A FOSAD 2005 short course on: SECURITY in (certain) Wireless Communication

Professor Gene Tsudik
Associate Dean for Research & Graduate Studies
Bren School of Information & Computer Science
University of California, Irvine

gts (at) ics.uci.edu
www.ics.uci.edu/~gts

OUTLINE of this course (0)

4-hour 2-day short course:

1. Overview, 802.11, etc.
2. Wireless Device Pairing
3. RFID security
4. WSN security
5. MANET security ?

Why this particular order?

Course Material Attributions:

- Adrian Perrig
- Yongdae Kim
- Jeong Yi
- Ari Juels
- David Wagner
- Claude Castelluccia
- Srdjan Capcun
- etc.

3

My Background in Brief

EDUCATION
- PhD in Computer Science, University of Southern California, March 1991.
- MS in Computer Science, University of Southern California, May 1987.
- BS in Computer Science, University of Houston, May 1985.

PROFESSIONAL EXPERIENCE
- **07/02 - today** Associate Dean of Research and Graduate Studies, School of ICS, UC Irvine
- **07/03 - today** Full Professor, Computer Science Department, UC Irvine
- **01/00 - 06/03** Associate Professor, Information and Computer Science, UC Irvine.
- **08/98 - 12/99** Research Associate Professor, Computer Science Department, USC.
- **04/96 - 12/99** Project Leader, USC Information Sciences Institute.
- **01/95 - 03/96** Project Leader, IBM Research Laboratory, Zurich.
- **04/91 - 01/95** Research Staff Member, IBM Research Laboratory, Zurich.
- **08/87 - 04/91** Research Assistant, USC Computer Networks Laboratory.
- **06/85 - 05/90** Member of Technical Staff, IBM Scientific Center, Los Angeles.

Current/Recent Areas of Research:
- Group signatures, secret handshakes, private authentication
- Group key management, secure group membership, etc.
- Privacy/Integrity for Outsourced Databases
- Networks: reliable broadcast in MANETs
- RFID privacy
- Human-assisted security

Some Past "Achievements":
- Visa Scheme → pre-cursor to firewalls
- Inter-Domain Policy Routing
- IBM KryptoKnight → NetSP
- iKP → SET
- CDPD/GSM security
- CLIQUES → Secure SPREAD
- SEM Architecture → Mediated Security Services, id-based crypto
- Admission in MANETs and P2Ps → Bouncer toolkit

4

Some of my relevant work

Secure Mobility:

- R. Molva and G. Tsudik, Authentication Method with Impersonal Token Cards, 1993 IEEE Symposium on Security and Privacy.
- A. Herzberg, H. Krawczyk and G. Tsudik, On Traveling Incognito, 1994 IEEE Mobile Computing Systems and Applications.
- R. Molva, D. Samfat and G. Tsudik, Authentication of Mobile Users, IEEE Network, Vol. 8, No. 2, pp. 26-34, March-April 1994.
- G. Ateniese, A. Herzberg, H. Krawczyk and G. Tsudik, Untraceable Mobility: How to Travel Incognito, Computer Networks, Vol. 31, No. 8, pp. 871-884, April 1999.

Reliable Broadcast/Multicast in MANETs:

- K. Obraczka, G. Tsudik and K. Viswanath, Flooding for Reliable Multicast in Multi-Hop Ad Hoc Networks, ACM Wireless Networks, Vol. 7, No. 6, December 2001.
- K. Obraczka, G. Tsudik and K. Viswanath, Exploring Mesh- and Tree Based Multicast Routing Protocols for MANETs, IEEE Transactions on Mobile Computing, to appear in 2005.
- K. Obraczka, G. Tsudik and K. Viswanath, Pushing the Limit of Multicast in Ad Hoc Networks, IEEE ICDCS'2001.

WSN Security:

- C. Castelluccia, E. Mykletun and G. Tsudik, Efficient Aggregation of Encrypted Data in Wireless Sensor Networks, IEEE Mobiquitous'05, July 2005.

5

Some of my relevant work

Admission Control in MANETs and P2Ps

- M. Narasimha, G. Tsudik and J. Yi, On the Utility of Distributed Cryptography in P2P Settings and MANETs, IEEE ICNP'03, November 2003.
- N. Saxena, G. Tsudik and J. Yi, Admission Control in Peer-to-Peer: Design and Performance Evaluation}, ACM SASN '03, November 2003.
- N. Saxena, G. Tsudik and J. Yi, Experimenting with Peer Group Admission Control, International Workshop on Advanced Developments in Software and Systems Security (WADIS'03), December 2003.
- N. Saxena, G. Tsudik and J. Yi, Identity-based Access Control for Ad Hoc Groups, ICISC'04, December 2004.
- N. Saxena, G. Tsudik and J. Yi, Efficient Node Admission for Short-lived Mobile Ad Hoc Networks, IEEE ICNP'05, November 2005.

MANET/Vehicular Security

- J. Kim and G. Tsudik, Securing Route Discovery in DSR, IEEE Mobiquitous'05, July 2005.
- M. El Zarki, S. Mehrotra, G. Tsudik and N. Venkatasubramanian, Security Issues in a Future Vehicular Network, EuroWireless 2002.

6

Expectations

- Learn about security in wireless communication:
  - Last-hop wireless (e.g., 802.11)
  - MANETs
  - WSNs
  - RFIDs and the like
- Understand the state-of-the-art
- Much of the material has to do with cryptography and its applications
- Disclaimer: 4 hours is not enough!!!
- I might not cover your favorite topic, e.g., Bluetooth security

7

Helpful Background

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

8

Some heretical statements to start with

- Wireless-ness does not cause brand new security problems
  - Most advances in wireless security aren't specific to wireless
- Mobility does!
- Ad-hoc-ness does!
- Most sensors don't network

9

Cryptography primer

10

Symmetric Cryptography

- also known as: conventional, shared-key or single-key
- 2 parties (sender/recipient or Alice/Bob) share a common key
- key used to encrypt and/or authenticate some (or all) of their communication
- all “classical” encryption algorithms are symmetric
- the only encryption type prior to invention of public-key in 1970’s

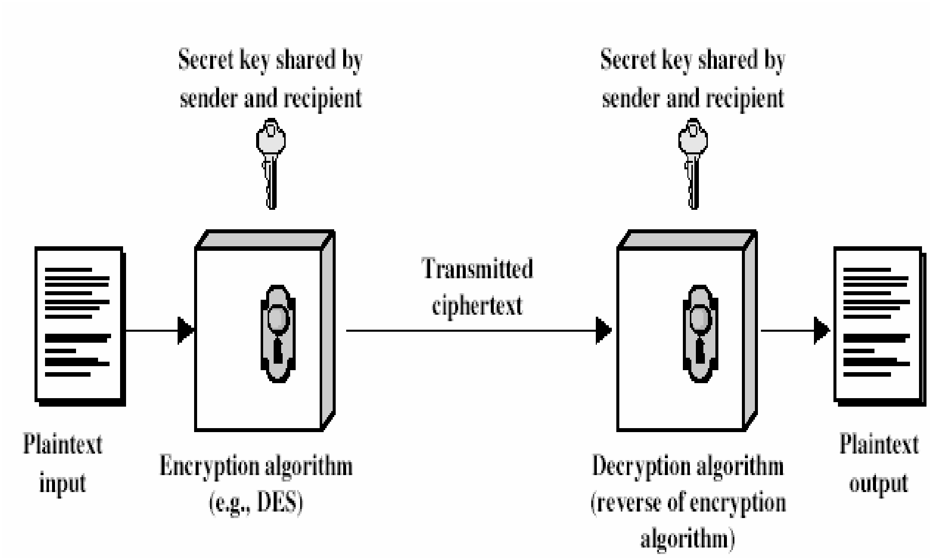
11

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 key
- **cryptology** - cryptography + cryptanalysis

12

Symmetric Cipher Model



13

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

14

Encryption Principles

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

15

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!

16

Cryptography

- can characterize by:
  - type of encryption operations used
    - substitution / transposition / product
  - number of keys used
    - single-key or private / two-key or public
  - way in which plaintext is processed
    - block / stream

17

Adversary's Goal

- Attack cipher/cryptosystem to
  - obtain/read ALL plaintext
  - forge authenticity checks (inject data)
- This usually requires obtaining the KEY(s)

18

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

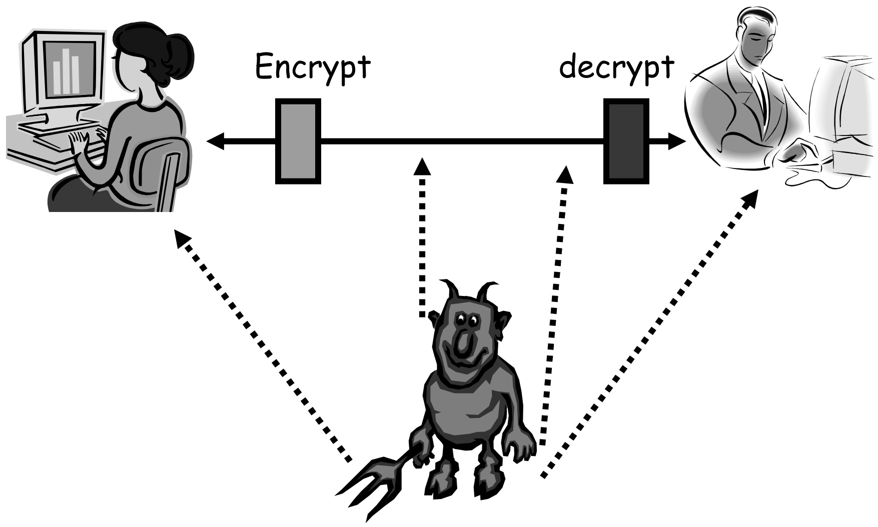
19

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

20

Alice, Bob and Eve



21

More Definitions

- **unconditional security**
  - no matter how much computer power is available, the cipher cannot be broken since the ciphertext provides insufficient information to uniquely determine the corresponding plaintext
- **computational security**
  - given limited computing resources (e.g., time needed for calculations is greater than age of universe), the cipher cannot be broken
- **ad hoc security**
  - the cipher is claimed secure. Often encountered in “snake oil” products

22

Message Authentication

Message Authentication

- Goal: offer protection against active attacks
  - Impersonation
  - Modification of contents
  - Replay
  - Interruption and denial of service

- Requirements
  - Message is authentic - has not been altered
  - Message source is authentic
  - Optional
    - Message arrived in correct sequence
    - Non-repudiation

Message Authentication Approaches

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

25

Message Authentication Code

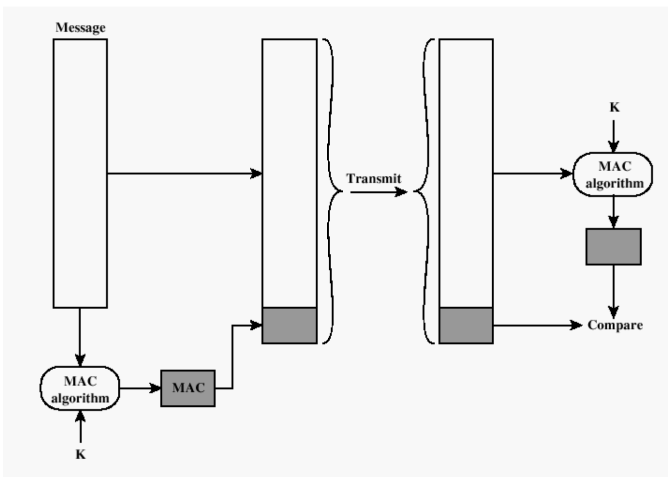


Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)

26

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
  - Hence, also non-repudiation
- Message is in sequence
  - Should add timestamp or other nonce to the message before calculating the MAC

- Any encryption algorithm can be used to generate MAC

27

Cryptographic HASH Functions

Observation: don't need "decryption" for MAC

- 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

28

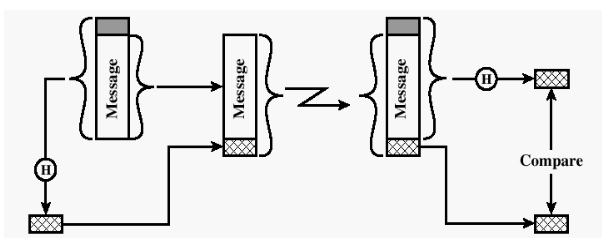
HASH Functions properties restated:

- 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)

29

Message Authentication with a Hash Function

1. Using a symmetric secret / key



(c) Using secret value

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

30

Well-known HASH Algorithms

	SHA-1	MD5	RIPEMD
Digest length	160 bits	128 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	80 (4 rounds of 20)	64 (4 rounds of 16)	160 (5 paired rounds of 16)
Maximum message size	2^{64}-1 bits	unlimited	unlimited

31

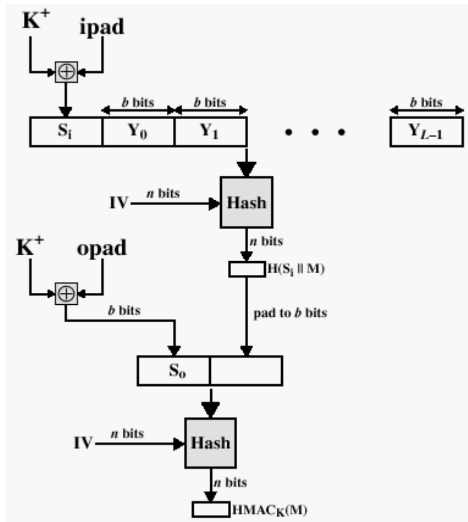
Hash Function MAC (HMAC)

- **HMAC Idea**: Use a MAC derived from any cryptographic hash function
  - Note that hash functions do not use a key, and therefore cannot serve directly 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 IP security
  - Also used in Transport Layer Security (TLS), which will replace SSL, and in SET

32

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



33

Public Key Crypto

34

Public-Key Cryptography

- Each user has a unique public-private key pair

  Alice - K_{Apriv}, K_{Apub}
  Bob - 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

35

Origins of Public Key

- Concept credited to Diffie and Hellman, 1976 “New Directions in Cryptography”
- Motivation - wanted a scheme whereby Alice could send a message to Bob without the need for Alice and Bob to share a secret or for a Trusted Third Party -- called “public-key” because Alice & Bob need only exchange public keys to set up a secret channel
- Invented earlier by British at CESG http://www.cesg.gov.uk/about/nsecret.htm

36

Whitfield Diffie

Martin Hellman

37

Public-Key Agreement

- Method whereby Alice and Bob can agree on a secret key to use with DES, AES, or some other symmetric encryption algorithm
  - Need a shared secret
- They do this after exchanging only public keys
- They each compute a secret session key K derived from their own private key and the other's public key. They both arrive at the same K independently

38

Diffie-Hellman Method

1) Shared prime p and generator g

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

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

2) They exchange public keys

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

What can K be used for?

39

Security/Strength

- Depends on the difficulty of computing the discrete logarithm
- Best-known methods are exponentially hard
- Need to use numbers on the order of 768 bits (230 digits) or bigger
- Implementations typically use 512 (155), 1024 (310) or 2048 (621) bits (digits).

40

On-The-Fly Approach

- Alice and Bob generate x_a, x_b, y_a, y_b on-the-fly
- They exchange y_a and y_b and compute K
- Drawbacks?
- What applications are appropriate?

41

Permanent

- Alice and Bob generate permanent keys and deposit y_a and y_b in public database (key center)
- Alice initially gets y_b from public database (or from Bob)
- Alice computes $K = y_b^{\ Xa} \bmod p = g^{Xb\ Xa} \bmod p$
- Alice -> Bob: y_a, $C = E_K(M)$
  (or Bob could get y_a from database)
- Bob computes $K = y_a^{\ xb} \bmod p = g^{Xa\ Xb} \bmod p$
- Bob decrypts C with K to get M
- Drawbacks?

42

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

43

Public-Key Encryption

- The public and private keys are used for message encryption and decryption for purpose of secrecy
- 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 and decrypt messages that contain symmetric keys (e.g., for DES/AES), and the symmetric keys are used to encrypt/decrypt bulk data

44

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…

45

RSA

Ron Rivest, Adi Shamir, Leonard Adleman

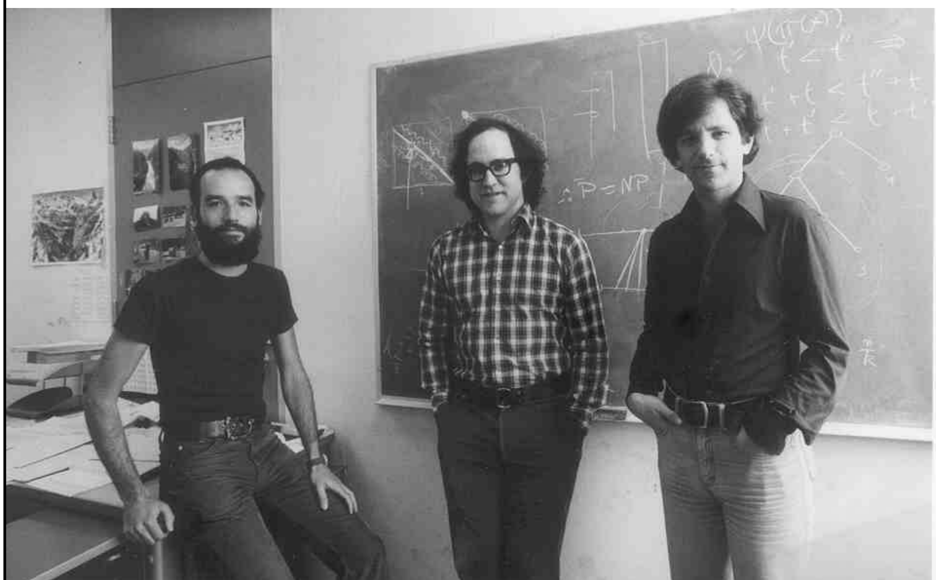
1977 -- all at MIT at the time

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

Mathematical strength is derived from the "conjectured" difficulty of factoring a large composite number into its 2 (also large) prime factors

46

S R A



47

RSA

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

Pick e, d such that:

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

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

(e, n) is the public key

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

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

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

48

Example

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

Pick $e = 71$

Compute d such that
  $71 * d = 1 \bmod (52 * 60)$
  get $d = 791$

Let $M = 1704$

Encrypt: $C = 1704^{71} \bmod 3233 = 3106$
Decrypt: $M = 3106^{791} \bmod 3233 = 1704$

49

Theory

Proof sketch for $\phi(n) = (p-1) * (q-1)$

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

50

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

51

Digital Signatures: Objectives

- Message integrity and authenticity
  detect tampering and bogus messages
- Source/sender authenticity
  detect forgeries
- Non-repudiation
  sender cannot repudiate signing a message
  trusted 3rd party (court?) can resolve disputes

52

Public-Key Signatures

- Signer has a public-private key pair
- A signature is produced 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 3^{rd} third party
- No keys of the receiver/verifier are used

53

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

54

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

55

Digital Signature Standard (DSS)

- FIPS PUB 186, adopted 1994
- Uses variant of methods invented by ElGamal and Schnorr, which in turn were 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

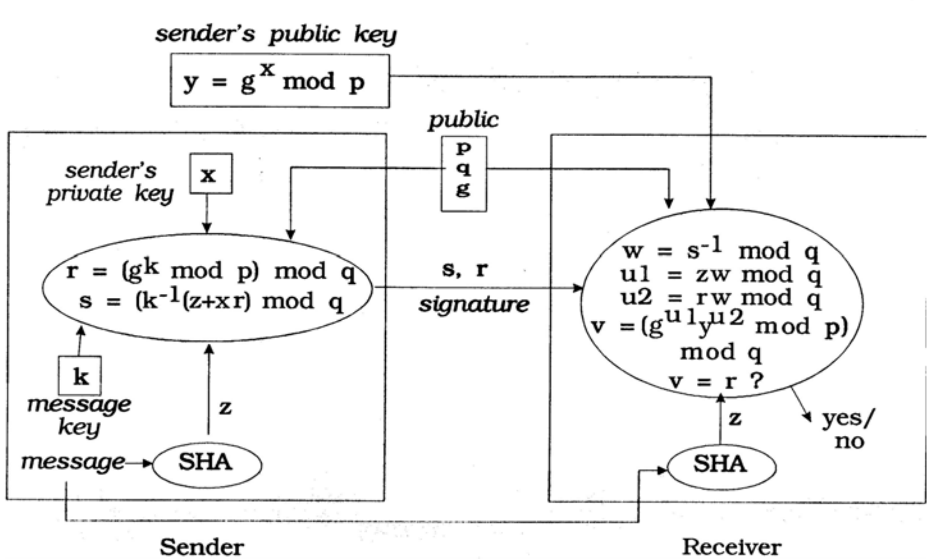
56

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
  - $y = g^x \bmod p$

57

Digital Signature Standard (DSS)



58

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 support both
- DSS not designed for encryption – use together with Diffie-Hellman key exchange or El Gamal PKCS

59

Other "tools"

We might need other crypto techniques in the course; to be covered later…

- One-time signatures, hash chains & trees
- Id-based crypto
- Threshold (and maybe proactive) crypto
- Group signatures (maybe)

60