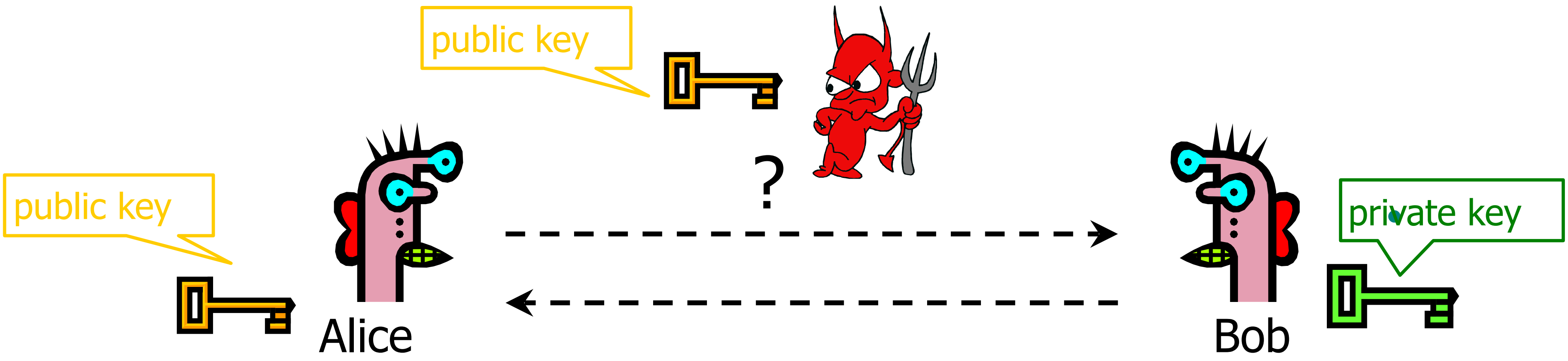
Public Key Cryptography (I)

> The era of "electronic mail" [Potter1977] may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are private, and (b) messages can be signed.
>
> R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. January 1978.

Yan Huang

Credits: David Evans,
Vitaly Shmatikov

Public-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

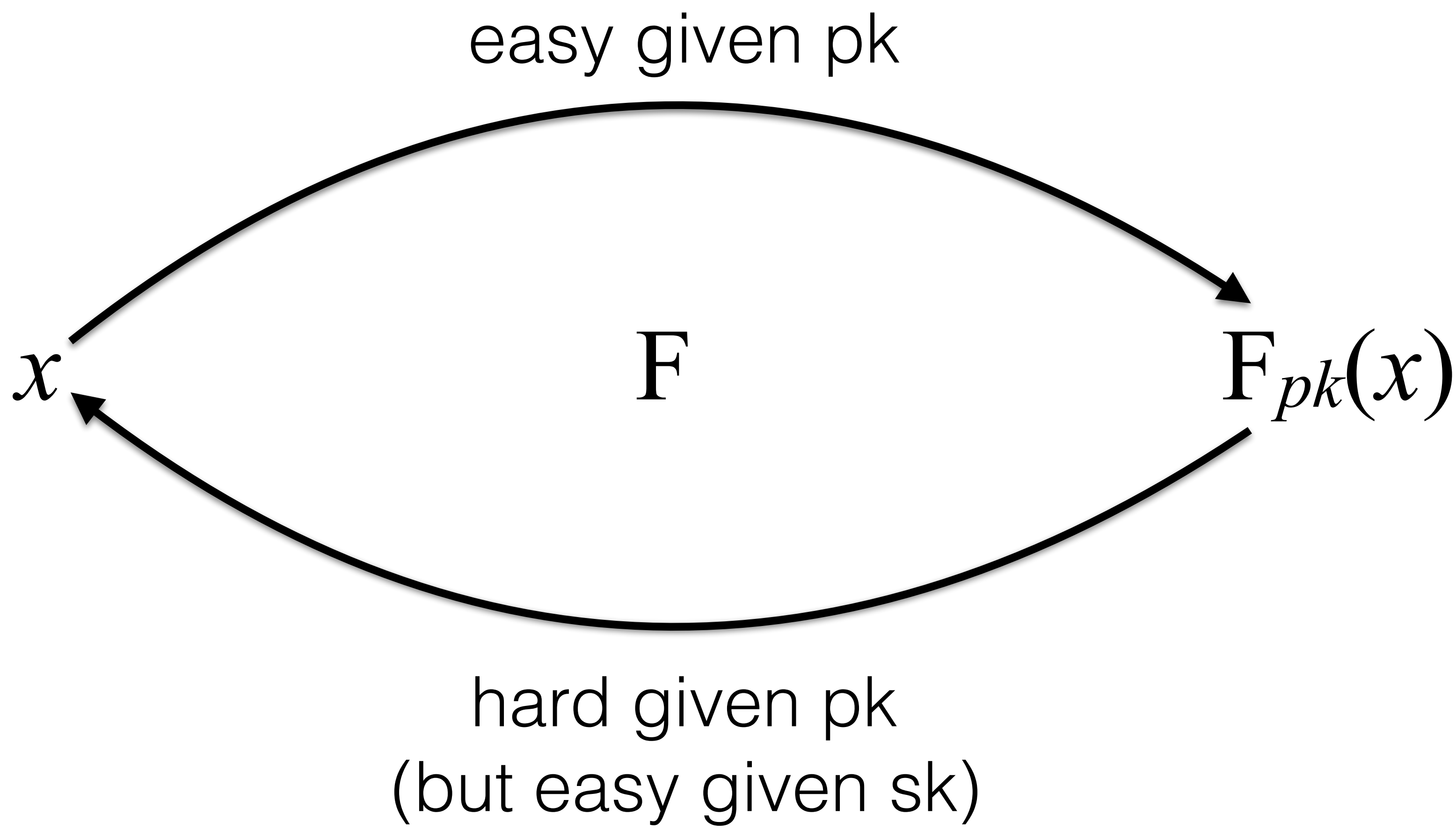
Applications of Public-Key Crypto

- Public key crypto as a solution to key management
- Encryption for confidentiality
  - Anyone can encrypt a message
  - Only someone who knows the private key can decrypt
  - Secret keys are only stored in one place
- 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: *computationally easy* to generate a pair (public key PK, private key SK)
- Encryption: given plaintext M and public key PK, easy to compute ciphertext $C=E_{PK}(M)$
- Decryption: given ciphertext $M=D_{SK}(C)$ and private key SK, easy to compute plaintext M
  - Infeasible to learn anything about M without SK
  - <u>Trapdoor</u> function: Decrypt(SK,Encrypt(PK,M))=M

Trapdoor functions



Some Number Theory Facts

- Euler totient function φ(n) where n≥1, is the number of integers in the interval [1,n] that are relatively prime to n
  - x and y are relatively prime if gcd(x, y) = 1
  - φ(n) is also the *size* of $\mathbb{Z}_n^*$

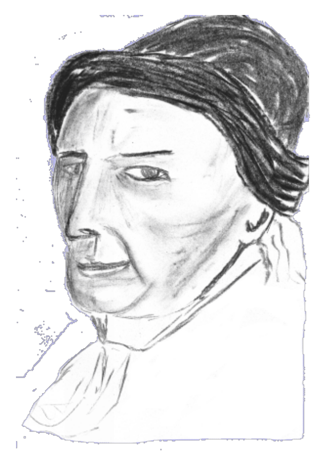
$$\mathbb{Z}_7^* = \{1,2,3,4,5,6\}, \quad \varphi(7) = \| \mathbb{Z}_7^* \| = 6$$

$$\mathbb{Z}_{15}^* = \{1,2,4,7,8,11,13,14\}, \quad \varphi(15) = \varphi(3 \cdot 5) = \| \mathbb{Z}_{15}^* \| = (3-1)\cdot(5-1) = 8$$

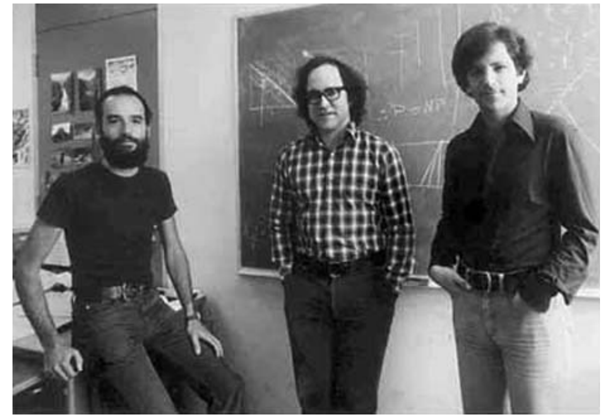
$$\varphi(n) = n \prod_{p|n,\, p:prime} (1 - 1/p)$$

- Euler's theorem:

$$\text{If } a \in \mathbb{Z}_n^*, \text{ then } a^{\phi(n)} \equiv 1 \bmod n$$



RSA Cryptosystem



[Rivest, Shamir, Adleman 1977]

- Key generation:
  - Generate large primes p, q
    - At least 1024 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 $ed \equiv 1 \bmod \varphi(n)$
  - Public key = (n,e); private key = d
- Encryption of m: $c = m^e \bmod n$
- Decryption of c: $c^d \bmod n = (m^e)^d \bmod n = m$

Why RSA Decryption Works

Because

$$e \cdot d \equiv 1 \bmod \varphi(n),$$

thus there exists integer k such that

$$e \cdot d = 1 + k \cdot \varphi(n)$$

So $m^{ed} \equiv m^{1+k \cdot \varphi(n)} \equiv m \mod n$. (Euler's theorem)

Why Is RSA Secure?

- RSA Problem: given c, $n=pq$, and e such that $\gcd(e, (p-1)(q-1))=1$, find an e^{th} root of c modulo n.
- RSA Assumption: there is no *efficient* algorithm to solve RSA problem.
- Factoring problem: given positive integer $n=pq$ where p, q are large primes (thousands of bits), factor n.
- If factoring is easy, then RSA problem is easy, but may be possible to break RSA without factoring n