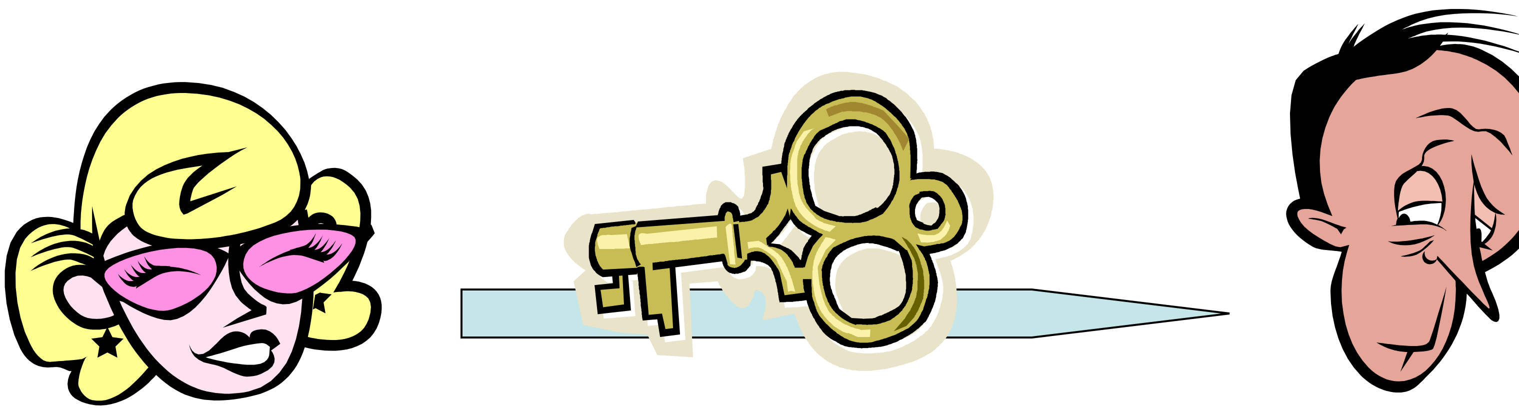
Public Key Cryptography

Introduction

Public Key Cryptography

- Unlike symmetric key, there is no need for Alice and Bob to share a common secret
  - Alice can convey her public key to Bob in a public communication:



Encrypting w/ Public Keys

- Public key schemes encrypt large blocks of data:
  - Smallest system with reasonable security has block sizes at least 160 bits (Elliptic Curves)
  - Key size generally equal to or close to block size
  - Orders of magnitude less efficient than symmetric key encryption

Why public key?

- The reason public keys are used is to establish secure communication when there is no way to exchange a key beforehand.
  - Confidential/authenticated channels for free?
- Must ensure that the public key belongs to the correct party (binding of identity to key). The public key directory may be corrupted:
  - Solution: Use a Public Key Infrastructure to certify your keys (PKI)

Encryption: Details

- Alice knows Bob's public key P_{Bob}
- Uses the encryption algorithm:
  - $Enc(P_{Bob}, Message) = C$
- Anybody may encrypt messages that only Bob may read, since he knows the private key S_{Bob}
- $Message = Dec(S_{Bob}, C)$

How does Bob know S_{Bob}?

- How did Bob come to know his private key to start with?
  - The answer is that Bob generates the pair (P_{Bob}, S_{Bob}) jointly. The key generation procedure is probabilistic and one-way.
  - The security of such methods is closely related to a class of mathematical problems from modular arithmetic

Modular arithmetic review

- Consider the integer interval $[0, N-1]$ for some integer N.
- This is the set of possible remainder values when one divides any integer by N.
  - Called the set of residues modulo N
- Note that addition is well defined in this set:
  - Let A be an integer with residue mod N equal to a; and $B \bmod N = b$, for a, b in $[0, N-1]$.
  - That means: $A = kN + a$, and $B = lN + b$, for some k, l (use long division)
  - Then $A + B = (k + l)N + (a+b)$
  - So: $(A + B) \bmod N = a + b = [(a \bmod N) + (b \bmod N)] \bmod N$

Adding residues

- Example:

+	0	1	2	3
0	0			
1	1	2		
2	2	3	0	
3	3	0	1	2

N = 4

+	0	1	2	3	4
0	0				
1	1	2			
2	2	3	4		
3	3	4	0	1	
4	4	0	1	2	3

N = 5

Properties of modular addition

- associativity:
  - $(a + b \bmod N) + c \bmod N = a + (b + c \bmod N) \bmod N$
- commutativity:
  - $(a + b \bmod N) = (b + a \bmod N)$
- identity element:
  - $(a + 0 \bmod N) = (a \bmod N)$
- existence of additive inverse:
  - For each residue $a \bmod N$, there is a residue b such that $(a + b \bmod N) = 0 \bmod N$.
  - Just take $b = N - a$ (as integers)
  - We write $b = -a$ as usual.

Multiplying residues

- We could define multiplication of residues as:
  - $(a \bmod N)*(b \bmod N) := (a * b \bmod N)$
- The definition works, because if
  - $A = k\,N + a$, $B = l\,N + b$
  - $(A * B) = k\,l\,N^2 + (k\,b + l\,a)N + a\,b$
  - So $(A * B \bmod N) = a\,b \bmod N$
- Multiplication of residues is also associative, commutative, and has an identity element ($1 \bmod N$)
- It distributes with addition:
  - $(c \bmod N)(a + b \bmod N) = ca + cb \bmod N$

Multiplication Example

*	0	1	2	3
0	0			
1	0	1		
2	0	2	0	
3	0	3	2	1

N = 4

*	0	1	2	3	4
0	0				
1	0	1			
2	0	2	4		
3	0	3	1	4	
4	0	4	3	2	1

N = 5

Multiplicative inverse

- Not always defined.
  - 2 is not invertible mod 4, from previous example
- if $c = 1 \bmod N$, c it is relatively prime to N.
- If a is NOT relatively prime to N, then:
  - No multiple ab of a is relatively prime to N, and so:
  - No b can satisfy $ab = 1 \bmod N$
  - For a to be invertible mod N it is *necessary* that $GCD(a, N) = 1$
- From the theory of the greatest common divisor (GCD):
  - If $g = GCD(a, N)$ then there exist b, k such that $a\,b + k\,N = g$
  - So if a is relatively prime to N, there exists b, k with $a\,b + k\,N = 1$
  - b is the inverse of a mod N
  - For a to be invertible mod N it is *sufficient* that $GCD(a, N) = 1$

Number rings and fields

*A set of elements with operations + and *, such that*

- addition is associative, commutative, has identity "0" and each element a has an additive inverse $-a$
- multiplication is associative, (commutative), and has identity

*is called a **(commutative) ring***

- If moreover, each element a (except for "0") has a multiplicative inverse a^{-1}, the set is called a *field*.
- In cryptography, *finite fields* and *finite rings* (finite sets with the above properties) are very important.
- Examples of finite rings, which are not fields: The set of residues $[0, N-1]$ with operations as before, where N is a composite number
- Examples of finite fields: The set of residues $[0, p-1]$, where p is a prime.

The discrete logarithm problem

- Let p be a prime number
  - A large one, say 1000 -- 2000 bits long.
- Take g to be in the interval $[2, p-2]$.
- Consider the exponential function:
  - $Exp_g(\bullet, \text{mod } p)$: $x \rightarrow g^x \text{ mod } p$
- $Exp_g(\bullet, \text{mod } p)$ is hard to invert.
  - unless p is a "weak" prime (rare case and easy to test for)

Example

- N = 11; g = 2.
- $2^2 = 4$, $2^3 = 8$, $2^4 = 5$, $2^5 = 10$, $2^6 = 9$, $2^7 = 7$, $2^8 = 3$, $2^9 = 6$, $2^{10} = 1 \pmod{11}$
- $3^2 = 9$, $3^3 = 5$, $3^4 = 4$, $3^5 = 1 \pmod{11}$
- The residue (2 mod 11) can create all non-zero residues mod 11 via exponentiation. It is called a *generator*.
- The residue (3 mod 11) does not have the same property.

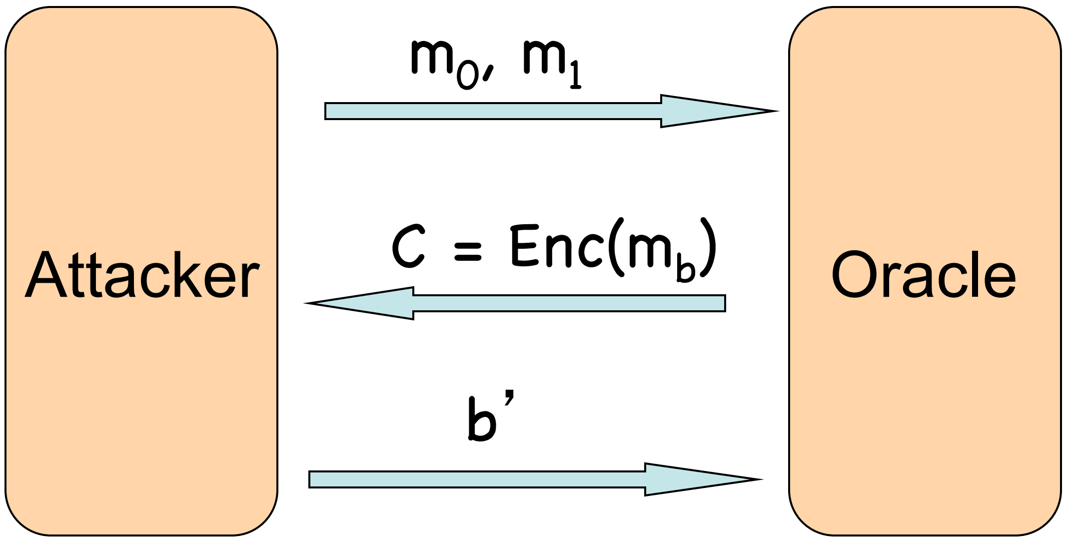
Encrypting a la Elgamal

- Take p a non-weak prime:
  - $p = uq + 1$, with q also prime, and u small.
  - This guarantees $Exp(\bullet, \bmod p)$ is hard.
- Take g' in $[2, p-2]$ and choose
  - $g = g'^{u} \bmod p$.
- Choose private key k at random
  - k in $[2, q-1]$
- Compute the public key $y = g^k \bmod p$.

ElGamal Encryption

- To encrypt m in $[1, p-1]$ for user Bob:
  - public key $y = g^k$, private key k
- Compute a random value r
- Compute
  - $(A, B) = (g^r \bmod p, m\, y^r \bmod p)$
- To decrypt, Bob computes
  - $m = B(A^{-k}) \bmod p$

Semantic security



If b' = b, the attacker wins.

If every attacker has only a negligible probability of success, we say that the scheme is *secure under chosen-plaintext attacks*.

Security of Elgamal encryption

- When the attacker receives:
  - $(g^r, m_b y^r)$
- it may divide the second term by $m_{b'}$
  - $(g^r, m_b (m_{b'})^{-1} y^r) = (A, B)$
- To decide if $b = b'$, need to decide if, given (g, y, A, B), the last two values have the form:
  - (g^r, y^r) for some r, or not.
- This is called the *Decision Diffie-Hellman* (DDH) problem, and it is considered a difficult number theory problem---no efficient algorithms for it are known.

Elgamal and chosen-ciphertext attacks

- Elgamal is NOT secure against chosen ciphertext attacks
- Suppose the system wants to prevent you from decrypting a ciphertext (A, B), but may allow you to decrypt a different ciphertext:
  - Compute
    - $(A', B') = (A, k\,B) \bmod p$
  - If you get
    - $m' = Dec\,(A', B')$,
  - then compute
    - $m = (k)^{-1}\,m' \bmod p$
  - Note that $m = Dec(A, B)$, so the attacker wins.
- This is not a problem in practice, because Elgamal is used in practice as a hybrid scheme (see next).

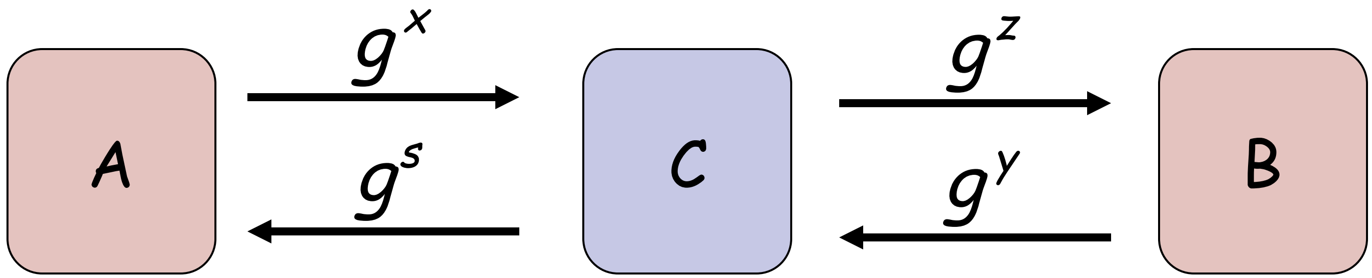
Hybrid Scheme

- Use the public key encryption scheme to encrypt a key for a symmetric encryption scheme (e.g., AES)
- Use the symmetric key to encrypt the data
- More generally, two algorithms:
  - Key Encapsulation Mechanism (KEM) wraps a symmetric key using the public key encryption algorithm
  - Data Encapsulation Mechanism (DEM) encrypts the message contents using the symmetric key encoded in the KEM

Key Agreement

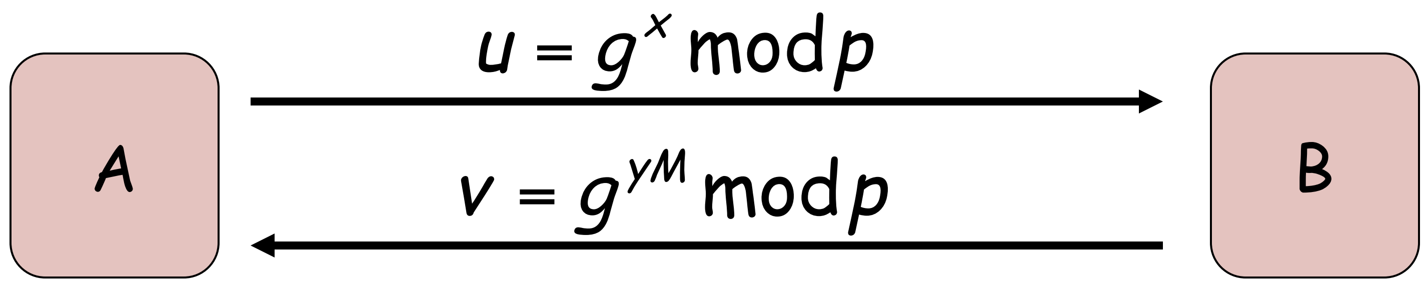
- Alice to Bob:
  - $g^a \bmod p$, with a random
- Bob to Alice:
  - $g^b \bmod p$, with b random
- Session key derived from shared secret, but without authentication:
  - $g^{ab} \bmod p$
- Computing the key g^{ab} from (g, g^a, g^b) is the computational Diffie-Hellman problem (CDH)
- CDH must be at least as hard as DDH
- CDH at most as hard as computing logarithms to basis $g \bmod p$

Man-in-the-middle attack



$K_1 = g^{xs}$ $\quad$ $K_2 = g^{zy}$

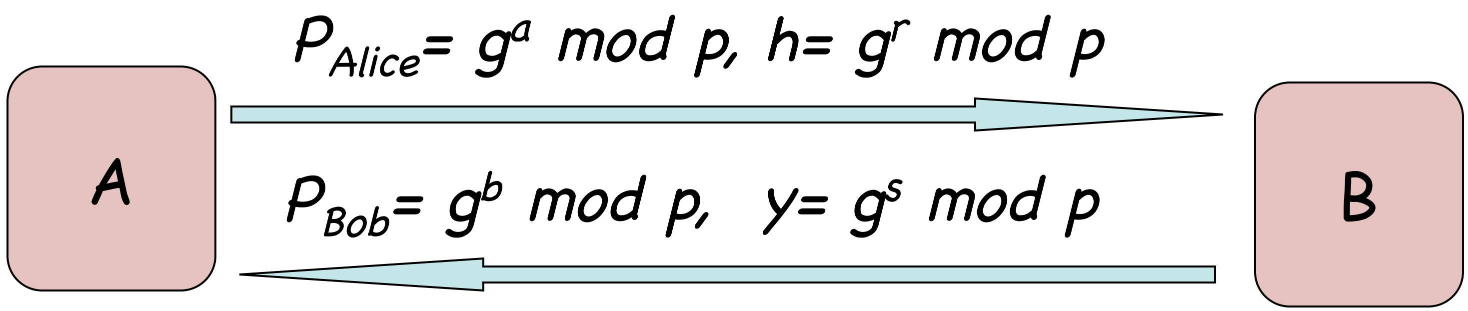
Adding authentication



$$K = v^{xM^{-1}} = g^{xy} \qquad\qquad K = u^y = g^{xy}$$

Here, $g^M = P_{Alice}$, the public key of Alice.

MTI: Authenticated DH



$K = (P_{Bob})^r \, y^a \bmod p$

$K = (P_{Alice})^s \, h^b \bmod p$

$$K = g^{sa + rb} \bmod p$$

DSA keys

- Generate large prime $p = kq + 1$,
  - p originally 512 bits, today 1024 or more
  - q originally 160 bits (still safer today).
- Generator g such that $g^q = 1 \mod p$.
  - Take $h \in [1, p - 1]$; set $g = h^{(p-1)/q} \mod p$
- Choose private-public key pair: $<T, S>$
  - S random in $[1, q]$; $T = g^S \mod p$

Signing w/ DSA

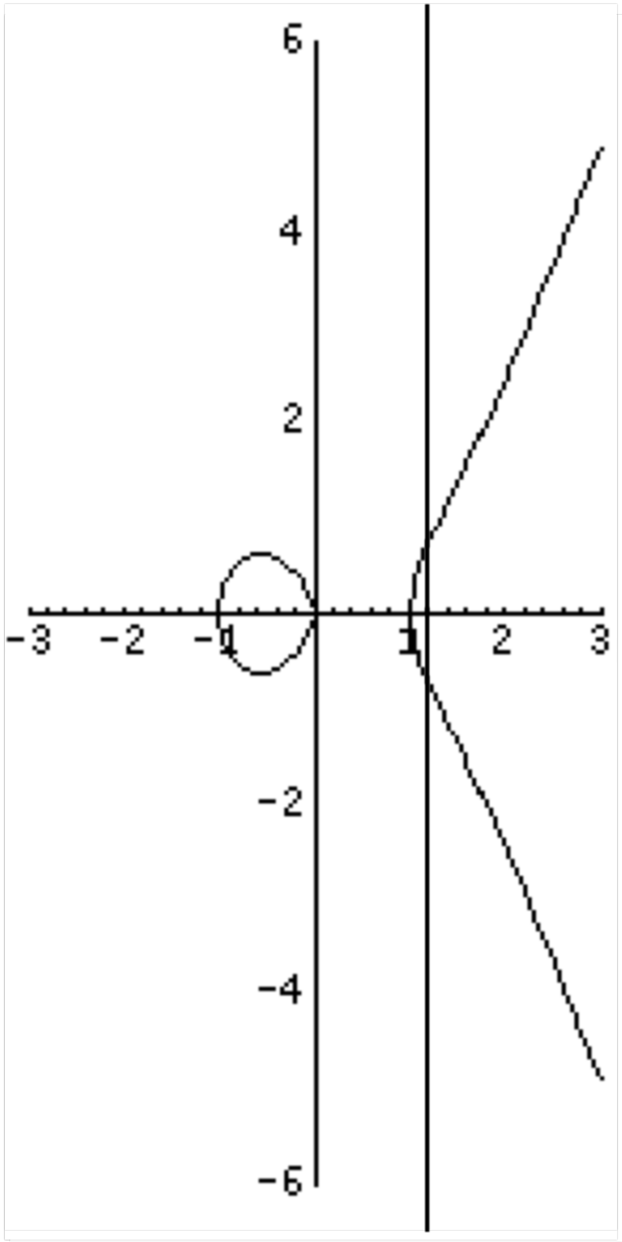
- Generate a per-message private/public key pair:
  - $<T_m, S_m> : T_m = g^{S_m} \mod p$
- d_m = digest of message (e.g., SHA-1)
- Compute the signature
  - $X = S_m^{-1} (d_m + S\ T_m) \mod q$
- The signing pair is $(T_m \bmod q, X)$

Verifying the DSA

- Calculate the inverse of X:
  - $X^{-1} \mod q$
- Calculate d_m from the message m
- Compute $a = d_m X^{-1} \mod q$
- Compute $b = T_m X^{-1} \mod q$
- Compute $z' = (g^a T^b \mod p)$
- If $z = T_m \mod q$, verification succeeds.

Elliptic Curves

- Alternative cryptographic settings for discrete-log based public key schemes
  - Elliptic Curve DSA (ECDSA)
  - Elliptic Curve Diffie-Hellman Key Exchange
- Given a finite field F, an elliptic curve is the set of solutions in $F \times F$ to an equation of the form:
  - $y^2 = x^3 + Ax + B$



- Graph of the equation $y^2 = x^3 - x$
- Most lines intersect the curve at 3 points
- Vertical lines intersect only at 2 points
  - Add a "virtual point" O at "vertical infinity" and say all vertical lines pass through it. Then all lines cut the curve in 3 points.

Operating on points

- We can define an operation $\otimes$ on the points of an elliptic curve.
- The "point at infinity" O will act as identity:
  - $O \otimes P = P \otimes O = P$, for all points
- If a line passes through three points, P_1, P_2, and P_3, then we say that
  - $P_1 \otimes P_2 \otimes P_3 = O$

Using the rules

- The inverse of a point P with coordinates (x, y) is the point $Q = (x, -y)$. Why?
  - If $P = (x, y)$ solves $y^2 = x^3 + Ax + B$, so does $Q = (x, -y)$, so at least makes sense
  - P and Q are in a vertical line, so our previous rule say that
  - $P \otimes Q \otimes O = O$, or $P \otimes Q = O$. Good.

Can we compute the product?

- Yes: Take two points, P and Q.
- Write the equation of the line α that passes through P and Q.
- Compute the third point R in the intersection of α with the curve E.
- By our rule, $P \otimes Q \otimes R = O$
- Then if S is the inverse of R (which we know how to compute as before), then
  - $S = P \otimes Q$

Elgamal, Diffie-Hellman, and DSA on Elliptic curves

- Now that we learned how to multiply points on the elliptic curve, we can do:
- EC-Elgamal encryption:
  - Public key (P, Q), where $Q = P^x$.
  - Encrypt m as $(P^r, m \oplus H(Q^r))$. (See next slide)
- ECDH: $P^a, P^b \rightarrow P^{ab}$
- ECDSA also same.
- Note: Sometimes operation called addition, instead of multiplication. In this case, ECDH would be written
  - ECDH: $aP, bP \rightarrow abP$

Caveats

- In order for using Elgamal encryption, it would be necessary to encode a message m as a point in the Elliptic curve.
- This is cumbersome. More practical to encrypt as:
  - $(P^r, m \text{ XOR } H(Q^r))$, where H is a hash function from the elliptic curve to binary strings.
  - Constructing this hash function is easier than encoding elements in the curve