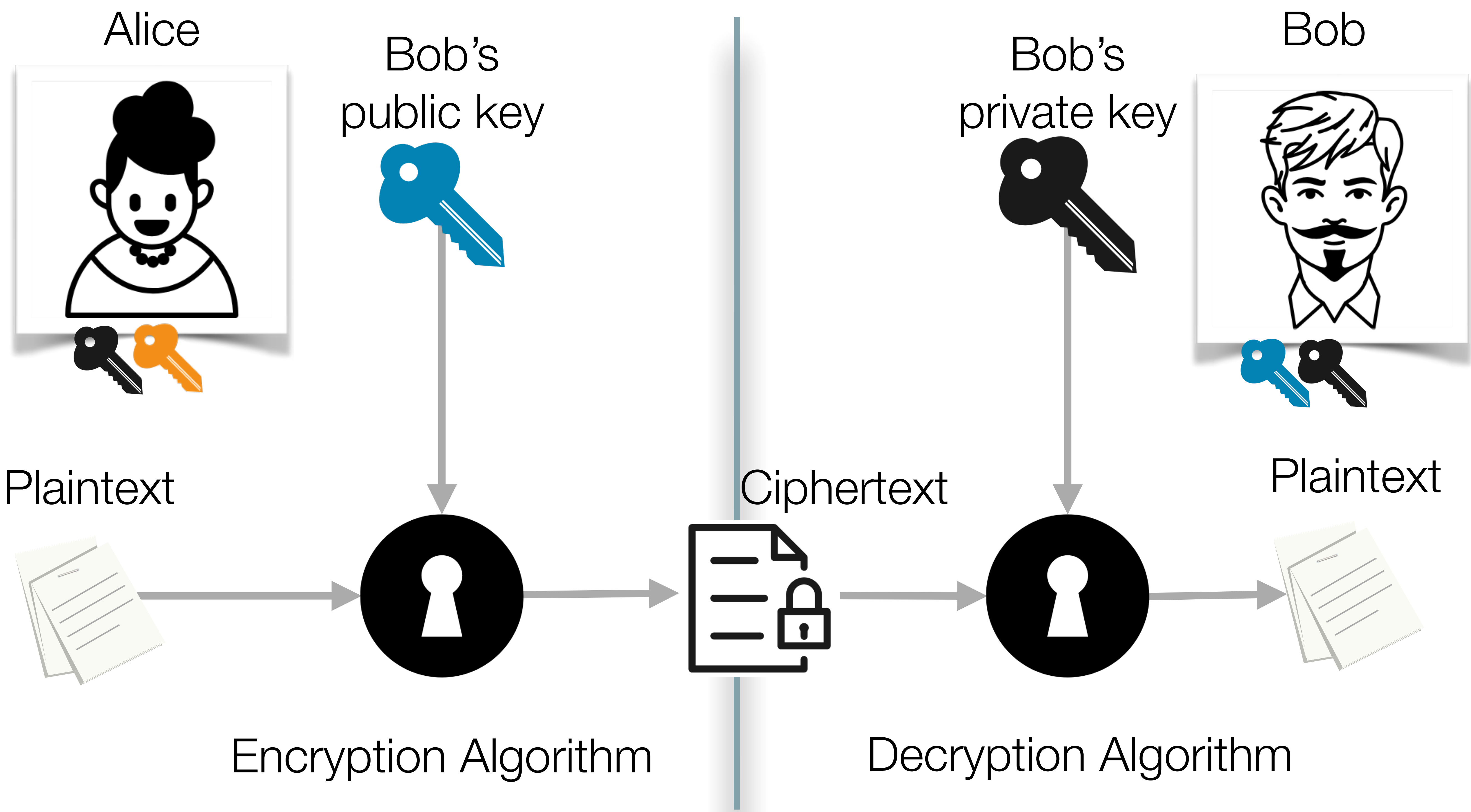
CS 465
Computer Security

Public Key Cryptography

Daniel Zappala, adapted from Tim van der Horst and Kent Seamons
Fall 2018

Public Key Encryption Model



Why Public Key Crypto is Cool

- Has a linear solution to the key distribution problem
  - Symmetric crypto has an exponential solution
- Send messages to people you don’t share a secret key with
  - So only they can read it
  - They know it came for you

Number Theory

Prime Numbers

- Definition: An integer whose only factors are 1 and itself
- There are an infinite number of primes
- How many primes are there?
  - Any large number n has about a 1 in ln(n) chance of being prime

Prime Number Questions

- If everyone needs a different prime number won't we run out?
    - Approximately 10^{151} primes 512 bits (or less)
    - Atoms in the universe: 10^{77}
    - If every atom in the universe needed 1 billion primes every microsecond from the beginning of time until now you would need 10^{109} primes
    - That means there's still about 10^{151} left

Source: Applied Cryptography (Schneier)

Prime Number Questions

- What if two people pick the same prime?
  - Odds are significantly less than the odds of your computer spontaneously combusting at the exact moment you win the lotto
- Couldn't someone create a database of all primes and use that to break public key crypto?
  - Assuming you could store 1 GB/gram, then the weight of a drive containing the 512-bit primes would exceed the Chandrasekhar limit (theoretical maximum mass a white dwarf star can have and still remain a white dwarf) and collapse into a black hole

Source: Applied Cryptography (Schneier)

Prime Factorization :
The Fundamental Theorem of Arithmetic

- All integers can be expressed as a product of (powers of) primes
  - 48 = 2 * 2 * 2 * 2 * 3
- Factorization is the process of finding the prime factors of a number
- This is a **hard** problem for large numbers

Greatest Common Divisor (GCD)

- A.k.a., greatest common factor
- The largest number that evenly divides two numbers
  - GCD (15, 25) = 5

Relatively Prime

- Two numbers *x* and *y* are relatively prime if their GCD = 1
- No common factors except 1
- Example – 38 and 55 are relatively prime
  - 38 = 2 * 19
  - 55 = 5 * 11

Modular (%) Arithmetic

- Sometimes referred to as “clock arithmetic” or “arithmetic on a circle”
- Two numbers a and b are said to be congruent (equal) modulo N iff (a-b)/N=0
  - Equivalent statements: their difference is divisible by N with no remainder, their difference is a multiple of N, a%n ≡ b%n
  - Example: 30 and 40 are congruent mod 10
- Modulo operation
  - Find the remainder, e.g.15 mod 10 = 5

Notation

- Z - the set of integers {…-2,-1,0,1,2…}
- Z_n - the set of integers modulo n; {0..n-1}
- Z_n* - the multiplicative group of integers modulo n
  - the set of integers modulo n that are relatively prime to n
  - Z_n* is closed under multiplication mod n
  - Z_n* does not contain 0 since the GCD(0,n)=n
  - Z_{10}^* = ? Z_{12}^* = ? Z_{14}^* = ?

Additive Inverse

- In Z, the additive inverse of 3 is -3,
  since 3 + -3 = 0, the additive identity.

- In Z_n, the additive inverse of a is n-a,
  since a+(n-a) = n, which is congruent to 0 (mod n).

    - What is the additive inverse of 4 mod 10?

Multiplicative Inverse

- In Z, the multiplicative inverse of 3 is 1/3, since 3*1/3=1
- The multiplicative identity in both Z and Z_n is 1
- The multiplicative inverse of 3 mod 10 is 7, since 3*7=21=1 (mod 10)
    - This could be written 3^{-1}, or (rarely) 1/3

Distributive Property

- Distribution in + and *
- Modular arithmetic is distributive.

```
a+b (mod n) =  (a mod n) + (b mod n) (mod n)
```

Big Examples

What is the sum of these numbers modulo 20?

$$
\begin{array}{r}
1325104987134069812734109243861723406983176 \\
1346139046817340961834764359873409884750983 \\
3632462309486723465794078340898340923876314 \\
3641346983862309587235093857324095683753245 \\
\underline{+\ \ 2346982743069384673469268723406982374936877}
\end{array}
$$

Big Examples

What is the product of these numbers modulo 25?

```
  1234659823572938572
  2139582753931306947
  1398173619384713413
  2496827464249812355
  2436781359183781379
* 1351839761361377050
```

Modular Exponentiation

- Problems of the form $c = b^e \bmod m$
  given base b, exponent e, and modulus m
- If b, e, and m are non-negative and $b < m$, then a unique solution c exists and has the property $0 \leq c < m$
- For example, $12 = 5^2 \bmod 13$
- Modular exponentiation problems are easy to solve, even for very large numbers
- However, solving the discrete logarithm (finding e given c, b, and m) is believed to be difficult

Brute Force Method

- The most straightforward method to calculating a modular exponent is to calculate b^e directly, then to take this number modulo m.

- Consider trying to compute c, given b = 4, e = 13, and m = 497:

    - Using a calculator, compute 4^{13} = 67,108,864., modulo 497, c = 445.

    - Note that b is only one digit in length and that e is only two digits in length, but the value b^e is 10 digits in length.

Source: wikipedia – modular exponentiation

Brute Force Method

- In strong cryptography, b is often at least 256 binary digits (77 decimal digits).
- Consider b = 5 * 1076 and e = 17, both of which are perfectly reasonable values. In this example, b is 77 digits in length and e is 2 digits in length, but the value b^e is 1304 decimal digits in length.
- Such calculations are possible on modern computers, but the sheer enormity of such numbers causes the speed of calculations to slow considerably. As b and e increase even further to provide better security, the value be becomes unwieldy.

Source: wikipedia – modular exponentiation

Brute Force Method

- The time required to perform the exponentiation depends on the operating environment and the processor. If exponentiation is performed as a series of multiplications, then this requires O(e) time to complete.

Source: wikipedia – modular exponentiation

Diffie Hellman Project

- Write your own modular exponentiation routine
  - Use a bignum library
  - Divide and conquer algorithm O(log e)

Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange

- Allows two users to establish a secret key over an insecure medium without any prior secrets
- Two system parameters ***p*** and ***g***.
  - Public values that may be used by all the users in a system
  - Parameter p is a large prime number
  - Parameter g (usually called a generator) is an integer less than p, such that for every number n with 0 < n < p , there is a power k of g such that **n = gk mod p**
  - g is called a primitive root

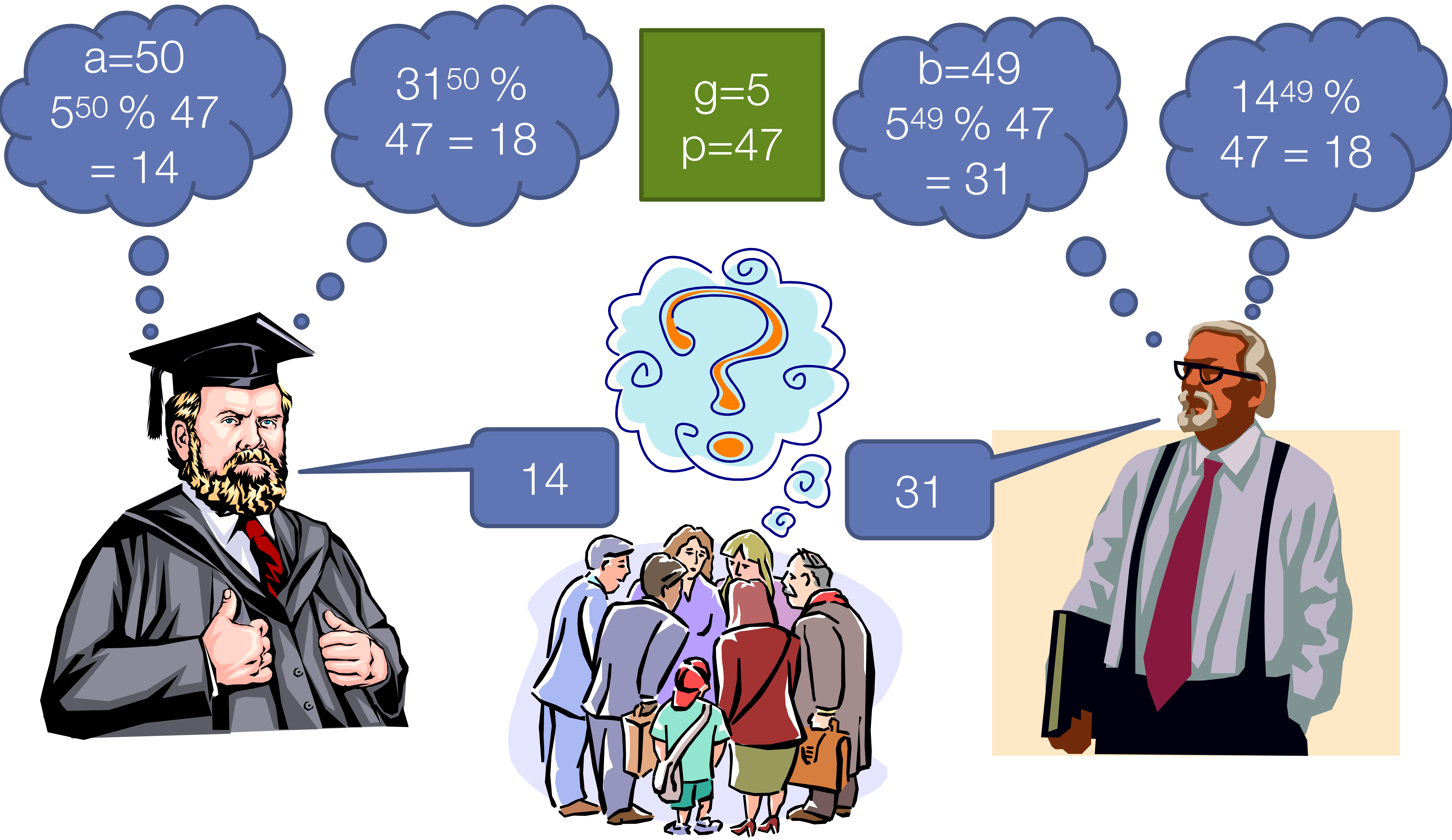
Diffie-Hellman Key Exchange

- Alice and Bob want to establish a shared secret key
- Alice and Bob agree on or use public values **p**, **g**
  - p is a large prime number
  - g is a generator
- Alice generates a random private value **a** and Bob generates a random private value **b** where a and b are integers

Diffie-Hellman Key Exchange

- Alice and Bob derive their public values using parameters p and g and their private values
  - Alice's public value = $g^a \bmod p$
  - Bob's public value = $g^b \bmod p$
- Alice and Bob exchange their public values
- Alice computes $\mathbf{g^{ba} = (g^b)^a \bmod p}$
  Bob computes $\mathbf{g^{ab} = (g^a)^b \bmod p}$
- Since $\mathbf{g^{ab} = g^{ba} = k}$, Alice and Bob now have a shared secret key **k**

A Crowded Room of Mathematicians



Why is DH Secure?

- Discrete logarithm problem
  - Inverse of modular exponentiation
- $c = b^e \bmod m$
  - e is called the “discrete logarithm”
- Solving the discrete logarithm (finding **e** given **c**, **b**, and **m**) is believed to be difficult for large numbers
- See https://www.nku.edu/~christensen/092mat483%20DH%20key%20exchange.pdf

Attacks Against DH

- Diffie-Hellman Key Exchange is secure against a passive attacker
- How can an active attacker disrupt the protocol? Consider a man in the middle
  - Modify Alice/Bob public values as they are exchanged
  - Replace with Mallory's public values
  - Replace g^a and g^b with the value 1
  - Replace g^a with h that has a small order (small number of elements generated by h mod p), which makes it easy to break — see small subgroup attacks
- Must use a protocol to provide authentication and integrity

Practical Considerations

- Chose a safe prime p where p=2q+1 where q is also prime
  - Safe prime means the group G has a subgroup of large size (q)
  - Unfortunately, very inefficient
- How big should p be?
  - Cryptography Engineering, published 2010: 2048 bits until 2022, 3072 bits until 2038, 4096 bits until 2050.
- Check public values for security properties
  - Both p and q are prime, q is 256 bits long, and p is sufficiently large
  - q is a divisor of (p - 1)
  - g != 1 and $g^q = 1$
- Hash final result of DH to generate a shared key for Alice/Bob

Practical Considerations

- How to fortify the protocol against active attackers?
  - Create a certified list of public values
  - Use digitally signed public parameters
- Public values for Diffie-Hellman:
  - https://datatracker.ietf.org/doc/rfc3526/?include_text=1