Authentication and Key Establishment Protocols

Outline

- Network security protocols
- Common attacks
- Security Assumptions
- Remote Identification/Authentication
- (Authenticated) Key Establishment
- AKE examples
- Password-based AKE
- Group Key Establishment

Network security protocols

- An interactive algorithm executed by two or multiple parties over an insecure network (e.g., the Internet).
- Examples:
  - Identification
  - Key exchange
  - E-voting
  - E-payment
  - E-auction
  - ……

Common attacks

- Eavesdropping attack
  - The attacker captures the information sent in the protocol.
- Man-in-the-middle attack
  - The attacker alters the information sent in the protocol.
- Replay attack
  - The adversary records information seen in the protocol, and then sends it to the same, or a different, entity, possibly during a later protocol run.
- Reflection attack
  - The adversary sends protocol messages back to the entity who sent them
- Known-key attack
  - The adversary obtains the key of one communication session, and uses it to attack another session
- ……

Security Assumptions

- **Assumption 1**
  The adversary is able to eavesdrop, modify, re-route, insert messages during the execution of a protocol.
- **Assumption 2**
  The adversary may be a legitimate protocol participant (an insider), or an external party (an outsider), or a combination of both.

Security Assumptions

- **Assumption 3**
  The adversary is able to compromise some past communication sessions
- **Assumption 4**
  The adversary may start any number of parallel protocol runs between any parties including different runs involving the same parties.

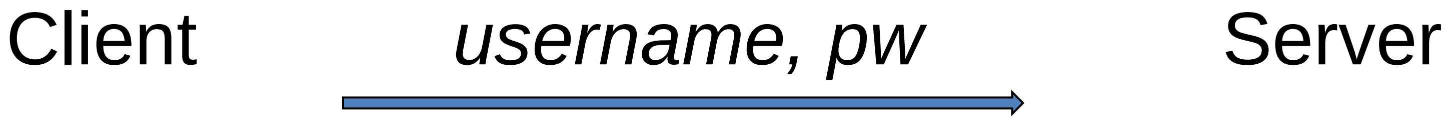
Entity Authentication

- Entity authentication - Definition
  - … is the process whereby one party is assured (*through acquisition of corroborative evidence*) of the identity of a second party involved in a protocol, and that the second has actually participated (i.e., *is active at, or immediately prior to, the time the evidence is acquired*).

--- Handbook of applied cryptography (Menezes et al.)

A Simple Example

- How to remotely login a server
- A naïve approach:
  username + password



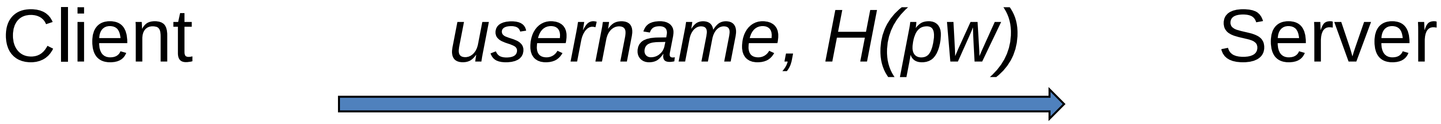
Client

username, pw

Server

Can you identify some security risks in this approach?

An Improve Scheme

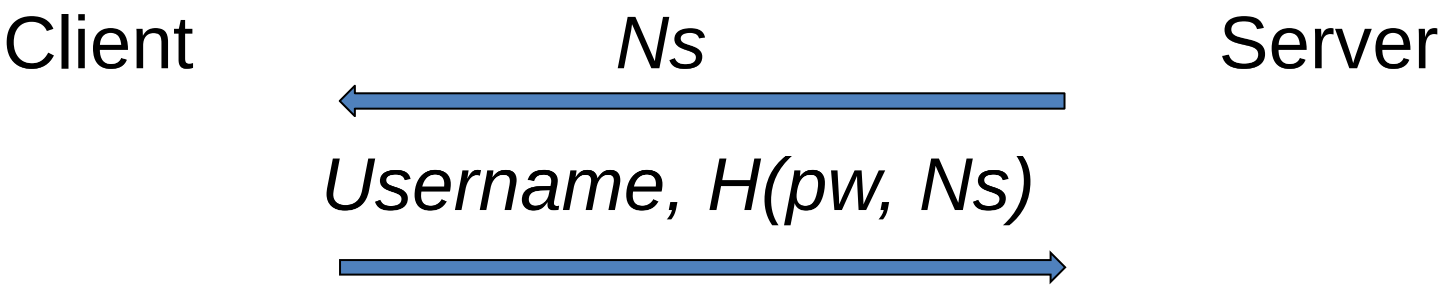


Client *username, H(pw)* Server

- Use a transformed password, e.g., H(pw) where H denotes a one-way cryptographic hash function

- Is this approach secure?
  - – No. A replay attack can still work
  - – Need an anti-replay mechanism

Improved Scheme II



Client *Ns* Server

Username, H(pw, Ns)

- The server sends a ***nonce*** Ns to the client as a challenge
  - Similar as a salt value in Unix
- The client gives a ***fresh*** response based on pw and Ns in each session

Entity Authentication Approaches

- Password based
- Token based
- Biometric based

Covered in CSCI262 – System Security

- Public key crypto based
  - Digital signature

Notations

- A – Alice
- B – Bob
- E – Eve
- E_A : E impersonating A
- i : step i of a protocol session
- i’ : step i of a concurrent/parallel protocol session

Entity Authentication using Public Key Crypto

- Step 0: users exchange and verify public key certificates
- First try - Protocol I

  B → A: “I’m Bob", Sig_B(“I’m Bob”)

  Any security issue?
- 1: B → A: “I’m Bob", Sig_B(“I’m Bob”)

  1’:E_B → A: “I’m Bob", Sig_B(“I’m Bob”)

  A *replay* attack

Entity Authentication

- A revised one – Protocol II

  1: A → B: “I’m Alice”, N_A

  2: B → A: “I’m Bob", Sig_B(“I’m Bob”, N_A)

N_A (nonce) must not repeat

Entity Authentication

- Another solution - Protocol III

  1: B → A: “I’m Bob", Sig_B(“I’m Bob”, T_B)

  T_B : a timestamp

- The protocol achieves entity authentication

Key Establishment

- Key Establishment – Definition
  - … is a process or protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use.

--- Handbook of applied cryptography (Menezes et al.)

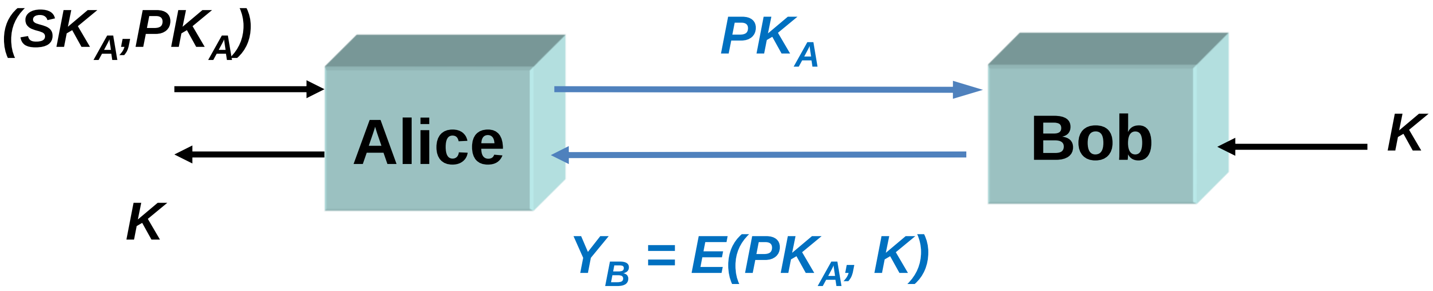
Key Establishment Goals

- The shared session key is a good key for A to use with B only if A has assurance that:
  - – The key is fresh (**key freshness**);
  - – The key is known only to A and B (and any mutually trusted parties) (**Key authentication**)
    - Key authentication – confidentiality of the session key

Key Establishment Protocols

- There are two main categories of key establishment:
  - *Key transport*, where one party generates and securely transfers the key to the other.
  - *Key agreement/exchange*, where the parties all obtain a shared secret, which is itself a function of input from all parties.
    - E.g. Diffie-Hellman key agreement.

Key Transport

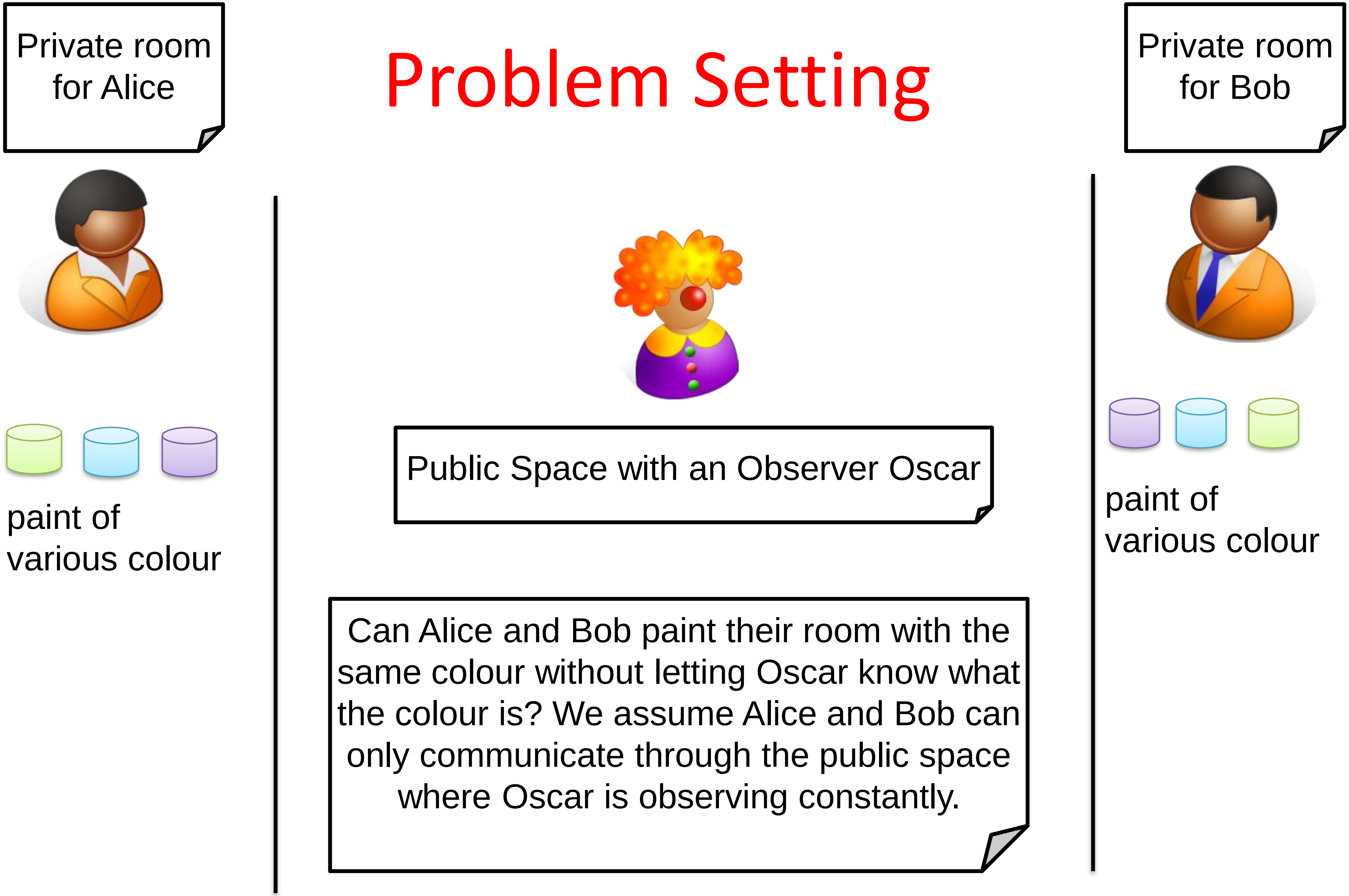


E: public key encryption, e.g., RSA

Diffie-Hellman Key Exchange

- An interesting explanation from Wikipedia
- Can two parties come up with the same paint color without letting an observer know what is the resulting color?

Problem Setting



Private room for Alice

paint of various colour

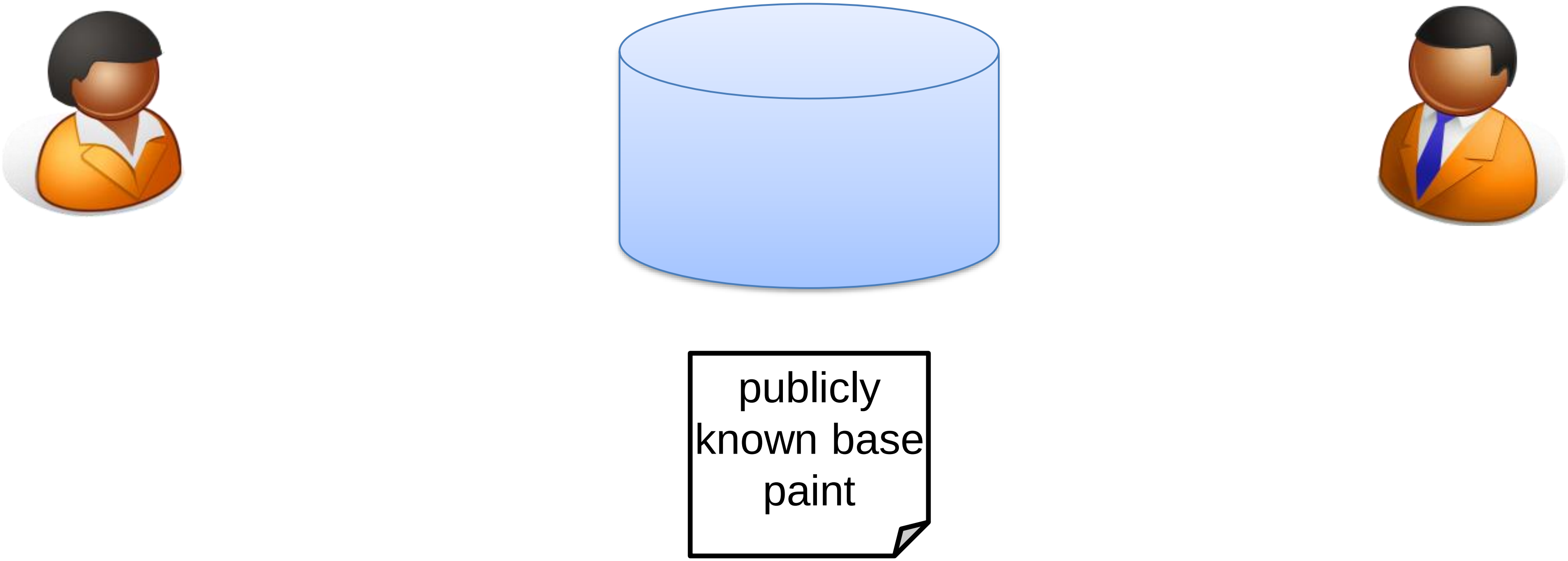
Public Space with an Observer Oscar

Can Alice and Bob paint their room with the same colour without letting Oscar know what the colour is? We assume Alice and Bob can only communicate through the public space where Oscar is observing constantly.

Private room for Bob

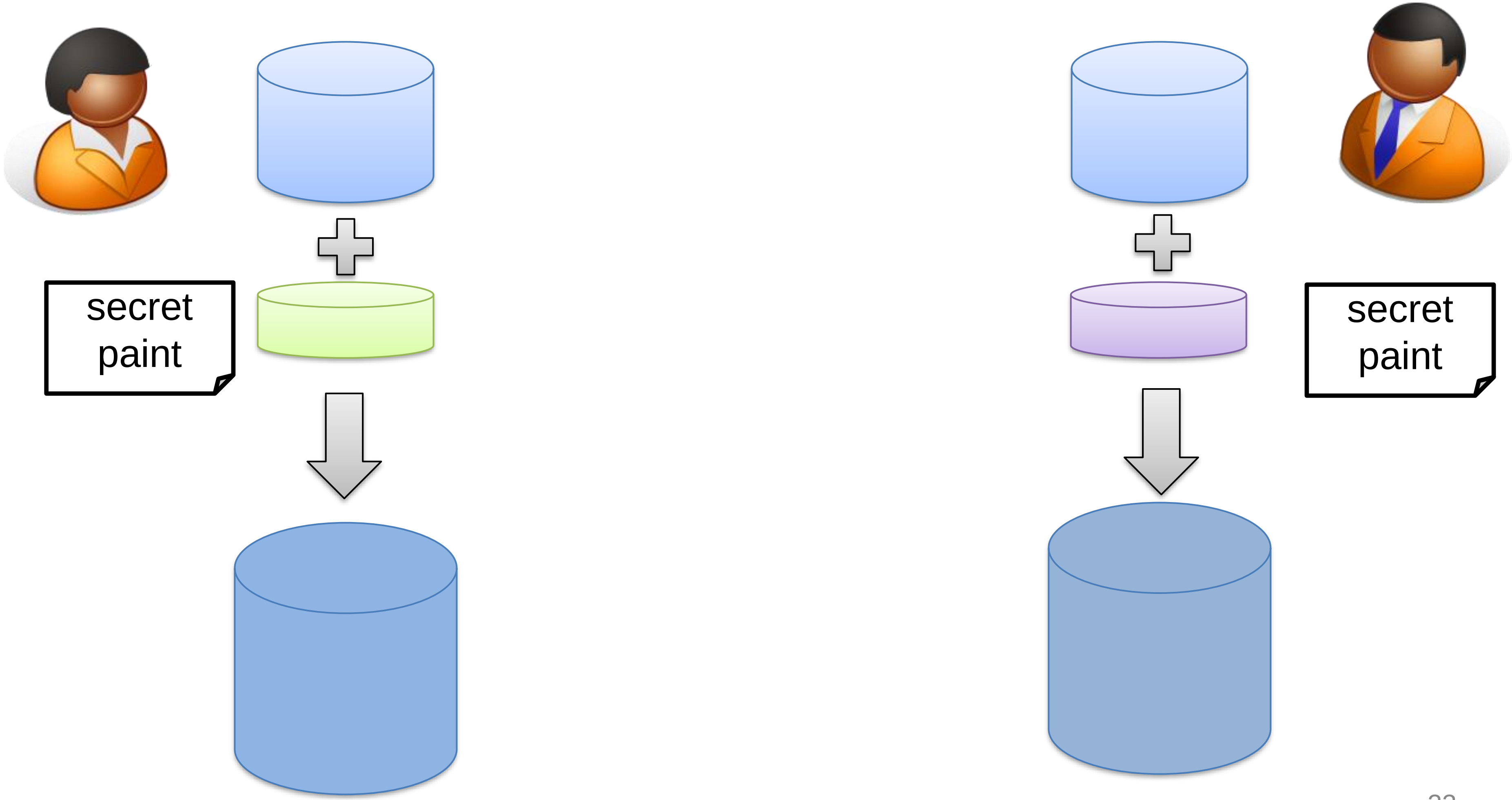
paint of various colour

Common Colour



publicly known base paint

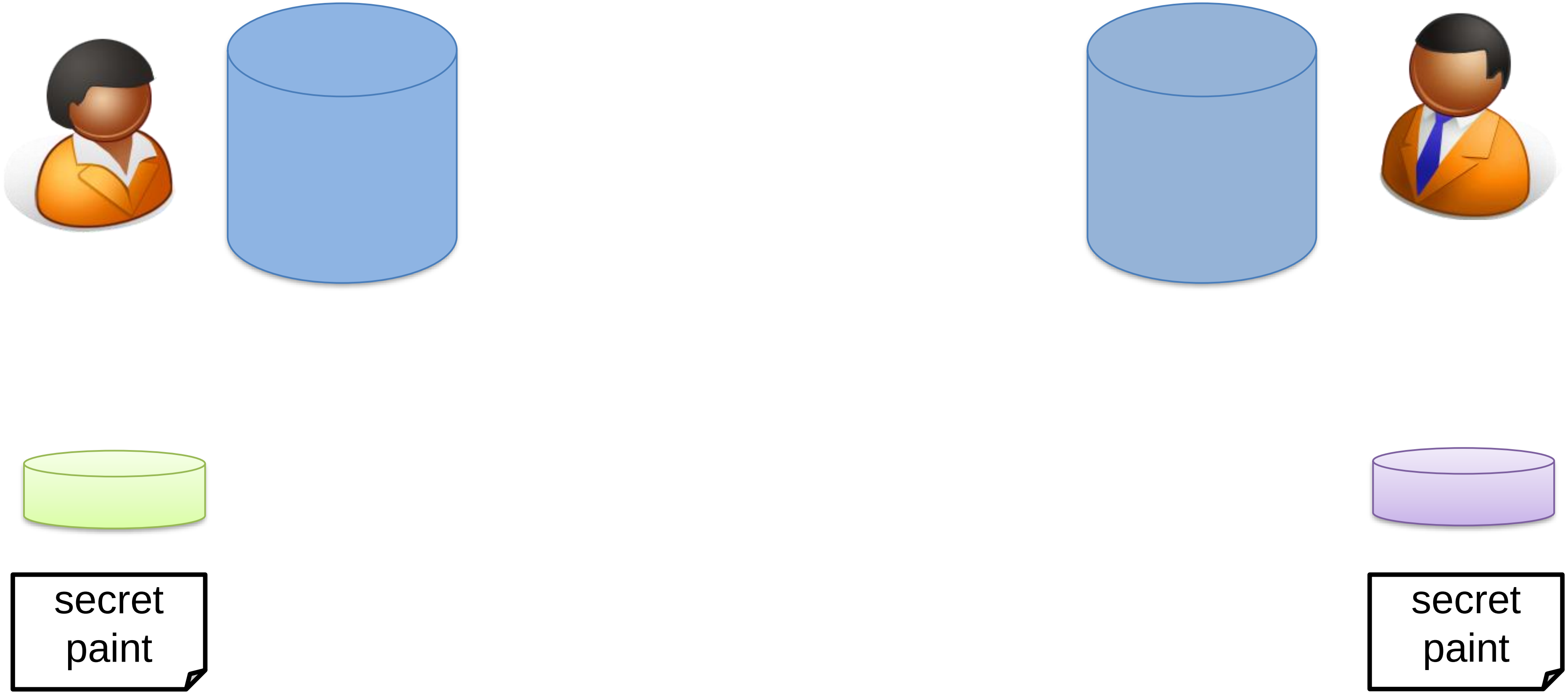
Private Mixing



secret paint

secret paint

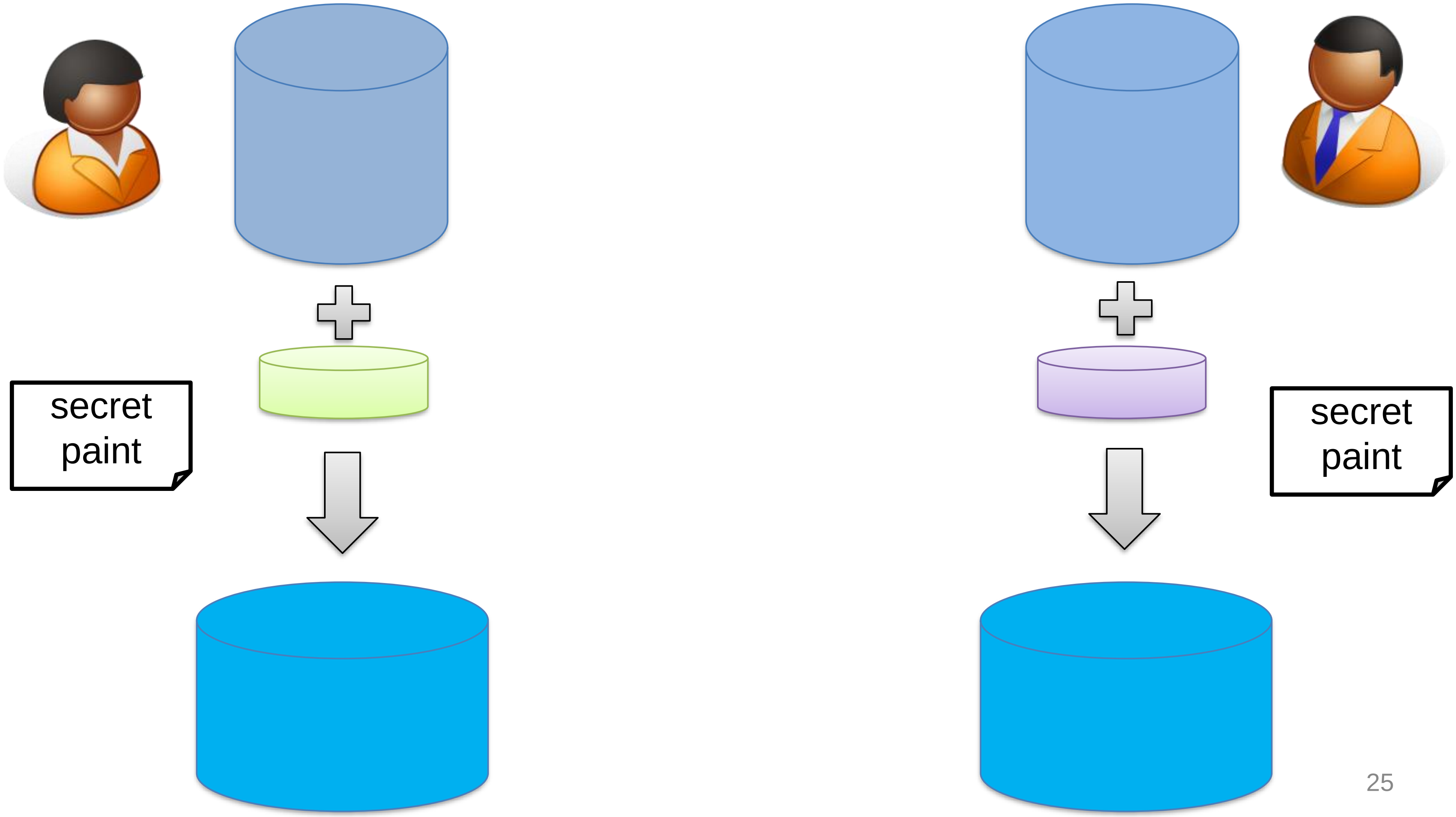
Paint Exchange



secret paint

secret paint

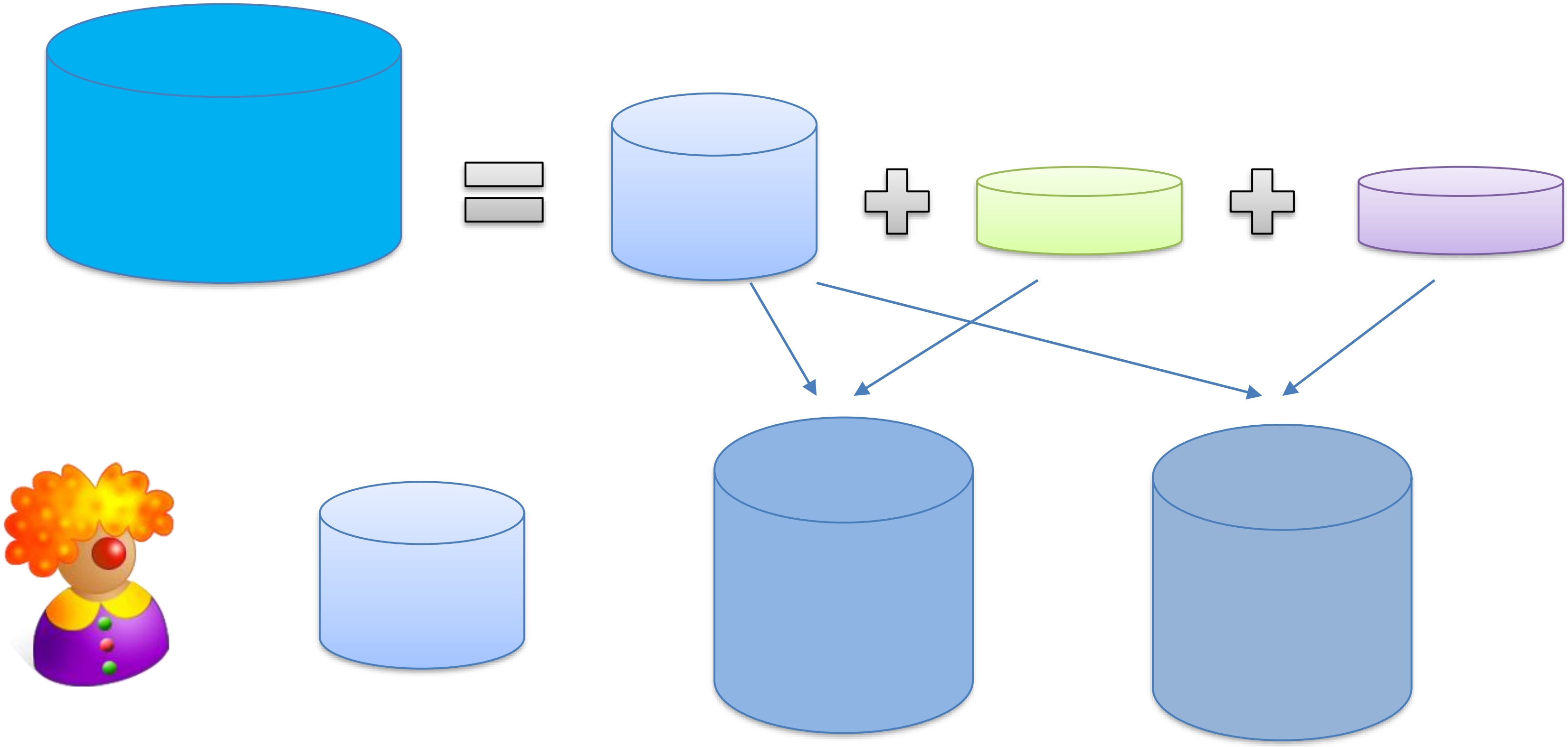
Create Secret Colour



secret paint

secret paint

The Final Colour



Diffie-Hellman Key Agreement

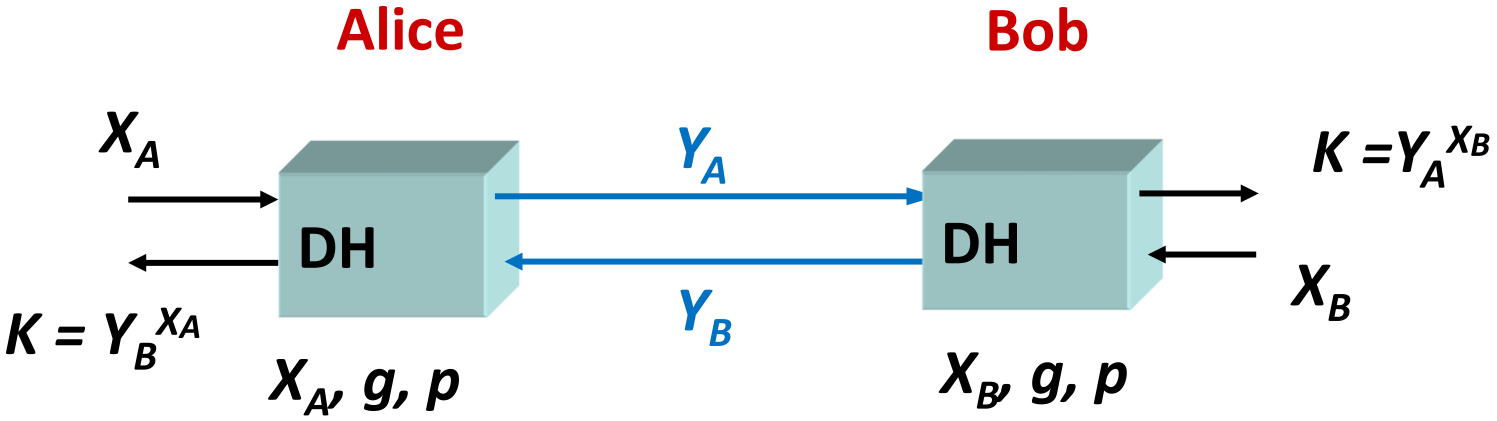
- The first public-key system (1976).
- Security is based on the difficulty of computing discrete logarithm.
- System Setup
  - $Z^*_p = \{1, \ldots, p-1\}$.
  - A generator $g \in Z^*_p$.
  - p and g are public.

Diffie-Hellman Key Agreement

- The Protocol:
  - Alice selects a secret X_A, for $X_A \in \mathrm{Z}_\mathrm{p}$, and computes her public key $Y_A = g^{X_A}$ mod p.
  - Bob selects a secret X_B, for $X_B \in \mathrm{Z}_\mathrm{p}$, and computes his public key $Y_B = g^{X_B}$ mod p.
  - Alice sends Y_A to Bob.
  - Bob sends Y_B to Alice.
  - Alice computes the shared secret key K = $Y_B^{X_A}$ mod p.
  - Bob computes the shared secret key K = $Y_A^{X_B}$ mod p.

Diffie-Hellman Key Agreement

- The Protocol

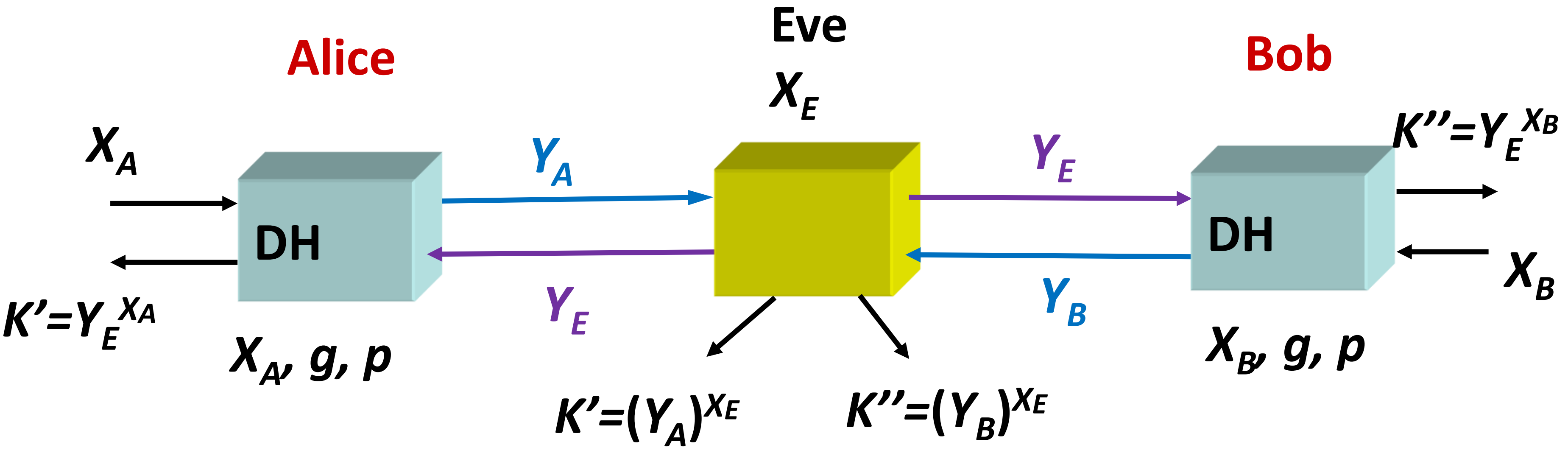


$$1\text{:}\ \ \text{A} \rightarrow \text{B:}\ \ Y_A = g^{XA} \, mod\ p$$

$$2\text{:}\ \ \text{B} \rightarrow \text{A:}\ \ Y_B = g^{XB} \, mod\ p$$

Diffie-Hellman Key Agreement

- **Man-in-the-Middle Attack**



1: A → E: Y_A

2: E → B: Y_E

3: B → E: Y_B

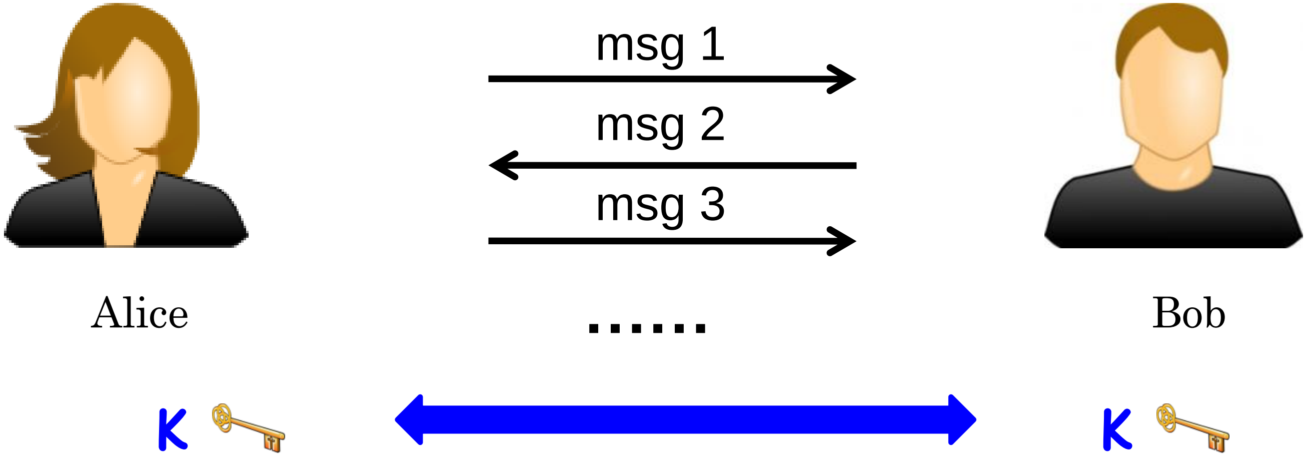
4: E → A: Y_E

How to solve the problem?

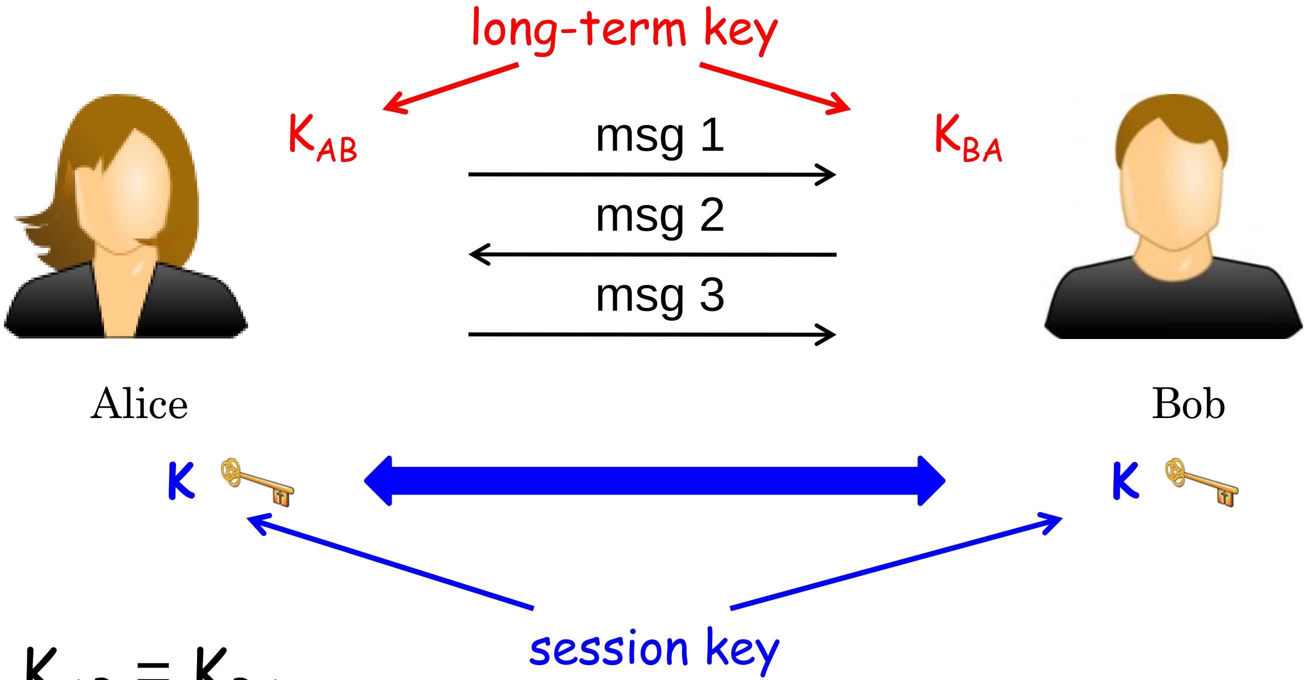
(hint: authentication)

Authenticated Key Establishment (AKE) Protocol

- Mechanisms that allow two parties (or multiple parties) communicating over an insecure network to authenticate each other, and establish a common secret key.



Symmetric-key Based AKE



long-term key

K_{AB}

K_{BA}

msg 1

msg 2

msg 3

Alice

Bob

K

K

session key

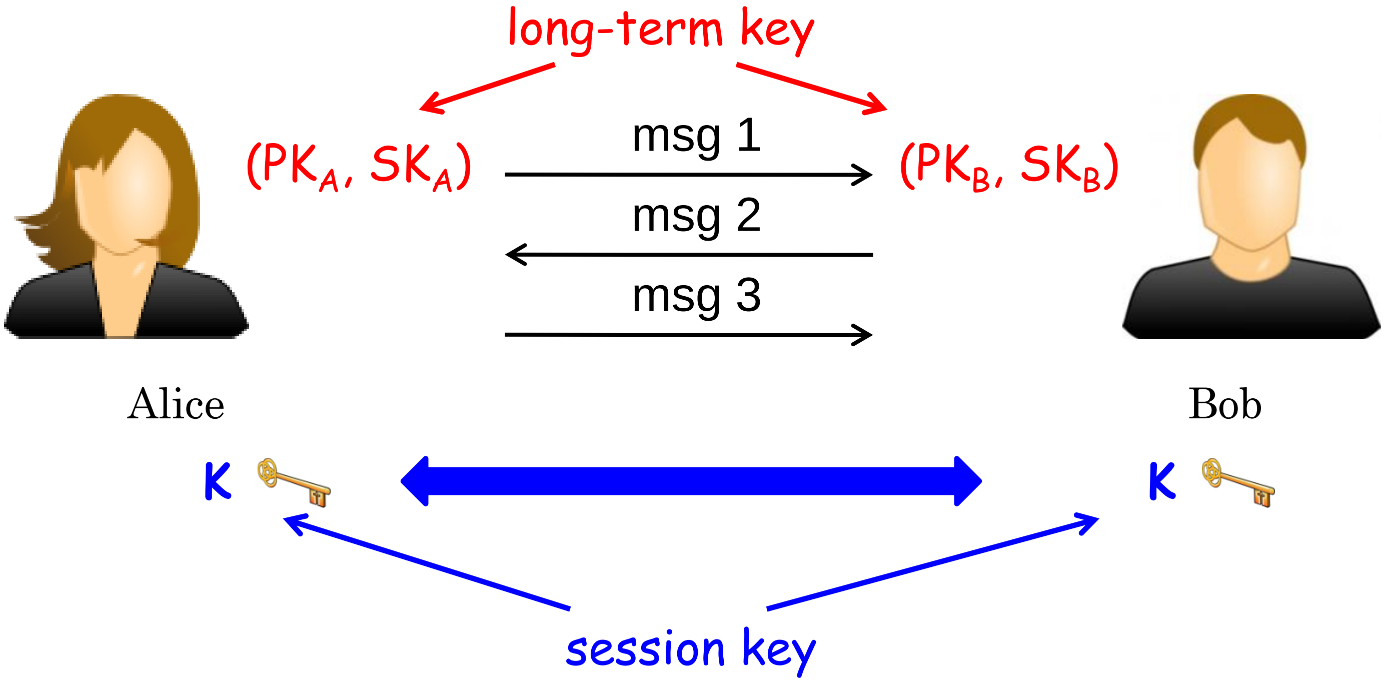
- $K_{AB} = K_{BA}$
- Established through another channel
- Can be derived from a shared password

Question: why not simply use the long-term key as the session key?

Why session keys

- They ….:
- … limit the amount of ciphertext available under a single key.
- … limit the effects, in both time and data quantity, of a key being exposed or compromised.
- … create independence across sessions, and/or applications. Compromise of a session due to use of a poor application doesn’t, or ideally shouldn’t, affect the security of other sessions and applications.

Public-key Based AKE



- Public key is certified by a trusted CA
- Distribution of long-term key is easy
  - Digital Cert (i.e., public key) can be sent as part of a message

How about this one?

1: $A \rightarrow B: A, Y_A$

2: $B \rightarrow A: B, Y_B, Sig_B(Y_B, Y_A)$

3: $A \rightarrow B: Sig_A(Y_A, Y_B)$

- Y_Q: Diffie-Hellman component of Q
  - $Y_Q = g^{X_Q} \bmod p$
- Sig_Q: signature of Q (Assume that A has B's certificate, and vice versa)

An unknown key share attack

1: $A \rightarrow E_B$: A, Y_A

1’: $E \rightarrow B$: E, Y_A

2’: $B \rightarrow E$: B, Y_B, $Sig_B(Y_B, Y_A)$

2: $E_B \rightarrow A$: B, Y_B, $Sig_B(Y_B, Y_A)$

3: $A \rightarrow E_B$: $Sig_A(Y_A, Y_B)$

3’: $E \rightarrow B$: $Sig_E(Y_A, Y_B)$

Consequence: A believes that she has agreed on a key with B, but B believes that he has agreed on a key with E

A Consequence of UKS Attack

- E doesn’t know the shared key K, but it created an “identity mis-binding”:
  - A thinks it is talking to B;
  - B thinks it is talking to E.
- Drone example:
  - A: fighter jet; B: central command
  - E: drone (controlled by enemy)

B to E: “destroy yourself” (encrypted with K)

E to A: “destroy yourself” (encrypted with K)

Result: fighter jet destroys itself.

Strong Entity Authentication

- **Strong entity authentication** of B to A is provided if A has a *fresh* assurance that B has knowledge of A as his peer entity.
- Protocol IV

1: A → B: “I’m Alice”, N_A

2: B → A: “I’m Bob", Sig_B(“I’m Bob”, A, N_A)

Diffie-Hellman Revisited

1: $A \rightarrow B$: A, Y_A

2: $B \rightarrow A$: B, Y_B, $\mathrm{Sig}_B(A, Y_B, Y_A)$,

3: $A \rightarrow B$: $\mathrm{Sig}_A(B, Y_A, Y_B)$

Shared key K = $Y_A^{X_B} = Y_B^{X_A}$ (mod p)

- ISO/IEC IS 9798-3

- **Forward secrecy**: compromising the long-term keys (i.e., the signing keys of A and B) does not allow the adversary to compute any *old* session keys generated by those parties.

Forward Secrecy

1: A → B: $E_{PK_B}(A, B, K_A)$

2: B → A: $E_{PK_A}(B, A, K_B)$

Shared key $K = Hash(K_A,K_B)$

No forward secrecy!

Password-based Protocols

- Long-term secret key = shared password
- Consider the following protocol

  1: A → B: A, E_{P_AB} (A,B,R_A)

  2: B → A: B, E_{P_AB} (B,A,R_B)

  Session key: H(R_A,R_B)

- This approach suffers **off-line dictionary attack**, since passwords are short strings with low entropy. That is, an attacker can try each possible password *P'* to decrypt E_{P_AB} (A,B,R_A). If the resulting plaintext has the correct format, *P'* is likely the correct password.

Password-based Protocols

- The following **Encrypted Key Exchange (EKE) protocol** can resist the off-line dictionary attack:

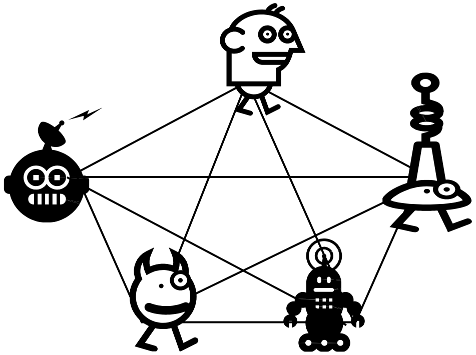
$$\begin{array}{llll} 1. & A \longrightarrow B: & & E_P(PK) \\ 2. & B \longrightarrow A: & & E_P(Enc_{PK}(K)) \end{array}$$

$$\underline{\text{Output}} : K \ \text{(Session key)}.$$

- *PK* is **an ephemeral public key** generated by *A*.
- *B* transfers *K* to A by using double encryption.
- Why is the EKE protocol immune to the off-line dictionary attack?

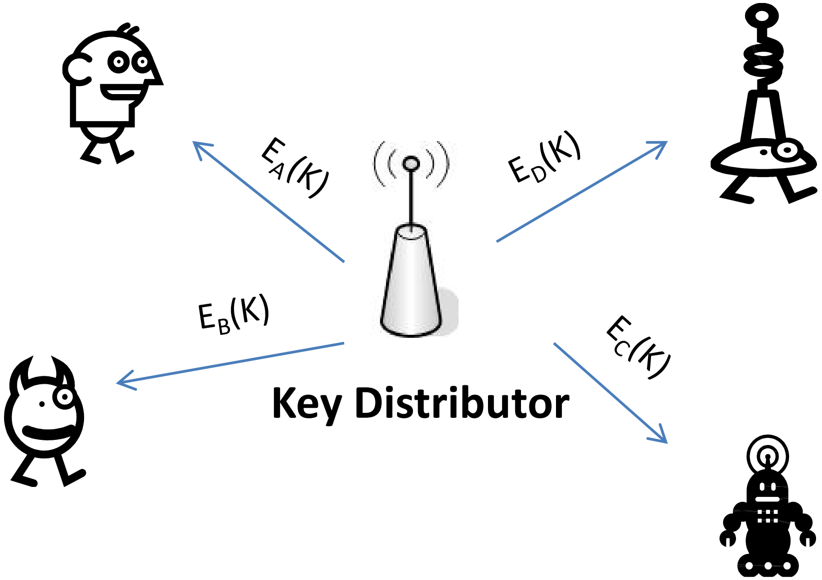
Group Key Establishment for Distributed Systems

- Allow a group of users to establish a common secret key over an insecure network
  - – Setup a broadcast channel
- (a.k.a) Conference key establishment



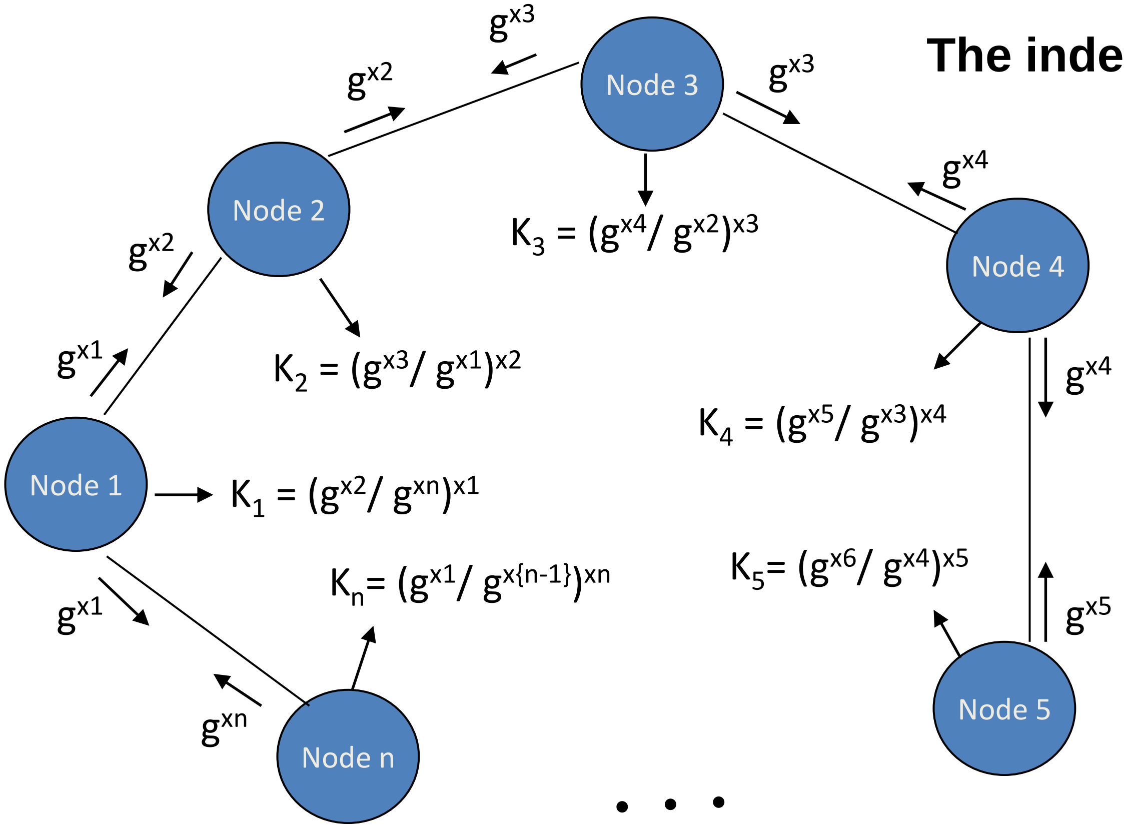
Group Key Transport

- The key distributor randomly selects a key K in the key space
- The key distributor encrypts the key K under each party’s long-term key and distributes the encrypted messages to the corresponding parties
- Each member decrypts the message for him/her and gets back the key K



Group Key Agreement

The indexes are taken in a cycle.


Node 1
Node 2
Node 3
Node 4
Node 5
Node n
g^{x1}
g^{x2}
g^{x3}
g^{x4}
g^{x5}
g^{xn}
$K_1 = (g^{x2}/ g^{xn})^{x1}$
$K_2 = (g^{x3}/ g^{x1})^{x2}$
$K_3 = (g^{x4}/ g^{x2})^{x3}$
$K_4 = (g^{x5}/ g^{x3})^{x4}$
$K_5= (g^{x6}/ g^{x4})^{x5}$
$K_n= (g^{x1}/ g^{x\{n-1\}})^{xn}$
• • •

1). Each party U_i $(i = 1, 2, \cdots)$ broadcasts $k_i = g^{x_i} \bmod p$.
2). Each party U_i broadcasts $K_i = (k_{i+1}/k_{i-1})^{x_i} \bmod p$.
3). Each party U_i computes the session key K by

$$K = k_{i-1}^{nx_i} \cdot K_i^{n-1} \cdot K_{i+1}^{n-2} \cdots K_{i-2} \bmod p.$$

Output: $K = g^{x_1x_2+x_2x_3+\cdots+x_nx_1} \bmod p$.