CSE 484 / CSE M 584: Computer Security and Privacy

Cryptography
[Finish Asymmetric Cryptography]

Spring 2019

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

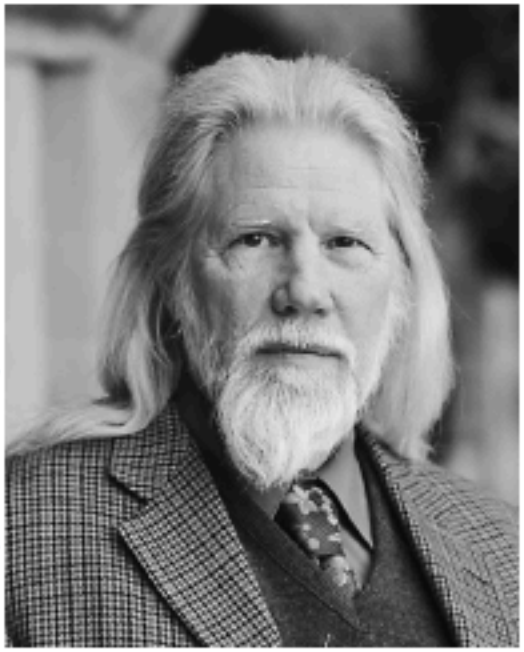
Admin

- Lab 1 done ☺
- Homework 2 (crypto) out
  - Get started now; shouldn't need the whole time
- Lab 2 (web security) out next week
  - Stay tuned for group signup instructions

More on Diffie-Hellman Key Exchange

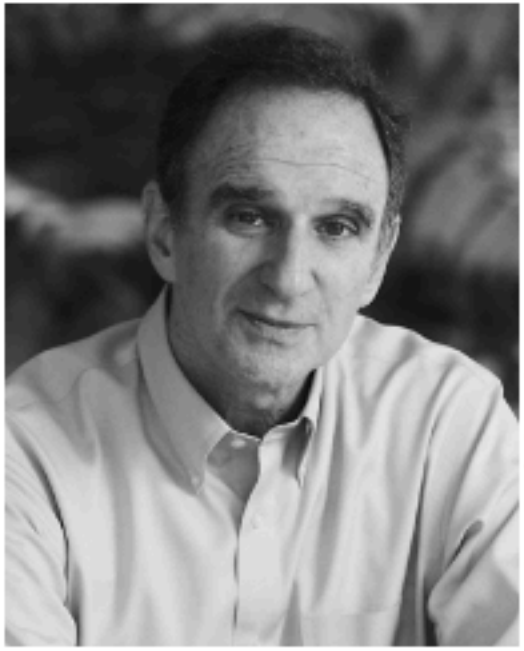
- Important Note:
  - We have discussed discrete logs modulo integers
  - Significant advantages in using elliptic curve groups
    - Groups with some similar mathematical properties (i.e., are "groups") but have better security and performance (size) properties

Diffie and Hellman Receive 2015 Turing Award



Rod Searcey/Stanford University.

Whitfield Diffie



Linda A. Cicero/Stanford News Service.

Martin E. Hellman

Public Key Encryption

Requirements for 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 $C=E_{PK}(M)$ and private key SK, easy to compute plaintext M
  - Infeasible to learn anything about M from C without SK
  - Trapdoor function: Decrypt(SK,Encrypt(PK,M))=M

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 $\varphi(ab) = \varphi(a)\,\varphi(b)$

RSA Cryptosystem [Rivest, Shamir, Adleman 1977]

- Key generation:
  - Generate large primes p, q
    - Say, 1024 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 ≡ 1 mod φ(n)
    - Modular inverse: $d \equiv e^{-1} \bmod \varphi(n)$
  - Public key = (e,n); private key = (d,n)

How to compute?

- 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 (FYI)

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

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

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

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

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

Proof omitted: True for all m in Z_n, not just m in Z_n^*

Why RSA Decryption Works (FYI)

- Decryption of c: $c^d \bmod n = (m^e \bmod n)^d \bmod n = (m^e)^d \bmod n = m$
- Recall **n**=pq and $\varphi(\mathbf{n})=(p-1)(q-1)$ and $ed \equiv 1 \bmod \varphi(n)$

- Chinese Remainder Theorem: To show $m^{ed} \bmod n \equiv m \bmod n$, sufficient to show:
  - $m^{ed} \bmod p \equiv m \bmod p$
  - $m^{ed} \bmod q \equiv m \bmod q$

- If $m \equiv 0 \bmod p \rightarrow m^{ed} \equiv 0 \bmod p$

- Else $m^{ed} = m^{ed-1}m = m^{k(q-1)(p-1)}m = m^{h(p-1)}m$ for some k, and h=k(q-1). Why? Recall how d was chosen and the definition of mod.
- Fermat Little Theorem: $m^{(p-1)h}m \equiv 1^h m \bmod p \equiv m \bmod p$

Why is RSA Secure?

- RSA problem: given c, n=pq, and e such that gcd(e, φ(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
- 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

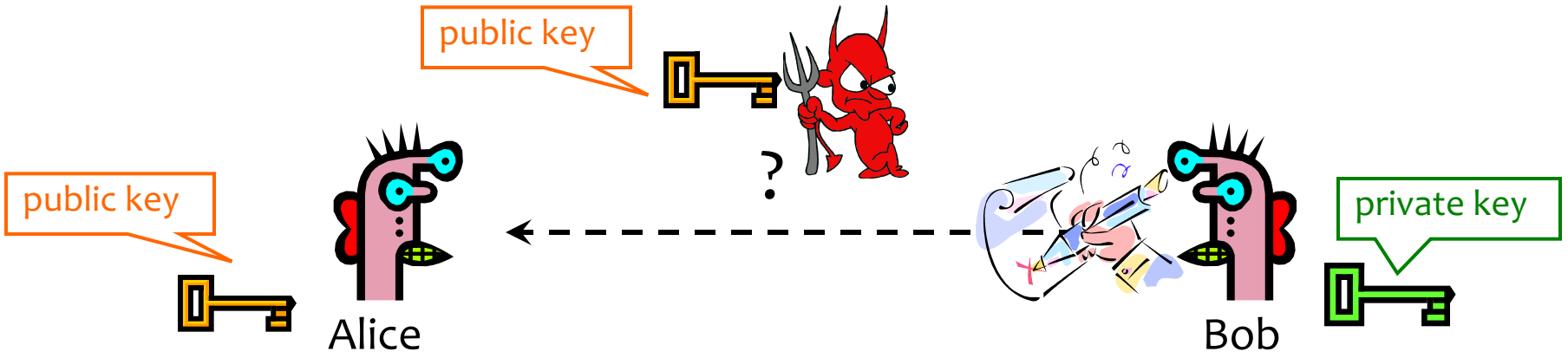
RSA Encryption Caveats

- 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

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: $s = m^d \bmod n$
  - 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 $s^e \bmod n = (m^d)^e \bmod n = m$
  - 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
  - Standard padding/hashing schemes exist for RSA signatures

DSS Signatures

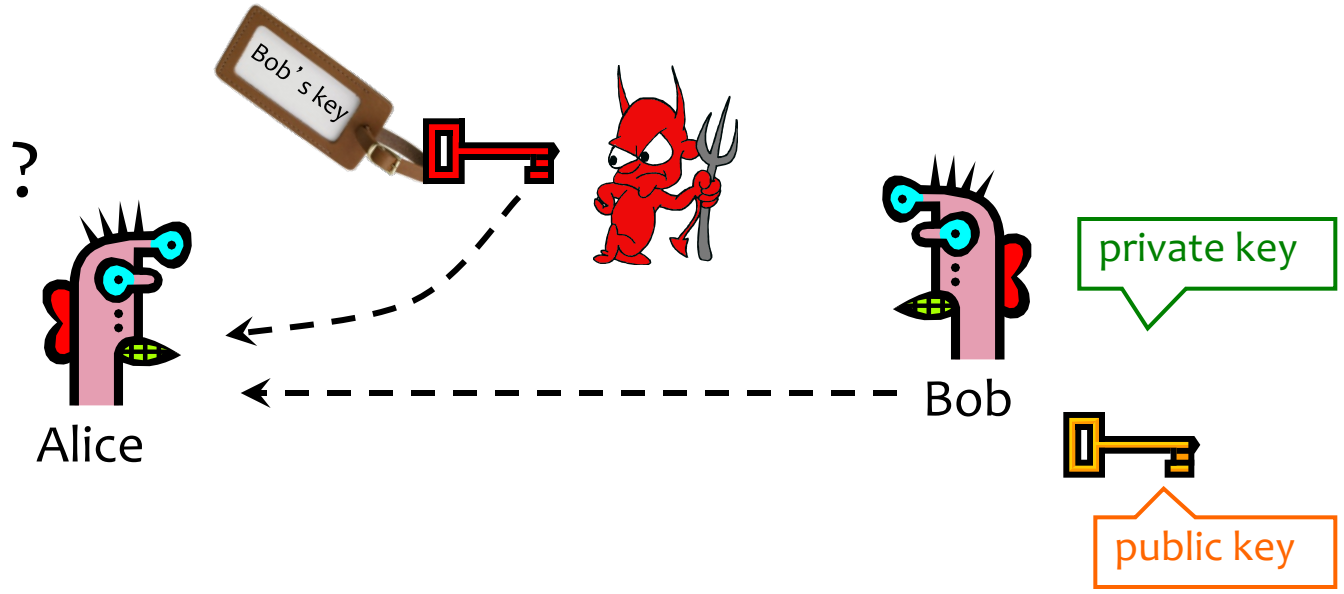
- Digital Signature Standard (DSS)
  - U.S. government standard (1991, most recent rev. 2013)
- Public key: $(p, q, g, y=g^x \bmod p)$, private key: x
- Security of DSS requires hardness of discrete log
  - If could solve discrete logarithm problem, would extract x (private key) from $g^x \bmod p$ (public key)

- Again: We've discussed discrete logs modulo integers; significant advantages to using elliptic curve groups instead.

Cryptography Summary

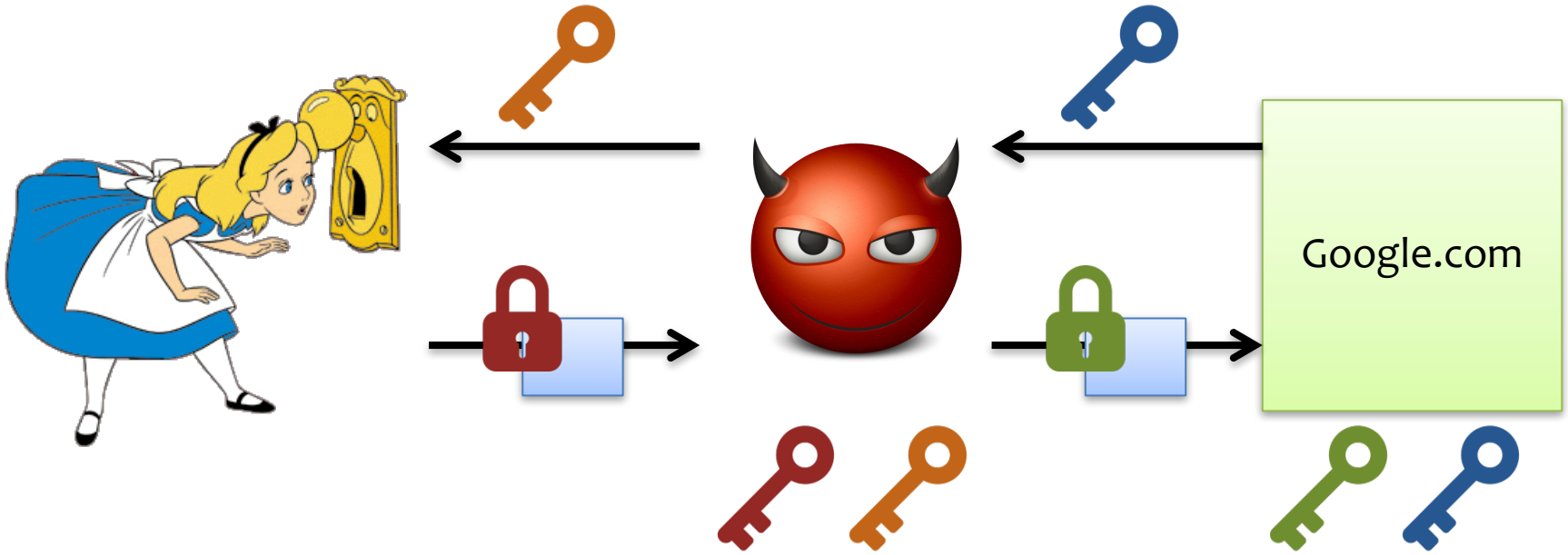
- Goal: Privacy
  - Symmetric keys:
    - One-time pad, Stream ciphers
    - Block ciphers (e.g., DES, AES) → modes: EBC, CBC, CTR
  - Public key crypto (e.g., Diffie-Hellman, RSA)
- Goal: Integrity
  - MACs, often using hash functions (e.g, SHA-256)
- Goal: Privacy and Integrity
  - Encrypt-then-MAC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

Authenticity of Public Keys



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

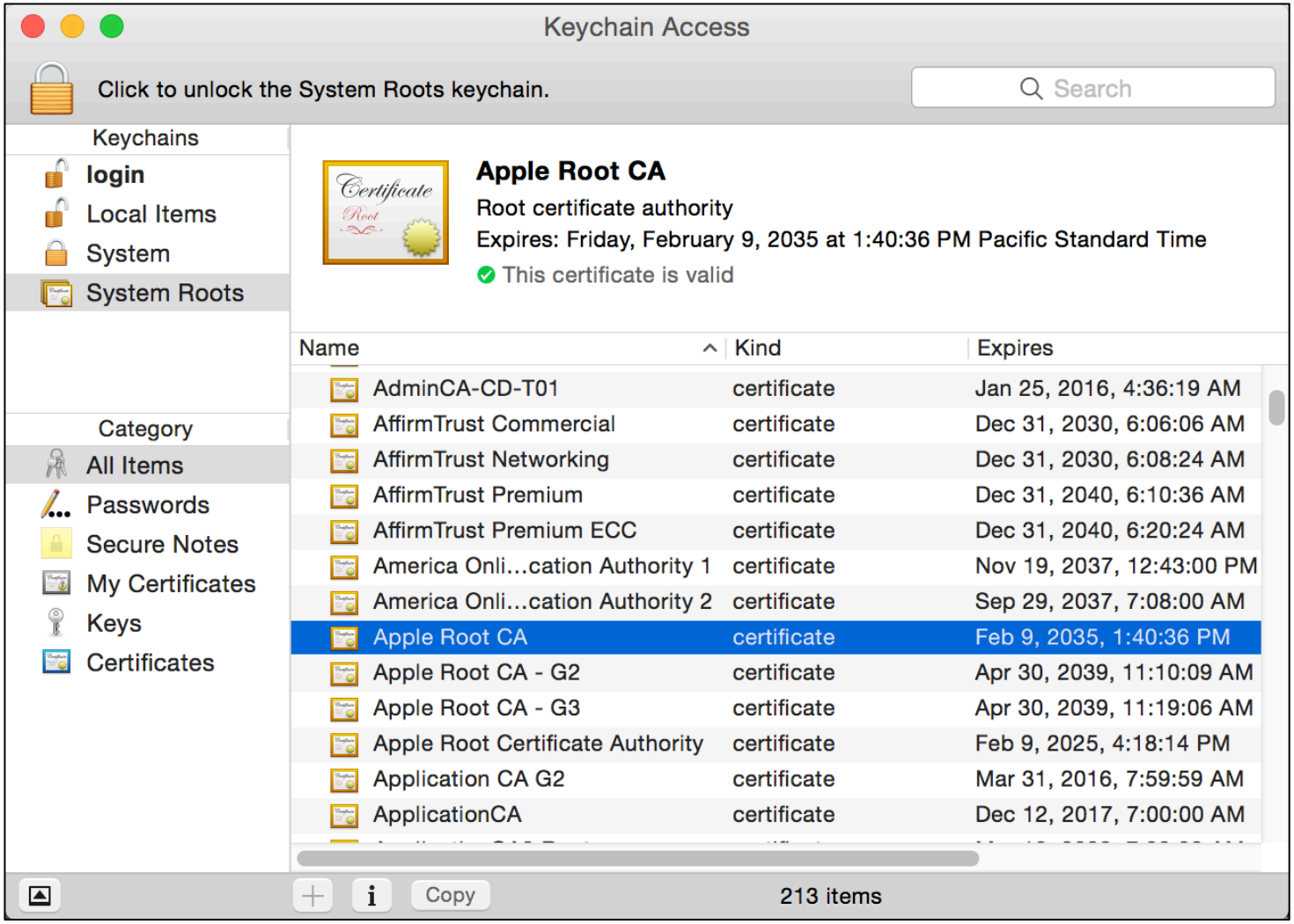
Threat: Man-In-The-Middle (MITM)



Distribution of Public Keys

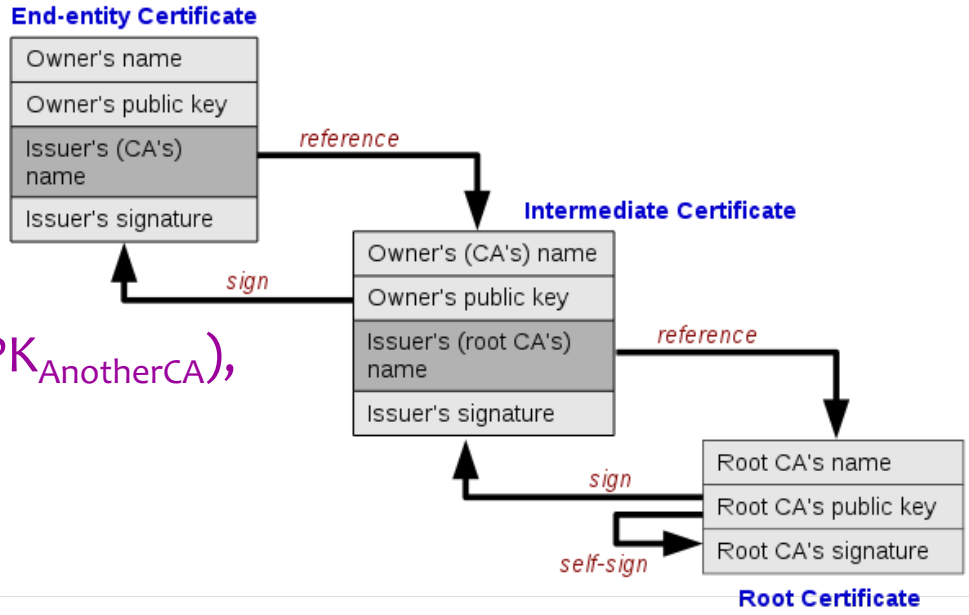
- Public announcement or public directory
  - Risks: forgery and tampering
- Public-key certificate
  - Signed statement specifying the key and identity
    - sig_{CA}("Bob", PK_B)
- Common approach: certificate authority (CA)
  - Single agency responsible for certifying public keys
  - After generating 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 pre-configured with CA's public key

Trusted(?) Certificate Authorities



Hierarchical Approach

- Single CA certifying every public key is impractical
- Instead, use a trusted root authority (e.g., Verisign)
  - Everybody must know the root's public key
  - Instead of single cert, use a certificate chain
    - $sig_{Verisign}$("AnotherCA", $PK_{AnotherCA}$), $sig_{AnotherCA}$("Alice", PK_A)



  - What happens if root authority is ever compromised?