Public Key Cryptography & Key Exchange

Sushmita Ruj

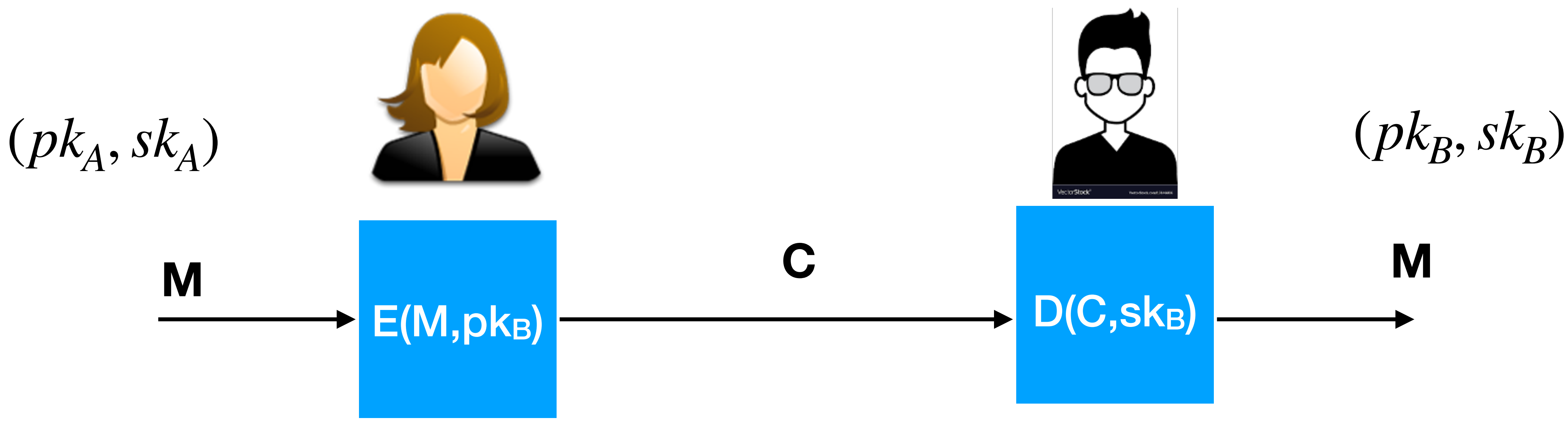
Recap

- Trapdoor Permutations
- Textbook RSA
- RSA algorithm
- Attack on Textbook RSA

This Lecture

- Key agreement
- Diffie Hellman Key Exchange
- Discrete Logarithm Problem
- Key Derivation Function
- El Gamal Encryption algorithm

Public Key Cryptography



Alice wants to send a message to Bob secretly

- $\varepsilon = (\mathscr{M}, \mathscr{C}, \mathscr{K})$
- $KG(1^k) \rightarrow (pk_A, sk_A), (pk_B, sk_B), \ldots$
- $For\ \ m \in \mathscr{M}, E(m, pk_B) \rightarrow c$

Public key of Bob known to everyone
E is a randomised algorithm

- $D(c, sk_B) \rightarrow m'$

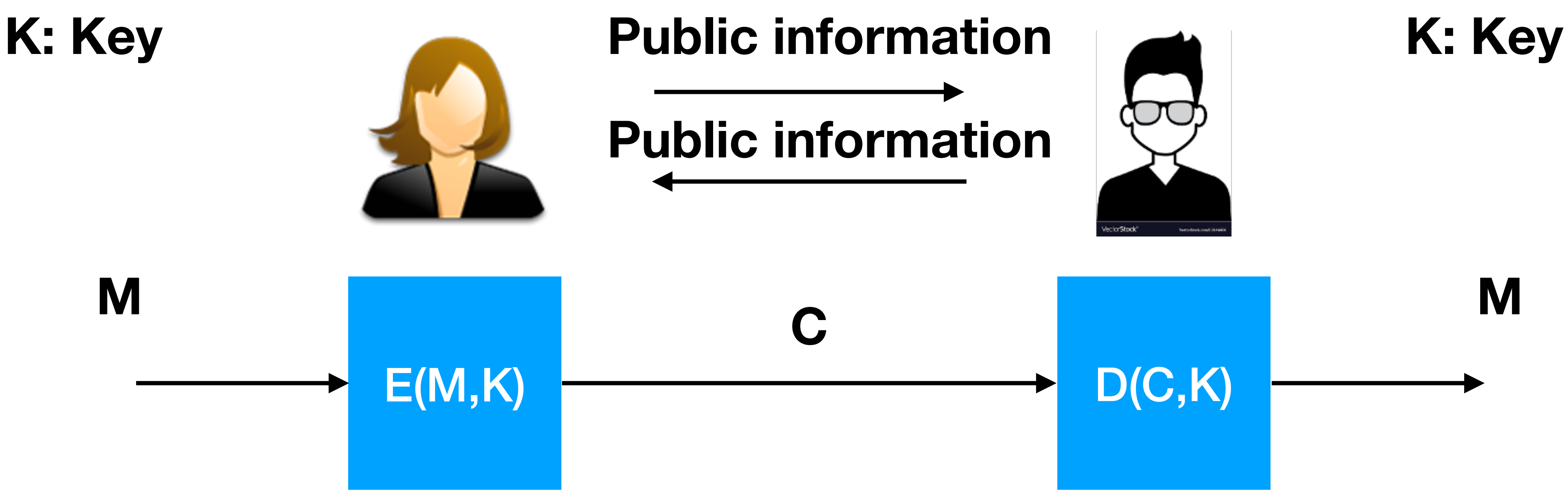
Secret key known only to Bob, else only Bob can decrypt
D is a deterministic algorithm

- Correctness: $\forall k \in \mathscr{K}$ and messages $m \in \mathscr{M}$, if we execute $c \xleftarrow{R} E(m, pk_B)$, $m' \leftarrow D(c, sk_B)$, then with probability 1, $m = m'$

Hard Problems

- No efficient solution, In spite of extensive efforts
  - But: **verification** of solutions is easy (`one-way' hardness)
  - Hardness in the average case
- Problem 1: Factoring
  - Choose randomly $p,q \in_R LargePrimes$
  - Given pq, it is infeasible to find p,q
  - Verification? Easy, just multiply factors
  - RSA cryptosystem and many other tools
- Problem 2: Discrete logarithm in cyclic group Z_p^*
  - Where p is a safe prime
  - Given g^a, find $a \in \mathbb{Z}_p^*$
  - Verification is efficient by exponentiation: $O((\log n)^3)$
  - Basis for the Diffie-Hellman Key Exchange and many other tools

Key Agreement



- Establish shared key between Alice and Bob
- **Without** assuming an existing shared ('master') key or trusted authority
- Use public information from A, B to setup shared secret key k.
- Eavesdropper cannot learn the key k.

Whitfield Diffie & Martin Hellman 1976 "New Directions of Cryptography"
Awarded Turing Award in 2015

Diffie-Hellman Key Agreement

Choose group cyclic G of order p and
Choose generator of $g \in G$, i.e. G (i.e. G = $\{1, g, g^2, g^3, \cdots, g^{p-1}\}$)

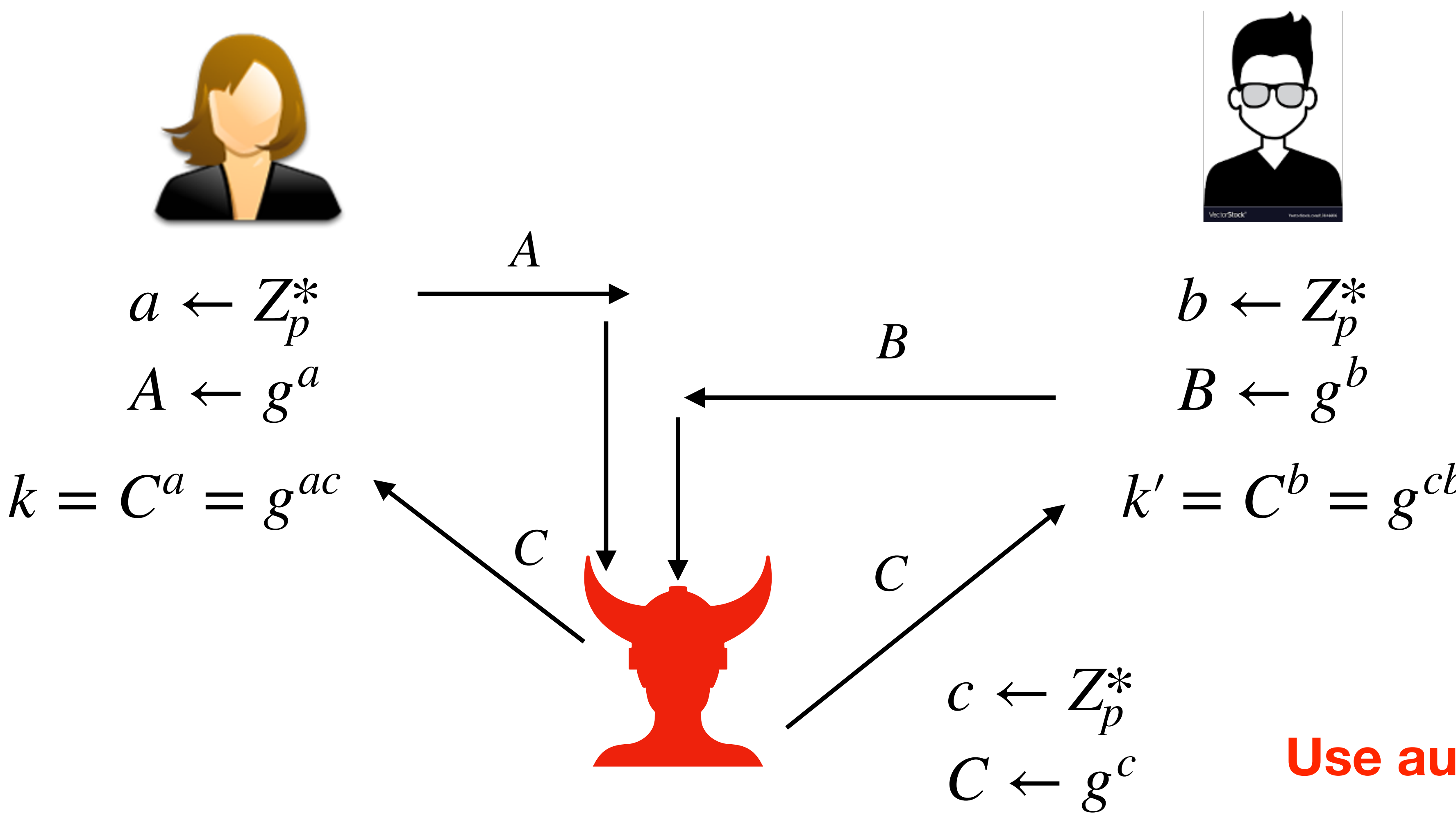
$a \leftarrow Z_p^*$	$\xrightarrow{A}$	$b \leftarrow Z_p^*$
$A \leftarrow g^a$	$\xleftarrow{B}$	$B \leftarrow g^b$
$k = B^a = g^{ab}$		$k = A^b = g^{ba}$

K is the common key

Given $A \leftarrow g^a$**, it is hard to find** $a = \log_g A \mod p$

(Wo)man in the middle attack

Choose group G of order p and generator of $g \in G$



Use authentication

Alice encrypts message with k which can be decrypted by Charlie
Bob encrypts message with k’ which can be decrypted by Charlie

Discrete Logarithm Problem

- Computing logarithm is quite efficient over the real numbers
- For a cyclic multiplicative group G
  - Cyclic group: exists <u>generator</u> g s.t. $(\forall a \in G)(\exists i)\left(a = g^i\right)$
  - Discrete log problem: given generator g and $a \in G$, find i s.t.$a = g^i$
  - Verification: exponentiation (efficient algorithm)
  - For prime p , the group $\mathbb{Z}_p^*$={1,…p-1} is cyclic [multiplications mod p]
- Is discrete-log hard?
  - Some ‘weak’ groups, i.e., where discrete log is **not** hard: (in next tutorial)
    - $\mathbb{Z}_p^*$ for prime p, where $(p-1)$ has only ‘small’ prime factors
      - Using the Pohlig-Hellman algorithm
      - Mistakes/trapdoors found, e.g., in OpenSSL’16
  - Other groups studied, considered Ok (‘hard’)
  - **Safe-prime** groups: $\mathbb{Z}_p^*$ for **safe prime: p= 2q+1 for prime q**

Discrete Log Assumption
for safe prime group

- Safe-prime groups: $\mathbb{Z}_p^*$ for safe prime: p= 2q+1 for prime q
- Given PPT adversary A, and n-bit safe prime p:

$$\Pr\begin{bmatrix} g \leftarrow Generator(p); \\ x \xleftarrow{\$} \{1 \ldots p\!-\!1\}; \\ A(x) = a \mid x = g^a mod\ p \end{bmatrix} \approx negl(n)$$

1. Similar assumptions for (some) other groups
2. Knowing q, it is easy to find a generator g
3. Any generator (primitive element) is ok

Security of DH Key Exchange

Choose group G of order p and
Choose generator of $g \in G$, i.e. G (i.e. G = $\{1, g, g^2, g^3, \cdots, g^{p-1}\}$)

$a \leftarrow Z_p^*$

$A \leftarrow g^a$

$k = B^a = g^{ab}$

$\xrightarrow{A}$

$\xleftarrow{B}$

$b \leftarrow Z_p^*$

$B \leftarrow g^b$

$k = A^b = g^{ba}$

K is the common key

But DH requires stronger assumption than Discrete Log:

From $g^b \bmod p$ and $g^a \bmod p$, Adversary can compute $g^{ab} \bmod p$ (without knowing/learning a,b or ab)?

Computational DH (CDH) Assumption

for safe prime group

- Safe-prime groups: $\mathbb{Z}_p^*$ for safe prime: p= 2q+1 for prime q
- Given PPT adversary A, and n-bit safe prime p:
- $$\Pr\left[\begin{array}{l}(p,q) \leftarrow primes\ s.t.\ \ p = 2q+1; \\ g \leftarrow Generator(p); \\ a, b \leftarrow \{1 \ldots p{-}1\}; \\ A(g^a, g^b\ mod\ p) = g^{ab}\ mod\ p\end{array}\right] \approx negl(n)$$

Given g^a and g^b, it is difficult to compute $g^{ab} \bmod p$

Decisional DH (DDH) Assumption

- Given an RSA instance (n, e, y) and an element $\hat{x} \in \mathbb{Z}_n^*$, it is easy to say if its a correct instance by checking if $\hat{x}^e = y$
- Given (g^a, g^b), it is hard to determine if g^{ab} is a solution of the problem instance
- Not only is the computational problem hard, the decisional problem is also hard
- Many problems use this fact to their advantage
- For $a, b, c \in \mathbb{Z}_q$ we call (g^a, g^b, g^c) a DH-triple, if $c = ab$.
- The DDH assumption tells us that there is no efficient algorithm that can effectively distinguish between a DH-triple and a random triple.
- DDH Assumption $\implies$ CDH Assumption

Partial Information Leak for DH

- Adversary compute *partial* information about g^{ba} *mod p*
- Consider $\mathbb{Z}_p$(multiplicative group for (safe) prime p)
  - Finding (at least) one bit about g^{ba} *mod p* is easy! (Show as an exercise) Specifically whether $x=g^{ba}$ *mod p* is quadratic residue *mod p*
    - I.e., if $(\exists y)(x = y^2 mod\ p)$
    - Denote: $x \in QR(p)$
  - *Comp*ute Legendre Symbol: $\left(\frac{x}{p}\right) = \begin{cases} 0 & if\ x = 0\ (mod\ p) \\ -1 & if x \notin QR(p) \\ 1 & else \end{cases}$
    - How? Using Euler's criterion: $\left(\frac{x}{p}\right)=x^{(p-1)/2}$ *mod p*
- So…how to use DH 'securely'? Two options!

How to use DH securely?

- Option 1: Use stinger group eg. Schnorr's not safe prime group, testing for QR appears to be a hard problem
- Option 2: Use DH with safe prime p but use key derivation function (KDF)
- Option 2 is preferable
- KDF: A KDF takes in some secret keying material s, which may or may not be uniformly distributed, and where the adversary may also have some auxiliary information about the keying material or its distribution, and outputs a uniformly distributed bit string.
- Naive way use hashing $t \leftarrow H(s)$
- Use t as secret key for example in AES

CDH+KDF

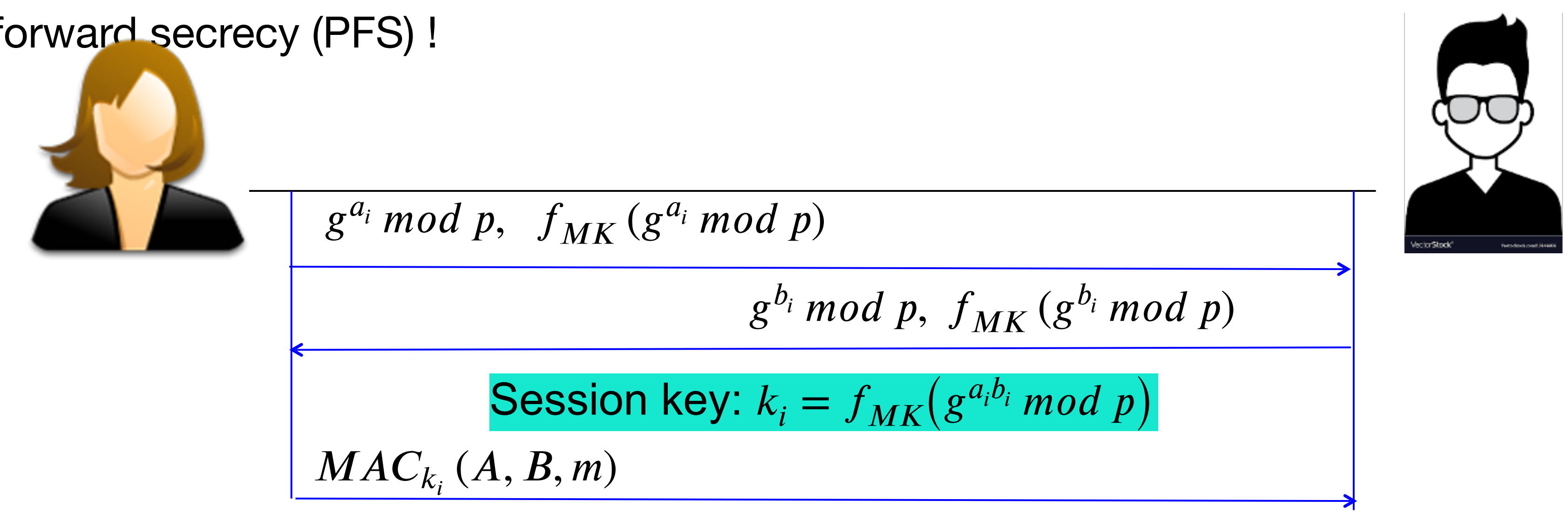
- With CDH, adversary may be able to compute some *partial* information about g^{ba} *mod p* …
  - But 'most bits are random'
- Solution: **Key Derivation Function (KDF)**
  - Two variants: random-keyed and unkeyed (deterministic)
- Randomized - KDF: $k = KDF_s(g^{ab} mod\ p)$ where *KDF* is a key derivation function and s is public random ('salt')
- Deterministic - crypto-hash: $k = h(g^{ab} mod\ p)$ where *h* is randomness-extracting crypto-hash
  - No need in salt, but **not** provably-secure
- Question: isn't (every) PRF a KDF?
- No: PRF requires a message and a uniform random key. KDF does not require input key to be random.
- KDF $\Longrightarrow$ PRF

Authenticated DH

- Recall: DH not secure against MitM attacker
  - We assumed authenticated channel [shared key?]
  - If we have shared key, why not just use it??
- Use DH for **resiliency to key exposure**
  - Do authenticated DH periodically (by establishing new sessions)
  - Use derived key for confidentiality, authentication
    - Some protocols use key to authenticate next exchange
  - ➔ Perfect Forward Secrecy (PFS):
    - Confidentiality of session i is resilient to exposure of all keys, except i-th session key, <u>after session i ended</u>

Authenticated DH

- Assume f which is both a PRF and a KDF
- MK is secret + f is PRF (& MAC) ➔ authentication (MK is the long term key)
- For every session there is a session key
  - And, assuming MK is secret, session keys are secure – even if discrete log would be easy (quantum computers or math break-thru)
- Assuming CDH and that f is **KDF**: secure if MK exposed
  - Since most bits of $g^{a_i b_i}$ are secret
  - Against eavesdropping adv. OR if MK-exposed only after session
  - Perfect forward secrecy (PFS) !



DH Key Establishment to PK Cryptosytem

Choose group G of order p and

Choose generator of $g \in G$, i.e. G (i.e. G = $\{1, g, g^2, g^3, \cdots, g^{p-1}\}$)

Alice		Bob
$a \leftarrow Z_p^*$	$\xrightarrow{A}$	$b \leftarrow Z_p^*$
$A \leftarrow g^a$	$\xleftarrow{B}$	$B \leftarrow g^b$
$k = B^a = g^{ab}$		$k = A^b = g^{ba}$

SK_B is b, PK_B = B

El-Gamal Public Key Cryptosystem

$\mathcal{E}_{EG} = (KG, E, D)$

KG: Choose group G of order p and
Choose generator of $g \in G$, i.e. G (i.e. G = $\{1, g, g^2, g^3, \cdots, g^{p-1}\}$)

- (E_s, D_s) : symmetric auth. encryption defined over (K,M,C)
- $H : G^2 \to K$ a hash function

$\underline{E(m, pk = (g, h = g^b))}$:

$a \leftarrow Z_n, \quad u \leftarrow g^a, \quad v \leftarrow h^a$

$k \leftarrow H(u, v), c \leftarrow E_s(m, k)$

output (u, c)

$\underline{D((u, c), sk = b)}$:

$v \leftarrow u^b$

$k \leftarrow H(u, v), m \leftarrow D_s(c, k)$

output m

Computation: two exponentiation in Encryption, fixed basis
1 Exponentiation in Decryption
Precompute: Can compute, g^i and get a speedup

Computational DH (CDH) Assumption

for safe prime group

- Safe-prime groups: $\mathbb{Z}_p^*$ for safe prime: p= 2q+1 for prime q
- Given PPT adversary A, and n-bit safe prime p:
- $$\Pr\left[\begin{array}{l} (p,q) \leftarrow primes\ s.t.\ \ p = 2q+1; \\ g \leftarrow Generator(p); \\ a, b \leftarrow \{1 \ldots p{-}1\}; \\ A(g^a, g^b\ mod\ p) = g^{ab}\ mod\ p \end{array}\right] \approx negl(n)$$

Given g^a and g^b, it is difficult to compute $g^{ab} \bmod p$

Hash DH (HDH) Assumption

- Given PPT adversary A, and G be a finite cyclic group of order p
- $H : G^2 \to K$ a hash function
- Hash DH (HDH) assumption for (G,h) if
  $(g, g^a, g^b, H(g^b, g^{ab})) \approx_p (g, g^a, g^b, R)$,
- where $g \leftarrow Generator(G), \;\; a, b \leftarrow Z_p, \;\; R \leftarrow K$ (R uniform on K)

Security of El Gamal

- Semantic Security of El Gamal in the Random Oracle Model: Proof using CDH assumption
- Semantic Security of El Gamal with the Random Oracles (using KDF): Proof using DDH Assumption
- You can also use HDH Assumption (Homework)

El Gamal is Semantically Secure in RO Model

$\underline{E(m, pk = (g, h = g^b))}$:

$a \leftarrow Z_n, \quad u \leftarrow g^a, \quad v \leftarrow h^a$

$k \leftarrow H(u, v), c \leftarrow E_s(m, k)$

output (u, c)

$\underline{D((u, c), sk = b)}$:

$v \leftarrow u^b$

$k \leftarrow H(u, v), m \leftarrow D_s(c, k)$

output m

- Assume $h : G^2 \rightarrow K$ is modelled as a random oracle. If the CDH assumption holds for G, and (E_s, D_s) is semantically secure, then $\mathscr{E}_{EG}$ is semantically secure.
- Proof: If H is a random oracle, then an adv. has to compute k. To compute k, it must know (u,v). u is known, but v can't be computed under CDH assumption on G (Full proof on Boneh-Shoup)

Semantic security of ElGamal without random oracles

- RO model simpler to visualize and fast
- When performance is not a concern do not use RO, prove in standard model
- NOTE: We try to give proofs in standard model as much as possible, because RO model relies on the security of the RO
- $H : G^2 \to K$ is a KDF. A secure KDF can't distinguish between (v,H(a,b)) and (v,k), $k \overset{R}{\leftarrow} K$
- If the DDH assumption holds for $G \quad and \quad H : G^2 \to K$ is a secure KDF, (E_s, D_s) is semantically secure, then $\mathscr{E}_{EG}$ is semantically secure.
- Proof Idea: If adversary sees the ciphertext (u, c), where $u = g^a$ and c is a symmetric encryption created using the key k := H(u,v), where $v = h^a$.
- Suppose the challenger replaces v by a random independent group element $w' \in G$ and constructs k as k := H(u, w'). By the DDH assumption the adversary cannot tell the difference between v and w'.
- Under the KDF assumption, k := H(u, w') looks like a random key in K, independent of the adversary's view, and therefore security follows by semantic security of (E_s, D_s).

Homomorphic Property

- RSA
- choose random primes $p,q \approx 1024$ bits. Set **N=pq**.
- choose integers $\mathbf{e}$ **,** $\mathbf{d}$ s.t. $\mathbf{e \cdot d = 1 \pmod{\varphi(N)}}$

Define: f: as $f(x) = x^e$ in

Lemma: f is one-way under the RSA assumption

Properties: $f(x \cdot y) = f(x) \cdot f(y)$ and **f has a trapdoor**

DH

Fix a finite cyclic group G (e.g $G = (Z_p)^*$) of order n

g: a random generator in G (i.e. $G = \{1, g, g^2, g^3, \ldots, g^{n-1}\}$)

DLog: $f\colon Z_n \longrightarrow G$ as $f(x) = g^x \in G$

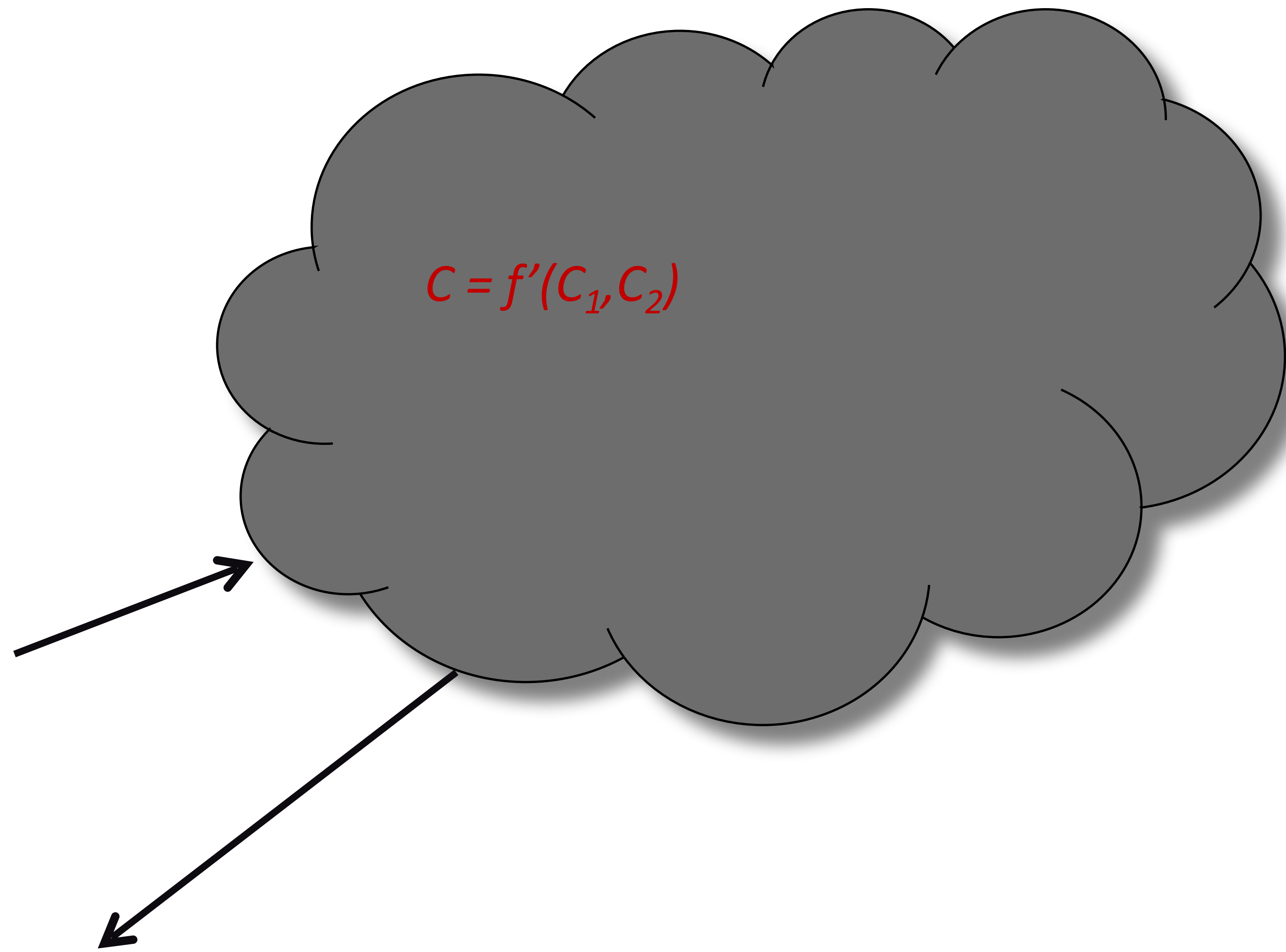
Lemma: Dlog hard in G $\Rightarrow$ f is one-way

Properties: $f(x), f(y) \Rightarrow f(x+y) = f(x) \cdot f(y)$

Homomorphic property

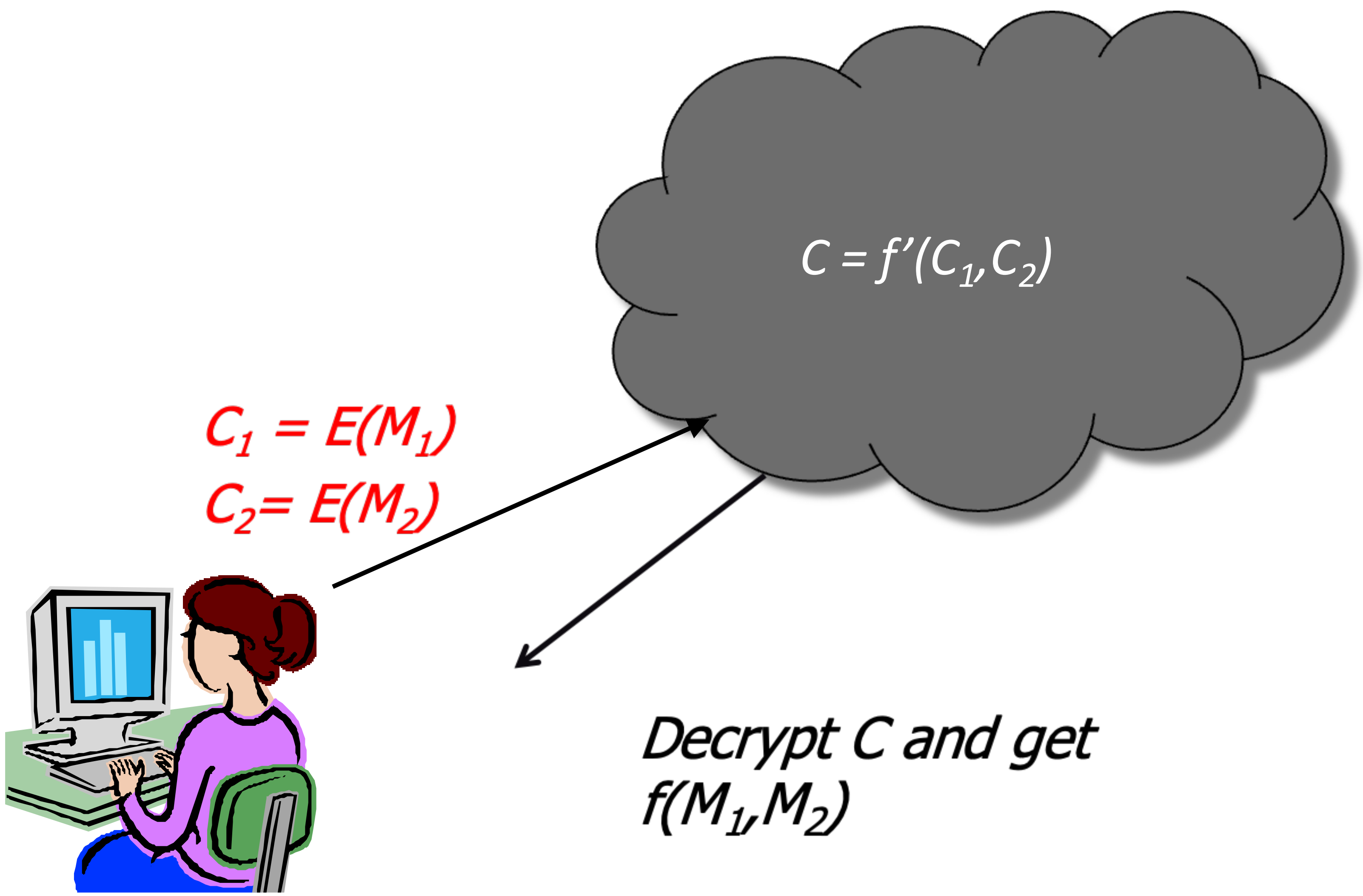
Homomorphic encryption

Given M_1 and M_2, Calculate $f(M_1,M_2)$



Decrypt C and get $f(M_1,M_2)$

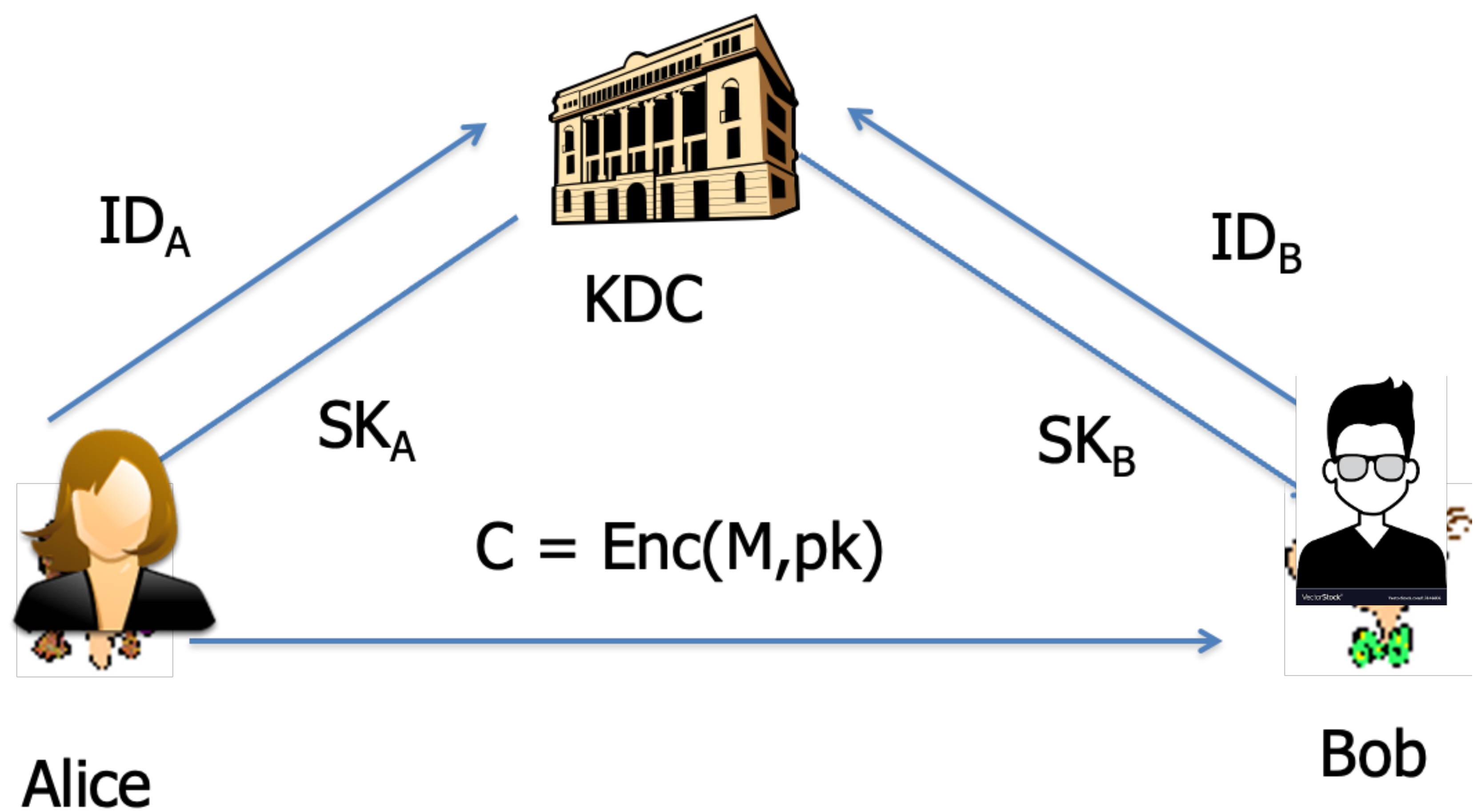
Homomorphic Encryption



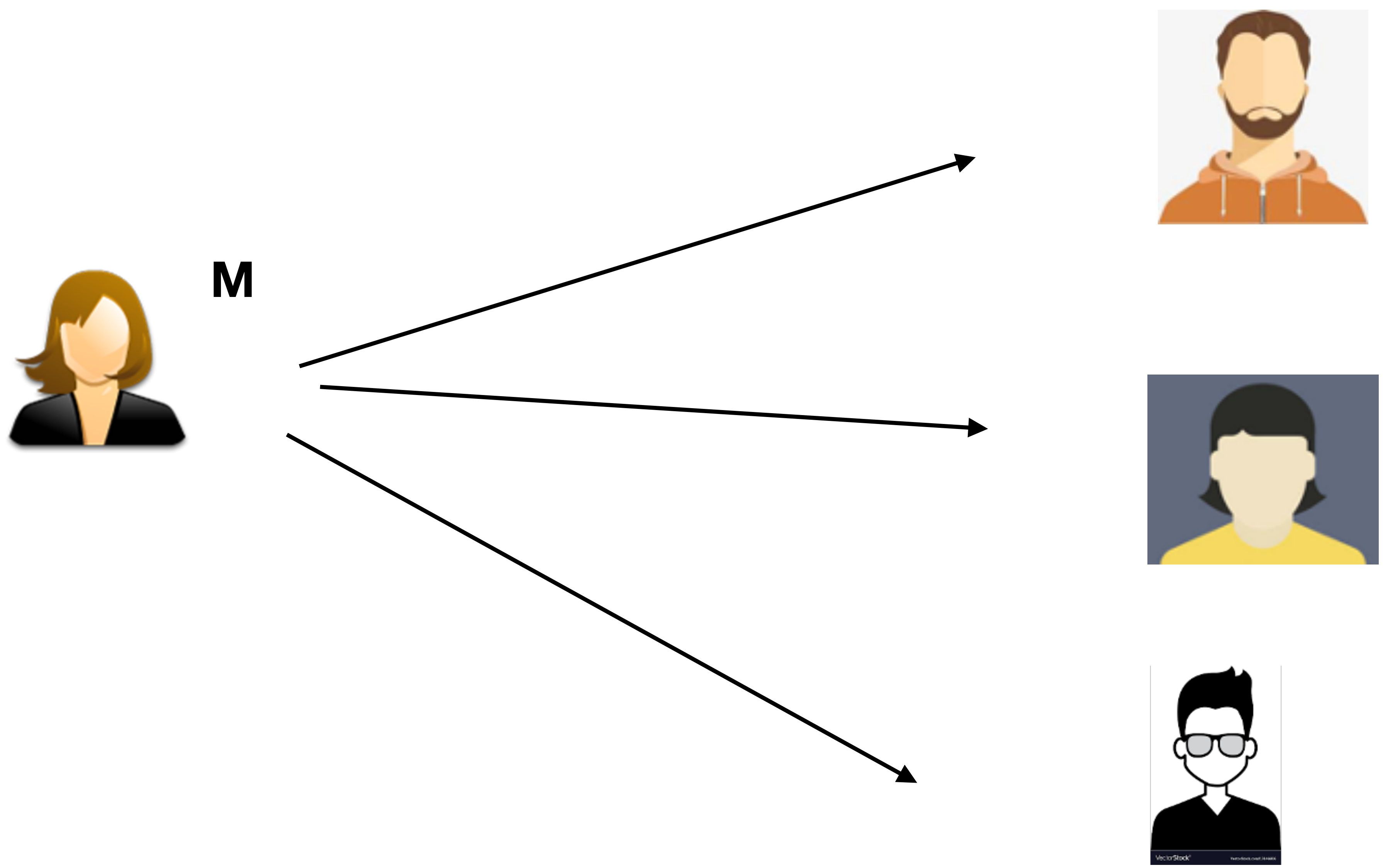
Homomorphic encryption

- Choose group G of order p with generator g
- Public key = (G,p,g,h), $h = g^x$
- Secret key = x
- $E(M) = (g^r, Mh^r)$, r is randomly chosen in Zp
- Cloud cannot calculate r, and hence M (does not know x)
- $E(M_1)E(M_2) = (g^{r1+r2}, M_1M_2h^{r1+r2}) = E(M_1.M_2)$
- Data owner knows r1 and r2, and x, can calculate $M_1M_2 = (M_1M_2h^{r1+r2})/(g^{r1+r2})^x$

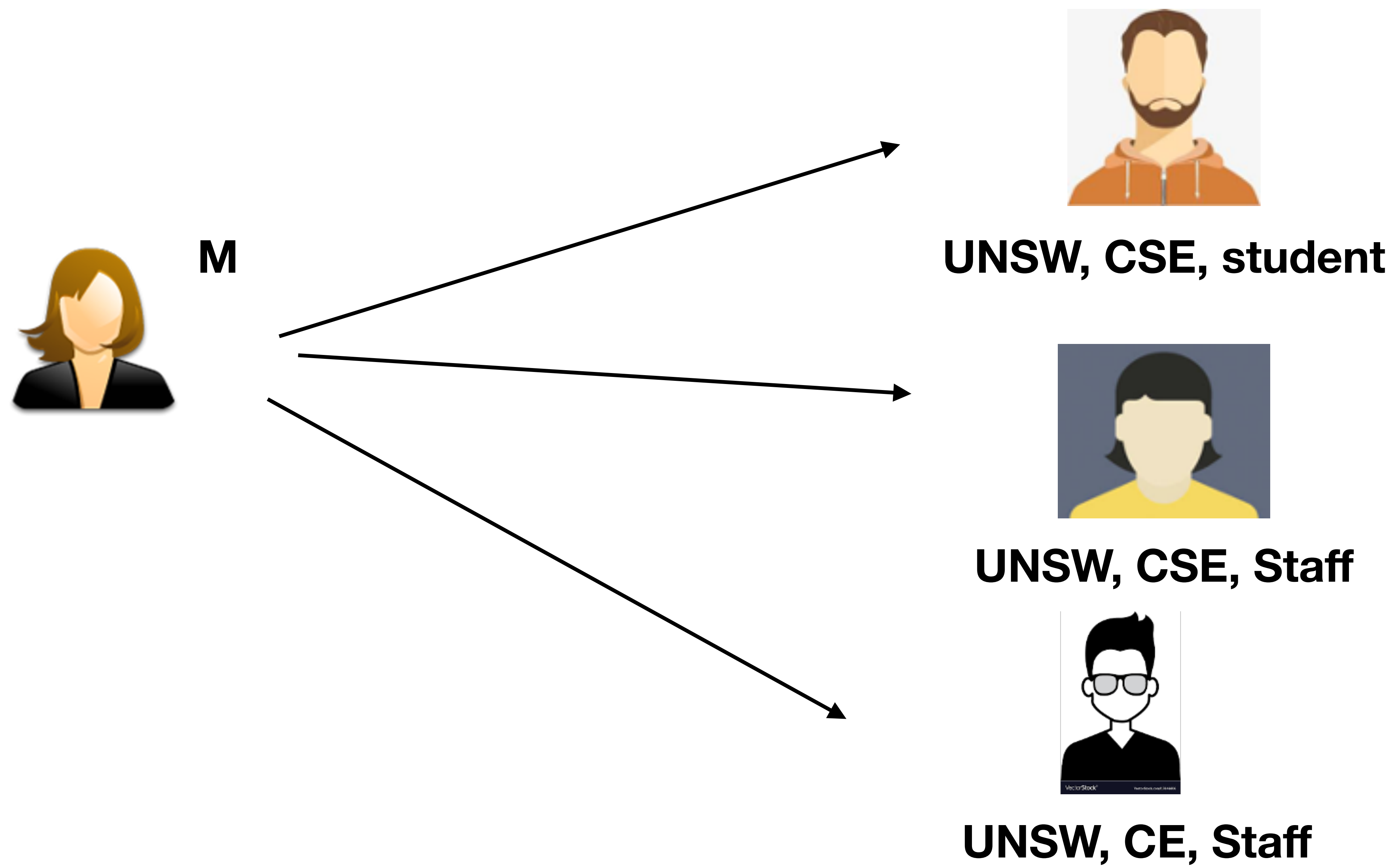
Identity-based Encryption



Broadcast Encryption



Attribute-Based Encryption



Thank you