<u>Cryptography</u>

12 Digital Signatures

- Integrity and Authenticity
- Dual to public-key encryption
- Applications:
  - signed email
  - certificates and public key infrastructures
  - software distribution
- universally verifiable (*neq* Message Auth.Codes - MACs)

12.1 Digital Signature Scheme

$\Gamma.\mathbb{M}$: message space
$\Gamma.\Sigma$: Signature space
$\Gamma.KeyGen() \rightarrow (pk, sk)$
$\Gamma.Sign(sk, m) \rightarrow \sigma$ (m message, σ signature)
$\Gamma.Ver(pk, m, \sigma) \rightarrow$ TRUE/FALSE

Completeness

$\forall m \in \mathbb{M}, (pk, sk) \leftarrow KeyGen():$
$Ver(pk, m, Sign(sk, m)) =$ TRUE

Security

Security means: $\mathbb{A}$ cannont <u>*forge*</u> a valid message/signature pair that was not produced by the legitimate signer.

Definition: A digital-sign.-schme Γ is *secure* (existential unforgability against **adaptive chosen-message attacks**) if

$\mathbb{L}^{\Gamma}_{sig-real}$
$(pk, sk) \leftarrow KeyGen()$
$\underline{\text{GetPK}():}$
return pk
$\underline{\text{GetSig}(m):}$
return $Sign(sk, m)$
$\underline{\text{CheckSig}(m, \sigma)}$
return $Ver(pk, m, \sigma)$

$\mathbb{L}^{\Gamma}_{sig-ideal}$
$(pk, sk) \leftarrow KeyGen()$
$\mathbb{S} := \emptyset$
$\underline{\text{GetPK}():}$
return pk
$\underline{\text{GetSig}(m):}$
$\sigma \leftarrow Sign(sk, m)$
$\mathbb{S} := \mathbb{S} \vee \{(m, \sigma)\}$
return σ
$\underline{\text{CheckSig}(m, \sigma)}$
return $(m, \sigma) \overset{?}{\in} \mathbb{S}$

Relaxations are possible by permitting multiple equivalent signatures for each message.

12.2 RSA signatures

Recall RSA function:

- $\underline{\text{KeyGen():}}$
  p, q ... large primes
  $N := p \cdot q$
  e ... (small) prime
  $d \ : \ d \cdot e \equiv_{\Phi(N)} 1 \qquad \Phi(N) := (p-1) \cdot (q-1)$
  $(pk, sk) := ((N, e), d)$

- $\underline{\text{Eval(pk,x):}}$
  return $x^e \ mod \ N$

- $\underline{\text{Invert(sk,c):}}$
  return $c^d \ mod \ N$

Textbook RSA signatures - INSECURE

Direct application of RSA function for signatures:
$\Gamma.KeyGen()$:
return RSA.KeyGen()

$\Gamma.Sign(sk, m)$
return RSA.Invert(sk,m)

$\Gamma.Ver(pk, m, \sigma)$
return RSA.Eval(pk,σ) $\overset{?}{=} m$

$\underline{m_1, m_2}$: (To forge sign on $\overline{m}$ pick m_1, m_2 s.t. $\overline{m} = m_1 \cdot m_2$)
$\sigma_1 \ := \ Sign(sk, m_1)$

$\sigma_2 \ := \ Sign(sk, m_2)$
It holds that:
$\sigma_1^e \equiv_N m_1$
$\sigma_2^e \equiv_N m_2$
$\}(\sigma_1 \cdot \sigma_2)^e \equiv_N m_1 \cdot m_2$
To forge signature of $\overline{m} = m_1 \cdot m_2$ compute $\overline{\sigma} = (\sigma_1 \cdot \sigma_2)^e \ mod \ N$

Textbook RSA
given m, difficult to produce σ
Attack to produce a valid message/signature pair:
pick random $\sigma^\star$
$m^\star := (\sigma^\star)^e \ mod \ N$
This satisfies $(\sigma^\star)^e \equiv_N m^\star$, $Ver(pk, m^\star, \sigma^\star) \ = \text{TRUE}$

RSA Full-domain-hash signatures
How to properly sign with RSA in the random oracle model (R.O.M.)
Idea: Break the connection between message m and a signature σ by using a hash-function H and signing its output.

Let $H : \{0,1\}^\star \rightarrow \mathbb{Z}_N^\star$
$J : \{0,1\}^\star \rightarrow \{0,1\}^k$
but $k \approx 256$
Define $H : \{0,1\}^\star \rightarrow \{0,1\}^{|N|}$
$H(x) := J(0\|x)\|J(1\|x)\|...\|J(l-1\|x)$ for $l = \lceil \frac{|N|}{k} \rceil$

FDH-RSA
KeyGen() - same as before

Sign(sk,m)
return RSA.Inverse(sk, H(m))

Ver(pk,mσ)
return RSA.Eval(pk, σ) $\stackrel{?}{=}$ H(m)

Hash and sign: Hash message first and then apply pk-transformation
Hybrid-encryption: pick a random symmetric key and then encrypt (long) plaintext with k and use the public-key-transformation to encrypt k itself

12.3 Schnorr Signatures

PK signatures based on the DLP
Group $G = \langle g \rangle$, generator g
$\mid G \mid = q$, for example $G \subset \mathbb{Z}_p$, p prime, s.t. $q \mid (p-1)$
$\mid p \mid \approx 2048$
$\mid q \mid \approx 256$

$H : \{0,1\}^\star \to \mathbb{Z}_q$

$\underline{\text{KeyGen():}}$
$x \leftarrow \mathbb{Z}_q$
return (g^x, x)

$\underline{\text{Sign(x,m):}}$
$r \leftarrow \mathbb{Z}_q$
$t := g^r \ mod\ p$
$c := H(m \| t)$
$s := (r - c \cdot x)\ mod\ q$
return (c, s)

$\underline{\text{Ver(y (= } g^x\text{), m, (c,s)):}}$
$\hat{t} := g^s \cdot y^c \ mod\ p\ (= g^s \cdot y^c \ = \ g^{r-c\cdot x} \cdot g^{x^c} \ = \ g^{r-c\cdot x+c\cdot x} \ = \ g^r$
return $c \stackrel{?}{=} H(m \| \hat{t})$

$\underline{\textbf{Completeness:}}$
$\hat{t} = t$ because $g^s \cdot y^c \equiv_p t \equiv_p g^r$
$c = H(m \| t) = H(m \| \hat{t})$ because H deterministic.