- Week6
- Public key encryption
  - Bob and alice
    - bob generates (PK,SK)
    - gives PK to alice
    - Alice C = E(PK,m)
    - Bob D(SK,C)
- Applications
  - Session setup (eavesdropping security only)
  - Non-interactive applications (e.g. Email)
    - Bob sends email encrypted to alice
    - Bob needs the pk_alice
- Public key encryption
  - Def: a public-key encryption systems is a triple of algorithms (G,E,D)
    - G(): randomized algorithm outputs a key pair (pk,sk)
    - E(pk,m): randomized algorithm that takes m element M and outputs c element C
    - D(sk,c): deterministic algorithm that takes c element C and outputs m element M or reject
  - Consistency: ForAll(pk,sk) output by G:
    - forall m element M: D(sk,E(pk,m)) = m
- Security: Eavesdropping
  - define two experiments b = 0,1
  - Def: E=(G,E,D) is semantic secure (a.k.a IND-CPA) if for all efficient A:
    - The advantage < negligible
    - semantically secure if the attacker cannot tell if it is the first experiment of the second experiment
- Relation to symmetric cipher security
  - symmetric cipher two security notions
    - One - time security
    - many - time security
  - for public key encryption
    - one-time security = many-time security (CPA)
    - (follows from the fact that attacker can encrypt by himself)
- Security against attacks
  - attacker is given decryption of messages that are routed to him
- Public key chosen ciphertext security: definition
  - E(G,E,D) public key encryption over (M,C) for b = 0,1 define EXP(b)
  - CCA phase 1
    - A sends cipher to B
    - B sends message to A
  - Challenge
    - A sends m0 and m1 |m0| = |m1|
    - B sends c<-E(pk,mb) to A
  - CCA phase 2
    - A sends cipher_i != cipher to B
    - B sends m_i to A
- Chosen ciphertext security: definition
  - Def: E is CCA secure (a.k.a IND-CCA) if for all efficient A:
    - AdvantageCCA is negligible

    - Challenge
        - Example: Suppose (to: alice, body) -> (to: charlie, body)
        - B sends PK to A
        - A sends chal: (to: alice, 0), (to:alice, 1)
        - B sends cipher <- E(pk,mb) to A
    - CCA phase 2
        - A sends c' = (to: charlie, b) != cipher
        - B sends m' <- D(sk,c') to A
- Constructions
- Trapdoor functions
    - (G,F,F^-1) is secure if F(pk,-) is a "one-way" function: can be evaluated, but cannot be inverted without sk
    - Adversary outputs x'
    - Def: (G, F, F^-1) is a secure TDF if for all efficient A:
    - Advantage[A,F] = Pr[x = x'] < negligible
- Public-key encryptions from TDFs
    - (G,F,F^-1): secure TDF X -> Y
    - (E_s,D_s): symmetric encryption defined over (K,M,C)
    - H: X -> K a hash function
    - we construct a public key encryption system (G,E,D):
        - Key generation G: same G for TDF
    - Encryption and decryption
        - E(pk,m)
            - x <-_r X
            - k <- H(x)
            - y <- F(pk,x)
            - c <- E_s(k,m)
            - output (y,c)
        - D(sk,(y,c)):
            - x <- F^-1(sk,y)
            - k <- H(x)
            - m <- D_s(k,c)
            - output m
    - Security Theorem:
        - if (G, F, F^-1) is a secure TDF, (E_s,D_s) provides authenticated encryption and H: X -> K is a "random oracle" then (G,E,D) is CCA secure
- Incorrect use of a Trapdoor Function (TDF)
    - E(pk,m)
        - output c <- F(pk,m)
    - D(sk, c):
        - output F^-1(sk, c)
    - Problems
        - deterministic: cannot be semantically secure
        - many attacks exist (next segment)
    - Never apply trapdoor function to the message m
- RSA Trapdoor permutation
- Trapdoor permutations review
    - the function F(pk,-) is one-way without the trapdoor sk
- The RSA trapdoor permutation

  - encryption exponent e
  - decryption exponent d
  - choose e and d s.t. e*d = 1 (mod phi(N)) see notes
  - G(): choose random primes p,q approximately 1024 bits
  - N = pq
  - output pk = (N,e), sk = (N,d)
  - F(pk , x): Z*_N => Z*_N
  - RSA(x) = x^e (in Z_n)
  - F^-1(sk,y) = y^d
  - y^d = RSA(x)^d = x^ed = … = x
- The RSA assumption
  - RSA is one-way permutation
  - For all efficient algorithms A:
    - PR[A(N,e,y) = y^1/e] < negligible
    - where p,q <-_r n bit primes
    - N <- pq
    - y <-_R Z*_n
- RSA public key encryption review
  - Symmetric encryption system
    - (E_s,D_s): symmetric encryption scheme providing authenticated encryption
    - H: Z_n -> K where K is key space of (E_s,D_s)
      - G(): generate RSA params: pk = (N,e), sk = (N,d)
      - E(pk,m):
        - (1) choose random x in Z_n
        - (2) y <- RSA(x) = x^e, k <- H(x)
        - (3) output (y, Es(k,m))
      - D(sk, (y,c)): output D_s(H(RSA^-1(y)),c) -> m
- Textbook RSA is insecure
  - Textbook RSA encryption
    - public key: (N,e) Encryption x <- m^e (in Zn)
    - secret key: (N,d) Decryption c^d -> m
  - Insecure cryptosystem
    - Is not semantically secure and many attacks exist
    - The RSA trapdoor permutation is not an encryption scheme
- A simple attack on textbook RSA
  - Step 1: build table
  - Step 2: test if k_2^e is in table
  - Output matching (k1,k2)
- RSA encryption in practice
  - Never use textbook RSA
  - RSA in practice
    - message key 128 bits -> preprocessing 2048 bits -> RSA -> ciphertext
- PKCS1
  - [PKCS1 mode 2 (16 bits 02) | random pad (encryption) | (FF) | (msg)]
  - resulting values is RSA encrypted
- Attack on PKCS
- Baby Bleichenbacher
- HTTPS Defense
- PKCS v2.0 OAEP (Optimal asymmetric encryption padding)

    - preprocessing function OAEP
        - check pad on decryption reject CT if invalid
    - Thm: RSA is a trap-door permutation => RSA-OAEP is CCA secure when H,G are random oracles
    - optimal because ciphertext is short as possible
    - Thm: is false if use general trap door permutation
- OAEP improvements
- Subtleties in implementing OAEP
    - Problem: timing information leaks type of error => attacker can decrypt any ciphertext
- Is RSA a one-way function ?
- Is RSA a one-way permutation?
    - To invert the RSA one-way function attack must compute x from c = x^e (mod N).
    - How hard is computing e'th roots modulo N??
    - Best known algorithm
        - Step 1: factor N (hard)
        - Step 2: compute e'th roots modulo p and q
- Wiener's attack
    - given (N,e) recover d
- RSA in Practice
- RSA With Low public exponent
    - to speed up RSA encryption use a small e
- Implementation attacks
    - Timing attack - decryption time should be independent of the arguments
    - Power attack - defend against power analysis attacks
    - Faults attack - one error reveals secret key
- Public Key Encryption Form Diffie-Hellman: ElGamal
- The ElGamal Public-key System
- Recap: public key encryption: (Gen,E,D)
    - Gen(): pk,sk
- Public key encryption applications
    - Key exchange
    - Encryption in non-interactive settings
        - Secure email
        - Encrypted File Systems
    - Key escrow: data recovery without Bob's key
- The Diffie-Hellman protocol
    - Fix a cyclic group G of order n
    - Fix a generator g in G
    - Alice choose random a in {1,…n}
    - Bob choose random b in {1,…n}
    - k_ab = g^ab
    - The attacker is allowed to see A and B
        - the secret key is AB
        - this believed to be a hard or difficult problem
- The ELGamal System (a modern view)
    - symmetric system encryption decryption
    - better to choose random generator every time
    - G: finite cyclic group of order n
    - (E_s,D_s): symmetric auth. encryption defined over (K,M,C)

    - H:G^s -> K hash function
    - E(pk = (g,h),m)
        - b <-_r Z_n
        - u <- g^b
        - v <- h^b
        - k <- H(u,v)
        - c <- E_s(k,m)
        - output (u,x)
    - D(sk = a, (u,c)):
        - v <- u^a
        - k <- H(u,v)
        - m <- D_S(k,c)
        - output m
- ElGamal Performance
    - windowed exponentiation is when you precompute the tables
- ElGamal Security
    - G: finite cyclic group of order n
    - for all efficient algorithms A:
    - Pr[g^ab] < negligible
        - g <- {generators of G}
        - a,b <- Z_n