)
AES (T&S)	

Cryptography and **Ciphers** usually refer to the same things, with Cryptography being the science and Ciphers being the encoded entities. The only exception is **Codes.**

Private key ciphers usually use S and T.

Public key systems use CF.

Codes

- A code substitutes a part of a message with *something else* so as to hide its meaning from outsiders.
  - The "something else" is not another part of the message
    - One word for another word
    - A word or phrase in one language for that of another language
    - A number for a letter or word
    - A symbol for part of a message

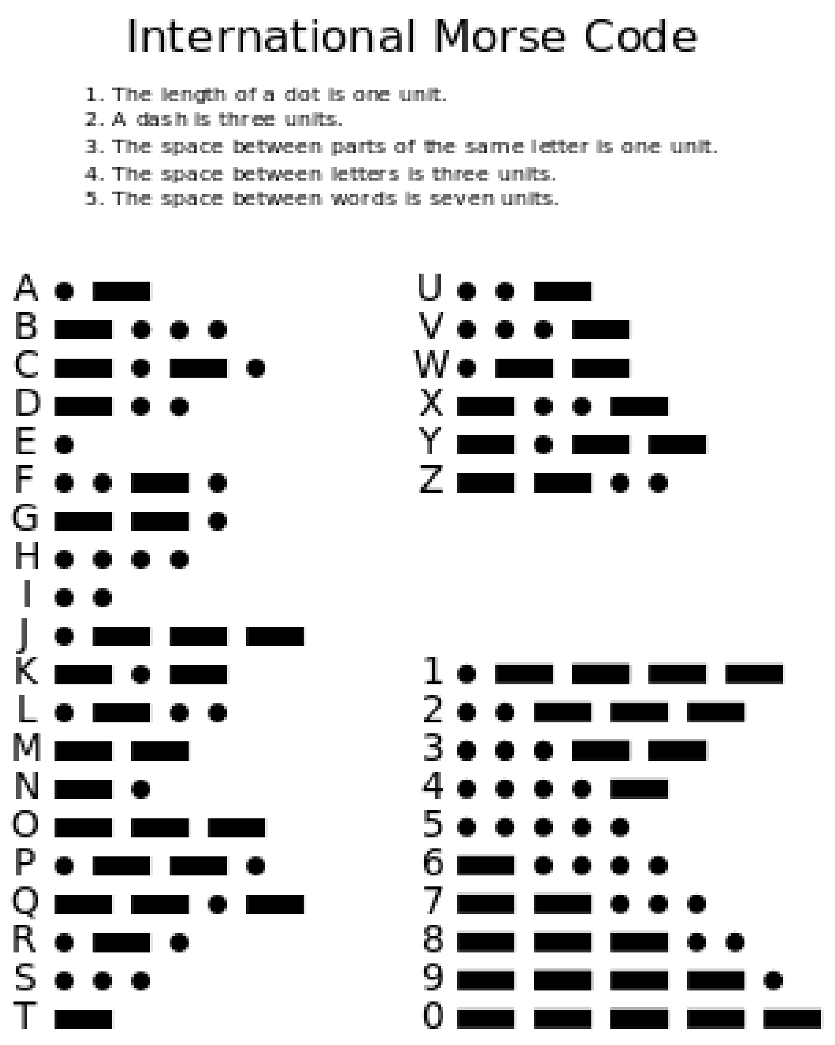
Codebook

- Codes often have a codebook which:
  - Maps the letter, word or phrase to the code

- Can anyone think of any real world examples of a code???

Morse Code

- Maps dots and dashes to letters



International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	● ▬	U	● ● ▬
B	▬ ● ● ●	V	● ● ● ▬
C	▬ ● ▬ ●	W	● ▬ ▬
D	▬ ● ●	X	▬ ● ● ▬
E	●	Y	▬ ● ▬ ▬
F	● ● ▬ ●	Z	▬ ▬ ● ●
G	▬ ▬ ●		
H	● ● ● ●		
I	● ●		
J	● ▬ ▬ ▬		
K	▬ ● ▬	1	● ▬ ▬ ▬ ▬
L	● ▬ ● ●	2	● ● ▬ ▬ ▬
M	▬ ▬	3	● ● ● ▬ ▬
N	▬ ●	4	● ● ● ● ▬
O	▬ ▬ ▬	5	● ● ● ● ●
P	● ▬ ▬ ●	6	▬ ● ● ● ●
Q	▬ ▬ ● ▬	7	▬ ▬ ● ● ●
R	● ▬ ●	8	▬ ▬ ▬ ● ●
S	● ● ●	9	▬ ▬ ▬ ▬ ●
T	▬	0	▬ ▬ ▬ ▬ ▬

Navajo Code

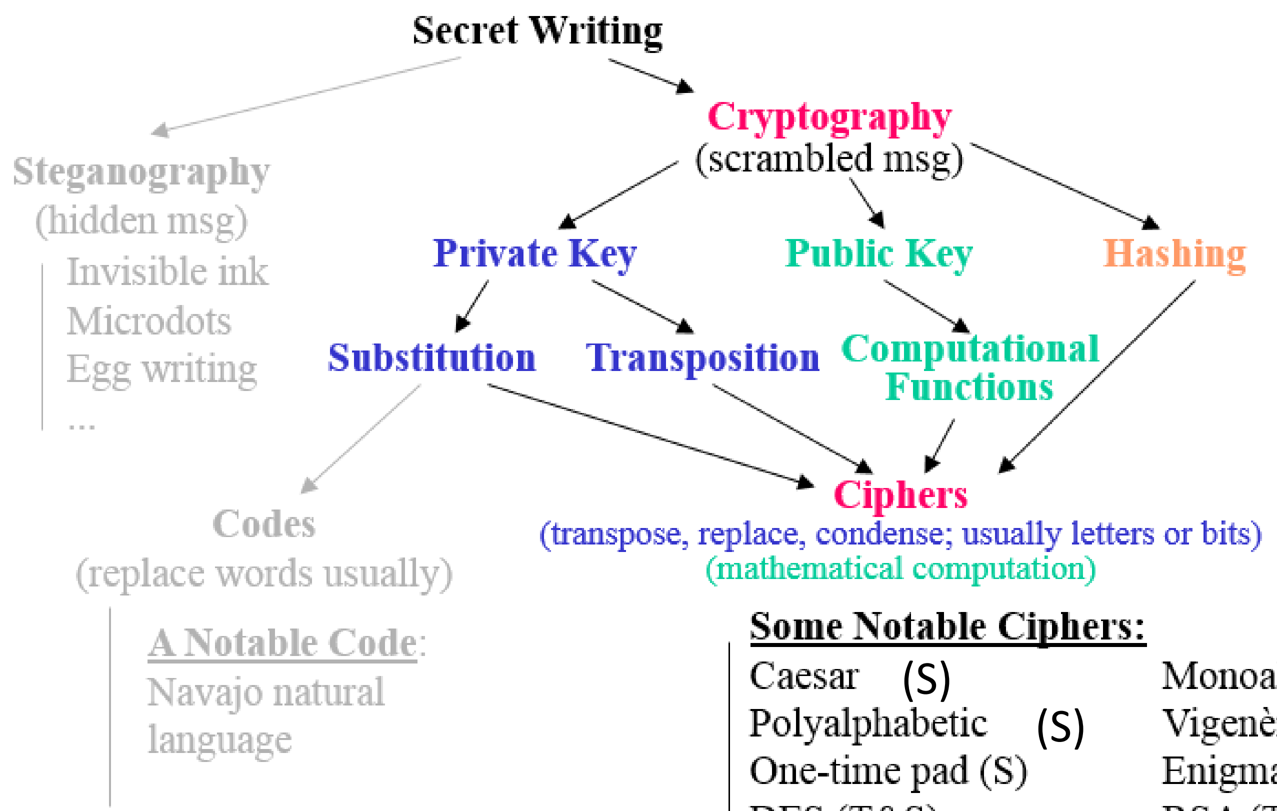
- U.S deployed Navajo American Indian soldiers who spoke the Navajo language to many regiments where fighting was going on
- Technically Navajo was really a form of word substitution "coding" and even phrase substitution
  - This is an example of a Code -- not a Cipher
  - Navajo is one of the Na-Dene languages
    - Other Na-Dene languages: Athabaskan, Tinglit, Eyak
  - It has no known link to any Asian or European language
- It was never broken by the Japanese

- Interesting book
  - Doris Paul, "The Navajo Code Talkers", ISBN 0805945903, Dorrance Publishing 1998.

Part III

Cryptographic Ciphers

A Taxonomy



Cryptography and **Ciphers** usually refer to the same things, with Cryptography being the science and Ciphers being the encoded entities. The only exception is **Codes.**

Private key ciphers usually use S and T.

Public key systems use CF.

Some Notable Ciphers:

Caesar (S)	Monoalphabetic (S)
Polyalphabetic (S)	Vigenère (S)
One-time pad (S)	Enigma (T&S)
DES (T&S)	RSA (T&S)
IDEA	MD5 (H)
RSA Pub (CF)	Diffie-Hellman (CF)
Elliptical (CF)	SHA-1 (H)
AES (T&S)	

S = substitution
T = transposition
H = hashing
CF = computational functions

Codes vs Ciphers

- In the last section, we talked about **Codes** which use substitution with external items such as words, objects, etc.
  - Not mathematical

- We will not cover **Ciphers** which use substitution and transposition
  - Involves mathematical computation to encrypt and decrypt

Ciphers

- Theoretically no cipher is undecipherable
  - Except maybe one (to be discussed later)
- The idea is to make the work and time needed to break the cipher such that it is not worth it

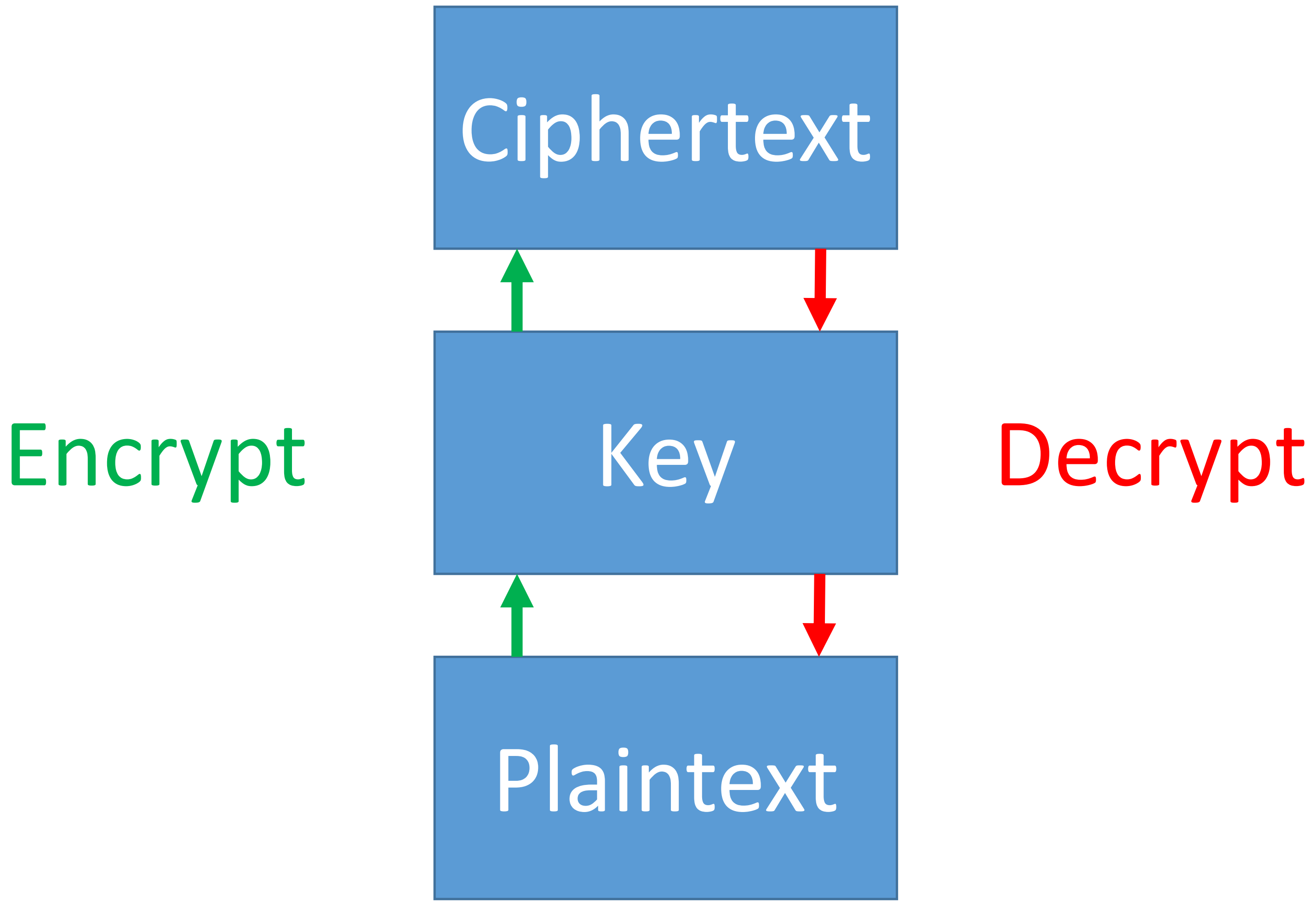
Goals of Cryptographic Ciphers

- **Confidentiality**
  - Ensures that the meaning of the message is hidden for outsiders.
- **Authentication**
  - Establishes the identity of the sender of the message.
- **Integrity**
  - Ensures that the message & its authentication have not been changed.
- **Non repudiation**
  - The sender cannot deny that she or he sent the message.
    - Closely related to Authentication.
  - Non-repudiation is sometimes not listed as goal because it is achieved via authentication methods
- Note: There is no attempt to hide the fact that a cipher exists.

How Goals Are Achieved

- The three (or four) goals are achieved by using one or more of these technologies, often in combination
  - Private key encryption
    - Also called ***symmetric*** or *secret*
  - Public key encryption
    - Also called ***asymmetric***
  - Hashing
    - Also called ***message digest***

How All Ciphers Generally Work



Two Types of Ciphers – Stream & Block

- Stream ciphers process messages a bit or byte at a time when encrypting or decrypting
- Block ciphers divide messages into blocks, each of which is then en/decrypted
  - Sort of like a substitution on very big characters
    - 64-bits or more
  - Many current ciphers are block ciphers

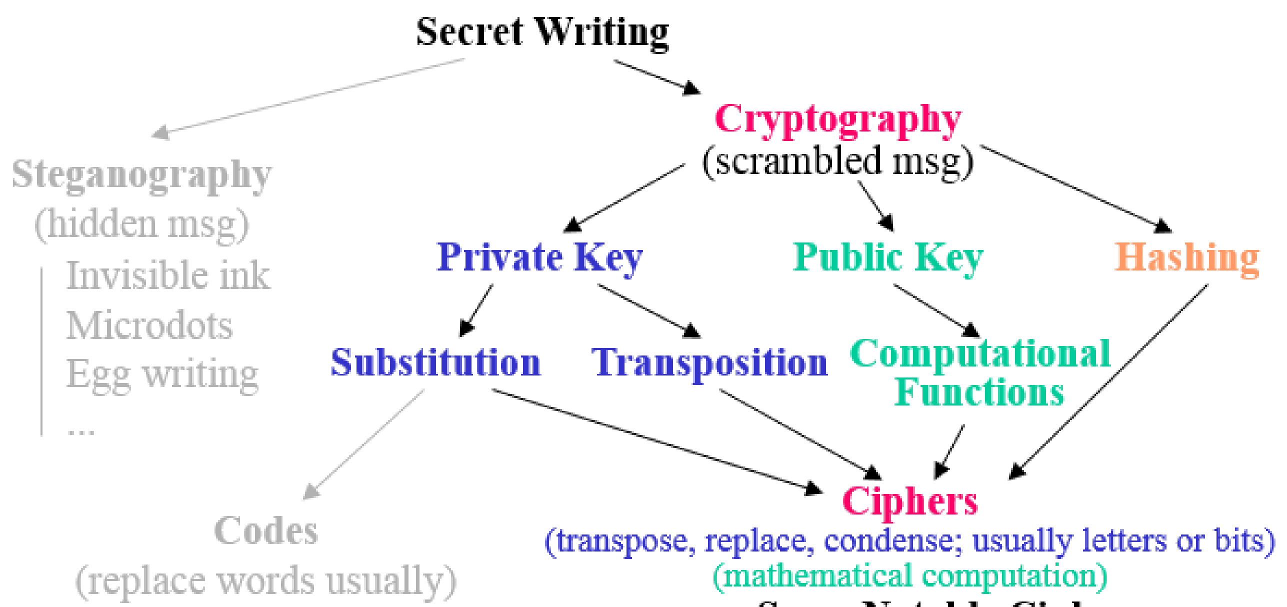
Part IV

Private Key Cryptography

Private Key Cryptography

- A single key is used to encrypt as well as decrypt
  - Both parties share that same key
- Private Key Crypto is also called:
  - Symmetric Key Cryptography
  - Secret Key Cryptography
- Uses Stream & Block Ciphers which perform:
  - Substitution
  - Transposition

A Taxonomy



A Notable Code: Navajo natural language

Cryptography and Ciphers usually refer to the same things, with Cryptography being the science and Ciphers being the encoded entities. The only exception is **Codes.** Private key ciphers usually use S and T. Public key systems use CF.

Some Notable Ciphers:

Caesar (S)	Monoalphabetic (S)	Stream
Polyalphabetic (S)	One-time pad (S)	
Enigma (T&S)	Vigenere (S)	Block
DES (T&S)	RSA (T&S)	
IDEA	MD5 (H)	
RSA Pub (CF)	Diffie-Hellman (CF)	
Elliptical (CF)	SHA-1 (H)	
AES (T&S)		

S = substitution
T = transposition
H = hashing
CF = computational functions

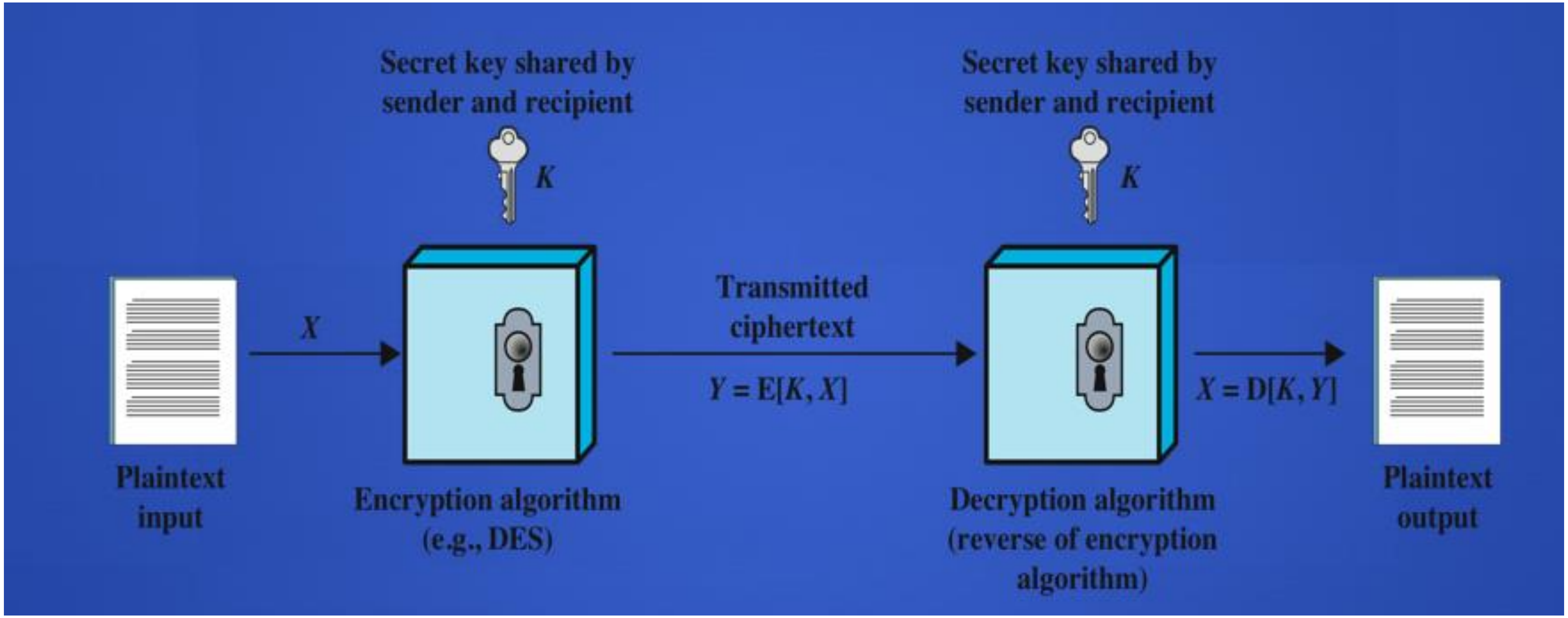
Private Key Ciphers

- Until late 1970s all ciphers were private key
- History of encryption has been a war between cryptography and cryptanalysis
  - First one has the upper hand; then the other!

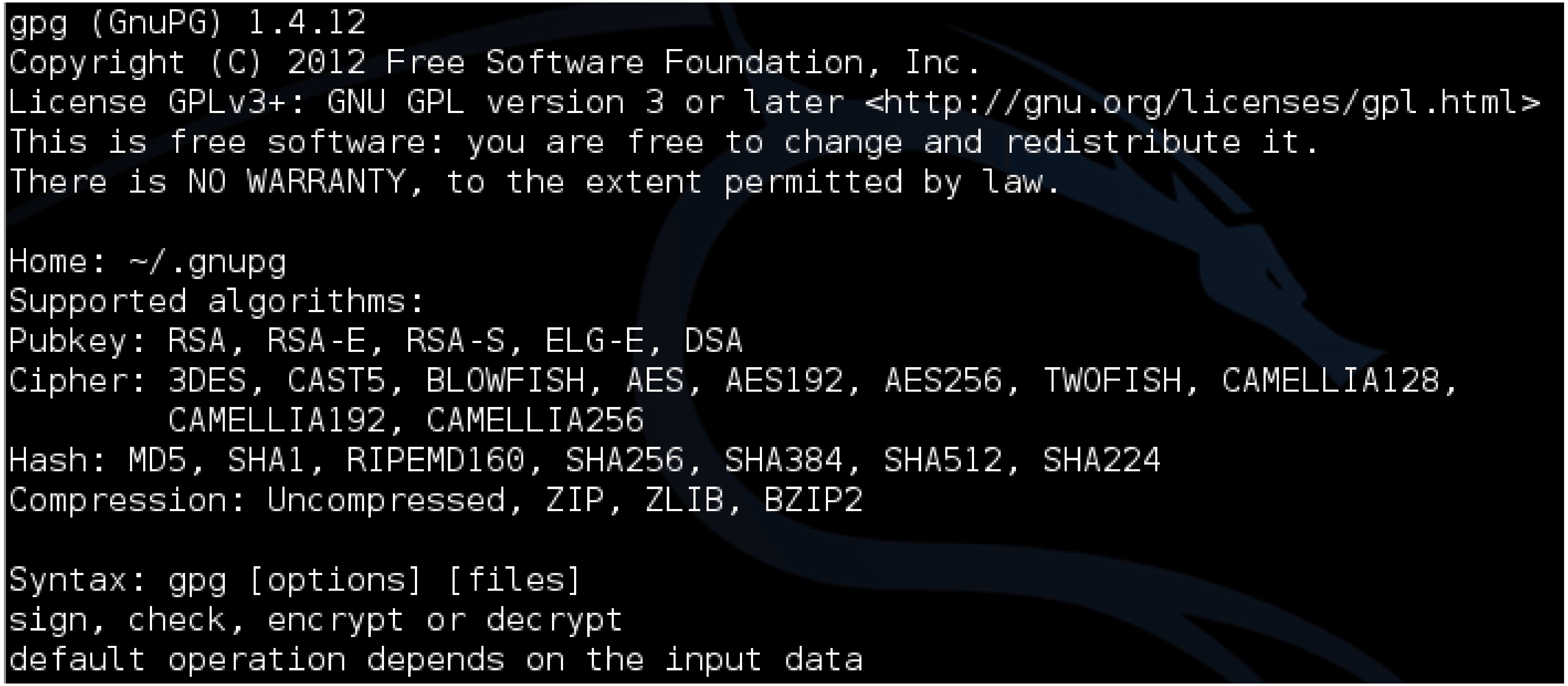
Private Ciphers

- All private (i.e., symmetric) key ciphers are combinations of *transposition* and *substitution*
- *Transposition* means moving message parts from one place in the message to another place
- *Substitution* means replacing message parts with other parts that are not explicitly part of the message
- Message parts can be characters, bytes, bits…

Private Key Symmetry


Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E[K, X]
X = D[K, Y]
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

GPG Lab



```
gpg (GnuPG) 1.4.12
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: ~/.gnupg
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA
Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH, CAMELLIA128,
        CAMELLIA192, CAMELLIA256
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2

Syntax: gpg [options] [files]
sign, check, encrypt or decrypt
default operation depends on the input data
```

GPG Lab

- Connect to RADISH and open a terminal in Kali
- Use a text editor (or **echo**) to create a message you would like to be kept secret and save the file
- Encrypt the file with **gpg** using the AES256 symmetric cipher
  - This will create a .gpg file with the ciphertext
- Decrypt the .gpg file to view the plaintext

GPG Lab

- If you feel confident, go ahead and perform the lab
- I will provide a few hints in the next two slides
- Please put in a good effort and don't view the solution slide until we get there as a class

GPG Lab Hints

- Enter **man gpg** to read the manual page on the GPG command
- You can search a manual page by hitting / and then typing in your query and hitting enter. Hit "n" for next result. Hit "q" to quit the manual page
- Find the part of the manual page that talks about "Encrypt with a symmetric cipher using a passphrase."
- Take some time and figure out the options to:
  - Encrypt your file with a symmetric cipher but avoid using the default CAST5 cipher and use AES256 instead
  - Use a secret passphrase of your choosing

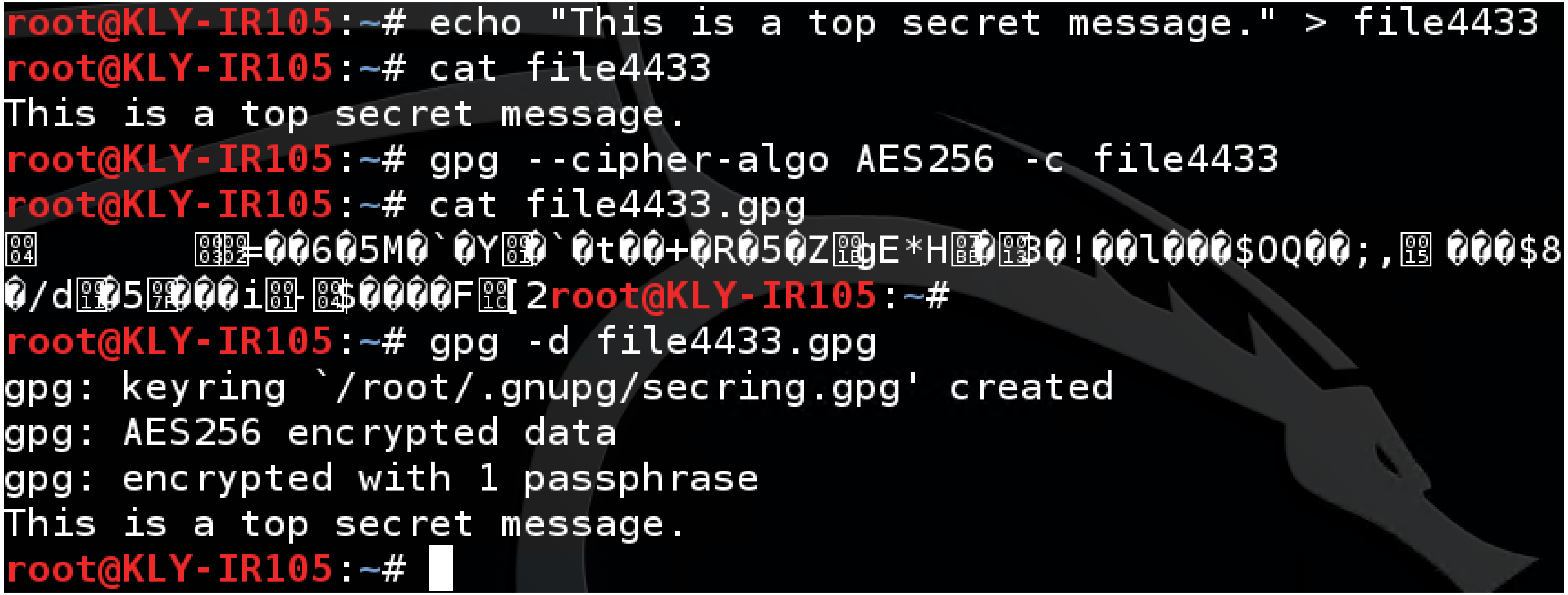
GPG Lab Hints

- Confirm your new .gpg file was encrypted by using the **cat** command to see if it contains ciphertext
- Then use the man page to figure out how to use GPG to "decrypt the file given on the command line" so that you can read the plaintext

GPG Lab

- How did you:
  - Encrypt the file with AES256?
  - Decrypt the file?

GPG - Solution



```
root@KLY-IR105:~# echo "This is a top secret message." > file4433
root@KLY-IR105:~# cat file4433
This is a top secret message.
root@KLY-IR105:~# gpg --cipher-algo AES256 -c file4433
root@KLY-IR105:~# cat file4433.gpg
0004        0302=��6�5M�`�Y01�`�t��+�R�5�Z1EgE*HBE133�!��l���$0Q��;,15 ���$8
�/d11�57F���i01-04$����F1C[2root@KLY-IR105:~#
root@KLY-IR105:~# gpg -d file4433.gpg
gpg: keyring `/root/.gnupg/secring.gpg' created
gpg: AES256 encrypted data
gpg: encrypted with 1 passphrase
This is a top secret message.
root@KLY-IR105:~# 
```

Private Key Crypto

- GPG has given you an example Private Key encryption
- The key is symmetrical since the same private key is used to encrypt and decrypt the message
- Let’s say you send the encrypted .gpg file to a friend
- Your friend receives the file but can’t decrypt it since they don’t have the symmetrical key

Private Key Crypto

- What is the big challenge with Private Key Crypto???

- What do you think the solution is???

Private Key Crypto

- The big challenge with Private Key Crypto is key distribution!
  - How to get the key to the other side with out it being intercepted.
- The solution is Public key crypto!
  - We will talk about this in the second half of today’s lecture

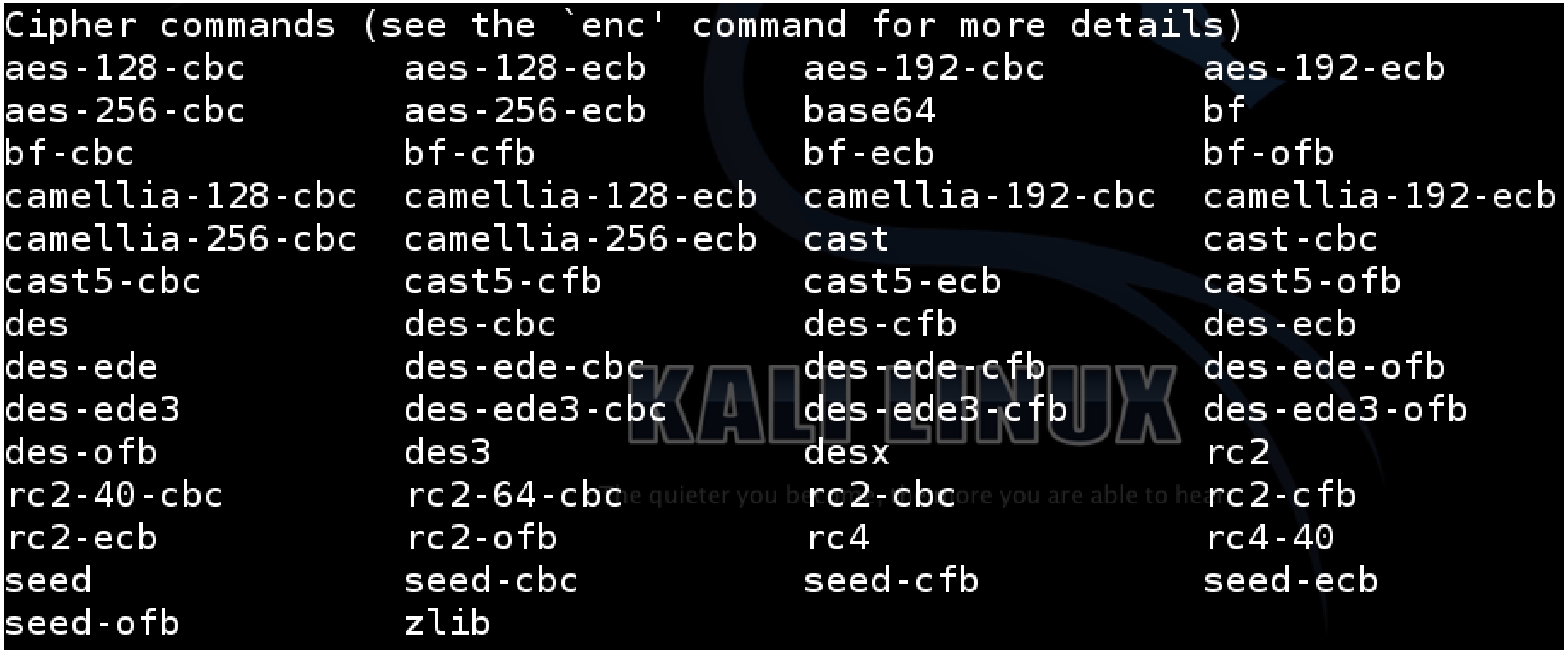
OpenSSL

- OpenSSL
  - Another command line tool providing crypto functions from the shell
  - Apache uses OpenSSL through the Mod_SSL interface

*Older versions of OpenSSL suffered from the Heartbleed attack

OpenSSL

- OpenSSL
  - Supports many ciphers:

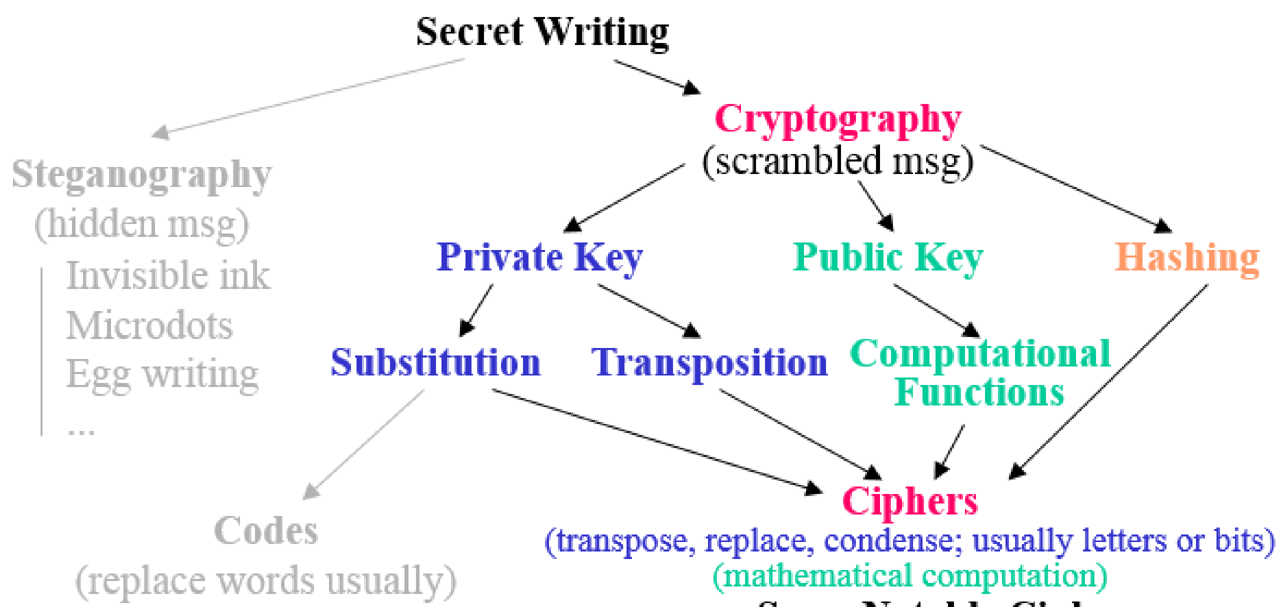


```
Cipher commands (see the `enc' command for more details)
aes-128-cbc       aes-128-ecb       aes-192-cbc       aes-192-ecb
aes-256-cbc       aes-256-ecb       base64            bf
bf-cbc            bf-cfb            bf-ecb            bf-ofb
camellia-128-cbc  camellia-128-ecb  camellia-192-cbc  camellia-192-ecb
camellia-256-cbc  camellia-256-ecb  cast              cast-cbc
cast5-cbc         cast5-cfb         cast5-ecb         cast5-ofb
des               des-cbc           des-cfb           des-ecb
des-ede           des-ede-cbc       des-ede-cfb       des-ede-ofb
des-ede3          des-ede3-cbc      des-ede3-cfb      des-ede3-ofb
des-ofb           des3              desx              rc2
rc2-40-cbc        rc2-64-cbc        rc2-cbc           rc2-cfb
rc2-ecb           rc2-ofb           rc4               rc4-40
seed              seed-cbc          seed-cfb          seed-ecb
seed-ofb          zlib
```

Steam & Block Ciphers

- We will now cover some of the common types of these ciphers

A Taxonomy



A Notable Code: Navajo natural language

Cryptography and **Ciphers** usually refer to the same things, with Cryptography being the science and Ciphers being the encoded entities. The only exception is **Codes.**

Private key ciphers usually use S and T.

Public key systems use CF.

Some Notable Ciphers:

Caesar (S)	Monoalphabetic (S)	**Stream**
Polyalphabetic (S)	One-time pad (S)	
Enigma (T&S)	Vigenére (S)	**Block**
DES (T&S)	RSA (T&S)	
IDEA	MD5 (H)	
RSA Pub (CF)	Diffie-Hellman (CF)	
Elliptical (CF)	SHA-1 (H)	
AES (T&S)		

S = substitution
T = transposition
H = hashing
CF = computational functions

Part V

Stream Ciphers

Stream Ciphers

- Processes messages a bit or byte at a time during encryption or decryption

- Think about relation to streaming music where you are receiving portions of the song as it plays

Stream Ciphers – 3 Common Types

1. Monoalphabetic Ciphers:
   - Uses substitution
   - Fixed alphabet during encryption process
   - Example: Caesar Cipher
2. Polyalphabetic Ciphers:
   - Uses substitution
   - Alphabet changes during encryption process
   - Example: Vigenére Cipher, Enigma Machine

Stream Ciphers – 3 Common Types (Cont.)

3. One-Time Pad (OTP)
   - Uses substitution
   - If used correctly, provides "Perfect Secrecy" and cannot be broken by cryptanalysis
   - Key is completely random and provides no information about the content of the plaintext
   - Key is also as long as the plaintext and is kept secret and is not ever reused

1. Monoalphabetic Substitution Cipher

- First recorded use is by Arabic historians
- Monoalphabetic substitution stream cipher
  - For each plaintext character substitute a ciphertext character
  - e.g., A→#, B→q, C→*, D→X, ...
- Here we have ***isomorphic*** substitution of one character for another
- Today easily deciphered by analyzing frequency of letters

Monoalphabetic Substitution Cipher (Cont.)

- An example of a Monoalphabetic Substitution Cipher is the Caesar Cipher
- Julius Caesar first documented its use
- Uses a fixed alphabet

Caesar Cipher

- Example:
  - Key: Shift 3 places toward the front of the alphabet
  - Thus `STEALTHEHOTDOG`
  - Is encrypted as `pqbxiqebelqald`
  - Today easily deciphered by analyzing frequency of letters
    - Letter frequency is sometimes called ***Side Information (SI)***

Caesar Breakdown

- Let's look at the first five letters of the previous example:
  - Plaintext = steal
  - Ciphertext = pqbxi
- Key = 23 (shifting three places forward in alphabet)

Caesar Breakdown

- What two ways could we figure this out???
  - Sliding alphabet
  - Modulus Math

Caesar Breakdown – Sliding Alphabet

- Sliding 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
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

← Key

STEAL = Plaintext

PQBXI = Ciphertext

Caesar Breakdown – Modulus Arithmetic

- Modulus Arithmetic
  - Encrypt: C=(P+k) mod 26
  - Decrypt: P=(C-k) mod 26

- C = Ciphertext
- P = Plaintext (in this case, the letter value [1-26])
- K = Key (in this case, the shift, which is 23)
- mod 26 since there are 26 letters in the alphabet

Caesar Encryption Breakdown

- Encrypt: C=(P+k) mod 26

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
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Plaintext		Ciphertext
▪S = 18	Encrypt (18 + 23) mod 26	15 = P
▪T = 19	Encrypt (19 + 23) mod 26	16 = Q
▪E = 04	Encrypt (04 + 23) mod 26	01 = B
▪A = 00	Encrypt (00 + 23) mod 26	22 = X
▪L = 11	Encrypt (11 + 23) mod 26	08 = I

Quick Refresher on Modulus Arithmetic

(18 + 23) mod 26 = 15

41 mod 26

$$\begin{array}{r} 1 \\ 26\overline{)41} \\ \underline{-26} \\ 15 \end{array}$$

(Remainder)

Caesar Decryption Breakdown

- Decrypt: P=(C-k) mod 26

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
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

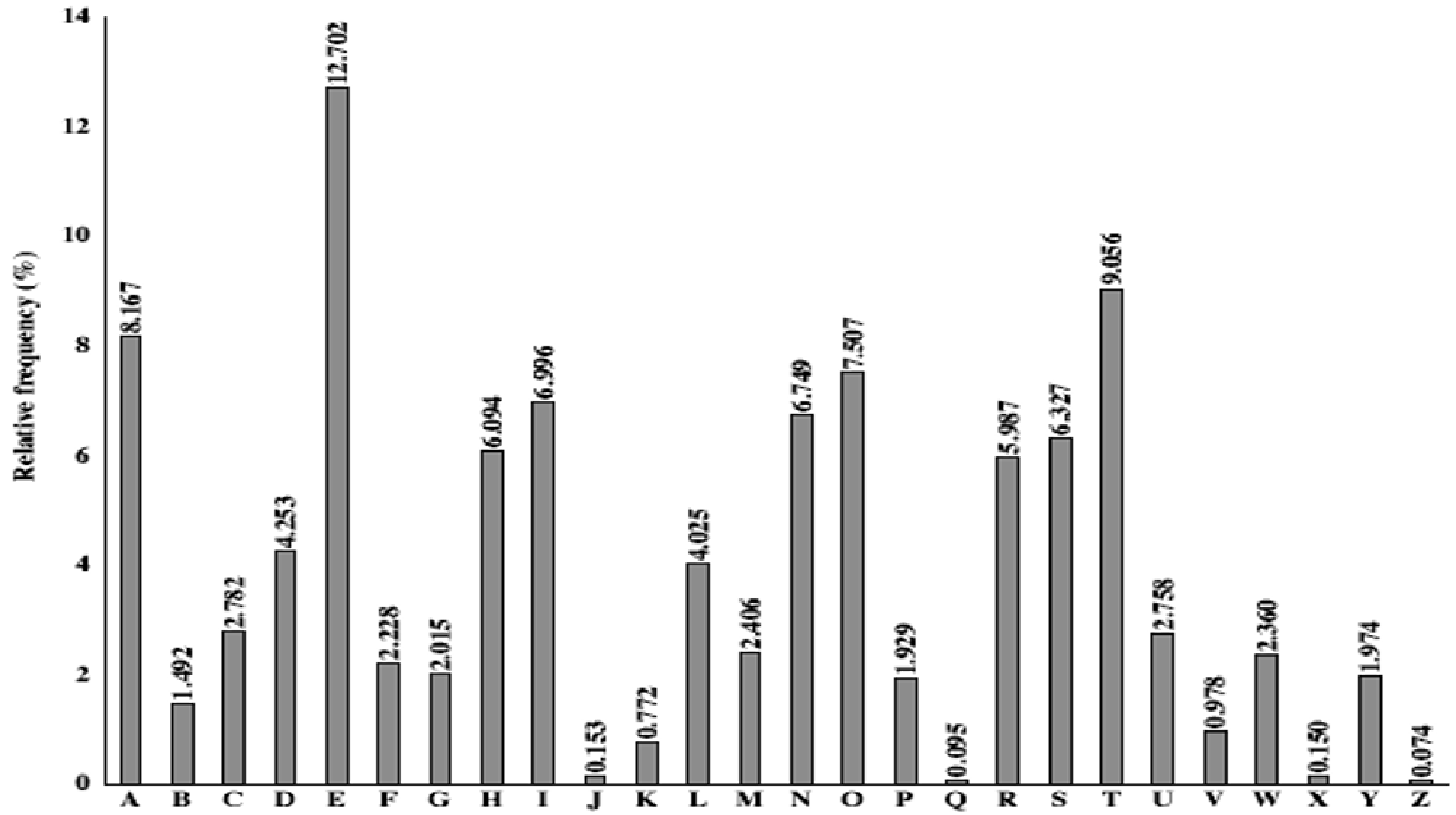
Ciphertext		Plaintext
▪ P = 15	Decrypt (15 - 23) mod 26	18 = S
▪ Q = 16	Decrypt (16 - 23) mod 26	19 = T
▪ B = 01	Decrypt (01 - 23) mod 26	04 = E
▪ X = 22	Decrypt (22 - 23) mod 26	00 = A
▪ I = 08	Decrypt (01 - 23) mod 26	11 = L

*(Must add 26 if difference is negative before calculating modulus)

Frequency Analysis

- A way to determine the key (shift) is through frequency analysis with a histogram
- First, determine standard relative frequency of text in the language in use
- For example, in English, the letter "E" is used most often
- If you analyze ciphertext and determine that "F" is used the most, then the shift may be 1 letter

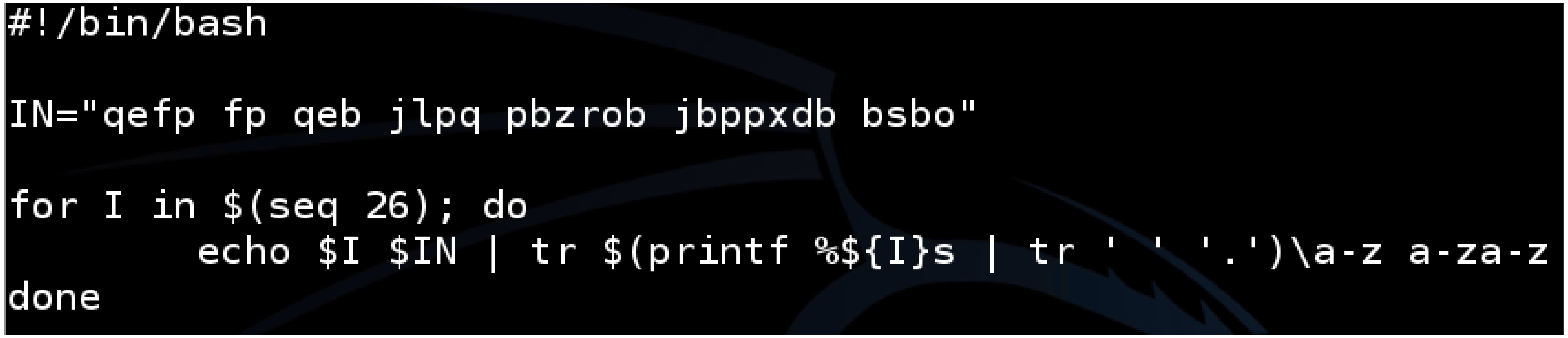
Relative Frequency of Letters in English Text



Caesar – Script

- Could also write a script to look at all of the available options
- Let's take this ciphertext for example:
  - qefp fp qeb jlpq pbzrob jbppxdb bsbo
- Found cool script online:
  - https://chris-lamb.co.uk/posts/decrypting-caesar-cipher-using-shell

Caesar – Script



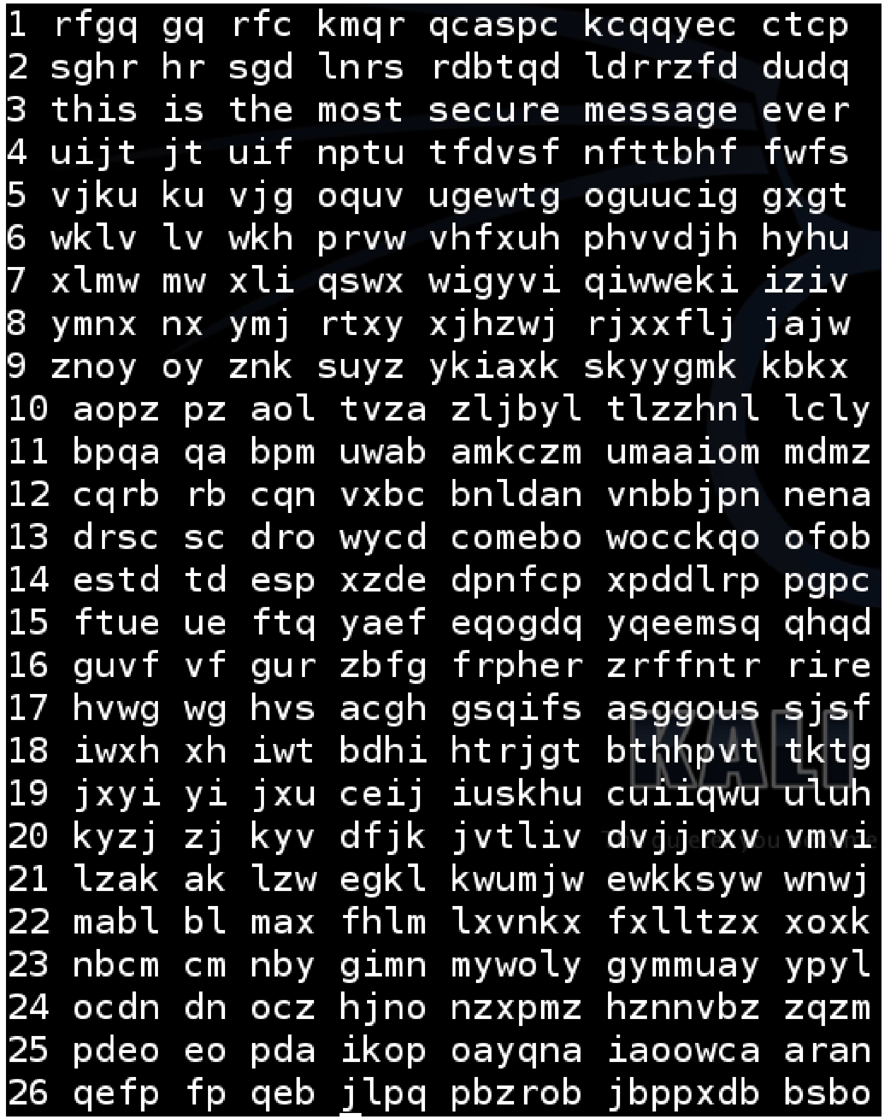
```
#!/bin/bash

IN="qefp fp qeb jlpq pbzrob jbppxdb bsbo"

for I in $(seq 26); do
        echo $I $IN | tr $(printf %${I}s | tr ' ' '.')\a-z a-za-z
done
```

- IN = ciphertext
- seq = 26 letters in alphabet

Caesar - Script



```
1 rfgq gq rfc kmqr qcaspc kcqqyec ctcp
2 sghr hr sgd lnrs rdbtqd ldrrzfd dudq
3 this is the most secure message ever
4 uijt jt uif nptu tfdvsf nfttbhf fwfs
5 vjku ku vjg oquv ugewtg oguucig gxgt
6 wklv lv wkh prvw vhfxuh phvvdjh hyhu
7 xlmw mw xli qswx wigyvi qiwweki iziv
8 ymnx nx ymj rtxy xjhzwj rjxxflj jajw
9 znoy oy znk suyz ykiaxk skyygmk kbkx
10 aopz pz aol tvza zljbyl tlzzhnl lcly
11 bpqa qa bpm uwab amkczm umaaiom mdmz
12 cqrb rb cqn vxbc bnldan vnbbjpn nena
13 drsc sc dro wycd comebo wocckqo ofob
14 estd td esp xzde dpnfcp xpddlrp pgpc
15 ftue ue ftq yaef eqogdq yqeemsq qhqd
16 guvf vf gur zbfg frpher zrffntr rire
17 hvwg wg hvs acgh gsqifs asggous sjsf
18 iwxh xh iwt bdhi htrjgt bthhpvt tktg
19 jxyi yi jxu ceij iuskhu cuiiqwu uluh
20 kyzj zj kyv dfjk jvtliv dvjjrxv vmvi
21 lzak ak lzw egkl kwumjw ewkksyw wnwj
22 mabl bl max fhlm lxvnkx fxlltzx xoxk
23 nbcm cm nby gimn mywoly gymmuay ypyl
24 ocdn dn ocz hjno nzxpmz hznnvbz zqzm
25 pdeo eo pda ikop oayqna iaoowca aran
26 qefp fp qeb jlpq pbzrob jbppxdb bsbo
```

CrypTool Lab - Caesar

- Follow along
  - Encrypt using Caesar cipher with shift of 2 (letter C)
  - Check out histogram before and after
  - Use Auto-Decryption option under Analysis

- When done, close all windows except the first one (startingexample-en)

2. Polyalphabetic Substitution Cipher

- Vigenère stream cipher is good example

*(Alphabet changes during encryption)

- The ***key*** is some word or string of characters or numbers – say the word "WHITE"
  - Align **A** with **W**: Then **S→O** to encrypt 1st letter
    - **A**BCDEFGHIJKLMNOPQR**S**TUVWXYZ
    - **W**XYZABCDEFGHIJKLMN**O**PQRSTUV
  - Align **A** with **H**: Then **T→A** to encrypt 2nd letter
    - **A**BCDEFGHIJKLMNOPQRS**T**UVWXYZ
    - **H**IJKLMNOPQRSTUVWXYZ**A**BCDEFG
  - **…**

Polyalphabetic Substitution Cipher

- Thus **STEALTHEHOTDOG**
- Is encrypted as **oamtppomaspkwz**
- Key: **WHITEWHITEWHIT**

*(Notice key is repeated if plaintext is longer than the key)

- Here strong letter frequency is reduced. Why?
- Analysis is more difficult but still quite possible

Hand’s On CrypTool - Vigenère

- Follow along.
  - Encrypt
  - Set key as BRIA
  - Put portion of encrypted text into new file
  - Show how key detection doesn’t always find the correct number of characters in the key
    - May take trial and error

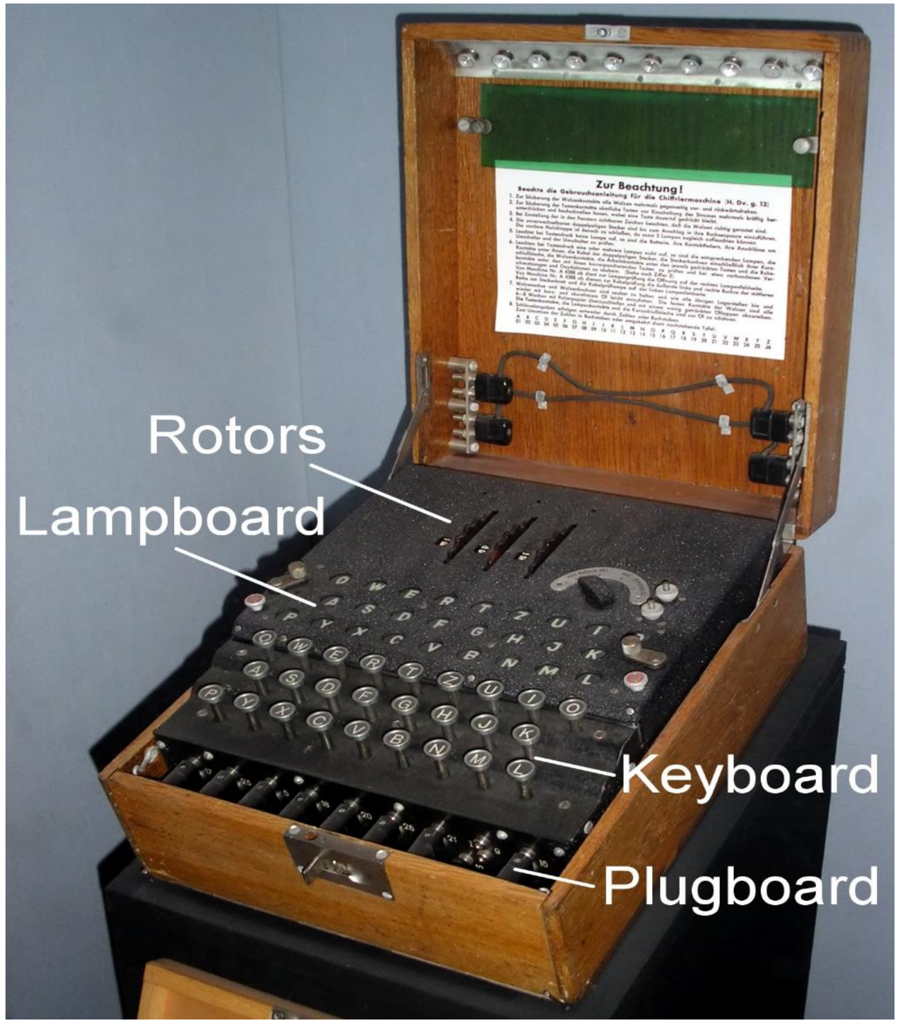
Enigma Machine

- Considered a polyaphabetic substitution cipher but uses some transposition as well
- Featured in “The Imitation Game” film about Alan Turing who eventually cracked Enigma

Enigma Machine

- Poland gave the information to the British and French before the invasion of Poland in Sept. 1939
  - So for the early part of WWII, the allies could read all the German and Italian encrypted messages
- But eventually the Germans started picking random keys
- British (especially Alan Turing) developed ability to decrypt German encryption that used random keys
  - British even built their own Enigma machines
  - Bletchley Park
- Breaking Enigma was a matter of very sophisticated frequency analysis (SI)

Enigma Machine



3. One-Time Pad (OTP)

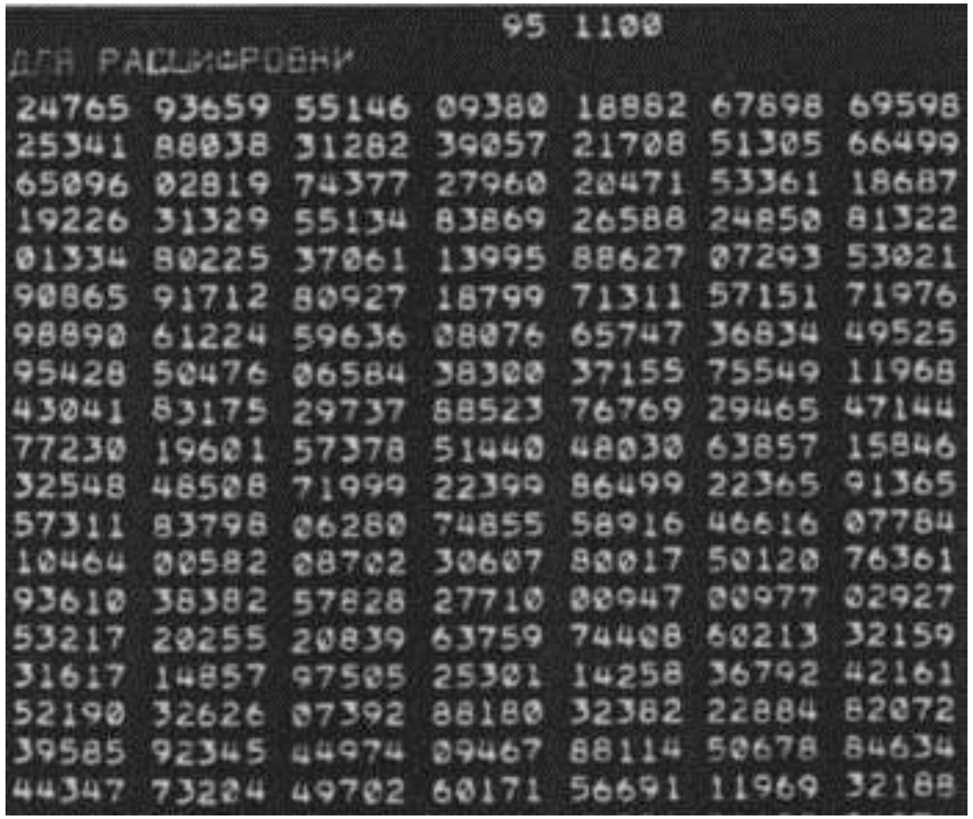
- Invented by banker Frank Miller and further developed by Gilbert Vernam.
- OTP is also referred to as the Vernam Cipher

One-Time Pad (OTP) (Cont.)

- Plaintext combined with a completely random key which makes cryptanalysis impossible
- Key must not be generated by a simple computer function (which are often not random)
- Key can only be used once
- Key and plaintext are calculated with mod10, mod26, or mod2
- Two copies of each key for sender and recipient
  - Often code books are used

OTP Example

- Aleksandr Ogorodnik was a Soviet diplomat that spied on the Soviet Union for the CIA
- Example of one of his OTPs:



95 1100

ДЛЯ РАСШИФРОВКИ

24765	93659	55146	09380	18882	67898	69598
25341	88038	31282	39057	21708	51305	66499
65096	02819	74377	27960	20471	53361	18687
19226	31329	55134	83869	26588	24850	81322
01334	80225	37061	13995	88627	07293	53021
90865	91712	80927	18799	71311	57151	71976
98890	61224	59636	08076	65747	36834	49525
95428	50476	06584	38300	37155	75549	11968
43041	83175	29737	88523	76769	29465	47144
77230	19601	57378	51440	48030	63857	15846
32548	48508	71999	22399	86499	22365	91365
57311	83798	06280	74855	58916	46616	07784
10464	00582	08702	30607	80017	50120	76361
93610	38382	57828	27710	00947	00977	02927
53217	20255	20839	63759	74408	60213	32159
31617	14857	97505	25301	14258	36792	42161
52190	32626	07392	88180	32382	22884	82072
39585	92345	44974	09467	88114	50678	84634
44347	73204	49702	60171	56691	11969	32188

OTP / Engima

- If the Germans had used One-Time Pads in the Enigma machines, their communications would not have been cracked which might have altered the course of WWII.

Part VI

Block Ciphers

Block Ciphers

- Block ciphers process messages into blocks, each of which is then en/decrypted
- Sort of like a substitution on very big characters
  - 64-bits or more

Block Cipher Principles

- Most symmetric block ciphers are based on a **Feistel Cipher Structure** *(Horst Feistel)*
- Feistel's work was based heavily upon the work of Claude Shannon
- Block ciphers allow efficient decryption of ciphertext

Block Cipher Principles

- Block ciphers look like an extremely large substitution
  - Substitute one block at a time
- Note: There are 2^{64} possible encryptions for a 64-bit plaintext block
  - ~ 18,000,000,000,000,000,000 or 18 billion billion

Claude Shannon And Substitution-Permutation Ciphers

- In 1949 Claude Shannon (Bell Labs) introduced idea of substitution-permutation (S-P) networks
  - A "father" of information theory
  - Modern substitution-permutation product cipher
    - *Communication Theory of Secrecy Systems*, BSTJ 1949
- These form the basis of modern block ciphers

Claude Shannon And Substitution-Permutation Ciphers

- S-P networks are based on the two primitive cryptographic operations we have seen before:
  - Substitution (S-box) (substitution)
  - Permutation (P-box) (transposition)
- Provide ***confusion*** and ***diffusion*** of message

Confusion and Diffusion

- Ciphers need to completely obscure the statistical properties of original message
  - (A one-time pad does this)
- More practically, Shannon suggested combining elements to obtain:
- **Confusion**
  - Make relationship between the **ciphertext** and the **key** as complex as possible
- **Diffusion**
  - Dissipate the statistical structure of **plaintext** over the entire **ciphertext**

Feistel Cipher Structure

- Horst Feistel devised the ***feistel cipher***
- Partition plaintext input block into two halves
- Process each block through multiple rounds which
  - Perform a substitution on left data half of a block based on a function of right half & a subkey
  - Then have a permutation by swapping halves
- This scheme implements Shannon's substitution-permutation concept

Feistel Cipher Design Principles

- **Block size**
  - Increasing size improves security, but slows cipher
- **Key size**
  - Increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **Number of rounds**
  - Increasing number improves security, but slows cipher

Feistel Cipher Design Principles

- **Subkey generation**
  - Greater complexity can make analysis harder, but slows cipher
- **Round function**
  - Greater complexity can make analysis harder, but slows cipher
- **Fast software en/decryption & ease of analysis**
  - These are more recent concerns for practical use and testing

Well Known Block Ciphers

1. DES (Data Encryption Standard)
2. 3DES (Triple Data Encryption Standard)
3. AES (Advanced Encryption Standard)
4. IDEA (International Data Encryption Algorithm)

1. DES (Data Encryption Standard)

- Message is a binary bit string
- Message is broken into 64-bit blocks
- Each block encrypted using 56-bit secret key
  - Arranged as eight 7-bit pieces

DES (Data Encryption Standard)

- Artificially expanded to 64 bits by adding one parity bit to each piece of the key
  - Compromise with NSA who was worried about not being able to decrypt messages
- $C = E_k[P]; \quad P = D_k[C] = D_k[E_k[P]]$
  - E and D are closely related
  - K is the same for encryption and decryption

DES (Data Encryption Standard)

- Complex algorithm on each block
  - 16 rounds of encipherment/decipherment on each block
  - 56-bit key used to generate 16 sub keys of 48 bits each
  - Each round uses one 48-bit sub key derived from the 56-bit key
- Process is symmetric: almost same process for encoding and decoding
  - In decipherment, subkeys are used in reverse order

DES 56-bit Key Too Short

- Considering only exhaustive attacks
  - 56 bit key yields 2^{56} or 7.2×10^{16} keys
- In 1998 the Electronic Frontier Foundation's *DES Deep Crack* machine demonstrated 90×10^{9} DES keys per second
  - Was able to crack a DES encoded message in 9.2 days
- In 1999 a machine was demonstrated to test 2.5×10^{11} keys per second (called Deep Crack Plus)
  - Was able to crack a DES encoded message in 3.3 days
- Guess what NSA can do today!!

Strength of DES – Timing Attacks

- Attacks the actual implementation of the cipher
- Uses knowledge of consequences of implementation to derive knowledge of some/all subkey bits
- Specifically uses the fact that calculations can take varying times depending on the value of inputs to it
- Particularly effective on smartcards

Strength of DES – Analytic Attacks

- Today several analytic attacks on DES exist
- These utilize deep structures of the cipher
  - By gathering information about encryptions
  - Can eventually recover some/all of the sub-key bits
  - If necessary then exhaustively search for the rest
- Generally these are statistical attacks include
  - Differential cryptanalysis
  - Linear cryptanalysis
  - Related key attacks

Modes of Operation

- Block ciphers encrypt fixed size blocks
  - eg. DES encrypts 64-bit blocks, with 56-bit key
- If the plaintext consists of an arbitrary amount of information to encrypt, how do we use block ciphers?
- Four mechanisms were defined for DES in ANSI standard ***ANSI X3.106-1983 Modes of Use***
- Increased to 5 mechanisms for DES and AES

Modes of Operation

- Basically, if the plaintext consists of data exceeding a single 64 bit block, these mechanisms either chain the blocks together or evaluate each individual block during encryption.

Modes of Operation

- ECB: Electronic Code Book
  - Each block of plaintext is encoded independently
- CBC: Cipher Block Chaining
  - Blocks are linked together in the encryption operation
- CFB: Cipher FeedBack
  - Combination of stream and block-chaining encryption

Modes of Operation

- OFB: Output FeedBack
  - Combination of stream and block-chain encryption but plaintext is XORed to the output of the block cipher in order to get the cipher text block.
- CTR: CounTeR
  - Similar to OFB but encrypts counter value rather than any feedback value
  - Used for high-speed network encryptions

Hand's On CrypTool – DES (ECB)

- Follow along
- 64 bit kit
  - 5 bytes (40 bits) can crack in around 20 seconds
  - 6 bytes (48 bits) can crack almost instantly

2. Triple DES

- 3DES
  - Uses the same algorithm as DES
  - Applies the algorithm three times
  - Uses a different key each time
- $C = E_{k1}[D_{k2}[E_{k3}[P]]]$
- Effectively a key of 168 bits - maybe???
- Pros & Cons
  - Uses same software as DES
  - Block size is smallish (64 bits), resulting in insecurities
  - Three times a slow as DES

3. AES (Advanced Encryption Standard)

- A replacement for DES was clearly needed
  - Theoretical attacks exist that can break it
  - Exhaustive key search attacks have been demonstrated
- Can use Triple-DES – but slow and has small blocks
- US NIST issued call for ciphers in 1997

AES (Advanced Encryption Standard)

- 15 candidates accepted in Jun 98
- 5 were short-listed in Aug-99
  - Serpent - Ross Anderson, Eli Biham, Lars Knudsen
  - Twofish - Bruce Schneier et al
  - Rijndael - Vincent Rijmen, Joan Daemen
  - MARS - Don Coppersmith (IBM). Derived from Feistel
  - RC6 - Ron Rivest (Rivest Cipher 6)

AES History

	Rijndael	Serpent	Twofish	MARS	RC6
Ease of Implementation	Good	Good	OK	Low	Low
Hardware Performance	High	High	OK	Low	OK
Software Performance	High	Low	Low	Medium	Medium
Embedded Sys, CCs, Smart Cards, SIMS…	Good	Good	OK	Low	Low
Overall Design	OK	Poor	Good	OK	Poor
Overall Security	OK	Good	Good	Good	OK

- Rijndael's *(RhineDahl)* proposal was selected as the AES in Oct-2000
- Issued as FIPS PUB 197 standard in Nov-2001

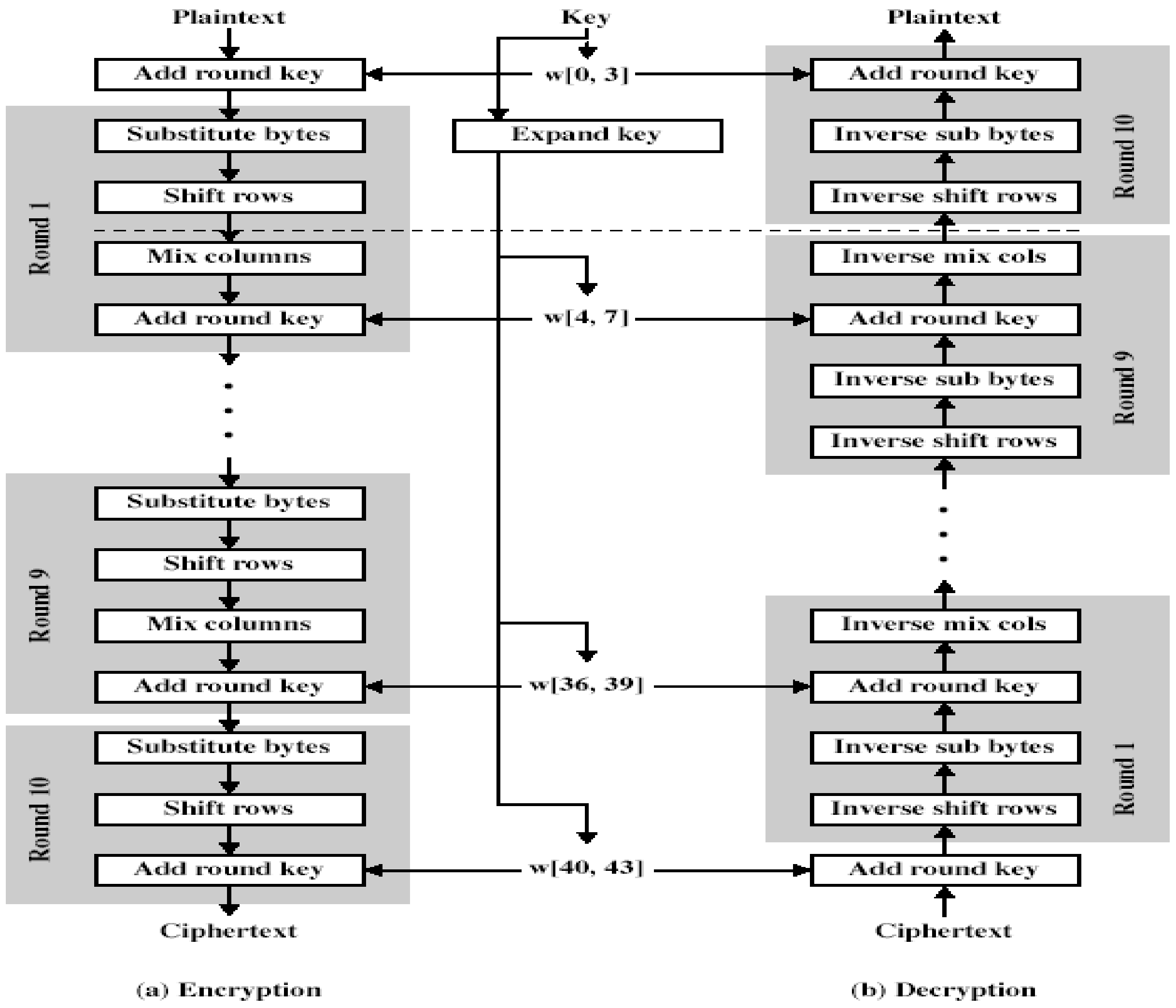
AES Details

- AES is Advanced Encryption Standard
- 128-bit block size
- Keys can be 128, 192, or 256 bits
- Approved by NIST in November 2001
- Today used all over
  - DES and 3DES not used too much any more

AES Details

- Encryption is accomplished as a series of "rounds"
  - The number of rounds varies
    - 10 rounds for 128-bit keys
    - 12 rounds for 192-bit keys
    - 14 rounds for 256-bit keys
- The key is used to create a set of 44 32-bit words, **w[i]**
- Four of these words (128 bits) are used as the key for each round of processing

AES Process Diagram for 128-bit Keys



AES Vulnerabilities

- AES-128 and its variants have been shown to be vulnerable
  - Side Information Attack
    - If software runs on same system as AES code, gets key in a few seconds
  - Others
- AES-192 & 256 Vulnerability
  - Related Keys Attack
    - Monitor ciphertext created by several different but mathematically related keys

4. IDEA

- IDEA is International Data Encryption Algorithm
- First 128-bit key widely used
  - Used by browsers and PGP
- Developed by Swiss
- Uses 64-bit block size
- Can use 4 different block chaining methods
- Used in several public encryption schemes that are applied to email

Checkpoint

- Steganography
- Cryptography
- Substitution (Coding)
- Transposing
- Hashing
- Plaintext vs Ciphertext
- Goals of Cryptographic Ciphers

Checkpoint (Cont.)

- Symmetric Key Crypto
- GPG
- Stream Ciphers
  - Caesar
  - Vigenere
- Block Ciphers
  - DES, 3DES, AES, IDEA
  - Modes of Operation – ECB, CBC, CFB, OFB, CTR

Homework

- See end of Crypto II Lecture slides

Conclusion

- We have now covered private key cryptography as well as codes and popular stream and block ciphers
- Any questions before we move on to public key cryptography?