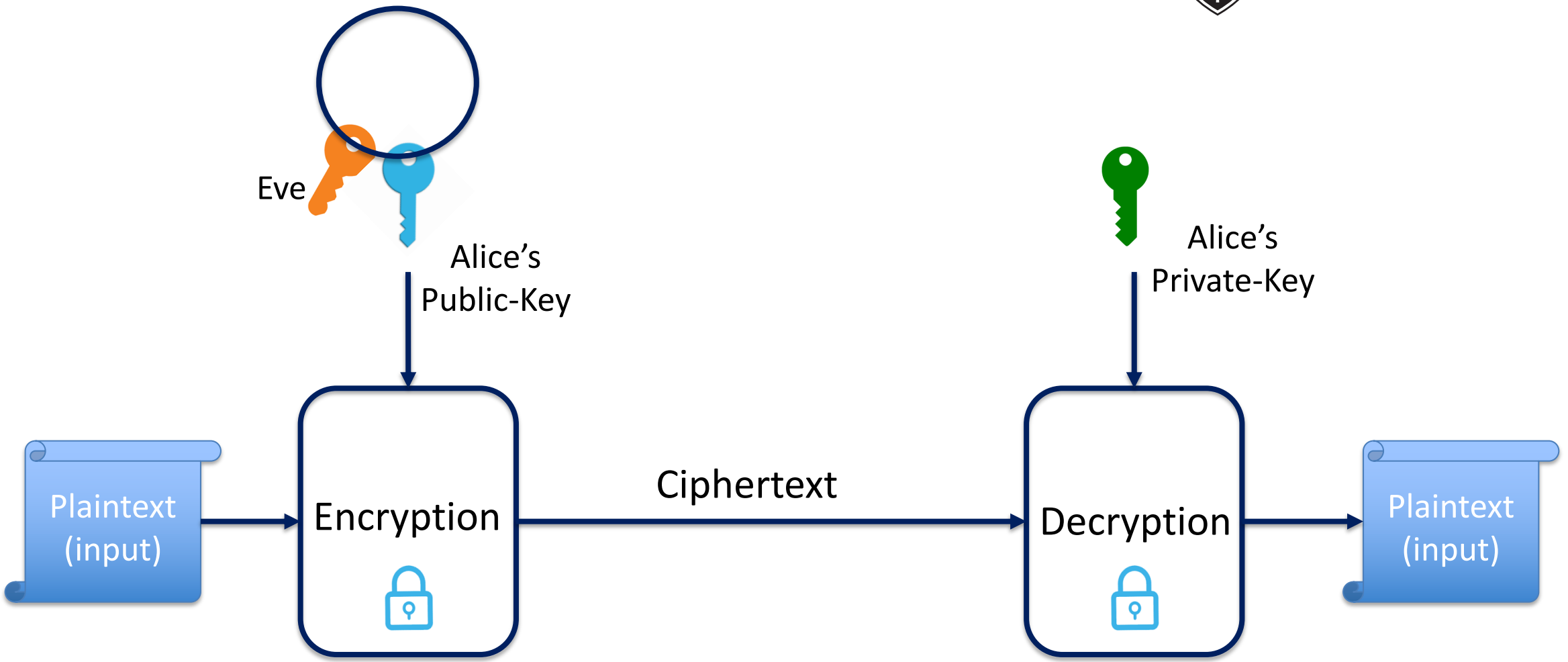
Public-Key Crypto

Public-Key Cryptography

Oregon State University
College of Engineering



Public Key Cryptography

Oregon State University
College of Engineering

- Two keys
  - *Private key* known only to individual
  - *Public key* available to anyone

- Usage:
  - Confidentiality: encipher using public key, decipher using private key
  - Integrity/authentication: encipher using private key, decipher using public one
  - Symmetric Key distribution

Security Services [1 of 2]

Oregon State University
College of Engineering

- Confidentiality
  - Only the owner of the private key knows it, so text enciphered with a public key cannot be read by anyone except the owner of the corresponding private key

- Authentication
  - Only the owner of the private key knows it, so text enciphered with a private key must have been generated by the owner

Security Services [2 of 2]

Oregon State University
College of Engineering

- Integrity
  - Enciphered letters cannot be changed undetectably without knowing private key
- Non-Repudiation
  - Message enciphered with private key came from someone who knew it

General Facts about Public Key Systems

Oregon State University
College of Engineering

- Public Key Systems are much slower than Symmetric Key Systems
  - RSA is 100 to 1000 times slower than DES. 10,000 times slower than AES
  - Generally used in **conjunction with a symmetric system (hybrid-encryption)** for bulk encryption
- Public Key Systems are based on "hard" problems
  - E.g., factoring large composites of primes, computing discrete logarithms
- Only a handful of public key systems perform both encryption and signatures

Major Public Key Algorithms

Oregon State University
College of Engineering

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Requirements

Oregon State University
College of Engineering

- It is computationally easy
  - to generate a key pair
  - to encrypt a message using a public key
  - to decrypt a message using the private key
- It is computationally infeasible
  - for an opponent knowing only the public key to determine the private key
  - for an opponent knowing the public key and a ciphertext to recover the original message
- Either of the two related keys can be used for enciphering with the other used for deciphering

Key Points

Oregon State University
College of Engineering

- Public-Key crypto systems use two types of keys
  - Private and public
- Public-Key crypto systems can provide
  - Confidentiality (key encryption)
  - Integrity/Authenticity
  - Symmetric key exchange
- Public-key crypto systems are computationally much more expensive than symmetric ciphers
- Their security relies on well-known hard problems

Math Background

Modular Arithmetic [1 of 2]

Oregon State University
College of Engineering

- a mod b = x if for some k >= 0, bk + x = a
- Associativity, Commutativity, and Distributivity hold in Modular Arithmetic
- Inverses also exist in modular arithmetic
  - a + (-a) mod n = 0
  - $a * a^{-1}$ mod n = 1

Modular Arithmetic [2 of 2]

Oregon State University
College of Engineering

- Reducibility also holds
  - (a + b) mod n = (a mod n + b mod n) mod n
  - a * b mod n = ((a mod n) * (b mod n)) mod n

- Fermat's Thm: if p is any prime integer and a is an integer, then a^p mod p = a
  - Corollary: a^{p-1} mod p = 1 if a != 0 and a is relatively prime to p

Totient Function ∅(n)

Oregon State University
College of Engineering

- Number of positive integers less than *n* and relatively prime to *n*
  - *Relatively prime* means with no factors in common with *n*

- Example: ∅(10) = ?
  - 4 because 1, 3, 7, 9 are relatively prime to 10 (no common factors)

- Example: ∅(p) = ? where p is a prime
  - p-1 because all lower numbers are relatively prime

Euler's Theorem

- Euler generalized Fermat's Thm for composite numbers.
  - Recall Fermat's Thm $a^{p-1}=1 \bmod p$ if a != 0

- Euler's Thm: $X^{\emptyset(n)}=1 \bmod n$
  - Where q and p are primes
  - $n = pq$
  - $\emptyset(n) = (p-1)(q-1)$

Diffie-Hellman Key Exchange

Public Key Cryptography

Oregon State University
College of Engineering

- Two keys
  - *Private key* known only to individual
  - *Public key* available to anyone

- Usage:
  - Confidentiality: encipher using public key, decipher using private key
  - Integrity/authentication: encipher using private key, decipher using public one
  - Symmetric Key distribution

Major Public Key Algorithms

Oregon State University
College of Engineering

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Modular Arithmetic [1 of 2]

Oregon State University
College of Engineering

- a mod b = x if for some k >= 0, and x < b, bk + x = a
  - Example: 25 mod 7 = 4; here k = 3 and 7*3 + 4 = 25

- Associativity, Commutativity, and Distributivity hold in Modular Arithmetic
  - ((a + b) mod n + c) mod n = (a + (b + c) mod n) mod n
  - (a + b) mod n = (b + a) mod n
  - a * (b + c) mod n = (a * b mod n + a * c mod n) mod n

- Inverses also exist in modular arithmetic
  - a + (-a) mod n = 0
  - $a * a^{-1} \bmod n = 1$

Modular Arithmetic [2 of 2]

Oregon State University
College of Engineering

- Reducibility also holds
  - (a + b) mod n = (a mod n + b mod n) mod n
  - a * b mod n = ((a mod n) * (b mod n)) mod n

Diffie-Hellman

Oregon State University
College of Engineering

- The first public-key cryptosystem proposed (in public domain)
- Usually used for exchanging keys securely
- Enables computing a common, shared key
  - Called a *symmetric key exchange protocol*
- Based on discrete logarithm problem

Discrete Logarithm Problem

Oregon State University
College of Engineering

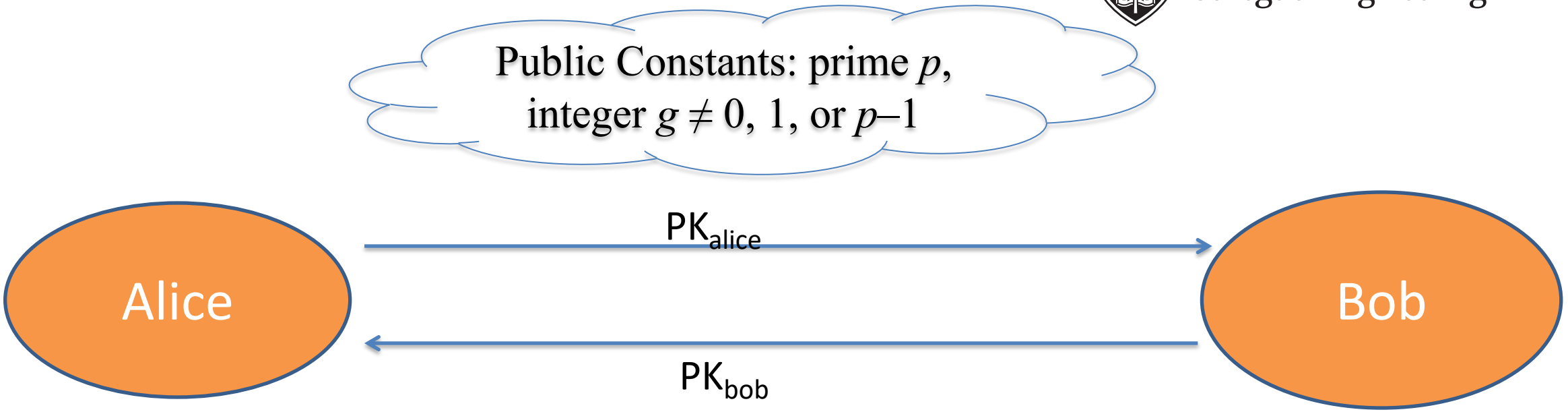
- Given integers n and g and prime number p, compute k such that $n = g^k \bmod p$

- Solutions known for small p
  - *Example: $n = 15$, $g = 2$, and $p = 17$ then $k = 5$*

- Solutions computationally infeasible as p grows large
  - Example: p is a 1024-bit number; ~ order of 10^{300}

Diffie-Hellman Key Exchange

Oregon State University
College of Engineering



Private: ka

Public: $PK_{alice} = g^{ka} \bmod p$

Both Compute $k_{ab} = (PK_{bob})^{ka} \bmod p$

$= (g^{kb} \bmod p)^{ka} \bmod p$

$= g^{kb.ka} \bmod p$

Private: kb

Public: $PK_{bob} = g^{kb} \bmod p$

Both Compute $k_{ab} = (PK_{alice})^{kb} \bmod p$

$= (g^{ka} \bmod p)^{kb} \bmod p$

$= g^{ka.kb} \bmod p$

Example

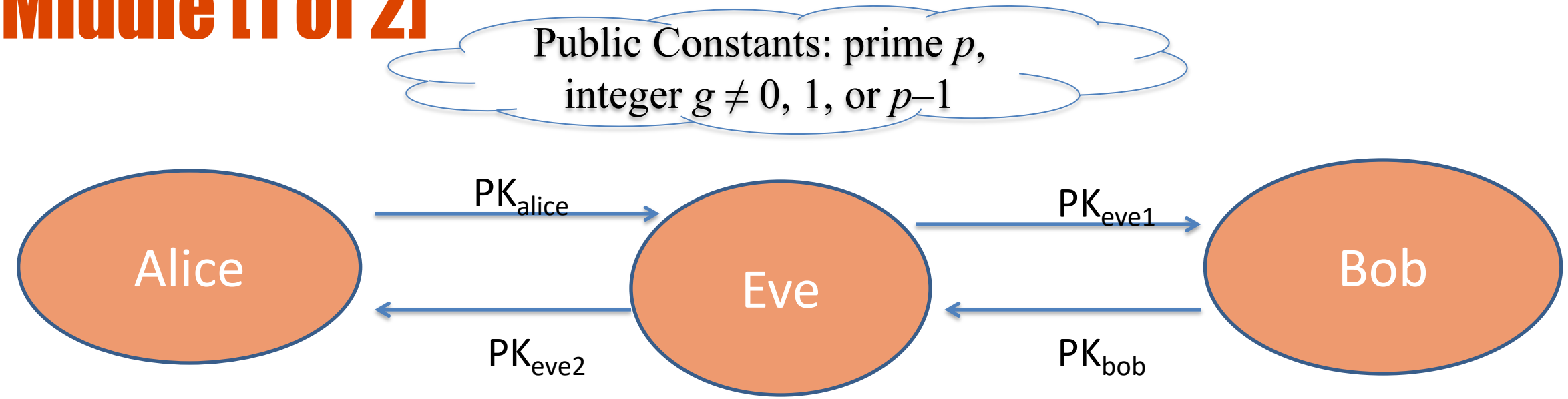
- Assume $p = 53$ and $g = 17$
- Alice chooses $ka = 5$
  - Then $PK_{Alice} = 17^5 \bmod 53 = 40$
- Bob chooses $kb = 7$
  - Then $PK_{Bob} = 17^7 \bmod 53 = 6$
- Shared key:
  - $PK_{Bob}^{ka} \bmod p = 6^5 \bmod 53 = 38$
  - $PK_{Alice}^{kb} \bmod p = 40^7 \bmod 53 = 38$

Real public DH values

- For IPSec and TLS, there are a small set of g's and p's published that all standard implementations support.
  - Group 1 and 2
    - https://tools.ietf.org/html/rfc4306
    - Group 1 Prime
      - FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE65381 FFFFFFFF FFFFFFFF
    - Group 1 generator – 2
  - Group 1 and 2 are not recommended anymore – considered weak as they use 768 and 1024 bit modulii (prime p)
  - Currently used Group 14 (2048 bit p) or Group 19 (256 bit p on elliptic curves) and above

Diffie-Hellman and Man-in-the-Middle [1 of 2]

Oregon State University
College of Engineering



Private: ka

Public: $PK_{alice} = g^{ka} \bmod p$

Both Compute $k_{ae2} = (PK_{eve2})^{ka} \bmod p$

$= (g^{ke2} \bmod p)^{ka} \bmod p$

$= g^{ke2.ka} \bmod p$

$= (g^{ka} \bmod p)^{ke2} \bmod p$

$= (PK_{alice})^{ke2} \bmod p$

Private: kb

Public: $PK_{bob} = g^{kb} \bmod p$

Both Compute $k_{be1} = (PK_{bob})^{ke1} \bmod p$

$= (g^{kb} \bmod p)^{ke1} \bmod p$

$= g^{kb.ke1} \bmod p$

$= (g^{ke1} \bmod p)^{kb} \bmod p$

$= (PK_{eve1})^{kb} \bmod p$

Diffie-Hellman and Man-in-the-Middle [2 of 2]

Oregon State University
College of Engineering

To defend against man-in-the-middle, distribution of public-keys needs authenticity

Key Points

Oregon State University
College of Engineering

- Diffie-Hellman is first (public domain) proposed crypto system
- It enables secure symmetric key exchange
- It is based on the hardness of discrete logarithm problem
- Susceptible to man-in-the-middle

Digital Signatures

Public Key Cryptography

Oregon State University
College of Engineering

- Two keys
  - *Private key* known only to individual
  - *Public key* available to anyone

- Usage:
  - Confidentiality: encipher using public key, decipher using private key
  - Integrity/authentication: encipher using private key, decipher using public one
  - Symmetric Key distribution

Digital Signatures (1 of 5)

Oregon State University
College of Engineering

- Alice wants Bob to know that she sent message, m
  - Sends digital signature along with message
  - m || {h(m)}_d_A or m || {h(m)}_SK_A
  - d_A is short for Alice's RSA private key (p, q, d_A)
  - SK_A represents generic private key of Alice in a asymmetric key-pair
  - {X}_K denotes X enciphered with key K
- How would Bob verify signature?
- Could Eve intercept and change message?
- How does Bob know that Alice is the sender?
- Why do we need the hash function? Can't we just sign m?

Digital Signatures (2 of 5)

Oregon State University
College of Engineering

- How would bob verify the signature?
  - STEP1: Bob takes m (first part of what Alice sent) and computes h(m)
  - STEP2: Bob uses Alice's public key to extract h(m) from {h(m)}_SK_A (second part of what Alice sent)
  - STEP3: Compare h(m) computed in STEP1 with that from STEP2. If they match the signature is valid. Otherwise signature verification fails.

Digital Signatures (3 of 5)

Oregon State University
College of Engineering

- Could Eve intercept and change message?
  - If Eve simply changes m to m’ the quantities computed by Bob in STEP1 (h(m’)) and STEP2 (h(m)) will not match. Signature verification fails.
  - If Eve is able to some how transform $\{h(m)\}\_SK_A$ to $\{h(m')\}\_SK_A$ without knowing SK_A it violates security of the public-key crypto system (i.e., implies Eve is able to break its security)
  - Or if Eve is able to find m’ such that h(m’) = h(m) so that the signature remains valid, this means that Eve broke the security of the cryptographic hash (breaks weak-collision resistance)
- In short: Eve cannot change the message without breaking the hash function or the public-key scheme

Digital Signatures (4 of 5)

Oregon State University
College of Engineering

- How does Bob know Alice is the sender?
  - If the signature verification succeeds with Alice's public-key that implies Alice signed the message
    - Alice's public-key will only correctly decipher messages enciphered by Alice's private key
    - Alice's private-key is only known to Alice

Digital Signatures (5 of 5)

Oregon State University
College of Engineering

- Why do we need the hash function? Can't we just sign m?≠
  - i.e., can't we send just {m}_SK_A
- Two reasons:
  - Efficiency: applying a public-key scheme directly to a large m is going to computationally expensive
  - Security:
    - Consider RSA signatures $\sigma_1 = \{m\}\_SK_A$ and $\sigma_2 = \{m'\}\_SK_A$
    - Then $\sigma_1 * \sigma_2 = \{m\}\_SK_A * \{m'\}\_SK_A = m^{d_A} \bmod N_A * m'^{d_A} \bmod N_A = (m * m')^{d_A} \bmod N_A = \{m * m'\}\_SK_A$
    - An adversary is able to create a valid signature on m*m' without knowing SK_A
    - Using a hash in the signature prevents this --- h(m)*h(m') ≠ h(m*m')

RSA Public-Key System

Public Key Cryptography

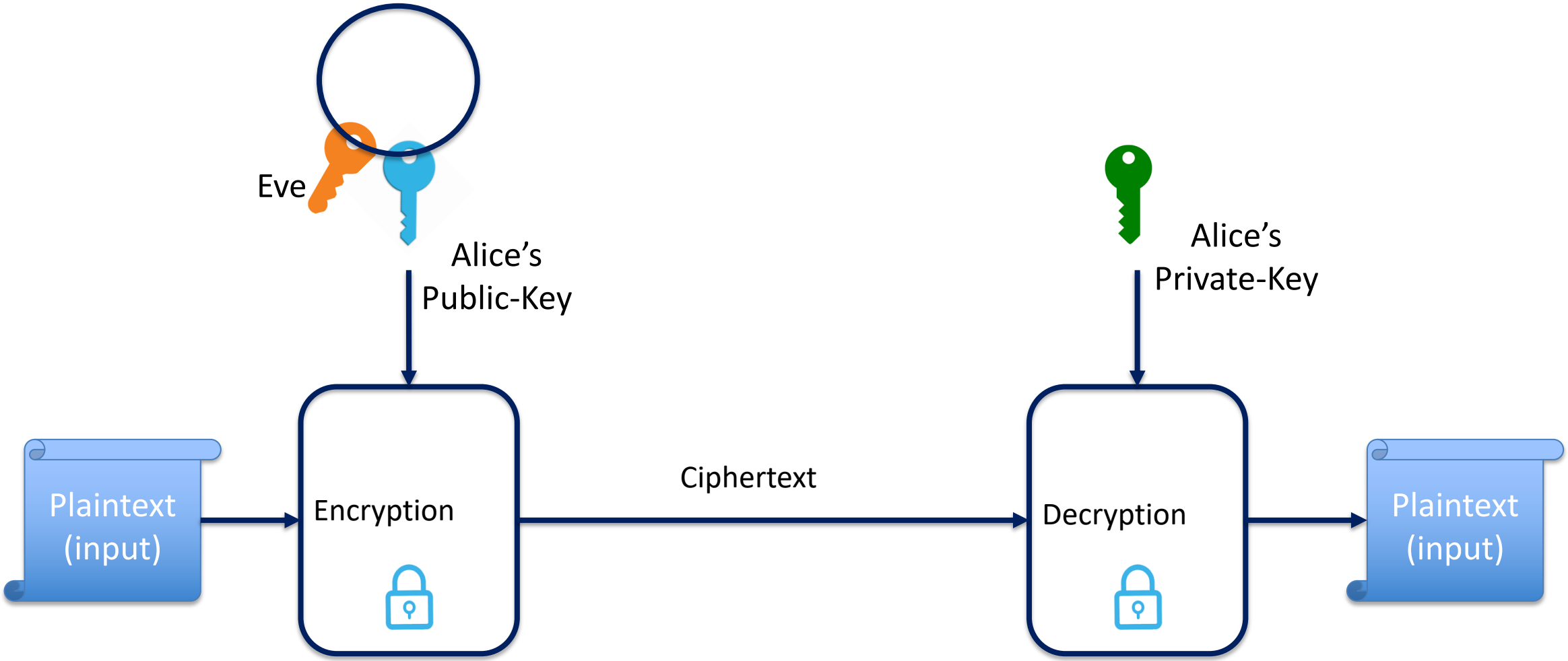
Oregon State University
College of Engineering

- Two keys
  - *Private key* known only to individual
  - *Public key* available to anyone

- Usage:
  - Confidentiality: encipher using public key, decipher using private key
  - Integrity/authentication: encipher using private key, decipher using public one
  - Symmetric Key distribution

Public-Key Cryptography

Oregon State University
College of Engineering



Major Public Key Algorithms

Oregon State University
College of Engineering

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Modular Arithmetic [1 of 2]

Oregon State University
College of Engineering

- a mod b = x if for some k >= 0, and x < b, bk + x = a
  - Example: 25 mod 7 = 4; here k = 3 and 7*3 + 4 = 25

- Associativity, Commutativity, and Distributivity hold in Modular Arithmetic
  - ((a + b) mod n + c) mod n = (a + (b + c) mod n) mod n
  - (a + b) mod n = (b + a) mod n
  - a * (b + c) mod n = (a * b mod n + a * c mod n) mod n

- Inverses also exist in modular arithmetic
  - a + (-a) mod n = 0
  - $a * a^{-1} \bmod n = 1$

Modular Arithmetic [2 of 2]

Oregon State University
College of Engineering

- Reducibility also holds
  - (a + b) mod n = (a mod n + b mod n) mod n
  - a * b mod n = ((a mod n) * (b mod n)) mod n

- Fermat's Thm: if p is any prime integer and a is an integer, then $a^p \bmod p = a$
  - Corollary: $a^{p-1} \bmod p = 1$ if a != 0 and a is relatively prime to p

Totient Function ∅(n)

Oregon State University
College of Engineering

- Number of positive integers less than *n* and relatively prime to *n*
  - *Relatively prime* means with no factors in common with *n*

- Example: ∅(10) = ?
  - 4 because 1, 3, 7, 9 are relatively prime to 10 (no common factors)

- Example: ∅(p) = ? where p is a prime
  - p-1 because all lower numbers are relatively prime

Euler's Theorem

Oregon State University
College of Engineering

- Euler generalized Fermat's Thm for composite numbers.
  - Recall Fermat's Thm $a^{p-1}=1 \bmod p$ if a != 0

- Euler's Thm: $X^{\emptyset(n)}=1 \bmod n$
  - Where q and p are primes
  - $n = pq$
  - $\emptyset(n) = (p-1)(q-1)$

RSA

- Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field over integers modulo a prime
  - nb. exponentiation takes $O((\log n)^3)$ operations (easy)
  - uses large integers (eg. 2048 bits)
- security due to cost of factoring large numbers
  - nb. factorization takes $O(e^{\log n \log \log n})$ operations (hard)

RSA Algorithm

Oregon State University
College of Engineering

- Choose two large prime numbers p, q
  - Let $n = pq$; then $\emptyset(n) = (p-1)(q-1)$
  - Choose $e < n$ such that e is relatively prime to $\emptyset(n)$.
  - Compute d such that $e*d \bmod \emptyset(n) = 1$
- Public key: (e, n); private key: (d, p, q)
- Encipher: $c = m^e \bmod n$
- Decipher: $m = c^d \bmod n$
- Generically: $F(V, x) = V^x \bmod n$

Working through the equations

Oregon State University
College of Engineering

- $C = F(M, e) = M^e \bmod n$
- $M = F(C, d) = F(F(M, e), d) = (M^e \bmod n)^d \bmod n$
- $M = M^{e*d} \bmod n$
  - $e*d \bmod \phi(n) = 1$ ➔ $k * \phi(n) + 1 = e*d$ for some k
- $M = (M \bmod n * M^{k*\phi(n)} \bmod n) \bmod n$
  - By Euler' theorem $X^{\phi(n)} \bmod n = 1$
- $M = M \bmod n$

Where is the security?

Oregon State University
College of Engineering

- What problem must you solve to discover d?
- Public key: (*e*, *n*); private key: (*d, p, q)*

Example: Confidentiality

- Take $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$
- Alice chooses $e = 17$, making $d = 53$
- Bob wants to send Alice secret message HELLO (07 04 11 11 14)
  - $07^{17} \bmod 77 = 28$
  - $04^{17} \bmod 77 = 16$
  - $11^{17} \bmod 77 = 44$
  - $11^{17} \bmod 77 = 44$
  - $14^{17} \bmod 77 = 42$
- Bob sends 28 16 44 44 42

Example: Confidentiality

Oregon State University
College of Engineering

- Alice receives 28 16 44 44 42
- Alice uses private key, $d = 53$, to decrypt message:
  - $28^{53} \bmod 77 = 07$
  - $16^{53} \bmod 77 = 04$
  - $44^{53} \bmod 77 = 11$
  - $44^{53} \bmod 77 = 11$
  - $42^{53} \bmod 77 = 14$
- Alice translates message to letters to read HELLO
  - No one else could read it, as only Alice knows her private key and that is needed for decryption

Example: Integrity/Authentication

Oregon State University
College of Engineering

- Take $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$
- Alice chooses $e = 17$, making $d = 53$
- Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)
  - $07^{53} \bmod 77 = 35$
  - $04^{53} \bmod 77 = 09$
  - $11^{53} \bmod 77 = 44$
  - $11^{53} \bmod 77 = 44$
  - $14^{53} \bmod 77 = 49$
- Alice sends 35 09 44 44 49

Example: Integrity/Authentication

Oregon State University
College of Engineering

- Bob receives 35 09 44 44 49
- Bob uses Alice's public key, $e = 17$, $n = 77$, to decrypt message:
  - $35^{17} \bmod 77 = 07$
  - $09^{17} \bmod 77 = 04$
  - $44^{17} \bmod 77 = 11$
  - $44^{17} \bmod 77 = 11$
  - $49^{17} \bmod 77 = 14$

- Bob translates message to letters to read HELLO
  - Alice sent it as only she knows her private key, so no one else could have enciphered it
  - If (enciphered) message's blocks (letters) altered in transit, would not decrypt to coherent message
    - In practice need to use hash

Warnings

Oregon State University
College of Engineering

- Examples shown for illustration only
- Encipher message in blocks considerably larger than the examples here
  - If 1 character per block, RSA can be broken using statistical attacks (just like classical cryptosystems)
  - Attacker cannot alter letters, but can rearrange them and alter message meaning

- Example: reverse enciphered message of text ON to get NO

Key Points

Oregon State University
College of Engineering

- RSA is a widely used public-key crypto system
- Can be used for encryption, key exchange and digital signatures
- It is based on the hardness of factoring large composites of prime numbers