RSA Encryption

CONSTRUCTION 11.25

Let GenRSA be as in the text. Define a public-key encryption scheme as follows:

- **Gen**: on input $1^n$ run GenRSA$(1^n)$ to obtain $N$, $e$, and $d$. The public key is $(N, e)$ and the private key is $(N, d)$.
- **Enc**: on input a public key $pk = (N, e)$ and a message $m \in \mathbb{Z}_N^*$, compute the ciphertext
  \[ c := [m^e \mod N]. \]
- **Dec**: on input a private key $sk = (N, d)$ and a ciphertext $c \in \mathbb{Z}_N^*$, compute the message
  \[ m := [c^d \mod N]. \]

The plain RSA encryption scheme.
Existential Unforgeability of Signatures under CMA

Attacker “wins” if:
1. $m^* \notin Q$
2. $Vrfy(pk, m^*, \sigma^*) = 1$

Security Requirement: Any efficient attacker wins with probability at most negligible
**CONSTRUCTION 12.5**

Let GenRSA be as in the text. Define a signature scheme as follows:

- **Gen**: on input $1^n$ run GenRSA$(1^n)$ to obtain $(N, e, d)$. The public key is $(N, e)$ and the private key is $(N, d)$.

- **Sign**: on input a private key $sk = (N, d)$ and a message $m \in \mathbb{Z}_N^*$, compute the signature
  \[
  \sigma := [m^d \mod N].
  \]

- **Vrfy**: on input a public key $pk = (N, e)$, a message $m \in \mathbb{Z}_N^*$, and a signature $\sigma \in \mathbb{Z}_N^*$, output 1 if and only if
  \[
  m = [\sigma^e \mod N].
  \]

The plain RSA signature scheme.
Attacks

No message attack:

Choose $s \in \mathbb{Z}_N^*$, compute $s^e$.
Output $(m = s^e, \sigma = s)$ as the forgery.
Attacks

Forging a signature on an arbitrary message:

To forge a signature on message $m$, choose arbitrary $m_1, m_2 \neq 1$ such that $m = m_1 \cdot m_2$. Query oracle for $(m_1, \sigma_1), (m_2, \sigma_2)$. Output $(m, \sigma)$, where $\sigma = \sigma_1 \cdot \sigma_2$. 
CONSTRUCTION 12.6

Let \text{GenRSA} be as in the previous sections, and construct a signature scheme as follows:

- **Gen**: on input $1^n$, run \text{GenRSA}(1^n) to compute $(N, e, d)$. The public key is $(N, e)$ and the private key is $(N, d)$.
  
  As part of key generation, a function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$ is specified, but we leave this implicit.

- **Sign**: on input a private key $(N, d)$ and a message $m \in \{0, 1\}^*$, compute
  
  $$
  \sigma := [H(m)^d \mod N].
  $$

- **Vrfy**: on input a public key $(N, e)$, a message $m$, and a signature $\sigma$, output 1 if and only if $\sigma^e \equiv H(m) \mod N$.

The RSA-FDH signature scheme.
Certificates and Public-Key Infrastructure
A single certificate authority

- $pk_{CA}$ must be distributed over an authenticated channel
  - Need only be carried out once
- Usually, $pk_{CA}$ included in browser, browser programmed to automatically verify certificates as they arrive.
- To obtain certificate, must prove that url is legitimate.
- All parties must completely trust CA.
Multiple certificate authorities

- Parties can choose which CA to use to obtain a certificate.
- Parties can choose which CA’s certificates to trust.
- Problem: some CA may become compromised.
- Each user must manually decide which CA to trust.
Delegation and certificate chains

• Example of certificate chain:
  \[ pk_A, cert_{B\rightarrow A}, pk_B, cert_{C\rightarrow B} \]
  Need only trust Charlie in the above example.

• Certificate asserts that legitimate party holds public key and \textit{that the party is trusted to issue other certificates}.
  – Delegation of CA’s ability to issue certificates
The “web of trust” model

• Model is used by PGP (“pretty good privacy”) email encryption software for distribution of public keys.
• Anyone can issue certificates to anyone else
• Each user must decide who to trust
• Example:
  – Alice holds $pk_1, pk_2, pk_3$ for users $C_1, C_2, C_3$
  – Bob has certificates $cert_{C_1\rightarrow B}, cert_{C_3\rightarrow B}, cert_{C_4\rightarrow B}$
• Public keys and certificates can be stored in a central database.
Invalidating Certificates

• Expiration: Include expiration date as part of the certificate.
  – Very coarse grained method. E.g. employee leaves company but certificate does not expire for a year.

• Revocation
  – CA includes a serial number in every certificate it issues.
  – At the end of each day, the CA will generate a certificate revocation list (CRL) with the serial numbers of all revoked certificates.
  – CA will sign the CRL and the current date.
  – Signed CRL is then widely distributed.
Putting it all together:
SSL/TLS

- **TLS**: Transport Layer Security Protocol
  - Protocol used by browser when connecting via https
- **Standardized protocol based on a precursor called SSL (Secure Socket Layer)**.
  - Latest SSL version: SSL 3.0
  - TLS version 1.0 released in 1999
  - TLS version 1.1 in 2006
  - TLS version 1.2 (current) in 2008
  - 50% of browsers still use TLS 1.0
- **Allows a client (web browser) and a server (website) to agree on a set of shared keys and then use those keys to encrypt and authenticate their subsequent communication.**
- **Two parts**:  
  - Handshake protocol performs authenticated key exchange to establish the shared keys  
  - Record-layer protocol uses shared keys to encrypt/authenticated the communication.
- **Typically used for authentication of servers to clients (usually only servers—websites—have certificates).**