Multiple Key Cryptography: Overview

Going beyond Public and Private Keys

- We've seen several uses for multiple key cryptography in the "Standard" PKE
  - Ex: public key, shared with everyone, used to check sender
  - Ex: private key, shared with no one, used to limit access

- Although this a common use of multiple keys, there are others.

Four Techniques beyond "standard" PKE

- Diffie-Helman Key Exchange - sharing a key
- Message Broadcast
- Secret Sharing
- Secret Splitting

Diffie Helman Key Exchange

- Considered a public key algorithm
  - This is the first one, invented late ‘70s
- Primarily used for generation of a key to be used in a symmetric algorithm

Two User Protocol

- Alice and Bob share “public” keys
  - They agree on two large numbers: n, g
  - $1 < g < n$
  - n should be a prime number
  - (n-1)/2 should also be a prime
  - n should be at least 512 bits <= that advice is from the early 90’s and is probably dated!
  - Note that there are some known bad choices for DH which should be avoided

The Private Keys

- Alice needs a private key
  - She chooses a large random integer, x
- Bob needs a private key
  - He also chooses a large random integer, y

The Exchange, part 1

- Alice uses her private key and the public keys to compute X (cap X):
  - $X = g^{x} \bmod n$
- Alice sends X (cap X) to Bob
  - g, n public
  - x private
  - X sent to Bob

- Bob uses his private key and the public keys to compute Y (cap Y):
  - $Y = g^{y} \bmod n$
- Bob sends Y (cap Y) to Bob
  - g, n public
  - y private
  - Ysent to Alice

The Exchange, part 2

- Alice computes k1 as follows
  - **$k1 = Y^x \bmod n$**
- **The keys**
  - **x is private**
  - **n is public**
  - **Y is from Bob**
  - **k1 is computed**

- Bob computes k2 as follows
  - **$k2 = X^y \bmod n$**
- **The keys**
  - **y is private**
  - **n is public**
  - **X is from Alice**
  - **k2 is computed**

Keys K1 and K2 are identicial!

$$
\begin{aligned}
k2 &= X^y \bmod n \\
&= (g^x \bmod n)^y \bmod n \\
&= (g^{xy}) \bmod n \\
&= (g^{yx}) \bmod n \\
&= (g^y \bmod n)^x \bmod n \\
&= (Y)^x \bmod n \\
&= k1
\end{aligned}
$$

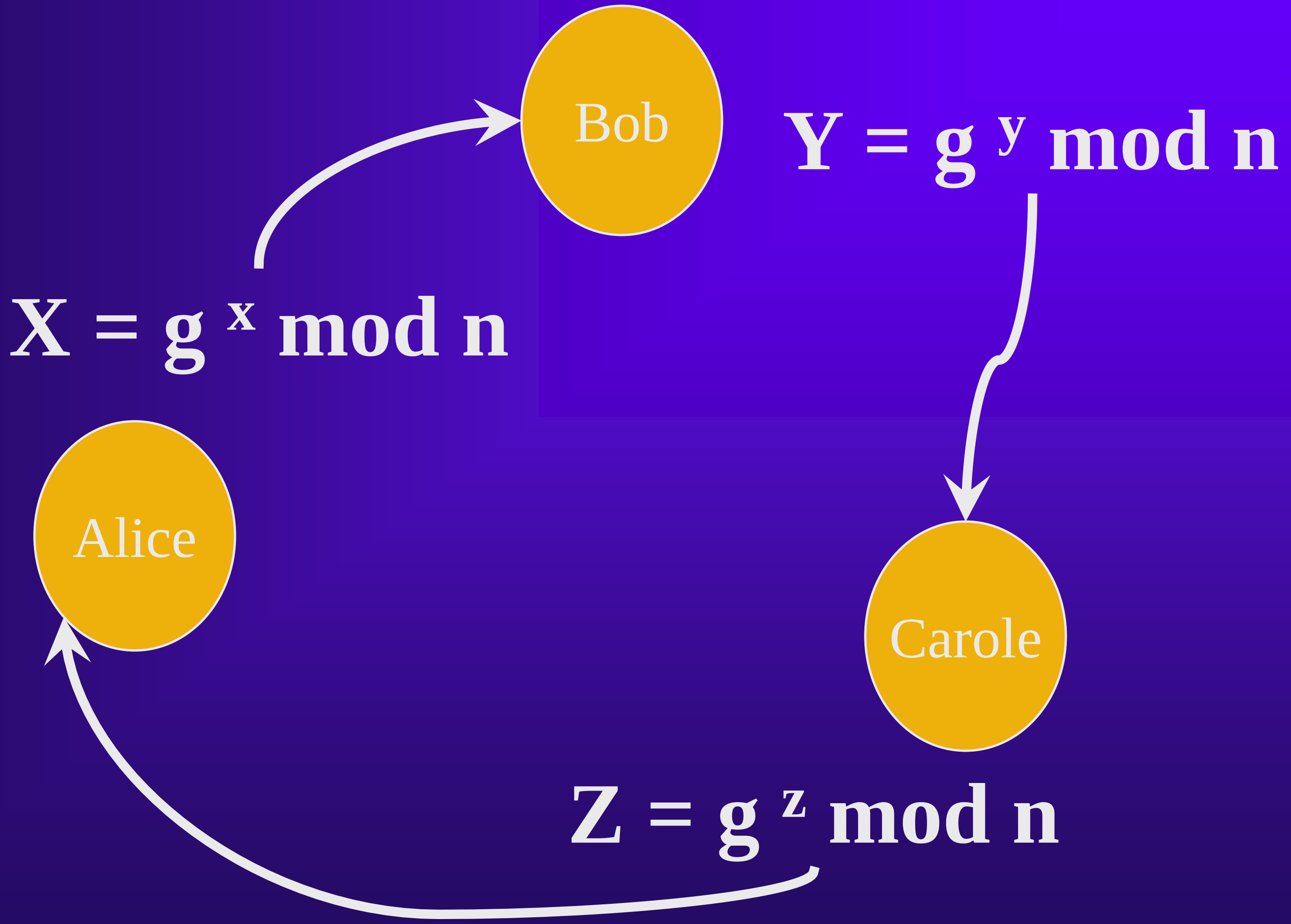
Useful points

- Even if Mal can obtain n, g, X, Y he cannot directly compute k1=k2
  - It may still be possible to *guess* the key, perhaps based on analysis of encrypted text, of course
- The algorithm is relatively efficient in terms of yielding a good key for use in a symmetric algorithm
- Especially nice for session keys
- Does involve additional communication
- No trusted third party is needed

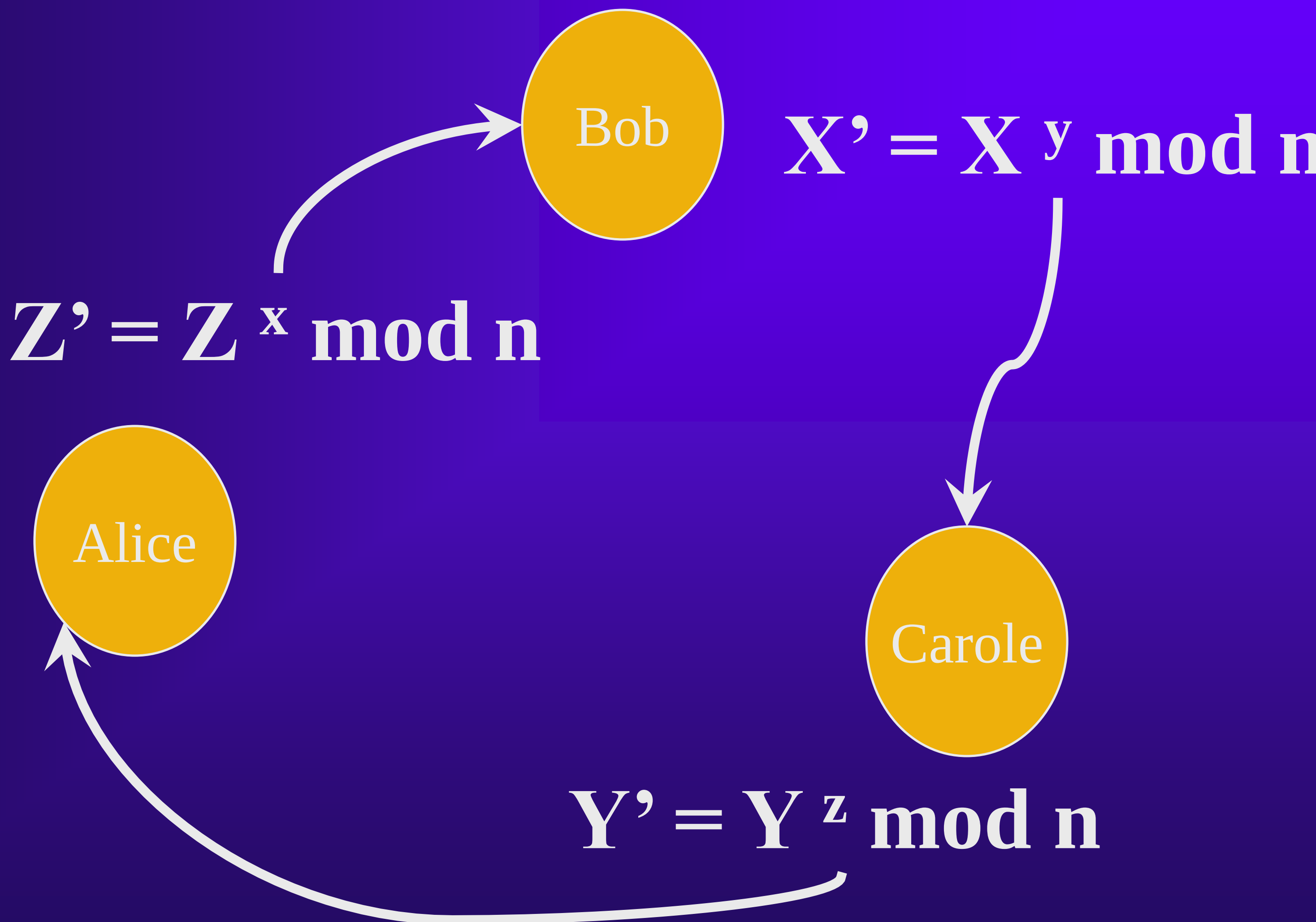
The algorithm works for more than two participants

- We can use an extended version of DH for key exchange between three or more participants
- Each additional participant will increase the number of rounds of exchange
- We start as usual, with g, n agreed upon in advance

Key Exchange, Step 1



Key Exchange, Step 2



Key Exchange, Step 3

Bob

$K = (Z')^{y} \bmod n$

$K = (Y')^{x} \bmod n$

Alice

$K = g^{xyz} \bmod n$

Carole

$K = (X')^{z} \bmod n$

Secure Message Broadcast

- Suppose that we have a very large number of entities that we wish to communicate with from a central location
- Sometimes we want to send the *same* message to a subset of these entities … but that the particular subset may vary

Some Options

- We could develop or share a different symmetric key with each entity
  - we would have as many key pairs as message recipients
  - we would have to encrypt the message for each recipient
  - Variant: We could also sign and send the *same* key to each recipient, then encrypt the message itself just once

Another Option

- We could identify all possible subsets of recipients, assign each one a “private” key (private to that group) in advance.
  - This would let us encrypt the message only once per subset of recipients
  - This requires *many* keys
  - This is a hard strategy to use when recipients can “leave” or “enter” groups

Using Multiple RSA

- We could also use the “multiple” version of RSA (details are left to homework)
- Operationally, it works like this:
  - Multiple Public Key Cryptography
  - Determine a set of “semi-public” keys based on the number of desired subsets.
  - Distribute the keys so that each subset has at least one in common
  - Encrypt with the “inverse set” of the common keys

An Example with $K_A K_B K_C$

Encrypt With:	Decrypt With
K_A	K_B K_C
K_A K_B	K_C
K_A K_C	K_B

Essentially, if "k" of the "n" keys are used to encrypt then we need the remaining "n-k" keys to decrypt.

Secret Splitting

- Usage: you want to keep “components” of some secret in several locations, so that compromise of one location won’t compromise the entire secret.
- This is a very “strong” method, in that it is not vulnerable to guessing IFF it is used correctly
- This method is vulnerable to destruction of one of the “secret” locations

Simple Technique!

- Trent has a message M to protect
- Trent generates a random message R with as many bits in it as message M
- Trent uses XOR of M and R to generate P

$$P = M \oplus R$$

- Trent gives P to Alice and R to Bob and destroys M
- Success!

Reconstruction

- If Trent wishes to reconstruct the original message:
  - Trent gets P from Alice
  - Trent gets R from Bob
  - Trent XORs them together; the result is M

$$P \oplus R$$

- As long as we do not reuse R, no amount of brute force guessing will allow an enemy to learn that he/she has the right M if only one of P or R is compromised

Quick Illustration

- Suppose that the message is 1101
- Trent chooses random number 0101

$$P = M \oplus R = (1101) \oplus (0101)$$

$$\text{Bitwise: } (1 \oplus 0,\ 1 \oplus 1, 0 \oplus 0, 1 \oplus 1)$$

$$P = (1,0,0,0)$$

- Reversing:

$$P \oplus R = (1000) \oplus (0101)$$

$$\text{Bitwise: } (1 \oplus 0,\ 0 \oplus 1, 0 \oplus 0, 0 \oplus 1)$$

$$M = (1,1,0,1)$$

We can extend this:

- For splitting the secret M amongst n individuals rather than two, Trent must generate a sequence of random strings:

  $R_1 \ldots R_{n-1}$

- Computing P is then:
  - $P = M \oplus R_1 \oplus \ldots \oplus R_{n-1}$
- Reconstruction involves XORing all of the Ri plus P.

Secret Sharing

- Note that secret splitting was vulnerable to the loss of one site
- If you wish to balance a desire to preserve a message with the need to keep the message secret, a "secret sharing" technique may be more appropriate
  - There are several
  - We'll talk about one involving Polynomials!
  - Note that this example is vastly scaled down for ease of typing :)

Threshold Scheme

- (m,n) Threshold Scheme:
  - A secret is divided into “n” pieces (called the shadows), such that combining any “m” of the shadows will reconstruct the original secret.
- We’ll use Shamir’s LaGrange Interpolating Polynomial Scheme as our example (scaled down!)

Shamir's Scheme

- Choose a (public) large polynomial "p" bigger than
  - the possible number of shadows
  - the size of the secret
  - other requirements for strength
  - all arithmetic will be "mod p"
- Generate an arbitrary polynomial of degree "m-1"
- Evaluate the polynomial at "n" different points to obtain the shadows "ki"
- Distribute the shadows and destroy M and all the polynomial coefficients

Example poly: (3,n) threshold

- Form of our arbitrary polynomial
- m=3 so polynomial is degree 2
  - $F(x) = ax^2 + bx + M \qquad \pmod{P}$
- We must decide on a size for n - this is the number of shadows. The number of shadows is independent of the size of the polynomial

(3,5) Threshold with M=11

- Suppose we want a (3,5) scheme - that means we will have 5 shadows - for hiding message "11" (eleven)
- We choose a prime >5, 11: say 13
- Our polynomial must be degree 2. We select the coefficients a, b randomly:

$$F(x) = 7x^2 + 8x + 11 \pmod{13}$$

Continuing …

- We must now generate five shadows. We decide to evaluate at points 1,2,3,4,5 (normally we'd mix them up!)
  - $F(x) = 7x^2 + 8x + 11 \pmod{13}$
  - $k1 = F(x1=1) = 7+8+11 = 0$
  - $k2 = F(x2=2) = \ldots = 3$
  - $k3 = F(x3=3) = \ldots = 7$
  - $k4 = F(x4=4) = \ldots = 12$
  - $k5 = F(x5=5) = \ldots = 5$

Distribution!

- We then put the shadows (the ki) somewhere. We need to either keep the selected value for x with the shadow or with the "coordinator". Discard M, a, b.

(1,0) (2,3) (3,7) (5,5) (4,12)

How do we get the message back?

- We know that this is a (3,5) scheme, so the polynomial is known to be of degree 2:

$$F(x) = Ax^2 + Bx + M$$

(1,0)

(2,3)

(3,7)

(5,5)

(4,12)

Restoring…

- We would obtain THREE shadows from any of the five locations below. That would give us three equations and three unknowns:

$$F(x) = Ax^2 + Bx + M$$

(1,0)

(2,3)

(3,7)

(5,5)

(4,12)

Restoring…

- For instance, choose shadows k2,k3,k5

$$F(5) = A5^2 + B5 + M = 5$$

(5,5)

(2,3)

$$F(2) = A2^2 + B2 + M = 3$$

$$F(3) = A3^2 + B3 + M = 7$$

(3,7)

This gives us three euqations and three unknowns, which is solvable, and yields A=7, B=8, M=11.