Introduction to Information Security

14-741/18-631 Fall 2021

Unit 2: Lecture 2

Asymmetric Key Cryptography

Limin Jia liminjia@andrew

This lecture’s agenda

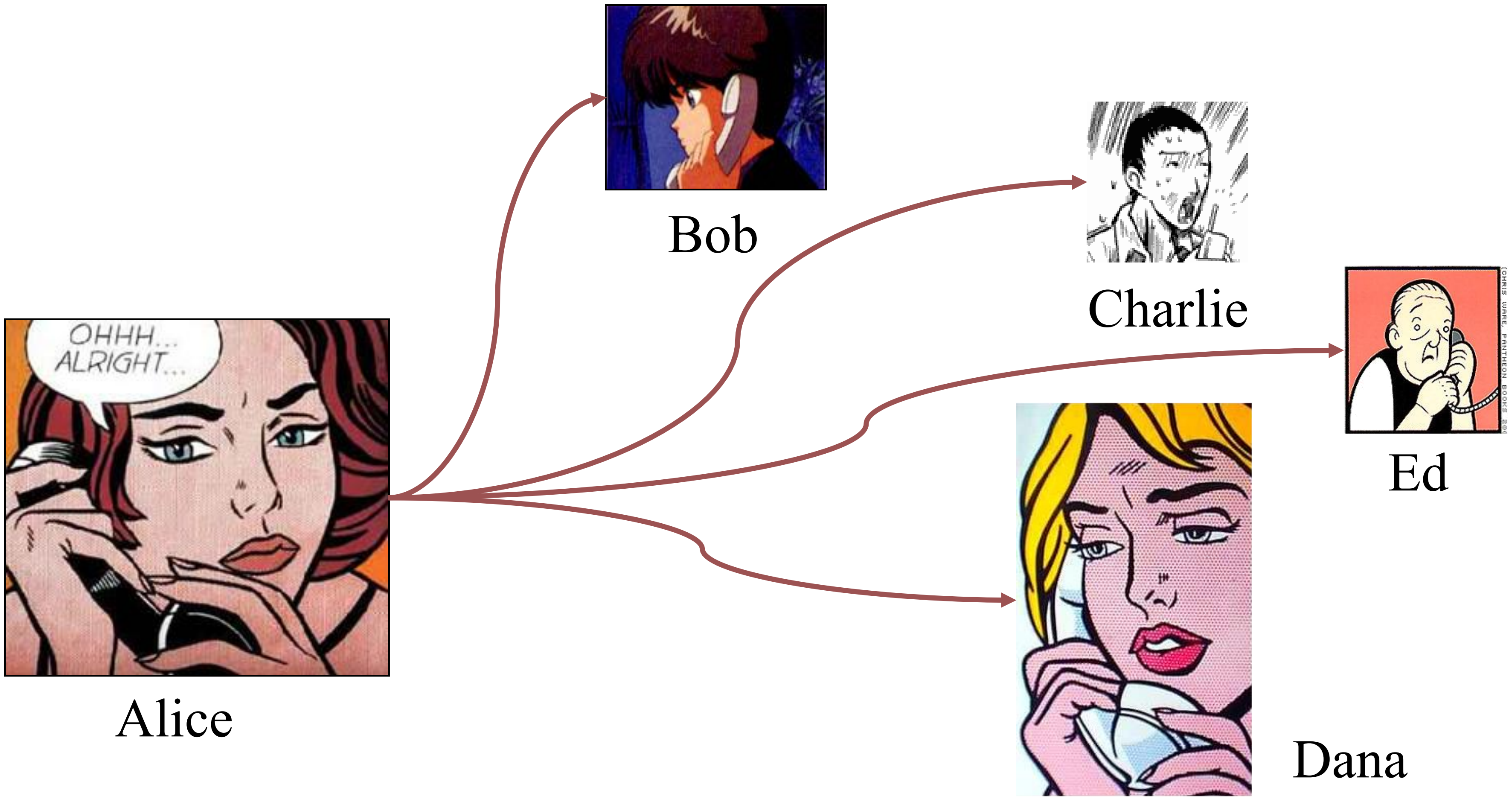
- **Outline**
  - Public key cryptography
    - Diffie-Hellman
    - Public key encryption schemes
    - A concrete implementation: RSA
    - Digital signature schemes
- **Objectives**
  - Continue our overview of basic cryptographic techniques

Difficulties w/ symmetric keys

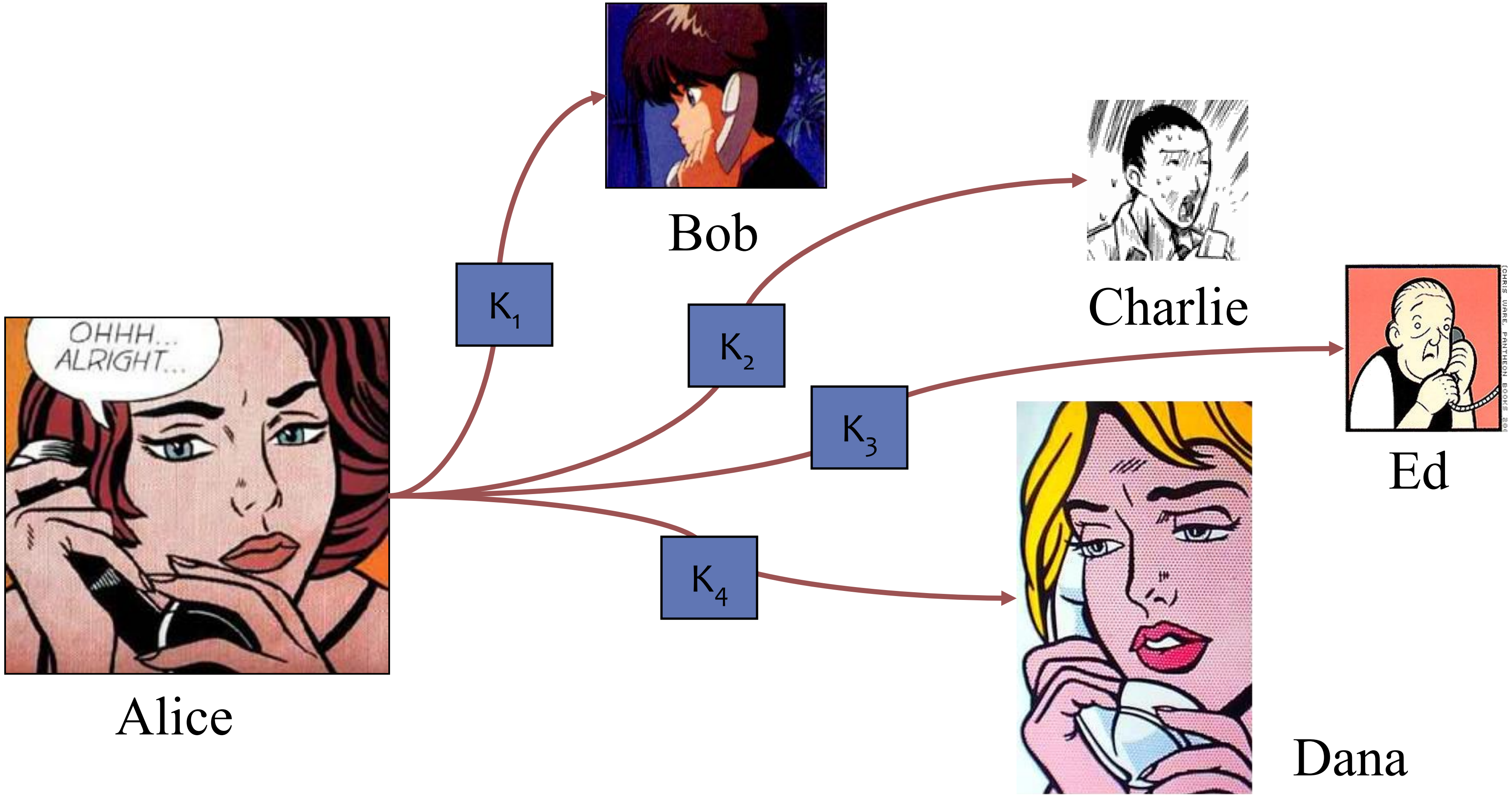
- **Suppose Alice wants to talk to Bob but doesn't want Eve to be able to listen**
- **Symmetric crypto**
  - E.g., DES, AES…

How can Alice and Bob share the secret key?

More difficulties w/ sym. keys



More difficulties w/ sym. keys



More difficulties w/ sym. keys

For *n* participants in the system, you need $O(n^2)$ keys if everyone can talk to everyone else!

Ed

Alice

Dana

Diffie-Hellman-Merkle key exchange

- **Attempts to solve the problem of secret key distribution by having people compute the secret key independently, using publicly available information and personal secrets**
- **Proposed by Diffie & Hellman in 1976**
  - Different way of doing crypto than had been proposed in the previous 4,000+ years
  - Foundation for public key crypto (RSA, ElGamal, etc)
- **Side notes:**
  - Merkle credited by Hellman as a strong inspiration for the design
  - Similar method developed in the 1960s at GCHQ (UK) by James Ellis, but classified…

Merkle, Hellman and Diffie (1977)

Diffie-Hellman-Merkle key exchange

Alice

1. Agree g (base) and p (prime)
2. Make information public (doesn't matter who gets it)

Bob

Diffie-Hellman-Merkle key exchange

3A. Pick secret value *A*

Alice

1. Agree *g* (base) and *p* (prime)
2. Make information public (doesn't matter who gets it)

3B. Pick secret value *B*

Bob

Diffie-Hellman-Merkle key exchange

4A. Send $g^A \bmod p$

Insecure physical channel

3A. Pick secret value *A*

3B. Pick secret value *B*

1. Agree *g* (base) and *p* (prime)
2. Make information public (doesn't matter who gets it)

Alice

Bob

4B. Send $g^B \bmod p$

Diffie-Hellman-Merkle key exchange

4A. Send g^A mod p

Insecure physical channel

3A. Pick secret value A

3B. Pick secret value B

5A. Compute
(g^B mod p)^A mod p = g^{AB} mod p

5B. Compute
(g^A mod p)^B mod p = g^{AB} mod p

Alice

Bob

4B. Send g^B mod p

Diffie-Hellman-Merkle key exchange

4A. Send g^A mod p

Insecure physical channel

3A. Pick secret value A

Alice and Bob
independently computed
the same secret number
by only sharing g and p

d p = g^{AB} mod p

3B. Pick secret value B

5B. Compute
$(g^A \bmod p)^B \bmod p = g^{AB} \bmod p$

Alice

Bob

4B. Send g^B mod p

Why Diffie-Hellman works

Eve

- **Based on hard discrete logarithm problem**
  - Given two large prime numbers g and p, *and* $x = g^A \bmod p$, computing A is very hard
  - The best known algorithm for finding A is **exponential** in time, (i.e., roughly equivalent to a brute force attack)
- **Eve (eavesdropper)**
  - Can easily get $g^A \bmod p$, $g^B \bmod p$
  - But can't compute (easily) $g^{AB} \bmod p$ without A and B
- **Later work on asymmetric key encryption use different hard mathematical problems**

What’s missing?

- **Desired properties:**
  - Only Alice and Bob know K
  - After exchange, if Alice thinks she shares a key K with Bob, then Bob also thinks he shares the same key K with Alice

- **Diffe-Hellman key exchange**
  - Does not provide authentication of the protocol participants

4A. Send g^A mod p

Alice

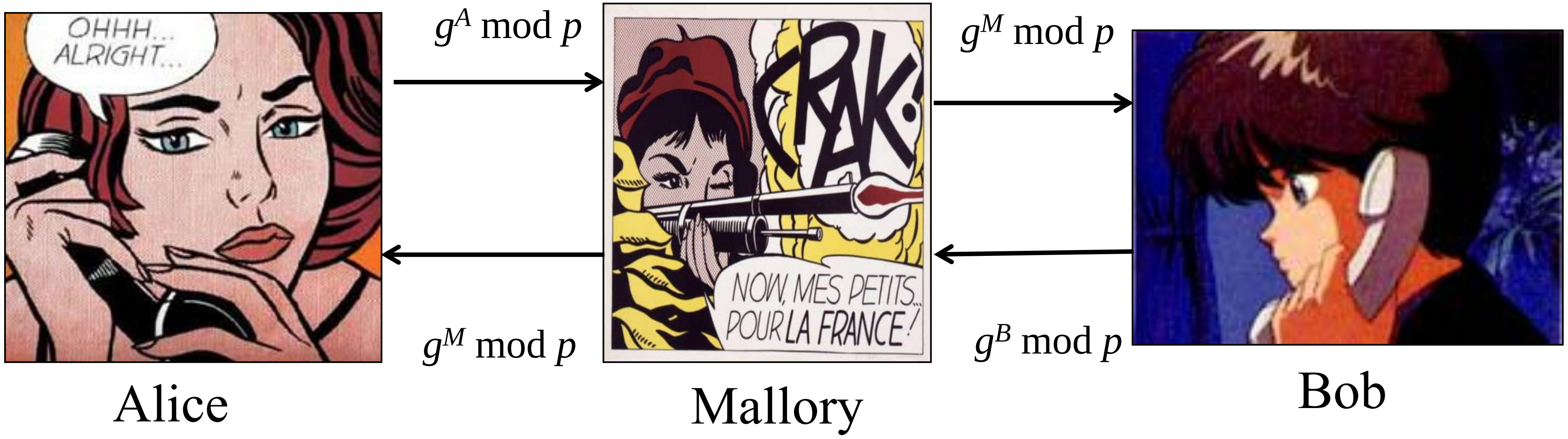
Goal: exchange a shared secret key between Alice and Bob

Bob

4B. Send g^B mod p

Man-in-the-Middle

- **Desired properties:**
  - Only Alice and Bob know K
  - After exchange, if Alice thinks she shares a key K with Bob, then Bob also thinks he shares the same key K with Alice



Outline

- **Diffe-Hellman key exchange**
- **Asymmetric (public) key crypto**
  - Public key encryption schemes
  - A concrete implementation: RSA
  - Digital signature schemes

Public key (asymmetric) crypto

- **Everybody has a key pair: private and public key**
- **Private key is not communicated to anyone**
- **Public key is freely distributed**
- **Allows encryption and authentication**

- **Side note:**
  - Diffie and Hellman **conjectured** this existed

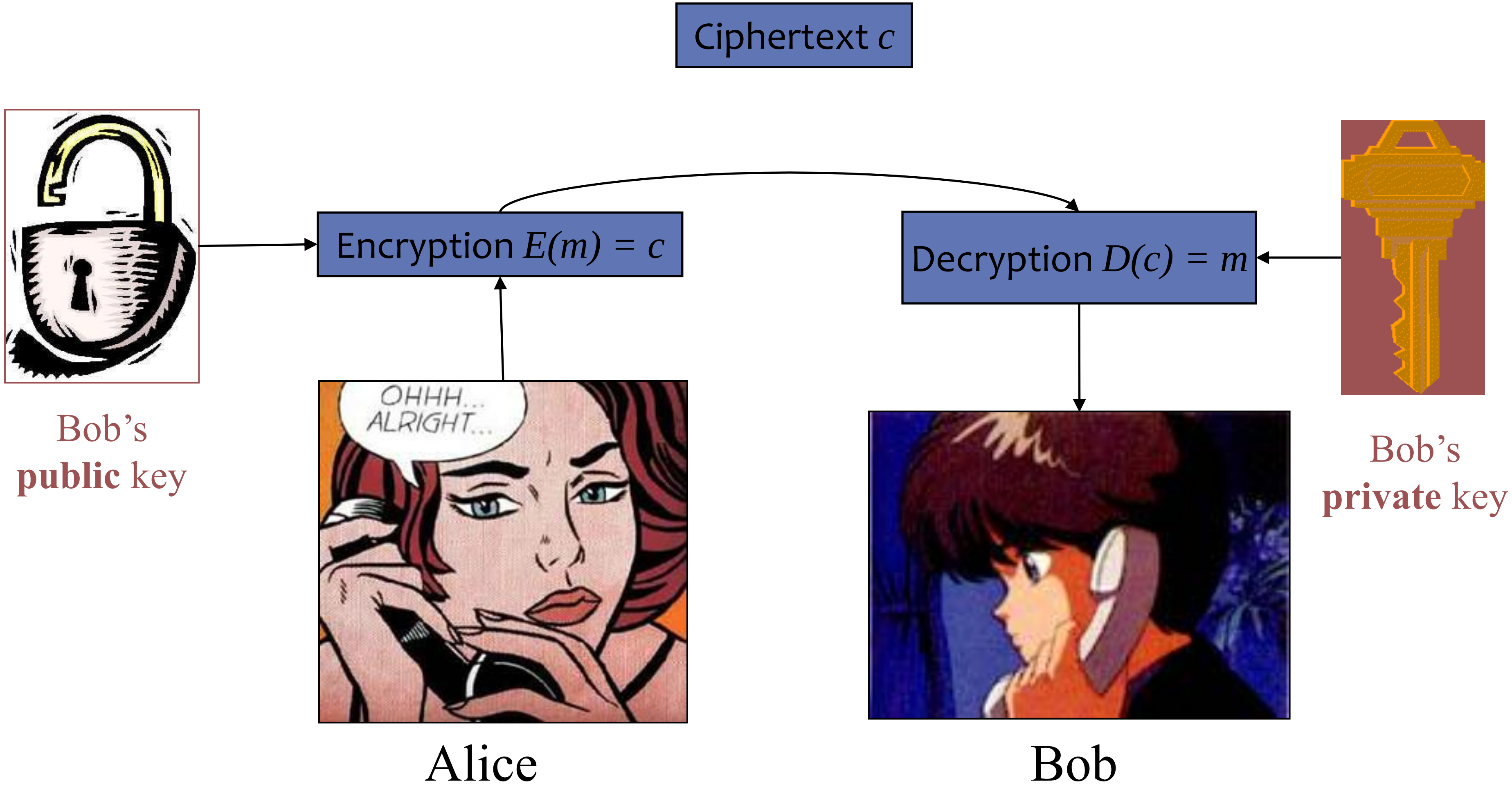
Requirements

- Public (encryption) and private (decryption) keys must be different
- Private key must be impossible (or, more formally, “extremely hard to”) to derive from the public key
- The ciphertext should not reveal anything about the private key
- Must be easy to encrypt/decrypt if knowing the right keys

Informal Definition of Public Key Encryption

- **A public key encryption scheme is a triple $\langle G, E, D\rangle$ of efficiently computable functions**
  - G outputs a "public key" K and a "private key" K^{-1}
    $$\langle K, K^{-1}\rangle \leftarrow G(\cdot)$$
  - E takes public key K and plaintext m as input, and outputs a ciphertext
    $$c \leftarrow E_K(m)$$
  - D takes a ciphertext c and private key K^{-1} as input, and outputs $\perp$ or a plaintext
    $$m \leftarrow D_{K^{-1}}(c)$$
  - If $c \leftarrow E_K(m)$ then $m \leftarrow D_{K^{-1}}(c)$
  - If $c \leftarrow E_K(m)$, then c and K should reveal "no information" about m

Public key encryption



RSA (1975-1978)

- **Developed shortly after Diffie-Hellman paper**
- **Takes its name from the initials of its inventors**
  - Ron **R**ivest
  - Adi **S**hamir
  - Leonard **A**delman
- **Possibly best known public key algorithm**
- **Allows encryption and authentication**
- **Clifford Cocks (w/ James Ellis and Malcolm Williamson), at GCHQ (UK), invented independently a particular case of this method 3 years before RSA, but it was classified by British intelligence**
  - Declassified in 1997

Shamir, Rivest and Adelman ↑
From: http://www.usc.edu/dept/molecular-science/RSApics.htm

Clifford Cocks →
From:http://www.ulm.ccc.de/old/chaos-seminar/krypto2

RSA

- **Key generation:**
  - Choose two large prime numbers p and q such that $p \neq q$, randomly and independently of each other.
  - Pick integer e coprime with (p-1)(q-1) (i.e., gcd(e, (p-1)(q-1)) = 1)
  - Compute d such that

    $ed \equiv 1 \bmod (p-1)(q-1)$ i.e., $ed \bmod (p-1)(q-1) = 1$
  - Private key =(n=pq , d)
  - Public key = (n=pq, e)
- **Encryption:**
  - $E_{(n,\, e)}(m) = m^e \bmod n$
- **Decryption:**
  - $D_{(n,\, d)}(c) = c^d \bmod n$

Why RSA works

- ed mod (p-1)(q-1) = 1
- n = pq
- $E_{(n, e)}(m) = m^e \bmod n$
- $D_{(n, d)}(c) = c^d \bmod n$

- Need $D_K^{-1}(E_K(m)) = m$

- $(m^e \bmod n)^d \bmod n$
  $= m^{ed} \bmod n$
  $= m^{h(p-1)(q-1)+1} \bmod n$
  $= m \bmod n$

Follows from Fermat's little theorem
Or use Chinese remainder theorem

Why RSA works

- **Hard problems:**
  - Integer factorization
    - Given a number n, find its prime factorization, i.e.,

      $$n = p_1^{e_1} p_2^{e_2} p_3^{e_3} p_4^{e_4} \dots$$

    - Computationally infeasible to find large prime factorization of $N = pq$ if p and q are large prime numbers
  - RSA problem:
    - Given $c = m^e \bmod n$ and (n,e), compute m
    - The best algorithm so far is to factor n

A note on RSA

- **Only presented the mathematical intuition**
- **Deploying RSA in practice is nowhere near that simple**
  - You need specific "add-ons" to avoid vulnerabilities (OAEP for encryption)
- **Choosing parameters properly is paramount**
  - Safely choosing and validating primes is mandatory
  - E.g., commonly chosen e=3 turns out to be less secure than previously thought
    - Instantiated as Bleichenbacher attack (2006) against Firefox
    - Now 65537 is recommended

- **Properly using RSA in practice requires more study/effort**

More Attacks on RSA (don't be naive!)

- **Don't pick e=3 (Hastad's Broadcast attack)**
  - If you get three identical messages to different people
  - $C1=M^3 \bmod N1$, $C2 = M^3 \bmod N2$, $C3 = M^3 \bmod N3$
  - Chinese remainder theorem gives $C' = M^3 \bmod N1*N2*N3 = M^3$
  - $M^3 < N1*N2*N3$ so M = cube root of C' (because $M < N1, N2, N3$)

More Attacks on RSA (don’t be naive!)

- **Timing Attacks**
  - Powermod algorithm uses repeated squaring and multiplication
  - Measure time to figure out if multiplications occur
- **Power Attacks**
  - Measure smartcard power consumption during signature generation

Digital Signatures (Informal Definition)

- **A digital signature scheme is a triple <G, S, V> of efficiently computable algorithms**
  - G outputs a "public key" K and a "private key" K^{-1}

$$< K, K^{-1} > \leftarrow G(\cdot)$$

  - S takes a "message" m and K^{-1} as input and outputs a "signature" σ

$$\sigma \leftarrow S_{K^{-1}}(m)$$

  - V takes a message m, signature σ and public key K as input, and outputs a bit b

$$b \leftarrow V_K(m, \sigma)$$

  - If $\sigma \leftarrow S_{K^{-1}}(m)$ then $V_K(m, \sigma)$ outputs 1 ("valid")
- **Security requirement**
  - Given only K and message/signature pairs $\{<m_i, S_{K^{-1}}(m_i)>\}_i$, it is computationally infeasible to compute $<m, \sigma>$ such that

    $V_K(m, \sigma) = 1$

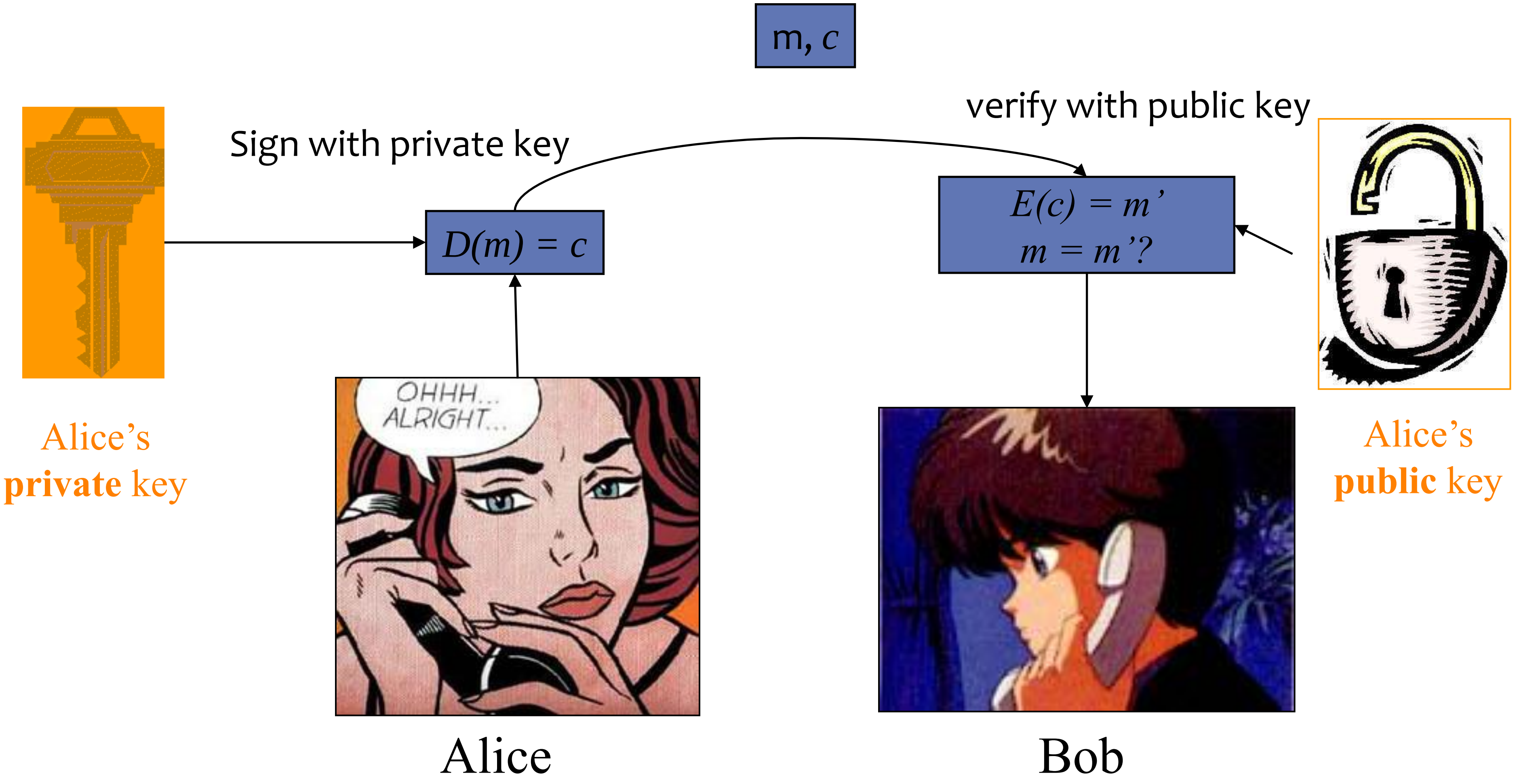
    for any new $m \neq m_i$

Digital Signatures (Public key authentication)

- **Scenario:**
  - Alice signs a message M with her **private** key
  - Bob can verify that M comes from Alice using Alice's **public** key
  - No one but Alice could sign the message that way

    (duplicating a private key is impossible unless the key is leaked)
- **Very effective defense against man-in-the middle attacks**
  - But you need a trusted way to verify keys (e.g. certificate authority that signs them)

Public key authentication (e.g., RSA)



Digital signatures compromises

- **Existential forgery**
  - The attacker manages to forge a signature of (at least) one message, but not necessarily of his choice
- **Selective forgery**
  - The attacker manages to forge a signature of (at least) one message of his choice
- **Universal forgery**
  - The attacker manages to forge a signature of any message
- **Total break**
  - The attacker can compute the signer’s private key

Comparison sym vs. asym crypto

Symmetric crypto (AES)

- Need shared secret
- 256-bit key for high security
- 1,000,000 ops/s on a 1 GHz processor
- >100x speedup in hardware

Asymmetric crypto*

- Need authentic public key
- 2048-bit key (RSA)
- 100 signatures/s and 1,000 verifications/s (RSA) on 1 GHz processor
- ~ 10x speedup in hardware

* Excludes Elliptic Curve Crypto

(With thanks to Adrian Perrig for this slide)

Take away slide

- **Exchanging secret keys is difficult, and doesn't scale well**
- **Diffie-Hellman-Merkle key exchange protocol makes each party independently compute the secret key based on**
    - publicly available information (g, p),
    - their own secret (A and B)
    - partial information about the other party's secret
  - Scheme does not support authentication
- **Public key crypto**
  - Builds on Diffie-Hellman-Merkle's ideas
  - Provides encryption **and** authentication
    - Encryption: use the recipient's public key
    - Authentication: use your private key
  - Much slower than symmetric cryptography
  - Must be careful with implementation!
- **Digital signatures**
  - Rely on public key crypto
  - Useful for authentication, and to thwart man-in-the-middle attacks