Cryptography, Part 1

CSC 348·648

WAKE FOREST
UNIVERSITY
Department of Computer Science

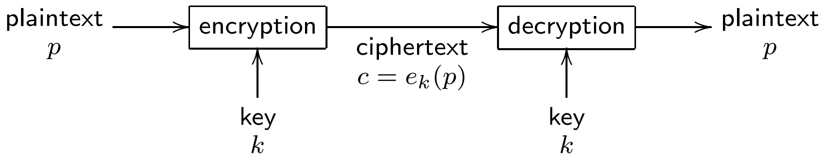
Spring 2013

Cryptography

- Cryptography comes from the Greek words
  - $\kappa\rho\upsilon\pi\tau o$ - hidden or secret
  - $\gamma\rho\alpha\phi\eta$ - writing
- *Art* of mangling information into an apparent unintelligible manner
  - Allowing a secret method of unmangling
- Cryptography (in its mathematical form) can provide other services
  - **Integrity checking** - Reassuring recipient of a message that is has not been altered
  - **Authentication** - Verifying identity
- For the larger part of history it has been an *art*
  - Still retains some *art*, but it has changed in the last 25 years
  - Has been supplemented with science

Ciphers

- Cipher - transforming **plaintext** into **ciphertext** using a **key**
  - Plaintext - the original message, p
  - Ciphertext - the encrypted message, $c = e_k(p)$
  - Key - input that selects one of many encryptions



plaintext p

encryption

ciphertext $c = e_k(p)$

decryption

plaintext p

key k

key k

- Encryption function $c = e_k(p)$
  - k is also a parameter of function e, but written as subscript
- Decryption function $p = d_k(e)$, therefore $d_k(e_k(p)) = p$

- Terminology
  - **Cryptography** - Devising ciphers
  - **Cryptanalysis** - Breaking ciphers
  - **Cryptology** - Devising and breaking ciphers

Goals and Settings

- Basic problem, ensuring communications over insecure medium
- **Ideal channel** - A dedicated, untappable, impenetrable pipe into which the sender can transmit a message and the receiver will hear
  - Unfortunately, this channel does not exist
  - Cryptography should provide the parties with a means to communicate with properties akin to those of a secure channel *What are the basic properties?*
- **Protocol** - A set of rules governing communication *What protocols are needed for secure transmission?*
- **Trust models** - Defines what initially is known *For example?*

Computational-Complexity

- Modern cryptography has added a new dimension
  - The amount of computing power required by the adversary
  - *Attacks are infeasible, not impossible*
- Cryptography has moved into realm of computer science
  - *Assuming the adversary uses no more than t computing cycles, her probability of breaking the scheme is at most* $\frac{t}{2^{200}}$
- Nothing will be stated about how the adversary operates *What? How does the effect the statement?*

Properties

- **Fundamental tenet of cryptography**
  - *If lots of smart people have failed to solve a problem, then it probably won't be solved soon*
  - Security of a method depends on the *work* required to break it
- As seen in the previous slide, e and d are mathematical functions
  - Parameters are p and k
  - Two important factors are $e_k(\cdot)$ and k
- Typically assume $e_k(\cdot)$ is known (publically available)
  - Prefer having the function public

    Why is keeping $e_k(\cdot)$ secret not a good idea?

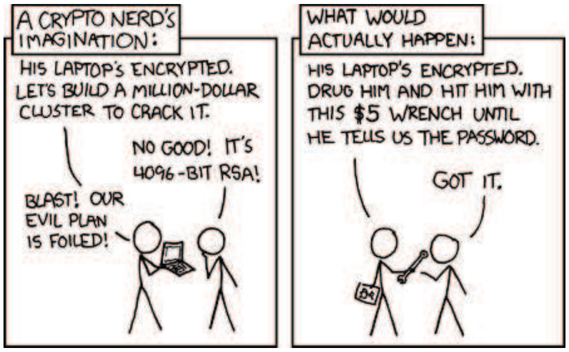
- *Strength* will depend on the **key**
  - Key *length* is a major design issue

    Given a combination lock, is the encryption method known?
    How does key length apply to its security?

    Consider the same questions regarding a key-lock.

- Remember breaking the cryptographic method is often just one way of getting what you want...

Key Length in Reality


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Types of Attacks

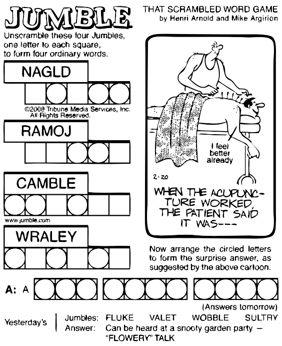
Depending on the information known, there are three types of attacks

- **Ciphertext only**
  - Attacker has the ciphertext only, can analyze at any time
  - Encryption method **must** be secure against this attack
- **Known plaintext**
  - Attacker has obtained some plaintext and corresponding ciphertext

    How could obtaining plaintext and corresponding ciphertext be possible?

- **Chosen plaintext**
  - Attacker can obtain the ciphertext of any plaintext needed

    Breaking a “cryptogram” is what type of attack?


JUMBLE
THAT SCRAMBLED WORD GAME
by Henri Arnold and Mike Argirion
Unscramble these four Jumbles, one letter to each square, to form four ordinary words.
NAGLD
©2008 Tribune Media Services, Inc. All Rights Reserved.
RAMOJ
CAMBLE
www.jumble.com
WRALEY
I feel better already
2-20
WHEN THE ACUPUNCTURE WORKED, THE PATIENT SAID IT WAS---
Now arrange the circled letters to form the surprise answer, as suggested by the above cartoon.
A: A
(Answers tomorrow)
Yesterday's
Jumbles: FLUKE VALET WOBBLE SULTRY
Answer: Can be heard at a snooty garden party – "FLOWERY" TALK

- Encryption methods should resist all types of attacks
  - Must resist ciphertext only
  - Chosen plaintext is the most difficult to resist

Types of Cryptographic Functions

Three types of functions are used, categories based on *keys*

- **Public (or Asymmetric) key**
  - Also called asymmetric cryptography, uses of two keys
  - One for encryption and another for decryption
- **Secret (or Symmetric) key**
  - Use one key for encryption and decryption
- **Hash**
  - No key at all...
  - *Zero key length has a purpose in security?*

  What are the trust models?

Substitution Ciphers

Each letter or group of letters is replaced by another letter or group of letters

- **Caesar Cipher** is a simple example
  - Replace each letter by the 3^{rd} letter on the right

plaintext	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
ciphertext	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

  - Notice ciphertext alphabet *wraps* after z
- **Captain Midnight Decoder Ring** (CMDR)
  - Given as a premium with Ovaltine in the 1940's
  - Allows the *shift* be any number from 0 to 25

  - If we map an integer i to each letter, where a$= 0$

$$e_k(i) = i + k \pmod{26}$$

for example, encode the letter w with $k = 5$

$$e_k(22) = 22 + 5 \pmod{26} = 27 \pmod{26} = 1$$

What is the decryption function?

  - For Caesar cipher $k = 3$

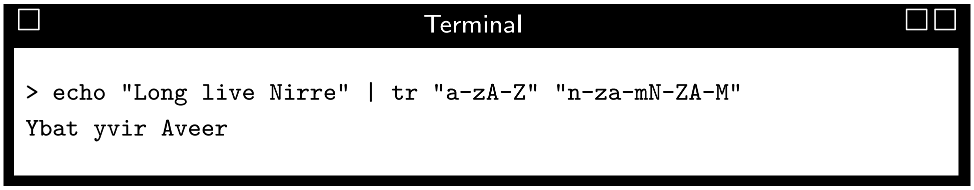
What is the key-space for CMDR?

- Cryptanalysis of CMDR
  - Key-space very small, use brute force (exhaustive key search)

ROT13

- ROT13 is a substitution cipher traditionally used in Usenet
  - It is a CMDR cipher, where $k = 13$
  - Done using the Unix `tr` command (translate)

`tr "a-zA-Z" "n-za-mN-ZA-M" <` *fileName*



```
> echo "Long live Nirre" | tr "a-zA-Z" "n-za-mN-ZA-M"
Ybat yvir Aveer
```

Monoalphabetic Substitution

- Instead of shifting by a fixed amount, *shuffle*
- Let each plaintext letter map to a unique random ciphertext letter

plaintext	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
ciphertext	p	l	v	x	h	t	j	k	f	r	n	o	d	q	m	s	i	u	f	w	g	y	z	a	b	c

What is the key length?

- Assume attacker has cipher text only
  - Assume it takes 1μsec to test a possible key
  - Possible keys $26! \approx 4 \times 10^{26}$ to test
  - It will take 10^{13} years to test all the keys
  - **Actually it will take a much shorter time...**

History of Monoalphabetic Cipher

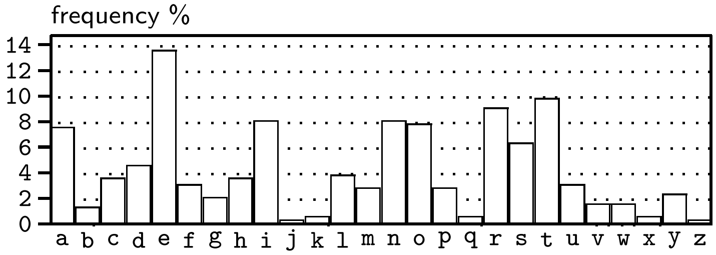
- Believed to be secure since $26!$ keys
  - Remembering the key is not easy... typically use a *keyword*
  - Given keyword, remove repeated letters and create table
  - For example assume `starwars` is the keyword

s	t	a	r	w
b	c	d	e	f
g	h	i	j	k
l	m	n	o	p
q	u	v	x	y
z				

  - Read columnwise for key $\{a \rightarrow s,\ b \rightarrow b,\ c \rightarrow g,\ ...\ \}$
- Believed secure for many centuries, into the middle ages
- Frequency analysis discovered by Abu al-Kindi in 9^{th} century

Language Redundancy

- Human languages are redundant
- Letters are not equally used
  - Most common letter is e
  - Other common letters include {t, r, n, i, o, a, s}



- Can use this information to break encryption

- Determine the frequency of letters in ciphertext
  - Look for *peaks* and *valleys* (*is this frequency domain analysis?*)
  - Highest frequency ciphertext-letter probably represents an `e`
  - Consider *digrams* {`th`, `in`, `er`, `re`, `an`}
  - Consider *trigrams* {`the`, `ing`, `and`, `ion`}
- For example, consider the following steps
  1. Determine relative frequencies of all letters in ciphertext
  2. Assign most common to `e` and the next common to `t`
  3. Look for trigram of the form `tXe`, which suggests `X` is `h`
  4. Look for pattern `thYt`, which suggests `Y` is `a`
  5. Continue with remaining ciphertext

  How does knowing the substitution method help?

Vigenére Cipher

- Vigenére developed the **polyalphabetic substitution cipher**
  - Use multiple monoalphabetic substitution alphabets
  - Use key to select which alphabet is used for each letter
  - i^{th} letter of keyword specifies the i^{th} alphabet to use

plaintext	t	h	i	s	p	r	o	c	e	s	s
keyword	c	i	p	h	e	r	c	i	p	e	r
ciphertext	v	p	x	z	t	i	q	k	t	z	w

plaintext		abcdefghijklmnopqrstuvwxyz
c	→	cdefghijklmnopqrstuvwxyzab
i	→	ijklmnopqrstuvwxyzabcdefgh
p	→	pqrstuvwxyzabcdefghijklmno
...		

Is this more secure than monoalphabetic substitution?

- Used in the *Zimmermann telegram* of WWI

Transposition Cipher

- Transposition ciphers reorder letters, but do **not** disguise them
  - Substitution disguises plaintext symbols (order maintained)
- A common transposition cipher is the **columnar transposition**
  - Based on a keyword, for example `megabuck`
  - Keyword is used to number columns (number of and order)
  - Consider plaintext {`please send million dollars`}

m	e	g	a	b	u	c	k
7	4	5	1	2	8	3	6
p	l	e	a	s	e	s	e
n	d	m	i	l	l	i	o
n	d	o	l	l	a	r	s

  - Read columnwise, number under keyword letter gives order

  - Ciphertext is {`ailsllsirlddemoeospnnela`}
- Cryptanalysis of transposition cipher
  - Determine if it is a transposition

    How can this be done?
  - Guess at the number of columns, having a probable long phrase in the message helps (e.g., `milliondollars`)
  - How the phrase *wraps* indicates the key length, (e.g., digrams {`mo`, `il`, `ll`, `la`, `ir`, `os`} resulted from key
  - Once determine the key length, guess the column order

    Assume k columns, how many orders exist?
  - Test English digram frequencies to determine if correct
- N.B. the procedure is more difficult with an incomplete matrix

The Unbreakable Cipher

- **Vernam** or **One-Time Pad**
  - Proven to be unbreakable by Shannon 1949
- Very simple algorithm
  1. Choose a long *random* bit string as the key (only use once)
  2. Convert the plaintext into binary
  3. Compute the exclusive OR ($\oplus$) of the two strings, bit by bit

$$p \oplus k = c \qquad c \oplus k = p$$

- For example

plaintext	11010100
key ⊕	10010110
ciphertext	01000010

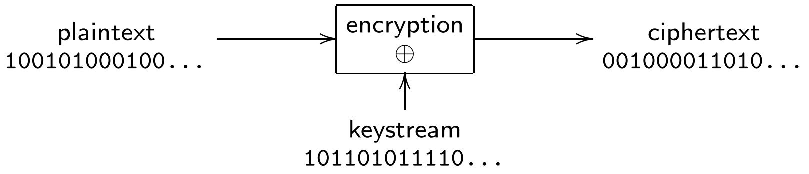
ciphertext	01000010
key ⊕	10010110
plaintext	11010100

- *Why is it unbreakable?*
  - Every possible plaintext is equally probable candidate
  - The preceding example could have been any ASCII letter
  - Ciphertext yields no information to the attacker
- Security depends on
  - Unpredictable key, probability of guessing next bit is $\frac{1}{2}$
- In most cases one-time pad is not practical
  - Random key must be as long as message (RNG's cycle)
  - Key management difficult
  - Synchronization is very important

Other than unbreakable, what is another advantage?

Stream Cipher

- One-time pad is also considered a **stream cipher**



plaintext
100101000100...

encryption
⊕

ciphertext
001000011010...

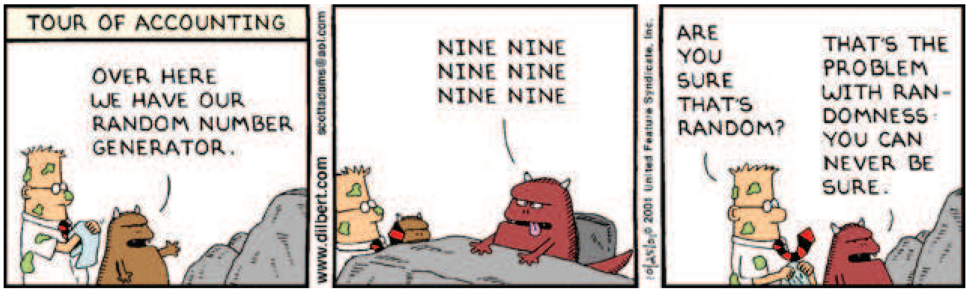
keystream
101101011110...

- To be secure the keystream must have the following properties
  - Long period (preferably longer than plaintext)
  - *RNG's are not really random...*
  - Pseudorandom properties (probability of next bit is x is $\frac{1}{2}$)

  Can Ceasar cipher be considered a stream cipher?

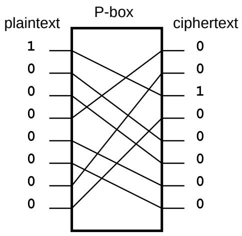
- Secure stream cipher (one-time pad) has many advantages
  - *But key management is difficult...*

The Problem with RNG


TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
scottadams@aol.com
www.dilbert.com
NINE NINE NINE NINE NINE NINE
10/15/01 © 2001 United Feature Syndicate, Inc.
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS. YOU CAN NEVER BE SURE.

Modern Cryptography

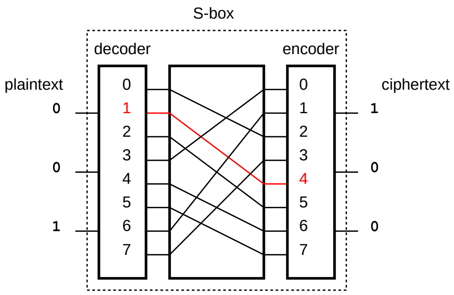
- Uses same basic ideas as *traditional cryptography*
  - Permutations (transpositions) and substitution
  - But focus is on shorter (manageable) key lengths
  - Will encrypt binary *blocks* of the message
- **Permutation** (P-box, provide *diffusion*)
  - A binary word (length k) has its bits reordered
  - The key describes how the permutation takes place (e.g., $1^{st} \rightarrow 3^{rd}$, $2^{st} \rightarrow 5^{rd}$, ...)



  - To allow all possible permutations requires $k \log_2 k$ bit key

    Why is that?

- **Substitution** (S-box, provide *confusion*)
  - A binary word (length k) replaced with 2^k possible outputs

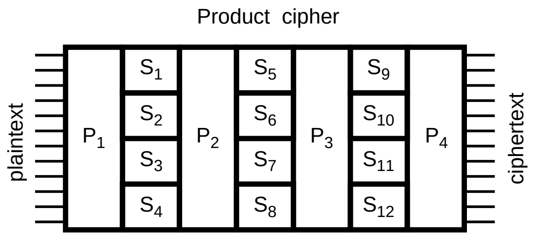


  - To allow all possible substitutions requires a $k \cdot 2^k$ length key

Advantages and disadvantages of S and P boxes?

Combining S and P

- Shannon developed the idea of S and P networks in 1949
  - Combine S and P boxes for encryption
  - Results in a **product cipher**

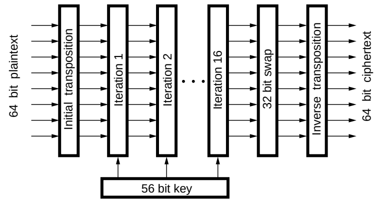


  - Each time bits pass through a box is called a *round*
  - The basis of modern block ciphers
- Can implement components in hardware for speed

  How would decryption occur?

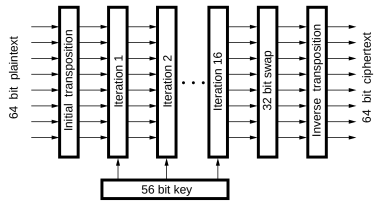
Data Encryption Standard

- Data Encryption Standard (DES) developed by IBM in 1970's
  - Based on the Feistel Cipher (same format)
- For each 64 bit block, algorithm yields 64 bit blocks of ciphertext
- Algorithm has a **56** bit key and 19 stages (rounds)



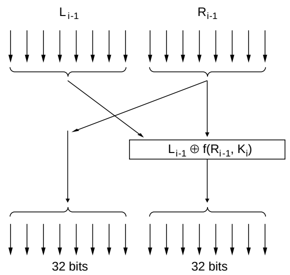
  - First stage is key independent transpose
  - Last state is the inverse of the transpose
  - Stage prior to last exchanges leftmost 32 bits with rightmost
  - Intermediate stages are equivalent, just use different *sub-key*

DES Overview



- 64-bit input subjected to initial permuation
- 56-bit key used to generate sixteen 48-bit per-round keys
  - Done by taking a different subsets of the 56-bit key
- Each round input is 64-bit output of previous round and 48-bit key
  - Produces a 64-bit output, repeats for 16 rounds
- Output halves from last round swapped and permutation done
  - Happens to be the inverse of the initial permutation

DES Stage (round) i Detail



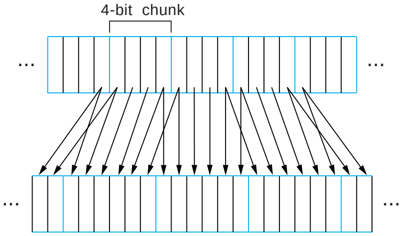
- Each block divided into 32 bit halves $\{L_i, R_i\}$
- Left output is copy of right input
- Right output is exclusive OR of right input and function result of left input and sub-key

$$L_i = R_{i-1}$$

$$R_i = l_{i-1} \oplus f(R_{i-1}, K_i)$$

- Complexity is within $f(R_{i-1}, K_i)$

1. 48 bit number E generated by expanding R_{i-1} (mangler)



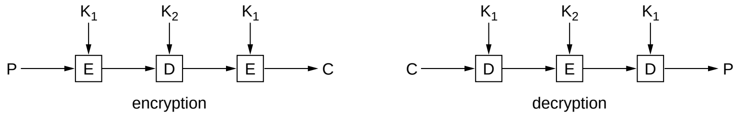
Expanded to 6 bits by stealing a bit from left and right chunks

2. Determine $E \oplus K_i$
3. Result partitioned in eight groups of 6 bits each
4. Each group sent to a different S-box (48 to 32), *key?*
5. Result is sent to P-box

- Each round uses a different sub-key K_i
  - Divide key into two 28 bit parts
  - Left shift both parts
  - Send to P-box (56 to 48)
- To decrypt use same process, just reverse the order of the sub-keys
- DES Strength (or lack of it)
  - Despite complexity, just a monoalphabetic substitution cipher
  - Key length is short

Triple DES

- *Accepted* method for a more secure DES is **multiple encryption**
  - Called EDE for (Encryption, Decryption, Encryption)



  - Using two keys the encryption is $c = e_{k_1}(d_{k_2}(e_{k_1}(p)))$
  - Using two keys the encryption is $p = d_{k_1}(e_{k_2}(d_{k_1}(c)))$
- *Why EDE not EEE?*
  - Encryption and decryption functions are mappings
  - Both map 64 bits to 64 bits and are reversible

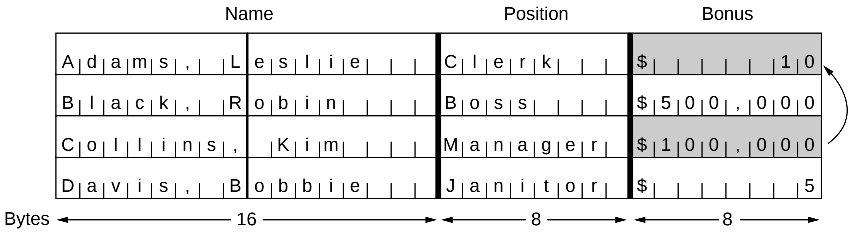
  - Each takes input *garbles* it and generates output
  - Can use just 2 keys, EEE requires 3

    Can you say EEE is more secure than EDE?

Encrypting Long Messages

What if you needed to encrypt a message longer than 64 bits?

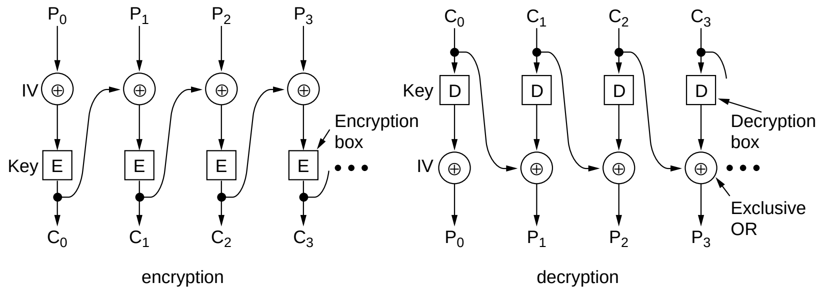
- **Electronic Code Book** (ECB)
  - Divide message into 64 bit blocks (pad if necessary)
  - Encrypt blocks independently (using same key per block)
- A possible ECB attack
  - Consider a file that lists the bonuses of employees
  - File has names, jobs, and bonus amounts in a specific format



Name		Position	Bonus
Adams, L	eslie	Clerk	$10
Black, R	obin	Boss	$500,000
Collins,	Kim	Manager	$100,000
Davis, B	obbie	Janitor	$5

-
  - Assuming Leslie can obtain the encrypted text
  - He could substitute DES block 11 for 3 (*could have substituted block 7, but no need to be greedy*)
- To defeat this attack, DES must be *chained* so that changing one block affects the remaining message

Cipher Block Chaining



- Cipher Block Chaining (CBC)
  - Input to the encryption is the exclusive OR of the current plaintext block and preceding ciphertext block

$$c_i = e_k(c_{i-1} \oplus p_i)$$

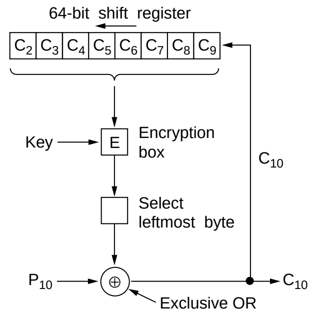
  - Initial block (IV) is needed to start process
  - Encryption of block i is a function of blocks $0...i-1$

- IV block must be know by receiver
  - Must keep secret (similar to a key)
- Decryption is similar to encryption

$$\begin{aligned} d_k(c_i) &= d_k(e_k(c_{i-1} \oplus p_i)) \\ d_k(c_i) &= (c_{i-1} \oplus p_i) \\ c_{i-1} \oplus d_k(c_i) &= c_{i-1} \oplus c_{i-1} \oplus p_i = p_i \end{aligned}$$

- CBC requires a full 64 bits before encryption can start

Cipher Feedback Mode



cipher feedback mode

- Cipher Feedback Mode (CFM) creates a byte-by-byte DES stream
  - Uses a queue (*shift register*) of ciphertext bytes $\{c_i, ..., c_{i+8}\}$
  - In the diagram, bytes 0 through 9 have been encrypted
  - When p_{10} arrives algorithm operates on the 64 bit shift register to generate a 64 bit ciphertext

  - Left byte of the cipertext is removed and exclusive OR with p_{10}
  - c_{10} is transmitted on the line
  - c_2 is shifted out and c_{10} is inserted in queue
- Shift register (queue) will depend on *entire* plaintext history
  - A plaintext pattern that repeats will be encrypted differently
    Was this not the case with ECB?

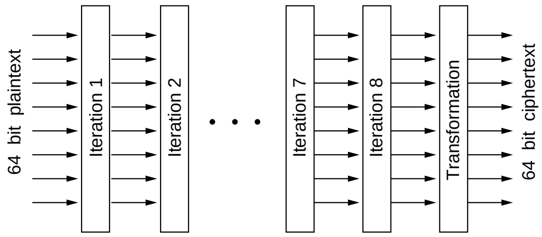
Breaking DES

- DES was controversial from its release in the 1970's
  - When the US government (NSA) wanted a standardized cipher it *invited* IBM to *discuss* the matter
  - IBM had developed Lucifer, same as DES but 128 bit key
  - After discussions the key was reduced to 56 bits
  - Many believed this was a NSA *trapdoor*
- In 1977 research on breaking DES started (*exhaustive search*)
  - Provided computer designs and proved theoretical bounds

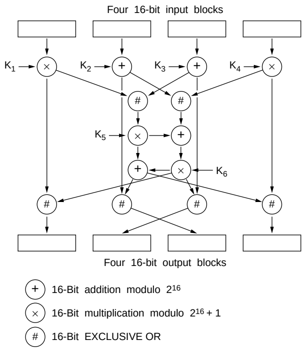
- In 1994 the 56 bit key could be broken in 4 hours
  - The idea of parallel searches used
  - Key space divided across multiple computers (home PCs)
- Final notes and observations
  - There is no published mathematical proof that DES is secure
  - Security is achieved through confusion and diffusion
  - Most applications will use triple DES

IDEA

- **International Data Encryption Algorithm** (IDEA)
  - Developed by researchers in Switzerland
  - Uses a 128 bit key, which makes it immune to brute force attacks (*or at least in the near future*)
  - None of the current attacks have been successful
- Basic structure is similar to DES
  - 64 bit plaintext is encrypted to produce 64 bit ciphertext
  - Encryption occurs over multiple iterations (rounds)



- One iteration consists of three operations on 16 bit numbers
  - Exclusive OR, addition modulo 2^{16}, and multiplication modulo $2^{16}+1$ (no S-boxes)


Four 16-bit input blocks
K_1
K_2
K_3
K_4
K_5
K_6
Four 16-bit output blocks
+ 16-Bit addition modulo 2^{16}
× 16-Bit multiplication modulo $2^{16}+1$
# 16-Bit EXCLUSIVE OR

  - Operations have the property that no two pairs obey the associative or distributive law
- *How well does it work?*
  - No one has completely broken it... yet

Advance Encryption Standard

- US Dept of Commerce needed to determine successor of DES
  - January 1997 sponsored a cryptographic *bake-off*
  - Invited proposals for the Advance Encryption Standard (AES)
- Requirement of the proposed encryption methods
  1. Algorithm must be symmetric
  2. Full design must be public
  3. Key lengths of 128, 192, and 256 must be supported
  4. Possible software and hardware implementations
  5. Algorithm public or licensed on nondiscriminatory terms

AES Finalists

- In 1998, the finalists and the votes were as follows
  1. Rijndael (Joan Daemen and Vicent Rijmen) 86 votes
  2. Serpent (Ross Anderson, et al.) 59 votes
  3. Twofish (Bruce Schneier, et al.) 31 votes
  4. RC6 (RSA Laboratories) 23 votes

     What is MC5?

  5. MARS (IBM) 13 votes
- In November 2001, NIST announced Rijdael as the US Government Federal Information Processing Standard FIPS 197
  - As the winner of the contest, 128 bit block size variant of Rijndael became the AES encryption standard

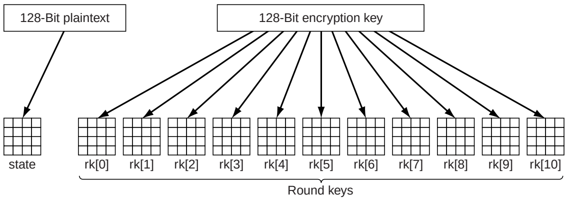
  What about IDEA?

Rijndael

- Rijndael is based on Galois field theory
  - Has some provable security properties
  - Uses substitutions and permutations in multiple rounds
- The `rijndael` function has three parameters
  - `plaintext` array of 16 bytes of plaintext
  - `ciphertext` array of 16 bytes of ciphertext
  - `key` array of 16 byte key
- Current *state* of the data is stored in `state` array
  - Initialized to the plaintext and modified by every step

Rijndael (AES) Algorithm

1. Expand (XOR and rotation) `key` into 11 arrays of the same size
   - One copy to start calculations, remaining for the rounds
2. Copy `plaintext` into `state` array (column-order)



3. `rk[0]` is XORed into `state` array
4. Loops executes for 10 iterations, transforming `state` per round where each iteration has 4 steps
   (a) Monoalphabetic substitution on `state`, S-box
   (b) Rotate the four rows of `state` to the left (diffuse)

(c) Mix columns independently of each other via matrix multiply

(d) XOR `key` into the `state` array

```c
#define LENGTH 16                   // num bytes in data block or key
#define NROWS 4                     // number of rows in state
#define NCOLS 4                     // number of columns in state
#define ROUNDS 10                   // number of iterations
typedef unsigned char byte;         // unsigned 8-bit integer

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH],
         byte key[LENGTH])
{
  byte state[NROWS][NCOLS];                   // current state
  struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];  // round keys
  expand_key(key, rk);                        // construct round keys
  copy_plaintext_to_state(state, plaintext);  // init current state
  xor_roundkey_into_state(state, rk[0]);      // XOR key into state
  for (int r = 1; r <= ROUNDS; r++)
  {
        substitute(state);                        // apply S-box to byte
        rotate_rows(state);                       // rotate row i by i bytes
        if (r < ROUNDS) mix_columns(state);       // mix function
        xor_roundkey_into_state(state, rk[r]);    // XOR key into state
  }
  copy_state_to_ciphertext(ciphertext, state);    // return result
}
```

Yes Chukar, I expect you to memorize this

Breaking Ciphers

- Breaking a cipher (cryptanalysis) refers to recovering the key
  - Most cases assume the general algorithm is known
  - Of course, brute force is one approach...
- A "break" is any attack which requires less than 2^n operations
- Given ciphertext, chosen ciphertext, ... ?
  - Where n is the size of the key, so faster than brute force
- There are several approaches, two worth mentioning
  - Linear cryptanalysis is based on finding line approximations to the action of a cipher
  - Differential cryptanalysis is the study of how differences in an input can affect the resultant difference at the output (method uses pairs of plaintext related by a constant difference)

Key Recovery Attacks

Below are publicly known attacks (attacker discovers the key) against popular encryption methods. Rows in red are demonstrated attacks, while yellow rows are theoretical breaks. The "Best Attack" column describes the complexity of the attack

- Time is number of cipher evaluations required
- Data is the amount on known plaintext-ciphertext required
- Memory is the number of blocks of data to be stored

Cipher	**Security Claim**	**Best Attack**	**Attack Date**	**Notes**
AES128	2^{128}	$2^{126.1}$ time, 2^{88} data, 2^{8} memory	8/17/2011	Independent biclique attacks
AES192	2^{192}	$2^{189.7}$ time, 2^{80} data, 2^{8} memory	8/17/2011	Independent biclique attacks
AES256	2^{256}	$2^{254.4}$ time, 2^{40} data, 2^{8} memory	8/17/2011	Independent biclique attacks
Blowfish	2^{448}	4 of 16 rounds	1997	
DES	2^{56}	2^{56} time	7/17/1998	Broken by brute force, see EFF DES cracker. Off-the-shelf hardware is available for $10,000
Triple DES	2^{168}	2^{113} time, 2^{32} data, 2^{88} memory	3/23/1998	
IDEA	2^{128}	6 of 8.5 rounds ($2^{126.8}$ time, 2^{64} data)	3/16/2007	Differential-linear attack
KASUMI	2^{128}	2^{32} time, 2^{26} data, 2^{30} memory, 4 related keys	2010-01-10	The cipher used in 3G cell phone networks. This attack takes less than two hours on a single PC, but isn't applicable to 3G due to known plaintext and related key requirements.
Serpent-128	2^{128}	10 of 32 rounds (2^{89} time, 2^{118} data)	2/4/2002	Linear cryptanalysis
Serpent-192	2^{192}	11 of 32 rounds (2^{187} time, 2^{118} data)	2/4/2002	Linear cryptanalysis
Serpent-256	2^{256}	11 of 32 rounds (2^{187} time, 2^{118} data)	2/4/2002	Linear cryptanalysis
Twofish	2^{128}...2^{256}	6 of 16 rounds (2^{256} time)	10/5/1999	