11.1. El Gamal Signature Scheme

Parameters:

p: prime, a: PE mod p, h: Hash functions

Select random x, $y = a^x \bmod p$

Public key: (p, a, y), Private key: x

Signature generation on a message m:

Compute $h(m)$

Select random k s.t. $k^{-1} \pmod{p-1}$ ∃

$r = a^k \bmod p$

$s = k^{-1}(h(m) - xr) \bmod (p-1)$ (*)

Signature on m: (r, s)

Signature verification

Verify: $1 \le r \le p-1$

$v_1 = y^r r^s \bmod p$

$v_2 = a^{h(m)} \bmod p$

$v_1 = v_2 \longrightarrow$ accept signature. ∎

Verification works:

$ks \equiv h(m) - xr \pmod{p-1}$

$\Leftrightarrow h(m) \equiv xr + ks \pmod{p-1}$

$\Leftrightarrow xr + ks = \ell(p-1) + h(m)$ for some $\ell \in \mathbb{Z}$

Hence

$$y^r r^s \equiv a^{xr} a^{ks}$$
$$\equiv a^{xr+ks} \equiv \underbrace{(a^{p-1})^{\ell}}_{\equiv 1 \pmod p \text{ 'Fermat'}} a^{h(m)}$$
$$\equiv a^{h(m)} \pmod p.$$

∎

Security

a) Don't use the same key twice! Otherwise

$$s_1 = k^{-1}(h(m_1) - xr) \bmod (p-1)$$
$$s_2 = k^{-1}(h(m_2) - xr) \bmod (p-1)$$

$$\Rightarrow (s_1 - s_2)\, k \equiv (h(m_1) - h(m_2)) \bmod (p-1)$$

$$\Rightarrow k \equiv (s_1 - s_2)^{-1}(h(m_1) - h(m_2)) \bmod (p-1)$$

provided $(s_1 - s_2)^{-1} \bmod (p-1)$ exists.

Once k is known, x can be obtained from (*)

b) OE can forge a signature on a message m as follows:

Select any pair (u, v) such that $\gcd(v, p-1) = 1$

Compute $r = a^u y^v = a^u a^{xv} \bmod p$

$$s = -rv^{-1} \bmod (p-1)$$

Then (r, s) is a valid signature on

$m = su \bmod (p-1)$. (Ex)

Avoid this by using hash functions.

$m \leftarrow h(m)$.

c) Verification requires checking of $1 \leq r \leq p-1$.
If this omitted then O can sign messages of his choice provided he has one valid signature.

8.4. Public Key Infrastructure (PKI)

Biggest challenge for PKI:
authenticity of public keys.

Example (TLS/SSL)

Alice wants to buy something from Bob.

Client A		Server B
	"I'm A" hello →	
	← "I'm B" hello	
	← PK, $sign_{CA}(PK)$	
verify PK		
generates MS	$y = e_{PK}(MS)$ →	
$(K_1, K_2) = h(MS)$		$MS = d_{PrK}(y)$
		$(K_1, K_2) = h(MS)$

K_1 is used to authenticate data by a $MAC(K_1)$.

K_2 is used for en-/decryption (e.g. AES)

Note: A may not have a public key or certificate. B relies on the validity of the credit card number.