

# CSCD 303

## Fall 2017



## Lecture 20

# Cryptography - Symmetric

# Symmetric Cipher Families



- Involve using one key for both encryption and decryption
- Symmetric modern crypto systems have two broad families of methods
  - Stream ciphers
  - Block ciphers

# Block vs. Stream Cipher

- **Block ciphers**



- Take text, divide it into blocks, and encrypt those blocks
- **Important part** **entire** text needed before can start encrypting

- **Stream Ciphers**



- Stream ciphers, treat their input as a **data stream** and encrypt it on the fly
- Don't need entire blocks, and don't need entire set of data before they can start encrypting

# Stream Cipher

- Processes Message bit by bit (as a stream)
  - Most famous of these is Vernam cipher
    - Also called **one-time pad**
  - Invented by Vernam, working for AT&T, in 1917
  - Simply add bits of message to random key bits
  - Need equal key bits to message bits, difficult in practice
  - Unconditionally secure provided key is **truly random**
  - Difficult to distribute so much key
    - For long message, need lots of key bits
    - Idea to generate keystream from a smaller (base) key
    - Key is expanded to create the keystream
  - Use some pseudo-random function to do this

# Stream Cipher



## Basic Idea

- Generate pseudorandom sequence of bytes called a **keystream**
- Combined with data using XOR

XOR combines two bytes to get one by exclusive or'ing each bit

00110101 XOR 11100011 = 11010110

Characteristic of XOR – apply same value twice, get original value

# Stream Ciphers



- Stream cipher similar to one-time pad
  - Difference, one-time pad is random
  - Stream cipher is pseudo random
  - Encryption should have large period of randomness
    - Longer period, more difficult to perform cryptanalysis
- RC4 based on pseudo random numbers
  - Look at it as an example of Stream Cipher
- Very simple algorithm, well known

# Stream Cipher – RC4 Example



- RC4 was designed by Ron Rivest of RSA Security in 1987
  - Officially termed **Rivest Cipher 4**
    - Alternatively understood to stand for “**Ron's Code**”
  - RC4 was initially trade secret, but 1994 description was **anonymously** posted to Cypherpunks mailing list
    - Soon posted to many sites on Internet
  - Leaked code was confirmed to be genuine as its output was found to match that of proprietary software using licensed RC4
- Used in SSL and TLS, WEP and WCA

<http://en.wikipedia.org/wiki/RC4>

# RC4 Algorithm Uses XOR

- XOR

Same key



00110101 XOR 11100011 = 11010110

11010110 XOR 11100011 = 00110101

A XOR B = C, C XOR B = A

Encryption use of RC4

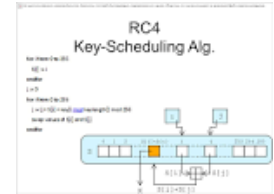
Same  
↓

Plaintext XOR Keysequence = Ciphertext

Ciphertext XOR Keysequence = Plaintext



# RC4 – How Does it Work?



The RC4 cipher consists of two parts:

1. The Key Scheduling Algorithm (KSA), and
2. The Pseudo Random (Byte) Generation Algorithm (PRGA).

The KSA takes a neatly arranged array of 256 elements (has values 0, 1, 2, ..., 255 in order)

Uses a variable length secret key to turn the array into a pseudo-random order

# RC4 – How Does it Work?

- Once the KSA has finished, the array is supposed to "look" randomly arranged.
- After the KSA, the PRGA part starts
- This part outputs one byte at a time.
- Each PRGA step further perturbs the array a little while outputting one byte

# RC4 Key Setup

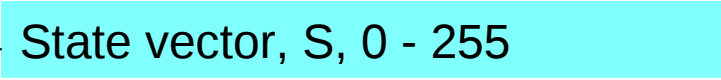
## Key setup Algorithm

**Inputs:** Typically, 5 to 32 bytes of key, key, stored in array, **s**  
char s[256]

for i from 0 to 255

S[i] := i

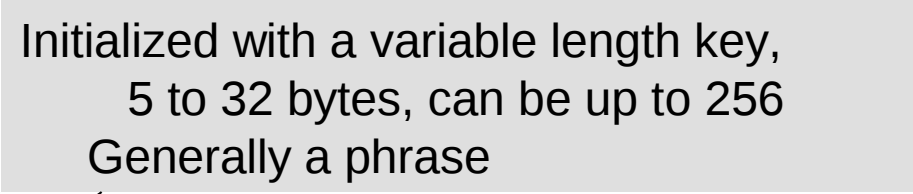
State vector, S, 0 - 255



endfor

j := 0

Initialized with a variable length key,  
5 to 32 bytes, can be up to 256  
Generally a phrase



for i from 0 to 255

j := (j + S[i] + key[i mod keylength]) mod 256

swap(&S[i], &S[j])

endfor

# RC4 Key Setup

- **What this does ....**
  - The RC4 key setup initializes the internal state, **S**, using a key **K** of up to 256 bytes
  - By exchanging two elements of the state in each step, it incrementally transforms the original array into a "random" permutation

# Pseudo-random generation algorithm (PRGA)

$i := 0$

$j := 0$

while GeneratingOutput:

$i := (i + 1) \bmod 256$

$j := (j + S[i]) \bmod 256$

swap(&S[i], &S[j])

output  $S[(S[i] + S[j]) \bmod 256]$

endwhile

Byte produced is XOR'd with plaintext = Ciphertext

For as many iterations as are needed, the PRGA modifies state and outputs a byte of the keystream

In each iteration, the PRGA increments  $i$ , adds the value of  $S$ , exchanges the values of  $S[i]$  and  $S[j]$ , and then outputs the value of  $S$  at the location  $S[i] + S[j]$  (modulo 256)

Each value of  $S$  is swapped at least once every 256 iterations

# RC4 Problems

- In 2001, a discovery was made by Fluhrer, Mantin and Adi Shamir
  - All possible RC4 keys, statistics for first few bytes of output keystream are strongly non-random, leaking information about key
  - Implementations use an initialization vector (IV) of limited size, implemented by concatenating key with IV, supposed to increase randomness
    - Long-term key can be discovered by analyzing a large number of messages encrypted with this key

[http://www.drizzle.com/~aboba/IEEE/rc4\\_ksaproc.pdf](http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf)

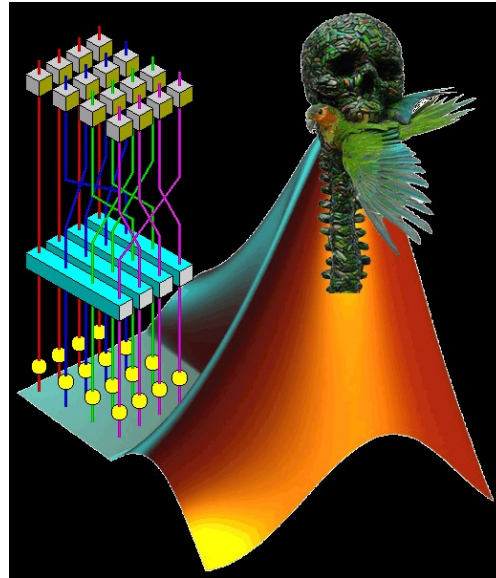
# Klein's Attack on WEP

- In 2005, Andreas Klein presented an analysis of RC4 stream cipher showing more correlations between RC4 keystream and key
  - Researchers used this analysis to create **aircrack-ptw**,
  - Paper on this: <http://eprint.iacr.org/2007/120.pdf>
  - Tool which cracks 104-bit RC4 used in 128-bit WEP in under a minute!!!!
  - Whereas Fluhrer, Mantin, and Shamir attack used around 10 million messages, **aircrack-ptw** can break 104-bit keys in 40,000 frames with 50% probability, or in 85,000 frames with 95% probability

# Final Comment RC4

- Problem is not with the algorithm itself
- Particularly with keys that are long enough
- Problem is with the generation of keys as input to RC4!
- Read Wikipedia page at end of slides
  - Tried to fix the algorithm
    - RC4-drop – drops the first few bytes
    - Improvements to RC4 by Ron Rivest - Called **Spritz**

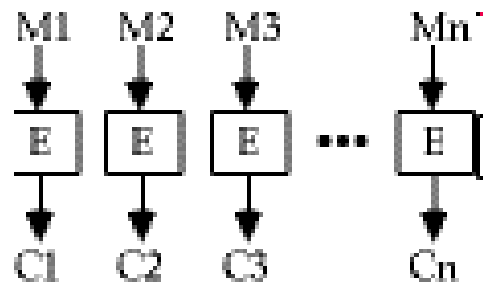




# Block Ciphers

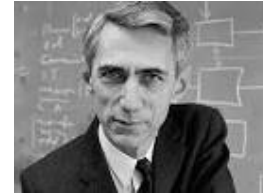
# Block Cipher

- In block cipher message is broken into fixed size blocks, each of which is then encrypted
- Most modern ciphers are of this form
- Contrasts with stream ciphers which encrypt individual bits



Block ciphers are based on information theory of Claude Shannon ...

# Shannon's Theory



- Claude Shannon wrote key papers on modern cryptology theory in 1949
- Developed concepts of:
  - **Entropy of a message – Variability**
  - **Redundancy in language,**
  - Theories about how much information is needed to break cipher
  - Defined concepts of computationally secure vs. unconditionally secure ciphers

# Shannon's Theory

- Substitution-Permutation Ciphers

- In his 1949 paper Shannon introduced idea of substitution-permutation (S-P) networks, which now form basis of modern block ciphers  
<http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf>
- An **S-P** network is modern form of a substitution-transposition product cipher
- **S-P** networks are based on two primitive cryptographic operations
  - Substitution
  - Permutation

A **combination** is an arrangement of items in which **ORDER DOES NOT MATTER.**

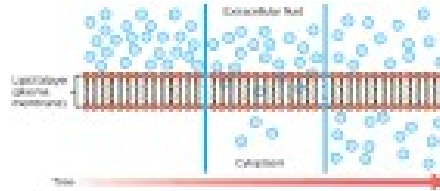
A **permutation** is an arrangement of items in a particular order. Notice, **ORDER MATTERS!**

# Shannon's Theory

- **Substitution-Permutation Network**
  - Shannon noted that two weak but complementary ciphers can be made more secure by applying them together
  - Combined these two primitives in a structure called **product cipher**
- **S-Boxes**
  - Provide **confusion** of input bits
- **P-Boxes**
  - Provide **diffusion** across S-box inputs

# Diffusion and Confusion

- **Diffusion**



- Dissipates statistical structure of plaintext over bulk of ciphertext

- **Confusion**



- Makes relationship between ciphertext and key as complex as possible

# Diffusion and Confusion



- Introduced by Shannon to thwart cryptanalysis based on statistical analysis
  - Assume attacker has some knowledge of statistical characteristics of plaintext
- Cipher needs to completely obscure statistical properties of original message
- One-time pad also does this

# Implementing S-P Networks



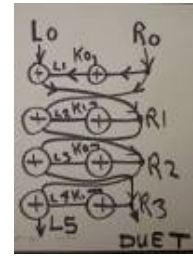
- **Horst Feistel**, working at IBM Research Labs devised this structure in early 70's, which we now call a **feistel cipher**
- **Implemented Feistel Structure**
  - Identical rounds of processing
  - Each round, substitution is performed on  $\frac{1}{2}$  of data
  - Then, permutation, exchanges halves
  - Original key is expanded and a different subkey is used for each round



# Feistel Cipher Design Principles

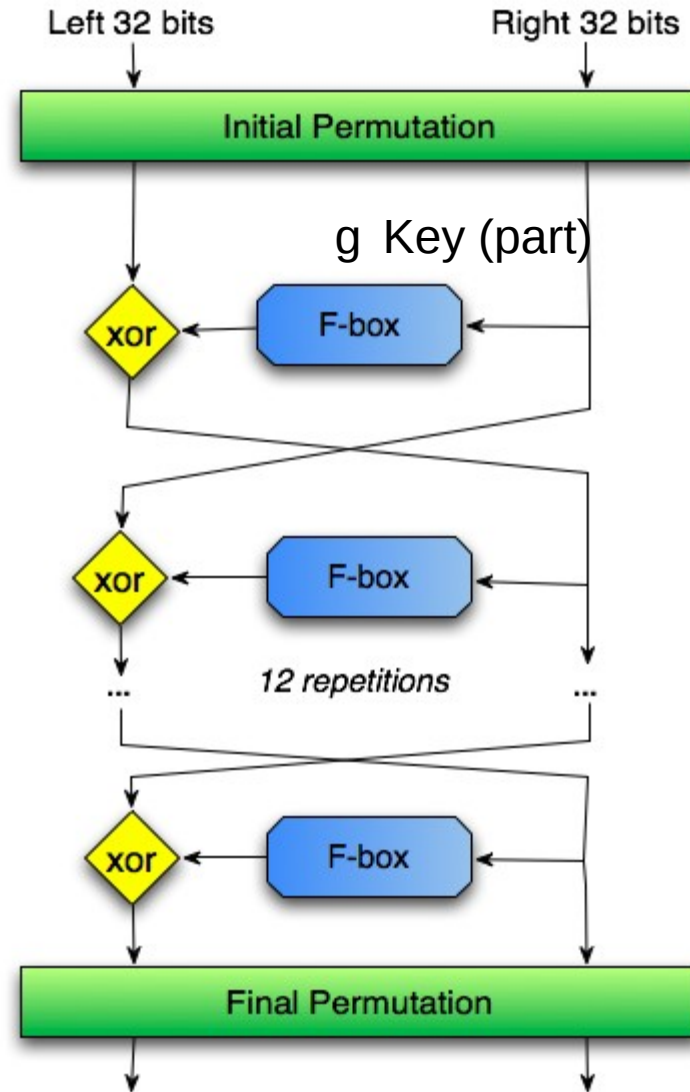
- **Block size**
  - Increasing size improves security, but slows cipher
- **Key size**
  - Increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **Number of rounds**
  - Increasing number improves security, but slows cipher
- **Subkey generation**
  - Greater 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**
  - More recent concerns for practical use and testing

# Feistel Cipher Structure



- Idea is to partition input block into two halves,
  - $L(i-1)$  and  $R(i-1)$ ,
  - Exchange blocks at each round,  $i$
- Have a Function,  $g$ , controlled by part of key  $K(i)$  and XOR'd to pass on to next stage

# Feistel Cipher Structure



# Feistel Cipher Structure

- This can be described functionally as:

$$L(i) = R(i-1)$$

$$R(i) = L(i-1) (+) g(K(i), R(i-1))$$

- In practice link a number of these stages together (typically 16 rounds) to form full cipher
- Feistel structure advantage is encryption and decryption operations are similar, even identical in some cases, requiring only reversal of key schedule
- Therefore size of code or circuitry required to implement such a cipher is nearly halved
- Used in DES ...

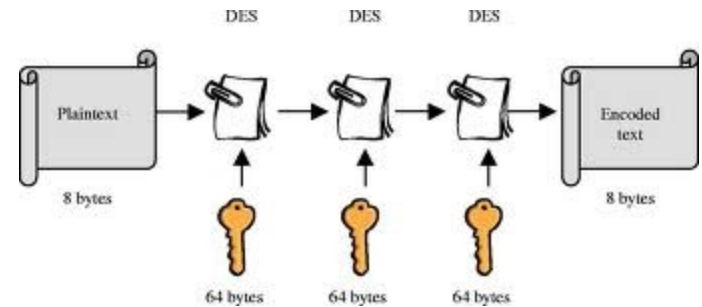
# DES Algorithm



- **DES**

- In 1970's US National Bureau of Standards (NIST) recognized general public needed secure encryption technology to protect sensitive information
- Historically, US DOD has strong interest in encryption systems (NSA)
- In 1972, NBS (NIST) issued call for proposals for producing a public encryption algorithm

# DES Algorithm



- Specified following criteria for an encryption algorithm:
  - High level of security
  - Easy to understand
  - Publishable, security does not depend on secrecy of the algorithm
  - Available to all users
  - Efficient to use
  - Exportable – means it couldn't be too secure!!!

# DES Algorithm



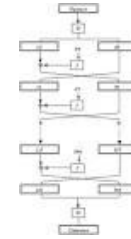
- Few Organizations responded to NBS call
- Second announcement in 1974
  - IBM responded with Lucifer algorithm
  - Lucifer used simple logical operations on relatively small quantities
  - Algorithm could be implemented in either hardware or software on conventional computers
  - Final algorithm was developed by IBM for NBS
    - Became known as DES
    - Stands for **Data Encryption Standard**

# DES Algorithm

- NSA analyzed DES algorithm
  - Found no serious flaws
- Became a standard in 1976
  - Authorized for use by all public and private sector unclassified communication
- Eventually, DES was accepted as an international standard



# DES Algorithm



- Overview of DES

- DES performs bit permutation, substitution, and recombination operations on blocks containing 64 bits of data and 56 bits of key
- 64 bits of input are permuted initially, and then input to a function using
  - Static tables of permutations **P-boxes**
  - and substitutions **S-boxes**

# S-box

- **S-box takes number of input bits,  $m$** 
  - Transforms them into number of output bits,  $n$
  - An  **$m \times n$  S-box** can be implemented as a lookup table with  $2^m$  words of  $n$  bits each
  - Fixed tables are normally used, DES

# S-box

- One good example is this 6×4-bit S-box from DES (S5):

See Wikipedia for this example

[http://en.wikipedia.org/wiki/Substitution\\_box](http://en.wikipedia.org/wiki/Substitution_box)

# DES Algorithm



Shifts left + Permuted

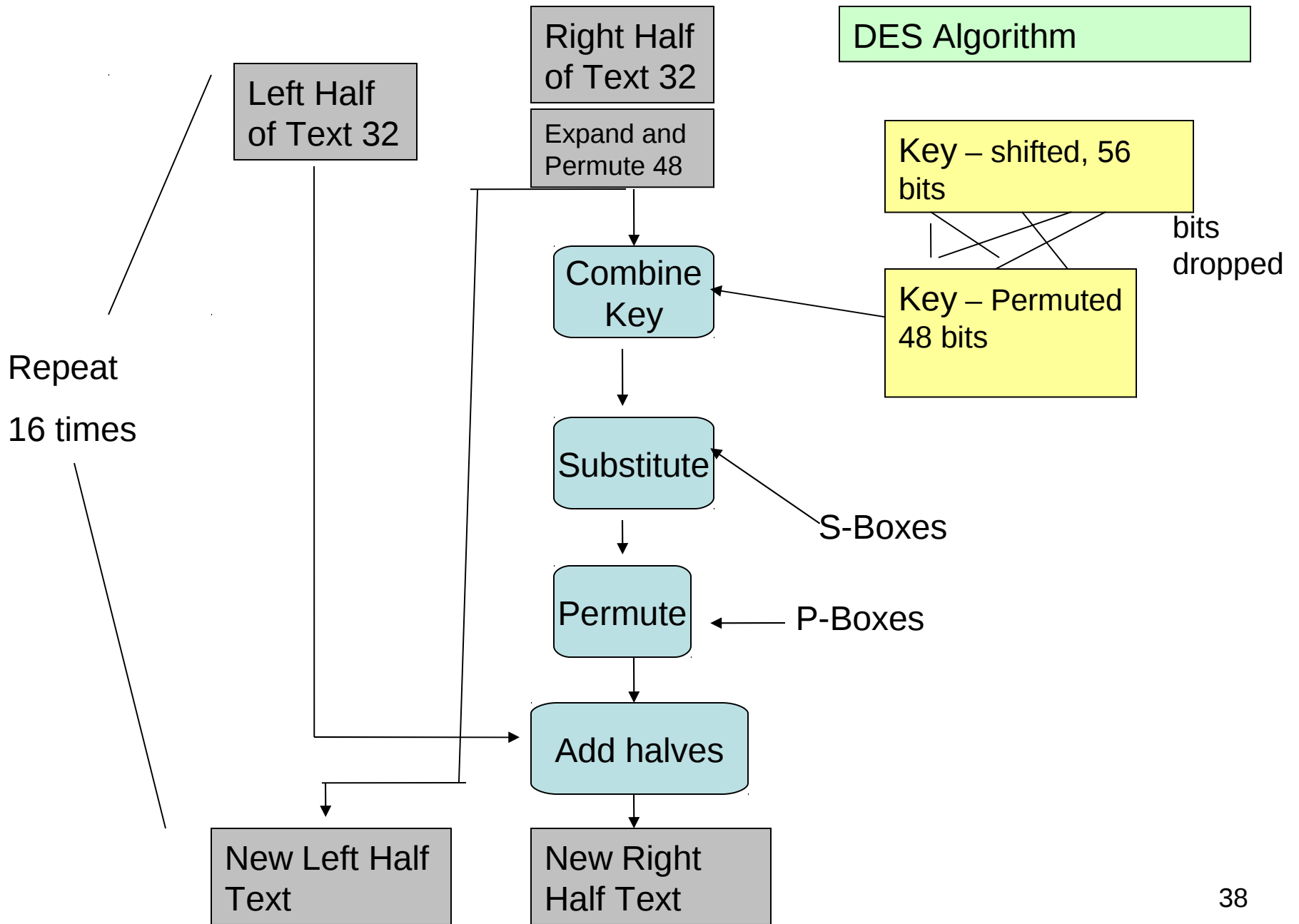
- **Cycles**

- Break permuted data into two halves, 32 bits each
- **Key gets transformed**
  - Key shifted left by some number and permuted
  - Key bits get dropped so only 48 bits used
- Right half of data expanded to 48 bits – duplicates certain bits
- Right half combined with 48 bits of key
- Result is substituted for another result and condensed to 32 bits
- 32 bits are permuted and then combined with left half to yield a new right half

# DES Algorithm

- Cycles

- Process iterated 16 times (rounds), each time with different set of tables and different bits from key
- Algorithm then performs final permutation, and 64 bits of output are provided



# DES Algorithm

- Decryption
  - Decryption uses the same algorithm and same secret key
  - Reversible process
  - Same function used but keys must be taken in reverse order ( $k_{16}, k_{15}, \dots, k_1$ )

# DES Algorithm

- Double DES
  - DES key is fixed at 56 bits
  - Not considered long by today's standards
  - Wanted to increase key length of DES but can't
    - DES algorithm is fixed at 56 bits
    - Researchers suggested doubling DES algorithm for greater security
    - Take two keys instead of 1 and perform two encryption's



# DES Algorithm



- Double DES

- Should in theory multiply difficulty making it harder to break

- Like two locks!

- However, two researchers, **Diffie and Hellman** showed that two encryptions are not better than one

- Strength of cipher is usually exponential in size of key

- So doubling key actually should square complexity  $2^{112}$ ,

- But applying DES encryption twice at best doubles complexity so only get  $2 \times 2^{56} = 2^{57}$

- However, Triple DES does work!

# Double DES

- Because message encrypted with DES can be forcibly decrypted by an attacker performing an exhaustive key search today, an attacker might also be able to forcibly decrypt a message encrypted with Double DES using a meet-in-the-middle attack at some point in the future

“ $2^{57}$  is still considerably more storage than one could comfortably comprehend, but it's enough to convince the most paranoid of cryptographers that double encryption is not worth anything,”

Bruce Schneier from *Applied Cryptography*

# DES Algorithm

- Triple DES
  - Using two keys, you apply them in 3 operations which adds strength
  - You encrypt with one key, decrypt with the second key and encrypt with the first key again
  - Three applications of the DES algorithm but it only doubles the effective key length – 112 bit key which is very strong against all feasible known attacks!

# DES Algorithm



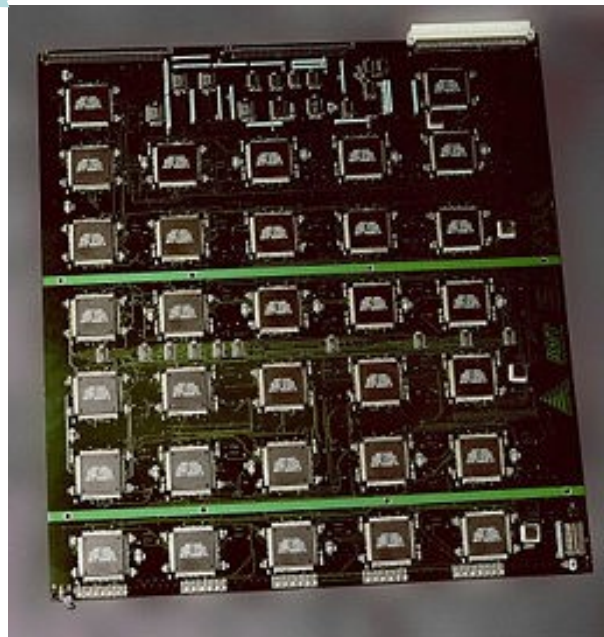
- **How Strong is DES?**
  - In 1997, researchers using over 3500 machines in parallel were able to infer DES key in 4 months
  - In 1998, researchers built DES cracker machine funded by Electronic Freedom Foundation and found DES key in 4 days
    - But it was clear that stronger algorithm was needed ....

# DES Cracked in a Few Days

- The EFF's US \$250,000 DES cracking machine contained 1,536 custom chips and could brute force a DES key in a matter of days — the photo shows a DES Cracker circuit board fitted with several Deep Crack chips

[http://en.wikipedia.org/wiki/EFF\\_DES\\_cracker](http://en.wikipedia.org/wiki/EFF_DES_cracker)

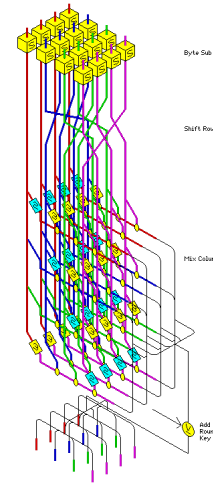
The entire machine was capable of testing over 90 billion keys per second



Deep Crack chips are 1856 custom ASIC DES chips

# AES Algorithm

- AES, the Beginning



- In 1997, NIST did another call for proposals
- Wanted an algorithm with these qualities:
  - Unclassified
  - Publicly disclosed
  - Available royalty-free for use worldwide
  - Symmetric block cipher algorithms
    - » For blocks of 128 bits
  - Usable with key lengths of 128, 192 and 256 bits

# AES Algorithm



Vincent Rijmen Joan Daemen

- **AES, the Beginning**

- In 1999, five finalists were selected, underwent intense public and private scrutiny
- Looked at security but also at cost or efficiency, ease of implementation in software
- Winning algorithm submitted by two Belgian cryptographers – Vincent Rijmen and Joan Daemen
- Became known as the Rijndahl algorithm
- Called AES and was adopted in 2001
  - Became Federal Information Processing Standard 197 (FIPS 197)

# AES Algorithm



- Overview of Rijndael
  - Fast algorithm
    - Can be implemented on simple processor
  - Uses substitution and transposition plus shift, XOR and addition operations
  - Uses repeat cycles – called rounds in Rijndael
  - Each cycle consists of 4 steps



# AES Algorithm

- Overview AES

- Each cycle has four steps

1. Byte substitution

- Uses byte substitution box structure similar to DES
  - Substituting each byte of a 128 bit block according to a substitution table

2. Shift Row

- Transposition step
  - For 128 and 192 bit block sizes, row  $n$  is shifted left circular  $(n-1)$  bytes
  - For 256 bit blocks, row 2 is shifted 1 byte and rows 3 and 4 are shifted 3 and 4 bytes respectively

# AES Algorithm

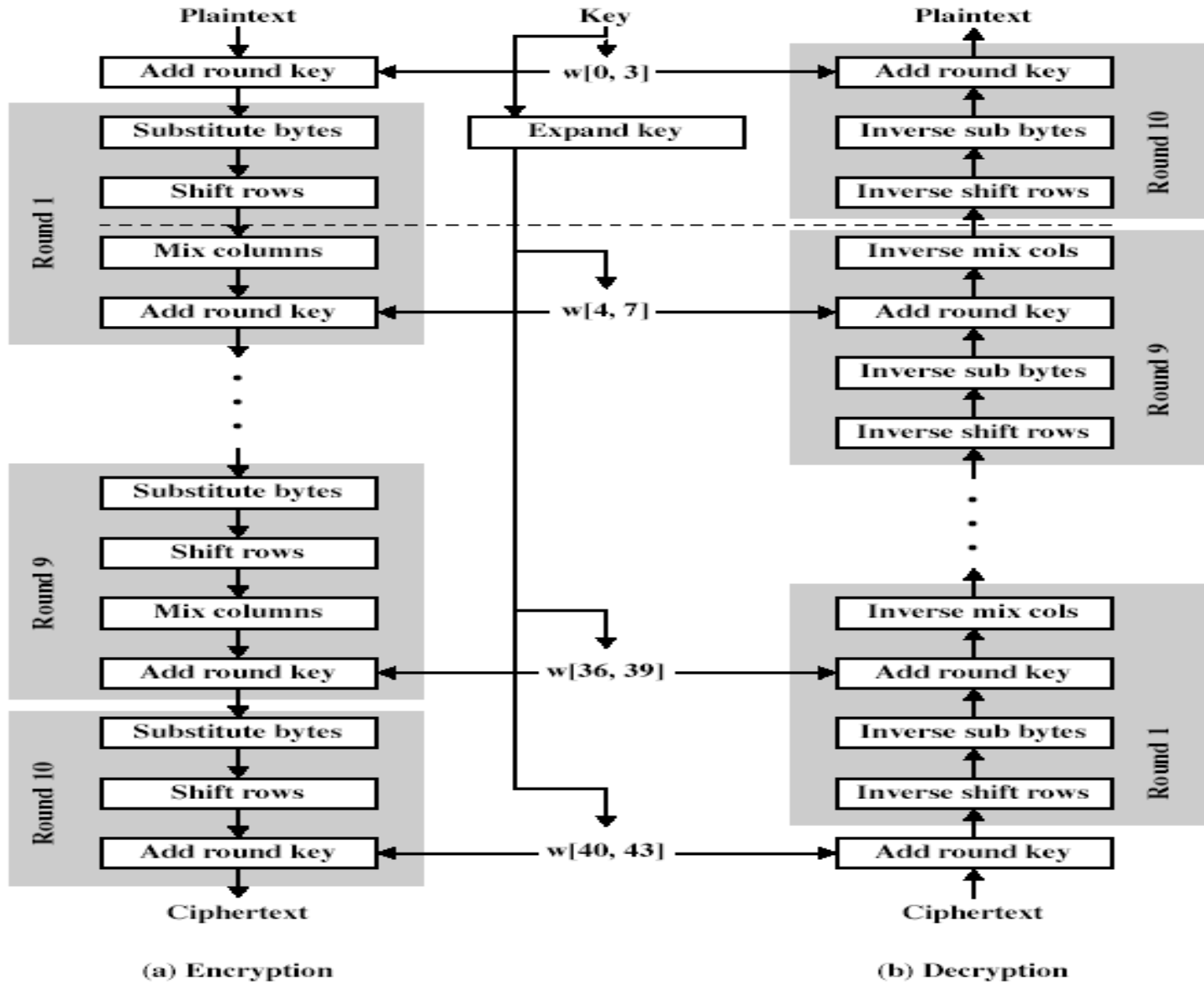
## 3. Mix Columns

- Shifting left and Xoring the bits with themselves

## 4. Add subkey

- Portion of key unique to this cycle is Xor'ed with cycle result
- Steps perform both confusion and diffusion on input data
- Bits from key are combined with intermediate results frequently so key bits will be well diffused

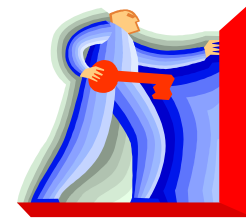
# Rijndael



# AES Algorithm

- Compare DES and AES
  - When evaluating DES, asked two questions ..
    - 1) How strong is DES, any backdoors?
    - 2) How long until encrypted code could be routinely cracked
  - In 20 years research has not found any major flaws in DES
  - Changes appear to weaken algorithm
  - DES does have a fixed key size

# AES Algorithm



- **Compare DES and AES**
  - Same questions for AES ...
    - AES algorithm defined with 128, 192 and 256 key lengths
    - Start with key size more than double that of DES
  - AES more flexible
    - Can extend the cycle number
    - Can change other aspects of algorithm without weakening it

# References

## RC4

<https://en.wikipedia.org/wiki/RC4>

<https://www.quora.com/Cryptography-What-is-an-intuitive-explanation-of-the-RC4-encryption-algorithm-and-its-weaknesses>

## DES General

[http://www.unix.org.ua/oreilly/networking/puis/ch06\\_04.htm](http://www.unix.org.ua/oreilly/networking/puis/ch06_04.htm)

## Double Strength Research DES

R. C. Merkle and M. Hellman, "On the Security of Multiple Encryption," *Communications of the ACM*, Volume 24, Number 7, July 1981, pp. 465-467

## DES - How it works with Diagrams

<http://accessscience.com/content/Cryptography/170600>

## DES Strength

Hellman, M. 1979. "DES will be totally Insecure in 10 years", *IEEE Spectrum*, V16, 7, Jul. 1979, pp. 32-39

## AES Animated – Very cool

[http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael\\_ingles2004.swf](http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf)

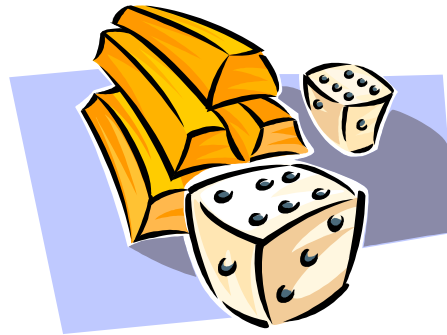
## AES

[http://en.wikipedia.org/wiki/Advanced\\_Encryption\\_Standard](http://en.wikipedia.org/wiki/Advanced_Encryption_Standard)

# The End

- Next Time

Use of Crypto products we know and love!!!



No Lab this week !!!











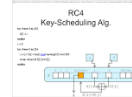








# RC4 – How Does it Work?



The RC4 cipher consists of two parts:

1. The Key Scheduling Algorithm (KSA), and
2. The Pseudo Random (Byte) Generation Algorithm (PRGA).

The KSA takes a neatly arranged array of 256 elements (has values 0, 1, 2, ..., 255 in order)

Uses a variable length secret key to turn the array into a pseudo-random order



## RC4 – How Does it Work?

- Once the KSA has finished, the array is supposed to "look" randomly arranged.
- After the KSA, the PRGA part starts
- This part outputs one byte at a time.
- Each PRGA step further perturbs the array a little while outputting one byte



































































































