CSE 484 / CSE M 584: Asymmetric Cryptography

Fall 2023

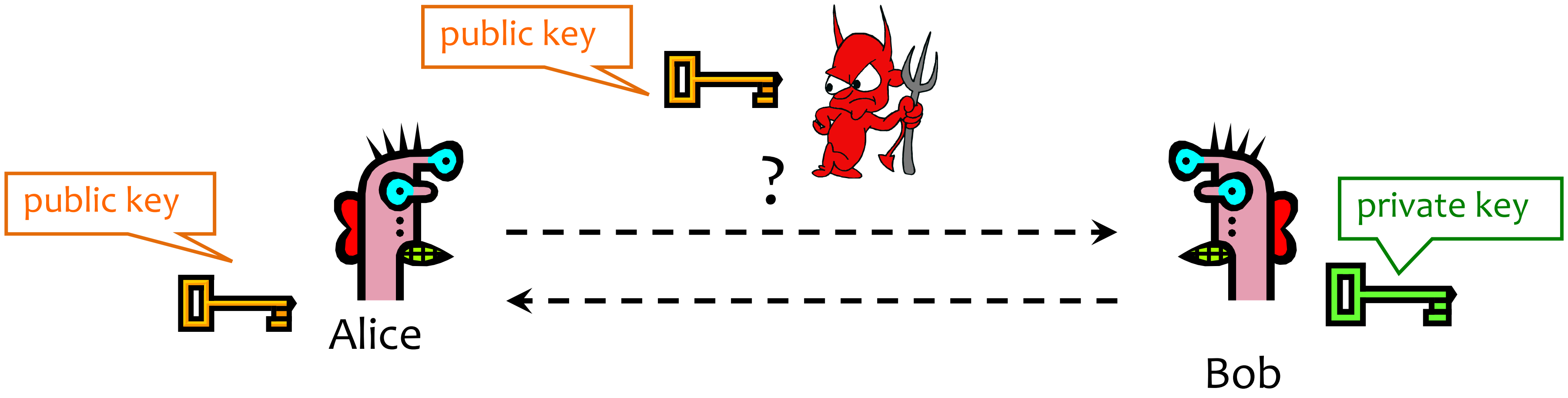
Franziska (Franzi) Roesner
franzi@cs

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Announcements

- Things due
  - Lab 1: today
    - Extra office hours with Kirsten 10:30am Thursday
  - Homework 2: Next Friday
    - Individual assignment (no groups)
  - CSE 584M: Don't forget about weekly research readings
- Roadmap
  - Today: Asymmetric crypto
  - Friday: Crypto in practice (on the web)
  - Next week: Web security

Public Key Crypto: Basic Problem



Ignore for now: How do we know it's REALLY Bob's??

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

Goals: 1. Alice wants to send a secret message to Bob
2. Bob wants to authenticate themself

Some Number Theory Facts

- Euler totient function $\varphi(n)$ $(n \geq 1)$ 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 $\varphi(ab) = \varphi(a)\,\varphi(b)$ if a & b are relatively prime

RSA Cryptosystem [Rivest, Shamir, Adleman 1977]

- Key generation:
  - Generate large primes p, q
    - Say, 2048 bits each (need primality testing, too)
  - Compute **n**=pq and φ(**n**)=(p-1)(q-1)
  - Choose small **e**, relatively prime to φ(n)
    - Typically, **e=3** or **$e=2^{16}+1=65537$**
  - Compute unique **d** such that $ed \equiv 1 \bmod \varphi(n)$
    - Modular inverse: $d \equiv e^{-1} \bmod \varphi(n)$
  - Public key = (e,n); private key = (d,n)
- Encryption of m: $c = m^e \bmod n$
- Decryption of c: $c^d \bmod n = (m^e)^d \bmod n = m$

How to compute?

- Extended Euclidian algorithm
- Wolfram Alpha ☺
- Brute force for small values

Why RSA Decryption Works (FYI)

$e \cdot d = 1 \bmod \phi(n)$, thus $e \cdot d = 1 + k \cdot \phi(n)$ for some k

Let m be any integer in Z_n^* (not all of Z_n)

$c^d \bmod n = (m^e)^d \bmod n = m^{1+k \cdot \varphi(n)} \bmod n$

$= (m \bmod n) * (m^{k \cdot \varphi(n)} \bmod n)$

Euler's theorem: if $a \in Z_n^*$, then $a^{\phi(n)} = 1 \bmod n$

$c^d \bmod n = (m \bmod n) * (1 \bmod n)$

$= m \bmod n$

Proof omitted (using Chinese Remainder Theorem):

True for all m in Z_n, not just m in Z_n^*

Why is RSA Secure?

- RSA problem: given c, $n=pq$, and e such that $\gcd(e, \varphi(n))=1$, 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 *without* knowing p and q
- Factoring problem: given positive integer n, find primes $p_1, \ldots, p_k$ such that $n=p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$

- If factoring is easy, then RSA problem is easy
  (knowing factors means you can compute d = inverse of e mod (p-1)(q-1))
  - It may be possible to break RSA without factoring n – but if it is, we don't know how

RSA Encryption Caveats (1)

- Encrypted message needs to be interpreted as an integer less than n
- Don't use RSA **directly** for privacy – output is deterministic! Need to pre-process input somehow.
- Plain RSA also does not provide integrity
  – Can tamper with encrypted messages

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

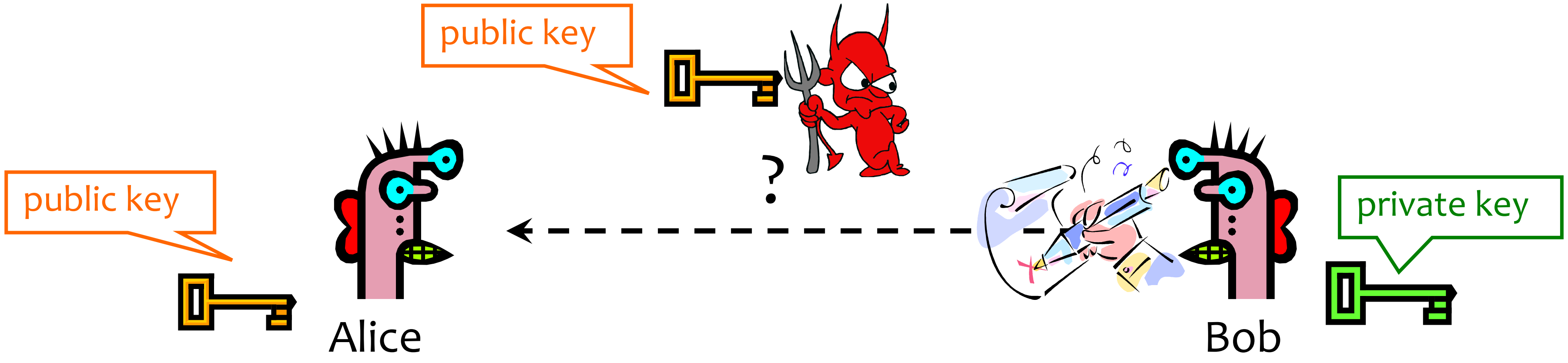
RSA Encryption Caveats (2)

- **There are better options now**
  - E.g., based on elliptic curve cryptography
- Lots of practical issues: https://blog.trailofbits.com/2019/07/08/fuck-rsa/
  - "While it may be theoretically possible to implement RSA correctly, decades of devastating attacks have proven that such a feat may be unachievable in practice."

Stepping Back: Asymmetric Crypto

- Last time we saw session key establishment (Diffie-Hellman)
  - Can then use shared key for symmetric crypto
- We just saw: public key encryption
  - For confidentiality
- Finally, now: digital signatures
  - For authenticity

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. To compute a signature, must know the private key
2. To verify a signature, only the public key is needed

RSA Signatures

- Public key is **(n,e)**, private key is **(n,d)**
- To sign message m: s = m^d mod n
  - Signing & decryption are same **underlying** operation in RSA
  - It's infeasible to compute **s** on **m** if you don't know **d**
- To verify signature s on message m: verify that s^e mod n = $(m^d)^e$ mod n = m
  - Just like encryption (for RSA primitive)
  - Anyone who knows **n** and **e** (public key) can verify signatures produced with d (private key)
- In practice, also need padding & hashing
  - Without padding and hashing: Consider multiplying two signatures together
  - Standard padding/hashing schemes exist for RSA signatures

DSS Signatures

- Digital Signature Standard (DSS)
  - U.S. government standard (1991, most recent rev. 2013)
- Public key: (p, q, g, $y=g^x$ mod p), private key: x
- Security of DSS requires hardness of discrete log
  - If could solve discrete logarithm problem, would extract x (private key) from g^x mod p (public key)
- Again: We've discussed discrete logs modulo integers; significant advantages to using elliptic curve groups instead.

Post-Quantum Cryptography

- If quantum computers become a reality
  - It becomes much more efficient to break conventional asymmetric encryption schemes (e.g., factoring becomes "easy")
- There exists efforts to make quantum-resilient asymmetric encryption schemes

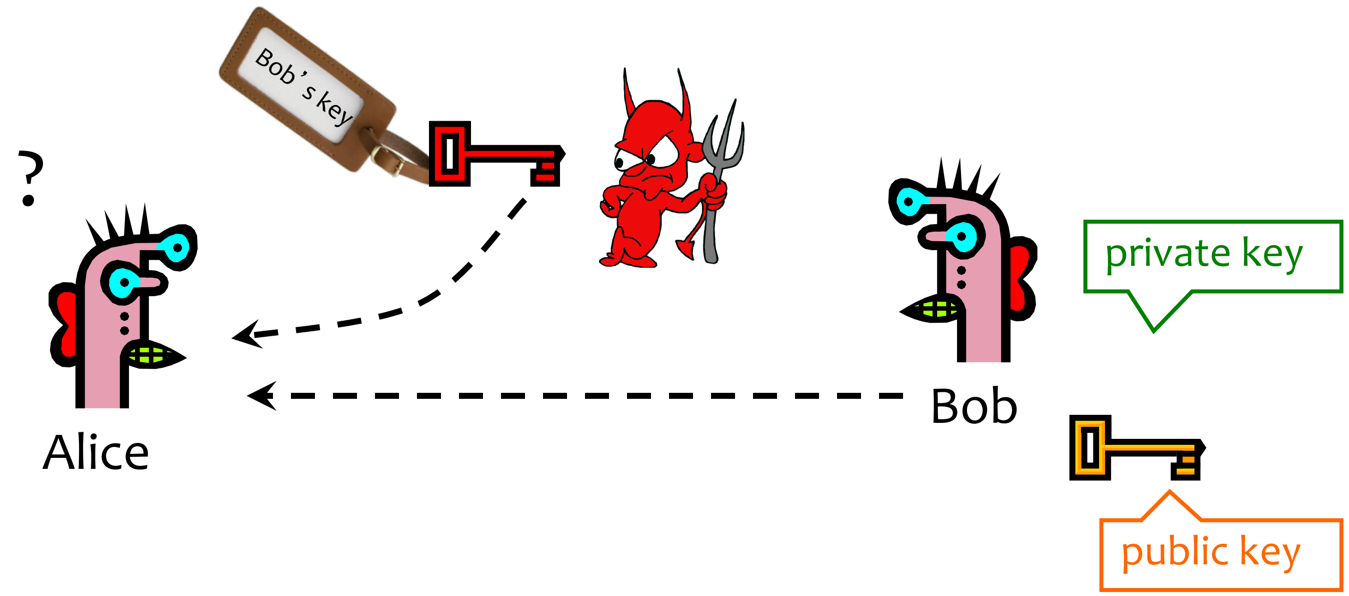
Cryptography Summary

- Goal: Privacy
  - Symmetric keys:
    - One-time pad, Stream ciphers
    - Block ciphers (e.g., DES, AES) → modes: ECB, CBC, CTR
  - Public key crypto (e.g., Diffie-Hellman, RSA)
- Goal: Integrity
  - MACs, often using hash functions (e.g, SHA-256)
- Goal: Privacy and Integrity ("authenticated encryption")
  - Encrypt-then-MAC
- Goal: Authenticity (and Integrity)
  - Digital signatures (e.g., RSA, DSS)

Want More Crypto?

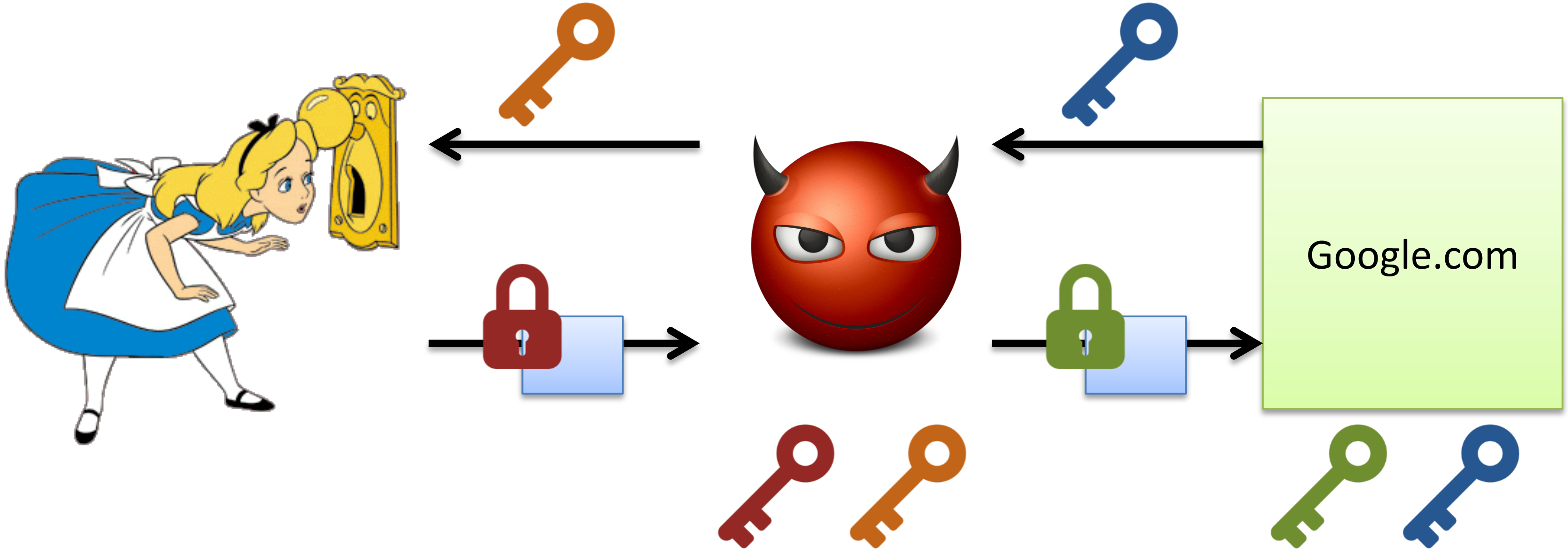
- Some suggestions:
  - CSE 426: Cryptography
  - Stanford Coursera (Dan Boneh): https://www.coursera.org/learn/crypto
- **Mid-quarter check-in:**
  - Questions / thoughts? See Canvas "Quiz" Activity

Authenticity of Public Keys



Problem: How does Alice know that the public key she received is really Bob's public key?

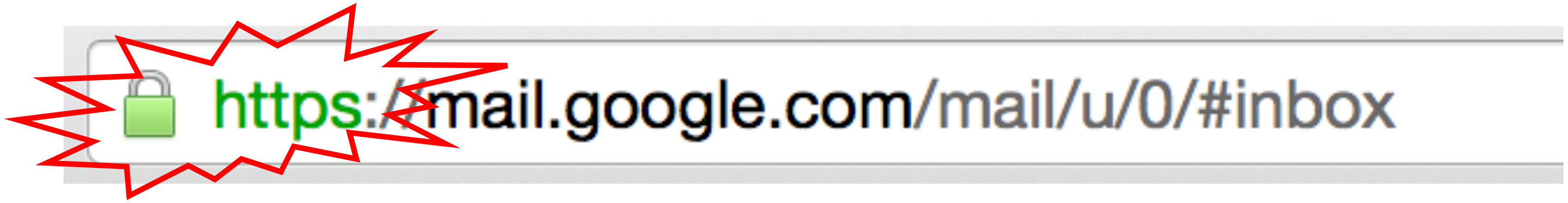
Threat: Person-in-the Middle



Distribution of Public Keys

- Public announcement or public directory
  - Risks: forgery and tampering
- Public-key certificate
  - Signed statement specifying the key and identity
    - sig_{CA}("Bob", PK_B)
- Common approach: certificate authority (CA)
  - Single agency responsible for certifying public keys
  - After generating a private/public key pair, user proves his identity and knowledge of the private key to obtain CA's certificate for the public key (offline)
  - Every computer is <u>pre-configured</u> with CA's public key

You encounter this every day…



SSL/TLS: Encryption & authentication for connections

SSL/TLS High Level

- SSL/TLS consists of two protocols
  - Familiar pattern for key exchange protocols
- Handshake protocol
  - Use public-key cryptography to establish a shared secret key between the client and the server
- Record protocol
  - Use the secret symmetric key established in the handshake protocol to protect communication between the client and the server

Example of a Certificate