Criptografía post-cuántica basada en retículos

Lección 2: Minicurso Mar del Plata, Noviembre 2025

Syllabus

- **Protocolo de intercambio de claves de Diffie-Hellman**
- **Protocolo de identificacion de Schnorr**
- **Firma digital**
- **Transformacion de Fiat-Shamir**

Índice

2.4 Diffie-Hellman: New Directions in Cryptography

> ...una de las aventuras intelectuales mas importante del siglo 20 que fue: como se puede lograr que dos partes lleguen a una clave en comun aunque esten espiadas todo el tiempo sin que el espia pueda conocer esa clave...
>
> **Hugo Scolnik, minuto 41:45**

2.4.1 Dos escenarios, una ecuacion y varios problemas asociados

Escenario multiplicativo (el original en el paper [DH76]):

$$g^a = b \tag{1}$$

donde g^a indica la multiplicacion $\underbrace{g \cdot g \cdot \cdots \cdot g}_{a \text{ veces}}$.

El escenario additivo (el de las curvas elipticas [Mi86], [Ko87]):

$$a \cdot G = P \tag{2}$$

donde $a \cdot G$ indica la multiplicacion $\underbrace{G{+}G{+}\cdots{+}G}_{a \text{ veces}}$.

En cadas escenario tenemos tres letras g, a, b y a, G, P. Dando valores concretos a dos de ellas se pone el problema de resolver la ecuacion para la tercera letra.

dando valores a g, a se pone el problema de calcular b, (**exponenciacion**)

dando valores a g, b se pone el problema de calcular a, (**logaritmo**)

dando valores a a, b se pone el problema de calcular g (**raiz**).

2.4.2 Diffie-Hellman's key exchange (DH)

En esta seccion el famoso intercambio de claves publicas publicado en [DH76, 649].

NOTE 2.4.2.1

Tener presente que **Alice** y **Bob** son computadoras.

Alice e **Bob** se ponen de acuerdo en utilizar un numero primo p y un elemento g de $\mathbb{F}_p^*$ de orden q.

2.4.2.2 Diffie-Hellman key-exchange (DH)

Alice		**Bob**
	$a \leftarrow_{\$} \mathbb{Z}_q$	
Compute g^a	mód $[p]$	
	$\xrightarrow{g^a}$	
		$b \leftarrow_{\$} \mathbb{Z}_q$
		Compute g^b mód $[p]$
	$\xleftarrow{g^b}$	
$\mathsf{k}_A = (g^b)^a$	mód $[p]$	$\mathsf{k}_B = (g^a)^b$ mód $[p]$

Como

$$\mathsf{k} = \mathsf{k}_A = \mathsf{k}_B \quad \text{mód } [p]$$

este numero k es un secreto compartido solo entre **Alice** y **Bob** que van a utilizar para construir otras claves secretas.

Los numeros a e b son las claves privadas, notar que sono exponentes y pertenecen a $\mathbb{Z}_q$.

En cambio g^a, g^b son las claves publicas, notar que pertenecen a $\mathbb{F}_p^*$

NOTE 2.4.2.3

El numero primo p y el generador g son parte de los domain parameters NIST.SP.800-56Ar3:

5.5.1 Domain-Parameter Selection/Generation

5.5.1.1 FFC Domain Parameter Selection/Generation

If p is a prime number, then $GF(p)$ denotes the finite field with p elements, which can be represented by the set of integers $\{0, 1, ..., p-1\}$. The addition and multiplication operations for $GF(p)$ can be realized by performing the corresponding integer operations and reducing the results modulo p. The multiplicative group of non-zero field elements is denoted by $GF(p)^*$. In this Recommendation, an FFC key-establishment scheme requires the use of public keys that are restricted to a (unique) cyclic subgroup of $GF(p)^*$ with prime order q (where q divides $p-1$). If g is a generator of this cyclic subgroup, then its elements can be represented as $\{1, g \bmod p, g^2 \bmod p, ..., g^{q-1} \bmod p\}$, and $1 = g^q \bmod p$.

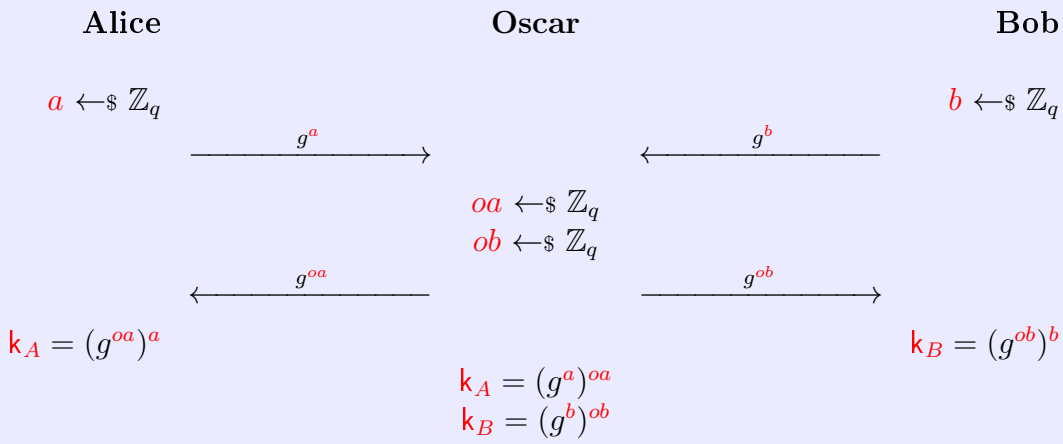
Domain parameters for an FFC scheme are of the form $(p, q, g\{, SEED, counter\})$, where p is the (odd) prime field size, q is an (odd) prime divisor of $p-1$, and g is a generator of the cyclic subgroup of $GF(p)^*$ of order q. The optional parameters, *SEED* and *counter*, are described below.

En las aplicaciones p tiene come minimo 2048 bits y q 256 bits e.g. para evitar ataques index calculus. Para mas informacion: `https://www.keylength.com/en/compare/`.

NOTE 2.4.2.4

Es importante subrayar que el canal debe evitar el ataque man-in-the-middle. Para evitarlo se utilizan los certificados digitales.

2.4.2.5 Man in the Middle



Notar que **Oscar** no solo lee los mensajes entre **Alice** y **Bob** sino que podria tambien modificarlos.

Ejercicio 2.4.2.6

Sea $(d_t d_{t-1} d_{t-2} \cdots d_0)_2$, $d_t = 1$, el numero n escrito en binario.
Que numero devuelve el `return(T)` del seguiente algoritmo ?

```
T=1
For i=t-1 downto i=0
    T = T + T
    if d_i = 1
       T = T + 1
return(T)
```

Ejercicio 2.4.2.7

Calcular a mano $2^{17} \pmod{19}$.

2.5 Firmas digitales

La firma digital proporciona:

- **autenticación del mensaje** (el receptor puede verificar el origen del mensaje),
- **integridad** (el receptor puede verificar que el mensaje no ha sido modificado desde que fue firmado),
- **no repudio** (el emisor no puede negar falsamente que ha firmado el mensaje).

2.5.0.1 Esquema de Firma Digital

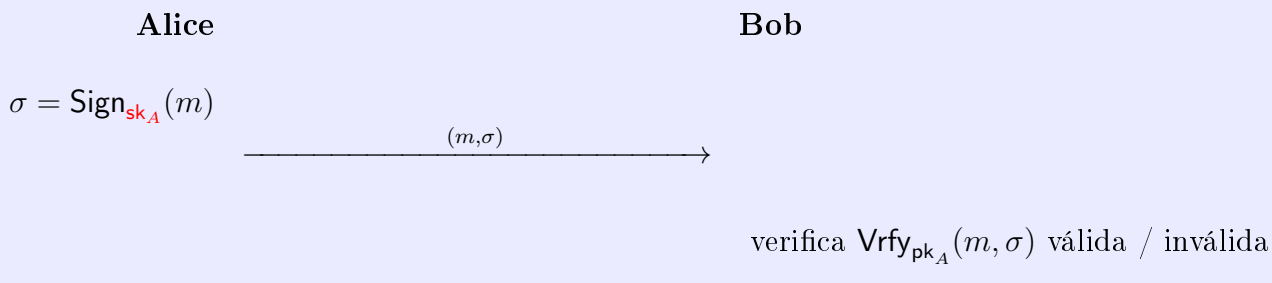
Consiste en tres algoritmos probabilísticos ($\mathsf{Gen}(\lambda)$, Sign , Vrfy)

- $\mathsf{Gen}(1^\lambda)$: la entrada λ es un parámetro de seguridad y la salida es el par de claves asimétricas $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sign}_{\mathsf{sk}}(m)$: recibe como entrada un mensaje m y la clave secreta sk. La salida es la firma digital σ del mensaje m.
- $\mathsf{Vrfy}_{\mathsf{pk}}(m, \sigma)$: recibe como entrada un mensaje m, la clave pública pk y la firma digital σ; devuelve 1 si "firma válida" o 0 para "firma inválida".

El esquema de firma digital es **seguro** si alguien que conoce pk (pero no sk) y muchas firmas válidas $(m_1, \sigma_1) \ldots (m_\ell, \sigma_\ell)$ no puede producir un nuevo mensaje m y una firma válida σ para él sin una complejidad $O(2^\lambda)$. Falsificación de firmas

2.5.0.2 Protocolo de Firma Digital

Alice va a firmar un mensaje m usando su clave secreta sk_A. Bob, de hecho cualquiera, conoce la clave pública de Alice pk_A generada usando $\mathsf{Gen}(1^\lambda)$.



This protocol also satisfies the characteristics we're looking for:

1. The signature is authentic; when Bob verifies the message with Alice's public key, he knows that she signed it.
2. The signature is unforgeable; only Alice knows her private key.
3. The signature is not reusable; the signature is a function of the document and cannot be transferred to any other document.
4. The signed document is unalterable; if there is any alteration to the document, the signature can no longer be verified with Alice's public key.
5. The signature cannot be repudiated. Bob doesn't need Alice's help to verify her signature.

Signing Documents and Timestamps

Actually, Bob can cheat Alice in certain circumstances. He can reuse the document and signature together. This is no problem if Alice signed a contract (what's another copy of the same contract, more or less?), but it can be very exciting if Alice signed a digital check.

Let's say Alice sends Bob a signed digital check for $100. Bob takes the check to the bank, which verifies the signature and moves the money from one account to the other. Bob, who is an unscrupulous character, saves a copy of the digital check. The following week, he again takes it to the bank (or maybe to a different bank). The bank verifies the signature and moves the money from one account to the other. If Alice never balances her checkbook, Bob can keep this up for years.

Consequently, digital signatures often include timestamps. The date and time of the signature are attached to the message and signed along with the rest of the message. The bank stores this timestamp in a database. Now, when Bob tries to cash Alice's check a second time, the bank checks the timestamp against its database. Since the bank already cashed a check from Alice with the same timestamp, the bank calls the police. Bob then spends 15 years in Leavenworth prison reading up on cryptographic protocols.

[Schneier15, page 38]

2.5.1 Identification Protocols and Fiat-Shamir Transform

Identification Schemes

An identification scheme is an interactive protocol that allows one party to prove its identity (i.e., to *authenticate* itself) to another. This is a very natural notion, and it is common nowadays to authenticate oneself when logging in to a website. We call the party identifying herself (e.g., the user) the "prover," and the party verifying the identity (e.g., the web server) the "verifier." Here, we are interested in the public-key setting where the prover and verifier do not share any secret information (such as a password) in advance; instead, the verifier only knows the public key of the prover. Successful execution of the identification protocol convinces the verifier that it is communicating with the intended prover rather than an imposter.

We will only consider three-round identification protocols of a specific form, where the prover is specified by two algorithms $\mathcal{P}_1, \mathcal{P}_2$ and the verifier's side of the protocol is specified by an algorithm $\mathcal{V}$. The prover runs $\mathcal{P}_1(sk)$ using its private key sk to obtain an initial message I along with some state st, and initiates the protocol by sending I to the verifier. In response, the verifier sends a challenge r chosen uniformly from some set Ω_{pk} defined by the prover's public key pk. Next, the prover runs $\mathcal{P}_2(sk, \mathsf{st}, r)$ to compute a response s that it sends back to the verifier. Finally, the verifier computes $\mathcal{V}(pk, r, s)$ and accepts if and only if this results in the initial message I; see Figure 12.1. Of course, for correctness we require that if the legitimate prover executes the protocol correctly then the verifier should always accept.

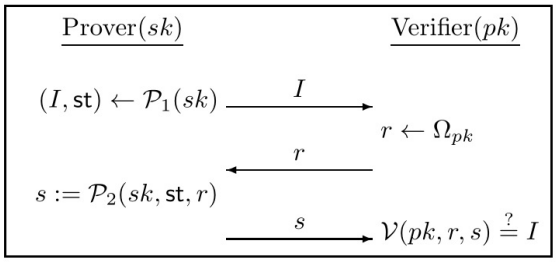


FIGURE 12.1: A three-round identification scheme.

[KatLin15, page 452]

I is called commitment.
r is the challenge.
s is the answer.

2.5.2 Protocolo de identificación de Schnorr

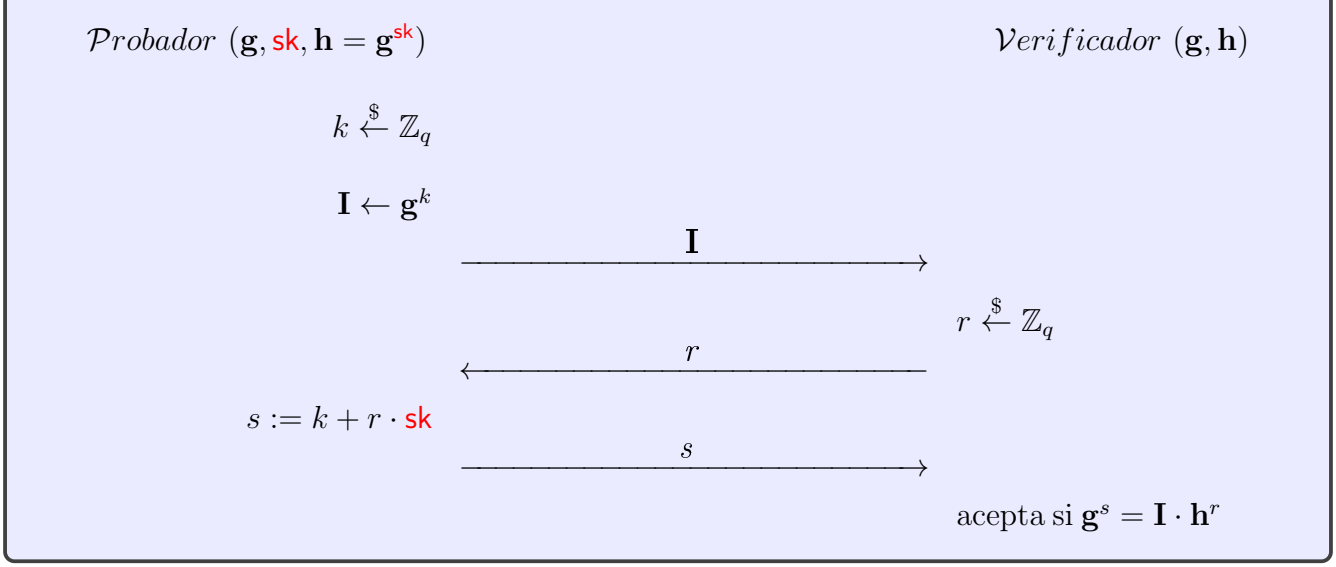
Los parámetros del dominio son como en el intercambio de claves Diffie-Hellman.

Es decir, $\mathsf{sk} \in \mathbb{Z}_q$ y $\mathsf{pk} = \mathbf{h} := \mathbf{g}^{\mathsf{sk}}$ son la clave secreta y pública del probador ($\mathcal{P}$) donde $\mathbf{g} \in \mathbb{F}_p^*$ tiene orden q.

Tanto el Probador como el Verificador ($\mathcal{V}$) conocen $\mathbf{g}, \mathbf{h} \in \mathbb{F}_p^*$.

$\mathcal{P}$ quiere demostrar a $\mathcal{V}$ que conoce sk sin revelarlo.

2.5.2.1 Protocolo Sigma de Identificación de Schnorr



El verificador acepta si el probador muestra una solución s de la ecuación "perturbada"

$$\mathbf{g}^x = \mathbf{I} \cdot \mathbf{h}^r \,.$$

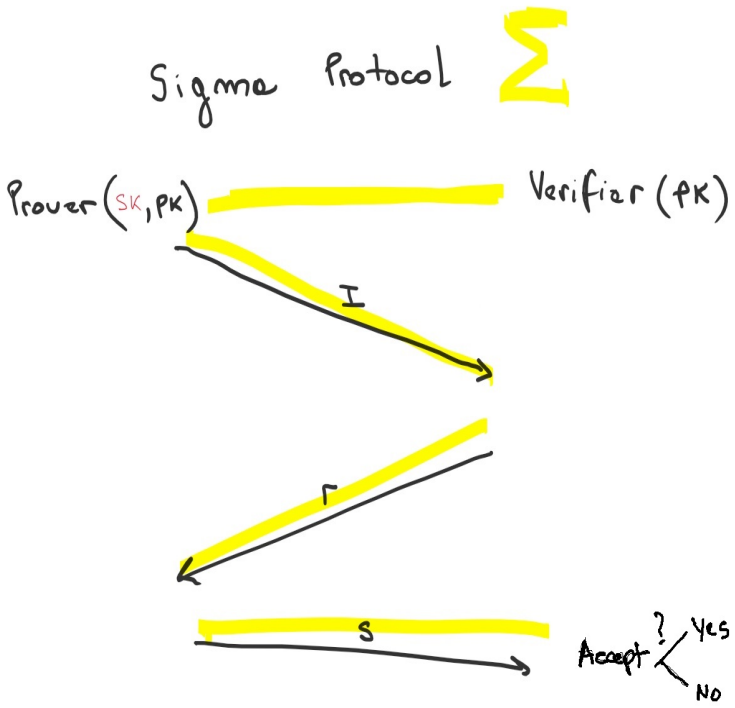
Perturbada con respecto a la ecuación $\mathbf{g}^x = \mathbf{h}$ cuya solución es $x = \mathsf{sk}$.

Nótese que si un probador falso puede adivinar el desafío r antes de comprometer su $\mathbf{I}$, entonces el protocolo de identificación se rompe.

Efectivamente, el probador falso puede elegir $\mathbf{I}$ por adelantado como $\mathbf{I} = \mathbf{g}^v \cdot \mathbf{h}^{-r}$ para un valor v que elija. Entonces el probador falso responde $s = v$. El verificador comprueba que

$$\mathbf{g}^v = \mathbf{I} \cdot \mathbf{h}^r = \mathbf{g}^v \cdot \mathbf{h}^{-r} \cdot \mathbf{h}^r$$

y acepta.



CONSTRUCTION 12.9

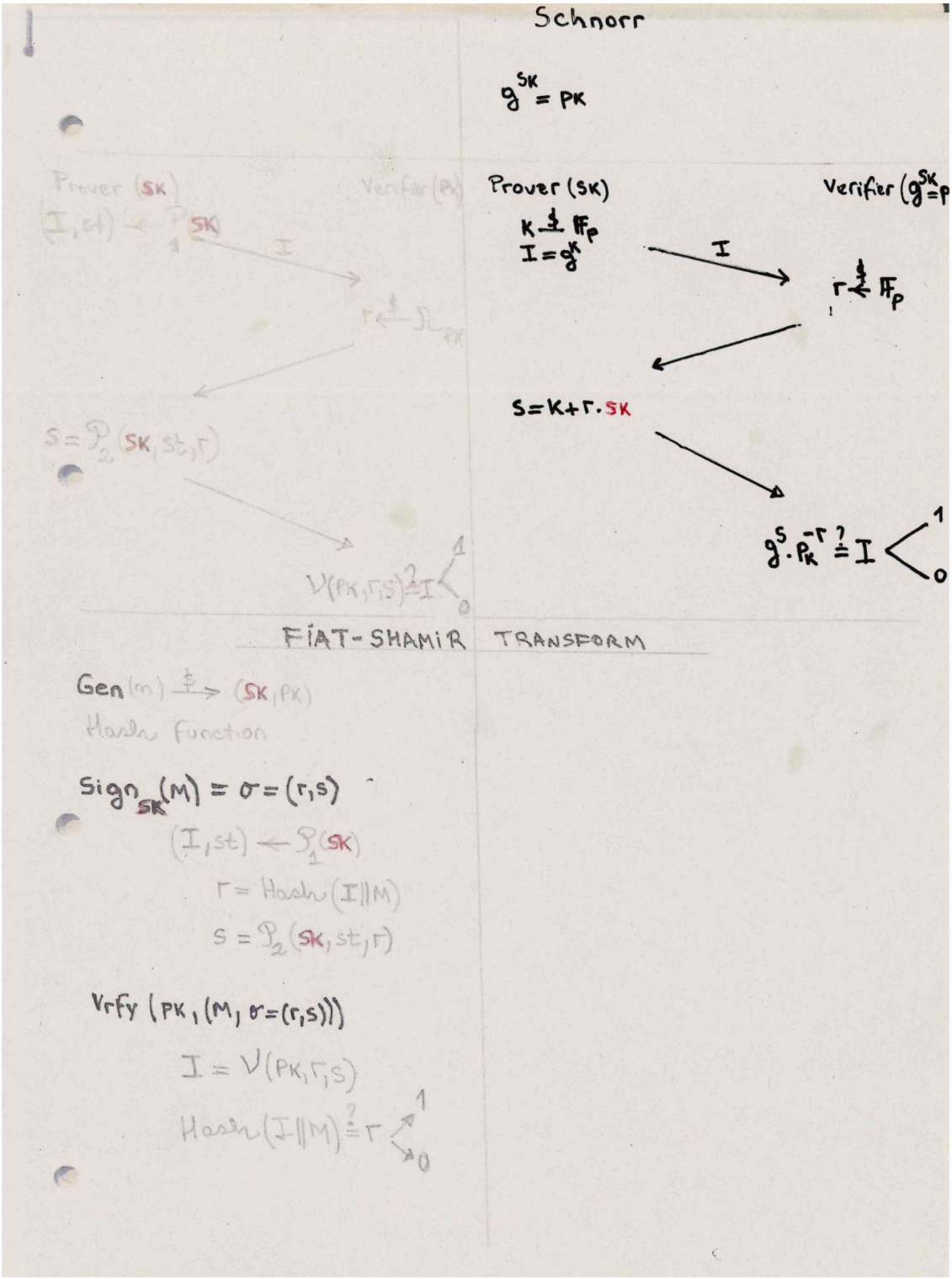
Let $(\mathsf{Gen}_{\mathsf{id}}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ be an identification scheme, and construct a signature scheme as follows:

- Gen: on input 1^n, simply run $\mathsf{Gen}_{\mathsf{id}}(1^n)$ to obtain keys pk, sk. The public key pk specifies a set of challenges Ω_{pk}. As part of key generation, a function $H : \{0,1\}^* \to \Omega_{pk}$ is specified, but we leave this implicit.
- Sign: on input a private key sk and a message $m \in \{0,1\}^*$, do:
  1. Compute $(I, \mathsf{st}) \leftarrow \mathcal{P}_1(sk)$.
  2. Compute $r := H(I, m)$.
  3. Compute $s := \mathcal{P}_2(sk, \mathsf{st}, r)$.

  Output the signature (r, s).
- Vrfy: on input a public key pk, a message m, and a signature (r, s), compute $I := \mathcal{V}(pk, r, s)$ and output 1 if and only if $H(I, m) \stackrel{?}{=} r$.

The Fiat–Shamir transform.

2.5.3 Schnorr Digital Signature as a Fiat-Shamir transform



Referencias

[DH76] Diffie, W.; Hellman, M. *New directions in cryptography*, (1976). IEEE Transactions on Information Theory. 22 (6): 644–654.

[Ko87] Koblitz, N. *Elliptic curve cryptosystems*, (1987). Mathematics of Computation. 48 (177): 203–209.

[Mi86] Miller, V. *Use of Elliptic Curves in Cryptography*, (1986). Advances in Cryptology – CRYPTO '85 Proceedings. Lecture Notes in Computer Science. Vol. 85. pp. 417–426

[Schneier15] Bruce Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C* ,Wiley; 20th Anniversary edition,2015.

[KatLin15] Jonathan Katz; Yehuda Lindell, *Introduction to Modern Cryptography* Second Edition, Chapman & Hall/CRC, Taylor & Francis Group, 2015.

[QUBIP23] QUBIP: *Transition to Post-Quantum Cryptography*
QUBIP project is co-funded by the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].
https://qubip.eu/
https://github.com/QUBIP
http://www.youtube.com/@qubip_eu