AINTEC
short course on:
SECURITY
in
Mobile/Wireless Networks

November 16-17, 2008

Gene Tsudik

Computer Science Department
University of California, Irvine
gts*AT*ics.uci.edu
http://www.ics.uci.edu/~gts
http://sconce.ics.uci.edu

1

OUTLINE

6-hour short course:

1. Introduction + Overview (45)
   - Preliminaries in Cryptography and Security
2. Last-Hop Wireless (45)
   - Security in Cellular and WLAN-type Networks
   - Location Privacy
3. Secure Wireless Device Association (90)
   - Human-assisted cryptographic protocols
4. RFID Security & Privacy (90)
5. Security in MANETs (90)

2

Expectations

- Learn about basics of security in wireless communications/networking
- Understand the state-of-the-art
- Much of the material has to do with cryptography and its applications
- Disclaimer: 6 hours is not enough!!!
- I might not cover your favorite topic, e.g., Bluetooth security

3

Helpful Background

- Basic Networking
  - TCP/IP, IP Multicast, 802.11,
- Network Security
  - Authentication, Key distribution, Protocols, Certification/Revocation, e.g., TLS/SSL, etc.
- Cryptography
  - Basic encryption concepts, symmetric vs public key, signatures, key management, hash chains & trees, etc.

4

Helpful Background & Materials

- Computer Networking, 4/e (Chapter 8, in particular)
  James Kurose, Keith Ross
  ISBN: 0-321-49770-8
- Cryptography and Network Security: Principles and Practice, 3/E (textbook)
  William Stallings
  ISBN: 0-130-91429-0
- Network Security: Private Communication in a Public World
  Charlie Kaufman, Radia Perlman, Mike Speciner
  ISBN: 0-130-46019-2
- Handbook of Applied Cryptography (good reference book, and free!)
  Alfred Menezes, Paul van Oorschot and Scott Vanstone
  ISBN: 0-8493-8523-7
  http://www.cacr.math.uwaterloo.ca/hac/

5

Some statements to start with:

- Wireless-ness does not prompt new fundamental security problems (except one)
  - Most advances in wireless security are not specific to wireless communication
- Mobility raises new security problems
- Ad hoc deployment/operation raises them too

- Most sensors don't network
- Most sensor networks are static
- Most so-called "mobility protocols" break when nodes move very fast or change directions often

6

Security & Cryptography Introduction

7

Symmetric Cryptography

Also known as: conventional, shared-key or single-key

- 2 parties (sender/recipient or Alice/Bob) share a common key
- this key is used to encrypt and/or authenticate their communication
- all “classical” encryption algorithms are symmetric
- the only encryption type prior to invention of public-key in 1970’s

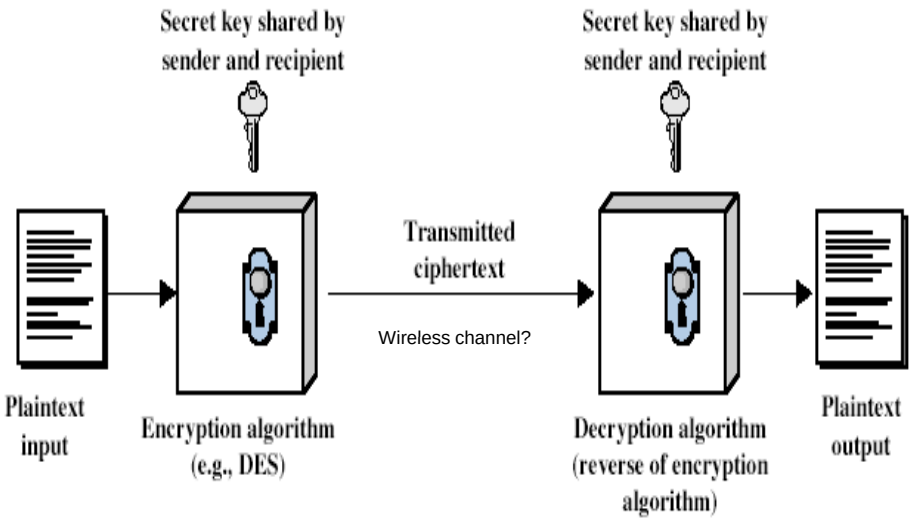
8

Basic Terminology

- **plaintext** - the original message
- **ciphertext** - the encrypted message
- **cipher** - algorithm for transforming plaintext to ciphertext
- **key** – secret info used in cipher, known only to appropriate parties (e.g., sender/receiver)
- **encipher (encrypt)** - convert plaintext to ciphertext
- **decipher (decrypt)** - recover ciphertext from plaintext
- **cryptography** - study of encryption principles/methods
- **cryptanalysis (codebreaking)** - the study of principles/ methods of deciphering ciphertext *without* knowing keys
- **cryptology** - cryptography + cryptanalysis

9

Symmetric Cipher Model



10

Open vs. closed cipher design

• **Open design**: algorithm, protocol, system design (and even possible plaintext) may be public information. The only secret is/are the key(s)

• **Closed design**: as much information as possible (including the algorithm) is kept secret

11

Encryption Principles

- A cipher (cryptosystem) has at least five components:
  - Plaintext
  - Secret Key(s)
  - Ciphertext
  - Encryption algorithm
  - Decryption algorithm
- Security usually depends on the secrecy of the key, not the secrecy of the algorithm

12

Requirements

- two requirements for secure use of symmetric encryption:
  - a strong encryption algorithm
  - a secret key known only to sender/receiver

    $Y = E_K(X)$

    $X = D_K(Y)$
- assume encryption algorithm is known to everyone (including the adversary)
- need secure channel to distribute keys!

13

Cryptography

Number of keys used:

- One:
  - Shared key, conventional, symmetric
  - Examples: 3-DES, AES
- Two:
  - Asymmetric, public key
  - Examples: RSA, ElGamal

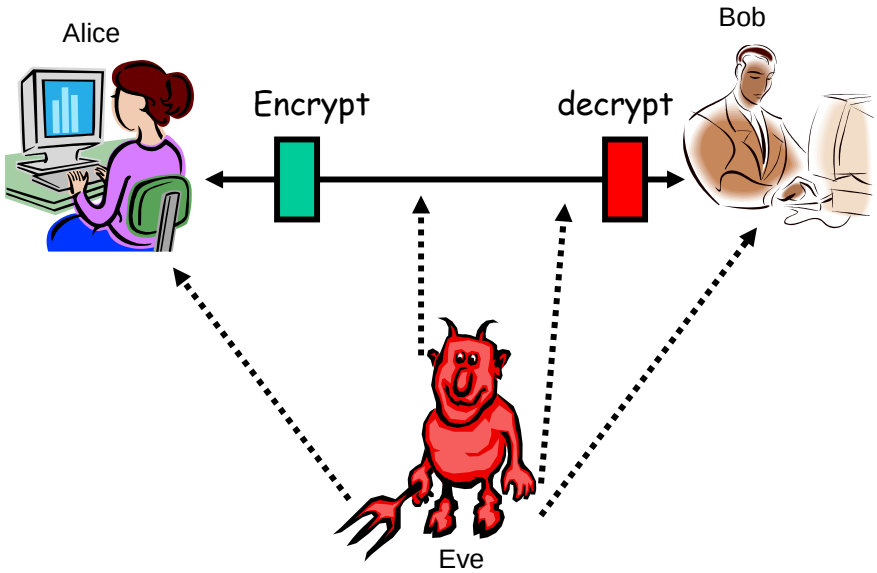
14

Adversary's Goal

- Attack cryptosystem to
  - obtain/read plaintext (violate confidentiality)
  - violate authenticity and/or integrity
- This usually (but not always) requires obtaining the secret/private KEY(s)

15

Alice, Bob and Eve (Adversary)



16

Types of Cryptanalytic Attacks:

- **ciphertext only**
  - only knows algorithm and lots of ciphertext but not the matching plaintext
- **known plaintext**
  - knows a number of (n) plaintext/ciphertext pairs
- **chosen plaintext**
  - selects n plaintexts and obtains corresponding ciphertexts
- **chosen ciphertext**
  - selects n ciphertexts and obtains corresponding plaintexts

17

Types of Cryptanalytic Attacks: most dangerous/sophisticated attacks

- **adaptive chosen plaintext**
  - selects n plaintexts and obtains corresponding ciphertexts
  - repeat above a number of times
- **adaptive chosen ciphertext**
  - selects n ciphertexts and obtains corresponding plaintexts
  - repeat above a number of times

18

Block Ciphers: Common Modes of Operation

- **Electronic code-book (ECB)**
  $C_i = E(K, P_i)$
  → Local error, permutation attack, parallel encr.
- **Chained block cipher (CBC)**
  $C_i = E(K, C_{i-1} \text{ XOR } P_i)$
  → Need IV, error causes 2-block loss, no parallel encr.
- **Output feedback (OFB)**
  $V_i = E(K, V_{i-1})$ $C_i = P_i \text{ XOR } V_i$
  →Stream cipher, local error, pre-computation
- **Cipher feedback (CFB)**
  $C_i = P_i \text{ XOR } E(K, C_{i-1})$
  OFB/CFB - encrypt only!
  → Plaintext dependence, avalanche effect, parallel decryption.
- **Counter Mode (CM)**
  same as OFB, but $V_i = E(K, i)$

19

More Definitions

- **unconditional security**
  - no matter how much computer power is available, the cipher cannot be broken, since ciphertext provides insufficient information to uniquely determine the corresponding plaintext
- **computational security**
  - given state-of-the-art computing resources, the cipher cannot be broken (today)
- **ad hoc security**
  - the cipher is claimed secure; typical claim in ciphers designed by amateurs or in “snake oil” products

20

Message & Origin Authentication

21

Message & Origin Authentication

- Goal: protection against active attacks
  - Impersonation
  - Modification of contents (integrity)
  - Replay
  - Interruption and denial of service
- Requirements
  - Message data is authentic: has integrity or has not been altered
  - Message source is authentic
  - Optional
    - Message arrived in correct sequence
    - Non-repudiation

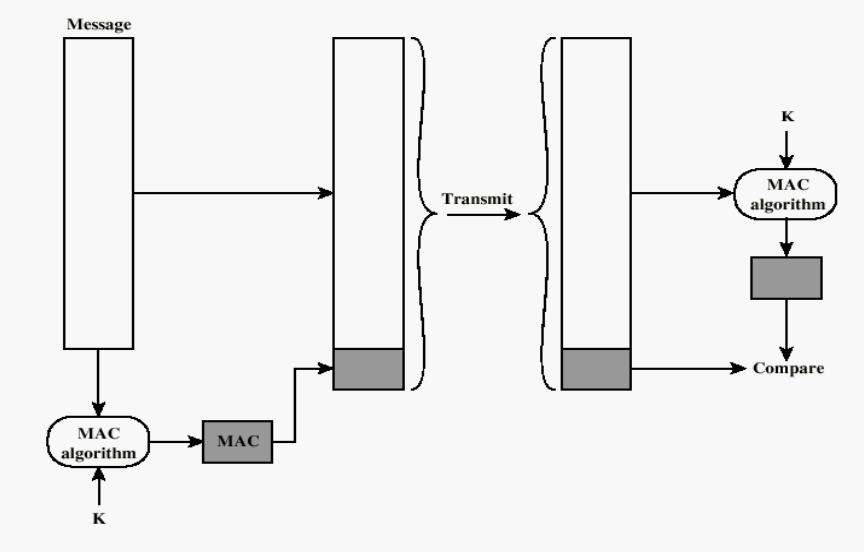
22

Message Authentication Approaches

- Conventional encryption
  - Assumes that only the correct parties should have access to key
  - Does that guarantee integrity?
- Message authentication without encryption
  - Authentication tag is attached to message to verify/assert its integrity and the integrity of the source
- Message Authentication Code (MAC)
  - MAC=F(Message,Key)

23

Message Authentication Code



24

MAC Properties

- Message is authentic
  - If the attacker modified the message, the MAC will most likely not match the one calculated by the receiver
- Source is authentic
  - No one else has the key to generate the same MAC
- Message is in sequence and/or timely
  - Should add timestamp or other nonce to the message before calculating the MAC
- Any encryption algorithm can be used to generate MAC

25

Cryptographic HASH Functions

- Purpose: produce a fingerprint or digest of input data
- Properties of a “good” HASH function H():

1. H() takes on input of any size
2. H() produces fixed-length output
3. H(x) is easy to compute (efficient)
4. Given any h, it is computationally infeasible to find x such that: H(x) = h
5. For any x, it is computationally infeasible to find y, such that: H(y) = H(x) and y<>x
6. It is computationally infeasible to find any (x, y) such that H(x) = H(y) and x<>y

26

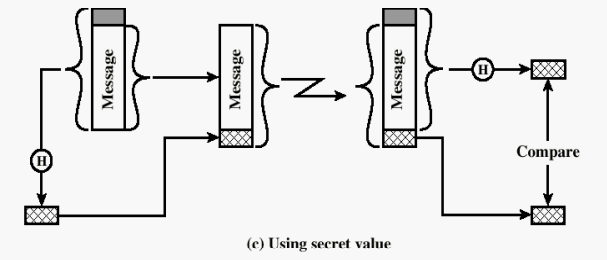
HASH Functions properties restated:

- Cryptographic properties of a “good” HASH function:
  - One-way-ness (#4)
  - Weak Collision-Resistance (#5)
  - Strong Collision-Resistance (#6)
- Non-cryptographic properties of a “good” HASH function
  - Fixed output (#1)
  - Arbitrary-length input (#2)
  - Efficiency (#3)

27

Message Authentication with a Hash Function

1. Using a symmetric secret / key



2. Using symmetric encryption
   - Generate H(M), which is small in size
   - Use $E_K(H(M))$ as the MAC

28

Well-known HASH Algorithms

	SHA-2	MD5	RIPEMD
Digest length	256,384,512	128	160
Block Size	512	512	512
# steps	80	64	160
Max message	$2^{64}-1$	unlimited	unlimited

29

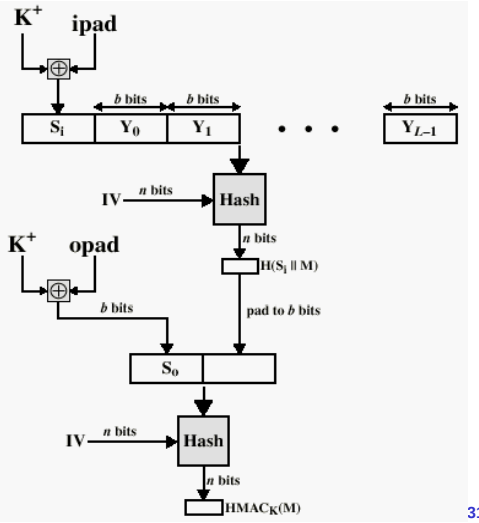
Hash Function MAC (HMAC)

- **HMAC**: Use a MAC derived from any cryptographic hash function
  - Note that a hash function does not use a key, and, therefore, cannot be directly used as a MAC
- **Motivations for HMAC:**
  - Cryptographic hash functions execute faster in software than encryption algorithms such as DES
  - No need for the reverseability of encryption
  - No export restrictions from the US
- **Status**: designated as mandatory for IPSec
  - Also used in Transport Layer Security (TLS)

30

HMAC Algorithm

- Compute H1= H of the concatenation of M and K1
- To prevent an “additional block” attack, compute again H2= H of the concatenation of H1 and K2
- K1 and K2 each use half the bits of K
- Notation:
  - K^+ =K padded with 0’s
  - ipad=00110110 x b/8
  - opad=01011100 x b/8
- Execution:
  - Same as H(M), plus 2 blocks



31

Public Key Crypto

32

Public-Key Cryptography

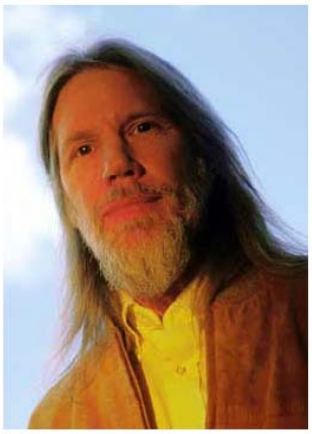
- Each user has a unique public-private key-pair:
  Alice has K_{Apriv}, K_{Apub}
  Bob has K_{Bpriv}, K_{Bpub}
- The public key can be given to anyone
- The private key is not shared with anyone, including a trusted third party (authentication server)
- The public key is a one-way function of the private-key (hard to compute private key from public one)
- Used for key distribution/agreement, message encryption, and digital signatures

33

Origins of Public Key

- Concept credited to Diffie & Hellman, 1976 “New Directions in Cryptography”
  - Not the first public key method (Merkle’s puzzles is the first)
- Motivation – allow Alice to send a message to Bob without a pre-shared secret or a mutually-trusted Third Party;
  - called “public-key” because Alice & Bob need only exchange public keys to set up a secret channel
- Invented earlier by the British at CESG: http://www.cesg.gov.uk/about/nsecret.htm

34



Whitfield Diffie

Martin Hellman

35

Public-Key Agreement

- Alice and Bob want to agree on a secret key to use with a symmetric encryption algorithm, e.g., 3-DES or AES
  - Need a shared secret
- Do this after exchanging only public keys
- Each computes a secret session key K derived from their own private key and the other's public key. Both compute the same K independently

36

Diffie-Hellman Method

1) Shared prime p and generator g in (Z_p)

Alice: private x_a and public $y_a = g^{Xa} \bmod p$
Bob: private x_b and public $y_b = g^{Xb} \bmod p$

$x_a = \log_g y_a \bmod p$ (hard to compute)

2) They exchange public keys

Alice computes: $K = y_b^{Xa} \bmod p = g^{Xb\ Xa} \bmod p$
Bob computes: $K = y_a^{Xb} \bmod p = g^{Xa\ Xb} \bmod p$

What can K be used for?

37

Hybrid Approach

- Alice & Bob generate x_a, x_b and deposit/publish their public keys y_a, y_b

- Alice gets y_b from database (or from Bob)
- Alice generates temporary pair x_t, y_t
- Alice computes $K = y_b^{Xt} \bmod p = g^{Xb\ Xt} \bmod p$
- Alice → Bob: y_t, $E_K(M)$

- Bob computes $K = y_t^{Xb} \bmod p = g^{Xt\ Xb} \bmod p$ and decrypts M

38

Public-Key Encryption

- Public key → to encrypt messages
  Private key → to decrypt

- Alice encrypts message to Bob with Bob's public key
- Bob decrypts incoming messages with his private key
- In practice, public-key encryption is used to encrypt/decrypt symmetric keys (e.g., AES), and the latter are then used to encrypt/decrypt bulk data

39

Sending Messages

To send message M to Bob, only Bob's keys used
Alice → Bob: $C = E_{Bpub}(M)$
Bob decrypts: $M = D_{Bpriv}(C)$

In practice, use to distribute symmetric key K
Alice → Bob: $C_K = E_{Bpub}(K)$, $C_M = E_K(M)$
Bob decrypts: $K = D_{Bpriv}(C_K)$, $M = D_K(C_M)$

Alice and Bob then use K to encrypt/decrypt messages

E.g., that's how PGP/GPG and SSL work…

40

RSA

Ron Rivest, Adi Shamir, Leonard Adleman

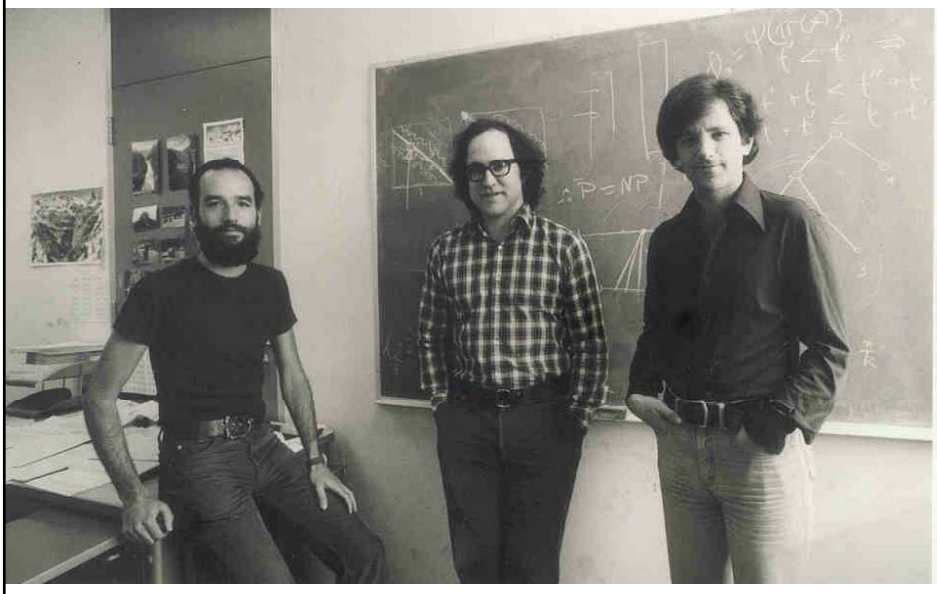
1977 -- all at MIT at the time

Basic idea: a modular exponentiation-based cipher where the modulus is the product of two large primes

Security derived from *conjectured* difficulty of factoring large composite integer into 2 or more*, (also large) prime factors

41

S R A



42

RSA

Pick two large (about 512-bit and up) primes p and q and compute n = p * q

Pick e, d such that:

$$e * d = 1 \bmod \phi(n)$$

where: $\phi(n) = (p-1) * (q-1)$

(e, n) is the public key

(d, [p,q]) is the private key

Encrypt: $C = M^e \bmod n$

Decrypt: $M = C^d \bmod n$

43

Example

p = 53, q = 61, n = 53 * 61 = 3233

Pick e = 71

Compute d such that

71 * d = 1 mod (52 * 60)

get d = 791

Let M = 1704

Encrypt: $C = 1704^{71} \bmod 3233 = 3106$

Decrypt: $M = 3106^{791} \bmod 3233 = 1704$

44

Theory

Why $\phi(n) = (p-1) * (q-1)$

$\phi(n)$ = # primes < n relatively prime to n

Consider the n=pq numbers 0, 1, ..., pq-1

All are relatively prime to n except for 0 and

p-1 elements: q, 2q, 3q, ..., (p-1)q

q-1 elements: p, 2p, 3p, ..., (q-1)p

so $\phi(n) = pq - [(p-1) + (q-1) + 1]$

$= pq - p - q + 1 = (p-1)*(q-1)$

45

Factoring

Given a number n, find primes p1, p2, ..., pk such that n = p1 * p2 * ... * pk

For RSA, there are known to be only 2 factors:

n = p * q

Factoring arbitrary numbers is harder than factoring special types of numbers, e.g., numbers of the form $n = 2^s - 1$ (Mersenne primes)

Strength of RSA – relation to factoring

1) If factoring easy ➔ breaking RSA is easy

find plaintext?

2) If breaking RSA is easy ➔factoring made easy ?

Breaking RSA can be no harder than factoring, but could be easier

46

Digital Signatures: Objectives

- Message integrity and authenticity
  - Goal: detect tampering
- Source/sender authenticity
  - Goal: detect forgeries
- Non-repudiation
  - Goal: sender cannot deny having signed a message
    - trusted 3rd party (court?) can resolve disputes

47

Public-Key Signatures

- Signer has a public-private key pair
- Signer generates a signature with the private key
  - Only the real signer can do this
- A signature is verified with the public key
  - Anyone can do this, including the intended recipient and a trusted 3rd third party, even adversary!
- No keys of the receiver/verifier are used

48

Sending a Signed Message

- Alice sends a signed message to Bob using her private key. Bob validates with her public key

  1) Alice → Bob: (M, S) where
     $S = sign_{Apriv}(h(M))$
     and h() is a "good" hash function

  2) Bob checks: $validate_{Apub}(M, S)$

- Hash function h is public and not keyed, but:
  - h() is hard to invert
  - Practical examples: MD5, SHA

- S is function of entire message M

49

RSA Signatures

- Let (e, n) be Alice's public key and (d, n) her private key

- Alice → Bob: (M, S) where
  $S = sign_{Apriv}(h(M)) = [\ h(M)\ ]^{d} \bmod n$

- Bob checks: $validate_{Apub}(M, S)$:
  1. h1 = h(M)
  2. $h2 = S^{e} \bmod n$
     Note: $S^{e} = [\ h(M)\ ]^{d*e} \bmod n = h(M)$
  3. if h1 = h2 then accept, else reject

50

Digital Signature Standard (DSS)

- US FIPS PUB 186, adopted 1994
- Uses variant of methods invented by ElGamal and Schnorr, which, in turn, are loosely based on Diffie-Hellman
- Uses exponentiations in modular arithmetic where security is based on difficulty of computing the discrete log (as for DH)
- Uses SHA for hashing

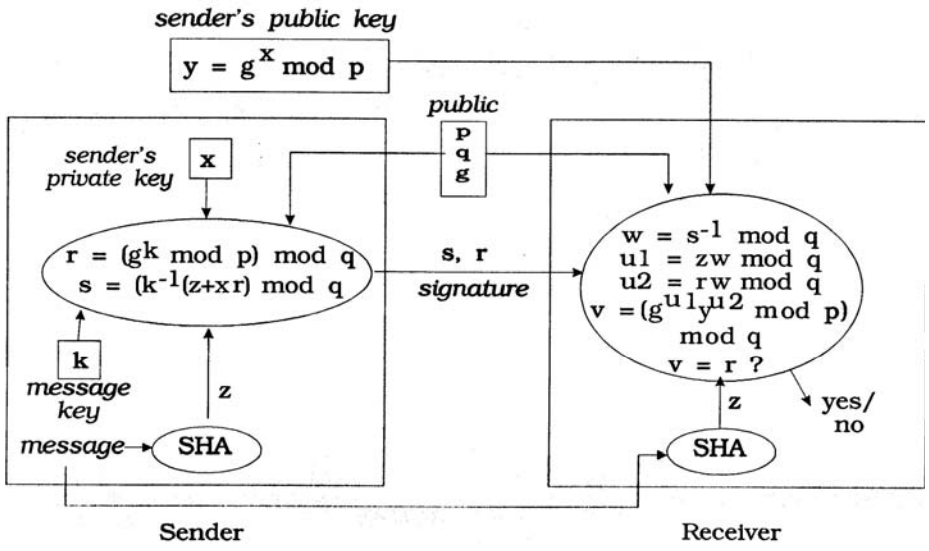
51

DSS

- Global public values (shared by group which can be as large as needed)
  - p - prime number (512-1024 bits)
  - q - 160-bit value (most computation mod q)
  - $g = h^{(p-1)/q} \bmod p$ where $h < (p-1)$ and $g > 1$
- User's private key
  - x - any number less than q
- User's public key
  - $y = g^x \bmod p$

52

Digital Signature Standard (DSS)



53

Notes

- Signing (can be) faster than in RSA
- Verification is slower than with RSA
- Signature size: 320 (DSS) vs 1024 (RSA) bits
- Both RSA and DSS are used extensively -- many products/systems support both
- DSS not designed for encryption – use together with Diffie-Hellman key exchange or El Gamal PKCS

54

End of introduction

Questions?

55