Lecture for January 29, 2016

ECS 235A

UC Davis

Matt Bishop

Cæsar's Problem

- Key is too short
  - Can be found by exhaustive search
  - Statistical frequencies not concealed well
    - They look too much like regular English letters
- So make it longer
  - Multiple letters in key
  - Idea is to smooth the statistical frequencies to make cryptanalysis harder

Vigènere Cipher

- Like Cæsar cipher, but use a phrase
- Example
  - Message `THE BOY HAS THE BALL`
  - Key `VIG`
  - Encipher using Cæsar cipher for each letter:

```
key    VIGVIGVIGVIGVIGV
plain  THEBOYHASTHEBALL
cipher OPKWWECIYOPKWIRG
```

Relevant Parts of Tableau

	G	*I*	*V*
A	G	I	V
B	H	J	W
E	L	M	Z
H	N	P	C
L	R	T	G
O	U	W	J
S	Y	A	N
T	Z	B	O
Y	E	H	T

- Tableau shown has relevant rows, columns only
- Example encipherments:
  - key V, letter T: follow V column down to T row (giving "O")
  - Key I, letter H: follow I column down to H row (giving "P")

Useful Terms

- *period*: length of key
  - In earlier example, period is 3
- *tableau*: table used to encipher and decipher
  - Vigènere cipher has key letters on top, plaintext letters on the left
- *polyalphabetic*: the key has several different letters
  - Cæsar cipher is monoalphabetic

Attacking the Cipher

- Approach
  - Establish period; call it *n*
  - Break message into *n* parts, each part being enciphered using the same key letter
  - Now you have *n* Caesar ciphers, so solve each part
    - You can leverage one part from another

One-Time Pad

- A Vigenère cipher with a random key at least as long as the message
  - Provably unbreakable
  - Why? Look at ciphertext `DXQR`. Equally likely to correspond to plaintext `DOIT` (key `AJIY`) and to plaintext `DONT` (key `AJDY`) and any other 4 letters
  - Warning: keys *must* be random, or you can attack the cipher by trying to regenerate the key
    - Approximations, such as using pseudorandom number generators to generate keys, are *not* random

Overview of the DES

- A block cipher:
  - encrypts blocks of 64 bits using a 64 bit key
  - outputs 64 bits of ciphertext
- A product cipher
  - basic unit is the bit
  - performs both substitution and transposition (permutation) on the bits
- Cipher consists of 16 rounds (iterations) each with a round key generated from the user-supplied key

How It Works

- Round keys: for each round key
  - Permute bits in key
  - Extract 48 bits for round key
- Heart of each round is *f* function:
  - Expand *R* (E table), xor with round key
  - Substitute every 6 bits with 4 bits (S boxes)
  - Permute remaining 32 bits (P table)

How It Works

- Full DES
  - Permute 64 bit input (IP table)
  - Split into left *L*, right *R* (32 bits each)
  - Do this 16 times:
    - Run *f* function on *R*
    - Xor result with *L*; this is new *L*
    - If not last round, swap new *L* and *R*
  - Permute 64 bits (IP^{-1} table); result is output

Controversy

- Considered too weak
  - Diffie, Hellman said in a few years technology would allow DES to be broken in days
    - Design using 1999 technology published
  - Design decisions not public
    - S-boxes may have backdoors

Undesirable Properties

- 4 weak keys
  - They are their own inverses
- 12 semi-weak keys
  - Each has another semi-weak key as inverse
- Complementation property
  - $\text{DES}_k(m) = c \Rightarrow \text{DES}_{k'}(m') = c'$
- S-boxes exhibit irregular properties
  - Distribution of odd, even numbers non-random
  - Outputs of fourth box depends on input to third box

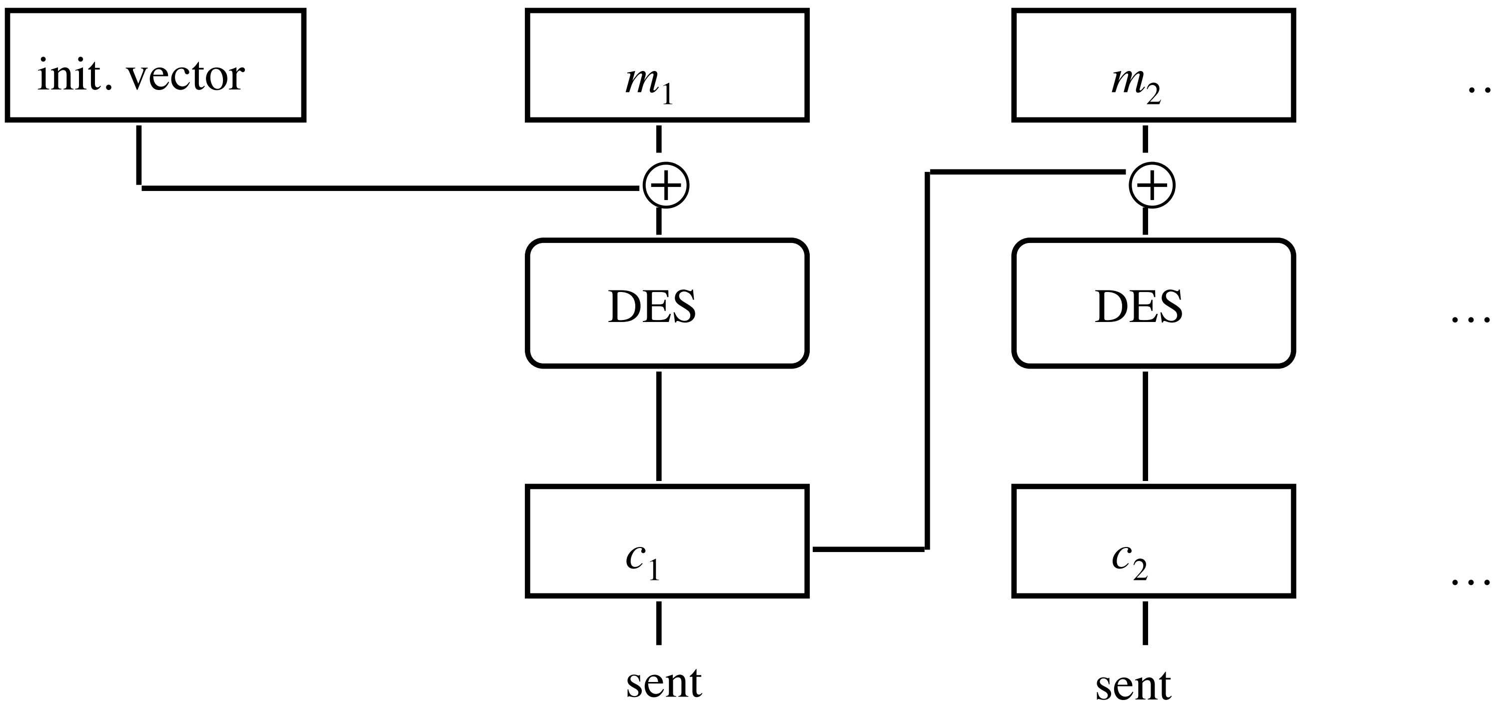
Differential Cryptanalysis

- A chosen ciphertext attack
  - Requires 2^{47} plaintext, ciphertext pairs
- Revealed several properties
  - Small changes in S-boxes reduce the number of pairs needed
  - Making every bit of the round keys independent does not impede attack
- Linear cryptanalysis improves result
  - Requires 2^{43} plaintext, ciphertext pairs

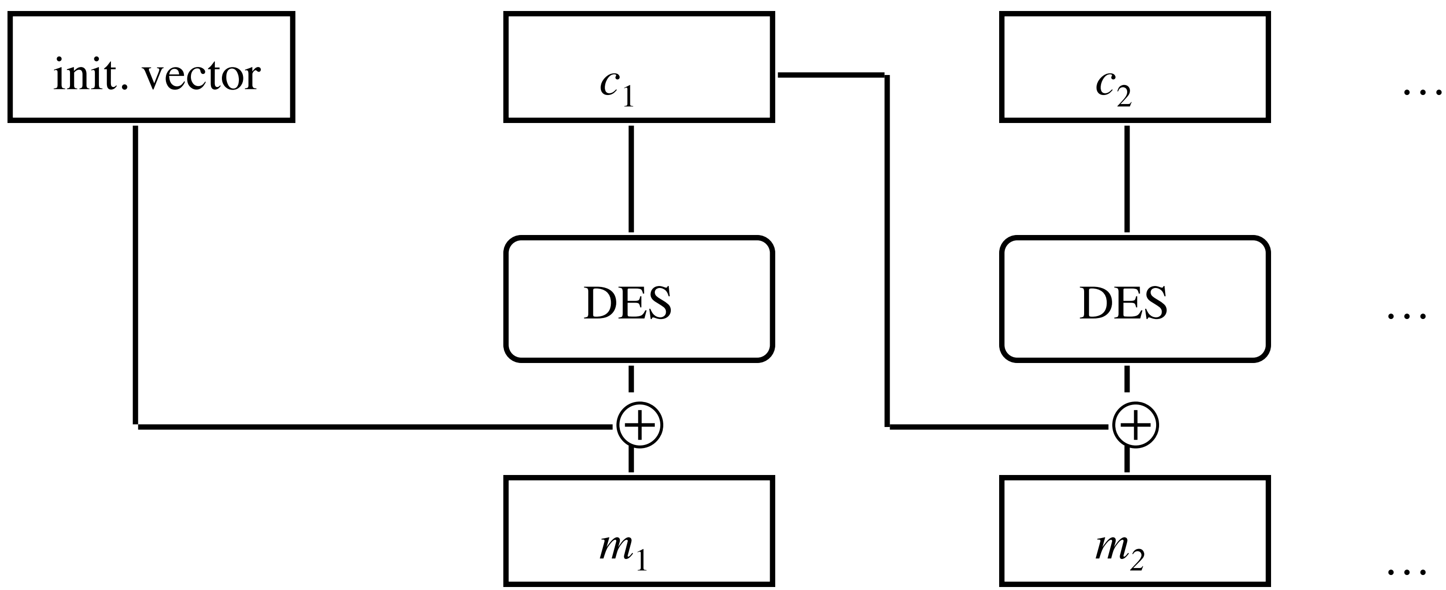
DES Modes

- Electronic Code Book Mode (ECB)
  - Encipher each block independently
- Cipher Block Chaining Mode (CBC)
  - Xor each block with previous ciphertext block
  - Requires an initialization vector for the first one
- Encrypt-Decrypt-Encrypt Mode (2 keys: k, k')
  - $c = \mathrm{DES}_k(\mathrm{DES}_{k'}^{-1}(\mathrm{DES}_k(m)))$
- Encrypt-Encrypt-Encrypt Mode (3 keys: k, k', k'')
  - $c = \mathrm{DES}_k(\mathrm{DES}_{k'}(\mathrm{DES}_{k''}(m)))$

CBC Mode Encryption



CBC Mode Decryption



Self-Healing Property

- Initial message
  - `3231343336353837 3231343336353837`
    `3231343336353837 3231343336353837`
- Received as (underlined 4c should be 4b)
  - `ef7c4cb2b4ce6f3b f6266e3a97af0e2c`
    `746ab9a6308f4256 33e60b451b09603d`
- Which decrypts to
  - `efca61e19f4836f1 3231333336353837`
    `3231343336353837 3231343336353837`
  - Incorrect bytes underlined
  - Plaintext "heals" after 2 blocks

Current Status of DES

- Design for computer system, associated software that could break any DES-enciphered message in a few days published in 1998
- Several challenges to break DES messages solved using distributed computing
- NIST selected Rijndael as Advanced Encryption Standard, successor to DES
  - Designed to withstand attacks that were successful on DES

Overview of the AES

- A block cipher
  - encrypts plaintext blocks of 128 bits
  - outputs 128 bits of ciphertext
- A product cipher
- 3 key lengths: 128, 192, 256 bits
- Cipher consists of rounds each with a round key generated from the user-supplied key
  - Number of rounds depends on length of key
  - Numbers are 10, 12, 14 respectively

Basic Transformations

- View input as a 4×4 array (the “state array”)
  - Input loaded down, going to next column when each column is finished
  - Input and output of each round is in this
- *RotWord*: rotate word by 1 byte
- *SubWord*: apply an S-box to the byte
- *ShiftRows*: cyclically shift rows
- *MixColumns*: alter columns independently

How It Works

- Round keys: 1 per round
  - Divide key into 4-byte words
    - Key is 4, 6, 8 words depending on on length of key
  - *RotWord*, *SubWord*, xor with bit string
  - Xor result with corresponding word of previous round (or initial key)

How It Works

- Encryption
  - *AddRoundKey* combines round key, state array
- Now the rounds
  - *SubBytes* substitutes new byte values
  - *ShiftRows* cyclically shifts rows
  - *MixColumns* alters each column independently
  - *AddRoundKey* xors state array with round key
- Last round omits *MixColumns*

How It Works

- Decryption: similar to encryption but:
  - Round key schedule reversed
  - *InvShiftRows* replaces *ShiftRows*
    - Inverts *ShiftRows* shifting
  - *InvSubBytes* replaces *SubBytes*
    - Inverts *SubBytes* substitution
  - *InvMixColumns* replaces *MixColumns*
    - Inverts *MixColumns* transformation

Properties

- S-box design critical
  - Non-linear, output not linear function of input
  - Algebraic complexity: inverse of each byte remapped with affine transformation
  - Result: no input ever mapped to itself or its bitwise complement
- Round keys non-linear with respect to original keys
- No weak or semiweak keys

Properties

- Diffuses input bits rapidly
  - After every 2 successive rounds, every bit in state array depends on every bit in state array 2 rounds earlier
- Designed to withstand the attacks that DES showed weakness to
  - Not vulnerable to differential, linear cryptanalysis

Modes

- All DES modes work with AES
  - With obvious modifications about block size, *etc*.
- EDE, "Triple AES" modes not used
  - Extended block, key size makes them unnecessary

Public Key Cryptography

- Two keys
  - *Private key* known only to individual
  - *Public key* available to anyone
    - Public key, private key inverses
- Idea
  - Confidentiality: encipher using public key, decipher using private key
  - Integrity/authentication: encipher using private key, decipher using public one

Requirements

1. It must be computationally easy to encipher or decipher a message given the appropriate key
2. It must be computationally infeasible to derive the private key from the public key
3. It must be computationally infeasible to determine the private key from a chosen plaintext attack

Diffie-Hellman

- Compute a common, shared key
  - Called a *symmetric key exchange protocol*
- Based on discrete logarithm problem
  - Given integers n and g and prime number p, compute k such that $n = g^k \bmod p$
  - Solutions known for small p
  - Solutions computationally infeasible as p grows large

Algorithm

- Constants: prime p, integer $g \neq 0, 1, p-1$
  - Known to all participants
- Anne chooses private key k_{Anne}, computes public key $K_{\text{Anne}} = g^{k\text{Anne}} \bmod p$
- To communicate with Bob, Anne computes $K_{\text{shared}} = K_{\text{Bob}}{}^{k\text{Anne}} \bmod p$
- To communicate with Anne, Bob computes $K_{\text{shared}} = K_{\text{Anne}}{}^{k\text{Bob}} \bmod p$
  - It can be shown these keys are equal

Example

- Assume $p = 53$ and $g = 17$
- Alice chooses $k_{\text{Alice}} = 5$
  - Then $K_{\text{Alice}} = 17^5 \bmod 53 = 40$
- Bob chooses $k_{\text{Bob}} = 7$
  - Then $K_{\text{Bob}} = 17^7 \bmod 53 = 6$
- Shared key:
  - $K_{\text{Bob}}{}^{k\text{Alice}} \bmod p = 6^5 \bmod 53 = 38$
  - $K_{\text{Alice}}{}^{k\text{Bob}} \bmod p = 40^7 \bmod 53 = 38$

RSA

- Exponentiation cipher
- Relies on the difficulty of determining the number of numbers relatively prime to a large integer *n*

Background

- Totient function $\phi(n)$
  - Number of positive integers less than *n* and relatively prime to *n*
    - *Relatively prime* means with no factors in common with *n*
- Example: $\phi(10) = 4$
  - 1, 3, 7, 9 are relatively prime to 10
- Example: $\phi(21) = 12$
  - 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are relatively prime to 21