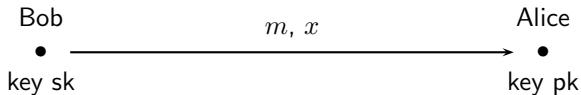


The following are important requirements for digital signatures.

- ▶ The signature must be tightly attached to the signed document.
- ▶ It should be easy to *sign* for the legitimate signer, easy to *verify* the signature for the recipient, and hard to sign for a forger.
- ▶ The signer should not be able to deny that he signed the document.
- ▶ Sometimes it is important that a signed document can only be used once for its legitimate purpose, not several times (say, a cheque).



$$x = \text{dec}_{\text{sk}}(m)$$

$$y = \text{enc}_{\text{pk}}(x); \text{ verify: } m \stackrel{?}{=} y$$

$$\text{ver}_{\text{pk}}(m, z) = \text{true} \iff m = \text{enc}_{\text{pk}}(z).$$

PROTOCOL 1. ElGamal signature scheme.

Set-up.

Input: a security parameter n given in unary.

Output: as below.

1. A cyclic group $G = \langle g \rangle$ with $d = \#G$ elements, where d is an n -bit number. We also have an injective encoding function $G \rightarrow \mathbb{Z}_d$, denoted as $x \mapsto x^*$, which is easy to compute but otherwise has no particular properties. All these are published.

Key generation.

Secret key $sk = a \xleftarrow{\$} \mathbb{Z}_d$ and public key $pk = A = g^a \in G$.

PROTOCOL 2. ElGamal signature scheme.

Signing.

Input: a message $m \in \mathbb{Z}_d$.

Output: the signature $\text{sig}_{\text{sk}}(m) \in G \times \mathbb{Z}_d$ of m .

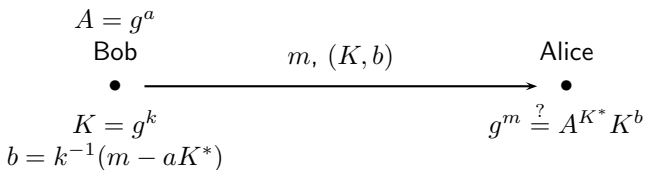
1. Choose a secret session key $k \xleftarrow{\text{R}} \mathbb{Z}_d^\times$.
2. $x \leftarrow g^k \in G$.
3. Calculate $b \leftarrow k^{-1} \cdot (m - ax^*)$ in \mathbb{Z}_d , where k^{-1} is the inverse of k in \mathbb{Z}_d .
4. Transmit m and its signature $\text{sig}_{\text{sk}}(m) = (K, b)$.

Verifying.

Input: message m and a pair $(z, c) \in G \times \mathbb{Z}_d$.

Output: $\text{ver}_{\text{pk}}(m, z, c)$, which is either “true” or “false”.

5. Compute $u \leftarrow g^m$ and $v \leftarrow Az^*z^c$ in G .
6. If $z \neq 1$ and $u = v$ then return “true” else return “false”.



Bob has set up the publicly known group $G = \mathbb{Z}_{17}^\times$, with order $d = 16$, and $g = 3$. Now he chooses his secret key, say $a = 9$, and publishes $A = 3^9 = 14$ in \mathbb{Z}_{17}^\times . Suppose that he wants to sign $m = 11$ and chooses $k = 5$ as secret session key. Then indeed $\gcd(5, 17 - 1) = 1$, and $k^{-1} = -3 = 13$ in \mathbb{Z}_{16} . He calculates

$$K = 3^5 = 5 \text{ in } \mathbb{Z}_{17},$$

$$b = 13 \cdot (11 - 9 \cdot 5) = 6 \text{ in } \mathbb{Z}_{16},$$

since $K^* = K = 5$. His secret key is $\text{sk} = (9, 5)$, and his public key is $\text{pk} = (14, 5)$. Bob sends the message 11 together with its signature $\text{sig}_{\text{sk}}(11) = (5, 6)$.

Alice computes

$$A^{K^*} K^b = 14^5 \cdot 5^6 = 7 \text{ in } \mathbb{Z}_{17},$$

$$g^m = 3^{11} = 7 \text{ in } \mathbb{Z}_{17},$$

and accepts the message as properly signed.

PROTOCOL 3. Schnorr's signature scheme (DSA).

Set-up.

Input: two security parameters $\ell < n$ in unary.

1. Choose an ℓ -bit prime d and an n -bit prime p , as above, with d dividing $p - 1$, a generator g of a group $G = \langle g \rangle \subseteq \mathbb{Z}_p^\times$ of order d , and a map $*$ from \mathbb{Z}_p^\times to \mathbb{Z}_d .

Key Generation.

Output: secret and public keys.

2. Secret key $sk = a \xleftarrow{\$} \mathbb{Z}_d$ and public key $pk = A \leftarrow g^a \in G$.

PROTOCOL 4. Schnorr's signature scheme (DSA).

Signing.

Input: a message $m \in \mathbb{Z}_d$.

Output: the signature $\text{sig}_{\text{sk}}(m) \in G \times \mathbb{Z}_d$ of m .

1. Choose a secret session key $k \xleftarrow{\$} \mathbb{Z}_d^\times$.
2. $K \leftarrow g^k \in G$.
3. Calculate $b \leftarrow k^{-1} \cdot (m - aK^*)$ in \mathbb{Z}_d , where k^{-1} is the inverse of k in \mathbb{Z}_d .
4. Transmit m and its signature $\text{sig}_{\text{sk}}(m) = (K, b)$.

Verifying.

Input: message m and a pair $(z, c) \in G \times \mathbb{Z}_d$.

Output: $\text{ver}_{\text{pk}}(m, z, c)$, which is either “true” or “false”.

5. Compute $u \leftarrow g^m$ and $v \leftarrow Az^* z^c$ in G .
6. If $z \neq 1$ and $u = v$ then return “true” else return “false”.

We compare Schnorr's signature scheme and ElGamal's signature scheme, using a 2048-bit prime.

| | ElGamal | DSS |
|------------------------|-----------------------------------|-----------------------------------------|
| cost A of arithmetic | $(2048)^2$ | $(2048)^2$ |
| cost of g^k | $A \log_2 p \approx A \cdot 2048$ | $A \log_2 d \approx A \cdot 200$ |
| message length | 2048 bits | 200 bits |
| signature length | 4096 bits | 400 bits (!) |
| attack | DL in \mathbb{Z}_p^\times | DL in \mathbb{Z}_p^\times or in G |

OAEP Padding

We have m -bit input messages x and produce n -bit inputs to the encryption scheme, with m much smaller than n , as follows. We write $n = m + \ell + k$ with positive integers k and ℓ , roughly equal, and $\mathbb{B} = \{0, 1\}$. Furthermore, $g: \mathbb{B}^\ell \mapsto \mathbb{B}^{m+k}$ is an expansion function. All these are publicly known. The padding goes as follows.

1. $x_1 \longleftarrow (x, 0, \dots, 0) \in \mathbb{B}^{m+k}$.
2. $r \xleftarrow{\$} \mathbb{B}^\ell$.
3. $y_1 \longleftarrow x_1 \oplus g(r) \in \mathbb{B}^{m+k}$.
4. $y_2 \longleftarrow h(y_1) \oplus r \in \mathbb{B}^\ell$.
5. Return $y = (y_1, y_2) \in \mathbb{B}^n$.

SIGNATURE SCHEME 5. FDH-RSA.

Key Generation.

Input: a security parameter n .

Output: an n -bit RSA modulus $N = pq$ with public key $\text{pk} = (N, e)$ and secret key $\text{sk} = (N, d)$ as in the RSA notation, and a surjective hash function $h: \mathbb{B}^* \rightarrow \mathbb{Z}_N^\times$.

Signing.

Input: message $m \in \mathbb{B}^*$.

Output: $\text{sig}_{\text{sk}}(m) \in \mathbb{Z}_N$.

1. Return $s = h(m)^d$ in \mathbb{Z}_N^\times .

SIGNATURE SCHEME 6. FDH-RSA.

Verifying.

Input: $m \in \mathbb{B}^*$ and $u \in \mathbb{Z}_N^\times$.

Output: “true” or “false”.

1. $\text{ver}_{\text{pk}}(m, u) = “u^e = h(m) \text{ in } \mathbb{Z}_N^\times”$.

ALGORITHM 7. Random oracle reduction \mathcal{A} from breaking RSA to existential forgery \mathcal{F} with chosen messages of FDH-RSA.

Input: N, e, y as in the RSA notation.

Output: Either x with $x^e = y$ in \mathbb{Z}_N^\times or “failure”.

1. \mathcal{A} sends (N, e) to \mathcal{F} .
2. \mathcal{A} receives hash queries from \mathcal{F} and delivers hash values to \mathcal{F} .
3. \mathcal{A} receives signature queries from \mathcal{F} and either delivers a valid signature to \mathcal{F} or returns “failure”.
4. \mathcal{F} sends a “forged” signature (m^*, s^*) of FDH-RSA to \mathcal{A} . Its hash value $h(m^*)$ is among those requested in step (2).
5. \mathcal{A} verifies (m^*, s^*) and returns “failure” if the verification answer is “false”.
6. \mathcal{A} returns either x with $x^e = y$ in \mathbb{Z}_N^\times or “failure”.

Theorem

Suppose we have an existential forger \mathcal{F} with chosen messages for FDH-RSA, using time $\tau_{\mathcal{F}}$, q_{sig} signature queries and q_{hash} hash queries and having success probability $\sigma_{\mathcal{F}}$. Then the above reduction \mathcal{A} can be implemented for n -bit RSA moduli in the random oracle model so that

$$\sigma_{\mathcal{A}} \geq \frac{\sigma_{\mathcal{F}}}{4q_{\text{sig}}},$$

$$\tau_{\mathcal{A}} \leq \tau_{\mathcal{F}} + (q_{\text{sig}} + q_{\text{hash}} + 1) \cdot \text{poly}(n).$$