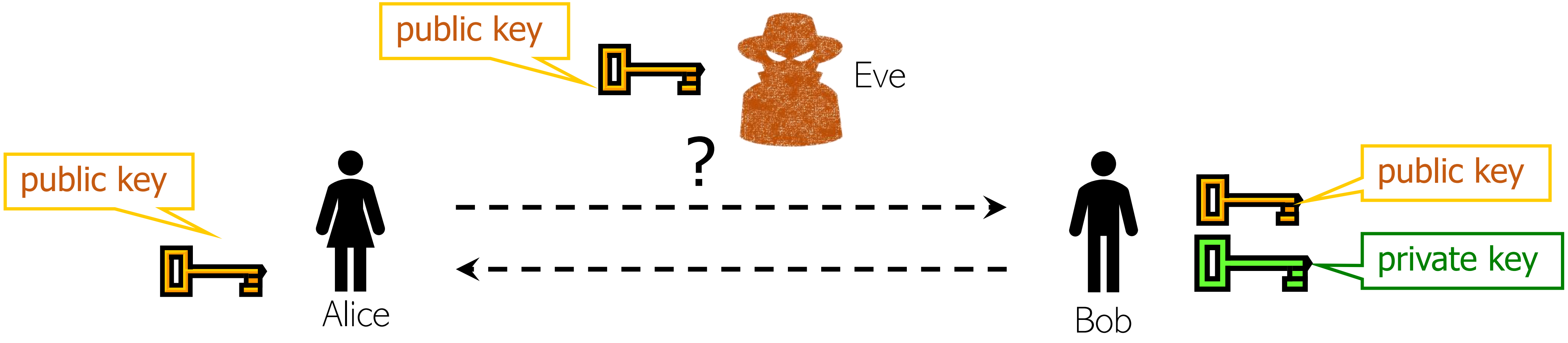
RSA, and Digital Signatures

CS 642: Computer Security and Privacy

Earlence Fernandes

Public-Key/Asymmetric-key cryptography



Given: Everybody knows Bob's public key

- How is this achieved in practice?

Only Bob knows the corresponding private key

Goals: 1. Alice wants to send a message that only Bob can read

2. Bob wants to send a message that only Bob could have written

Applications of Public-Key Crypto

- Encryption for confidentiality
  - Anyone can encrypt a message
    - With symmetric crypto, must know the secret key to encrypt
  - Only someone who knows the private key can decrypt
  - E.g., Emails, messaging
- Digital signatures for authentication
  - Only someone who knows the private key can sign
- Session key establishment
  - Exchange messages to create a secret session key
  - Then switch to symmetric cryptography (why?)

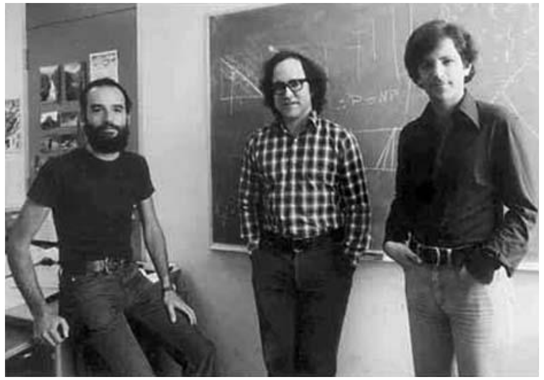
Public-Key Encryption

- Key generation: generate a pair (public key PK, private key SK)
  - Should be computationally easy
- Encryption: given plaintext M and public key PK, easy to compute ciphertext $C=E_{PK}(M)$
- Decryption: given ciphertext $C=E_{PK}(M)$ and private key SK, easy to compute plaintext M
  - Infeasible to learn anything about M from C and PK without SK
  - Trapdoor function: Decrypt(SK, Encrypt(PK, M))=M

Some Number Theory Facts

- Euler totient function $\varphi(n)$ where $n \geq 1$ is the number of integers in the [1,n] interval that are relatively prime to n
  - Two numbers are relatively prime if their greatest common divisor (gcd) is 1
  - Easy to compute for primes $\varphi(p) = p - 1$
  - Note that $\varphi(ab) = \varphi(a)\,\varphi(b)$

RSA Cryptosystem



[Rivest, Shamir, Adleman 1977]

- Key generation:
  - Generate large primes p, q
    - At least 2048 bits each… need primality testing!
  - Compute $n=pq$
    - Note that $\varphi(n)=(p-1)(q-1)$
  - Choose small e, relatively prime to $\varphi(n)$
    - Typically, $e=3$ (may be vulnerable) or $e=2^{16}+1=65537$ (why?)
  - Compute unique d such that $ed \equiv 1 \mod \varphi(n)$
  - Public key = (e,n); private key = d
- Encryption of m: $c = m^e \mod n$
- Decryption of c: $c^d \mod n = (m^e)^d \mod n = m$

Why Is RSA Secure?

- RSA problem: given c, $n=pq$, and e such that $\gcd(e,(p-1)(q-1))=1$, find m such that $m^e=c \bmod n$
  - In other words, recover m from ciphertext c and public key (n, e) by taking e^{th} root of c modulo n
  - There is no known efficient algorithm for doing this

Factoring problem: given positive integer n, find primes $p_1, \ldots, p_k$ such that $n=p_1^{e_1}p_2^{e_2}\ldots p_k^{e_k}$

It is widely *believed* factoring is hard: we don't know how to do it yet.

If factoring is easy, then RSA problem is easy,
but it might be possible to break RSA without factoring n

"Textbook" RSA Is Bad Encryption

- Deterministic
  - Attacker can guess plaintext, compute ciphertext, and compare for equality
  - If messages are from a small set (for example, yes/no), can build a table of corresponding ciphertexts
- Does not provide semantic security (security against chosen-plaintext attacks)
- Can tamper with encrypted messages
- Small e can be dangerous ($\log_e(m^e) \to m$, if e and m are small)
- If ¼-th of the secret key d is leaked, one can recover the full key

Integrity in RSA Encryption

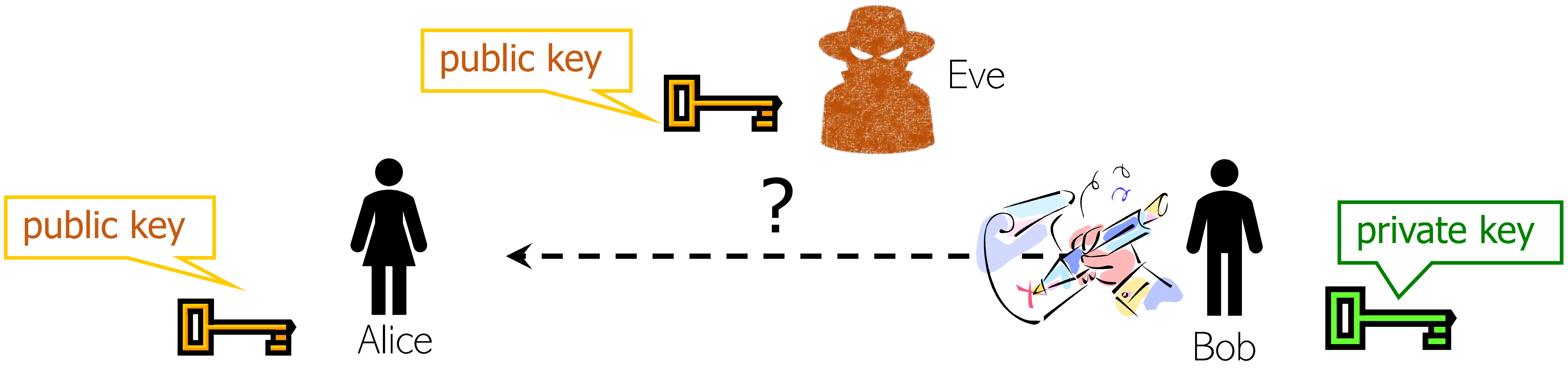
- "Textbook" RSA does not provide integrity
  - Given encryptions of m_1 and m_2, attacker can create encryption of $m_1 \cdot m_2$
    - $(m_1^e) \cdot (m_2^e) \bmod n \equiv (m_1 \cdot m_2)^e \bmod n$
  - Attacker can convert m into m^k without decrypting
    - $(m^e)^k \bmod n \equiv (m^k)^e \bmod n$

No Integrity

- In practice, OAEP is used: instead of encrypting M,
  - encrypt $M \oplus G(r)\ ;\ r \oplus H(M \oplus G(r))$
  - r is random and fresh, G and H are hash functions
  - Resulting encryption is plaintext-aware: infeasible to compute a valid encryption without knowing plaintext
    - … if hash functions are "good" and RSA problem is hard

Digital Signature

Digital Signatures: Basic Idea



Given: Everybody knows Bob's public key
Only Bob knows the corresponding private key

Goal: Bob sends a "digitally signed" message

1. **Only bob can sign**: To compute a signature, must know the private key
2. **Anyone can verify**: To verify a signature, only the public key is needed

RSA Signatures

- Public key is (n, e), private key is d
- To sign message m: $s = (hash(m))^d \bmod n$
  - Signing and decryption are the same mathematical operation in RSA
- To verify signature s on message m:

  $s^e \bmod n = (hash(m)^d)^e \bmod n = hash(m)$
  - Verification and encryption are the same mathematical operation in RSA
- Message must be hashed
  - Roughly, RSA cannot accommodate messages larger than the key
  - In symmetric encryption, we solve using block cipher modes
  - In signature scheme, we solve using cryptographic hash

Cryptography summary

Goal	Tools/techniques
Privacy/Confidentiality	Symmetric keys - One-time pad - Block cipher (e.g., 3DES, AES) → modes: ECB, CBC, CTR Asymmetric keys - RSA
Integrity	- Hash functions (e.g., MD5, SHA-256) - MACs (HMAC, CBC-MAC)
Confidentiality & Integrity	- Authenticated Encryption w/ Associated Data (AEAD) - Encrypt-then-MAC, AEAD
Authenticity & Integrity	- Digital signatures (e.g., RSA, DSS)