

# Introduction to Information Security

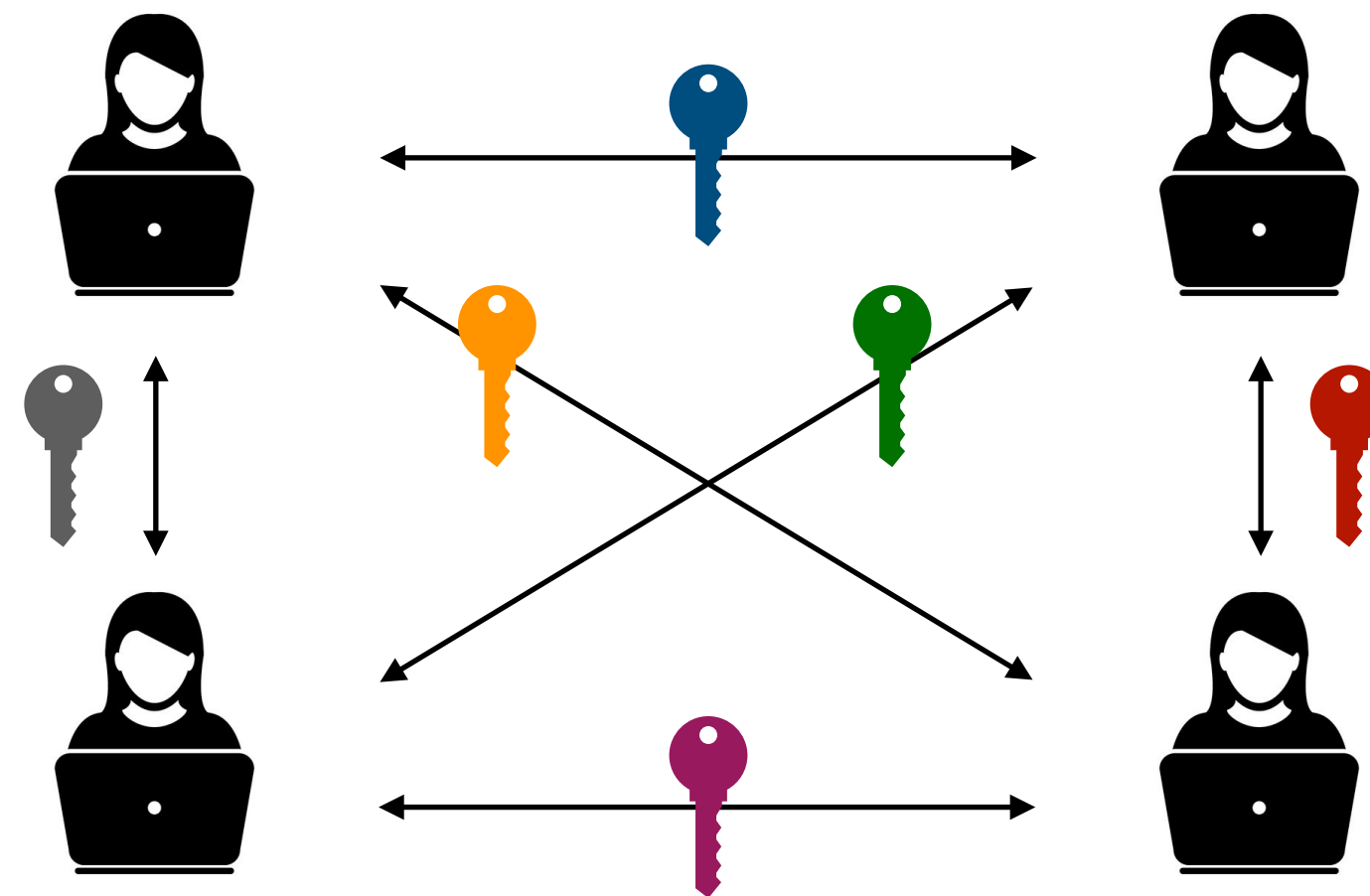
## 7. Public-key Cryptography

Kihong Heo



# Symmetric-Key Encryption

- Recap: the same key shared between two parties
- What happens if there are many users?
  - $n$  users:  $\binom{n}{2} = n(n-1)/2$
  - Example: 4950 keys / 100 users
- Key distribution and maintenance problem

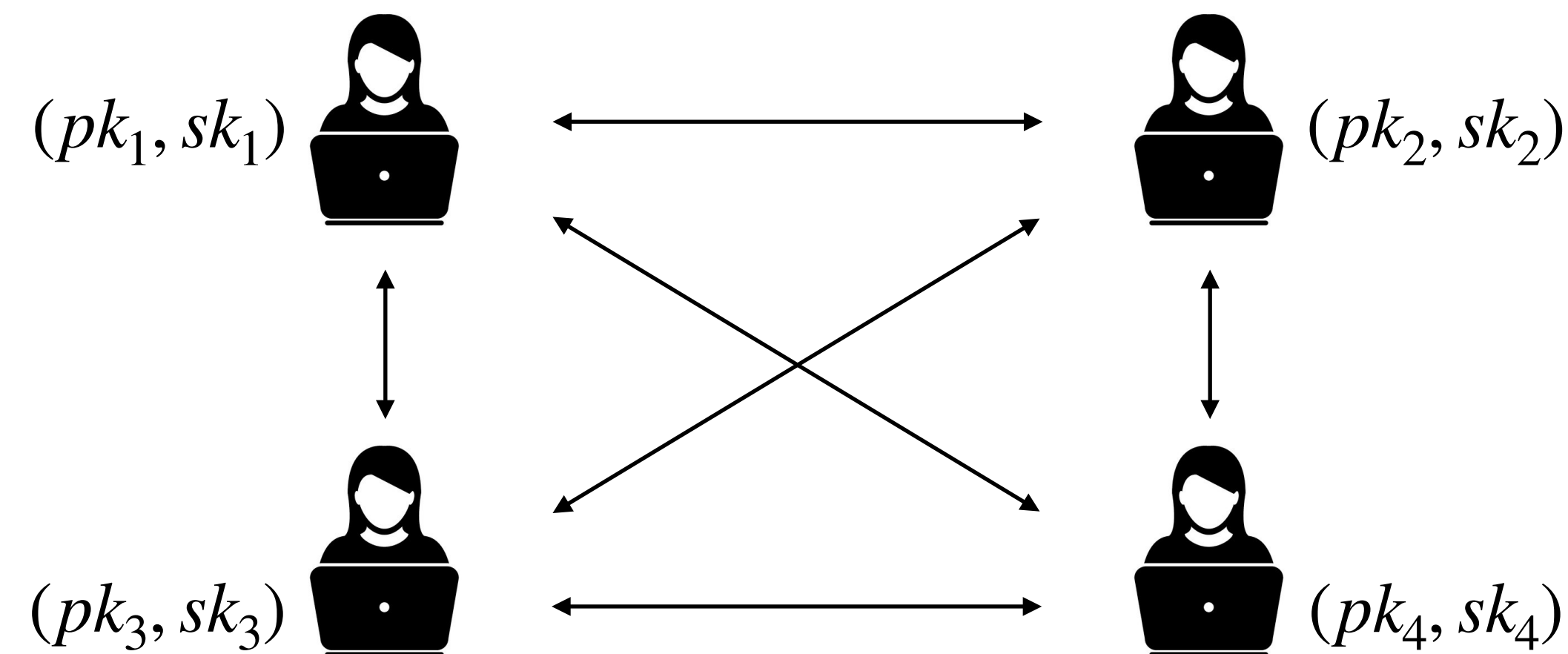


How to solve this issue?



# Public-Key Revolution

- Invented in 1976 by Diffie and Hellman (ACM Turing Award 2015)
- Problem
  - $pk$  : public key, widely disseminated, used for encryption
  - $sk$  : private key, kept secretly, used for decryption
- $n$  users:  $2n$  keys



# Instances

- Secret-key exchange (Diffie-Hellman key exchange)
- Confidentiality: public-key encryption (RSA)
- Integrity: digital signature

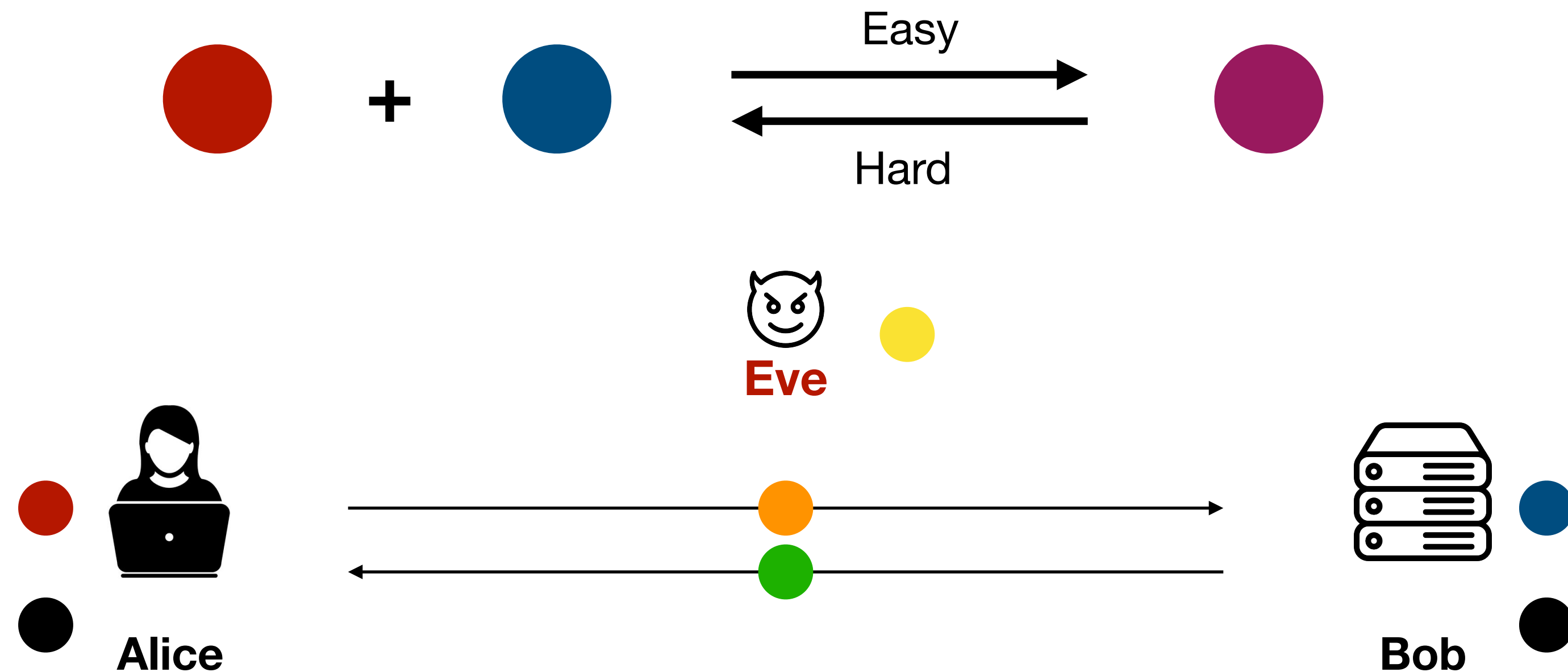
# Secret Key Exchange

- Setting: Alice and Bob want to share a secret key using an insecure channel
- Problem: How can two people (who have never met) agree on a secret key?



# Idea: One-way Function

- Easy in one direction but hard in the reverse direction
  - E.g., discrete logarithm (math), integer factorization (math), color mixing (painting), 비빔밥



# Diffie-Hellman Key Exchange (1)

- Pick two public values: large prime  $p$  and generator  $g$
- Alice has secret value  $a$
- Bob has secret value  $b$

$$p = 23, g = 9$$

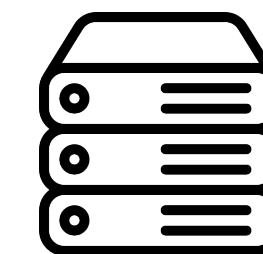


Eve



Alice

$$a = 4$$

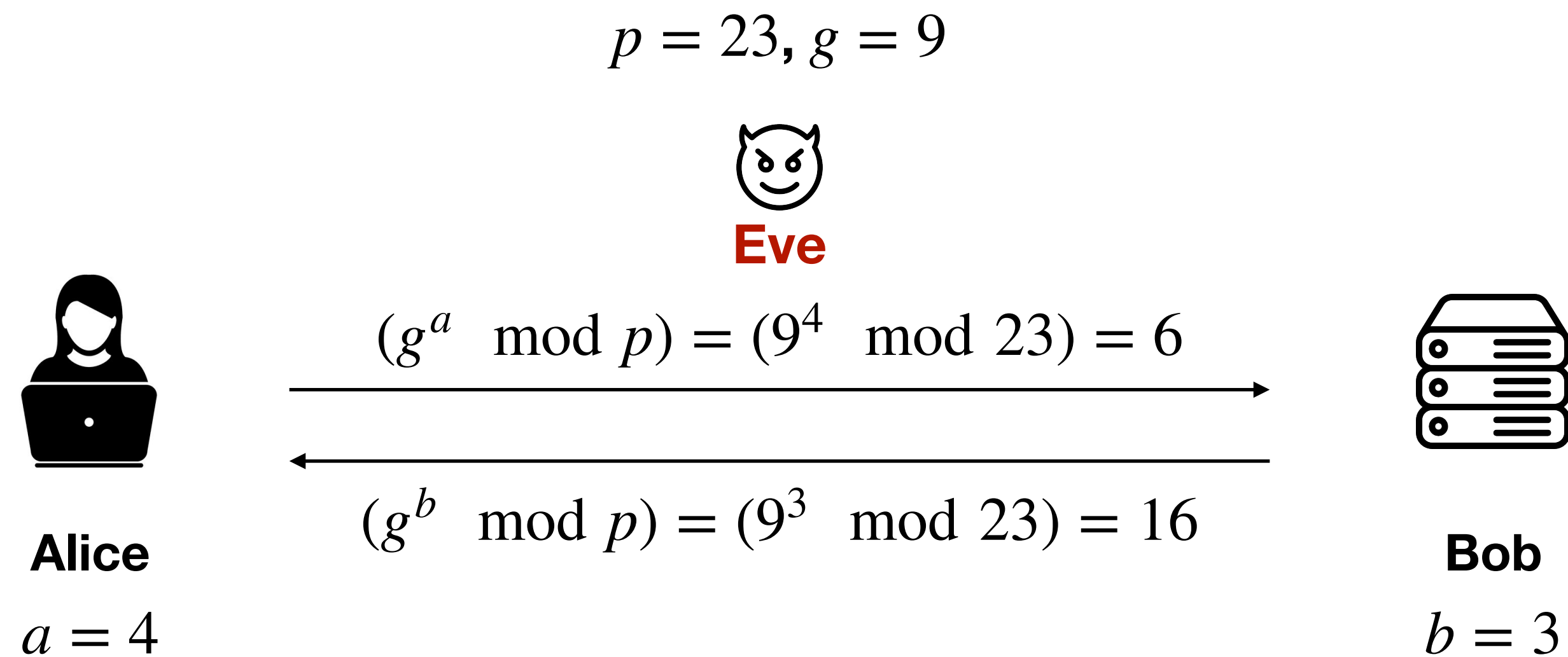


Bob

$$b = 3$$

# Diffie-Hellman Key Exchange (2)

- Alice sends  $A = (g^a \bmod p)$  to Bob
- Bob sends  $B = (g^b \bmod p)$  to Alice





# Diffie-Hellman Key Exchange (3)

- Alice computes  $(B^a \bmod p) = ((g^b \bmod p)^a \bmod p) = (g^{ab} \bmod p)$
- Bob computes  $(A^b \bmod p) = ((g^a \bmod p)^b \bmod p) = (g^{ab} \bmod p)$
- Secret key:  $g^{ab} \bmod p$

$$p = 23, g = 9$$



**Eve**

$$K = (16^4 \bmod 23) = 9$$

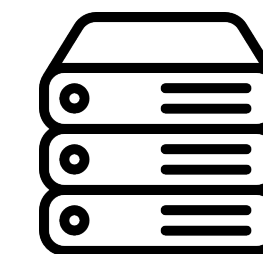


**Alice**

$$a = 4$$

$$(g^a \bmod p) = (9^4 \bmod 23) = 6$$

$$(g^b \bmod p) = (9^3 \bmod 23) = 16$$



**Bob**

$$b = 3$$

$$K = (6^3 \bmod 23) = 9$$

# Correctness & Security

- Correctness: Is  $K_{Alice} = (B^a \bmod p)$  equal to  $K_{Bob} = (A^b \bmod p)$ ?

$$(B^a \bmod p) = ((g^b \bmod p)^a \bmod p) = (g^{ab} \bmod p)$$

$$(A^b \bmod p) = ((g^a \bmod p)^b \bmod p) = (g^{ab} \bmod p)$$

## Theorem

$$((X \bmod p)^k \bmod p) = (X^k \bmod p)$$

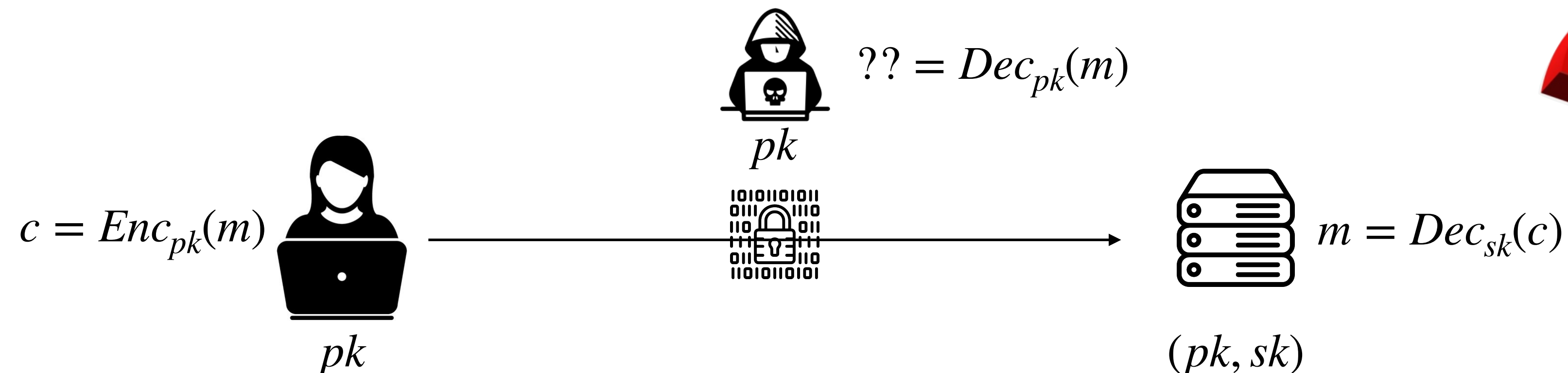
- Security
  - Eve cannot efficiently compute  $(g^{ab} \bmod p)$  without knowing  $a$  and  $b$ 
    - Eve can observe  $p$ ,  $g$ ,  $(g^a \bmod p)$ , and  $(g^b \bmod p)$
  - Discrete logarithm problem: given  $m$ ,  $n$ , and  $p$ , find  $x$  s.t.  $(m^x = n \bmod p)$ 
    - No efficient algorithms

# Instances

- Secret-key exchange (Diffie-Hellman key exchange)
- Confidentiality: public-key encryption (RSA)
- Integrity: digital signature

# Confidentiality with Public-Key

- Generate  $(pk, sk)$  and publicize  $pk$
- Anyone with  $pk$  can encrypt message
- Only the one with  $sk$  can decrypt message
  - Cannot decrypt with  $pk$



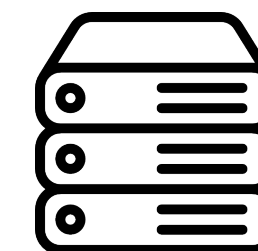
How is it possible?

# RSA Cryptosystem

- Invented by Rivest, Shamir, and Adleman in 1977
  - ACM Turing award in 2002
- Rely on the practical difficulty of factoring the product of two large prime numbers
  - But efficiently solvable by quantum computers

# RSA Algorithm (1)

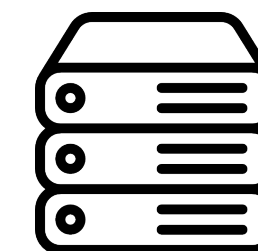
- Select two large primes  $p$  and  $q$
- Compute  $n = pq$  and  $\phi(n) = (p - 1)(q - 1)$



$$\begin{aligned} p &= 7, q = 13 \\ n &= 91, \phi(n) = 72 \end{aligned}$$

# RSA Algorithm (2)

- Choose  $e$  s.t.  $1 < e < \phi(n)$  and  $\gcd(e, \phi(n)) = 1$
- Choose  $d$  s.t.  $1 < d < \phi(n)$  and  $ed = 1 \pmod{\phi(n)}$



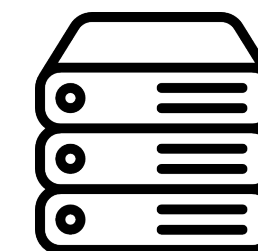
$p = 7, q = 13$   
 $n = 91, \phi(n) = 72$   
 $e = 5, d = 29$

# RSA Algorithm (3)

- Public key:  $(n, e)$
- Private key:  $(n, d)$



$$pk = (n, e) = (91, 5)$$



$$pk = (n, e) = (91, 5)$$

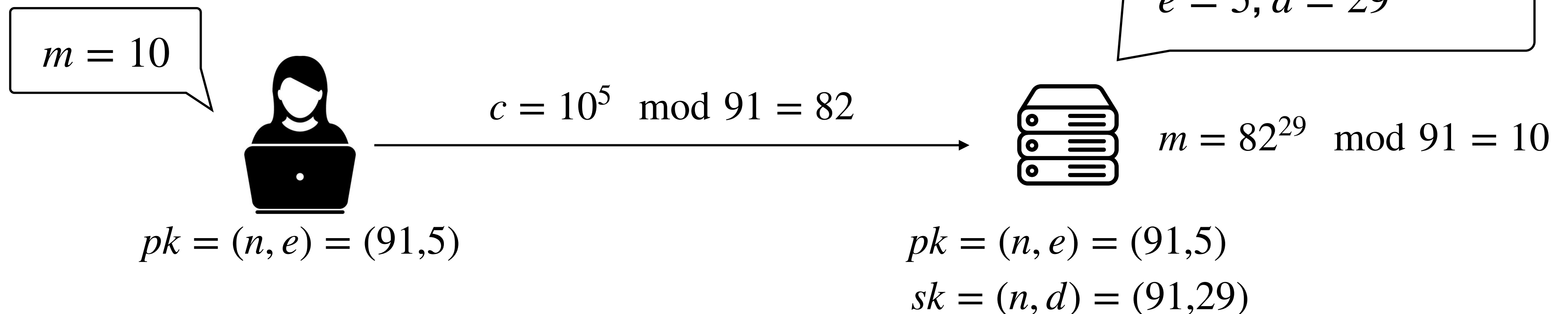
$$sk = (n, d) = (91, 29)$$

$$\begin{aligned} p &= 7, q = 13 \\ n &= 91, \phi(n) = 72 \\ e &= 5, d = 29 \end{aligned}$$



# RSA Algorithm (4)

- Encryption
  - For plaintext  $m < n$ ,  $c = m^e \bmod n$
- Decryption
  - $m = c^d \bmod n$
- Correctness:  $(m^e \bmod n)^d \bmod n = m$



# Correctness & Security

- Correctness:  $(m^e \bmod n)^d \bmod n = m$

$$(m^e \bmod n)^d \bmod n = (m^e)^d \bmod n$$

$$(m^e)^d = m^{ed} = m^{1+k \cdot \phi(n)}$$

$$m^{1+k \cdot \phi(n)} \equiv m \bmod n$$

## Theorem

$$((X \bmod p)^k \bmod p) = (X^k \bmod p)$$

“Choose  $d$  s.t.  $1 < d < \phi(n)$  and  
 $ed = 1 \bmod \phi(n)$ ”

## Euler's Theorem

If  $p$  and  $q$  are primes,  $n = pq$ , then  
 $\forall a \in \mathbb{Z}_n. a^{k \cdot \phi(n) + 1} \equiv a \bmod n$

- Security
  - Adversary cannot efficiently compute  $p$  and  $q$  from  $n$
  - Adversary can observe  $n$  and  $e$  (i.e., public key) but cannot efficiently compute  $d$
  - Integer factorization problem: given  $n$ , find prime number  $p$  and  $q$  s.t.  $n = pq$

# Comparison to Private-Key Encryption

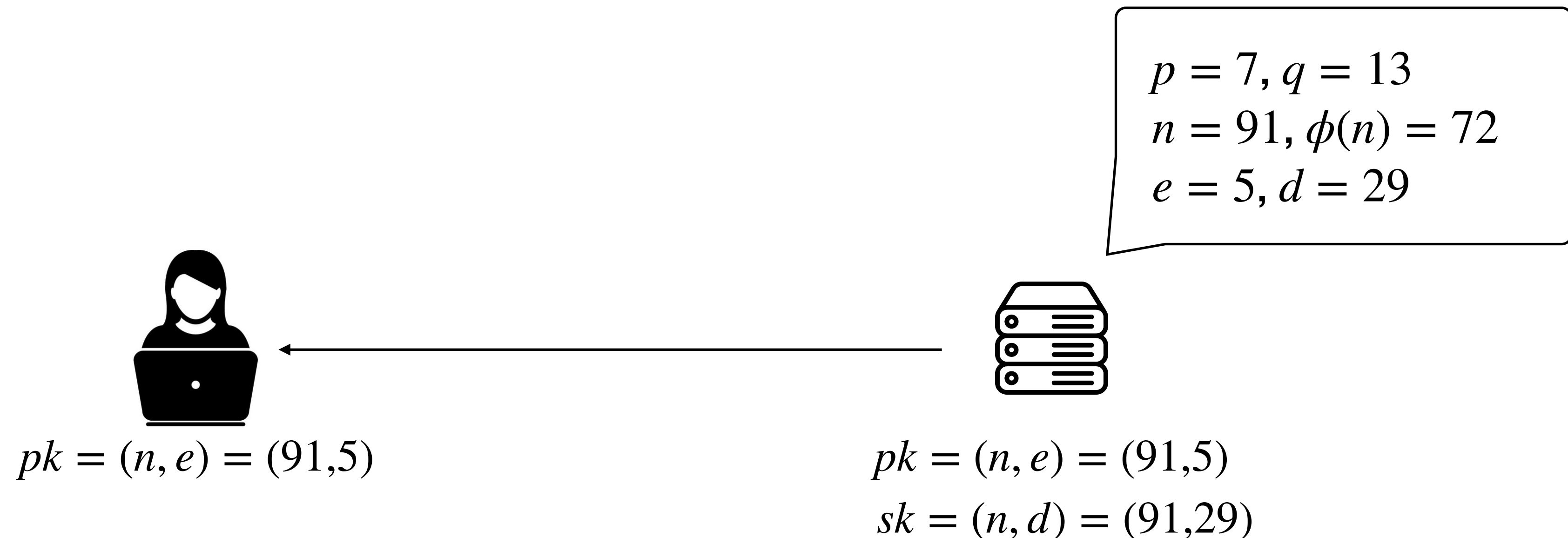
- Pros
  - Does not need any secure key distribution
  - Enable multiple senders to communicate privately with a single receiver
- Cons
  - Roughly 2-3 orders of magnitude slower

# Instances

- Secret-key exchange (Diffie-Hellman key exchange)
- Confidentiality: public-key encryption (RSA)
- Integrity: digital signature

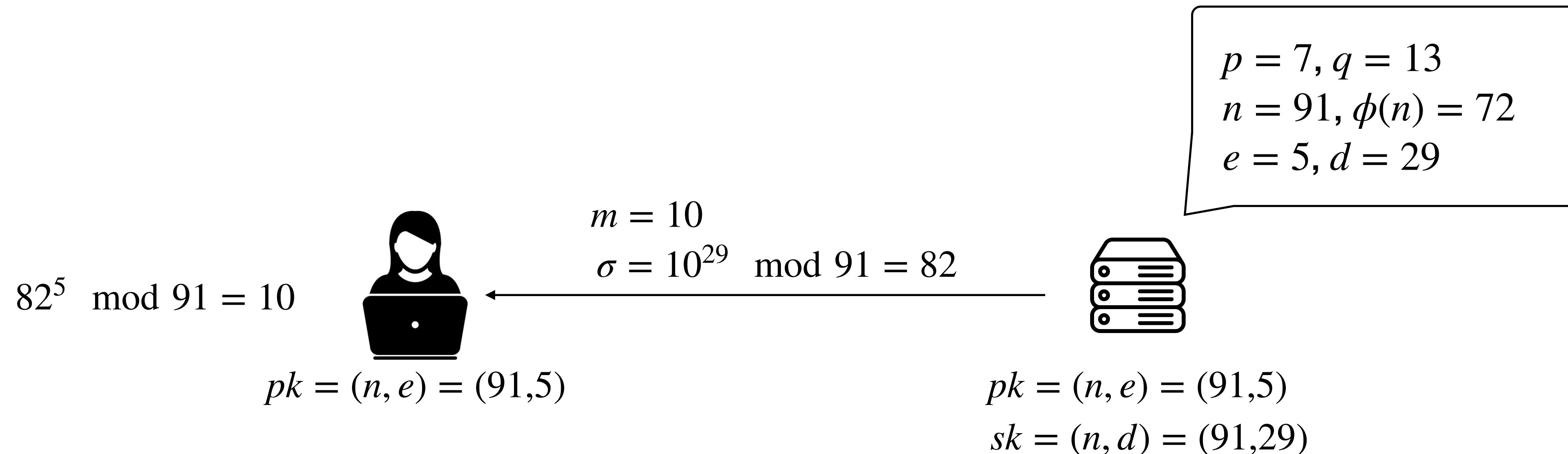
# Digital Signature (1)

- Only the one with  $sk$  can generate signature  $\sigma$  for message  $m$
- Anyone with  $pk$  can verify the signature (i.e., integrity)
- Example: SW patch distribution



# Digital Signature (2)

- Send message  $m$  and signature  $\sigma = m^d \bmod n$
- Verify the integrity by checking  $m$  is equal to  $\sigma^e \bmod n$
- Correctness:  $(m^d \bmod n)^e \bmod n = m$



# Correctness & Security

- Correctness:  $(m^e \bmod n)^d \bmod n = m$

$$(m^e \bmod n)^d \bmod n = (m^e)^d \bmod n$$

$$(m^e)^d = m^{ed} = m^{1+k \cdot \phi(n)}$$

$$m^{1+k \cdot \phi(n)} \equiv m \bmod n$$

## Theorem

$$((X \bmod p)^k \bmod p) = (X^k \bmod p)$$

“Choose  $d$  s.t.  $1 < d < \phi(n)$  and  
 $ed = 1 \bmod \phi(n)$ ”

## Euler's Theorem

If  $p$  and  $q$  are primes,  $n = pq$ , then  
 $\forall a \in \mathbb{Z}_n. a^{k \cdot \phi(n) + 1} \equiv a \bmod n$

- Security
  - Adversary cannot efficiently compute  $p$  and  $q$  from  $n$
  - Adversary can observe  $n$  and  $e$  (i.e., public key) but cannot efficiently compute  $d$
  - Integer factorization problem: given  $n$ , find prime number  $p$  and  $q$  s.t.  $n = pq$

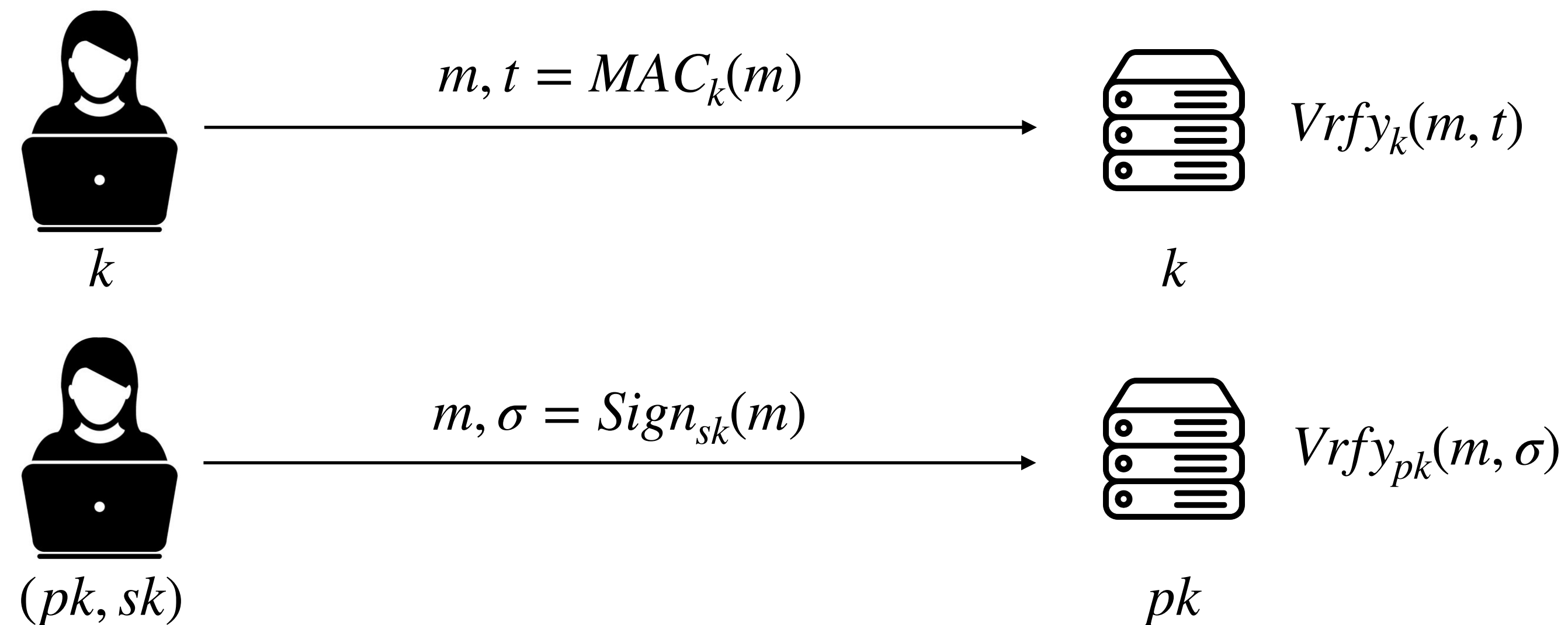
# Digital Signature in Practice

- Authentication: proof that you or something you created is legitimate
- Non-repudiation: signed document becomes proof that Alice indeed signed the document
  - Only Alice can generate  $(m, \sigma)$  and
  - Cannot deny having created the signature



# Comparison to MAC

- Both: ensure the integrity of transmitted messages
- Pros: public verifiability
  - Multiple receivers can verify the signature
- Cons: efficiency

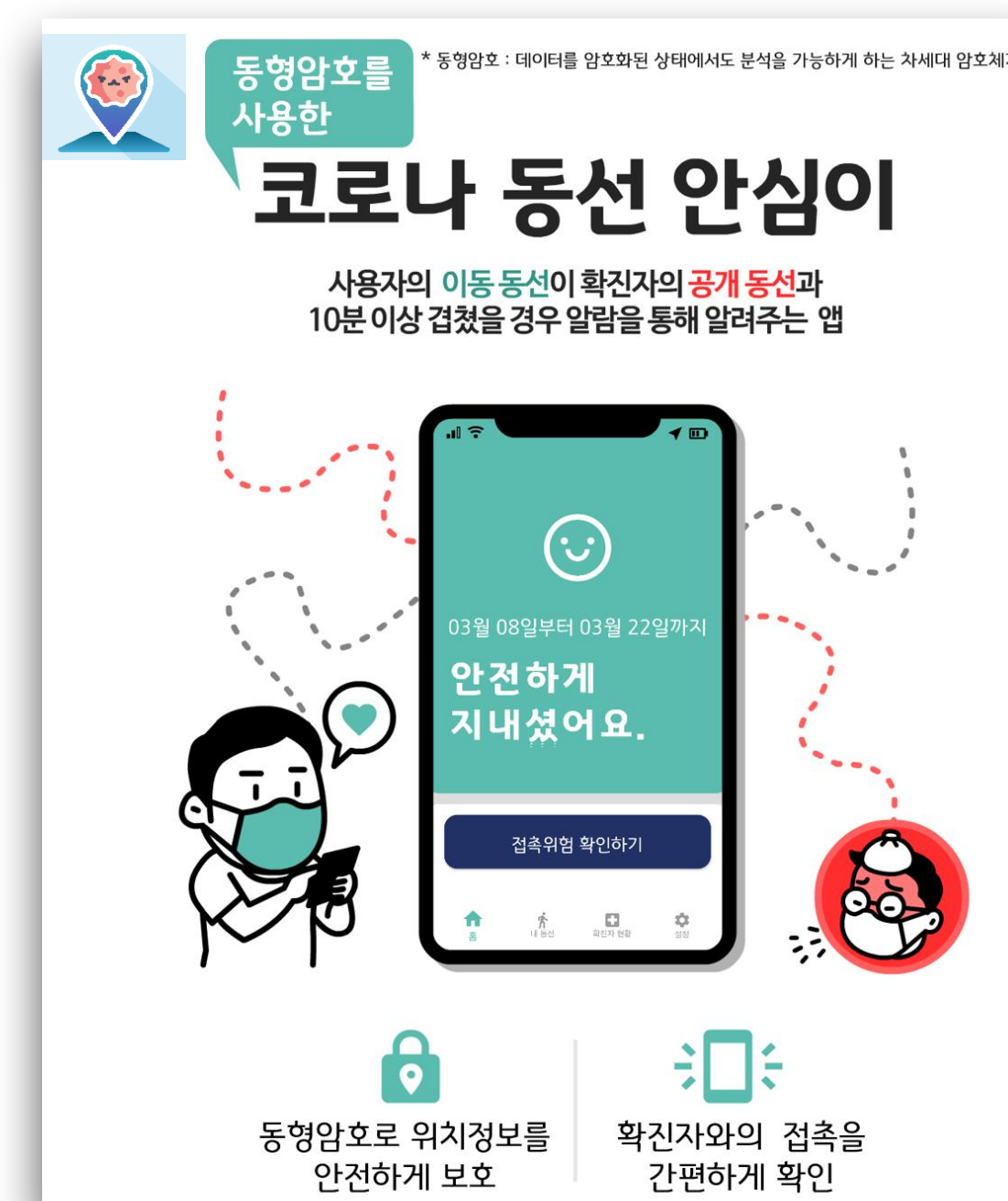
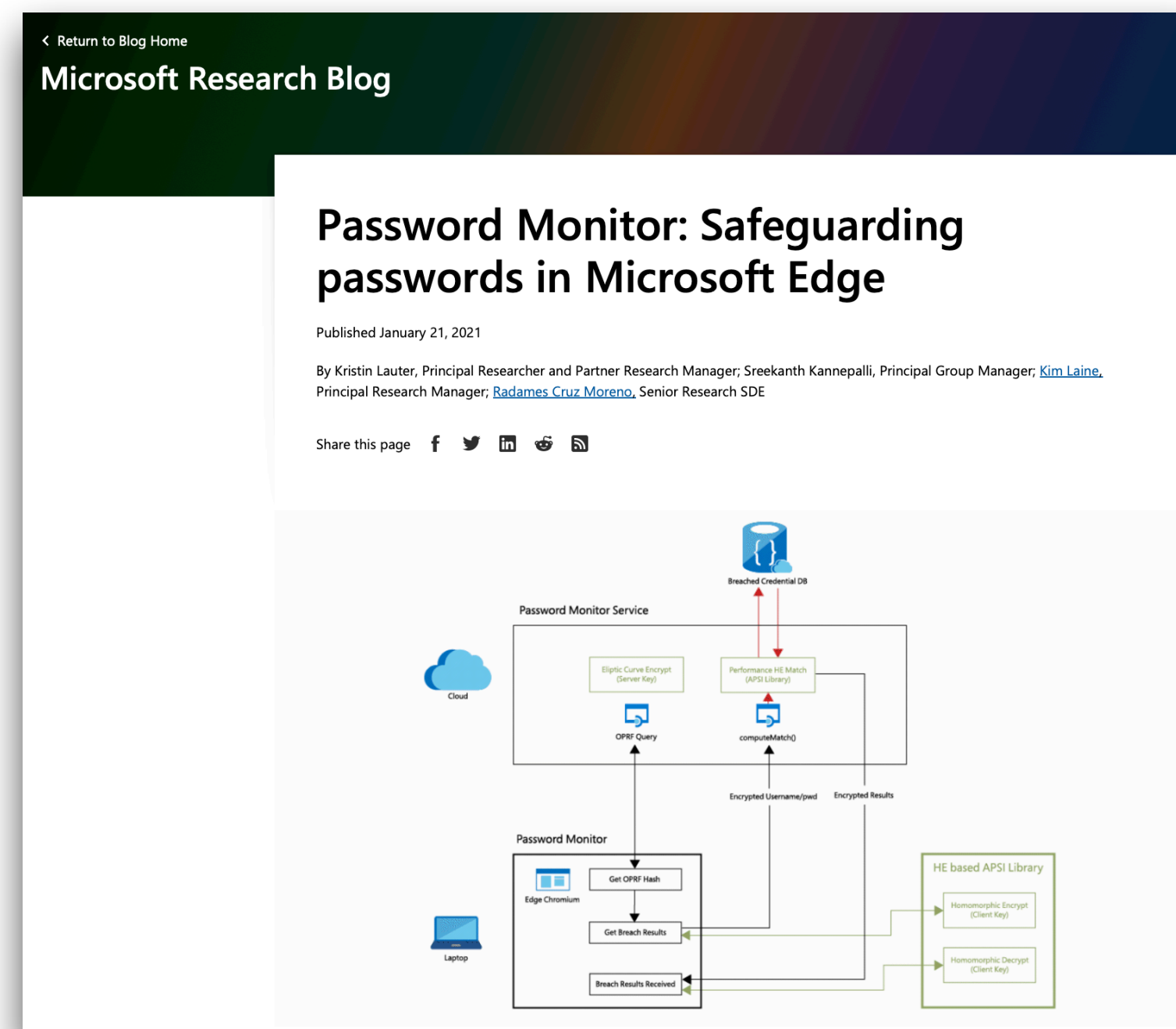


# Holy Grail of Cryptography

- Is it possible to provide a secure public service? (i.e., computations on encrypted data)
- Example
  - Average GPA in the class with encrypted individual GPAs
  - Covid-19 alert with encrypted location information
  - Searchable cloud storage with encrypted data
  - Election with encrypted votes
- Necessary property: homomorphism
  - $Dec(c_1 \oplus c_2) = Dec(c_1) \oplus Dec(c_2)$

# Homomorphic Encryption (동형 암호)

- Allows computations on encrypted data
- “A Fully Homomorphic Encryption Scheme”, C. Gentry, 2009
- Applications



# A Simplified Symmetric HE

- Plaintext space:  $\{0,1\}$
- Secret key:  $p$
- Random numbers:  $q$  and  $\epsilon$
- Encryption:  $Enc(m) = m + pq + 2\epsilon$
- Decryption:  $Dec(c) = (c \bmod p) \bmod 2$
- Homomorphism
  - $Dec(Enc(m_1) + Enc(m_2)) = Dec(Enc(m_1 + m_2)) = m_1 + m_2$
  - $Dec(Enc(m_1) \times Enc(m_2)) = Dec(Enc(m_1 \times m_2)) = m_1 \times m_2$

# Summary

- Public-key revolution: solve key distribution and maintenance problem
  - Diffie-Hellman key exchange
  - Public-key encryption
  - Digital signature
- New emerging technology: homomorphic encryption
  - Computation on encrypted data
  - Application: privacy-preserving services