Máxima Seguridad para Firmas Digitales con Verificación Distribuida

Javier Herranz[1], Alexandre Ruiz[1], Germán Sáez[1]

***Abstract*—Una de las opciones para proteger el nivel de anonimato o privacidad de un firmante es construir firmas digitales con verificación distribuida: se requiere la colaboración de un subconjunto autorizado de usuarios para verificar la (in)validez de una firma. En RECSI'08, se propuso un esquema de este tipo, pero que no alcanzaba el máximo nivel de seguridad. En este trabajo proponemos el primer esquema de firma digital con verificación distribuida que consigue seguridad máxima, en términos de infalsificabilidad y privacidad. Demostramos formalmente estas dos propiedades por reducción a problemas computacionales estándar, en el modelo del oráculo aleatorio.**

***Index Terms*—Firma digital, compartición de secretos, modelo del oráculo aleatorio, indistinguibilidad**

I. Introducción

En algunas situaciones la propiedad de verificación universal en una firma digital puede ser no deseable, si el firmante desea un cierto nivel de anonimato o de privacidad. Una posible solución a este problema consiste en exigir la colaboración de varios usuarios para que el protocolo de verificación se pueda ejecutar correctamente. Este tipo de esquemas recibe el nombre de esquemas de firma con verificación distribuida, que pueden aplicarse en situaciones reales como subastas o votaciones electrónicas.

En [6], se definen las propiedades de seguridad (infalsificabilidad y privacidad) que debe satisfacer un esquema de firma con verificación distribuida. También se propone un esquema concreto, pero dicho esquema no alcanza el máximo nivel de seguridad respecto a la propiedad de privacidad.

En este trabajo proponemos el primer esquema de firma con verificación distribuida que satisface las máximas propiedades de seguridad. En particular, el esquema es seguro incluso ante atacantes que conocen las claves secretas de todos los participantes (excluido el participante que se está atacando). Conviene remarcar que esta propiedad (conocida como *insider security*, en inglés) no es en absoluto fácil de conseguir: incluso construcciones genéricas obtenidas al combinar un esquema de firma con un esquema de cifrado con descifrado distribuido no satisfacen este nivel máximo de seguridad. La definición detallada de estas propiedades de seguridad se puede encontrar en la Sección III. El diseño del nuevo esquema, que se presenta en la Sección IV, sigue las ideas del esquema de cifrado distribuido de Shoup y Gennaro [9]. En la Sección V, demostraremos formalmente las dos propiedades de seguridad, en el modelo del oráculo aleatorio, por reducción a dos problemas estándar: el problema del logaritmo discreto, y el problema Computacional de Diffie-Hellman.

[1]MAIV, UPC, Barcelona, Spain,
{jherranz,aruiz,german}@ma4.upc.edu

II. Esquemas de firma con verificación distribuida

Un esquema Σ de firma con verificación distribuida consiste en cuatro protocolos probabilísticos y de tiempo de ejecución polinómico:

1) ***Ini.*** La entrada es un parámetro de seguridad λ. Las salidas son unos parámetros públicos $\mathtt{params}$ utilizados en todo el esquema.

$$\Sigma.\mathsf{Ini}(1^\lambda) = \mathtt{params}$$

2) ***Gen_Cla.*** Este protocolo utiliza dos algoritmos. El primero corresponde al firmante A que obtendrá un par de claves (sk_A, pk_A), donde sk_A es la clave privada para firmar y pk_A es la correspondiente clave pública. El segundo algoritmo corresponde a un conjunto $\mathcal{B}$ de n verificadores, que tiene asociada una estructura de acceso (monótona creciente) $\Gamma_\mathcal{B} \subset 2^\mathcal{B}$, que contiene los subconjuntos autorizados a verificar. Estos usuarios obtendrán cierta información privada $\{sk_j\}_{j\in\mathcal{B}}$ que va a ser usada más tarde en el proceso de verificación distribuida, y cierto valor público $pk_\mathcal{B}$ común para el conjunto $\mathcal{B}$. El proceso de generación de claves para el colectivo $\mathcal{B}$ puede ser ejecutado por una tercera autoridad de confianza o de manera conjunta por ellos mismos, usando técnicas conocidas [3].

$$\Sigma.\mathsf{GC}(\mathtt{params}, A, \text{`individual'}) = (sk_A, pk_A)$$

$$\Sigma.\mathsf{GC}(\mathtt{params}, \mathcal{B}, \Gamma_\mathcal{B}, \text{`colectivo'}) = (\{sk_j\}_{j\in\mathcal{B}}, pk_\mathcal{B})$$

3) ***Firm.*** Este algoritmo es ejecutado por el firmante A; toma como entrada un mensaje m, su clave privada sk_A y la clave pública asociada a un grupo $\mathcal{B}$ de verificadores, y da como salida una firma $\theta(m)$ del mensaje.

$$\Sigma.\mathsf{Firm}(\mathtt{params}, m, pk_\mathcal{B}, sk_A) = \theta(m)$$

4) ***Ver_Dist.*** Dado $B \in \Gamma_\mathcal{B}$ un subconjunto autorizado de verificadores, este protocolo toma como entrada un mensaje m, una firma θ, la clave pública pk_A y los fragmentos sk_j de los usuarios $j \in B$. La salida será 1 si $\theta(m)$ es una firma válida de m y 0 en el caso contrario.

$$\Sigma.\mathsf{Ver}(\mathtt{params}, m, \theta, pk_A, B, \{sk_j\}_{j\in B}) = 1 \text{ ó } 0$$

Hay que remarcar que el primer y segundo protocolos se ejecutan sólo una vez. Los otros dos protocolos, i.e. el proceso de firma y de verificación, se ejecutan tantas veces como los participantes quieran firmar o verificar.

III. MODELO DE SEGURIDAD

Las propiedades de seguridad que un esquema de firma con verificación distribuida deberá satisfacer son las de *infalsificabilidad* y *privacidad*. Nuestro objetivo es considerar las nociones de seguridad más fuertes posibles. Por esta razón al adversario se le permite hacer peticiones de firma y verificación para diferentes usuarios, mensajes y firmas.

Además se le permite corromper al mayor número de participantes posibles (con la excepción de los usuarios que sean objetivo de su ataque en cada caso). En particular, la infalsificabilidad se alcanza incluso cuando el adversario conoce toda la información secreta de todos los participantes con la excepción del firmante que quiere atacar. Por otra parte, la privacidad se consigue incluso en el caso que el adversario conozca las claves secretas de todos los posibles firmantes y de un subconjunto no autorizado de verificadores. Este nivel de seguridad recibe en inglés el nombre de *insider security*.

A. Infalsificabilidad

La *infalsificabilidad existencial contra ataques de mensaje escogido* [5] requiere que cualquier atacante debe tener probabilidad despreciable [1] de falsificar una firma válida de un usuario (del cual no conoce su clave secreta), incluso si el atacante puede obtener previamente otros pares (mensaje,firma) válidos, para mensajes y conjuntos de verificadores que él escoge adaptativamente.

Esta propiedad se formaliza con el siguiente juego, en el que dado un parámetro de seguridad λ un retador externo reta a un atacante $\mathcal{F}_{\mathsf{INF}}$ para que intente falsificar una firma válida nueva:

1) El retador ejecuta $\mathtt{params} \leftarrow \Sigma.\mathsf{Ini}(1^\lambda)$ y da todos los valores obtenidos junto con una estructura de acceso Γ a $\mathcal{F}_{\mathsf{INF}}$.
2) $\mathcal{F}_{\mathsf{INF}}$ escoge un participante $A^\star$ para ser atacado. El retador ejecuta $(sk_{A^\star}, pk_{A^\star}) \leftarrow \Sigma.\mathsf{GC}(\mathtt{params}, A^\star, \text{'individual'})$, se guarda $sk_{A^\star}$ y le da $pk_{A^\star}$ a $\mathcal{F}_{\mathsf{INF}}$.
3) [Generación de nuevas claves] El atacante puede ejecutar $(sk_A, pk_A) \leftarrow \Sigma.\mathsf{GC}(\mathtt{params}, A, \text{'individual'})$ para firmantes $A \neq A^\star$ de su elección, y también puede ejecutar $\Sigma.\mathsf{GC}(\mathtt{params}, \mathcal{B}, \Gamma_{\mathcal{B}}, \text{'colectivo'}) = (\{sk_j\}_{j\in\mathcal{B}}, pk_{\mathcal{B}})$ para conjuntos $\mathcal{B}$ de su elección.
4) [Peticiones hash] Si la seguridad se considera en el modelo del oráculo aleatorio [1], $\mathcal{F}_{\mathsf{INF}}$ puede hacer peticiones al oráculo que modela el comportamiento de ciertas funciones hash.
5) [Peticiones firma] $\mathcal{F}_{\mathsf{INF}}$ puede escoger, de manera adaptativa, tuplas $(m_\ell, pk_{\mathcal{B}_\ell})$ y enviarlas a un oráculo de firma para el firmante $A^\star$. $\mathcal{F}_{\mathsf{INF}}$ obtiene como respuesta las firmas $\theta(m_\ell) \leftarrow \Sigma.\mathsf{Firm}(\mathtt{params}, m_\ell, pk_{\mathcal{B}_\ell}, sk_{A^\star})$.
6) [Falsificación] En un cierto momento, $\mathcal{F}_{\mathsf{INF}}$ publica un par $(m^\star, \theta^\star)$ y una clave $pk_{\mathcal{B}^\star}$ para un conjunto $\mathcal{B}^\star$ y una estructura de acceso $\Gamma_{\mathcal{B}^\star}$. El atacante $\mathcal{F}_{\mathsf{INF}}$ gana el juego si $(m^\star, \theta^\star) \neq (m_\ell, \theta(m_\ell))$, para toda firma obtenida durante el ataque, y además $\Sigma.\mathsf{Ver}(\mathtt{params}, m^\star, \theta^\star, pk_{A^\star}, B, \{sk_j\}_{j\in B}) = 1$, para algún subconjunto $B \in \Gamma_{\mathcal{B}^\star}$.

La ventaja de un adversario $\mathcal{F}_{\mathsf{INF}}$ en romper la infalsificabilidad de un esquema de firma con verificación distribuida se define como

$$\mathsf{Vent}_{\mathcal{F}_{\mathsf{INF}}}(\lambda) = \Pr[\mathcal{F}_{\mathsf{INF}} \text{ gana el juego}].$$

Definición 1: Un esquema de firma con verificación distribuida Σ es infalsificable si para cualquier adversario $\mathcal{F}_{INF}$ de tiempo polinómico, el valor $\mathsf{Vent}_{\mathcal{F}_{INF}}(\lambda)$ es despreciable con respecto al parámetro de seguridad λ.

B. Privacidad

Intuitivamente, en una firma digital con verificación distribuida se requiere que un atacante que corrompa a un subconjunto de usuarios no autorizado, no pueda obtener ninguna información sobre la (in)validez de las firmas calculadas por el usuario A. Para formalizar exactamente qué quiere decir 'no obtener ninguna información', se adapta el concepto de seguridad semántica (inicialmente introducido para esquemas de cifrado y conocido también como *indistinguibilidad contra ataques de cifrado escogido* [4] o IND-CCA). De manera informal, dados dos mensajes escogidos por un atacante, y una firma válida para uno de estos mensajes, el adversario no debe ser capaz de distinguir qué mensaje ha sido firmado con probabilidad significativamente mayor que $1/2$ (respuesta aleatoria).

Para formalizar esta idea intuitiva, detallamos aquí un juego de indistinguibilidad donde un atacante $\mathcal{F}_{\mathsf{IND}}$ intenta ganar a un retador externo:

1) El retador ejecuta $\mathtt{params} \leftarrow \Sigma.\mathsf{Ini}(1^\lambda)$ y da todos los valores obtenidos a $\mathcal{F}_{\mathsf{IND}}$.
2) $\mathcal{F}_{\mathsf{IND}}$ escoge un conjunto de verificadores $\mathcal{B}^\star$, una estructura de acceso $\Gamma_{\mathcal{B}^\star} \subset 2^{\mathcal{B}^\star}$ y un subconjunto no autorizado $\widetilde{B} \notin \Gamma_{\mathcal{B}^\star}$, cuyos usuarios puede corromper. El retador ejecuta el protocolo $(\{sk_j\}_{j\in\mathcal{B}^\star}, pk_{\mathcal{B}^\star}) \leftarrow \Sigma.\mathsf{GC}(\mathtt{params}, \mathcal{B}^\star, \Gamma_{\mathcal{B}^\star}, \text{'colectivo'})$, da al atacante $\mathcal{F}_{\mathsf{IND}}$ los valores $pk_{\mathcal{B}^\star}$ y $\{sk_j\}_{j\in\widetilde{B}}$, y mantiene el resto de valores sk_j en secreto.
3) [Generación de nuevas claves] El atacante puede ejecutar $(sk_A, pk_A) \leftarrow \Sigma.\mathsf{GC}(\mathtt{params}, A, \text{'individual'})$ para firmantes A de su elección, y también puede ejecutar $\Sigma.\mathsf{GC}(\mathtt{params}, \mathcal{B}, \Gamma_{\mathcal{B}}, \text{'colectivo'}) = (\{sk_j\}_{j\in\mathcal{B}}, pk_{\mathcal{B}})$ para parejas $(\mathcal{B}, \Gamma_{\mathcal{B}}) \neq (\mathcal{B}^\star, \Gamma_{\mathcal{B}^\star})$ de su elección.
4) [Peticiones hash] Si la seguridad se considera en el modelo del oráculo aleatorio, $\mathcal{F}_{\mathsf{IND}}$ puede hacer peticiones al oráculo que modela el comportamiento de ciertas funciones hash.
5) [Peticiones verificación] $\mathcal{F}_{\mathsf{IND}}$ escoge diferentes tuplas $(m_\ell, \theta_\ell, pk_{A_\ell})$ para firmantes A_ℓ de su elección y hace peticiones, de manera adaptativa, a un oráculo de verificación para estas firmas, con conjunto de verificadores $\mathcal{B}^\star$. $\mathcal{F}_{\mathsf{IND}}$ obtiene como respuesta toda la información emitida durante la ejecución del protocolo $\Sigma.\mathsf{Ver}(\mathtt{params}, m_\ell, \theta_\ell, pk_{A_\ell}, \mathcal{B}^\star, \{sk_j\}_{j\in\mathcal{B}^\star})$.

[1] Formalmente, decimos que una función f es *despreciable* (o *negligible*, en inglés) en k si existe un polinomio $p(\cdot)$ y un valor entero positivo k_0 tal que $f(k) \leq 1/p(k)$ para todo $k \geq k_0$. Usualmente, se escribe $f(k) = \mathrm{negl}(k)$ para las funciones f despreciables en k.

6) $\mathcal{F}_{\mathsf{IND}}$ escoge dos mensajes m_0, m_1 de la misma longitud y un firmante $A^\star$ con claves $(sk_{A^\star}, pk_{A^\star})$, que $\mathcal{F}_{\mathsf{IND}}$ envía al retador.
7) [Desafío] El retador escoge un bit aleatorio $b \in \{0,1\}$ y ejecuta $\theta^\star \leftarrow \Sigma.\mathsf{Firm}(\mathtt{params}, m_b, pk_{\mathcal{B}^\star}, sk_{A^\star})$. La firma resultante $\theta^\star$ se envía a $\mathcal{F}_{\mathsf{IND}}$.
8) [Más peticiones] Los pasos 4 y 5 son repetidos, con la restricción que las tuplas $(m_i, \theta^\star, pk_{A^\star})$ no pueden ser enviadas al oráculo de verificación, para $i = 0, 1$.
9) Finalmente, $\mathcal{F}_{\mathsf{IND}}$ devuelve un bit $b' \in \{0,1\}$.

Decimos que $\mathcal{F}_{\mathsf{IND}}$ gana el juego si $b' = b$. La *ventaja* de un tal adversario $\mathcal{F}_{\mathsf{IND}}$ en romper la privacidad de un esquema de firma con verificación distribuida se define como

$$\mathsf{Vent}_{\mathcal{F}_{\mathsf{IND}}}(\lambda) = |2\Pr[b' = b] - 1|\,.$$

Definición 2: Un esquema de firma con verificación distribuida Σ satisface la propiedad de privacidad si para cualquier adversario $\mathcal{F}_{IND}$ de tiempo polinómico, el valor $\mathsf{Vent}_{\mathcal{F}_{IND}}(\lambda)$ es despreciable con respecto al parámetro de seguridad λ.

IV. El esquema propuesto

En esta sección se describe un esquema específico de firma con verificación distribuida. El diseño de este nuevo esquema sigue las ideas del esquema de cifrado distribuido propuesto por Shoup y Gennaro [9]. Demostraremos que el nuevo esquema satisface las nociones máximas de seguridad de infalsificabilidad y privacidad. Para simplificar y por falta de espacio, consideramos el escenario donde los verificadores actúan correctamente en el proceso distribuido. Una modificación simple de nuestro esquema, incluyendo pruebas no interactivas de conocimiento cero sobre la igualdad de dos logaritmos discretos, permite añadir robustez al esquema para detectar participantes deshonestos en el proceso distribuido de verificación.

Detallamos a continuación los protocolos que componen nuestro esquema de firma con verificación distribuida Σ, para un firmante A y un conjunto $\mathcal{B} = \{1, \ldots, n\}$ de n verificadores.

1) ***Ini.*** $\Sigma.\mathsf{Ini}(1^\lambda)$.
Dado un parámetro de seguridad $\lambda \in \mathbb{N}$, se escogen dos números primos p y q tales que $|q| = \lambda$ y $q|(p-1)$. Se escoge también un grupo cíclico $\mathbb{G} = \langle g \rangle$ de orden primo q. Se escoge otro parámetro κ de seguridad, suficientemente grande (por ejemplo $\kappa = 160$) para evitar ataques de colisión al esquema. Posteriormente se escogen y publican cuatro funciones hash $H_0 : \{0,1\}^* \to \{0,1\}^\ell$, $H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \{0,1\}^* \to \mathbb{Z}_q$ y $H_3 : \{0,1\}^* \to \{0,1\}^\kappa$. En la demostración de seguridad, supondremos que las funciones hash H_0, H_1, H_2 se comportan como un oráculo aleatorio [1]. Del protocolo se obtienen los valores $\mathtt{params} = (p, q, \mathbb{G}, g, \ell, H_0, H_1, H_2, H_3)$.
2) ***Gen_Cla.*** $\Sigma.\mathsf{GC}(\mathtt{params}, A, \text{`individual'})$
$\Sigma.\mathsf{GC}(\mathtt{params}, \mathcal{B}, \Gamma_{\mathcal{B}}, \text{`colectivo'})$.
Para el firmante A, la clave secreta es un elemento aleatorio $x_A \in \mathbb{Z}_q^*$ que guarda de manera privada, mientras que la clave pública correspondiente es $y_A = g^{x_A} \bmod p$.
Para el colectivo $\mathcal{B}$ de n usuarios se publica el valor $y_{\mathcal{B}} = g^{x_{\mathcal{B}}}$ para un valor aleatorio $x_{\mathcal{B}} \in \mathbb{Z}_q^*$ que es desconocido para los miembros de $\mathcal{B}$. Cada verificador j de $\mathcal{B}$ recibe un fragmento s_j del secreto $x_{\mathcal{B}}$, correspondiente a un esquema de compartición de secretos de espacio vectorial [2] para la estructura de acceso $\Gamma_{\mathcal{B}}$. Es decir, para cada conjunto autorizado $B \in \Gamma_{\mathcal{B}}$ existen coeficientes $\{\lambda_j^B\}_{j \in B}$ tales que $\sum_{j \in B} \lambda_j^B s_j = x_{\mathcal{B}}$.
3) ***Firm.*** $\Sigma.\mathsf{Firm}(\mathtt{params}, m, y_{\mathcal{B}}, x_A)$.
Si A quiere firmar un mensaje $m \in \{0,1\}^*$, ejecuta los siguientes pasos:
   a) Escoge un valor aleatorio $r \in \mathbb{Z}_q^*$ y calcula $R = g^r \bmod p$.
   b) Calcula $k = H_0(R, y_{\mathcal{B}}, (y_{\mathcal{B}})^r, y_A)$ y $c = H_3(k, m)$.
   c) Elige valores aleatorios $\alpha_1, \alpha_2 \in \mathbb{Z}_q^*$ y calcula $Y_1 = g^{\alpha_1} \bmod p$, $Y_2 = g^{\alpha_2} \bmod p$.
   d) Calcula $\bar{g} = H_1(c, R, Y_1, Y_2, y_A, Y_{\mathcal{B}}) \in \mathbb{G}$, y posteriormente $\bar{R} = \bar{g}^r \bmod p$, $\bar{Y}_1 = \bar{g}^{\alpha_1} \bmod p$.
   e) Calcula $h = H_2(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, y_A, y_{\mathcal{B}})$.
   f) Calcula $s_1 = \alpha_1 - h \cdot r \bmod q$.
   g) Calcula $s_2 = \alpha_2 - h \cdot x_A \bmod q$.
   h) Devuelve la firma $\theta(m) = (c, R, \bar{R}, h, s_1, s_2)$.
4) ***Ver_Dist.*** $\Sigma.\mathsf{Ver}(\mathtt{params}, m, \theta, y_A, B, \{s_j\}_{j \in B})$.
Si los participantes de un subconjunto autorizado $B \in \Gamma_0$ quieren cooperar para verificar la firma $\theta(m) = (c, R, \bar{R}, h, s_1, s_2)$ del mensaje m, ejecutan los siguientes pasos.
   a) Cada verificador $j \in B$ calcula $\bar{g} = H_1(c, R, g^{s_1} \cdot R^h, g^{s_2} \cdot (y_A)^h, y_A, y_{\mathcal{B}})$ y comprueba entonces que la siguiente igualdad se verifica: $h = H_2(c, R, \bar{g}, \bar{R}, g^{s_1} \cdot R^h, g^{s_2} \cdot (y_A)^h, \bar{g}^{s_1} \cdot \bar{R}^h, y_A, y_{\mathcal{B}})$
   b) Si la igualdad no se verifica, j devuelve $(j, 0)$.
   c) En caso contrario, $j \in B$ devuelve el valor $T_j = R^{s_j} \bmod p$.
   d) Una vez se han recibido valores válidos T_j, diferentes de $(j, 0)$, correspondientes al subconjunto autorizado $B \in \Gamma_{\mathcal{B}}$, se recupera el valor $R^{x_{\mathcal{B}}}$: $\prod_{j \in B} T_j^{\lambda_j^B} = R^{x_{\mathcal{B}}} \bmod p$, donde $\lambda_j^B \in \mathbb{Z}_q$ son los coeficientes definidos por el esquema de compartición de secretos de espacio vectorial.
   e) Se calcula $k = H_0(R, y_{\mathcal{B}}, R^{x_{\mathcal{B}}}, y_A)$.
   f) Finalmente, para verificar la validez de la firma se comprueba la igualdad $c = H_3(k, m)$ devolviendo 1 si la igualdad es válida y 0 en caso contrario.

Intuitivamente, dada una firma $\theta(m) = (c, R, \bar{R}, h, s_1, s_2)$, los dos primeros elementos (c, R) son un código de autenticación del mensaje (MAC, en inglés) para el mensaje m, que puede ser verificado sólo si suficientes miembros de $\mathcal{B}$ cooperan. El resto de elementos $(\bar{R}, h, s_1, s_2)$ corresponden a una prueba de conocimiento cero del logaritmo discreto de y_A y de la igualdad entre los logaritmos discretos $\mathsf{LogDisc}_g(R) = \mathsf{DiscLog}_{\bar{g}}(\bar{R})$. La verificación de dicha prueba de conocimiento se realiza en los pasos a) y b) del

protocolo *Ver_Dist*.

V. ANÁLISIS DE SEGURIDAD

En este apartado demostramos que el esquema de firma con verificación distribuida propuesto en la sección anterior satisface las propiedades definidas en la Sección III, alcanzando pues el máximo nivel de seguridad que se puede exigir a este tipo de firmas.

La seguridad se considera en el modelo del oráculo aleatorio [1], donde el atacante puede hacer peticiones al oráculo que modela el comportamiento de ciertas funciones de hash.

Basaremos la seguridad del esquema propuesto en los siguientes problemas computacionales. Dado un parámetro de seguridad $\lambda \in \mathbb{N}$, un número primo q de λ bits y un grupo cíclico $\mathbb{G} = \langle g \rangle$ de orden primo q:

- El problema del *Logaritmo Discreto* (LD): para una entrada $(\mathbb{G}, y)$, tal que $y \in \mathbb{G}$, el objetivo del algoritmo $\mathcal{A}^{LD}$ es encontrar un entero $x \in \mathbb{Z}_q^*$ tal que $y = g^x$. Para un algoritmo en tiempo polinómico $\mathcal{A}^{LD}$ que recibe la tupla $(\mathbb{G}, y)$, definimos $\mathsf{Vent}_{\mathcal{A}^{LD}}(\lambda)$ como la probabilidad de que $\mathcal{A}$ encuentre ese valor $x \in \mathbb{Z}_q^*$ tal que $y = g^x$. La *hipótesis del Logaritmo Discreto* asume que el problema LD es difícil de resolver, es decir que $\mathsf{Vent}_{\mathcal{A}^{LD}}(\lambda)$ es depreciable en λ.
- El problema *Computacional de Diffie-Hellman* (CDH): para una entrada $(\mathbb{G}, g, g^a, g^b)$, tal que $a, b \in \mathbb{Z}_q^*$ son valores aleatorios, el objetivo del algoritmo $\mathcal{A}^{CDH}$ es calcular el valor de $g^{ab} \in \mathbb{G}$. Definimos $\mathsf{Vent}_{\mathcal{A}^{CDH}}(\lambda)$ y la *hipótesis Computacional de Diffie-Hellman* de manera análoga al problema LD.

La infalsificabilidad del nuevo esquema se basará en la dificultad de resolver el problema LD, mientras que la privacidad se basará en la dificultad de resolver el problema CDH.

A. Infalsificabilidad

Antes de proceder con la demostración de infalsificabilidad, recordamos una simplificación del *Forking Lemma*, introducido por Pointcheval y Stern en [7].

Lema 1: [Forking Lemma modificado] Consideremos un esquema de firma digital genérico [2] *con parámetro de seguridad λ. Dado un algoritmo $\mathcal{B}$ que obtiene una firma válida (m, R, h, s) con probabilidad al menos $\varepsilon(\lambda)$, existe otro algoritmo $\mathcal{B}'$ que utiliza $\mathcal{B}$ como subrutina y que produce con probabilidad $\varepsilon'(\lambda) \geq \mathcal{O}\left(\varepsilon(\lambda)^2\right)$ dos firmas válidas (m, R, h, s) y (m, R', h', s') tales que $h \neq h'$.*

A continuación demostramos por reducción que nuestro esquema de firma con verificación distribuida es infalsificable, en el modelo del oráculo aleatorio [1], basándonos en la suposición que el problema del logaritmo discreto es computacionalmente irresoluble en grupos de orden primo.

Teorema 1: Sea $\lambda \in \mathbb{N}$ un parámetro de seguridad. Para cualquier atacante $\mathcal{F}_{INF}$ contra la infalsificabilidad de nuestro esquema de firma con verificación distribuida, existe un algoritmo $\mathcal{A}^{LD}$ para el problema del logaritmo discreto, esencialmente con el mismo tiempo de ejecución que $\mathcal{F}_{INF}$, tal que

$$\mathsf{Vent}_{\mathcal{A}^{LD}}(\lambda) \geq \mathcal{O}\left(\mathsf{Vent}_{\mathcal{F}_{INF}}(\lambda)^2\right).$$

Proof: Asumiendo que tenemos un atacante $\mathcal{F}_{\mathsf{INF}}$ que tiene ventaja $\mathsf{Vent}_{\mathcal{F}_{\mathsf{INF}}}(\lambda)$ en romper la infalsificabilidad de nuestro esquema de firma, vamos a construir un algoritmo $\mathcal{A}^{LD}$, que va a ir ejecutando a su vez el atacante $\mathcal{F}_{\mathsf{INF}}$ como subrutina, simulando su entorno y respondiendo a sus peticiones. Aplicando el Lema 1 al atacante $\mathcal{F}_{\mathsf{INF}}$, el algoritmo $\mathcal{A}^{LD}$ será capaz de resolver el problema del logaritmo discreto.

$\mathcal{A}^{LD}$ recibe como entrada un grupo cíclico $\mathbb{G} = \langle g \rangle$ de orden primo q, junto con un valor $y \in \mathbb{G}$. El objetivo de $\mathcal{A}^{LD}$ es encontrar un entero $x \in \mathbb{Z}_q$ tal que $y = g^x$.

INICIALIZACIÓN DE $\mathcal{F}_{\mathsf{INF}}$. El protocolo $\Sigma.\mathsf{Ini}(1^\lambda)$ es ejecutado por $\mathcal{A}^{LD}$: éste da a $\mathcal{F}_{\mathsf{INF}}$ los valores $\mathtt{params} = (p, q, \mathbb{G}, g, \ell, H_0, H_1, H_2, H_3)$. Aquí las funciones hash $H_0 : \{0,1\}^* \to \{0,1\}^\ell$, $H_1 : \{0,1\}^* \to \mathbb{G}$ y $H_3 : \{0,1\}^* \to \{0,1\}^\kappa$ son elegidas arbitrariamente por $\mathcal{A}^{LD}$. Sin embargo, H_2 es modelada como un oráculo aleatorio y por ello $\mathcal{A}^{LD}$ mantendrá una tabla TAB_2 que servirá para responder a las peticiones hash de $\mathcal{F}_{\mathsf{INF}}$.

Para simular la ejecución del protocolo $\Sigma.\mathsf{GC}(\mathtt{params}, A^\star, \text{'individual'})$, para el firmante $A^\star$ escogido por $\mathcal{F}_{\mathsf{INF}}$, el algoritmo $\mathcal{A}^{LD}$ define la clave pública de $A^\star$ como $y_{A^\star} = y$ y se la envía a $\mathcal{F}_{\mathsf{INF}}$. Nótese que la correspondiente clave secreta $x_{A^\star}$, que es desconocida por $\mathcal{A}^{LD}$, es precisamente la solución buscada al problema del Logaritmo Discreto.

GENERACIÓN DE NUEVAS CLAVES. El atacante $\mathcal{F}_{\mathsf{INF}}$ puede generar libremente nuevas claves públicas y secretas para otros firmantes $A \neq A^\star$ y para colectivos $(\mathcal{B}, \Gamma_{\mathcal{B}})$ de verificadores de su elección.

PETICIONES HASH. Como la prueba es en el modelo del oráculo aleatorio para la función hash H_2, $\mathcal{F}_{\mathsf{INF}}$ puede hacer peticiones de esta función aleatoria. Para ello, $\mathcal{A}^{LD}$ crea y mantiene una tabla TAB_2 que responde de la siguiente manera a estas peticiones: la primera vez que se hace una petición, se escoge un valor aleatorio $h \in \mathbb{Z}_q$, se devuelve h a $\mathcal{F}_{\mathsf{INF}}$, y se guardan los valores de la petición junto con el valor devuelto h en TAB_2. Si la misma petición se hace en el futuro, buscamos en la tabla y $\mathcal{A}^{LD}$ responde con el mismo valor h que se encuentra en la salida existente.

PETICIONES FIRMA. Cuando $\mathcal{F}_{\mathsf{INF}}$ solicita firmas válidas para mensajes m_ℓ y claves públicas $y_{\mathcal{B}_\ell}$ de su elección, donde el firmante es $A^\star$ y $\mathcal{B}_\ell$ es el colectivo de verificadores, $\mathcal{A}^{LD}$ simula y devuelve firmas $\theta(m)$, de la siguiente manera:

1) Elige un valor aleatorio $r \in \mathbb{Z}_q^*$ que le sirve para calcular $R = g^r \bmod p$, $k = H_0(R, y_{\mathcal{B}_\ell}, (y_{\mathcal{B}_\ell})^r, y_{A^\star})$ y $c = H_3(k, m_\ell)$.
2) Elige valores aleatorios $h, s_1, s_2 \in \mathbb{Z}_q$ y calcula $Y_1 = g^{s_1} \cdot R^h \bmod p$, $Y_2 = g^{s_2} \cdot (y_{A^\star})^h \bmod p$.
3) Calcula $\bar{g} = H_1(c, R, Y_1, Y_2, y_{A^\star}, y_{B_\ell})$, y posteriormente $\bar{R} = \bar{g}^r \bmod p$, $\bar{Y}_1 = \bar{g}^{s_1} \cdot \bar{R}^h \bmod p$.

[2] Las firmas genéricas tienen la forma (m, R, h, s), donde R es un valor aleatorio escogido dentro de un conjunto muy grande (de tamaño exponencial en el parámetro de seguridad), y $h = H(m, R)$ para una función de hash H.

4) Si la entrada $(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, y_{A^\star}, y_{B_\ell})$ se encuentra en TAB_2 (lo que ocurre con probabilidad despreciable), se vuelve al paso 2.
5) $\mathcal{A}^{LD}$ 'falsifica' el oráculo aleatorio para H_2, imponiendo la relación $h = H_2(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, y_{A^\star}, y_{B_\ell})$ en TAB_2. Más tarde, si $\mathcal{F}_{\mathsf{INF}}$ llamara al oráculo aleatorio con esa misma entrada, se le devolvería el valor h.
6) Devuelve la firma $\theta(m_\ell) = (c, R, \bar{R}, h, s_1, s_2)$ a $\mathcal{F}_{\mathsf{INF}}$.

Es fácil comprobar que la firma devuelta es consistente, si no existen colisiones a la hora de 'falsificar' el oráculo aleatorio.

FALSIFICACIÓN. En algún momento $\mathcal{F}_{\mathsf{INF}}$ produce con probabilidad $\mathsf{Vent}_{\mathcal{F}_{\mathsf{INF}}}(\lambda)$ una clave $y_{\mathcal{B}^\star}$ y una firma falsificada $(m^\star, \theta^\star)$ para un conjunto de verificadores $(\mathcal{B}^\star, \Gamma_{\mathcal{B}^s tar})$, donde $\theta^\star = (c^\star, R^\star, \bar{R}^\star, h^\star, s_1^\star, s_2^\star)$ verifica las siguientes dos propiedades. Primero, la firma $(m^\star, \theta^\star)$ debe ser diferente a las solicitadas anteriormente durante las peticiones de firma y segundo, se debe verificar $\Sigma.\mathsf{Ver}(\mathtt{params}, m^\star, \theta^\star, y_{A^\star}, B, \{s_j\}_{j\in B}) = 1$, para algún subconjunto $B \in \Gamma_{\mathcal{B}^\star}$.

Puesto que la firma falsificada es válida, obtenemos que $h = H_2(c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, y_{A^\star}, y_{\mathcal{B}^\star})$, donde $Y_1^\star = g^{s_1^\star} \cdot (R^\star)^{h^\star}$, $Y_2^\star = g^{s_2^\star} \cdot (y_{A^\star})^{h^\star}$, $\bar{Y}_1^\star = (\bar{g}^\star)^{s_1^\star} \cdot \left(\bar{R}^\star\right)^{h^\star}$.

Además, puesto que esta falsificación es diferente de las firmas obtenidas durante las peticiones de firma, podemos estar seguros que la entrada $\mathsf{petición}^\star = (c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, y_{A^\star}, y_{B^\star})$ para H_2 no ha sido 'falsificada' y añadida a TAB_2 por $\mathcal{A}^{LD}$.

REPLICANDO EL ATAQUE. Ahora podemos aplicar las técnicas de replicación del Forking Lemma modificado, descrito en el enunciado del Lema 1. De manera informal, $\mathcal{A}^{LD}$ repetirá la ejecución del adversario $\mathcal{F}_{\mathsf{INF}}$, con la misma aleatoriedad pero cambiando los valores salida del oráculo aleatorio H_2 para la entrada $\mathsf{petición}^\star$.

De esta manera, después que $\mathcal{A}^{LD}$ ejecute dos veces a $\mathcal{F}_{\mathsf{INF}}$, obtendremos con probabilidad cuadrática en $\mathsf{Vent}_{\mathcal{F}_{\mathsf{INF}}}(\lambda)$ (la probabilidad de obtener la primera firma falsificada) dos firmas válidas $\theta^\star = (c^\star, R^\star, \bar{R}^\star, h^\star, s_1^\star, s_2^\star)$ y $\theta'^\star = (c'^\star, R'^\star, \bar{R}'^\star, h'^\star, s'^\star_1, s'^\star_2)$ para los mismos valores $(c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, y_{A^\star}, y_{B^\star})$ de H_2 tales que $h \neq h'$.

Puesto que las dos firmas son válidas, tenemos que

$$g^{s_2^\star} \cdot (y_{A^\star})^{h^\star} = Y_2^\star = g^{s'^\star_2} \cdot (y_{A^\star})^{h'^\star},$$

lo cual implica que obtenemos la siguiente relación para el valor inicial $y = y_{A^\star} = \left(g^{s_2^\star - s_2'^\star}\right)^{1/(h'^\star - h^\star)}$.

Por tanto, $\mathcal{A}^{LD}$ devuelve el valor $x = \frac{s_2^\star - s_2'^\star}{h'^\star - h^\star} \bmod q$ y resuelve el problema del logaritmo discreto de y en base g, con probabilidad de éxito $\mathsf{Vent}_{\mathcal{A}^{LD}}(\lambda) \geq \mathcal{O}\left(\mathsf{Vent}_{\mathcal{F}_{\mathsf{INF}}}(\lambda)^2\right)$. ■

B. Privacidad

En el siguiente teorema demostraremos que nuestro esquema de firma con verificación distribuida verifica la propiedad de privacidad definida en la Sección III, por reducción al problema computacional de Diffie-Hellman en grupos de orden primo.

Teorema 2: Sea $\lambda \in \mathbb{N}$ un parámetro de seguridad. Para cualquier atacante $\mathcal{F}_{\mathsf{IND}}$ contra la privacidad de nuestro esquema de firma con verficación distribuida, existe un solucionador $\mathcal{A}^{CDH}$ del problema computacional de Diffie-Hellman, esencialmente con el mismo tiempo de ejecución que $\mathcal{F}_{\mathsf{IND}}$, tal que

$$\mathsf{Vent}_{\mathcal{A}^{CDH}}(\lambda) \geq \mathsf{Vent}_{\mathcal{F}_{\mathsf{IND}}}(\lambda)/2.$$

Proof: Asumiendo que tenemos un atacante $\mathcal{F}_{\mathsf{IND}}$ que tiene ventaja $\mathsf{Vent}_{\mathcal{F}_{\mathsf{IND}}}(\lambda)$ en romper la privacidad de nuestro esquema de firma, vamos a construir un algoritmo $\mathcal{A}^{CDH}$ que irá usando a su vez a $\mathcal{F}_{\mathsf{IND}}$ como subrutina, para resolver el problema Computacional de Diffie-Hellman.

El algoritmo $\mathcal{A}^{CDH}$ recibe como entrada un grupo cíclico $\mathbb{G} = \langle g \rangle$ de orden primo q, junto con una tupla (g, g^a, g^b). El objetivo de $\mathcal{A}^{CDH}$ es calcular g^{ab}.

INICIALIZACIÓN DE $\mathcal{F}_{\mathsf{IND}}$. El adverdario $\mathcal{F}_{\mathsf{IND}}$ escoge un conjunto de verificadores $\mathcal{B}^\star = \{1, \ldots, n\}$, una estructura de acceso $\Gamma_{\mathcal{B}^\star} \subset 2^{\mathcal{B}^\star}$ y un subconjunto de participantes corruptos $\widetilde{B} \notin \Gamma_{\mathcal{B}^\star}$.

$\mathcal{A}^{CDH}$ simula una ejecución del protocolo $\Sigma.\mathsf{Ini}(1^\lambda)$: da a $\mathcal{F}_{\mathsf{IND}}$ los valores $\mathtt{params} = (p, q, \mathbb{G}, g, \ell, H_0, H_1, H_2, H_3)$, cuyas funciones hash H_0, H_1 y H_2 las modelará como oráculos aleatorios, mientras que la función $H_3 : \{0,1\}^* \to \{0,1\}^\kappa$ la define explícitamente. Por tanto, $\mathcal{A}^{CDH}$ mantendrá tres tablas TAB_0, TAB_1 y TAB_2 que servirán para responder a las peticiones hash de $\mathcal{F}_{\mathsf{IND}}$.

La ejecución del protocolo $\Sigma.\mathsf{GC}(\mathtt{params}, \mathcal{B}^\star, \Gamma_{\mathcal{B}^\star}, \text{'colectivo'})$ es simulada por $\mathcal{A}^{CDH}$ de la siguiente manera: los fragmentos de los participantes corruptos, $\{s_j\}_{j\in\widetilde{B}}$, son elegidos aleatoria e independientemente en $\mathbb{Z}_q$ y dados posteriormente a $\mathcal{F}_{\mathsf{IND}}$. En este punto, $\mathcal{A}^{CDH}$ define la clave pública $y_{B^\star} = g^b$ que también se envía a $\mathcal{F}_{\mathsf{IND}}$. Remarcamos que esto significa que la clave secreta $x_{B^\star}$ asociada está definida implícitamente como b).

GENERACIÓN DE NUEVAS CLAVES. El atacante $\mathcal{F}_{\mathsf{IND}}$ puede generar libremente nuevas claves públicas y secretas para firmantes A y para colectivos $(\mathcal{B}, \Gamma_{\mathcal{B}}) \neq (\mathcal{B}^\star, \Gamma_{\mathcal{B}^\star})$ de verificadores de su elección.

PETICIONES HASH. Como la prueba es en el modelo del oráculo aleatorio para las funcioness hash H_0, H_1 y H_2, $\mathcal{A}^{CDH}$ crea y mantiene tres tablas TAB_0, TAB_1, y TAB_2 que simulan estas funciones hash. Para responder a las peticiones hash solicitadas por $\mathcal{F}_{\mathsf{IND}}$, el atacante $\mathcal{A}^{CDH}$ comprueba si ya existe una entrada en la correspondiente tabla para la entrada de esa petición. En ese caso, la salida existente es enviada a $\mathcal{F}_{\mathsf{IND}}$. En caso contrario, una nueva salida es elegida aleatoriamente para ser enviada a $\mathcal{F}_{\mathsf{IND}}$. Posteriormente, la relación entre la entrada y la salida es añadida a la correspondiente tabla.

Para peticiones de H_1, $\mathcal{A}^{CDH}$ elige un valor aleatorio $\beta \in \mathbb{Z}_q^*$ y devuelve el valor $\bar{g} = (g^b)^\beta$ como nueva salida de H_1. El

valor β es guardado como valor adicional de la nueva entrada en la tabla TAB_1.

En el caso que $\mathcal{A}^{CDH}$ reciba peticiones de H_0 cuyos primeros valores sean g^a y g^b, el tercer elemento de la entrada será guardado en una tabla adicional $\mathsf{TAB}^\star$. La tabla $\mathsf{TAB}^\star$ será la respuesta final de $\mathcal{A}^{CDH}$ al problema CDH.

PETICIONES VERIFICACIÓN. Cuando $\mathcal{F}_{\mathsf{IND}}$ solicita una petición de verificación $(m_\ell, \theta_\ell, y_{A_\ell})$ para firmantes A_ℓ, con $\theta_\ell = (c_\ell, R_\ell, \bar{R}_\ell, h_\ell, s_{1\ell}, s_{2\ell})$, $\mathcal{A}^{CDH}$ comprueba la validez de la prueba de conocimiento cero $(\bar{R}_\ell, h_\ell, s_{1\ell}, s_{2\ell})$. Para ello, obtiene mediante una petición hash el valor $\bar{g}_\ell = H_1(c_\ell, R_\ell, g^{s_{1\ell}} \cdot R_\ell{}^{h_\ell}, g^{s_{2\ell}} \cdot (y_{A_\ell})^{h_\ell}, y_{A_\ell}, y_{B^\star}) = (g^b)^{\beta_\ell}$ y verifica que se cumpla $h_\ell = H_2(c_\ell, R_\ell, \bar{g}_\ell, \bar{R}_\ell, g^{s_{1\ell}} \cdot R_\ell{}^{h_\ell}, g^{s_{2\ell}} \cdot (y_{A_\ell})^{h_\ell}, \bar{g}_\ell^{s_{1\ell}} \cdot \bar{R}_\ell^{h_\ell}, y_{A_\ell}, y_{B^\star})$. En el caso que no se cumpla, la respuesta de $\mathcal{A}^{CDH}$ a la petición será 0.

En caso contrario, $\mathcal{A}^{CDH}$ debe calcular los valores $\{R_\ell^{s_j}\}_{j\in\mathcal{B}^\star}$ y enviárselos a $\mathcal{F}_{\mathsf{IND}}$. Para los participantes corruptos $j \in \widetilde{B}$, $\mathcal{A}^{CDH}$ puede calcular estos valores fácilmente ya que conoce $\{s_j\}_{j\in\widetilde{B}}$. Además, como es válida la prueba de conocimiento de la igualdad de los logaritmos $\mathsf{LogDisc}_g(R_\ell) = \mathsf{LogDisc}_{\bar{g}_\ell}(\bar{R}_\ell)$, siendo $\bar{g}_\ell = g^{b\beta_\ell}$, tenemos que $R_\ell^{b\beta_\ell} = \bar{R}_\ell$. Por tanto, $\mathcal{A}^{CDH}$ puede calcular $R_\ell^{x_{B^\star}} = R_\ell^b = \bar{R}_\ell^{1/\beta_\ell}$. Conociendo $R_\ell^{x_{B^\star}}$ y $\{R_\ell^{s_j}\}_{j\in\tilde{B}}$, el algoritmo $\mathcal{A}^{CDH}$ puede usar en el exponente las relaciones lineales entre fragmentos y secretos, determinadas por el esquema para compartir secreto que realize $\Gamma_{\mathcal{B}^\star}$, y obtener el resto de valores $\{R_\ell^{s_j}\}_{j\in\mathcal{B}^\star\setminus\widetilde{B}}$.

Finalmente, $\mathcal{A}^{CDH}$ ejecuta una petición hash $H_0(R_\ell, y_{B^\star}, R_\ell^{x_{B^\star}}, y_{A_\ell}) = k_\ell$, actualizando la tabla TAB_0 en caso que esta petición sea nueva, y verifica si $H_3(k_\ell, m_\ell) = c_\ell$. Según el resultado de esa verificación, $\mathcal{A}^{CDH}$ devuelve la salida 1 (firma válida) o 0 (firma no válida).

DESAFÍO. En un momento dado, $\mathcal{F}_{\mathsf{IND}}$ escoge y publica dos mensajes m_0, m_1 de la misma longitud, junto con un firmante $A^\star$ con valores $(x_{A^\star}, y_{A^\star})$. Ahora $\mathcal{A}^{CDH}$ debe enviar una firma $\theta^\star$ a $\mathcal{F}_{\mathsf{IND}}$, que obtiene de la siguiente manera: primero define $R^\star = g^a$ y elige valores aleatorios $c^\star \in \{0,1\}^\ell$, $h^\star, s_1^\star, s_2^\star \in \mathbb{Z}_q$ y $\beta^\star \in \mathbb{Z}_q^*$. Con estos valores $\mathcal{A}^{DH}$ define $\bar{g}^\star = g^{\beta^\star}$, $\bar{R}^\star = (g^a)^{\beta^\star}$, $Y_1^\star = g^{s_1^\star} \cdot (R^\star)^{h^\star}$, $Y_2^\star = g^{s_2^\star} \cdot (y_{A^\star})^{h^\star}$ y $\bar{Y}_1^\star = \bar{g}^{s_1^\star} \cdot (\bar{R}^\star)^{h^\star}$.

Si la entrada $(c^\star, R^\star, Y_1^\star, Y_2^\star, y_{A^\star}, y_{B^\star})$ ya existe en TAB_1, o la entrada $(c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, y_{i^\star}, D_{i^\star}^{(i^\star)})$ ya existe en TAB_2, entonces $\mathcal{A}^{DH}$ elige nuevos valores aleatorios $c^\star, h^\star, s_1^\star, s_2^\star$ y $\beta^\star$. Por último, las relaciones $\bar{g}^\star = H_1(c^\star, R^\star, Y_1^\star, Y_2^\star, y_{A^\star}, y_{B^\star})$ y $h^\star = H_2(c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, y_{A^\star}, y_{B^\star})$ son añadidas a las tablas TAB_1 y TAB_2 respectivamente.

La firma final $\theta^\star = (c^\star, R^\star, \bar{R}^\star, h^\star, s_1^\star, s_2^\star)$ es enviada al atacante $\mathcal{F}_{\mathsf{IND}}$.

MÁS PETICIONES. El atacante $\mathcal{F}_{\mathsf{IND}}$ puede hacer más peticiones hash y de verificación, que son respondidas de igual manera a como ha sido descrito anteriormente.

El único problema podría aparecer cuando $\mathcal{F}_{\mathsf{IND}}$ pidiese una verificación de una firma válida (m, θ, y_A), con $\theta = (c, R, \bar{R}, h, s_1, s_2)$ tal que el valor $\bar{g} = H_1(c, R, g^{s_1} \cdot R^h, g^{s_2} \cdot (y_A)^h, y_A, y_{B^\star})$ coincidiese con el valor $\bar{g}^\star = g^{\beta^\star}$, ya que este último valor no tiene por qué ser de la forma $(g^b)^\beta$. Sin embargo, esto ocurriría únicamente cuando los valores de entrada de H_1 tanto para esta firma como para la firma del desafío fuesen los mismos. Puesto que las pruebas de conocimiento cero son válidas, obtendríamos que los valores $\bar{R}$ son iguales en ambos casos y, por tanto, que los valores h, s_1, s_2, y_A también deberían ser iguales. La conclusión es que la firma solicitada θ sería la misma que la firma $\theta^\star$ del desafío. Como esto está prohibido por definición, nunca nos encontraremos con este caso.

ANÁLISIS FINAL. $\mathcal{F}_{\mathsf{IND}}$ devuelve un bit b' con el que gana el juego con una probabilidad significativamente mayor que $1/2$. Puesto que H_0 se comporta como una función aleatoria, $\mathcal{F}_{\mathsf{IND}}$ puede ganar sólo si $\mathcal{F}_{\mathsf{IND}}$ ha preguntado anteriormente al oráculo aleatorio el valor $H_0(g^a, g^b, g^{ab}, y_{A^\star})$ correspondiente al desafío $\theta^\star$. Por tanto, con una probabilidad no despreciable $\mathsf{Vent}_{\mathcal{F}_{\mathsf{IND}}}(\lambda)/2$, el valor g^{ab} está en la tabla $\mathsf{TAB}^\star$ construida por $\mathcal{A}^{CDH}$, que contiene los candidatos a solución del problema CDH. Tal y como indican los autores de [9], podríamos usar el auto-corrector Diffie-Hellman descrito en [8] para transformar al atacante $\mathcal{A}^{CDH}$ en otro que en vez de devolver toda una lista de candidatos $\mathsf{TAB}^\star$, devuelve únicamente la solución correcta del problema CDH. ■

Cabe remarcar que en contraposición al esquema propuesto en [6], que sólo asolía la privacidad débil, el esquema propuesto aquí es seguro incluso frente a adversarios que pueden hacer peticiones a un oráculo de verificación.

REFERENCES

[1] M. Bellare y P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *Proceedings of CCS'93*, ACM Press, pp. 62–73 (1993).

[2] E.F. Brickell. Some ideal secret sharing schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **9**, pp. 105–113 (1989).

[3] R. Gennaro, S. Jarecki, H.Krawczyk and T. Rabin. Secure distributed key generation for Discrete-Log based cryptosystems. *Journal of Cryptology*, vol. **4** (1), Springer-Verlag, pp. 51–83 (2007).

[4] S. Goldwasser y S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, **28**, pp. 270–299 (1984).

[5] S. Goldwasser, S. Micali y R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. of Computing* **17** (2), pp. 281–308 (1988).

[6] J. Herranz, A. Ruiz y G. Sáez. Esquemas de firma digital con verificación distribuida. *Actas de la X Reunión Española de Criptología y Seguridad de la Información, RECSI'08*, pp. 209–216 (2008).

[7] D. Pointcheval y J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology* **13** (3), Springer-Verlag, pp. 361–396 (2000).

[8] V. Shoup. Lower bounds for discrete logarithms and related problems. *Proceedings of Eurocrypt'97*, LNCS **1233**, Springer-Verlag, pp. 256–266 (1997).

[9] V. Shoup y R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, vol. **15** (2), Springer-Verlag, pp. 75–96 (2002).