Deber 2 - Seguridad Informática

Universidad San Francisco de Quito

Xavier Sebastián Tandazo Cobo

13 de noviembre de 2025

Pregunta 1: Firma Digital Estándar y Análisis de Mecanismos Alternativos

0.1 Protocolo Estándar de Firma Digital

Para poder ilustrar el funcionamiento del protocolo estándar de firma digital de una manera entendible, consideremos el siguiente ejemplo:

> Alice desea enviar a Bob el mensaje: *"Transferir $100 a la cuenta de Mark"*. Ella quiere garantizar que Bob pueda verificar que el mensaje proviene realmente de ella y que no ha sido modificado durante la transmisión.

Como se explica en el enunciado del deber, este proceso se desarrolla en cuatro etapas principales:

Protocolo Estándar de Firma Digital

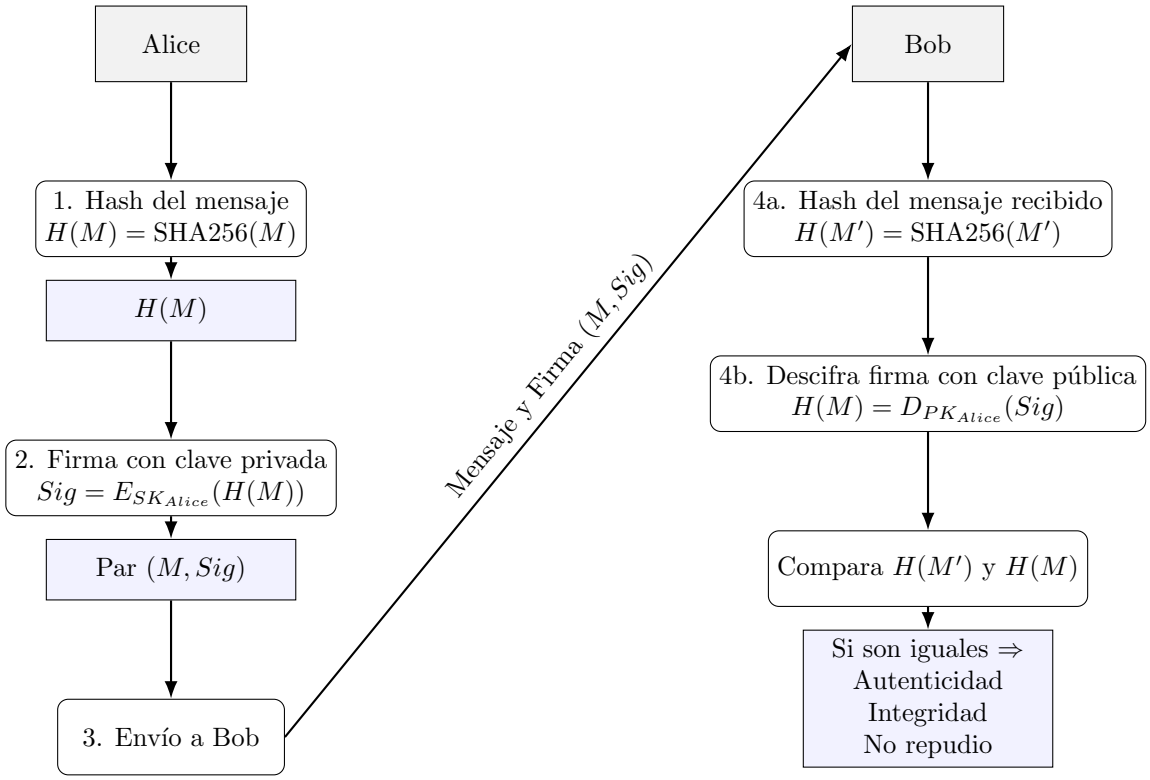

Figure 1: Flujo del protocolo estándar de firma digital entre Alice y Bob.

1. Digest Computation

Alice aplica una función hash criptográfica segura (diremos que la SHA-256) sobre el mensaje M:

$$H(M) = \text{SHA256}(\text{“Transferir \$100 a la cuenta Mark”})$$

El resultado es un valor de longitud fija, por ejemplo:

$$H(M) = \texttt{"hash-unico"}$$

Un hash representa de forma única el contenido de un mensaje y cualquier cambio, por más mínimo, produciría un hash completamente diferente.

2. Signing

Alice cifra su hash con su **clave privada** SK_{Alice}:

$$Sig = E_{SK_{Alice}}(H(M))$$

Este valor cifrado (Sig) constituye la **firma digital**, pero solo Alice, al poseer su clave privada, puede generar este valor específico.

3. Transmission

Alice envía a Bob el par (M, Sig), es decir, el mensaje y la firma:

$$(\text{“Transferir \$100 a la cuenta 1234”},\ Sig)$$

El mensaje no necesita ser cifrado, ya que la firma digital garantiza autenticación e integridad, no confidencialidad.

4. Verification

Bob realiza dos operaciones:

1. Calcula el hash del mensaje recibido:

$$H(M') = \text{SHA256}(\text{“Transferir \$100 a la cuenta 1234”})$$

2. Descifra la firma usando la **clave pública** de Alice:

$$H(M) = D_{PK_{Alice}}(Sig)$$

Si $H(M') = H(M)$, Bob puede concluir que:

- El mensaje proviene de Alice.
- No fue modificado.
- Alice no puede negar haberlo firmado.

0.2 Análisis de Mecanismos Alternativos

Siguiendo el objetivo del enunciado, se analizan dos esquemas alternativos que no cumplen con las propiedades criptográficas requeridas.

Mecanismo 1: Uso de criptografía simétrica

> En este esquema, se propone utilizar un sistema de clave simétrica en lugar de un esquema basado en criptografía de clave pública. El firmante y el receptor comparten una clave secreta común, usada tanto para generar como para verificar la “firma”.

A primera vista, este mecanismo puede parecer funcional, pero no cumple con las propiedades esenciales de una firma digital. La autenticación simétrica depende de confianza mutua, mientras que la firma digital extiende esa confianza a terceros, garantizando prueba verificable del origen del mensaje. De acuerdo con los principios descritos en *Cryptography and Network Security*, este método puede autenticar, pero no ofrecer **no repudio** ni verificación independiente.

Problemas y propiedades violadas:

- **Pérdida del no repudio:** Ambos conocen la misma clave, por lo que cualquiera podría haber generado la firma.
- **Dependencia del firmante en la verificación:** El receptor necesita que el emisor valide la firma, rompiendo la independencia.
- **Riesgo de manipulación:** El receptor podría alterar el mensaje y solicitar una verificación, obteniendo firmas sobre contenido modificado.

Ejemplo: Si Alice y Bob comparten una clave K_{AB}, Alice "firma" el mensaje cifrando su hash:

$$S = E_{K_{AB}}(H(M))$$

Bob no puede verificarla sin reenviarla a Alice, eliminando la verificación independiente. Por tanto, este esquema **no cumple autenticación independiente ni no repudio**.

Mecanismo 2: Cifrado completo del mensaje con la clave privada

> En este mecanismo, el firmante cifra el mensaje completo M con su clave privada, y el receptor lo descifra con la clave pública para obtener el mensaje y su autenticidad.

Aun cuando parece ofrecer autenticidad, este esquema confunde cifrado con firma digital. Cifrar con la clave privada no protege la confidencialidad (ya que todos poseen la clave pública) y además es ineficiente. Según los principios del libro de Stallings, la firma debe aplicarse sobre un *hash* del mensaje, no sobre el mensaje completo.

Problemas y propiedades violadas:

- **Ineficiencia:** Cifrar mensajes grandes con algoritmos asimétricos es costoso.
- **Pérdida de confidencialidad:** Cualquier persona con la clave pública puede descifrar el contenido.

Ejemplo: Alice cifra el mensaje:

$$C = E_{K_{priv}}(M)$$

Bob lo descifra con la clave pública:

$$M = D_{K_{pub}}(C)$$

El resultado es el mensaje original, pero cualquiera puede hacer lo mismo. Por tanto, este esquema sacrifica **eficiencia, confidencialidad y correcta semántica** de firma digital.

0.3 Conclusión

Ambos mecanismos fallan en reproducir las propiedades esenciales de las firmas digitales: el primero elimina el no repudio y la verificación independiente; el segundo confunde el cifrado con la firma digital, comprometiendo la eficiencia y confidencialidad. Solo el protocolo estándar basado en hash y firma asimétrica garantiza autenticidad, integridad y verificabilidad pública.

Pregunta 2: Vulnerabilidad ante Vectores de Ataque

Para esta pregunta, se plantea el siguiente escenario:

> Oscar actúa como un adversario que intenta comprometer la integridad y autenticidad de las comunicaciones entre Alice y Bob.

A continuación se analiza el funcionamiento, los posibles ataques, las vulnerabilidades explotadas y las soluciones criptográficas aplicables en cada caso.

0.4 Escenario 1: Modificación del mensaje en tránsito

Alice envía a Bob el mensaje:

$$M = \text{“Transferir \$1000 a Mark”}$$

junto con su correspondiente firma digital:

$$auth(M)$$

Oscar intercepta la transmisión y modifica el contenido del mensaje, cambiando el destinatario a:

$$M' = \text{“Transferir \$1000 a Oscar”}$$

Análisis del ataque

Oscar está ejecutando un **message tampering** (un ataque de manipulación de mensaje). Aunque no posee la clave privada de Alice, intenta modificar el mensaje durante la transmisión, esperando que Bob no detecte el cambio.

Pero al recibir el mensaje modificado, Bob calculará el valor hash del nuevo mensaje $H(M')$ y lo comparará con el valor descifrado de la firma $D_{PK_{Alice}}(auth(M))$, que corresponde a $H(M)$.

Y entonces, como $H(M) \neq H(M')$, se espera que la verificación de la firma falle inmediatamente.

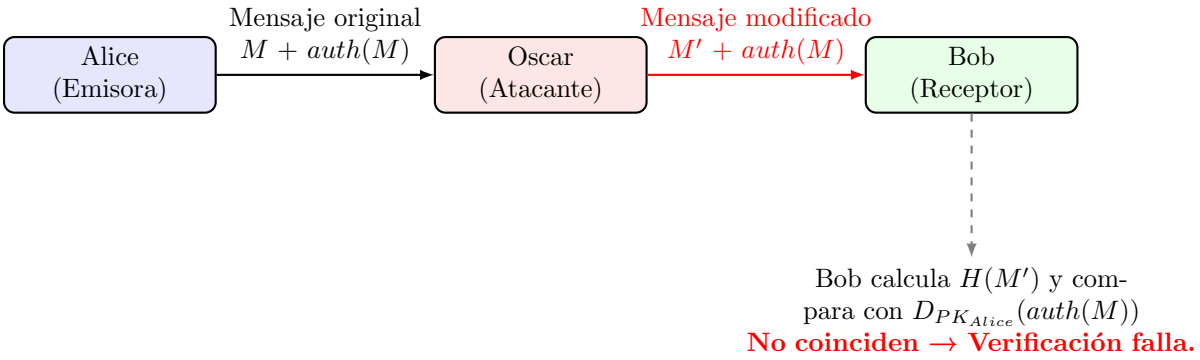

Figure 2: Ilustración sobre como Bob se daría cuenta que el mensaje ha sido alterado.

Vulnerabilidad explotada

En este escenario, Oscar intenta explotar la **ausencia de verificación de integridad**. Si, por ejemplo, Bob no validara la firma digital o el protocolo careciera de una función hash segura, el ataque sería un éxito.

Solución criptográfica

La protección ante este tipo de ataque está garantizada por las propiedades inherentes a una **firma digital estándar**, que incluyen:

- **Integridad:** cualquier modificación del mensaje cambia el hash.
- **Autenticidad:** solo Alice puede generar una firma válida.

Resultado: Bob detectará la alteración, ya que la firma digital no coincidirá con el mensaje modificado.

0.5 Escenario 2: Ataque de repetición

Oscar observa una transacción legítima enviada por Alice:

$$M = \text{“Transferir \$1000 a Mark”}, \quad auth(M)$$

y decide interceptarla y retransmitirla múltiples veces (por ejemplo, 100 veces) hacia Bob.

Análisis del ataque

Para este caso, Oscar no altera el contenido del mensaje ni la firma digital, sino que realiza un **replay attack**. El objetivo es engañar a Bob para que procese varias veces la misma instrucción de transferencia, generando así efectos económicos o lógicos repetidos.

Aunque la firma digital garantiza autenticidad e integridad, nunca se menciona protección contra la **reutilización de mensajes válidos**.

Por lo que Bob, al recibir el mensaje idéntico con una firma válida, no tendría manera de distinguir si se trata de una retransmisión fraudulenta o de una comunicación legítima repetida.

Vulnerabilidad explotada

Oscar se aprovecha de la **falta de mecanismos de frescura temporal o unicidad** en el protocolo, pues la firma digital por sí sola no incorpora información sobre cuándo o cuántas veces debe ser válida una transacción.

Soluciones criptográficas

- **Uso de números de identificadores únicos:** Cada mensaje debería incluir un valor único o incremental que el receptor verifique para evitar repeticiones.
- **Incorporación de marcas de tiempo:** Los mensajes se consideran válidos solo dentro de un intervalo temporal permitido.

Combinando estos mecanismos con firmas digitales, el sistema podría aspirar a garantizar no solo integridad y autenticación, sino también **protección frente a la reutilización maliciosa de mensajes**.

Resultado: Bob no detectará la repetición si no hay control de unicidad. Se requiere incluir nonces o timestamps en el protocolo para evitar este tipo de ataque.

Pregunta 3: Investigación — TLS 1.3 (Transport Layer Security)

0.6 Evolución y Arquitectura del Protocolo TLS

El protocolo **TLS (Transport Layer Security)** es la evolución del antiguo **SSL (Secure Socket Layer)** desarrollado originalmente por Netscape.

Su función principal es proporcionar **confidencialidad, integridad y autenticación** entre dos aplicaciones que se comunican a través de la red; en términos de redes, se ubica entre la capa de transporte (TCP) y la capa de aplicación.

De acuerdo con material académico de seguridad informática, el SSL/TLS provee:

- **Confidencialidad**, mediante cifrado simétrico.
- **Integridad**, a través de códigos de autenticación de mensaje (MAC o AEAD).
- **Autenticación**, gracias al uso de certificados digitales y firmas asimétricas.

En un inicio, SSL ofrecía estos servicios a través de un conjunto de subprotocolos jerárquicos, como *Handshake, Change Cipher Spec,* y *Alert*. TLS heredó esta estructura, pero la mejoró progresivamente en eficiencia y seguridad.

La versión **1.3 de TLS** representa una evolución significativa respecto a TLS 1.2:

- Reduce el número de rondas en el *Handshake*, logrando un establecimiento más rápido (1-RTT o incluso 0-RTT).
- Elimina algoritmos inseguros como RC4, MD5 y SHA-1.
- Introduce **Forward Secrecy**, asegurando que la exposición de una clave privada no comprometa sesiones pasadas.
- Simplifica la negociación de parámetros criptográficos, reduciendo la superficie de ataque.

TLS, al operar sobre TCP, se convierte en la base de servicios seguros como **HTTPS (HTTP over TLS)**, y también de otros protocolos como **SMTP**, **IMAP** y **FTPS**.

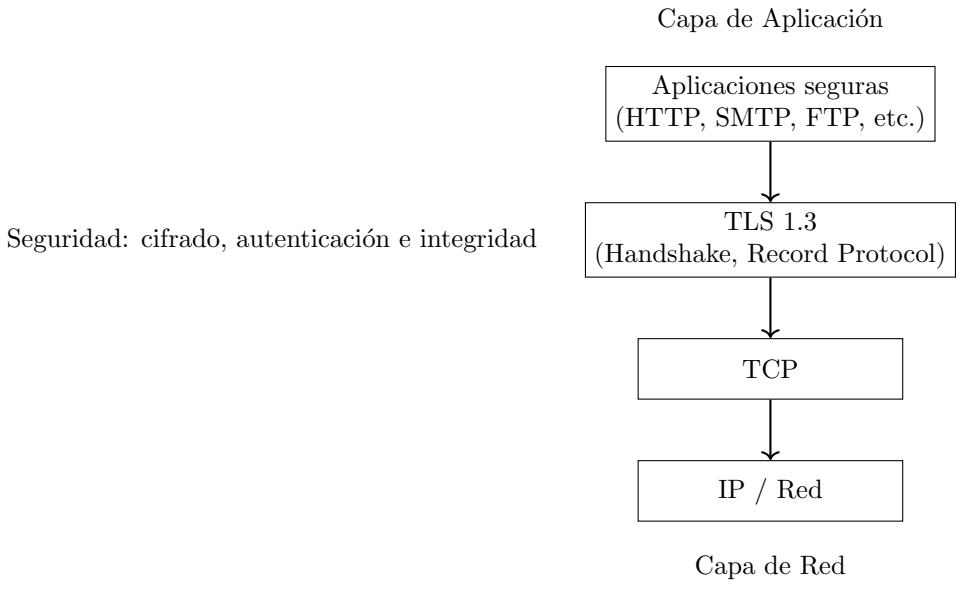

Figure 3: Arquitectura conceptual de TLS: capa de seguridad entre TCP y aplicaciones.

0.7 Handshake de TLS 1.3

El proceso de *Handshake* es el encargado de negociar los parámetros criptográficos y establecer las claves compartidas entre el cliente y el servidor. En TLS 1.3, esto se logra esto en una sola ida y vuelta, lo que mejora la latencia y el rendimiento.

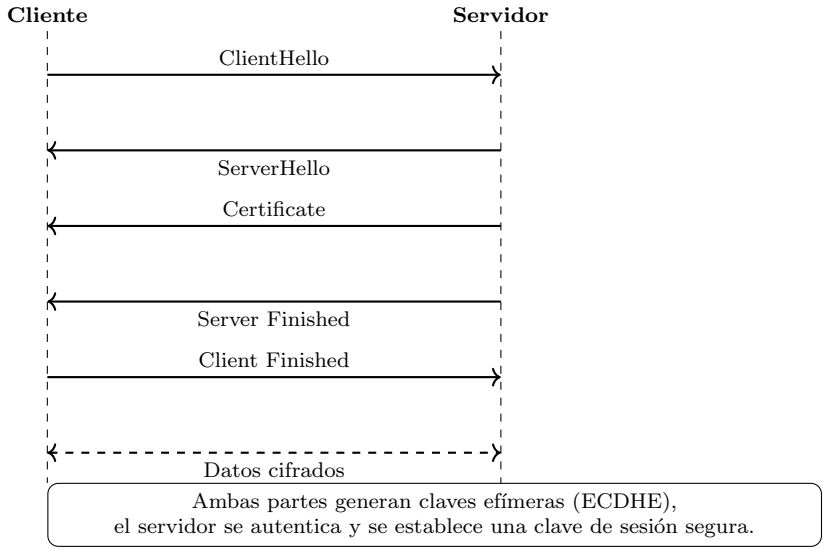

Figure 4: Handshake simplificado del protocolo TLS 1.3.

0.8 Primitivas Criptográficas Utilizadas

TLS 1.3 emplea primitivas modernas y seguras, eliminando mecanismos obsoletos. Entre las principales se encuentran:

- **Cifrado simétrico:** AES-GCM y ChaCha20-Poly1305 para proporcionar confidencialidad e integridad.
- **Intercambio de claves:** ECDHE (Elliptic Curve Diffie-Hellman Ephemeral), garantizando secreto perfecto hacia adelante.
- **Firmas digitales:** Ed25519, ECDSA o RSA-PSS, utilizadas para autenticar entidades.
- **Funciones hash:** SHA-256 y SHA-384, usadas en el cálculo de claves y autenticación.

0.9 Cumplimiento de los Objetivos de Seguridad

Confidencialidad

Los datos se cifran con una clave simétrica derivada del intercambio ECDHE, impidiendo que terceros interpreten la información incluso si interceptan el tráfico.

$$C = E_{K_{sym}}(M)$$

Integridad

TLS 1.3 emplea modos AEAD (Authenticated Encryption with Associated Data), que proporcionan cifrado y verificación de integridad en una sola operación.

$$Tag = MAC(K_{auth}, M)$$

Autenticación

Durante el *Handshake*, el servidor presenta su certificado digital y prueba su autenticidad firmando datos con su clave privada. El cliente puede opcionalmente autenticarse de la misma forma.

0.10 Aplicaciones Modernas de TLS

TLS 1.3 es hoy un pilar fundamental en la seguridad de Internet y sistemas corporativos. Algunos ejemplos concretos incluyen:

- **HTTPS (Navegación Web Segura):** Sitios como Google, Amazon o Wikipedia usan TLS 1.3 para cifrar las conexiones, evitando espionaje o manipulación de datos.
- **Correo Electrónico Seguro:** Servidores como Gmail y Outlook emplean SMTPS, POP3S e IMAPS para proteger credenciales y contenido de mensajes.
- **Servicios en la Nube:** Plataformas como AWS, Azure o Google Cloud aseguran sus APIs y paneles administrativos con TLS 1.3.
- **Aplicaciones de Mensajería y Videollamadas:** WhatsApp Web, Zoom y Microsoft Teams cifran sus canales de control y autenticación mediante TLS.
- **VPNs Empresariales:** Soluciones como OpenVPN o WireGuard utilizan TLS o sus derivados para establecer túneles cifrados entre sedes o empleados remotos.

En conjunto, TLS 1.3 no solo protege la confidencialidad, sino también la confianza en la infraestructura digital moderna.

Pregunta 4: Diseño de un Sistema con No Repudio Verificable

0.11 Sistema de Firma Electrónica Segura

En el enunciado se planeta el siguiente escenario:

> Se requiere diseñar una arquitectura para un sistema de **firma electrónica de contratos**, denominado **SECS (Secure Electronic Contract Signing System)**, en el cual dos partes: **Alice** (proveedora del servicio) y **Bob** (cliente), deben firmar digitalmente un contrato electrónico con plena validez jurídica.
>
> El sistema debe garantizar:
>
> - **No Repudio de Origen:** Ninguna de las partes puede negar haber firmado el contrato.
> - **No Repudio de Recepción:** Ninguna de las partes puede negar haber recibido el documento firmado.
>
> Ambas propiedades son esenciales para asegurar la **validez legal, autenticidad y trazabilidad** del proceso de firma electrónica.

0.12 Requisitos Criptográficos Fundamentales

Este sistema basa su lógica y funcionamiento en los siguientes mecanismos criptográficos:

- **Infraestructura de Clave Pública:** Cada usuario tiene un par de claves (PK, SK) certificado por una Autoridad.
- **Firmas Digitales:** Se emplea un algoritmo robusto como **RSA-PSS**.
- **Hash Seguro:** Se utiliza **SHA-256** para generar resúmenes únicos del contrato.
- **Sello de Tiempo:** Emitido por una Autoridad para asociar la firma con un instante verificable.
- **Registros Inmutables:** Almacenamiento en una base de datosque preserve el historial de firmas y recepciones.

0.13 Arquitectura Propuesta del SECS

El sistema SECS está compuesto por los siguientes módulos:

1. **Módulo de Autenticación:** Verifica la identidad de los participantes mediante certificados digitales.
2. **Módulo de Firma:** Genera y aplica la firma digital de cada parte sobre el hash del contrato.
3. **Módulo de Notificación:** Registra y certifica la entrega del documento a la contraparte.
4. **Autoridad de Registro:** Almacena los registros de transacciones firmadas.
5. **Autoridad de Tiempo:** Garantiza la validez temporal de las firmas.

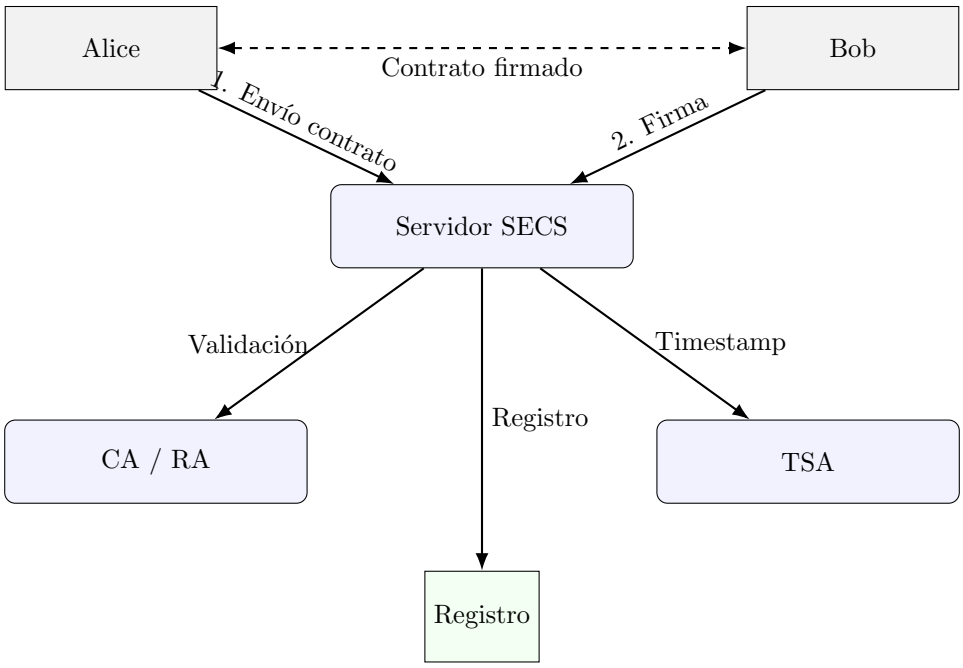

Figure 5: Arquitectura simplificada del sistema SECS para firma electrónica con no repudio.

0.14 Flujo del Protocolo SECS

1. **Inicialización:** Alice prepara el contrato M y lo envía al servidor SECS para iniciar el proceso.
2. **Firma de Alice:**
$$Sig_A = E_{SK_A}(H(M))$$
El servidor SECS solicita un sello temporal a la TSA y almacena el par (M, Sig_A, T_A).
3. **Entrega a Bob:** El contrato firmado por Alice se envía a Bob, quedando registrada la entrega.
4. **Firma de Bob:** Bob verifica la firma de Alice, añade la suya,
$$Sig_B = E_{SK_B}(H(M))$$
y envía de vuelta (M, Sig_A, Sig_B) al servidor.
5. **Confirmación Final:** El servidor SECS aplica un segundo sello temporal T_B y guarda todas las evidencias en el registro inmutable.

0.15 Cumplimiento del No Repudio

- **No Repudio de Origen:** La firma de Alice, validada con su clave pública certificada, constituye prueba verificable del origen del documento.
- **No Repudio de Recepción:** La firma de Bob, junto al sello temporal y los registros del SECS, prueban que el documento fue recibido y aceptado.

Ambas pruebas pueden verificarse de forma independiente por un tercero (una autoridad legal, por ejemplo), sin intervención de los firmantes.

0.16 Consideraciones Legales y de Seguridad

El sistema SECS cumple con principios de las normativas internacionales como:

- **Integridad de los registros:** El uso de sellos de tiempo y almacenamiento inmutable garantiza la trazabilidad y validez de las pruebas.
- **Auditoría:** Cualquier intento de manipular registros firmados o sellados invalidará la verificación criptográfica.

La arquitectura SECS combina mecanismos criptográficos con un diseño de protocolo robusto para lograr:

- Autenticación verificable de cada firmante.
- Integridad total del contrato firmado.
- Evidencia legal inmutable de envío y recepción.

De esta forma, se ofrece una solución **técnica y jurídicamente sólida** para la firma electrónica de contratos críticos, garantizando no repudio, transparencia y validez legal.

Bibliografía

- Chirag Bhalodia. *Digital Signature Algorithm (DSA) in Network Security.* Disponible en: `https://www.youtube.com/watch?v=eiNZi-G-CE4`
- Cloudflare. *Why Use TLS 1.3?*. Disponible en: `https://www.cloudflare.com/es-es/learning/ssl/why-use-tls-1.3/`
- Stallings, W. *Cryptography and Network Security*, 5th Edition. Prentice Hall.