# 51.505 – Foundations of Cybersecurity

## Week 10 – Public-Key Cryptography

Created by **Pawel Szalachowski** (2017)
Modified by **Jianying Zhou** (2018)

Last updated: 6 Nov 2018

# Recap

- Questions on Week 9’s exercises?

# Primes

- ***a* divides *b***, if you can divide ***b*** by ***a*** w/o leaving a reminder.
  - ✓ $a \mid b$, e.g., $7 \mid 35$
- A number is a ***prime*** when it has only two positive divisors (1 and itself).
  - ✓ Otherwise the number is called a ***composite***.
  - ✓ Is the number 1 prime or composite?

# Primes

- **Lemma 1:** If $a \mid b$ and $b \mid c$, then $a \mid c$.

  *Proof.* $a \mid b \rightarrow \exists\ s\ (integer), as = b$

  $b \mid c \rightarrow \exists\ t\ (integer), bt = c$

  $\rightarrow (as)t = c \rightarrow a(st) = c \rightarrow a \mid c$

- **Lemma 2:** Let $n > 1$ and $d > 1$ be the smallest divisor of $n$, then $d$ is prime.

  *Proof ?*

# Primes

- **Lemma 3:** There are an infinite number of primes.

  *Proof.* Assume the number of primes is finite (with $k$ primes)

  We can define $n = p_1 p_2 \ldots p_k + 1$ (the product of *<u>all</u>* primes plus one)

  $n$ is not prime, otherwise there are $k+1$ primes.

  Suppose $d$ is the smallest divisor of $n$, $d \mid n$, then $d$ is prime (by Lemma 2).

  That means there are $k+1$ primes. → a contradiction.

  [Proven by Euclid over 2000 years ago !]

- **Goldbach conjecture:** Every even number greater than 2 is the sum of two primes.

  *Proof ?* (→ Fields award ? ☺)

# Modulo

- Modulo operation: ***a mod N*** returns remainder after division of *a* by *N.*
  - ✓ Results are $0,1,...,N-1$,     *e.g.*, $25 \ (mod\ 7) = 4$
  - ✓ To compute $r = a \ (mod\ N)$, find integers $q$ and $r$: $a = qN + r$
  - ✓ $-1 \ (mod\ N) = ?$
- In cryptography $N$ is usually a prime.
  - ✓ we use notation *mod p.*

# Computations Modulo

- Addition
  - ✓ *(a + b) mod N*
    - Compute and reduce modulo
  - ✓ *(a + b + c + d) mod N*
    - Compute *(a mod N + b mod N + ... ) mod N*
- Subtraction
  - ✓ *(a – b) mod N*
    - Add *N* if the result is negative.

# Computations Modulo

- Multiplication

  ✓ $x*y \bmod N = y*x \bmod N$

  ✓ $\underbrace{x*x*\ldots*x}_{a} \bmod N = x^a \bmod N$

  ✓ $x^{ab} \bmod N = x^{ba} \bmod N$

  ✓ $(x^a)^b \bmod N = x^{ab} \bmod N$

# Computations Modulo

- Division

✓ $a/b \bmod N$ is the multiplication $ab^{-1} \bmod N$.

  - Another notation of $b^{-1}$ is $1/b$

✓ $b^{-1}$ (a modular inverse of b) is a number such that $bb^{-1} = 1 \bmod N$.

  - *What is $5^{-1} \bmod 7$ ?*

✓ How to compute modular inverses ?

# The Greatest Common Divisor

- ***gcd(a, b)*** = the largest $k$ such that $k \mid a$ and $k \mid b$.
- Euclid gave an algorithm for computing GCD over 2000 years ago.

```
function gcd(a,b)
  while a ≠ b
    if a > b
      a := a − b;
    else
      b := b − a;
  return a;
```

# Extended Euclidean Algorithm

- ***egcd(a,b):*** Given (*a*,*b*) returns (*r*,*s*,*t*) such that *r* = *gcd(a,b)* and *sa* + *tb* = *r.*

```
function egcd(a, b)
    s := 0;    old_s := 1
    t := 1;    old_t := 0
    r := b;    old_r := a
    while r ≠ 0
        quotient := old_r div r
        (old_r, r) := (r, old_r - quotient * r)
        (old_s, s) := (s, old_s - quotient * s)
        (old_t, t) := (t, old_t - quotient * t)
    return (old_r, old_s, old_t)
```

- **How to compute $b^{-1}$ *mod p* (for $1 \leq b < p$)?**
  - ✓ Compute *egcd(b, p), output r,s,t.*
  - ✓ $sb + tp = r$ (r = gcd(b,p) = 1 as *p* is prime)
  - ✓ $sb = 1 - tp$ → $sb = 1 \bmod p$ → $\mathbf{s = b^{-1} \bmod p}$

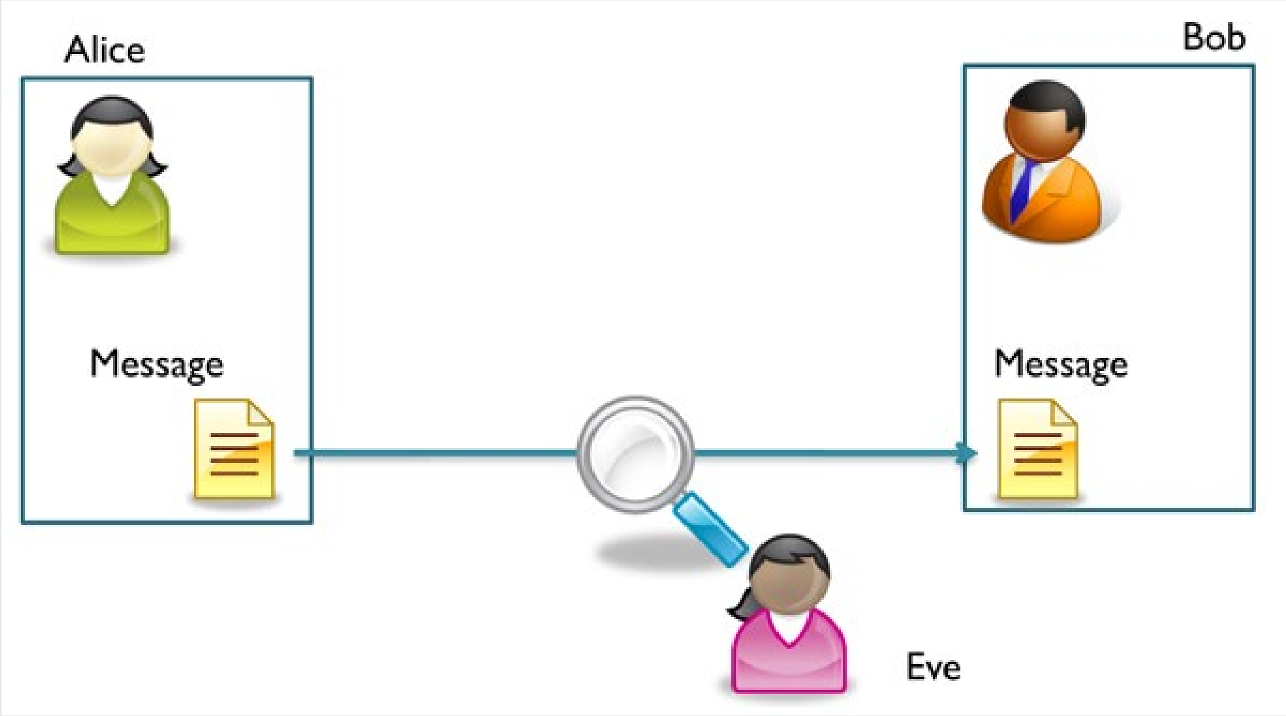
# Generating Large Primes

- 2048-8192 bits long primes
- Primality testing (probabilistic)
  - ✓ Take a random number and check if it passes primality test(s)
  - ✓ The Rabin-Miller test
  - ✓ $2^{-128}$ error bound

# Diffie-Hellman (DH)

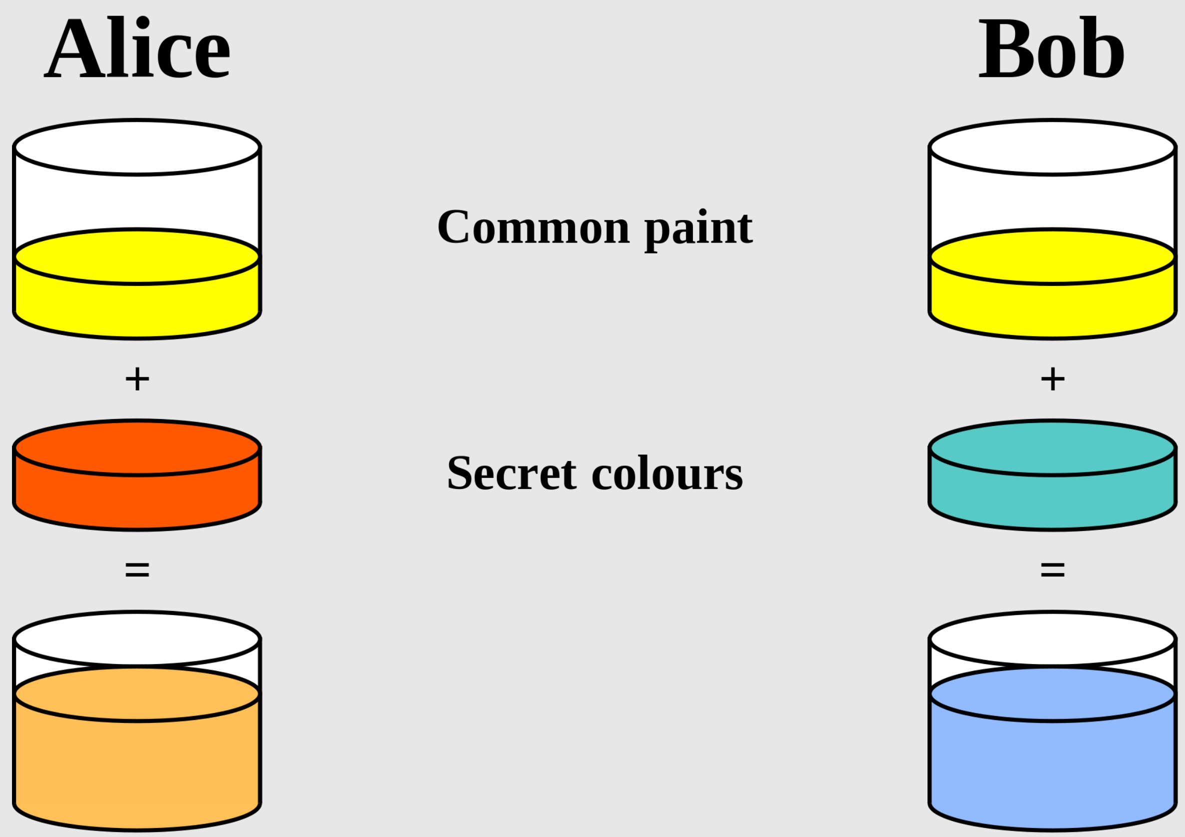
# Problem Definition

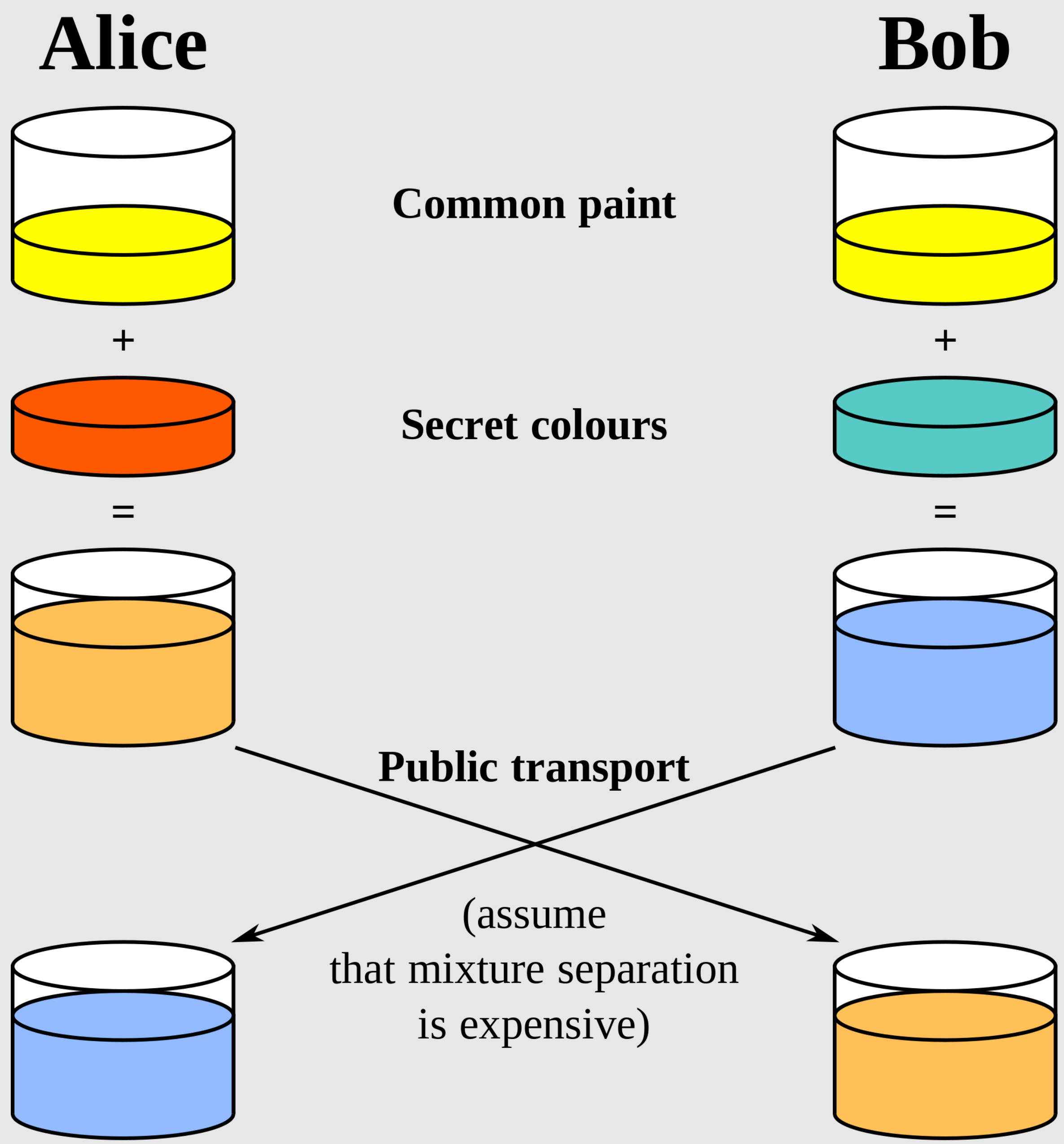
- Secure communication over insecure channel?
  - ✓ Two parties: **A**lice and **B**ob
  - ✓ Eavesdropping adversary: **E**ve

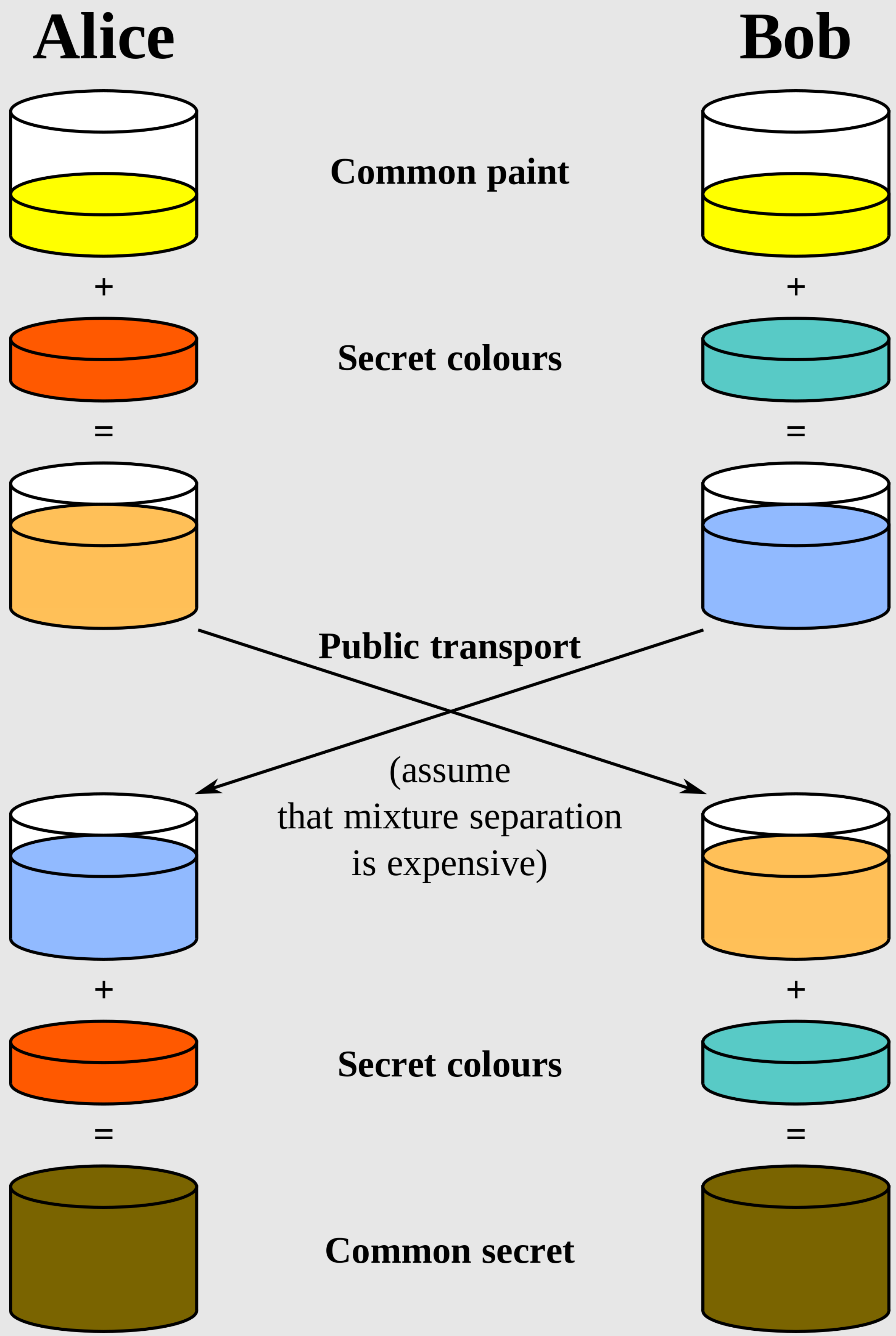


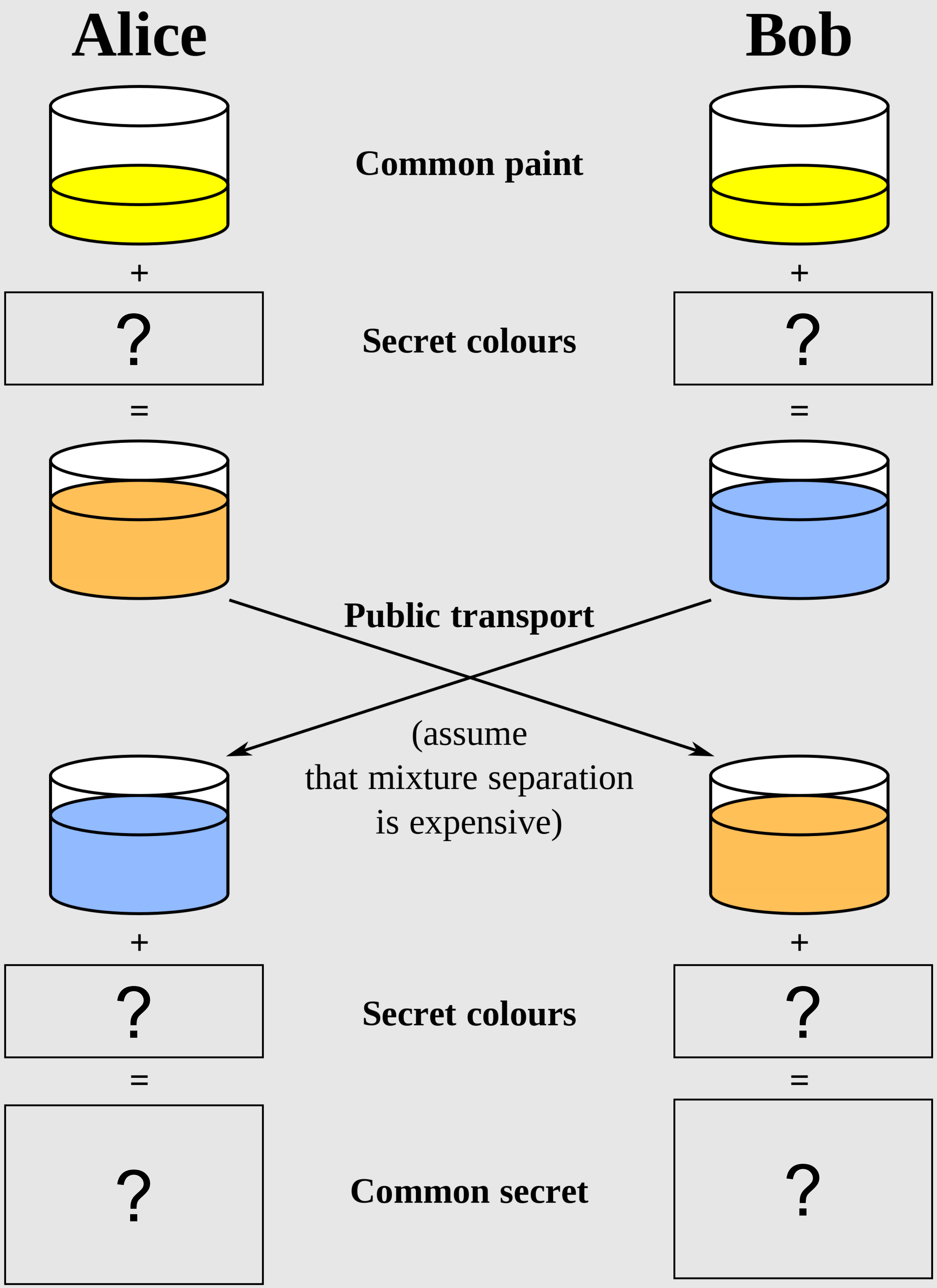

- How Alice and Bob can establish a shared secret?


Alice
Bob
Common paint


Alice
Bob
Common paint
+
+
Secret colours
=
=

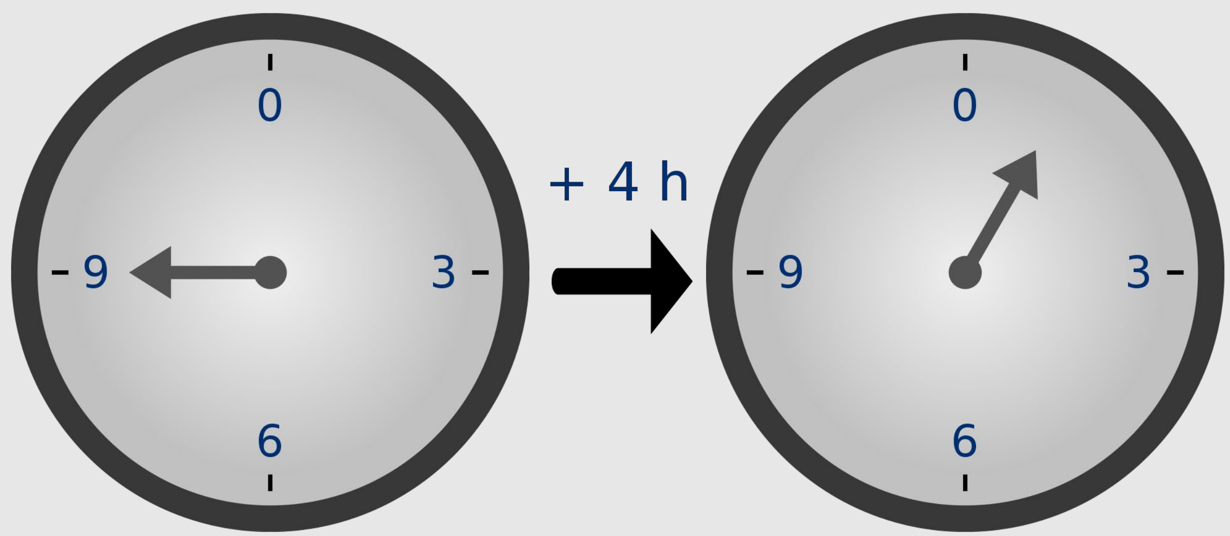

Alice
Bob
Common paint
+
+
Secret colours
=
=
Public transport
(assume
that mixture separation
is expensive)


Alice
Bob
Common paint
+
Secret colours
=
Public transport
(assume
that mixture separation
is expensive)
+
Secret colours
=
Common secret


Alice
Bob
Common paint
+
?
Secret colours
=
Public transport
(assume
that mixture separation
is expensive)
Secret colours
Common secret

# Math Background: Cyclic Group

- **Group:** a set of numbers with an operation (addition, or multiplication)
  - ✓ Example: set = [0, 11], operation = addition; (9+4) *mod* 12 = 1




- **$Z^*_p$:** Multiplicative group modulo *p*
  - ✓ set = [1, *p*-1], operation = multiplication; *a*b mod p*

# Math Background: Cyclic Group

- ***g*** is a <u>generator</u> of ***mod p*** if every element of **[1*, p-1*]** can be written as

$$g^x \bmod p$$

- There is at least one *g* (<u>primitive element</u>) that generates the *entire group*.
  - ✓ *q* is <u>order of *g*</u> if $g^q = 1 \bmod p$ (means *g* can generate *q* elements of the group)
  - ✓ g is <u>primitive element</u> if $q = p-1$ (means *g* can generate all elements of the group)

- Example: $g = 2, p = 11$

| | |
|---|---|
| $2^0 \bmod 11 = 1$ | $2^5 \bmod 11 = 10$ |
| $2^1 \bmod 11 = 2$ | $2^6 \bmod 11 = 9$ |
| $2^2 \bmod 11 = 4$ | $2^7 \bmod 11 = 7$ |
| $2^3 \bmod 11 = 8$ | $2^8 \bmod 11 = 3$ |
| $2^4 \bmod 11 = 5$ | $2^9 \bmod 11 = 6$ |

- **Is *g = 2* a primitive element?**
- **If *g = 4, q = ?***
- **If *g = 5, q = ?***

- <u>Subgroup</u> = [1, 3, 4, 5, 9], *g = ?*

# Discrete Logarithm Problem (DLP)

- Discrete Logarithm Problem (DLP):

  for known ***Y*****,*g, p*** find ***X*** such that: $Y = g^X \bmod p$

- Examples: $g = 2,\ p = 13$

  $2 = 2^X \bmod 13$ $\qquad X = 1$

  $3 = 2^X \bmod 13$ $\qquad X = 4$

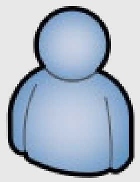
  $4 = 2^X \bmod 13$ $\qquad X = 2$

  $5 = 2^X \bmod 13$ $\qquad X = 9$

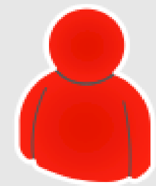
- Difficult (secure) when *p* is a large prime (e.g., 2048 bits)

21435120827721043063114917062790527573328193653502702369166196362676514731108527945946901215887590463048823428151199854
28892042604427608335711847366885192193296128232974167042736105925970485551575408786146057302507914866994805958463029863
67423150776760586541931852829272503569987859584155758818414110310938806580866330674698300811397645221051701085628555581
39043580800539734898746108361004674150661832306964399024263472249734260526991394535358856194229841900239384394337166360
04634473477960016553086587936214475293986333099769703657851952708437791021602574554141661123790470681951139502943964009
4554495074110424652379

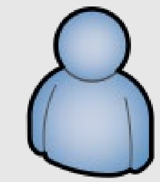
# DH Protocol



**Alice**



**Eve**



**Bob**

Publicly known parameters: *g, p* (large prime)

Random secret *a*

$$\xrightarrow{\quad g^a \bmod p \quad}$$

Random secret *b*

$$\xleftarrow{\quad g^b \bmod p \quad}$$

$K = (g^b)^a \bmod p$

$K = (g^a)^b \bmod p$

# Properties

- Parameters can be sent by Alice (don't have to be hardcoded).
  - ✓ Bob needs to check $p$ is a <u>safe prime</u>: large and in the form of $\boldsymbol{p = 2q + 1}$ where $q$ is also a prime.
- DH problem: Eve has to compute K with $g^a \bmod p$ and $g^b \bmod p$.
  - ✓ If she can solve DLP then it is trivial to compute K.
  - ✓ At least as easy as DLP. Can it be easier than solving DLP?
- Efficiency
  - ✓ $g^{p-1} \bmod p = 1$, thus $g^a \bmod p = g^{(a \bmod p-1)} \bmod p$
  - ✓ easy for $g = 2$ (can express other generators as $2^x$)

# Security

- Key and parameters sizes

| Date | Symmetric | Factoring Modulus | Discrete Logarithm | | Elliptic Curve | Hash | |
|---|---|---|---|---|---|---|---|
| | | | Key | Group | | | |
| 2017 - 2022 | 128 | 2000 | 250 | 2000 | 250 | SHA-256<br>SHA-512/256<br>SHA-384<br>SHA-512 | SHA3-256<br>SHA3-384<br>SHA3-512 |
| > 2022 | 128 | 3000 | 250 | 3000 | 250 | SHA-256<br>SHA-512/256<br>SHA-384<br>SHA-512 | SHA3-256<br>SHA3-384<br>SHA3-512 |

- The protocol is <u>unauthenticated</u>.

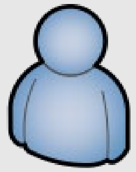
  ✓ Secure only against passive adversaries.

  ✓ Eve can impersonate Alice to Bob and Bob to Alice (MITM).

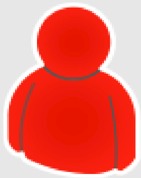
# Authenticated DH

- One extra (final) message.
- The messages are signed (except the first one).
- The parameters *g, p* are not fixed (just sent by Alice).

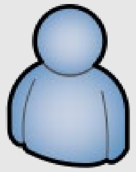
**Is it secure?**



**Alice**



**Eve**



**Bob**

Select $g$, $p$
Random $a$
$A = g^a \bmod p$

$\xrightarrow{\text{Alice}, g, p, A}$

Random $b$
$B = g^b \bmod p$

$\xleftarrow{\text{Bob}, B, \text{Sign}_{\text{Bob}}(\text{Alice}, g, p, A, B)}$

Verify signature

$\xrightarrow{\text{Sign}_{\text{Alice}}(\text{Bob}, g, p, A, B)}$

Verify signature

$K = (B)^a \bmod p$

$K = (A)^b \bmod p$

# RSA

# Math Background: CRT

- $n = pq$ where $p$ and $q$ are different primes**.**

- For any given a = $x$ *mod* $p$ and b = $x$ *mod* $q$, **1)** $x$ can be reconstructed, and **2)** there is unique solution of $x$ in [$0$, $n$-$1$] ($Z_n$).

**2)** *Proof.* Suppose $x' \neq x$ is also a solution. Let $d = x - x' > 0$.

- ✓ $d \bmod p = (x - x') \bmod p = x \bmod p - x' \bmod p = a - a = 0$ → $d$ is a multiple of $p$. For the same reason, $d$ is also a multiple of $q$.
- ✓ Then $d$ is a multiple of $lcm(p,q)$.
- ✓ $p$ and $q$ are different primes. → $lcm(p,q) = pq = n$ → $d = x - x'$ is a multiple of $n$.
- ✓ Both $x$ and $x'$ are in [$0$, $n$-$1$] → $x - x'$ is a multiple of $n$ in [$0$, $n$-$1$].
- ✓ There is only one solution: $x = x'$.

# Math Background: CRT

- $n = pq$ where $p$ and $q$ are different primes.
- For any given a = $x \bmod p$ and b = $x \bmod q$, **1)** $x$ can be reconstructed, and **2)** there is only one solution of $x$ in $[0, n\text{-}1]$ ($Z_n$).

**1)** <u>Garner's Formula</u>: $x = (((a - b)(q^{-1} \bmod p)) \bmod p)\, q + b$

✓ $x \bmod q = ((((a - b)(q^{-1} \bmod p)) \bmod p)\, q + b) \bmod q$

$= (Kq + b) \bmod q$, for some $K$

$= b \bmod q$

$= b$

✓ $x \bmod p = ((((a - b)(q^{-1} \bmod p)) \bmod p)\, q + b) \bmod p$

$= (((a - b)(q^{-1})\, q + b) \bmod p$

$= a \bmod p$

$= a$

# Public-Key Encryption

- **Gen()**
  - ✓ return a key pair (i.e., public and private key).
- **Enc(pub_key, msg)**
  - ✓ Encrypt a message using a public key.
  - ✓ Return a ciphertext.
- **Dec(priv_key, ctxt)**
  - ✓ Decrypt a ciphertext using a private key.
  - ✓ Return a message.

# RSA Encryption

- **Gen()**
  - ✓ Select (large) random prime numbers $p, q$ ($p \neq q$, but with almost equal size)
  - ✓ Compute modulus $n = pq$
  - ✓ Compute $\Phi = (p-1)(q-1)$
  - ✓ Select public exponent $e$, $1 < e < \Phi$, such that $gcd(e, \Phi) = 1$
  - ✓ Compute private exponent $d = e^{-1} \bmod \Phi$
  - ✓ Return public key ($n$, $\mathbf{e}$), and private key (p, q, $\Phi$, $\mathbf{d}$)
- **Enc(*e, m*)**
  - ✓ Return $m^e \bmod n = c$
- **Dec(*d, c*)**
  - ✓ Return $c^d \bmod n = m$

# Digital Signature

- **Gen()**
  - ✓ Return a key pair (i.e., public and private key).
- **Sign(priv_key, msg)**
  - ✓ Sign the message using the private key.
  - ✓ Return the signature.
- **Verify(pub_key, msg, sign)**
  - ✓ Verify the signature of the message, using the public key.
  - ✓ Return *Boolean* (true/false).

# RSA Signature

- **Gen()** (the same as in encryption)
  - ✓ Select (large) random prime numbers $p, q$ ($p \neq q$, but with almost equal size)
  - ✓ Compute modulus $n = pq$
  - ✓ Compute $\Phi = (p-1)(q-1)$
  - ✓ Select public exponent $e$, $1 < e < \Phi$, such that $gcd(e, \Phi) = 1$
  - ✓ Compute private exponent $d = e^{-1} \bmod \Phi$
  - ✓ Return public key ($n, e$), and private key (p, q, $\Phi, d$)
- **Sign($d, m$)**
  - ✓ Return $H(m)^d \bmod n = \sigma$
- **Verify($e, m, \sigma$)**
  - ✓ Return $\sigma^e \bmod n == H(m)$ ?
- **Why do we need to hash *m* before signing?**

# Properties

- Factorization Problem
  - ✓ Compute $m$ given $(n,e)$ and $c = m^e \bmod n$.
  - ✓ At least as easy as integer factorization of $n$. Can it be easier?

- Efficiency
  - ✓ Choose small value for $e$ (3 or 5), more efficient for signature verification (multiple times).
  - ✓ Use CRT to compute $m = c^d \bmod n$, can save computing with a factor of 4.
    - Compute CRT representation ($m_a = c^d \bmod p$, $m_b = c^d \bmod q$).
    - Use Garner's formula to compute $m$ from $m_a$ and $m_b$.

# Properties

- Encryption

✓ $e$ is usually small to speed up computations.

  – Be careful with encrypting a very small message.

  – If $m^e < n$, there is no modular reduction. Attack can recover $m$ by simply taking the $e$-th root of $m^e$.

✓ RSA encryption is expensive.

  – Typical application is $E_{RSA}(K)$, $E_K(m)$.

# Security

- Do not use the same key pair for encrypting and signing.
  - ✓ Signing "message" $c$ is the same operation as decrypting the ciphertext $c$.
- $n$ should be $\geq$ 2048 bits.

| Date | Symmetric | Factoring Modulus | Discrete Logarithm Key | Discrete Logarithm Group | Elliptic Curve | Hash | |
|---|---|---|---|---|---|---|---|
| 2017 - 2022 | 128 | 2000 | 250 | 2000 | 250 | SHA-256<br>SHA-512/256<br>SHA-384<br>SHA-512 | SHA3-256<br>SHA3-384<br>SHA3-512 |
| > 2022 | 128 | 3000 | 250 | 3000 | 250 | SHA-256<br>SHA-512/256<br>SHA-384<br>SHA-512 | SHA3-256<br>SHA3-384<br>SHA3-512 |

- p and q should be of equal size.
- Small $d$ is insecure. (Small $e$ is OK.)

# Key Points

- Foundation of public-key crypto:
  - ✓ DLP – one-way function (e.g., DH protocol)
  - ✓ Factorization problem – *trapdoor* one-way function (e.g., RSA)
- Key applications of PKC:
  - ✓ Key establishment (e.g., DH), or key distribution (e.g., RSA encryption)
  - ✓ Digital signature (e.g., RSA signature)

# Exercises & Reading

- Classwork (Exercise Sheet 10): due on Fri Nov 16, 10:00 PM
- Homework (Exercise Sheet 10): due on Fri Nov 23, 6:59 PM
- Reading: FSK [Ch10, Ch11, Ch12]

# End of Slides for Week 10