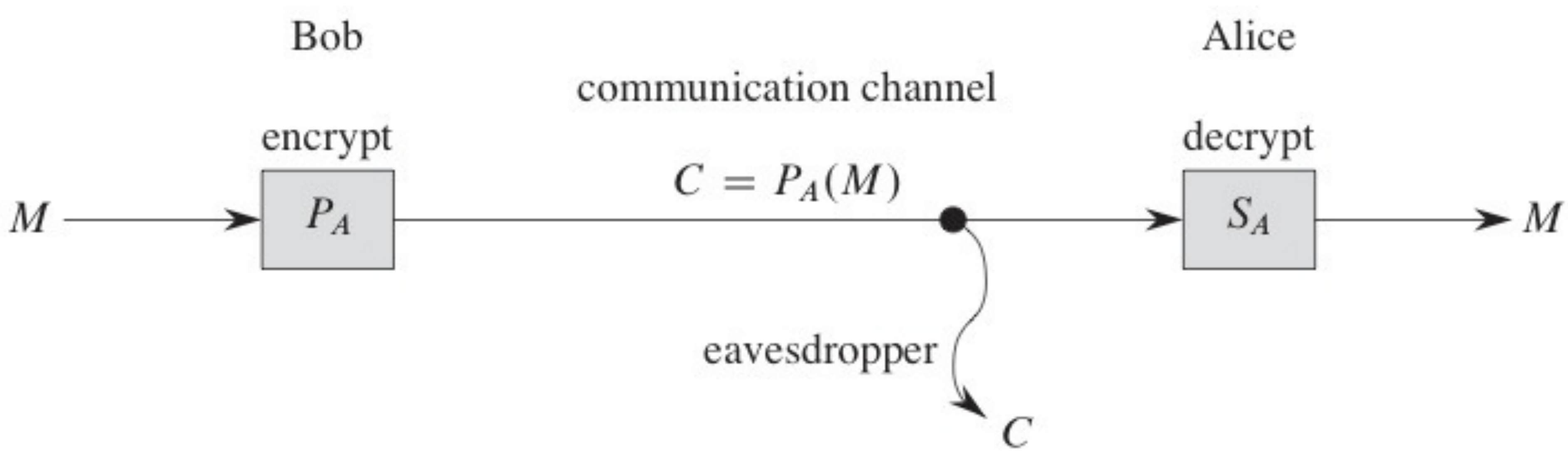
RSA Cryptography

basics of security/cryptography



- Bob encrypts message M into ciphertext C=P(M) using a public key; Bob sends C to Alice
- Alice decrypts ciphertext back into M using a private key (secret) M = S(C)
- anyone else listening gets C but cannot decrypt to M without the private key

Modulo arithmetics

- all variables in this lecture are integers
- "$x=y \mod n$" means x-y is a multiple of n
  - for example 22=2 mod 5, since 22-2=20 is a multiple of 5
  - x and y have the same reminder on division with n
- $a=b \mod n$ and $c=d \mod n$ imply
  - $a+c = b+d \mod n$
  - $a*c = b*d \mod n$
- exponentiation works too, logarithm a bit tricky
  - $a^n = a*a*a...*a \mod n$ //product of a n times
- ax=b mod n equation solvable if all common factors of and n are also factors of b (see 31.4 in the book)
- GCD (greatest common divisor) solution via Extended-Euclid algorithm

RSA

- $n=p*q$; p,q large prime numbers
- $\phi(n) = (p-1)(q-1)$
- e = small integer, r

me with $\phi(n)$
- d = inverse of e mo
  - $d*e = 1 \bmod \phi(n)$

- encoding of message M : $C = P(M)= M^e \bmod n$
- decoding of ciphertext C : $M = S(C) = C^d \bmod n$

RSA demo

- http://www.screencast.com/t/MLcTfBesFvo7

RSA is correct - prelim 1

- Fermat theorem :
  - if p prime, and $a \neq 0 \bmod p$,
  - then $a^{p-1} = 1 \bmod p$
- proof (idea)
  - set S={1, 2, 3,...p-1} is the same as set T= {1a mod p, 2a mod p, 3a mod p, ... (p-1)a mod p. Proof by contradiction: if fa and ga mod p are the same number in S, then

    fa = ga mod p => p| a(f-g)=> p|(f-g) => f=g

  - in S every number can be paired up with its inverse mod p (also in S), so that we can have (p-1)/2 pairs of u*v=1 mod p. That means :

    1*2*3...*(p-1) mod p = (p-1)! mod p = 1 mod p

  - 1= (p-1)! mod p = ∏(elem in S) mod p

    = ∏(elem in T) mod p = 1a*2a*3a*...*(p-1)a mod p

    = $(p-1)!\ a^{p-1}$ mod p = a^{p-1} mod p

RSA is correct - prelim 2

- Chinese Reminder Theorem (simplified) :
  - p,q primes; a fixed integer
  - `x = a mod p ; x = a mod q`
  - then `x = a mod p*q`
- proof (idea)
  - `x = a mod p => x = up+a`; similarly `x=vq+a`
  - `x = up+a = vq+a => up=vq`; since p,q primes => `u=zq`
  - thus `x = up+a = zpq+a = a mod p*q`

RSA is correct - proof

- e,d inverse to each other mod (p-1) (q-1) means

  $$ed = 1+k(p-1)(q-1)$$

- Alice decrypting result is

  $C^d \bmod n = (M^e \bmod n)^d \bmod n = M^{ed} \bmod n.$

- From Fermat Theorem, using $ed = 1+k(p-1)(q-1)$
  - $M^{ed} = M \bmod p$
  - $M^{ed} = M \bmod q$

- From Chinese Reminder Theorem

  $n=p*q$; p,q primes; $M^{ed} = M \bmod p$; $M^{ed} = M \bmod q$

  then $M^{ed} = M \bmod n$

- thus Alice gets back the original message M

RSA easy to implement

- both Bob and Alice only have to execute a modular exponentiation of a given power:
  - given x, compute x^k mod n
- such exponentiation can be implemented efficiently, even for large numbers

Why RSA is secure

- Only known way to break RSA is to factorize n into factors n=p*q
  - p, q unknown
  - there might be other ways to break RSA, but currently unknown
- Factorization is hard when p and q are large
  - although primality testing is easy
  - See the blog page "Factoring Again" (pdf provided) by Richard J. Lipton

How to find large primes p

- pick a random large number (1024 bits) and test if prime

How to find large primes p

- pick a random large number (1024 bits) and test if prime
- **FERMAT (p, $t \neq 0,1$ mod p)**
  - `if` $t^{p-1} \neq 1 \pmod{p}$ `RETURN 0;` *// definitely p not prime due to Fermat's theorem*
  - `if` $t^{p-1} = 1 \pmod{p}$ `RETURN 1` *//we dont know, but we have some belief p might be prime*

How to find large primes p

- pick a random large number (1024 bits) and test if prime
- **FERMAT (p, $t \neq 0,1$ mod p)**
  - if $t^{p-1} \neq 1 \pmod{p}$ RETURN 0; *// definitely p not prime due to Fermat's theorem*
  - if $t^{p-1} = 1 \pmod{p}$ RETURN 1 *//we dont know, but we have some belief p might be prime*
- this procedure can be implemented efficiently by extracting powers of 2 from p-1 first (see book page 969)

How to find large primes p

- pick a random large number (1024 bits) and test if prime
- **FERMAT (p, t≠0,1 mod p)**
  - `if` $t^{p-1} \neq 1 \pmod{p}$ `RETURN 0;` *// definitely p not prime due to Fermat's theorem*
  - `if` $t^{p-1} = 1 \pmod{p}$ `RETURN 1` *//we dont know, but we have some belief p might be prime*
- this procedure can be implemented efficiently by extracting powers of 2 from p-1 first (see book page 969)

- **MILLER-RABIN primality testing (p, s)**
  - `for s independent rounds`
    - `pick t = random (2, p-1)`
    - `if (FERMAT(t,p)==0) RETURN "not prime"` *// definitely correct*
  - `return "prime"` *// rarely incorrect for large s*

How to find large primes p

- pick a random large number (1024 bits) and test if prime
- **FERMAT (p, t≠0,1 mod p)**
  - `if` $t^{p-1} \neq 1 \pmod{p}$ `RETURN 0;` *// definitely p not prime due to Fermat's theorem*
  - `if` $t^{p-1} = 1 \pmod{p}$ `RETURN 1` *//we dont know, but we have some belief p might be prime*
- this procedure can be implemented efficiently by extracting powers of 2 from p-1 first (see book page 969)

- **MILLER-RABIN primality testing (p, s)**
  - `for s independent rounds`
    - `pick t = random (2, p-1)`
    - `if (FERMAT(t,p)==0) RETURN "not prime"` *// definitely correct*
  - `return "prime"` *// rarely incorrect for large s*
- Error probability for MILLE-RABIN (return "prime" on non prime p) is at most 2^{-s}

How many primes are there?

- there are infinitely many primes
- $\pi(n)$ = number of primes smaller or equal to n
- when n is big, $\pi(n) \approx n/\ln(n)$
  - for example $n=10^9$
  - number of primes is up to 10^9 is about $10^9/\ln(10^9) = 48{,}254{,}942$