c) Verification of step (i) of El Gamal signatures requires checking of $1 \leq r \leq p-1$

If this check is omitted, than Oscar can sign messages of his choice provided he has one valid signature and $h(m)^{-1} \bmod p-1$ should exist.

Suppose (r, s) is a signature for message m.

O selects a message m' of his choice and computes

$h(m')$ and $u = h(m')\left(h(m)\right)^{-1} \bmod p-1$

$s' = su \bmod p-1$

r' such that $r' \equiv ru \pmod{p-1}$ and $r' \equiv r \pmod{p}$

Solve this by CRT, because $p, p-1$ are relatively prime

The pair (r', s') is a signature for m' which would be accepted if $1 \leq r' \leq p-1$ is ignored.

11.2 The Digital Signature Algorithm (DSA)

- Proposal by the NIST in Aug '91
- Standardized as FIPS 186, named DSS (Digital Signature Standard)
- Developed by the NSA (not publicly)
- DSA is a variant of the ElGamal signature scheme
- Needs a hash function $h: \{0,1\}^* \to \mathbb{Z}_q$ as a building block
  The standard prescribes SHA-1

System parameters

Each user generates a public and private key as follows:

1. Choose a prime q with $2^{159} < q < 2^{160}$ (160 bits)
2. Choose t, $0 \leq t \leq 8$, further a prime p such that

   $2^{511+64t} < p < 2^{512+64t}$ and $q \mid p-1$ (512 ... 1024 bits)

   (Recommendation by NIST from Oct 2001: $t=8$, 1024 bits)
3. (i) Select $g \in \mathbb{Z}_p^*$, compute $\alpha = g^{(p-1)/q} \bmod p$

   (ii) If $\alpha = 1$, repeat step (i)

   (α is a generator of a cyclic subgroup of order q in $\mathbb{Z}_p^*$)
4. Choose some random $x \in \{1, \dots, p-1\}$ // Could be $q-1$ as well
5. Compute $y = \alpha^x \bmod p$
6. Public key: (p, q, α, y), private key x

Signing a message $m \in \{0,1\}^*$

1. Choose a random $k \in \{1, \dots, q-1\}$
2. $r = (\alpha^k \bmod p) \bmod q$
3. Compute $k^{-1} \bmod q$
4. $s = k^{-1}(h(m) + x \cdot r) \bmod q$
5. Signature (r, s) (320 bits in total)

Verification of signature (r,s) on message m:

1. Check if $0 < r < q$ and $0 < s < q$, otherwise decline
2. $w = s^{-1} \mod q$
3. $u_1 = (w\, h(m)) \mod q$, $u_2 = (r \cdot w) \mod q$
4. $v = (a^{u_1} \cdot y^{u_2} \mod p) \mod q$
5. Accept the signature if $v = r$

Proof that the verification is correct:

For a valid signature (r,s) it holds that

$$h(m) \equiv k \cdot s - x \cdot r \pmod q$$

Hence, $\quad a^{u_1} \cdot y^{u_2} \equiv a^{u_1 + x u_2} \pmod p$

$$u_1 + x u_2 \equiv w\, h(m) + x r w \equiv w k \cdot s - w x r + x r w \equiv k \pmod q$$

$$v \equiv \big((\underbrace{a^{lq+k}}_{}) \mod p\big) \mod q = (a^{k} \mod p) \mod q = r \quad \checkmark$$

a has order q, i.e. $a^{q} \equiv 1 \pmod p$

Security

- Security relies on ~~the~~ two DL problems
  a) in $\mathbb{Z}_p^*$ b) in $\langle a \rangle \leq \mathbb{Z}_p^*$ ($\langle a \rangle$ denotes the subgroups gen. by a)
- Security principles of the ElGamal scheme carry over:
  - always choose ~~...~~ a new k
  - use of hash functions is mandatory
  - always verify 1. in the verification procedure. Otherwise signatures for arbitrary messages can be generated provided one valid signature is known.

Remarks!

a) Modular exponentiation is in the range of q (160 bits) (rather than 1024 El Gamal)

b) k, k^{-1}, r, xr may be generated, computed and stored in advance

c) Verification needs 2 instead of 3 modular exponentiations

d) Signature by DSA is short, 320 bits, instead of 2048 bits for El Gamal.

e) In the verification step, also check, if $r \neq 0, s \neq 0$. otherwise the signature is rejected. But this happens with a very small probability.

12. Identification and Entity Authentication

This chapter considers techniques to allow the "verifier" to establish the identity of the "claimant", thereby preventing impersonation.

Requirements on authentication protocols:

1. A is able to uniquely identify herself to B
2. B cannot reuse an identification exchange with A so as to impersonate A to a third party C. (transferability)
3. It is practically infeasible that a third party C can cause B to wrongly accept the identity of A. (impersonation)
4. Even if C observes the identification process between A and B very often he cannot impersonate A.

Three main categories of identification:

1. Something is known: password, PIN, private key
2. Something possessed: key, magnetic-striped cards, chipcards, PIN or password generators, ...
3. Something inherent: human physical characteristics, face recognition, fingerprint, retinal pattern, hand written signatures

12.1 Passwords

Fixed password schemes

Rather than storing a cleartext user password pwd in a file, a hash value h(pwd) of each user password is stored. Verification is done by comparing the hash value of the entered password with the stored one for a given user.

Main attacks are :

- replay of fixed passwords
- exhaustive password search
- password-guessing and dictionary attacks

Defense strategies are

- Choose a random password, or nearly random, use of special characters (increasing entropy)
- Slowing down the password mapping
- Salting passwords
  Extend the password by some random string, the salt, before hashing. Both the hashed password and the salt are stored
  h(password, salt), salt
  This does not complicate exhaustive search itself, simultaneous dictionary attacks against a large set of passwords

One-time passwords

Protects against eavesdropping and replay of passwords or "phishing".

Lamport's protocol

Objective: A identifies herself B
Use a ~~secret~~ one way function H
Notation $H^k(\omega) = \underbrace{H(H(\dots H(\omega)))}_{k\text{-times}}$

Initial parameters: t: max number of identif. ($t = 100, 1000$)
A chooses an initial password ω
A transfers $\omega_0 = H^t(\omega)$ to B
B initializes his counter for A to $i_A = 1$

Protocol actions for session i:

A computes $\omega_i = H^{t-i}(\omega)$, transfers to B: (A, i, ω_i)
B checks that $i = i_A$ and $\omega_{i-1} = H(\omega_i)$. If both checks succeed B accepts and sets $i_A \leftarrow i_A + 1$ and stores ω_i