Introduction to Information Security

0368-3065, Spring 2015

Lecture 7:

Applied cryptography: asymmetric

Eran Tromer

Slides credit:

John Mitchell, Stanford

Public-Key Encryption

Public-key encryption

- Alice generates a public encryption key *pk* and secret decryption key *sk:*

$$(sk, pk) \leftarrow G$$

- Anyone can send encrypted message

$$c \leftarrow E(pk, m)$$

- Only Alice can decrypt

$$m \leftarrow D(sk, c)$$

Example: RSA

A random s-bits integer is prime with probability $\sim 1/s$, and primality can be checked efficiently.

Large-integer modular arithmetic

- Key generation
  - Generate large primes p, q (e.g., 2048-bit each). Let $n = pq$.
  - Generate numbers e, d fulfilling $\forall x: x^{ed} \equiv x \bmod n$ (typically, $e = 65537$)
- Encryption:

  Public key: $pk = (n, e)$ Encrypt(pk, m) = $m^e \bmod n$
- Decryption:

  Secret key: $sk = (n, d)$ Decrypt(sk, c) = $c^d \bmod n$
- Main properties
  - This appears to be a “trapdoor permutation”
  - Infeasible to compute d from n, e (*Apparently*, need to factor $n = pq$.)

Why RSA works

- Let p, q be two distinct primes and let $n = pq$
  - Encryption, decryption based on group Z_n^*
  - For $n = pq$, group order is $\phi(n) = (p-1) \cdot (q-1)$
    Keys contain e, d with $ed \equiv 1 \bmod \phi(n)$
  - Encrypt(m) = $m^e \bmod n$
  - Decrypt(c) = $c^d \bmod n$
  - Since $ed \equiv 1 \bmod \phi(n)$,
    we have $(m^e)^d = m^{ed} \equiv m \bmod n$
    - By Eurler's theorem, generalizing Fermat's little theorem
    - (if gcd(m,n) != 1, then by "Chinese remainder theorem")

Textbook RSA is insecure

- What if message is from a small set (yes/no)?
  - Can build table

  (Deterministic)
- What if there's some protocol in which I can learn other message decryptions?
  (Chosen ciphertext attack)
- What if I want to outbid you in secret auction?
  - I take your encrypted bid c and submit

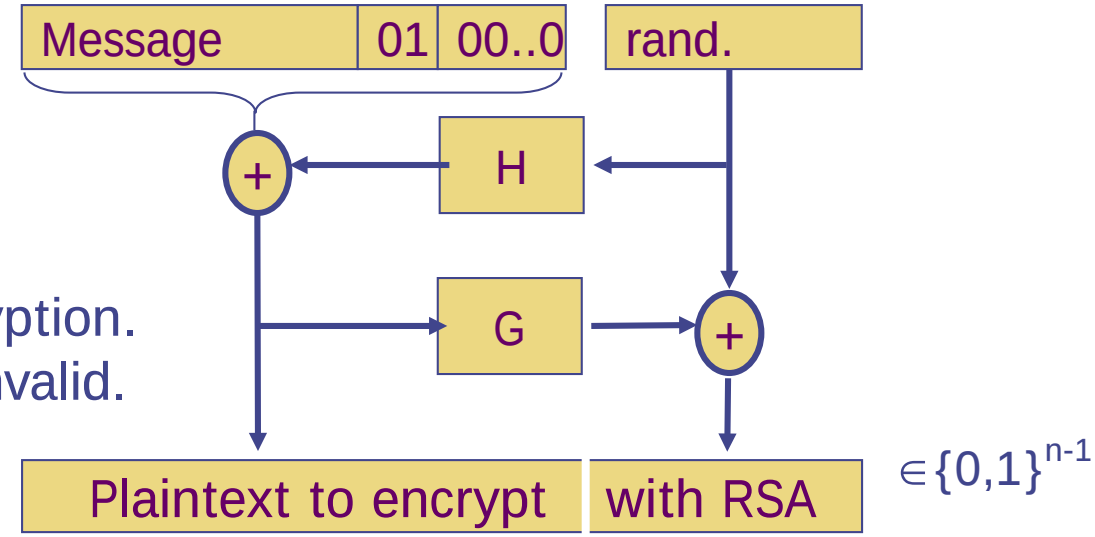
    $c\ (101/100)^e \bmod n$

  (Malleability)

RSA Padding: OAEP

[Bellare Rogaway '94] [Shoup '01]
[PKCS#1 v2] [RFC 2437]

Preprocess message for RSA



- Decryption:
  Apply plain RSA decryption.
- Check pad, reject if invalid.

- H and G are cryptographic hash functions (e.g., SHA-1)
  If RSA is *trapdoor permutation*, then this is
  chosen-ciphertext secure
  (if H,G "behave like random oracles")

Security of (properly-padded) RSA

- If factoring is easy, RSA is broken.
  - Converse conjectured but unproven.
- Best factoring algorithm: *Number Field Sieve* (subexponential complexity)
- Key size:
  - Record: 768 bits, in 2009, using ~ 2000 core-years.
  - Popular until recently: 1024-bit. Estimated to be breakable by a large botnet or special-purpose hardware (<1M$ marginal cost).
  - NIST recommendation:
    - 3072 bits (equivalent to 128 bit symmetric).
    - 2048 bits (equiv. to 112 bit symmetric) “acceptable until 2030”.
- *Quantum computers* can factor in polynomial time (Shor’s algorithm).
  - Appears possible in theory, but many believe it will take decades to solve the ngineering/technological challenges.
  - Record: factoring 15 and 21.

RSA discussion

- Encrypting long messages using *hybrid encryption*:
  - Alice generates key k for symmetric encryption
  - Alice sends k to Bob, encrypted under asymmetric encryption
  - Alice sends "payload" message m to Bob, symmetrically encrypted under k
- Faulty RSA key generation
  - Bad randomness source during key generation can cause common factor p or q.
  - Given two keys with common factor, can break both using GCD
  - Empirically: 0.2% of the RSA keys on the Internet can be broken this way (due to embedded devices with insufficient randomness, or VMs restored into the same state)

Other public-key encryption schemes

- Rabin (similar to RSA, with e=2)
  - Equivalent to factoring
  - But problem decrypting: $m^2 = (-m)^2$
- ElGamal
  - based on hardness of *discrete logarithm:* find x such that $g^x \equiv h \bmod p$
- Schemes based on *elliptic curves*
  - Popular in modern systems due to faster operations and smaller key size
- Lattice-based schemes (not yet popular, but advantages in efficiency and perhaps resists quantum computing)

Some schemes are *homomorphic* operations, allowing computation on ciphertexts.

- Example: RSA for multiplication: $(m_1)^e (m_2)^e = (m_1 m_2)^e$
- New: *Fully Homomorphic Encryption*, envisioned to be useful, e.g., for computing on the "cloud" while preserving confidentiality. Currently inefficient. [whiteboard discussion]

Digital Signatures

Digital Signatures

- Alice publishes key for verifying signatures
- Anyone can check a message signed by Alice
- Only Alice can send signed messages

Properties of signatures
(for case of deterministic signatures)

- Functions to sign and verify
  - Sign$(sk, m) \rightarrow \sigma$ (signature)
  - Verify(vk, m, σ) = $\begin{cases} true & \text{if } \sigma = \text{Sign}(sk, m) \\ false & \text{otherwise} \end{cases}$
- Resists forgery
  - Cannot compute Sign(sk, m) from m and vk
  - Resists existential forgery:

    given vk and prior signed messages, cannot produce valid signature for any new message

RSA Signature Scheme

- Publish decryption instead of encryption key
  - Alice publishes decryption key
  - Anyone can decrypt a message encrypted by Alice
  - Only Alice can send encrypt messages
- In more detail,
  - Alice generates primes p, q and key pair ⟨e, d⟩
  - Sign(sk, m): m^d
  - Verify(vk, m, σ) : $\sigma^e \equiv ? m$ mod n
  - Since ed ≡ 1 mod ϕ(n), have $x^{de} \equiv x$ mod n

Hybrid signature:
sign hash of message instead of full plaintext

Other digital signature schemes

- DSA (Digital Signature Algorithm)
  - Relies on hardness of discrete logarithms
- Schemes based on *elliptic curves*
  - Popular in modern systems due to faster operations and smaller key size
- Signatures based just on hash functions (Lamport), with stateful signing algorithm and limited #messages.
- Lattice-based schemes

Generalization:

succinct noninteractive proofs of knowledge (SNARK) allowing verifying the correctness not just of *data*, but also of *computation*. [whiteboard discussion]

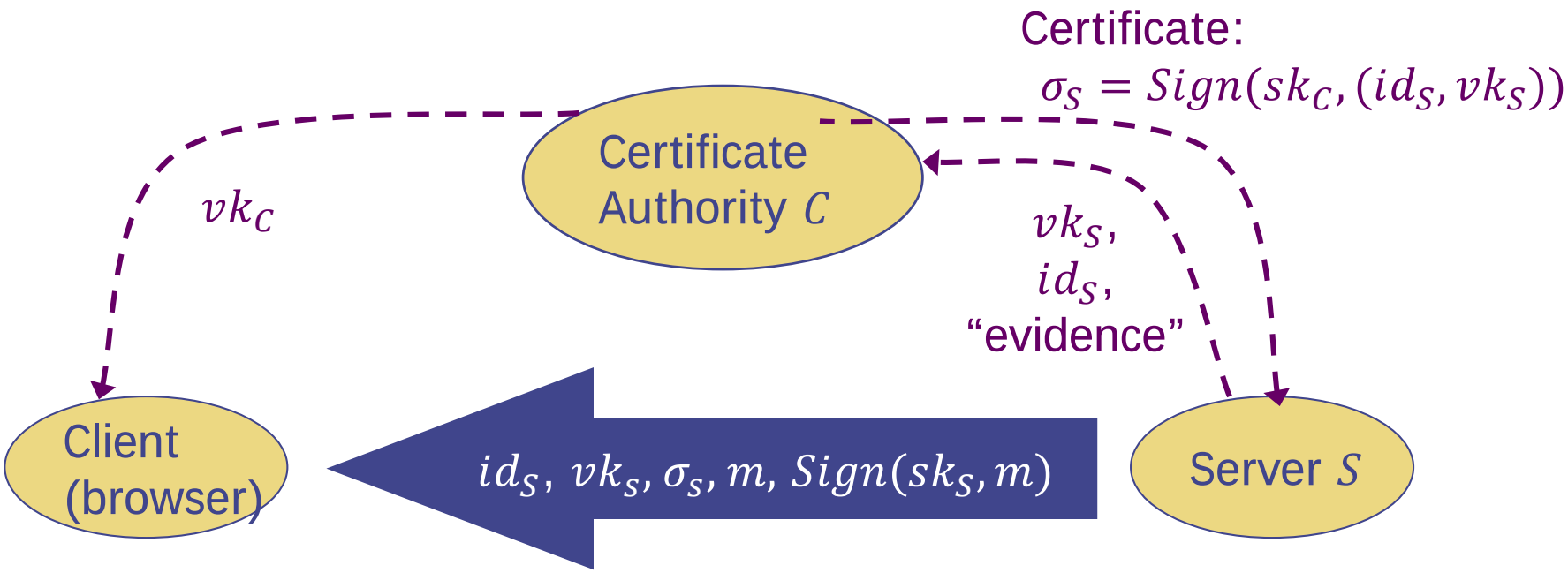
Public-key infrastructure

Public-Key Infrastructure (PKI)

- Anyone can send Bob a secret message
  - Provided they know Bob's public key
- How do we know a key belongs to Bob?
  - If imposter substitutes another key, can read Bob's mail
- One solution: PKI
  - Trusted root authority (VeriSign, IBM, United Nations)
    - Everyone must know the verification key of root authority
    - Check your browser; there are hundreds!
  - Root authority can sign certificates
  - Certificates identify others, by linking their ID (e.g., domain name or legal name) to a verification key they own
  - Certifiicates can also delegate trust to other certificate authorities
    - Leads to certificate chains
  - Most common standard "X.509"

Public-Key Infrastructure

Known public signature verification key vk_C of certificate authority



Server certificate can be verified by any client that has CA key vk_C.

Certificate authority is "off line".

CA


Certificate Manager
Your Certificates
Other People's
Web Sites
Authorities
You have certificates on file that identify these certificate authorities:
Certificate Name
Security Device
Comodo CA Limited
Digital Signature Trust Co.
Entrust.net
Equifax
Equifax Secure
Equifax Secure Inc.
GTE Corporation
GeoTrust Inc.
GlobalSign nv-sa
Government Root Certification A...
IPS Internet publishing Services s.l.
IPS Seguridad CA
NetLock Halozatbiztonsagi Kft.
QuoVadis Limited
RSA Data Security, Inc.
RSA Security Inc
SECOM Trust.net
Sonera
View
Edit
Import
Delete
OK

Certificate authorities – practical problems

- Certification policy – when to sign server's certificates?
- Inclusion in database of trusted Cas
  - Default database in browsers, OSs
  - Updates
- Transitive trusts, sub-CAs
- Practically:
  - Lax verification (attacks known)
  - Lax security (attacks known)
  - National/commercial bodies with diverse interests