Warnings….

- Format Warning:
  - Today's slides are borrowed from CSE473 without being properly converted to this class's google slides format….
- Coverage Warning
  - Included are some details that we have not covered the background material for so we will gloss over some areas.

CSE 523S: Systems Security

Computer & Network Systems Security

Spring 2018

Jon Shidal

(slides borrowed from CSE473)

Plan for Today

- Questions
- Assignment
- System Design & Security
  - [x] Why are our computer systems vulnerable?
  - [x] Working with binaries and processes
  - [x] Why are our networks vulnerable?
  - Working with packets -- Next class
  - Network security revisited

Assignment

- For Monday
  - HW2 Due
  - Readings
    - HTAOE: Ch. 4 195-223

- For Wednesday
  - Readings
    - HTAOE: Ch. 3 115-132

- For Monday (2/19)
  - The following sections of [Metasploit Unleashed]
    - Introduction, Metasploit Fundamentals, Information Gathering, Vulnerability Scanning, Exploit Development

Principles of Network Security/ Internet Attacks and Defenses

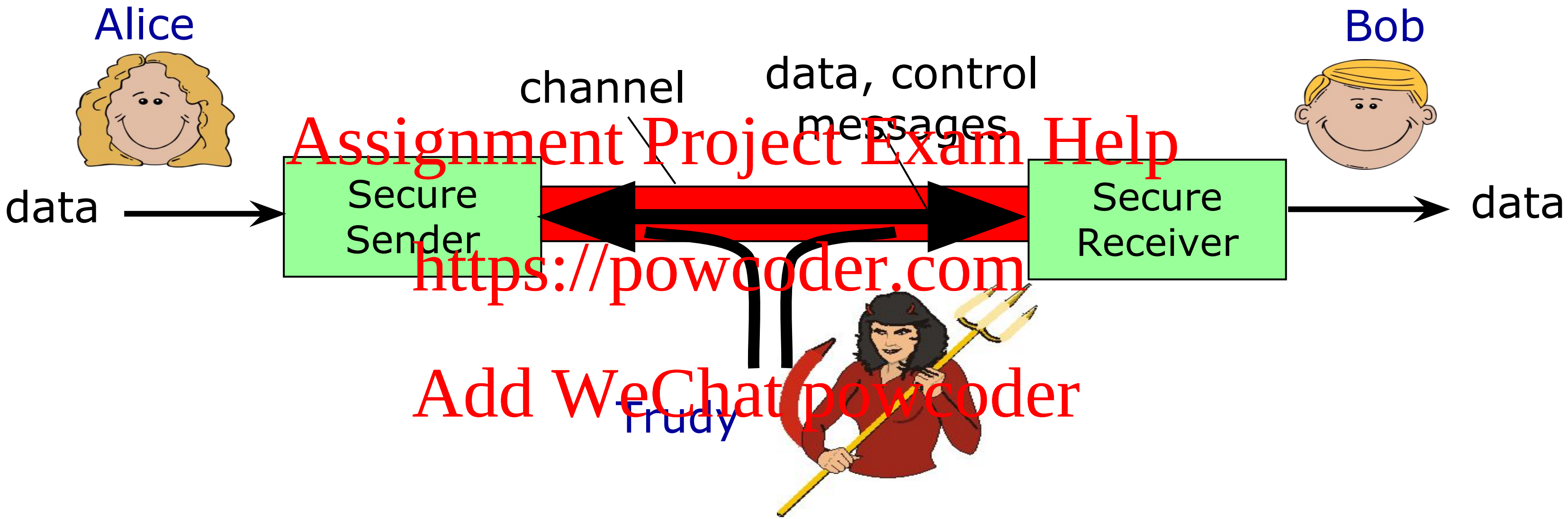
- Basic principles
- Symmetric encryption
- Public-key encryption
- Signatures, authentication, message integrity
- Denial-of-Service & Distributed Denial-of-Service

John DeHart
Based on material from Jon Turner, Roch Guerin and Kurose & Ross

Four Elements of Network Security

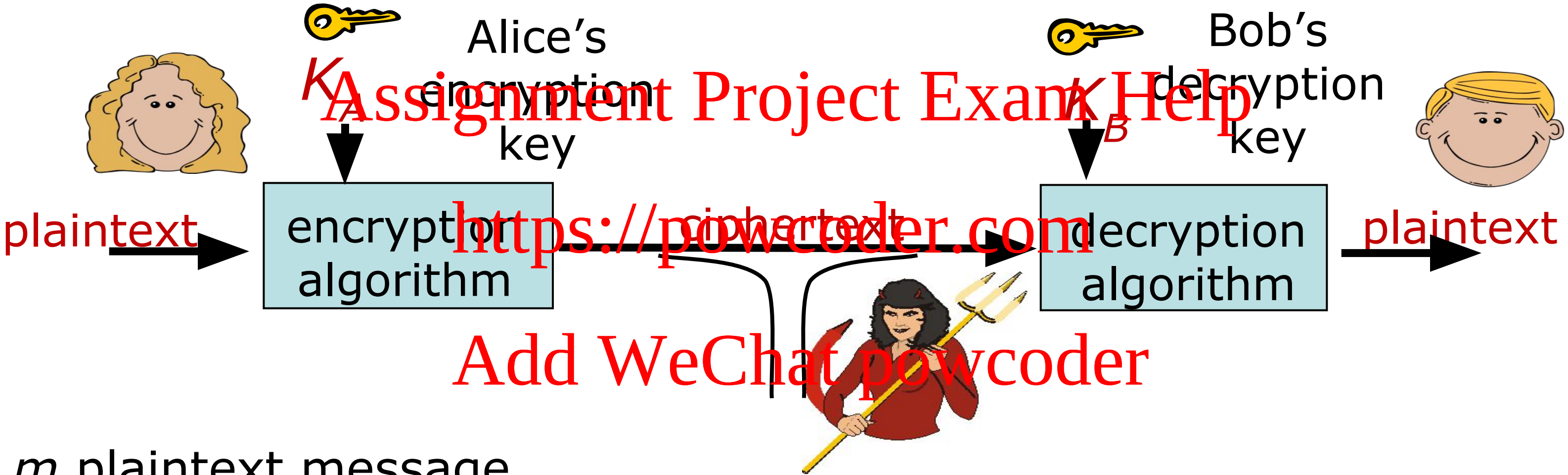
- ***Confidentiality***
  - » only sender, intended receiver should "understand" message
  - » sender encrypts message, receiver decrypts
- *Authentication*
  - » sender, receiver want to confirm identity of each other
  - » Use of "certification of authenticity" issued by trusted entity
- *Message integrity*
  - » sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- *Access and availability*
  - » services must be accessible and available to users

A Traditional Model of Security



- Alice & Bob want to communicate "securely"
- Trudy (intruder) may intercept, delete, add, and modify messages

The Language of Cryptography



m plaintext message

$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

Note that K_A and K_B need not be identical

i.e., symmetric vs. asymmetric encryption

Simple Encryption Scheme

- *Substitution cipher*
  - » substituting one thing for another
  - » Mono-alphabetic cipher: substitute one letter for another

plaintext: `abcdefghijklmnopqrstuvwxyz`

ciphertext: `mnbvcxzasdfghjklpoiuytrewq`

plaintext: `bob. i love you. alice`

ciphertext: `nkn. s gktc wky. mgsbc`

Encryption key: mapping from set of 26 letters to set of 26 letters (26! Possible mappings to choose from)

Breaking an Encryption Scheme

- Cipher-text only attack
  - » Trudy just has ciphertext she can analyze
  - » two approaches:
    - brute force: search through all keys
    - statistical analysis – *e.g.*, using fact that 'e' is most common letter
- Known-plaintext attack
  - » Trudy has at least some plaintext corresponding to ciphertext
  - » *e.g.*, in mono-alphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- Chosen-plaintext attack
  - » Trudy can get ciphertext for chosen plaintext
- Ideally, an encryption scheme should be resistant to even a chosen-plaintext attack

Block Cipher Encryption – (1)

- *Transposition block cipher*
  - Changing the order of the input
  - a.k.a. a scrambler

3-bit block: 1 2 3

3-bit transposed block: 2 3 1

input: 011 110 001 010 000

ciphertext: 110 101 010 100 000

Encryption key: permutation of *k*-bit blocks (*k*!=6 distinct permutations for *k*=3, *i.e.*, key of size $\lceil \log_2 k! \rceil$ or $\lceil \log_2 3! \rceil = 3$ bits)

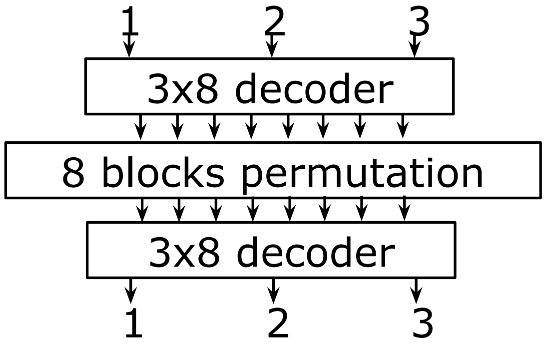
Why 3 bits? What do we use the 3 bits to identify?

Block Cipher Encryption – (2)

- <u>*Substitution*</u> *block cipher*
  - » Maps a *k*-bit block to another uniquely distinct *k*-bit block
  - » *k*-bit block input is one out of 2^k possible input
  - » Substitution applies permutation to all possible 2^k inputs

input: 011 110 001 010 000

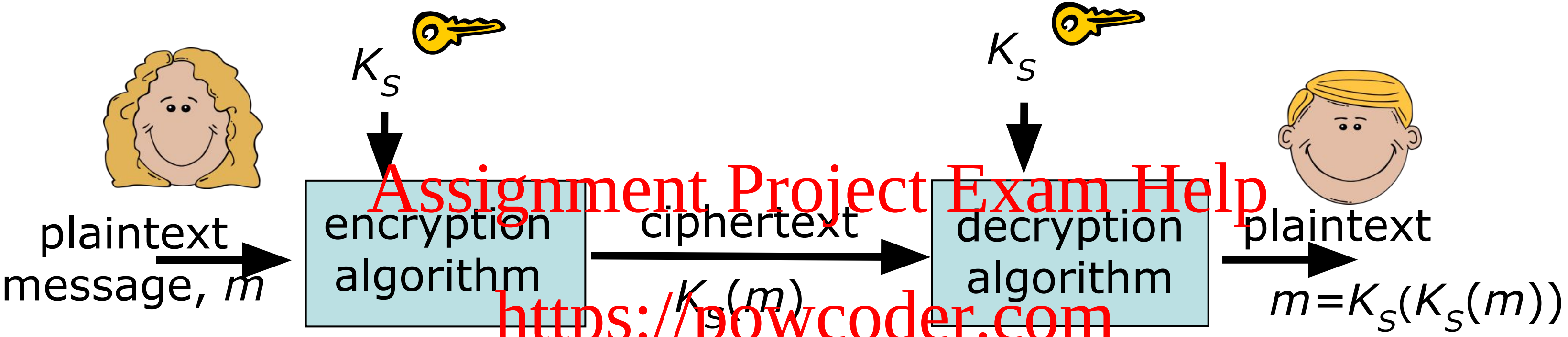
000	101
001	011
010	100
011	111
100	000
101	010
110	001
111	110



ciphertext: 111 001 011 100 101

Encryption key: permutation among 2^3=8 3-bit blocks (8!=40,320 distinct permutations, *i.e.*, key of size $\lceil \log_2 8! \rceil$ = 16 bits Why 16 bits? What do we use the 16 bits for?

Symmetric Key Cryptography



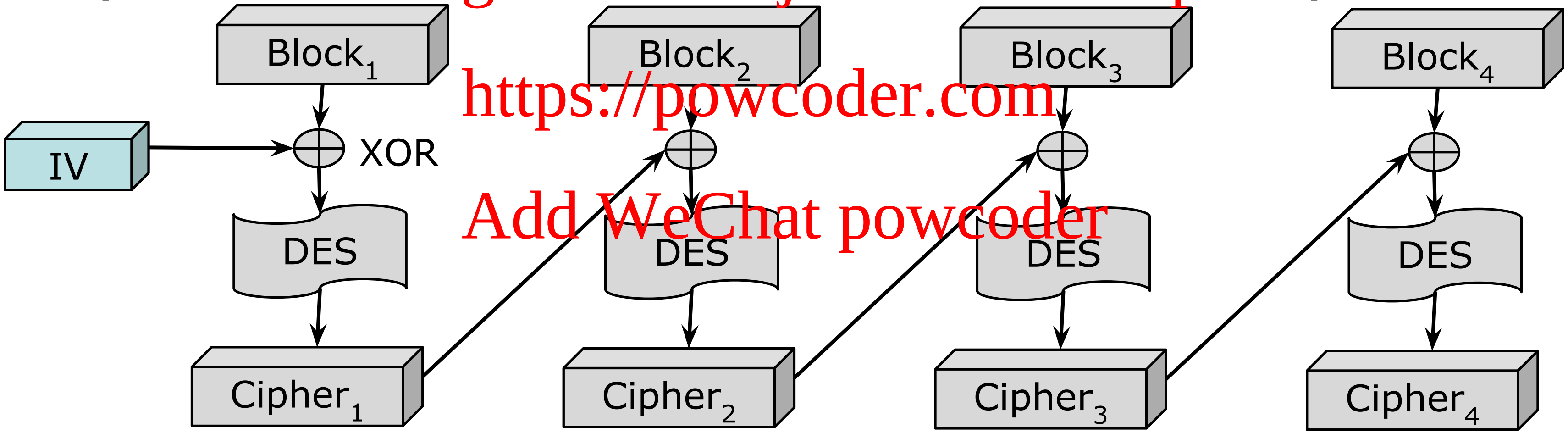
- Symmetric key cryptography
  - Bob and Alice share same (symmetric) key: K_S
  - ***e.g.***, key might be knowing the substitution pattern in mono alphabetic substitution cipher
- Main issue: how do Bob and Alice agree on key value?
  - need a separate, secure channel (to exchange key)
  - governments can use couriers, but that's not a practical solution for individuals over the Internet

Block Ciphers

- DES (Data Encryption Standard) is an example of a *block cipher*
  - » encrypts fixed length chunks separately (each chunk is a letter in an alphabet of size 2^k, where k is the chunk size in bits)
- Naive implementation can be vulnerable
  - » if each block is encrypted in the same way, repeated clear-text blocks produce repeated cipher-text blocks
  - » statistics of repeated blocks can aid attacker
- Cipher Block Chaining (CBC) used to address this
  - » makes identical clear-text blocks look different when encrypted
  - » example: each clear-text block m is xor-ed with a different "random" value before encryption
    - start with random *Initialization Vector* (IV) and xor this with first block before encrypting (IV sent to receiver, but need not be secret)
    - before encrypting each subsequent block, xor it with the ciphertext of the previous block

General Cipher Block Chaining

- Repeat across independent blocks
  (IV = Initial Vector – can be sent in the clear)



- Any other cipher block encryption can be used in lieu of DES

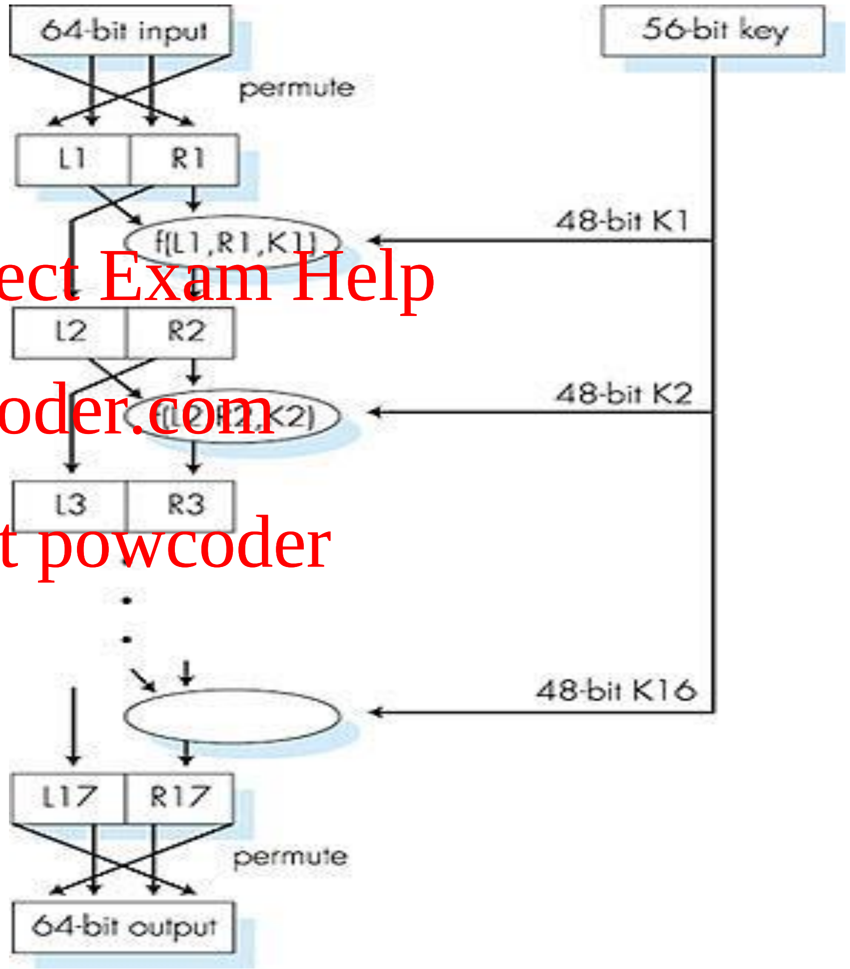
Data Encryption Standard (DES)

- Block cipher with cipher block chaining
  - » 56-bit symmetric key, 64-bit plaintext input
- How secure is it?
  - » DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day in January 1999
  - » no known good analytic attack
  - » Has been withdrawn as a NIST standard.
- More secure variant
  - » 3DES: encrypt 3 times with 3 different keys
  - » Advanced Encryption Standard (AES)
    - replaced DES in 2001
    - processes data in 128 bit blocks
    - 128, 192, or 256 bit keys
    - a computer that could break DES in one second (by brute force) would need 149 trillion years to break AES

DES Cipher

DES operation
(encryption by obfuscation)

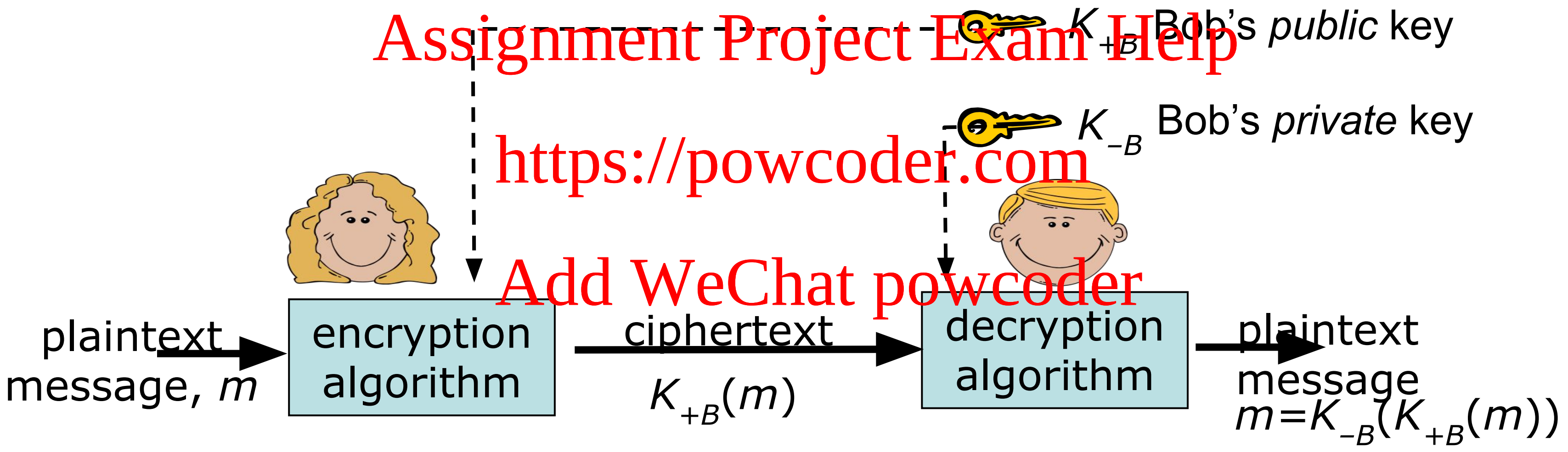
- encrypt 64 bit chunks
- initial permutation
- 16 identical "rounds" of function application, each using different 48 bits of key = F(56 bit key)
- final permutation



Public Key Cryptography

- The problem with symmetric keys
  - They require sender & receiver to know a shared secret key
  - ok for governments perhaps, but no good for public internet
- Public key cryptography
  - radically different approach [Diffie-Hellman76, RSA78]
  - built around idea of "one-way functions" that are easy to compute, but computationally difficult to invert
  - uses two keys
    - public key known to all (used to encrypt messages)
    - private key known only to message recipient (used to decrypt)
  - since no common shared key, allows communication with strangers over insecure network
  - drawback: computationally expensive for large messages
    - in practice, used to encrypt and share symmetric keys

Public Key Cryptography



One-Way Functions

- Function that is easy to compute, hard to invert
  - » example: easy to multiply two large prime numbers, but hard to find prime factors of a large composite number
    - no known method that is substantially better than trial-and error
    - a 300 digit number has about 10^{150} candidate factors
- Key idea leading to practical public-key encryption
  - » compute product of two large primes and make product public, while keeping prime factors private
  - » product can be used to encrypt message, but to decrypt it, you must know the prime factors
- RSA method based on this idea
  - » named for its inventors **R**ivest, **S**hamir and **A**delman
- Alternate one-way functions have been proposed
  - » based on variety of hard (NP-complete) computational problems

Background: Modulo Arithmetic

- x mod n = remainder of x when divided by n
- Basic properties

  $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$

  $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$

  $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$

- Consequently,

$(a \bmod n)^d \bmod n = a^d \bmod n$

$= [(a \bmod n)^1 \bmod n]*[(a \bmod n)^{d-1} \bmod n] \bmod n$

- Example: $a=14$, $n=10$, $d=3$:

  $(a \bmod n)^d \bmod n = (14 \bmod 10)^3 \bmod 10$

  $= 4^3 \bmod 10$

  $= 64 \bmod 10 = 4$

  $a^d = 14^3 = 2744 \quad a^d \bmod 10 = 4$

Creating an RSA Key Pair

1. Choose two large prime numbers p, q (say, 1024 bits long) and compute $n=pq$
2. Choose a number $e<(p–1)(q–1)$ with no common factor >1 with $(p–1)(q–1)$, *i.e.*, e and $(p–1)(q–1)$ are ***relatively prime***
3. Choose a number d such that ed–1 is a multiple of $(p–1)(q–1)$

   equivalently, $d = (k(p–1)(q–1)+1)/e$ for some positive integer k
4. Public key $K_+=(n,e)$, private key $K_-=(n,d)$
5. Advertise K_+ but keep K_- private, and discard (do not disclose) p and q (if p and q are known, e and d can be easily inferred)

Example with small numbers:

$$p=5,\ q=7,\ n=35,\ (p–1)(q–1)=24,\ e=5,\ d=29$$

(d = (6*4*6+1)/5 = 29   for k=6, p-1=4, q-1=6, e=5)

Dependent on having an efficient way to generate large prime numbers and efficient ways to select e and d

RSA Encryption/Decryption

Sending encrypted message to owner of (K_+ K_-)

- Given (n,e), (n,d) as discussed, and message $\mathbf{m<n}$
  - m MUST be less than n

- Encrypt by computing $K_+(m) = c = m^e \bmod n$
- Decrypt by computing $K_-(c) = c^{\,d} \bmod n = m$ (you need to know d to successfully decrypt a message)

- This works because

$$\begin{aligned} c^{\,d} \bmod n &= (m^e \bmod n)^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{ed \bmod (p-1)(q-1)} \bmod n \ * \\ &= m^1 \bmod n = m \ ** \end{aligned}$$

* by the magic of number theory (details on next slide)

** since $ed \bmod (p\text{-}1)(q\text{-}1) = 1$ by construction of d and $m<n$

From **number theory**, *p* & *q* prime with *n* = *pq* implies

$$a^b \bmod n = a^{b \bmod [(p-1)(q-1)]} \bmod n$$

So that

$$\left(m^e \bmod n\right)^d \bmod n = m^{ed} \bmod n$$

$$= m^{ed[\bmod (p-1)(q-1)]} \bmod n$$

$$= m^1 \bmod n$$

$$= m$$

Since *ed*=1 mod (*p*-1)(*q*-1)
by construction of *d*

Simple RSA Example

1. Pick $p=7$, $q=11$ prime
   » $n = pq = 77$, $z = (p-1)(q-1) = 60$
2. Choose Encryption key $e<z$ such that e & z are relatively prime:
   » $e = 17$
3. pick Decryption key d such that ed [mod z] = 1
   » $d = 53$ ($53 \times 17 = 901$ which mod 60 is 1)
4. Pub. Key: $(n,e)=(77,17)$; Priv. Key: $(n,d)=(77,53)$

- Assume message value of $m = 9$

  encode it as $c = 9^{17}$ [mod 77] = 4,

  decode this as 4^{53} [mod 77] = 9

Note: If too big, compute x^y mod v progressively,

i.e., $(x \bmod v)^y \bmod v$

Simple RSA Example

encode it as $c = 9^{17}$ [mod 77] = 4,

decode this as 4^{53} [mod 77] = 9

Note: If too big, compute x^y mod v progressively

i.e., (x mod v)^y mod v

$c = 9^{17}$ [mod 77] = ((9^2 mod 77)^8 * (9 mod 77)) mod 77

= ((81 mod 77)^8 * 9) mod 77

= ((4 mod 77)^8 * 9) mod 77

= ((256 mod 77) * (256 mod 77) * 9) mod 77

= (25 * 25 * 9) mod 77

= ((125 mod 77) * (5 * 9 mod 77)) mod 77

= (48 * 5 * 9) mod 77

= ((240 mod 77) * (9 mod 77)) mod 77

= (9 * 9) mod 77

= 4

Simple RSA Example

encode it as $c = 9^{17}$ [mod 77] = 4,

decode this as 4^{53} [mod 77] = 9

Note: If too big, compute x^y mod v progressively,

i.e., $(x \bmod v)^y \bmod v$

$$
\begin{aligned}
c = 9^{17} \text{ [mod 77]} &= ((9^2 \bmod 77)^8 * (9 \bmod 77)) \bmod 77 \\
&= ((81 \bmod 77)^8 * 9) \bmod 77 \\
&= ((4 \bmod 77)^8 * 9) \bmod 77 \\
&= ((4^6 \bmod 77) * (4^2 * 9 \bmod 77)) \bmod 77 \\
&= ((4096 \bmod 77) * (16 * 9 \bmod 77)) \bmod 77 \\
&= (15 * 16 * 9) \bmod 77 \\
&= (3 * 80 * 9) \bmod 77 \\
&= (3 * 3 * 9) \bmod 77 \\
&= 4
\end{aligned}
$$

More About RSA Operation

- To break RSA, need to find *d*, given *e* and *n*
  - » this can be done if we know $(p-1)(q-1)$, but that requires knowing *p* and *q*
  - » and that requires being able to factor *n*, which is hard

- Session keys

  - » exponentiation required by RSA is expensive for large values
    - because multiplication time grows as product of the number of bits

  - » in practice, use RSA to exchange "session keys" for use with symmetric encryption method like AES
- Keys can also be "reversed" – useful for authentication (coming next…)
  - » Sign with K_- (private) and verify signature with K_+ (public)

$$K_-(K_+(m)) = m^{ed} \bmod n = m = m^{de} \bmod n = K_+(K_-(m))$$

Elements of Network Security

- *Confidentiality*
  - » only sender, intended receiver should "understand" message
  - » sender encrypts message, receiver decrypts
- ***Authentication***
  - » sender, receiver want to confirm identity of each other
  - » Use of "certification of authenticity" issued by trusted entity
- *Message integrity*
  - » sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- *Access and availability*
  - » services must be accessible and available to users

Digital Signatures

- **Authentication**
- Digital signatures allow user to "sign" a document in a way that can't be forged

  - » this ensures that user cannot repudiate a signed document

- *A* can sign a message by "encrypting" it using *A*'s private key
  - » message can then be "decrypted" using *A*'s public key

  - » so long as no one but *A* has access to the private key, the message must have come from *A*
- *A* can also encrypt message using *B*'s public key to provide privacy
  - » $K_{+B}(K_{-A}(m))=c \;\; => \;\; K_{+A}(K_{-B}(c))=m$
  - » Only B can decrypt it and B can confirm it came from A.

Certificate Authorities

- Public-key systems require a secure way of making public keys available
  - » can't simply start by exchanging public keys in the clear, as this allows a "man-in-the-middle" attack
    - intruder, sitting between *A* and *B*, can substitute its own public key, causing *A* to encrypt messages using intruder's public key
    - intruder can then snoop on messages and re-encrypt using *B*'s public key, so *B* can't detect intrusion
- Certificate Authority (CA) vouches for the association between a user and their public key
  - » CA provides Bob with *signed certificate* of Bob's identity
    - CA encrypts Bob's identifier and public key using CA's private key
  - » so, Alice decrypts certificate using CA's public key
    - public keys for "reputable" CAs "built in" to browsers
  - » security depends on trustworthiness/reliability of CAs

Elements of Network Security

- *Confidentiality*
  - » only sender, intended receiver should "understand" message
  - » sender encrypts message, receiver decrypts

- *Authentication*

  - » sender, receiver want to confirm identity of each other
  - » Use of "certification of authenticity" issued by trusted entity

- ***Message integrity***
  - » sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- *Access and availability*
  - » services must be accessible and available to users

Verifying Message **Integrity**

- How do we prevent an intruder from tampering with messages?
  - » can encrypt and sign messages, but is this necessary?
- Use a *hash function h* to produce *message digest*
  - » sender computes $h(m)$ and sends pair $(m, h(m+s)=MAC)$
    - s is a ***shared secret***, hash value is ***M****essage* ***A****uthentication* ***C****ode*
  - » receiver computes $h(m+s)$ and compares to received value
  - » requires hash function that is hard to invert
    - MD5, SHA-1, SHA-2, SHA-3 are commonly used “cryptographic hash functions”
- Can also use this to reduce effort for digital signatures
  - » sender encrypts $h(m)$ and sends pair $(m, K\_(h(m)))$
  - » receiver computes $h(m)$ and compares it to received value, after decrypting it using sender’s public key

Elements of Network Security

- *Confidentiality*
  - » only sender, intended receiver should "understand" message
  - » sender encrypts message, receiver decrypts

- *Authentication*

  - » sender, receiver want to confirm identity of each other
  - » Use of "certification of authenticity" issued by trusted entity

- *Message integrity*
  - » sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- ***Access and availability***
  - » services must be accessible and available to users

Traffic Attacks & Defenses Overview

- **Access and Availability**
- Traffic attacks: The goal is to overwhelm the target's resources at either the network or host/application level
  - » Network attacks
    - DNS amplification attack: Requires access to open DNS server and use of spoofed addresses (that of the target)
    - Bandwidth flooding: If you have a large enough botnet, you can generate lots of traffic without resorting to address spoofing
  - » Application attacks
    - TCP SYN attack: Seeks to exhaust server state resources by opening lots of fake connections
    - HTTP GET flood: Same concept but with HTTP
    - TCP "shrew" attacks: takes advantage of TCP's own behavior (more on later slide)
- Defenses: Aimed at detecting, redirecting, and preventing attacking packets from reaching their target (or the target's network)
  - » Address filtering: Primarily aimed at countering address spoofing
  - » Unicast Reverse Path Filtering (uRPF): Discards traffic arriving from incorrect or invalid interface (only works when routing is symmetric)
  - » Black holes and sink holes: Used to attract unwanted traffic (backscatter) or redirect traffic for attack target

First Some Definitions

- Bogon prefix
  - » route that should never appear in an internet routing table.
    - Private, reserved, unallocated, etc.
  - » Often used by attackers as their source address.
  - » IANA (Internet Assigned Numbers Authority) maintains bogon list
  - » IPv4 bogon list is shrinking as address space is used up.
- Internet Background Noise (IBN)
  - » Packets addressed to addresses or ports where there is no network device to receive them.
- Backscatter
  - » IBN resulting from DDoS attack using spoofed addresses

Network Ingress Filtering

- Defeating Denial of Service Attacks which employ IP Source Address Spoofing – BCP 38 (RFC 2827)
  - » BCP: Internet **B**est **C**urrent **P**ractices
- Covers cases involving spoofing of both unreachable as well as valid addresses
  - » The latter can translate into a "double" attack, *i.e.*, the spoofed source may now be filtered by the domain under attack, or the response traffic may swamp the unwitting source, *e.g.*, as with a DNS amplification attack
- Filter traffic entering router from a known domain to ensure that source address is from that domain.

Black-Hole Router

- Helps identify attacks when they start, including on the network infrastructure itself

- Also called Network Telescope

  - Targets the dark/unused address space of Internet.
- Advertise reachability to prefix in bogon address space

- Inferring DDoS attacks from backscatter measurements
  - Assumes that attackers use randomly selected spoofed addresses, with "responses" from victims sent back to those random source addresses
  - Extrapolates frequency, magnitude, and types of attacks from backscatter responses sent to address located in a "quiet" /8 network (1/256th of the Internet address space)

Sink Holes

- The network equivalent of a honey pot: One or more dedicated network/router that seeks to attract or divert attack traffic and support its analysis
  - » A double monitoring and defense role
  - » Advertise host route for server under attack
    - Diverts all attack traffic to sink hole network
  - » Advertise default route in local domain
    - Pulls in all internal (and external) "junk" traffic, *e.g.*, to bogon address space
- Other uses
  - » Monitoring scanning of infrastructure addresses (pre-attack)
    - By advertising default route of routed for bogon IPs
  - » Monitoring activity on dark space (worms for locally infected clients)
  - » Capture backscatter, *i.e.*, responses (from attack victims) to bogon address space and addresses spoofed by attackers

DNS Attacks

- Redirecting traffic to an attacker by hijacking DNS replies
  - Faking a response to a query requires only spoofing a source address and guessing an ID field value (DNS has no authentication)
  - This together with DNS caching behavior makes it easy to implement various DoS attacks a.k.a. cache poisoning (setting the TTL value of the reply to a high value will ensure that resolvers keep the fake answer for a long time)
  - The scope of cache poisoning can range from a single client to a slave primary server handling an entire zone (the attack then targets the zone transfer messages)
  - DNSSEC (RFCs 4033, 4035) adds one-way authentication to DNS responses, *i.e.*, provides data integrity and origin authentication

DNS Attacks (continued)

- DNS Amplification Attack
  - » Attacker issues DNS request with source address spoofed to target machine
    - Request asks for large amount of data, type "ANY".
  - » Amplification is a function of the number of replies that can be directed to the host under attack, **and** the size of those replies (creating fake DNS records that can be used during attacks can significantly augment the size of the DNS replies)
- DNSSEC does not prevent DNS amplification attacks
  - » They only require spoofing the source address of DNS queries, but depend on access to open DNS servers

Application Layer attacks: Low-Rate TCP-Targeted Denial of Service Attacks

- Most servers now have mechanisms to defend against TCP SYN attacks, so attackers need to be a bit more creative
- Rather than blast traffic to swamp a server, take advantage of TCP’s behavior to mount effective attacks that are harder to detect (low rate)
- Relies on sending properly timed short periodic bursts of packets
  - » Packet bursts induce multiple losses and delay retransmissions for RTO
    - RTO: Retransmission TimeOut
  - » Another burst after another RTO can result in many/most flows experiencing repeated time-outs
- Effective even in the presence of flows with heterogeneous RTO and RTT values
  - » Select appropriate intermediate RTO value
  - » Can actually force the time-out synchronization of heterogeneous flows
- Neither router based schemes (RED-PD) nor end-host based schemes (RTO randomization) are able to successfully detect or diffuse the attacks

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

The End.