Foundations of Network and Computer Security

John Black

Lecture #3
Aug 31st 2004

CSCI 6268/TLEN 5831, Fall 2004

Assignment #0

- Please add yourself to the class mailing list
  - Send mail to listproc@lists.colorado.edu
  - Subject is ignored
  - In body of message write

    “subscribe CSCI-6268 *Your Name*”
- Due by September 7th (Tuesday)

Review

- Summing up last lecture on blockciphers:
  - Small keysize bad (exhaustive search)
  - Small blocksize bad (frequency analysis)
  - Our first try at a 64-bit blockcipher (ie, blocksize is 64 bits) with a 64-bit key was no good
    - $C = P \oplus K$
    - Can distinguish from random with only two queries to one of the black boxes

Let's build a Better Blockcipher

- DES – The Data Encryption Standard
  - Formerly called "Lucifer"
    - Developed by Horst Feistel at IBM in early 70's
  - Tweaked by the NSA
    - No explanation given for tweaks
    - Some people worried that NSA was adding backdoors/weaknesses to allow it to be cracked!
    - NSA shortened key from 64 bits to 56 bits (definite added weakness)
  - Adopted by NIST (then called NBS) as a Federal Information Processing Standard (FIPS 46-3)
    - NIST is retiring it as a standard this year after nearly 30 years

The DES Key

- Was 64 bits

k0	k1	k2	k3	k4	k5	k6	k7	k8	k9	•	•	•	•	•	k60	k61	k62	k63

- But NSA added 8 parity bits

k0	k1	k2	k3	k4	k5	k6	**P0**	k8	k9	•	•	•	•	•	k60	k61	k62	**P7**

- Key is effectively only 56 bits!

Exhaustive Key Search -- DES

- This meant that instead of 2^{64} keys there were only 2^{56} keys
  - Expected number of keys to search before finding correct value is 2^{55}
    - Note that we need a handful of plaintext-ciphertext pairs to test candidate keys
  - NSA surely could do this in a reasonable amount of time, even in the 70's

Exhaustive Key Search -- DES

- In 1994, Michael Wiener showed that you could build a DES-cracking machine for $1,000,000 that would find the key in an expected 3.5 hours
  - In 1998 he revised this to 35 minutes for the same cost
  - In 1997, Rocke Verser used 10,000+ PCs to solve DES Challenge I to win $10,000 (Loveland, CO!)
  - distributed.net solved the DES Challenge II in 41 days with 50,000 processors covering 85% of the keyspace
  - Later the same year the EFF built the DES Cracker machine which found the same key in 56 hours
    - $210,000 for the machine
    - 92 billion key trials per second

No Better Attack has Ever Been Found against DES

- This is saying something:
  - Despite lots of cryptanalysis, exhaustive key search is still the best known attack!

- Let's have a look at (roughly) how DES works and see in what ways it's still in use

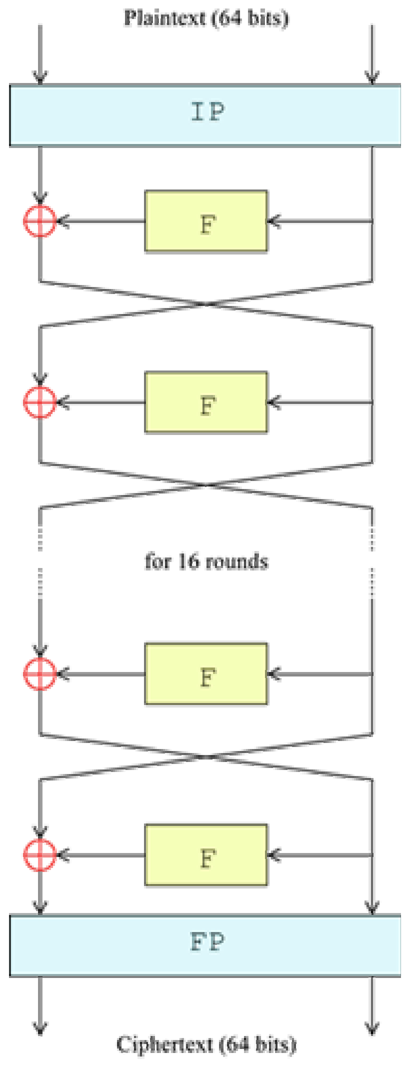
DES -- Feistel Construction

- IP – Initial permutation swaps bits around for hardware purposes
  - Adds no cryptographic strength; same for FP

- Each inner application of F and the XOR is called a “round”

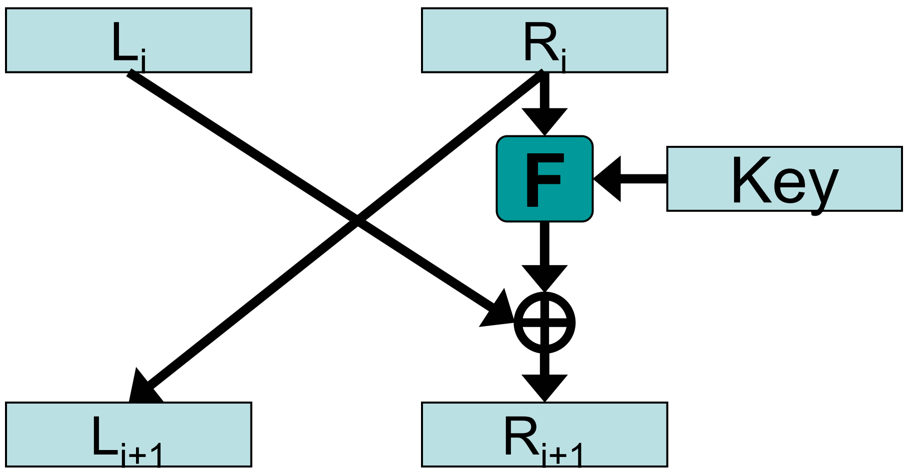
- F is called the “round function”
  - The cryptographic strength of DES lies in F

- DES uses 16 rounds



One Round

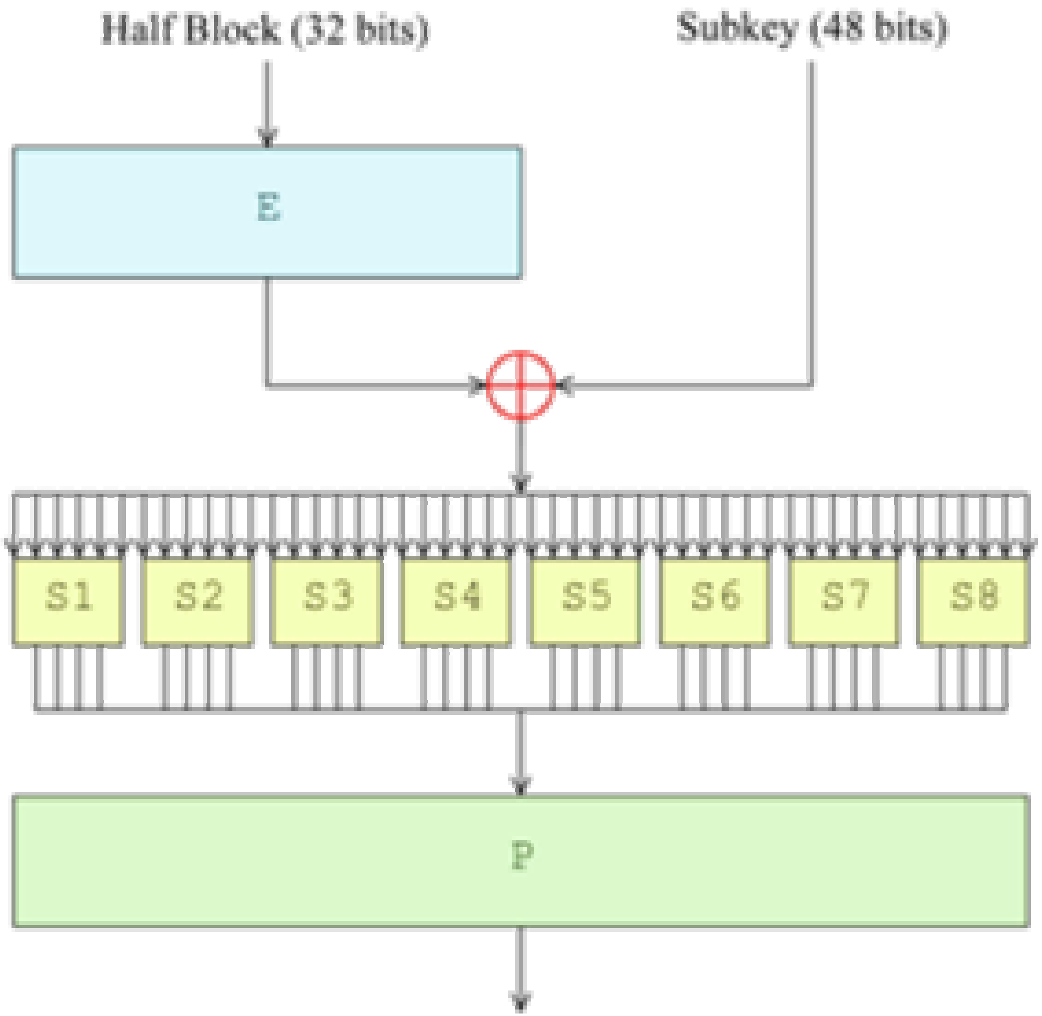
- Each half is 32 bits
- Round key is 48 bits
- Is this a permutation (as required)?
    - How do we invert?
- Note that F need not be invertible with the round key fixed



Why so many Rounds?

- Can we just have one round of Feistel?
  - Clearly this is insecure
- How about two rounds?
  - Expect to be asked a related question on the first quiz
- DES has 16 rounds
  - It's easily broken with 8 rounds using differential cryptanalysis

The DES Round Function



DES Round Function (cont)

- F takes two inputs
  - 32 bit round value
  - 48 bits of key taken from 56 bit DES key
    - A different subset of 48 bits selected in each round
  - E is the “expansion” box
    - Turns each set of 4 bits into 6, by merely repeating some bits
  - S boxes take 6 bits back to 4 bits
    - Non-linear functions and they are the cryptographic heart of DES
    - S-boxes were tweaked by NSA back in the 70’s
    - It is believed that they IMPROVED DES by doing this

Full Description of DES

- If you want all the gory details

  http://en.wikipedia.org/wiki/DES

- Challenge Problem:
  - Alter the S-boxes of DES any way you like so that with ONE plaintext-ciphertext pair you can recover all 56 key bits
  - (Warning: you need some linear algebra here)

So if not DES, then what?

- Double DES?
- Let's write DES(K, P) as $DES_K(P)$
- Double DES (DDES) is a 64-bit blockcipher with a 112 bit key K = (K1, K2) and is

$$DDES_K = DES_{K2}(DES_{K1}(P))$$

- We know 112 bits is out of exhaustive search range… are we now secure?

Meet in the Middle Attack

- With enough memory, DDES isn't much better than single DES!
- Attack (assume we have a handful of pt-ct pairs P1,C1; P2, C2; …)
  - Encipher P1 under all 2^{56} possible keys and store the ciphertexts in a hash table
  - Decipher C1 under all 2^{56} possible keys and look for a match
  - Any match gives a candidate 112-bit DDES key
  - Use P2, C2 and more pairs to validate candidate DDES key until found

Meet in the Middle (cont)

- Complexity
  - $2^{56} + 2^{56} = 2^{57}$ DES operations
  - Not much better than the 2^{55} expected DES operations for exhaustive search!
  - Memory requirements are quite high, but there are techniques to reduce them at only a slightly higher cost
  - End result: no one uses DDES

How about Triple-DES!

- Triple DES uses a 168-bit key K=(K1, K2, K3)

$$TDES_K = DES_{K3}(DES_{K2}(DES_{K1}(P)))$$

- No known attacks against TDES
  - Provides 112-bits of security against key-search
  - Widely used, standardized, etc
  - More often used in “two-key triple-DES” mode with EDE format (K is 112 bits like DDES):

$$TDES_K = DES_{K1}(DES^{-1}_{K2}(DES_{K1}(P)))$$

  - Why is the middle operation a decipherment?

AES – The Advanced Encryption Standard

- If TDES is secure, why do we need something else?
  - DES was slow
  - DES times 3 is three times slower
  - 64-bit blocksize could be bigger without adding much cost
  - DES had other annoying weakness which were inherited by TDES
  - We know a lot more about blockcipher design, so time to make something really cool!

AES Competition

- NIST sponsored a competition
  - Individuals and groups submitted entries
    - Goals: fast, portable, secure, constrained environments, elegant, hardware-friendly, patent-free, thoroughly analyzed, etc
  - Five finalists selected (Aug 1999)
    - Rijndael (Belgium), MARS (IBM), Serpent (Israel), TwoFish (Counterpane), RC6 (RSA, Inc)
  - Rijndael selected (Dec 2001)
    - Designed by two Belgians

AES – Rijndael

- Not a Feistel construction!
  - 128 bit blocksize
  - 128, 192, 256-bit keysize
  - SP network
    - Series of invertible (non-linear) substitutions and permutations
  - Much faster than DES
    - About 300 cycles on a Pentium III
  - A somewhat risky choice for NIST

Security of the AES

- Some close calls last year (XL attack)
  - Can be represented as an overdetermined set of very sparse equations
  - Computer-methods of solving these systems would yield the key
  - Turns out there are fewer equations than previously thought
  - Seems like nothing to worry about yet

Block Ciphers – Conclusion

- There are a bunch out there besides AES and DES
  - Some are pretty good (IDEA, TwoFish, etc)
  - Some are pretty lousy
    - LOKI, FEAL, TEA, Magenta, Bass-O-Matic
- If you try and design your own, it will probably be really really bad
  - Plenty of examples, yet it still keeps happening

Aren’t We Done?

- Blockciphers are only a start
  - They take n-bits to n-bits under a k-bit key
  - Oftentimes we want to *encrypt* a message and the message might be less than or greater than n bits!
  - We need a “mode of operation” which encrypts any $M \in \{0,1\}^*$
  - There are many, but we focus on three: ECB, CBC, CTR

ECB – Electronic Codebook

- This is the most natural way to encrypt
  - It's what we used with the Substitution Cipher
  - For blockcipher E under key K:
  - First, pad (if required) to ensure $M \in (\{0,1\}^n)^+$
  - Write $M = M_1\ M_2\ \ldots\ M_m$ where each M_i has size n-bits
  - Then just encipher each chunk:
    - $C_i = E_K(M_i)$ for all $1 \leq i \leq m$
  - Ciphertext is $C = C_1\ C_2\ \ldots\ C_m$

ECB (cont)

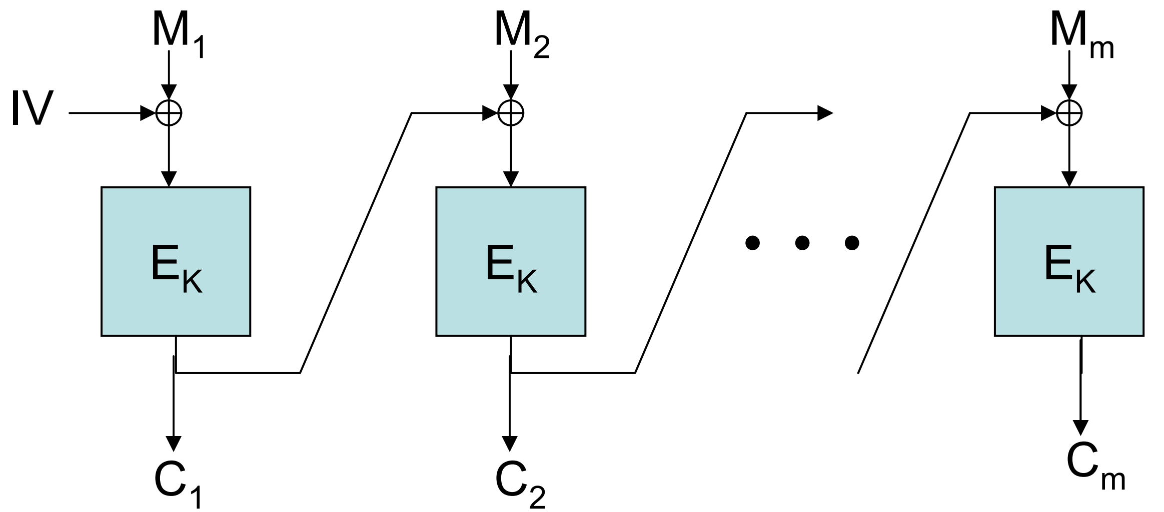
- What's bad about ECB?
  - Repeated plaintext blocks are evident in the ciphertext
    - Called "deterministic encryption" and considered bad
    - This was the feature of the Substitution Cipher that allowed us to do frequency analysis
    - Not as bad when n is large, but it's easy to fix, so why not fix it!
  - Encrypting the same M twice will yield the same C
    - Usually we'd like to avoid this as well

Goals of Encryption

- Cryptographers want to give up exactly two pieces of information when encrypting a message
  1) That M exists
  2) The approximate length of M
- The military sometimes does not even want to give up these two things!
  - Traffic analysis
- We definitely don’t want to make it obvious when a message repeats

CBC Mode Encryption

- Start with an n-bit “nonce” called the IV
  - Initialization Vector
  - Usually a counter or a random string
- Blockcipher E under key K, M broken into m blocks of n bits as usual
  - $C_0 = IV$
  - $C_i = E_K(M_i \oplus C_{i-1})$ for all $1 \leq i \leq m$



Features of CBC Mode

- Ciphertext is $C = C_0\ C_1 \ldots C_m$
  - Ciphertext expansion of n-bits (because of C_0)
- Same block M_i, or same message M looks different when encrypted twice under the same key (with different IV's)
- No parallelism when encrypting
  - Need to know C_i before we can encipher M_{i+1}
  - Decryption *is* parallelizable however
- CBC mode is probably the most widely-used mode of operation for symmetric key encryption

Digression on the One-Time Pad

- Suppose Alice and Bob shared a 10,000 bit string K that was secret, uniformly random
  - Can Alice send Bob a 1KB message M with “perfect” security?
  - 1KB is 8,000 bits; let X be the first 8,000 bits of the shared string K
  - Alice sets $C = M \oplus X$, and sends C to Bob
  - Bob computes $C \oplus X$ and recovers M
    - Recall that $M \oplus X \oplus X = M$

Security of the One-Time Pad

- Consider any bit of M, m_i, and the corresponding bits of X and C, (x_i, c_i)
  - Then $c_i = m_i \oplus x_i$
  - Given that some adversary sees c_i go across a wire, what can he discern about the bit m_i?
    - Nothing! Since x_i is equally likely to be 0 or 1
  - So why not use the one-time pad all the time?
    - Shannon proved (1948) that for perfect security the key must be at least as long as the message
      - Impractical

One-Time Pad (cont)

- Still used for very-top-secret stuff
  - Purportedly used by Russians in WW II
- Note that it is very important that each bit of the pad be used at most one time!
  - The infamous “two time pad” is easily broken
    - Imagine $C = M \oplus X$, $C' = M' \oplus X$
    - Then $C \oplus C' = M \oplus X \oplus M' \oplus X = M \oplus M'$
    - Knowing the xor of the two messages is potentially *very* useful
    - n-time pad for large n is even worse (WEP does this)

Counter Mode – CTR

- Blockcipher E under key K, M broken into m blocks of n bits, as usual
- Nonce N is typically a counter, but not required

$$C_0 = N$$

$$C_i = E_K(N{+}{+}) \oplus M_i$$

- Ciphertext is $C = C_0\ C_1\ \ldots\ C_m$

CTR Mode

- Again, n bits of ciphertext expansion
- Non-deterministic encryption
- Fully parallelizable in both directions
- Not that widely used despite being known for a long time
  - People worry about counter overlap producing pad reuse

Why I Like Modes of Operation

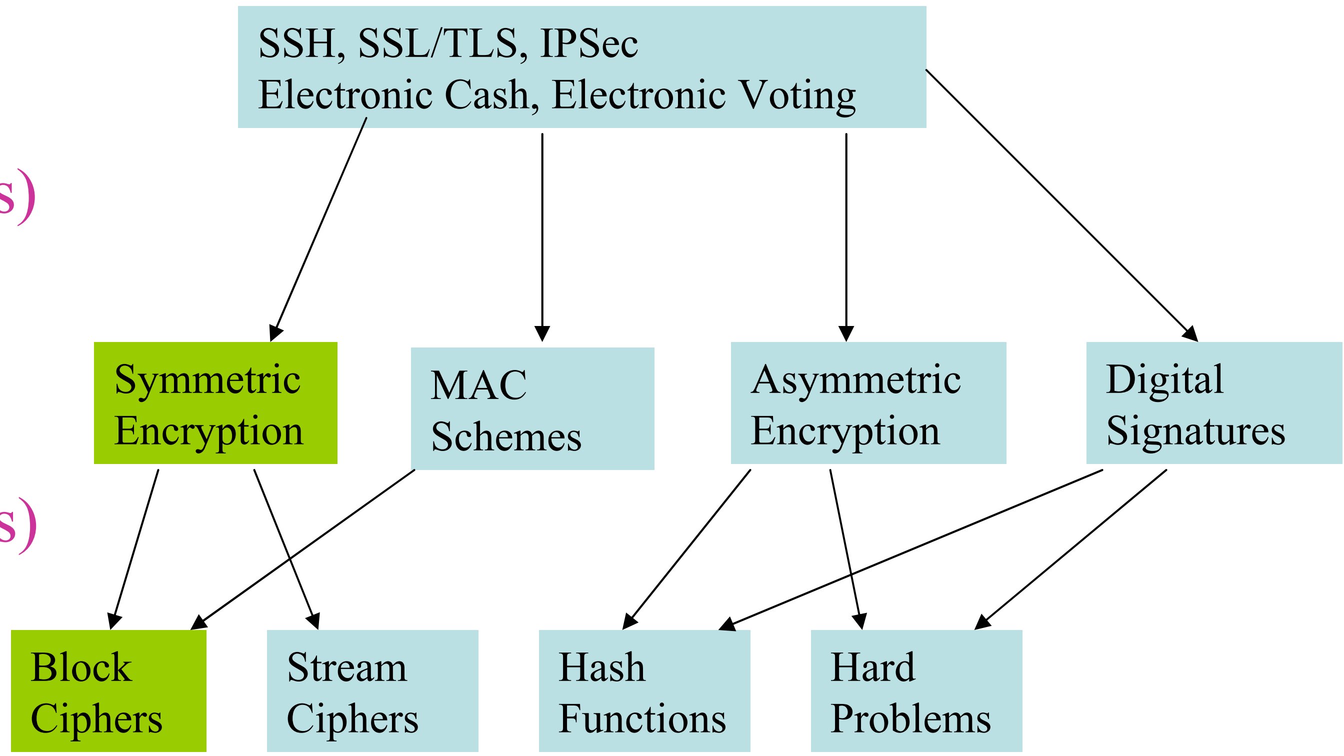
- Modes are "provably secure"
  - Unlike blockciphers which are deemed "hopefully secure" after intense scrutiny by experts, modes can be proven secure like this:
    - Assume blockcipher E is secure (computationally indistinguishable from random, as we described)
    - Then the mode is secure in an analogous black-box experiment
      - The proof technique is done via a "reduction" much like you did in your NP-Completeness class
      - The argument goes like this: suppose we could break the mode with computational resources X, Y, Z. Then we could distinguish the blockcipher with resources X', Y', Z' where these resources aren't that much different from X, Y, and Z

The Big (Partial) Picture

Second-Level Protocols
(Can do proofs)

First-Level Protocols
(Can do proofs)

Primitives



(No one knows how to prove security; make assumptions)