

# Foundations of Network and Computer Security

John Black

Lecture #7  
Sep 14<sup>th</sup> 2004

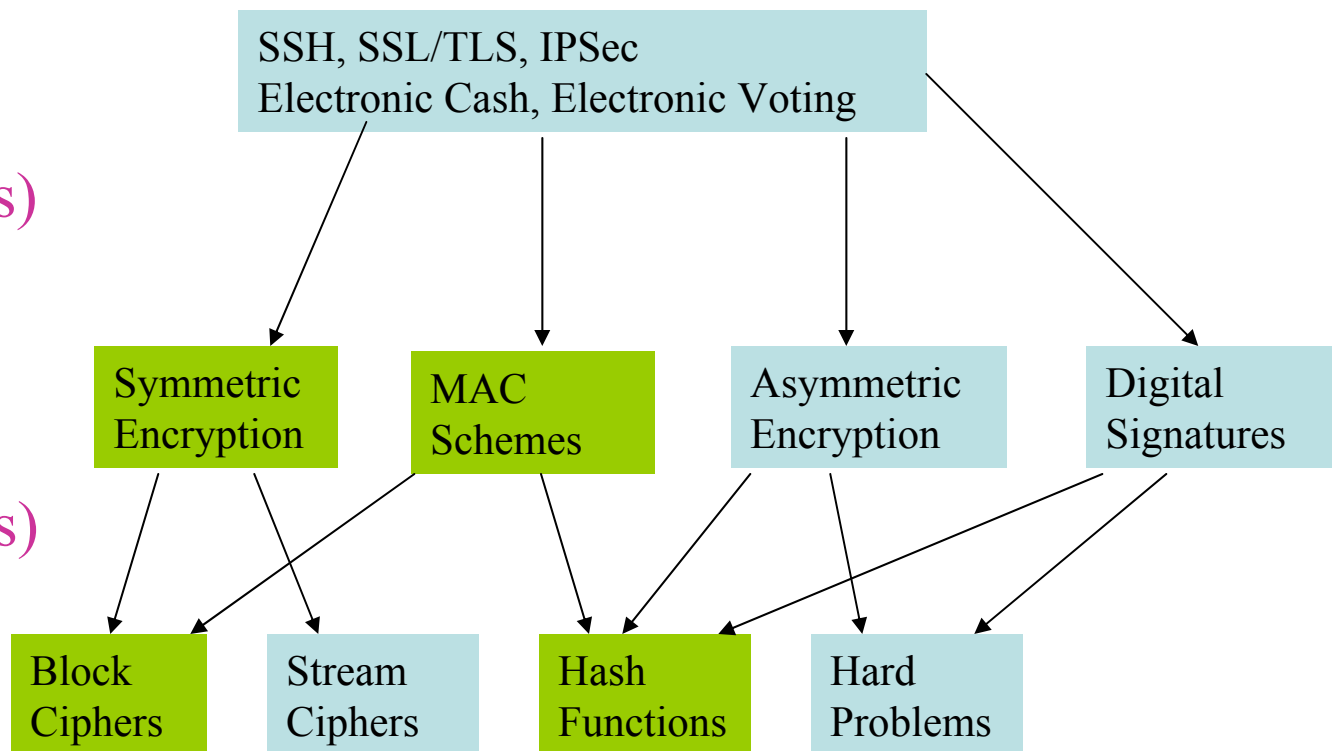
CSCI 6268/TLEN 5831, Fall 2004

# The Big (Partial) Picture

Second-Level  
Protocols  
(Can do proofs)

First-Level  
Protocols  
(Can do proofs)

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



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

# But first, a little math...

- A group is a nonempty set  $G$  along with an operation  $\# : G \times G \rightarrow G$  such that for all  $a, b, c \in G$ 
  - $(a \# b) \# c = a \# (b \# c)$  (associativity)
  - $\exists e \in G$  such that  $e \# a = a \# e = a$  (identity)
  - $\exists a^{-1} \in G$  such that  $a \# a^{-1} = e$  (inverses)
- If  $\forall a, b \in G, a \# b = b \# a$  we say the group is “commutative” or “abelian”
  - All groups in this course will be abelian

# Multiplicative Groups

- Is  $\{0, 1, \dots, m-1\}$  a group under multiplication mod  $m$ ?
  - No, 0 has no inverse
- Ok, toss out 0; is  $\{1, \dots, m-1\}$  a group under multiplication mod  $m$ ?
  - Hmm, try some examples...
    - $m = 2$ , so  $G = \{1\}$  ✓
    - $m = 3$ , so  $G = \{1,2\}$  ✓
    - $m = 4$ , so  $G = \{1,2,3\}$  oops!
    - $m = 5$ , so  $G = \{1,2,3,4\}$  ✓

# Multiplicative Groups (cont)

- What was the problem?
  - 2,3,5 all prime
  - 4 is composite (meaning “not prime”)
- Theorem:  $G = \{1, 2, \dots, m-1\}$  is a group under multiplication mod  $m$  iff  $m$  is prime

Proof:

- : suppose  $m$  is composite, then  $m = ab$  where  $a, b \in G$  and  $a, b \neq 1$ . Then  $ab = m = 0$  and  $G$  is not closed
- ← : follows from a more general theorem we state in a moment

# The Group $Z_m^*$

- $a, b \in \mathbb{N}$  are relatively prime iff  $\gcd(a, b) = 1$ 
  - Often we'll write  $(a, b)$  instead of  $\gcd(a, b)$
- Theorem:  $G = \{a : 1 \leq a \leq m-1, (a, m) = 1\}$  and operation is multiplication mod  $m$  yields a group
  - We name this group  $Z_m^*$
  - We won't prove this (though not too hard)
  - If  $m$  is prime, we recover our first theorem

# Examples of $Z_m^*$

- Let  $m = 15$ 
  - What elements are in  $Z_{15}^*$ ?
    - $\{1, 2, 4, 7, 8, 11, 13, 14\}$
  - What is  $2^{-1}$  in  $Z_{15}^*$ ?
    - First you should check that  $2 \in Z_{15}^*$
    - It is since  $(2, 15) = 1$
  - Trial and error:
    - 1, 2, 4, 7, 8 ✓
  - There is a more efficient way to do this called “Euclid’s Extended Algorithm”
    - Trust me

# Euler's Phi Function

- Definition: The number of elements of a group  $G$  is called the order of  $G$  and is written  $|G|$ 
  - For infinite groups we say  $|G| = \infty$
  - All groups we deal with in cryptography are finite
- Definition: The number of integers  $i < m$  such that  $(i, m) = 1$  is denoted  $\phi(m)$  and is called the “Euler Phi Function”
  - Note that  $|Z_m^*| = \phi(m)$
  - This follows immediately from the definition of  $\phi()$



# Evaluating the Phi Function

- What is  $\phi(p)$  if  $p$  is prime?
  - $p-1$
- What is  $\phi(pq)$  if  $p$  and  $q$  are distinct primes?
  - If  $p, q$  distinct primes,  $\phi(pq) = \phi(p)\phi(q)$
  - Not true if  $p=q$
  - We won't prove this, though it's not hard

# Examples

- What is  $\phi(3)$ ?
  - $|Z_3^*| = |\{1,2\}| = 2$
- What is  $\phi(5)$ ?
- What is  $\phi(15)$ ?
  - $\phi(15) = \phi(3)\phi(5) = 2 \times 4 = 8$
  - Recall,  $Z_{15}^* = \{1,2,4,7,8,11,13,14\}$

# LaGrange's Theorem

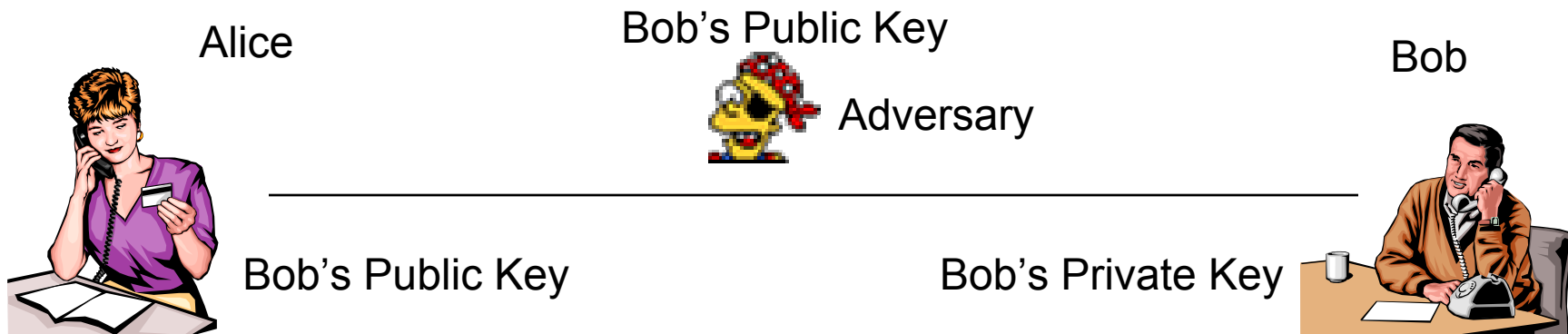
- Last bit of math we'll need for RSA
- Theorem: if  $G$  is any finite group of order  $n$ , then  $\forall a \in G, a^n = 1$ 
  - Examples:
    - $6 \in \mathbb{Z}_{22}$ ,  $6+6+\dots+6$ , 22 times =  $0 \pmod{22}$
    - $2 \in \mathbb{Z}_{15}^*$ ,  $2^8 = 256 = 1 \pmod{15}$
    - Consider  $\{0,1\}^5$  under  $\oplus$ 
      - $01011 \in \{0,1\}^5$ ,  $01011^{32} = 00000^{16} = 00000$
  - It always works (proof requires some work)

# Basic RSA Cryptosystem

- Basic Setup:
  - Alice and Bob do *not* share a key to start with
  - Alice will be the sender, Bob the receiver
    - Reverse what follows for Bob to reply
  - Bob first does key generation
    - He goes off in a corner and computes two keys
    - One key is  $pk$ , the “public key”
    - Other key is  $sk$ , the “secret key” or “private key”
  - After this, Alice can encrypt with  $pk$  and Bob decrypts with  $sk$

# Basic RSA Cryptosystem

- Note that after Alice encrypts with  $pk$ , she cannot even decrypt what she encrypted
  - Only the holder of  $sk$  can decrypt
  - The adversary can have a copy of  $pk$ ; we don't care



# Key Generation

- Bob generates his keys as follows
  - Choose two large distinct random primes  $p, q$
  - Set  $n = pq$  (in  $\mathbb{Z} \dots$  no finite groups yet)
  - Compute  $\phi(n) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$
  - Choose some  $e \in \mathbb{Z}_{\phi(n)}^*$
  - Compute  $d = e^{-1}$  in  $\mathbb{Z}_{\phi(n)}^*$
  - Set  $pk = (e, n)$  and  $sk = (d, n)$ 
    - Here  $(e, n)$  is the ordered pair  $(e, n)$  and does not mean gcd

# Key Generation Notes

- Note that  $pk$  and  $sk$  share  $n$ 
  - Ok, so only  $d$  is secret
- Note that  $d$  is the inverse in the group  $Z_{\phi(n)}^*$  and not in  $Z_n^*$ 
  - Kind of hard to grasp, but we'll see why
- Note that factoring  $n$  would leak  $d$
- And knowing  $\phi(n)$  would lead  $d$ 
  - Bob has no further use for  $p$ ,  $q$ , and  $\phi(n)$  so he shouldn't leave them lying around

# RSA Encryption

- For any message  $M \in \mathbb{Z}_n^*$ 
  - Alice has  $pk = (e, n)$
  - Alice computes  $C = M^e \bmod n$
  - That's it
- To decrypt
  - Bob has  $sk = (d, n)$
  - He computes  $C^d \bmod n = M$ 
    - We need to prove this



# RSA Example

- Let  $p = 19$ ,  $q = 23$ 
  - These aren't large primes, but they're primes!
  - $n = 437$
  - $\phi(n) = 396$
  - Clearly  $5 \in \mathbb{Z}_{396}^*$ , so set  $e=5$
  - Then  $d=317$ 
    - $ed = 5 \times 317 = 1585 = 1 + 4 \times 396$  ✓
  - $pk = (5, 437)$
  - $sk = (396, 437)$

# RSA Example (cont)

- Suppose  $M = 100$  is Alice's message
  - Ensure  $(100, 437) = 1$  ✓
  - Compute  $C = 100^5 \bmod 437 = 85$
  - Send 85 to Bob
- Bob receives  $C = 85$ 
  - Computes  $85^{317} \bmod 437 = 100$  ✓
- We'll discuss implementation issues later

# RSA Proof

- Need to show that for any  $M \in \mathbb{Z}_n^*$ ,  $M^{\text{ed}} = M \pmod n$ 
  - $ed = 1 \pmod{\phi(n)}$  [by def of  $d$ ]
  - So  $ed = k\phi(n) + 1$  [by def of modulus]
  - So working in  $\mathbb{Z}_n^*$ ,  $M^{\text{ed}} = M^{k\phi(n) + 1} = M^{k\phi(n)} M^1 = (M^{\phi(n)})^k M = 1^k M = M$ 
    - Do you see LaGrange's Theorem there?
- This doesn't say anything about the security of RSA, just that we can decrypt

# Security of RSA

- Clearly if we can factor efficiently, RSA breaks
  - It's unknown if breaking RSA implies we can factor
- *Basic* RSA is not good encryption
  - There are problems with using RSA as I've just described; don't do it
  - Use a method like OAEP
    - We won't go into this

# Factoring Technology

- Factoring Algorithms
  - Try everything up to  $\sqrt{n}$ 
    - Good if  $n$  is small
  - Sieving
    - Ditto
  - Quadratic Sieve, Elliptic Curves, Pollard's Rho Algorithm
    - Good up to about 40 bits
  - Number Field Sieve
    - State of the Art for large composites

# The Number Field Sieve

- Running time is estimated as

$$e^{(1.526+o(1))}(\log n)^{1/3}(\log \log n)^{2/3}$$

- This is super-polynomial, but sub-exponential
  - It's unknown what the complexity of this problem is, but it's thought that it lies between P and NPC, assuming  $P \neq NP$

# NFS (cont)

- How it works (sort of)
  - The first step is called “sieving” and it can be widely distributed
  - The second step builds and solves a system of equations in a large matrix and must be done on a large computer
    - Massive memory requirements
    - Usually done on a large supercomputer

# The Record

- In Dec, 2003, RSA-576 was factored
  - That's 576 bits, 174 decimal digits
  - The next number is RSA-640 which is

31074182404900437213507500358885679300373460228427  
27545720161948823206440518081504556346829671723286  
78243791627283803341547107310850191954852900733772  
4822783525742386454014691736602477652346609

- Anyone delivering the two factors gets an immediate A in the class (and 10,000 USD)



# On the Forefront

- Other methods in the offing
  - Bernstein's Integer Factoring Circuits
  - TWIRL and TWINKLE
    - Using lights and mirrors
  - Shamir and Tromer's methods
    - They estimate that factoring a 1024 bit RSA modulus would take 10M USD to build and one year to run
      - Some skepticism has been expressed
  - And the beat goes on...
    - I wonder what the NSA knows

# Implementation Notes

- We didn't say anything about how to *implement* RSA
  - What were the hard steps?!
    - Key generation:
      - Two large primes
      - Finding inverses mod  $\phi(n)$
    - Encryption
      - Computing  $M^e \bmod n$  for large  $M, e, n$
  - All this can be done reasonably efficiently

# Implementation Notes (cont)

- Finding inverses
  - Linear time with Euclid's Extended Algorithm
- Modular exponentiation
  - Use repeated squaring and reduce by the modulus to keep things manageable
- Primality Testing
  - Sieve first, use pseudo-prime test, then Rabin-Miller if you want to be sure
    - Primality testing is the slowest part of all this
    - Ever generate keys for PGP, GPG, OpenSSL, etc?

# Note on Primality Testing

- Primality testing is *different* from factoring
  - Kind of interesting that we can tell something is composite without being able to actually factor it
- Recent result from IIT trio
  - Recently it was shown that deterministic primality testing could be done in polynomial time
    - Complexity was like  $O(n^{12})$ , though it's been slightly reduced since then
  - One of our faculty thought this meant RSA was broken!
- Randomized algorithms like Rabin-Miller are far more efficient than the IIT algorithm, so we'll keep using those