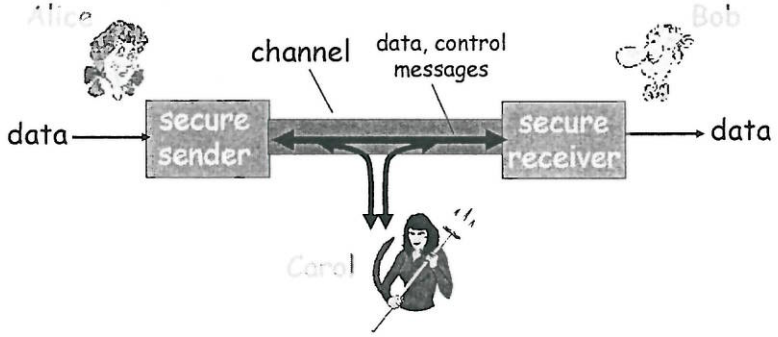
Network Security

- Introduction
- Symmetric-Key Cryptography
- Asymmetric-Key Cryptography
- Digital Signatures, X.509 Certs & PKI
- Authentication Protocols
- Secure Socket Layer (SSL)
- IPsec
- DNSSEC

What is Network Security?

- Confidentiality
  - Only sender, intended receiver should "understand" message contents
    - Sender encrypts message
    - Receiver decrypts message
- Authentication
  - Sender, receiver want to confirm identity of each other
- Message Integrity
  - Sender, receiver want to ensure message not altered (in transit, or after) w/o. detection

Friends and Enemies



- Well-known in network security world
- Bob, Alice want to communicate "securely"
- Carol (intruder) may intercept, delete, add messages

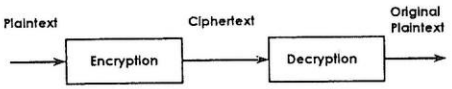
Who might Bob and Alice be?

- Well, real-life Bobs and Alices!
- Web browser/server for electronic transactions
  - e.g. on-line purchases
- On-line banking client/server
- DNS servers
- Routers exchanging routing table updates

What can the bad guys do?

- Passive Attack
  - Eavesdrop or intercept messages
- Active Attack
  - Actively insert messages into connection
- Impersonation
  - Fake (spoof) source address in packet (or any field in packet)
- Hijacking
  - "Take over" ongoing connection by removing sender or receiver, inserting himself in place

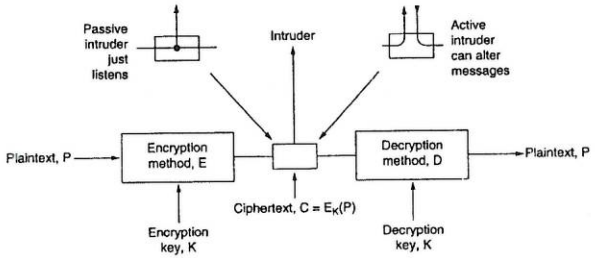
Cryptography



- Original data to be transferred is called Plaintext or Cleartext
  - Encrypted version is called Ciphertext
- Plaintext is denoted *P*, whereas ciphertext is denoted *C*
  - Encryption function *E* operates on *P* to produce *C*
- In the reverse process
  - Decryption function *D* operates on *C* to produce *P*
    - $D(C) = P$
- Following identity must also hold true for the cryptosystem to function correctly
  - $D(E(P)) = P$

Cryptographic Keys

- All modern encryption algorithms use a key denoted by *K*
- The key can take on many possible values
  - Range of possible values is called the key space



- The encryption and decryption functions now become
  - $E_K(P) = C$ and $D_K(C) = P$

Substitution Ciphers

- Each letter or a group of letters is replaced by another letter or group of letters to disguise it
- Caesar Cipher - Mono-alphabetical Substitution
  - In this system the alphabet is written out twice

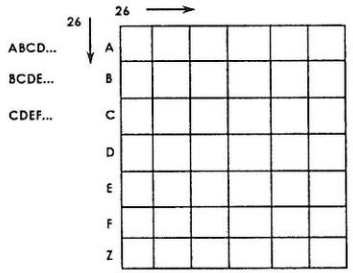
  a b c d e f g h i j k l m n o p q r s t u v w x y z
  D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
- Q: What is the key?
- To send a secret message
  - Letters of the message are taken one by one and the letters appearing below are written instead
- The message "send spears" would be enciphered as VHQG VSHDUV

Substitution Ciphers II

- Attacks
  - Identify commonly occurring characters
    - e, t, o, a, n, i
  - Commonly occurring bigrams/digrams
    - th, he, in, er
  - Domain specific buzz words
    - system, login, password, money etc...
- Substitution ciphers preserve the order of the text symbols but disguise them

The Vigenère Cipher

- Some protection from the above can be gained by using a poly-alphabetical cipher
  - Create matrix of 26 different alphabets



- Now pick a key e.g. AFGHANISTANBANANISTAN
  - Use row A to encrypt first letter of plaintext, row F the second time around etc...

Exercise: How can we break this cipher?

Transposition Ciphers

- Transposition ciphers reorder the symbols
  - Do not disguise them

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	l	x	t	w
o	t	w	o	a	b	c	d

Plaintext

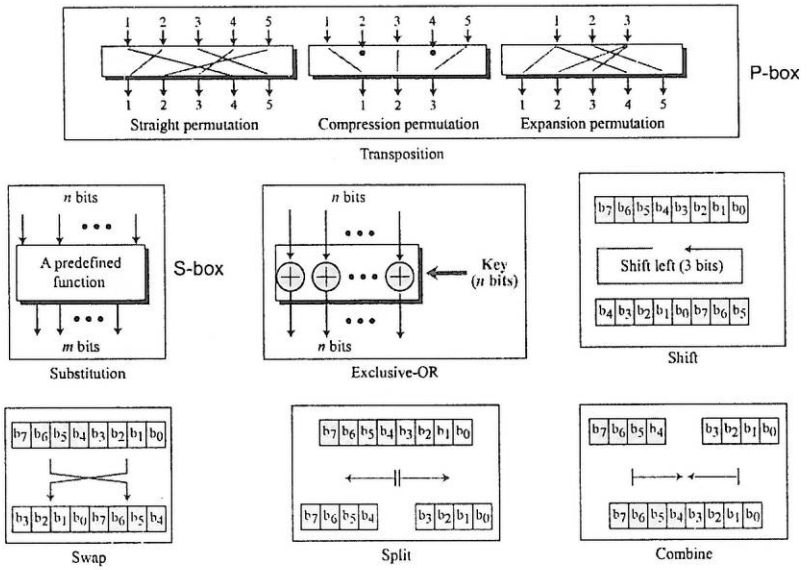
pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Ciphertext

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

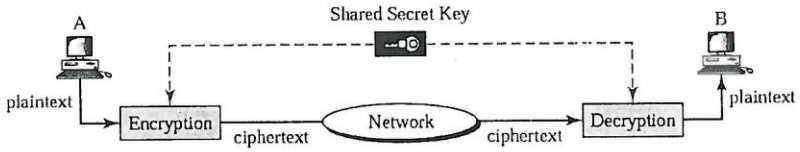
- The plaintext is written horizontally in rows
- Ciphertext is read out in columns
  - Starting with the column whose key is the lowest

Components of a Modern Block Cipher



Symmetric-Key Encryption

- Based on the sender and the receiver of a message knowing and using
  - The same (secret) key
  - Cryptosystem



- Sender uses the secret key to encrypt the message
  - Receiver uses the same secret key to decrypt the message

Q. How do Bob and Alice agree on a key value?

Key Management

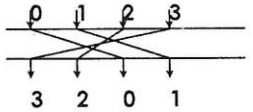
- Main problem is getting the sender and receiver to agree on a secret key without anyone else finding out
  - Especially if no cryptosystem in place to begin with
- Examples of symmetric key algorithms are
  - DES, Triple DES, IDEA, AES

Data Encryption Standard (DES)

- In January 1977 a standard encryption method was adopted by the U.S. gov
  - Origins lie in an internal IBM project codenamed Lucifer
- Though the algorithm used is complex
  - It is easily implemented in hardware
  - Software implementations are also widely available
- DES is a Block Cipher
  - Operates on a single chunk of data at a time
    - 64 bits (8 bytes)
  - Produces 64 bit output
- The key length is 56 bits
  - Often expressed as a 8 character string with the extra bits used as a parity check

DES

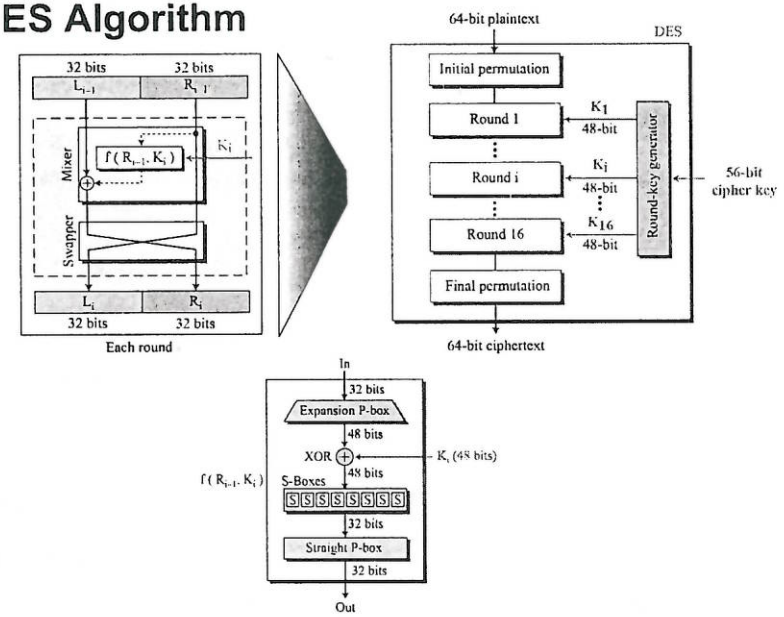
- Algorithm has 19 distinct stages
- First stage re-orders the bits of the 64-bit input block by applying a fixed permutation (P-box)



Transposition of 4 bits (example only)

- Last stage is the exact inverse of this permutation
- Stage penultimate to the last one
  - Exchanges the leftmost 32 bits with the rightmost 32 bits
- Remaining 16 stages are called Rounds
  - Functionally identical but take as an input a quantity computed from the key and the round number
    - Round key is 48 bits

DES Algorithm

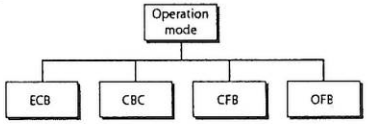


Cracking DES

- 56 bits is a short key
- Brute force attack (try every key)
  - 2^{56} encryptions to try all keys
  - Special chips can check 4 million keys/sec
  - $1 million DES cracking machine could break it in a few hours
- Jan'99 Challenge III won in 22 hrs and 15 mins using a supercomputer and 100,000 Internet nodes
  - Tested 245 billion keys per second
- Improvement - Triple DES or 3DES
  - Make use of 2 keys (112 or 168 bits)
    - Uses EDE or DED mode

2 keys 3 keys

Modes of Operation for Block Ciphers



- The previous discussion of DES is known as Electronic Code Book (ECB) mode
- Each 64-bit block is encoded *independently* of all other blocks
  - Block synchronization between the Tx and Rcv is not necessary
- Bit errors from noisy transmission lines only affect corresponding block
  - But not succeeding blocks
- Block ciphers operating in ECB mode can be parallelized
  - High-speed implementations

ECB Problems

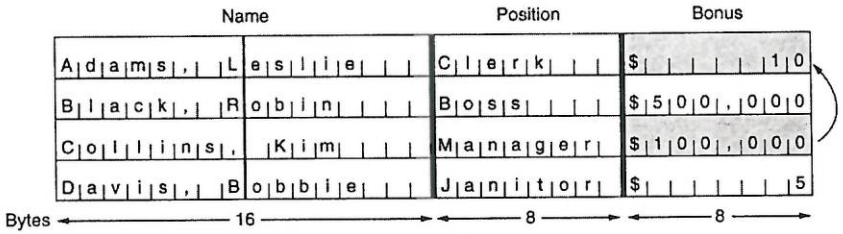
- ECB mode encrypts in a highly deterministic manner
- Identical plain text blocks result in identical ciphertext blocks
  - As long as the key doesn't change

Original Image

Encrypted with ECB Mode

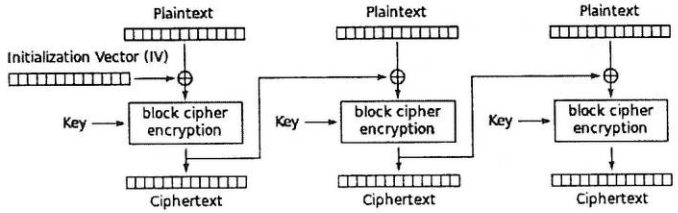
ECB Substitution Attacks

- Each block is encrypted independently of all other blocks
  - Allows for passive intruder to manipulate ciphertext blocks w/o receiver noticing



Cipher Block Chaining Mode (CBC)

- In CBC mode each block of plaintext is XORed with previous ciphertext block before being encrypted
- Each ciphertext block depends on all plaintext blocks processed up to that point



Cipher Block Chaining (CBC) mode encryption

- To make each message unique an initialization vector (IV) must be used in first block.

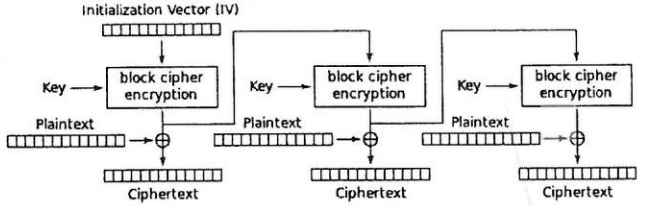
CBC Mode

Definition 5.1.2 Cipher block chaining mode (CBC)
Let $e()$ be a block cipher of block size b; let x_i and y_i be bit strings of length b; and IV be a nonce of length b.
Encryption (first block): $y_1 = e_k(x_1 \oplus IV)$
Encryption (general block): $y_i = e_k(x_i \oplus y_{i-1}), \quad i \geq 2$
Decryption (first block): $x_1 = e_k^{-1}(y_1) \oplus IV$
Decryption (general block): $x_i = e_k^{-1}(y_i) \oplus y_{i-1}, \quad i \geq 2$

- If we encrypt a string of blocks $x_1, \ldots, x_t$ once with a first IV and a second time with a different IV
  - Two resulting ciphertext sequences look completely unrelated to each other
- Note that we do not have to keep the IV secret
  - Make use of nonce i.e. use it only once

Output Feedback Mode (OFB)

- In OFB mode a block cipher is used to build a *stream cipher* encryption scheme
  - The key stream is not generated bitwise but in a blockwise fashion
- The output of the cipher outputs b key stream bits, where b is the width of the block cipher used
  - We can then encrypt b plaintext bits using the XOR operation



Output Feedback (OFB) mode encryption

OFB Mode

- The OFB mode forms a *synchronous stream cipher* as the key stream does not depend on the plain or ciphertext
- Encryption and decryption are exactly the same operation

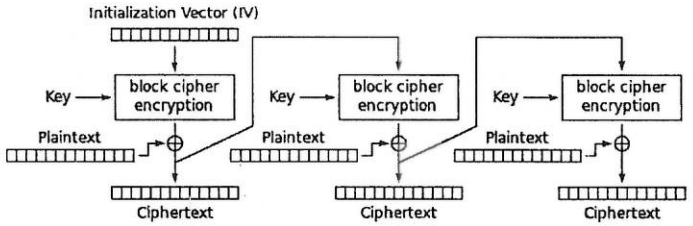
> **Definition 5.1.3** Output feedback mode (OFB)
> *Let $e()$ be a block cipher of block size b; let x_i, y_i and s_i be bit strings of length b; and IV be a nonce of length b.*
> ***Encryption (first block):*** $s_1 = e_k(IV)$ *and* $y_1 = s_1 \oplus x_1$
> ***Encryption (general block):*** $s_i = e_k(s_{i-1})$ *and* $y_i = s_i \oplus x_i, \quad i \geq 2$
> ***Decryption (first block):*** $s_1 = e_k(IV)$ *and* $x_1 = s_1 \oplus y_1$
> ***Decryption (general block):*** $s_i = e_k(s_{i-1})$ *and* $x_i = s_i \oplus y_i, \quad i \geq 2$

- One advantage of the OFB mode is that the block cipher computations are independent of the plaintext

– Can precompute several blocks s_i of key stream material

Cipher Feedback Mode (CFB)

- It is similar to the OFB mode but instead of feeding back the output of the block cipher, the ciphertext is fed back



Cipher Feedback (CFB) mode encryption

- CFB mode is an example of an *asynchronous stream cipher*
  - Since the stream cipher output is also a function of the ciphertext

CFB Mode

- A variant of the CFB mode can be used in situations where *short* plaintext blocks are to be encrypted
  - e.g. encryption of the link between a (remote) keyboard and a computer

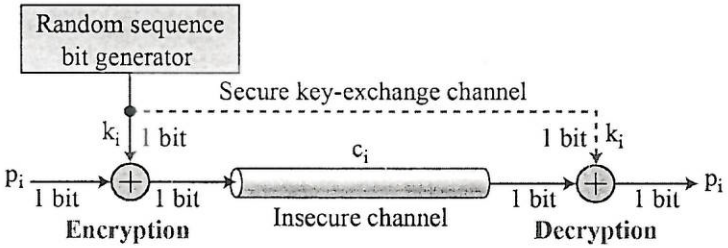
> **Definition 5.1.4** Cipher feedback mode (CFB)
> *Let $e()$ be a block cipher of block size b; let x_i and y_i be bit strings of length b; and IV be a nonce of length b.*
> ***Encryption (first block):*** $y_1 = e_k(IV) \oplus x_1$
> ***Encryption (general block):*** $y_i = e_k(y_{i-1}) \oplus x_i, \quad i \geq 2$
> ***Decryption (first block):*** $x_1 = e_k(IV) \oplus y_1$
> ***Decryption (general block):*** $x_i = e_k(y_{i-1}) \oplus y_i, \quad i \geq 2$

- Plaintexts generated by the keyboard are typically only 1 byte long, e.g., an ASCII character

– In this case, only 8 bits of key stream are used for encryption & ciphertext also only consists of 1 byte.

The Vernam Cipher

- Simplest and most secure stream cipher is called the One-time Pad
- Chooses a key stream (k) that is randomly chosen for each encipherment – makes use of XOR operator

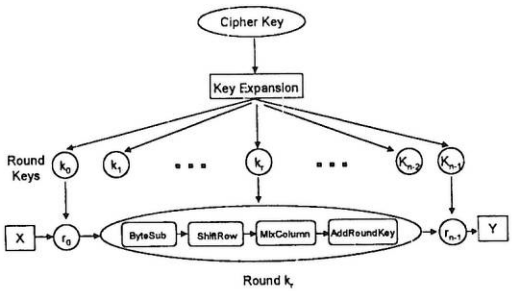


– $k \geq p$

Advanced Encryption Standard (AES)

- In 1997 NIST announced a call for proposals to develop a new Advanced Encryption Standard
  - After a long vetting process five algorithms were short listed
    - Rijndael (Rhine-doll) was eventually chosen as the new AES
- Symmetric cipher with variable key and block sizes of 128, 192 and 256 bits
  - Most common is 128 bit key & block size
  - Support for fast encryption and decryption in s/w - 700 Mbps
    - Can be implemented efficiently in smartcards
- A device that could check a 10^{18} AES keys/s would in theory require about 3×10^{51} years to exhaust the 256-bit key space
  - Brute force decryption taking 1 sec on DES, takes 149 trillion years for AES

AES



- The cipher consists of between 10 or 14 rounds (N_r)
  - Depending on key length (N_k) & block length (N_b)
- A plaintext block X undergoes n rounds of operations to produce an output block Y
  - Each operation is based on the value of the n^{th} round key
- Round keys are derived from the Cipher key by first expanding the key
  - Then selecting parts of the expanded key for each round

Asymmetric-Key Cryptosystems

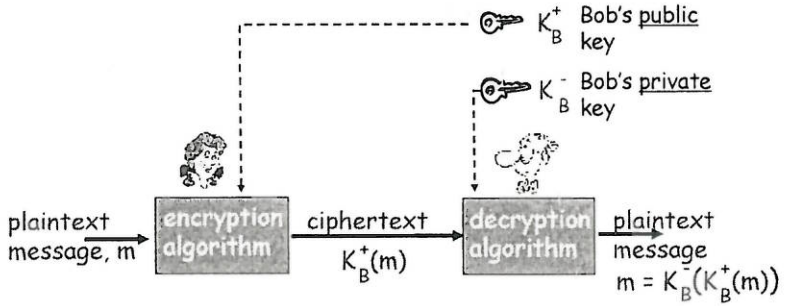
- Public key cryptography was invented by Diffie and Hellman in 1976
  - Solves the key management problem associated with symmetric key cryptosystems
- In public key cryptography each person generates a pair of keys
  - Public and private key
- Public key is published and widely distributed
  - While the private key is kept secret
- Examples of public-key cryptographic algorithms are
  - RSA, Diffie-Hellman, ElGamal, ECC

Properties of Asymmetric-Key Cryptosystems

- Must be computationally easy to encipher or decipher a message given the appropriate key
- Must be computationally infeasible to derive the private key from the public key
- Need for exchanging secret keys is eliminated
  - All secure comms now only involves public keys

Public-Key Cryptography

- Each user in a public-key system selects his own private key (K^-) and his own public key (K^+)



- When user Alice wants to send an encrypted message to Bob
  - she looks up his public key in public directory
  - Or obtains it by some other means

RSA

- De-facto standard algorithm for implementing asymmetric-key cryptography
  - Named after Rivest, Shamir and Adleman who developed it in 1978 at MIT
- It's security is based on the difficulty of factoring very large numbers
  - One way trap door function
- Example: Prime Factoring
  - Easy to calculate product of 2 large prime numbers
  - Difficult to calculate the prime factors from product

Modular Arithmetic I

- Most number sets we are used to are infinite e.g. set of real numbers
  - However most cryptographic algorithms are based on arithmetic with finite sets of numbers
- Consider the hours of a clock
  - 1*h*, 2*h*, 3*h*, …, 11*h*, 12*h*, 1*h*, 2*h*, 3*h*, …., 11*h*, 12*h*, 1*h*, 2*h*, 3*h*, ….

Modulo Operation

- Let *a*, *r*, *m* ∈ ℤ (where ℤ is the set of all integers) and *m* > 0
  - $a \equiv r \bmod m$
- *a* is said to be congruent to *r* mod *m* if *m* divides *a* - *r*
  - *m* is called the modulus and *r* is called the remainder

Modular Arithmetic II

- It is always possible to write $a \in \mathbb{Z}$ such that
  - $a = q \cdot m + r \quad 0 \le r < m$
- Since $a - r = q \cdot m$ (*m* divides $a - r$) we can now write
  - $a \equiv r \bmod m$
- Example : Let $a = 42$, $m = 9$
  - $42 = 4 \cdot 9 + 6$
  - $42 \equiv 6 \bmod 9$
- Q: $-11 \equiv x \pmod 7$

Multiplicative Inverse

- The integers modulo m, denoted $\mathbb{Z}_m$ is the set of integers {0, 1, 2, … m - 1}
  - Addition, subtraction and multiplication are performed modulo m
    - $\mathbb{Z}_m$ is referred to as an *Integer Ring*
- $\mathbb{Z}_{25}$ = {0, 1, 2, …, 24}
  - 13 + 16 = 4
    - $29 \equiv 4 \bmod 25$
- The multiplicative inverse of a modulo m is an integer $x \in \mathbb{Z}_m$ such that
  - $ax \equiv 1 \pmod m$
- The multiplicative inverse only exits for an element $a \in \mathbb{Z}_m$ iff
  - $\gcd(a, m) = 1$

Q: Does the multiplicative inverse of 15 exist in $\mathbb{Z}_{26}$?

Extended Euclidean Algorithm (EEA)

- Division of a by b modulo m is a product of a and b^{-1} modulo m
  - b*la* is equivalent to $a \cdot b^{-1} \pmod m$
- Q: What is 4^{-1} modulo 11?
  - $x \equiv 4^{-1} \pmod{11}$
    - $4 \cdot x \equiv 1 \pmod{11}$
- The modular multiplicative inverse of a modulo m can be found with the Extended Euclidean Algorithm
  - $s \cdot r_0 + t \cdot r_1 = \gcd(r_0, r_1)$
  - $s \cdot r_0 + t \cdot r_1 = 1$
  - $s \cdot 0 + t \cdot r_1 = 1 \bmod r_0$
  - $t = r_1^{-1} \bmod r_0$

Exercise: Compute 15^{-1} mod 26? $t = 7$

- gcd(11, 4) = 1
  - $11 = 2 \cdot 4 + 3$
  - $4 = 1 \cdot 3 + 1$
  - $3 = 3 \cdot 1 + 0$
- Back substitution
  - $1 = 4 - 1 \cdot 3$
  - $= 4 - 1 \cdot (11 - 2 \cdot 4)$
    $= 3 \cdot 4 - 1 \cdot 11$
    $= (-1) \cdot 11 + (3) \cdot 4$
  - $t = 3$
    - $4 \cdot 3 \equiv 1 \pmod{11}$

Euler's Totient Function $\phi(m)$

- No. of positive integers less than m and relatively prime to m
  - Relatively prime means gcd(a, m) = 1
- 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
- Q: What is $\phi(7)$ and $\phi(11)$?
  - 6 10
  - $\phi(m) = m - 1$ when m is prime

Euler's Phi Function

- In general let m have the following canonical factorization

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n}$$

  - where p_i are distinct primes and e_i are positive integers

$$\Phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1})$$

$\phi(10) = 2 \times 5$

$2^1 * 2^0 * 5^1 * 5^0$

$2 - 1 \times 5 + 1$

$1 \times 4 = 4$

- Exercise: Calculate $\phi(240)$? 64
  - Hint: $m = 240 = 16 \times 15 = 2^4 \cdot 3 \cdot 5$

Fermat's Little Theorem

- Let a be an integer and p be a prime, then
  - $a^p \equiv a \pmod{p}$
  - $a^{p-1} \equiv 1 \pmod{p}$
  - $a \cdot a^{p-2} \equiv 1 \pmod{p}$
    - $a^{-1} \equiv a^{p-2} \pmod{p}$
  - Thus we have a way for inverting an integer a modulo a prime
- Compute $4^{-1} \bmod 11$ ~use last formula

$4^9 \pmod{11}$

$4^9 = 262144 \equiv 3 \bmod 11$

RSA Algorithm

- Choose two large distinct primes p and q
- Compute the product (modulus)
  - $n = p \cdot q$

$\phi(n) = (p-1)\cdot(q-1)$

- Randomly choose an encryption key e, less than n that has no common factors with $\phi(n)$
  - e and $\phi(n)$ are relatively prime

• e is invertible iff $\gcd(e, \phi(n)) = 1$

- Finally compute the decryption key, d such that
  - $e \cdot d \equiv 1 \pmod{\phi(n)}$
  - $d \equiv e^{-1} \bmod \phi(n)$
  - $d \equiv e^{-1} \bmod (p-1)(q-1)$

RSA Usage

- The numbers e and n are the public key
  - The number d is the private key
- Break the plaintext message into a number of blocks
  - Represent each block as an integer
- Encryption
  - CiphertextBlock = $(\text{PlaintextBlock})^e \bmod n$
- Decryption
  - PlaintextBlock = $(\text{CiphertextBlock})^d \bmod n$
- Key Sizes
  - 1024, 2048, 3072, 7680 bits

• Recommended key size from 2015 onwards is 3072

RSA with Workable Numbers

- Let $p = 3$ and $q = 11$
- Using $n = p \cdot q = 33$
  - $\phi(n) = (p-1) \cdot (q-1) = 20$
- Choose $e = 3$ (e and $\phi(n)$ have no common factors)
- Solving $e \cdot d \equiv 1 \bmod 20$ and $d < 20$
  - $d \equiv e^{-1} \bmod 20$

- $d = 7$

• $7 \cdot 3 = 21 = 20 \cdot 1 + 1$ (EEA)

RSA Example

Plaintext (P)			Ciphertext (C)		After decryption	
Symbolic	Numeric	P^3	P^3 (mod 33)	C^7	C^7 (mod 33)	Symbolic
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
	Sender's computation			Receiver's computation		

- Since the primes chosen in this example are small, P must be less than 33
  - Each block can only contain a single character
- If p and $q \approx 2^{512}$, we would have $n \approx 2^{1024}$

- Each block would be up to 1024 bits or 128 bytes

Fast Exponentiation

- A straightforward way of exponentiation is like this:

$$x \xrightarrow{SQ} x^2 \xrightarrow{MUL} x^3 \xrightarrow{MUL} x^4 \xrightarrow{MUL} x^5 \ldots$$

  - Where *SQ* denotes squaring and *MUL* multiplication
- How many multiplications are required to compute x^8?

$$x \xrightarrow{SQ} x^2 \xrightarrow{MUL} x^3 \xrightarrow{MUL} x^4 \xrightarrow{MUL} x^5 \xrightarrow{MUL} x^6 \xrightarrow{MUL} x^7 \xrightarrow{MUL} x^8$$

- Alternatively we can compute

$$x \xrightarrow{SQ} x^2 \xrightarrow{SQ} x^4 \xrightarrow{SQ} x^8$$

- Wat about x^{26}?

$$x \xrightarrow{SQ} x^2 \xrightarrow{MUL} x^3 \xrightarrow{SQ} x^6 \xrightarrow{SQ} x^{12} \xrightarrow{MUL} x^{13} \xrightarrow{SQ} x^{26}$$

Square-and-Multiply Algorithm

- Based on scanning the bit of the exponent from the left (the most significant bit) to the right (the least significant bit)
- In every iteration
  - i.e., for every exponent bit the current result is squared
- Iff the currently scanned exponent bit has the value 1
  - A multiplication of the current result by x is executed following the squaring
- Example: We consider the exponentiation x^{26}. For the square-and-multiply algorithm, the binary representation of the exponent is:

$$x^{26} = x^{11010_2} = x^{(h_4 h_3 h_2 h_1 h_0)_2}$$

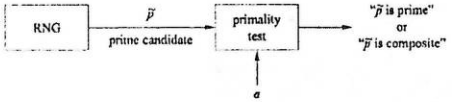
Square-and-Multiply Algorithm II

Step		
#0	$x = x^{1_2}$	inital setting, bit processed: $h_4 = 1$
#1a	$(x^1)^2 = x^2 = x^{10_2}$	SQ, bit processed: h_3
#1b	$x^2 \cdot x = x^3 = x^{10_2} x^{1_2} = x^{11_2}$	MUL, since $h_3 = 1$
#2a	$(x^3)^2 = x^6 = (x^{11_2})^2 = x^{110_2}$	SQ, bit processed: h_2
#2b		no MUL, since $h_2 = 0$
#3a	$(x^6)^2 = x^{12} = (x^{110_2})^2 = x^{1100_2}$	SQ, bit processed: h_1
#3b	$x^{12} \cdot x = x^{13} = x^{1100_2} x^{1_2} = x^{1101_2}$	MUL, since $h_1 = 1$
#4a	$(x^{13})^2 = x^{26} = (x^{1101_2})^2 = x^{11010_2}$	SQ, bit processed: h_0
#4b		no MUL, since $h_0 = 0$

- Modulo reduction is applied after each multiplication and squaring operation
  - In order to keep the intermediate results small

Finding Large Primes

- General approach is to generate integers at random which are then checked for primality



- Chance that a randomly picked integer $\tilde{p}$ is a prime is $\frac{1}{\ln(\tilde{p})}$
- In practice we only test odd nos so the probability doubles

$$P(\tilde{p} \text{ is a prime}) \approx \frac{2}{\ln(\tilde{p})}$$

Example: For RSA with a 1024-bit modulus n, the primes p and q each should have length 512 bits i.e. $p, q \approx 2^{512}$

$$P(\tilde{p} \text{ is a prime}) \approx \frac{2}{\ln(2^{512})} = \frac{2}{512\ln(2)} \approx \frac{1}{177}$$

Primality Tests

- A simple primality test can be based on Fermat's Little Theorem
  - $a^p \equiv a \pmod{p}$
  - $a^{p-1} \equiv 1 \pmod{p}$
- However there are certain composite numbers which may fulfill the above condition
  - In order to detect them the algorithm runs s times with different values of a
- Exercise: Is 221 a prime number? $a = 2, p = 221$ for test

$2^{220} \equiv 1 \pmod{221}$, should be 0 if prime

not prime

Miller-Rabin Primality Test

- Given the decomposition of an odd prime candidate $\tilde{p}$

$$\tilde{p} - 1 = 2^u r$$

  - where r is odd
- If we can find an integer a such that

$$a^r \not\equiv 1 \bmod \tilde{p} \quad \text{and} \quad a^{r2^j} \not\equiv -1 \bmod \tilde{p}$$

  - for all $j = \{0, 1, \ldots, u-1\}$, then $\tilde{p}$ is probably a prime

Miller-Rabin Primality Test II

should be $\not\equiv$ not $\equiv$

Example:

- If $\tilde{p} = 13$ then $\tilde{p} - 1 = 4 \cdot 3$, so $u = 2$, $r = 3$
  - $a^3 \equiv 1 \bmod \tilde{p}$ or $a^3 \equiv -1 \bmod \tilde{p}$ or $a^6 \equiv -1 \bmod \tilde{p}$ for each a from 1 to 12

Exercise: Is 561 a prime?

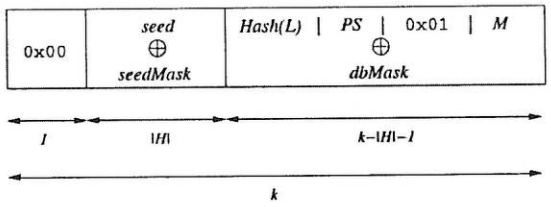
- If $\tilde{p} = 561$ then $\tilde{p} - 1 = 16 \cdot 35$, so $u = 4$, $r = 35$

$a^{35} \equiv 1 \bmod 561$

$-a^{35} \equiv 1 \bmod \tilde{p}$ or $a^{35} \equiv -1 \bmod \tilde{p}$ or $a^{70} \equiv -1 \bmod \tilde{p}$ or $a^{105} \equiv -1 \bmod \tilde{p}$ or $a^{140} \equiv -1 \bmod \tilde{p}$

RSA in Practice

- RSA encryption is *deterministic*
  - i.e. for specific key, particular plaintext is always mapped to particular ciphertext
- In practice RSA has to be used with a padding scheme
  - Optimal Asymmetric Encryption Padding (OAEP) for padding RSA messages are specified
    - Standardized in Public Key Cryptography Standard #1 (PKCS #1)

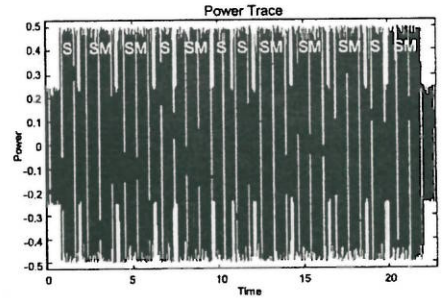


RSA Malleability

- A crypto scheme is said to be *malleable*
  - If an attacker is capable of transforming the ciphertext into another ciphertext, which leads to a known transformation of the plaintext
- Easily achieved in the case of RSA if the attacker replaces the ciphertext y by $s^e y$, where s is some integer
  - $y \equiv x^e \bmod n$
- Receiver computes
  - $(s^e y)^d \equiv s^{ed} x^{ed} \equiv sx \bmod n$
- If x were an amount of money which is to be transferred
  - By choosing $s = 2$ Trudy could exactly double the amount in a way that goes undetected by the receiver

RSA Side Channel Attacks

- Exploit information about the private key which is leaked through physical channels such as the
  - Power consumption or



- The figure shows the electric current drawn by the processor over time
- Our goal is to extract the private key d

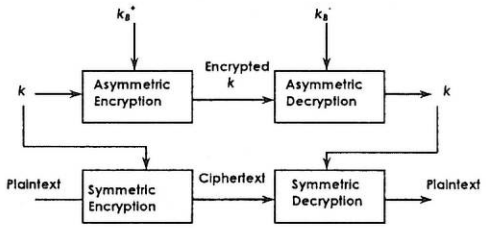
RSA Side Channel Attacks II

- We see that there are high activity intervals which are short and others which are longer
- Explained by the square-and-multiply algorithm
  - If an exponent bit has the value 0, only a SQ is performed
  - If an exponent bit has value 1, SQ + MUL is performed
- A long period of activity corresponds to the bit value 1 of the secret key
  - Short period to a key bit with value 0

operations:	*S SM SM S SM S S SM SM SM S SM*
private key:	0 1 1 0 1 0 0 1 1 1 0 1

Hybrid Schemes

- Asymmetric-key algorithms are not a replacement for symmetric-key algorithms such as DES or AES
  - Rather they supplement AES or any other fast bulk encryption cipher



- The above example shows
  - How we can use a public key algorithm to securely transfer a session key (k), and

- use session key for bulk encryption & decryption

An Important Property of RSA

The following *commutative* property will be very useful later

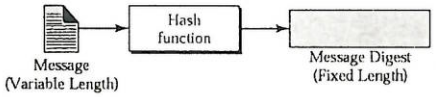
$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

use public key first, followed by private key

use private key first, followed by public key

result is the same!

Msg Auth & Integrity

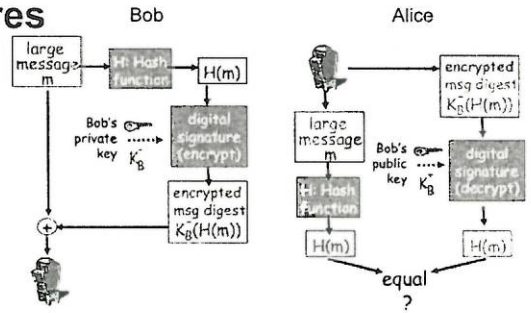


- Bob receives a msg from Alice, wants to ensure
  - Msg originally came from Alice - *Authentication*
  - Msg not changed since sent by Alice – *Integrity*

Message Digest/Cryptographic Hash

- A message digest is a strong digital fingerprint of a message
- Takes input m, produces fixed length value, $H(m)$
  - 128/160/256 bits
- Computationally infeasible to find two different messages, x and y such that
  - $H(x) = H(y)$
- Examples of message digest algorithms are
  - MD2, MD4, MD5, SHA-1 and SHA-2

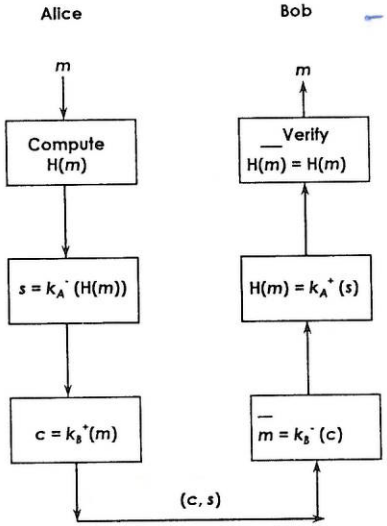
Digital Signatures



- Cryptographic technique analogous to hand-written signatures
- Sender (Bob) digitally signs the document establishing he is the document owner/creator
- Recipient (Alice) can prove to someone that Bob, and no one else (including Alice) must have signed document

- Verifiable & non-forgeable (Non-repudiation)

Enveloped and Signed Data



- Assuming public key is good.

Key Management

- When Alice obtains Bob's public-key (from a website, e-mail etc.), how does she know it is Bob's public key, not Trudy's?
- Public key cryptography is based on the idea that
  - An individual will generate a key pair
  - Keep one component secret and publish the other component (public key)
- Other users on the network
  - Must be able to retrieve this public key and associate the user's identity with it
- One way to form this association is to
  - Enlist services of trusted party (TTP)

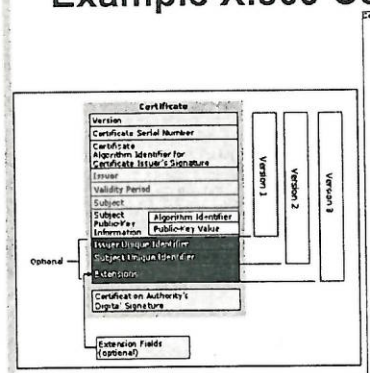
X.509 Certificates

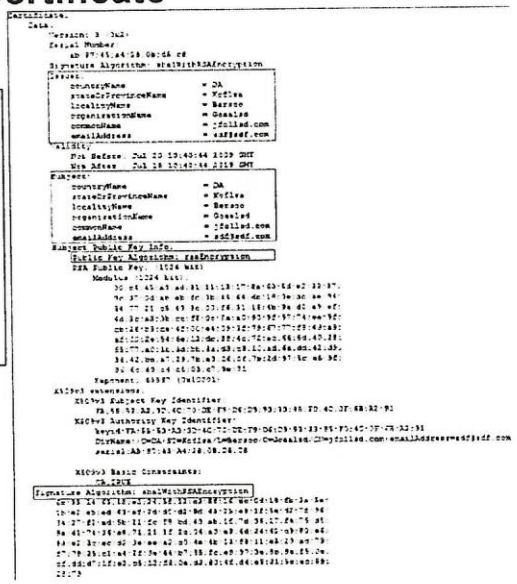
- The TTP will construct a message referred to as a Certificate

Subject (Identity of User)	Public Key	Validity Period	Issuer (Identity of TTP)	Other fields	Signature of TTP

- The cert contains a number of fields
  - Identity of the user
  - Public key of the user
  - .....
  - Digital signature on above fields w/ private key of TTP.
- Assumes that every user in the system is equipped with the public-key of the TTP
  - Allows one to verify the digital signature on the certificate
    - Guaranteeing that the public key is associated with the named user

Example X.509 Certificate

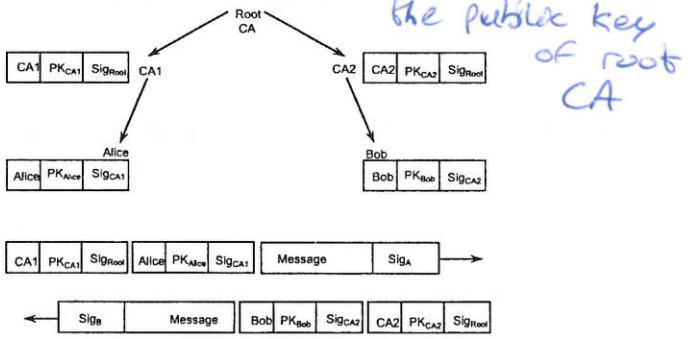




Certification Hierarchy

- TTPs that issue certificates are referred to as certification authorities (CAs)
- The root CA issues certificates only to other CAs

• Each user of system need only hold the public key of root CA

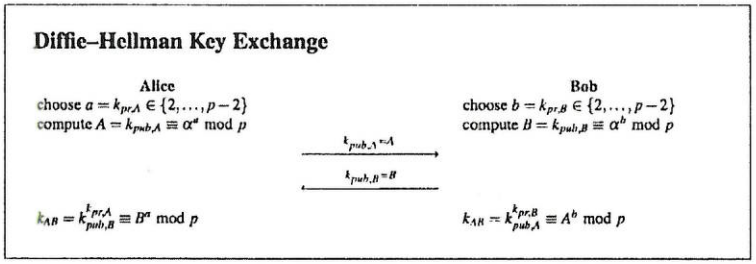


Diffie-Hellman Key Exchange (DHKE)

- A protocol that allows strangers to establish a shared symmetric key without them having to meet, and
  - Without the need for a cryptosystem to be in place!
- The basic idea behind DHKE is
  - That exponentiation in multiplicative group $\mathbb{Z}_p^*$ (p is prime) is one-way function, and
  - That exponentiation is commutative
    - $k \equiv (\alpha^x)^y \equiv (\alpha^y)^x \bmod p$
- The value $k \equiv (\alpha^x)^y \equiv (\alpha^y)^x \bmod p$ is a "joint secret"
  - Can be used as a *session key* between the two parties

DHKE

- Securely choose the *domain parameters*
  - Choose a large prime p
  - Choose an integer $\alpha \in \{2, 3, \ldots, p-2\}$
  - Publish p and α



- Exercise: Generate a DH key using the domain parameters p = 29 and α = 2. Assume that a = 5 and b = 12

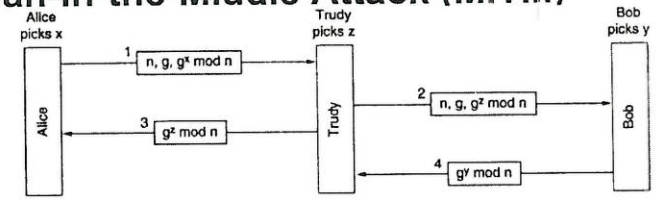
A=3, B=7, K_{AB} = 16

Security of DHKE

- The domain parameter p should have a length of 1024 bits (308 digits) or longer
- α should be a *primitive element* or *generator* of the group (G), whose powers modulo p generate all integers from 1 to p -1
  - Every element a of G can be written as
    - $\alpha^i = a$
- Q: Is 3 is a primitive element modulo $\mathbb{Z}_7^*$?

3, 2, 6, 4, 5, 1

Man-in-the-Middle Attack (MITM)



- Alice computes the key g^{xz} mod n
  - So does Trudy (for messages to Alice)
- Bob computes g^{yz} mod n
  - So does Trudy (for messages for Bob)
- Alice thinks she is talking to Bob and so establishes a session key (with Trudy) and so does Bob
  - Also known as the bucket brigade attack
- Exercise: What is the Diffie-Hellman Station-to-Station (STS) protocol? Google it

used in practice to stop MITM

Finite Fields

- A *finite field*, sometimes also called *Galois field*, is a set with a finite number of elements in which we can
  - add, subtract, multiply, invert
- Before we introduce the definition of a field, we first need the concept of a a simpler algebraic structure, a *group*
- A group is set with one operation and the corresponding inverse operation
  - If the operation is addition, the inverse operation is subtraction
  - If the operation is multiplication, the inverse operation is division (or multiplication with the inverse element)

Group

- *A* group *is a set of elements G together with an operation ∘ which combines two elements of G* and has the following properties
  - The group operation ∘ is *closed*
    - For all $a, b \in G$ it holds that $a \circ b = c \in G$
  - The group operation is *associative*
  - There is an element $1 \in G$, called the *identity element*
    - $a \circ (b \circ c) = (a \circ b) \circ c$ for all $a, b, c \in G$
    - $a \circ 1 = 1 \circ a = a$ for all $a \in G$
  - For each $a \in G$ there exists an element $a^{-1} \in G$, called the *inverse* of a
    - $a \circ a^{-1} = a^{-1} \circ a = 1$
  - A group G is *commutative*
    - If $a \circ b = b \circ a$ for all $a, b \in G$

Group II

- Example: The set of integers $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ and the operation addition modulo m form a group with the neutral element 0
  - Every element a has an inverse $-a$ such that $a + (-a) = 0 \bmod m$
  - Note that this set does not form a group with the operation multiplication because most elements a do not have an inverse
    - such that $a \cdot a^{-1} = 1 \bmod m$
- In order to have all four basic arithmetic operations (i.e., addition, subtraction, multiplication, division) in one structure
  - We need a set which contains an *additive* and a *multiplicative* group
    - This is what we call a field

Field

- A field *F* is a set of elements with the following properties
  - All elements of *F* form an *additive group* with the group operation "+" and the neutral element *0*
  - All elements of *F* except 0 form a *multiplicative group* with the group operation "·" and the neutral element *1*
  - When the two group operations are mixed, the *distributivity* law holds
    - For all $a,b,c \in F: a\cdot(b+c) = (a\cdot b)+(a\cdot c)$

Prime Fields

- The set $\mathbb{Z}_p$ (where *p* is a prime) is denoted as *GF*(*p*) and is referred to as a prime field - aka Galois field with a prime number of elements
- Elements of the field *GF*(*p*) can be represented by integers 0,1, . . . *p* − 1
  - All nonzero elements of *GF*(*p*) have an inverse
  - Arithmetic in *GF*(*p*) is done modulo *p*
- Two operations of the field are *modular integer addition* and *integer multiplication modulo p*
  - e.g. $GF(5) = \{0,1,2,3,4\}$

Finite Groups

- A group $(G, \circ)$ is finite if it has a finite number of elements
  - e.g. $(\mathbb{Z}_n,+)$, $(\mathbb{Z}_n,\cdot)$
- We denote the *cardinality* or *order* of the group *G* by |*G*|
  - $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$
    - $|\mathbb{Z}_n| = n$
  - $\mathbb{Z}_n^*$ consists of integers $i = 0,1, , n-1$ for which gcd $(i, n) = 1$
    - $|\mathbb{Z}_n^*| = \phi(n)$
    - $\phi(9) = 3^2 - 3^1 = 6$ i.e. elements $\{1,2,4,5,7,8\}$
- The order *ord*(*a*) of an element *a* of a group $(G, \circ)$ is the smallest positive integer such that
  - $a^k = a \circ a \circ \ldots \circ a = 1$

Finite Groups II

- Example: We try to determine the order of $a = 3$ in $\mathbb{Z}_{11}^*$
  - For this, we keep computing powers of *a* until we obtain the identity element 1
- $a^1 = 3$
- $a^2 = a \cdot a = 3 \cdot 3 = 9$
- $a^3 = a^2 \cdot a = 9 \cdot 3 = 27 \equiv 5 \pmod{11}$
- $a^4 = a^3 \cdot a = 5 \cdot 3 = 15 \equiv 4 \pmod{11}$
- $a^5 = a^4 \cdot a = 4 \cdot 3 = 12 \equiv 1 \pmod{11}$
- Q. What is a^6 and a^7 ?

Cyclic Groups

- A group G which contains an element α with maximum order i.e. $ord(\alpha) = |G|$ is said to be *cyclic*
- Elements with maximum order are called *primitive elements* or *generators*
- Exercise: Check if $a = 2$ is primitive element of $\mathbb{Z}_5^*$
- $ord(a) = 4 = |\mathbb{Z}_5^*|$
  - Implies that $a = 2$ is a primitive element and $|\mathbb{Z}_5^*|$ is cyclic
- For every prime p, $(\mathbb{Z}_p^*, \cdot)$ is a commutative finite cyclic group
  - i.e. the multiplicative group of every prime field is cyclic

Cyclic Groups II

- Let G be a finite cyclic group - then it holds that
  - The number of primitive elements of G is $\phi(|G|)$
  - If $|G|$ is prime, then all elements $a \neq 1 \in G$ are primitive

Example: Find the number of primitive elements in $\mathbb{Z}_5^*$

$= \phi(4)$

$= (2^2 - 2^1)$

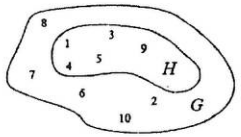
$= 2$

The primitive elements of $\mathbb{Z}_5^* = \{2, 3\}$

Exercise: Find the number of primitive elements in $\mathbb{Z}_{11}^*$?

Subgroups

- Let $(G, \circ)$ be a cyclic subgroup. Every element $a \in G$ with $ord(a) = s$ is the primitive element of a cyclic subgroup with s elements
  - i.e. any element of a cyclic group is the generator of a subgroup which in turn is cyclic
- Example: Consider a subgroup of $G = \mathbb{Z}_{11}^*$
  - $ord(3) = 5$ which generates the subgroup $H = \{1, 3, 4, 5, 9\}$



- More precisely it is a subgroup of prime order 5

– 3 not only generator of H, also 4,5 & 9

The Discrete Logarithm Problem (DLP) in $\mathbb{Z}_p^*$

- Given a finite cyclic group $\mathbb{Z}_p^*$ of order p - 1 and a primitive element $\alpha \in \mathbb{Z}_p^*$ and another element $\beta \in \mathbb{Z}_p^*$
- The DLP is determining an integer $1 \leq x \leq p$ - 1 such that
  - $\alpha^x \equiv \beta \bmod p$
- x is called the discrete logarithm of β to the base α

– $x = \log_\alpha \beta \bmod p$

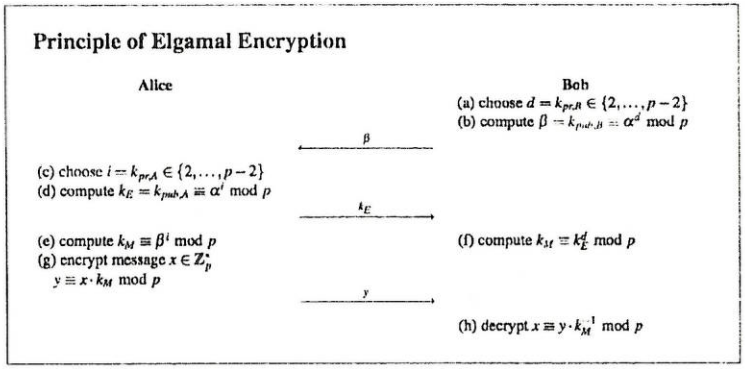
- Exercise: Find x given cyclic group $\mathbb{Z}_{47}^*$ in which $\alpha = 2$ is a primitive element and $\beta = 36$

– $2^x \equiv 36 \bmod 47$

The ElGamal Encryption Scheme

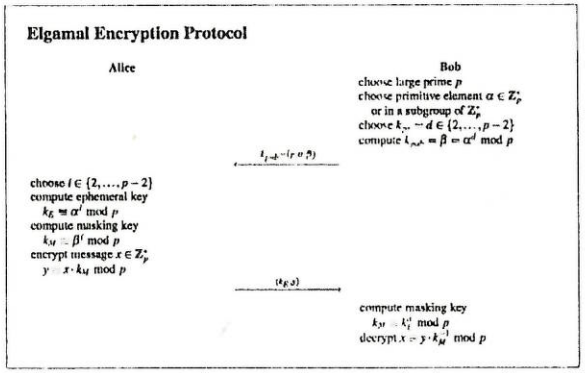
- The ElGamal encryption scheme can be viewed as an extension of the DHKE protocol
  - Its security is also based on the intractability of the discrete logarithm problem
- We consider the ElGamal encryption scheme over the group $\mathbb{Z}_p^*$, where p is a prime
- If Alice wants to send an encrypted message x to Bob, both parties first perform a DHKE to derive a shared key kM
  - Assume p and α have been generated
- The new idea is that Alice uses kM as a *multiplicative mask* to encrypt x

The ElGamal Encryption Scheme II



- Bob computes his private key d and public key β
  - This key pair does not change, i.e. can be used for encrypting many messages
- Alice, however, has to generate a new public–private key pair for the encryption of every message
  - Her private key is denoted by i and her public key by kE
    - The latter is an ephemeral key i.e. existing only temporarily
- The joint key is denoted by kM and is used for masking the plaintext

The ElGamal Protocol



- The set-up phase is executed once by the party who issues the public key and who will receive the message
- The encryption phase and the decryption phase are executed every time a message is being sent
- In contrast to the DHKE, no trusted third party is needed to choose a prime and primitive element

ElGamal Miscellaneous Issues

- ElGamal is a *probabilistic encryption scheme*. Encrypting two identical messages x_1 and x_2, where $x_1, x_2 \in \mathbb{Z}_p^*$ using the same public key
  - Results (with extremely high likelihood) in two different ciphertexts $y_1 \neq y_2$
  - Session key $k_M = \beta^i$ used for encryption is chosen at random for each encryption
- The ciphertext consists of two parts, the ephemeral key k_E and the masked plaintext y
- Since in general all parameters have a bit length of $\lceil \log_2 p \rceil$
  - The ciphertext (k_E, y) is twice as long as the message
    - Thus, the message expansion factor of ElGamal encryption is two

Attacks Against DLP

- The security of many asymmetric primitives is based on the difficulty of computing the DLP in cyclic groups, i.e.

  - To compute x for a given α and β in G such that

  $$\beta = \underbrace{\alpha \circ \alpha \circ \ldots \circ \alpha}_{x \text{ times}} = \alpha^x$$

- Brute-Force Search
  - A brute-force search is the most naive and computationally costly way for computing the discrete logarithm $\log_\alpha \beta$
    - We simply compute powers of the generator α successively until the result equals β

$$\alpha^1 \stackrel{?}{=} \beta$$
$$\alpha^2 \stackrel{?}{=} \beta$$
$$\vdots$$
$$\alpha^x \stackrel{?}{=} \beta$$

Brute-Force Search

- For a random logarithm x, we do expect to find the correct solution after checking half of all possible x
  - This gives us a complexity of $O(|G|)$ steps, where $|G|$ is the cardinality of the group
- To avoid brute-force attacks on DL-based cryptosystems in practice
  - The cardinality $|G|$ of the underlying group must thus be sufficiently large
- In the case of the group $\mathbb{Z}_p^*$, p prime, $(p - 1)/2$ tests are required on average to compute a discrete logarithm

  - $|G| = p-1$ should be at least in order of 280 bits

Elliptic Curve Cryptography (ECC)

- Elliptic Curve Cryptography is based on the generalized discrete logarithm problem of finding the integer x such that:

$$\beta = \underbrace{\alpha \circ \alpha \circ \ldots \circ \alpha}_{x \text{ times}} = \alpha^x$$

- ECC is more efficient

  - Fewer bits needed for same security, fewer computations

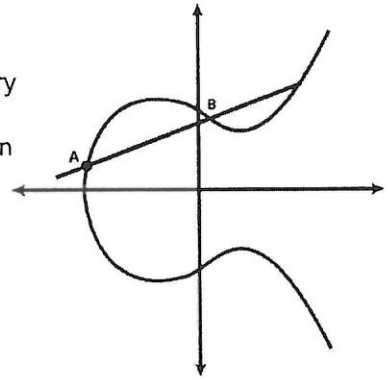
- An elliptic "curve" over $\mathbb{Z}_p$, $p > 3$ is the set of all pairs $(x, y) \in \mathbb{Z}_p$ which fulfill

$$y^2 = x^3 + ax + b \bmod p$$

- Together with an imaginary point $\mathcal{O}$, and the condition $4 * a^3 + 27 \cdot b^2 \neq 0 \bmod p$ where $a, b \in \mathbb{Z}_p$
  - The curve is non-singular i.e. has no self-intersections or vertices

Elliptic Curve Properties II

- An elliptic curve has several interesting properties
- One of these is horizontal symmetry
  - Any point on the curve can be reflected over the x-axis and remain the same curve



- A more interesting

Elliptic Curve Properties

$E_1 : Y^2 = X^3 - 3X + 3$ $\quad$ $E_2 : Y^2 = X^3 - 6X + 5$

- An amazing feature of elliptic curves is that there is a natural way to take two points on an elliptic curve and "add" them

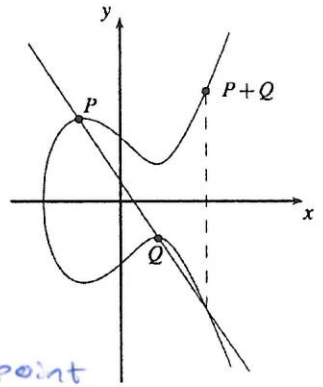
- To produce a 3rd point

Group Operations on Elliptic Curves

- Given two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ we compute the coordinates of the third point R such that

$$P + Q = R$$

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$



Point Addition P + Q

- Compute $R = P + Q$ where $P \neq Q$
- Draw a line through P and Q and obtain a third point of intersection between the elliptic curve and the line

• Mirror 3rd intersection point along x-axis to obtain R

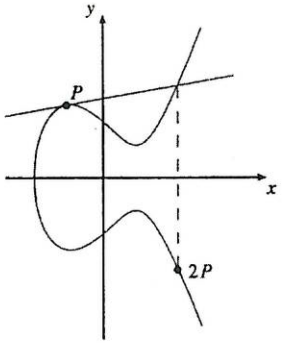
Group Operations on Elliptic Curves II

Point Doubling P + P

- Compute $R = P + Q$
  - where $P = Q$

• R = P+P = 2P

- Draw draw the tangent line through P
  - Obtain a second point of intersection between this line and the elliptic curve
- Mirror the point of the second intersection along the x-axis to obtain R



Computations on Elliptic Curves

Elliptic Curve Point Addition and Doubling Formulas

$$x_3 = s^2 - x_1 - x_2 \bmod p$$
$$y_3 = s(x_1 - x_3) - y_1 \bmod p$$

where

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \bmod p \text{ ; if } P \neq Q \text{ (point addition)} \\ \frac{3x_1^2 + a}{2y_1} \bmod p \text{ ; if } P = Q \text{ (point doubling)} \end{cases}$$

Example: Given E: $y^2 = x^3 + 2x + 2 \bmod 17$ and point $P = (5,1)$ compute $2P$?

$2P = P + P = (5,1) + (5,1) = (x_3, y_3)$

$s = \frac{3x_1^2 + a}{2y_1}$

$s = (2 \cdot 1)^{-1} \cdot (3 \cdot 5^2 + 2) = 2^{-1} \cdot 9 \equiv 9 * 9 \equiv 13 \bmod 17$

$x_3 = s^2 - x_1 - x_2 = 13^2 - 5 - 5 = 159 \equiv 6 \bmod 17$

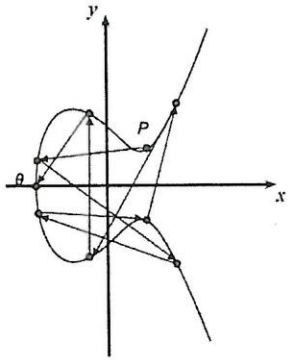
$y_3 = s(x_1 - x_3) - y_1 = 13(5 - 6) - 1 = -14 \equiv 3 \bmod 17$

$2P = (5, 1) + (5, 1) = (6, 3)$

Computations on Elliptic Curves II

- The points on an elliptic curve and the point at infinity $\mathcal{O}$ form cyclic subgroups

2P = (5,1) + (5,1) = (6,3)	11P = (13,10)
3P = 2P+P = (10,6)	12P = (0,11)
4P = (3,1)	13P = (16,4)
5P = (9,16)	14P = (9,1)
6P = (16,13)	15P = (3,16)
7P = (0,6)	16P = (10,11)
8P = (13,7)	17P = (6,14)
9P = (7,6)	18P = (5,16)
10P = (7,11)	19P = $\mathcal{O}$



- This elliptic curve has order $\#E = |E| = 19$ since it contains 19 points in its cyclic group

Number of Points on an Elliptic Curve

- Determining the point count on elliptic curves in general is hard
- Hasse's theorem bounds the number of points to a restricted interval

> **Theorem 9.2.2** Hasse's theorem
> *Given an elliptic curve E modulo p, the number of points on the curve is denoted by #E and is bounded by:*
>
> $$p+1-2\sqrt{p} \leq \#E \leq p+1+2\sqrt{p}.$$

- The number of points is "close to" the prime p
  - e.g. To generate a curve w/ about 2^{160} points
  - A prime w/ length of about 160 bits is required

Elliptic Curve Discrete Logarithm Problem

- ECC cryptosystems are based on the idea that d is large, kept secret, and attackers cannot compute it easily

> **Definition 9.2.1** Elliptic Curved Discrete Logarithm Problem (ECDLP)
> *Given is an elliptic curve E. We consider a primitive element P and another element T. The DL problem is finding the integer d, where $1 \leq d \leq \#E$, such that:*
>
> $$\underbrace{P+P+\cdots+P}_{d \text{ times}} = dP = T. \qquad (9.2)$$

- If d is known, an efficient method to compute the point multiplication dP is required to create a reasonable cryptosystem

Double-and-Add Algorithm for Point Multiplication

- Point multiplication is analog to exponentiation in multiplicative groups
- In order to do it efficiently, we can directly adopt the square-and-multiply algorithm (RSA)
  - Only difference is that squaring becomes doubling & Multiplication becomes addition

Algorithm

Input: Elliptic curve E, an elliptic curve point P and a scalar d with bits d_i

Output: $T = d\,P$

Initialization: $T = P$

FOR $i = t-1$ DOWNTO 0
  $T = T + T \bmod n$
  IF $d_i = 1$
    $T = T + P \bmod n$
RETURN (T)

Example of Double-and-Add Algorithm

- We consider the scalar multiplication 26 P, which has the following binary representation:

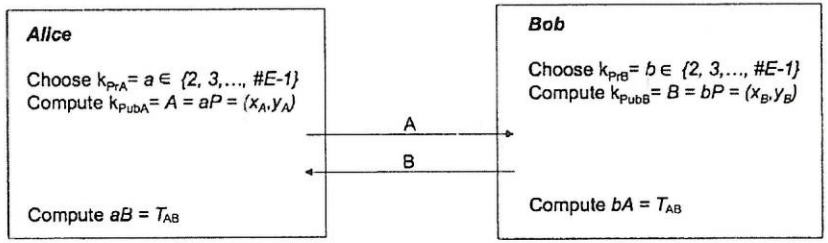
  $26\,P = (11010_2)\,P = (d_4d_3d_2d_1d_0)_2\,P$

- The algorithm scans the scalar bits starting on the left with d_4 and ending with the rightmost bit d_0

Step		
#0	$P = \mathbf{1}_2 P$	inital setting, bit processed: $d_4 = 1$
#1a	$P+P = 2P = \mathbf{10}_2 P$	DOUBLE, bit processed: d_3
#1b	$2P+P = 3P = 10_2 P + 1_2 P = \mathbf{11}_2 P$	ADD, since $d_3 = 1$
#2a	$3P+3P = 6P = 2(11_2 P) = \mathbf{110}_2 P$	DOUBLE, bit processed: d_2
#2b		no ADD, since $d_2 = 0$
#3a	$6P+6P = 12P = 2(110_2 P) = \mathbf{1100}_2 P$	DOUBLE, bit processed: d_1
#3b	$12P+P = 13P = 1100_2 P + 1_2 P = \mathbf{1101}_2 P$	ADD, since $d_1 = 1$
#4a	$13P+13P = 26P = 2(1101_2 P) = \mathbf{11010}_2 P$	DOUBLE, bit processed: d_0
#4b		no ADD, since $d_0 = 0$

Elliptic Curve Diffie-Hellman Key Exchange (ECDH)

- Given a prime p, a suitable elliptic curve E and a point $P = (x_P, y_P)$
  - The ECDH Key Exchange is defined by the following protocol



- Joint secret between Alice and Bob: $T_{AB} = (x_{AB}, y_{AB})$
  - One of the coordinates of point T_{AB} (usually x coordinate) can be used as session key
    - Often after applying a hash function

ElGamal with Elliptic Curves

- As before choose a suitable elliptic curve E and a point $P = (x_P, y_P)$

Bob chooses $k_{PrB} = b$
Computes $K_{PubB} = B = bP$

B ←

Alice chooses $k_{PrA} = a$
Computes $C_1 = kP$
Encrypts the msg x: $C_2 = x + Bk$

→ C_1, C_2

Bob computes $C_2 - bC_1$
Decrypts to get $x = x + bPk - bkP$

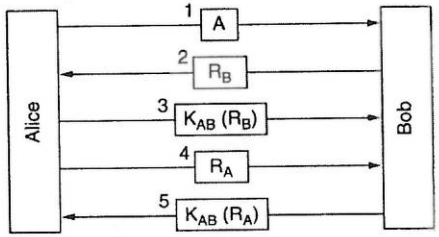
Key Lengths and Efficiency

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

- 256-bit ECC key provides the same security as a 3072-bit RSA key
  - Can be up to 10 times faster depending on implementation

Authentication Protocols

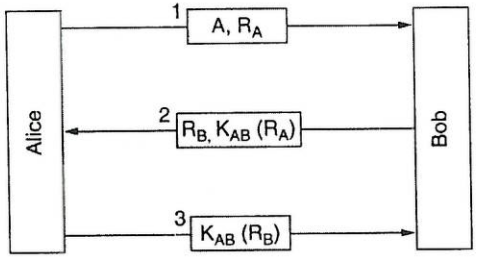
- Authentication is the technique by which a process verifies that a communicating partner is who it is supposed to be and not an impostor
- Authentication based on a *shared secret key*



- The above is an example of a *challenge-response* protocol

-R_B is a random number which is used only once, known as nonce

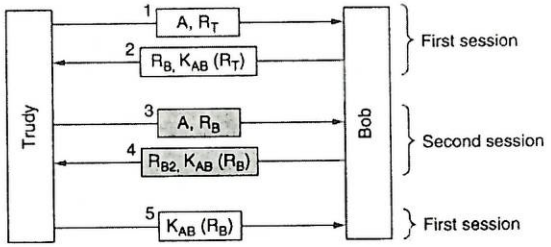
A Shortened Two-Way Protocol



Q: Can you see a problem?

The Reflection Attack

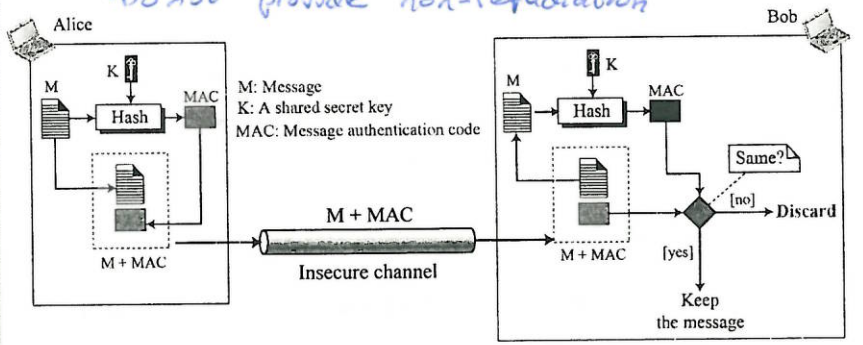
- Trudy can break this if it is possible to open multiple sessions with Bob
  - e.g. if Bob is a bank and is prepared to accept many connections from teller machines at once



Message Authentication Code from Hash Functions (HMAC)

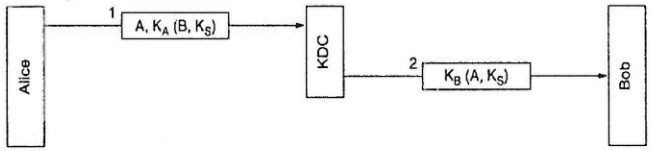
- Calculated using a cryptographic hash function in combination with a *secret key*
- Like digital signatures they can be used to quickly verify *data integrity* and *authenticity* of a msg

-Do not provide non-repudiation



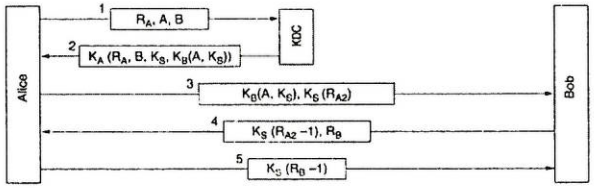
Authentication Using a KDC

- To talk to *n* people using a shared secret would require setting up how many keys?
  - $n \cdot (n-1)/2$
- Alternative is to introduce a Key Distribution Center (KDC)
- Each user has a *single shared key* with the KDC
  - Authentication & session key management happens through the KDC



Needham-Schroder Authentication Protocol

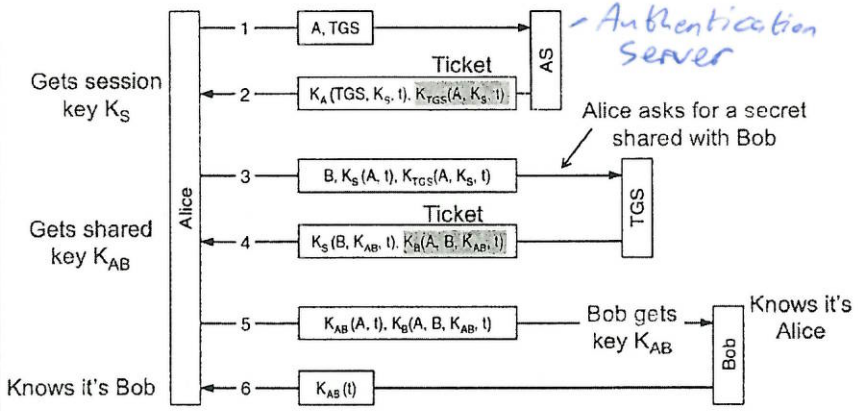
- Alice tells the KDC that she wants to talk to Bob
  - Sends a msg containing a random number R_A as a nonce
- The KDC sends back msg 2 containing R_A, a session key and a ticket that she can send to Bob



- Alice now sends the ticket to Bob along with a new random number R_{A2} encrypted with the session key K_S
- Bob sends back $K_S(R_{A2} - 1)$ to prove to Alice that she is talking to the real Bob

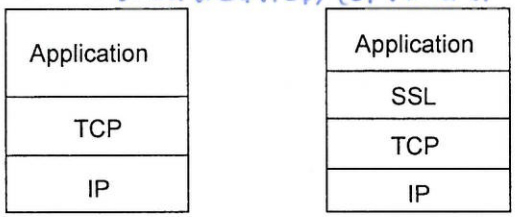
Kerberos

- Kerberos V5 is a widely used protocol (e.g. Windows)
  - Authentication includes a Ticket Granting Server (TGS)



Secure Sockets Layer (SSL)

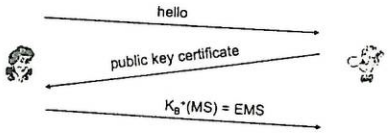
- Provides transport layer security to any TCP-based application using SSL services
  - e.g., Between Web browsers and servers for E-commerce
    - In practice Transport Layer Security (TLS) v1.2 should be used
- Security Services
  - Server authentication, data encryption, client authentication (optional)



normal application *application with SSL*

Toy SSL: A Simple Secure Channel

- Handshake
  - Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- Key Derivation
  - Alice and Bob use shared secret to derive set of keys
- Data Transfer
  - Data to be transferred is broken up into series of records
- Connection Closure
  - Special messages to securely close connection



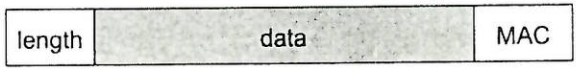
MS: master secret
EMS: encrypted master secret

Toy SSL: Key Derivation

- Considered bad to use same key for more than one cryptographic operation
  - Use different key for message authentication code (MAC) and encryption
- Four Keys
  - K_c = Encryption key for data sent from client to server
  - M_c = MAC key for data sent from client to server
  - K_s = Encryption key for data sent from server to client
  - M_s = MAC key for data sent from server to client
- Keys derived from Key Derivation Function (KDF)
  - Takes master secret and (possibly) some additional random data and creates the keys

Toy SSL: Data Records

- Why not encrypt data in constant stream as we write it to TCP?
  - Where would we put the MAC? If at end, no message integrity until all data processed
- Instead, break stream in series of records
  - Each record carries a MAC
- Receiver can act on each record as it arrives
  - Need to distinguish MAC from data
    - Want to use variable-length records

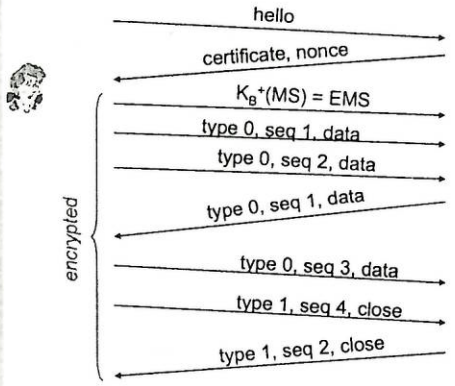


Toy SSL: Control Information

- Problem: Attacker can capture and replay, record or re-order records
- Sol: Put sequence number into MAC
  - MAC = MAC(M_x, sequence||data)
    - Note: no sequence number field
- Problem: Attacker could replay all records
- Sol: Use nonce

- Problem: Truncation Attack
  - Attacker forges TCP connection close segment
    - One or both sides thinks there is less data than there actually is
- Sol: Record Types, with one type for closure
  - Type 0 for data; Type 1 for closure
  - MAC = MAC(M_x, sequence||type||data)

Toy SSL: Summary & Outstanding Issues



- How long are fields?
- Which encryption protocols?
- Want negotiation?
  - Allow client and server to support different encryption algorithms
  - Allow client and server to choose together specific algorithm before data transfer

SSL Cipher Suite

- Cipher Suite
  - Public-key algorithm
  - Symmetric encryption algorithm
  - MAC algorithm
- SSL supports several cipher suites
- Negotiation
  - Client, server agree on cipher suite
  - Client offers choice
    - Server picks one

Common SSL symmetric ciphers
- DES – Data Encryption Standard: block
- 3DES – Triple strength: block
- RC2 – Rivest Cipher 2: block
- RC4 – Rivest Cipher 4: stream

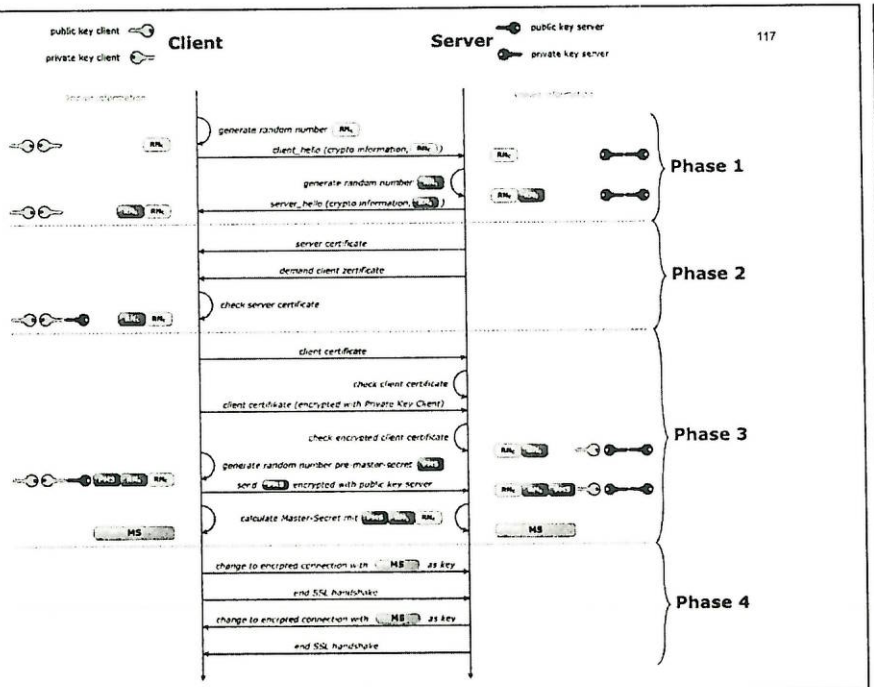
SSL Public key encryption
- RSA

Real SSL: Handshake

- Server Authentication
- Negotiation: Agree on crypto algorithms
- Establish Keys
- Client Authentication (optional)

Real SSL: Handshake II

- Client sends list of algorithms it supports, along with client nonce
- Server chooses algorithms from list; sends back: choice + certificate + server nonce
- Client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
- Client and server independently compute encryption and MAC keys from pre_master_secret and nonces
- Client sends a MAC of all the handshake messages
- Server sends a MAC of all the handshake messages
  - Q: Why do we this? Make sure no one has messed w/. sequence of events

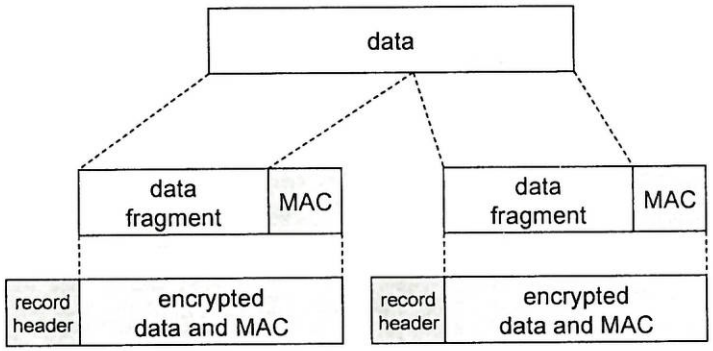


Real SSL: Handshake III

- Last 2 steps protect handshake from tampering
  - Client typically offers range of algorithms, some strong, some weak
- MITM could delete stronger algorithms from list
  - Last 2 steps prevent this
- Why two random nonces?
  - Suppose Trudy sniffs all messages between Alice & Bob
- Next day, Trudy sets up TCP connection with Bob (Amazon), sends exact same sequence of records
  - Bob thinks Alice made two separate orders for the same thing
- Sol: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days

-Trudy's message will fail integrity check

SSL Record Protocol



Record Header: content type; version; length

MAC: includes sequence number, MAC key M_x

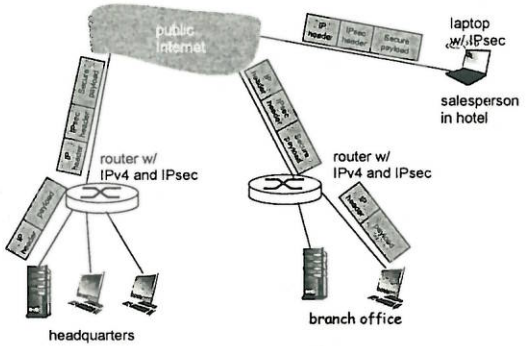
Fragment: each SSL fragment 2^{14} bytes (~16 Kbytes)

SSL Key Derivation

- Client nonce, server nonce, and pre-master secret input into pseudo random-number generator
  - Produces Master Secret (MS)
- Master Secret and nonces input into another random-number generator
  - To produce the "key block"
- Key block sliced and diced
  - Client MAC key
  - Server MAC key
  - Client encryption key
  - Server encryption key
  - Client initialization vector (IV)
  - Server initialization vector (IV)

Virtual Private Networks (VPNs)

- Institutions often want private networks for security
  - Costly! Separate routers, links, DNS infrastructure



- With a VPN, institution's inter-office traffic is sent over public Internet instead

-But inter-office traffic is encrypted before entering public internet

IPsec - Transport vs Tunnel Mode

- Transport Mode

  - IPsec datagram emitted and received by end-system

• Protects upper level protocols

- Tunnel Mode

  - End routers are IPsec aware

• Hosts need not be

IPsec – Network Layer Security

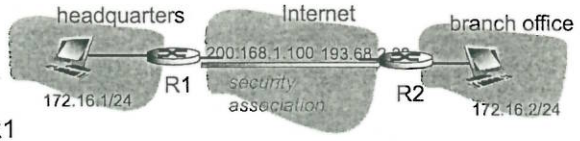
- Network-layer authentication
  - Destination host can authenticate source IP address
- Network-layer secrecy
  - Sending host encrypts the data in IP datagram
    - TCP and UDP segments; ICMP and SNMP messages
- Two principal protocols
  - Authentication header (AH) protocol
  - Encapsulation security payload (ESP) protocol
- Most common

-Tunnel mode w/. ESP

IPsec – Network Layer Security II

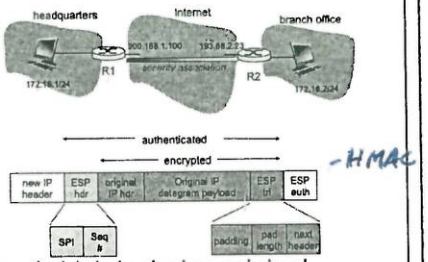
- Source and destination perform a handshake
  - Create network-layer logical channel called a security association (SA)

• Each SA unidirectional



- SA at R1
  - 32-bit identifier for SA: Security Parameter Index (SPI)
  - Origin interface of the SA (200.168.1.100)
  - Destination interface of the SA (193.68.2.23)
  - Type of encryption to be used (e.g. 3DES with CBC)
  - Encryption key
  - Type of integrity check (e.g. HMAC with with MD5)
  - Authentication key

ESP with Tunnel Mode



- Appends to back of original datagram (which includes original header fields!) an "ESP trailer" field
  - Encrypts result using algorithm & key specified by SA
- Appends to front of this encrypted quantity the "ESP header
- Creates authentication MAC over the four fields, using algorithm and key specified in SA
  - Appends MAC to the end forming payload
- Creates brand new IP header, with all the classic IPv4 header fields, which it appends before payload

DNS Security Extension (DNSSEC)

- Plain old DNS does not allow you to check the *authenticity* or *integrity* of a message
- MITM Attack
  - A resolver has no way to verify the authenticity and integrity of the data sent by name servers
- Packet Sniffing
  - DNS sends an entire query or response in a single unsigned and unencrypted UDP pkt
- Transaction ID Guessing
  - An attacker can respond with false answers to a predicted query
    - On the client there are 2^{32} possible combinations of ID (2^{16}) and UDP ports (2^{16})

DNSSEC

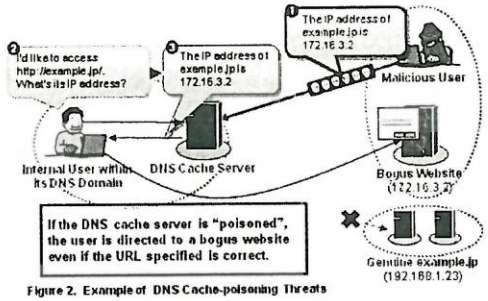


Figure 2. Example of DNS Cache-poisoning Threats

- DNSSEC provides origin authentication and integrity assurance services for DNS data
- DNSSEC has two perspectives
  - Domain owners sign their zone and publish the signed zone on their authoritative name servers
  - Querying hosts validate the digital signatures they receive in answers, along a chain of trust

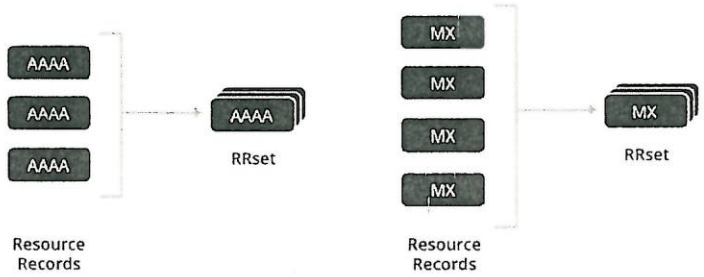
DNSSEC RRs

DNSSEC adds a number of new resource record types:

- RRSIG – Contains a cryptographic signature
- DNSKEY – Contains a public signing key
- DS – Contains a hash of a DNSKEY record
- Others ….

RRsets

- First step towards securing a zone with DNSSEC is to group all the records with the same type into a resource record set (RRset)
  - E.g., if you have three AAAA records in your zone, they would all be bundled into a single AAAA RRset



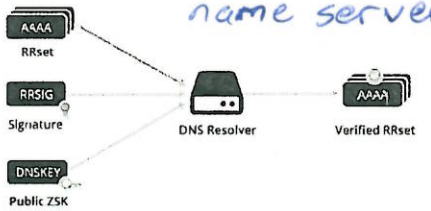
Zone-Signing Keys

public & private

- Each zone in DNSSEC has a zone-signing key pair (ZSK)
- A zone operator creates digital signatures for each RRset using the private ZSK
  - stores them in their name server as ~~RRS~~ RRSIG records

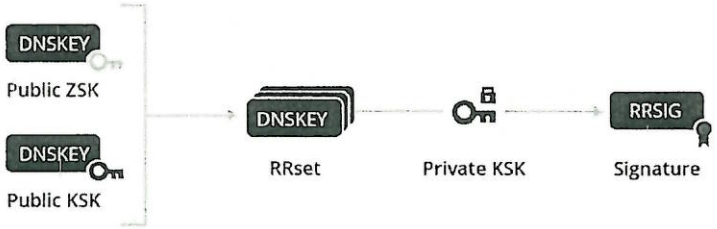
Zone-Signing Keys II

- Zone operators also needs to make their public ZSK available by adding it to their name server in a DNSKEY record
- When a DNSSEC resolver requests a particular record type (e.g., AAAA), the name server also returns the corresponding RRSIG
  - Resolver can then pull DNSkey record containing the public ZSK from the name server



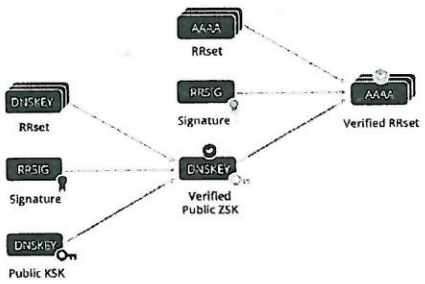
Key-Signing Keys

- The KSK validates the DNSKEY record
- It signs the public ZSK (which is stored in a DNSKEY record)
  - Creating an RRSIG for DNSkey



DNSSEC Validation

- Request the desired RRset
  - Returns the corresponding RRSIG record
- Request the DNSKEY records containing the public ZSK and public KSK
  - Returns the RRSIG for the DNSKEY RRset
- Verify the RRSIG of the requested RRset with the public ZSK
- Verify the RRSIG of the DNSKEY RRset with the public KSK



Delegation Signer Records

- DNSSEC introduces a delegation signer (DS) record to allow the transfer of trust from a parent zone to a child zone
- A zone operator hashes the DNSKEY record containing the public KSK
  - Gives it to the parent zone to publish as DS record
- Every time a resolver is referred to a child zone, the parent zone also provides a DS record
  - To check the validity of the child zone's public KSK
    - Resolver hashes it and compares it to the DS record from the parent