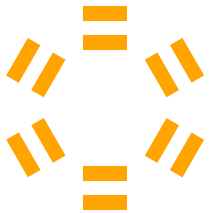
Autenticación y privacidad en grupos de mensajería: ¿por qué siempre preguntan “¿Quién eres?” y nunca “¿Qué tal estás?”

David Soler García

Desayuno TIC Talk

24 de Noviembre de 2026





Contexto

- Popularidad de las aplicaciones de mensajería
  - Comunidades en línea

- Autenticación
  - Identidad digital
  - Autenticación mutua

Identidad digital

- Infraestructura actual de identidad
- Basada en Clave Pública
- **Certificados digitales**

- **Identidad digital** / identidad real

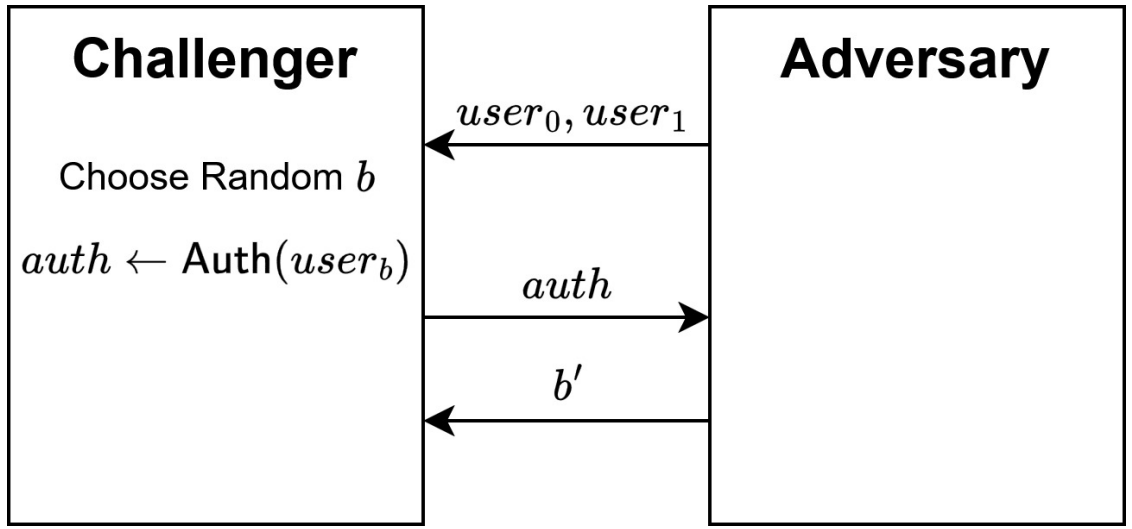
```
Data:
    Version: 3 (0x2)
    Serial Number:
        3a:d7:ec:c6:da:63:f2:8e:63:45:19:cb:ec:23:fa:d1
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=ES, O=FNMT-RCM, OU=Ceres, CN=AC FNMT Usuarios
    Validity
        Not Before: Oct 11 07:22:51 2022 GMT
        Not After : Oct 11 07:22:51 2026 GMT
    Subject: GN=DAVID, SN=SOLER GARCIA, CN=SOLER GARCIA DAVID - 39487084V
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            [...]
    X509v3 extensions:
        X509v3 Subject Alternative Name:
            email:solergarciadavid@gmail.com
        X509v3 Key Usage: critical
            Digital Signature, Non Repudiation, Key Encipherment
Signature:
    [...]
```

Privacidad en el mundo real

- Preocupaciones de los usuarios
  - Impacto en mundo real
  - Seguimiento (***Linkability***)
- Privacidad y moderación
  - Identidades no enlazables
  - Control de acceso, ***Blocklisting***
  - Legislación

Formalización de la Seguridad: Juegos

- *Challenger*
  - Configura el juego
  - Responde a *queries*
- *Adversary*
  - Persigue alcanzar un objetivo
- El juego debe reflejar las condiciones del mundo real
  - Capacidades del adversario



Formalización de la Seguridad: Propiedades

- ***Unforgeability***: no se puede autenticar sin credenciales válidas

- ***Anonymity***: no se puede obtener información sobre el usuario
  - Si se pueden observar intentos de autenticación: ***Unlinkability***

Formalización de la Seguridad: Protocolo

- Protocolo criptográfico
- Composición de primitivas
- **Prueba de seguridad**
  - Si primitivas son seguras, entonces protocolo es seguro
  - **Prueba por reducción**

function INITAS(1^λ)
$acc \leftarrow \mathsf{AC.Create}(1^\lambda)$
$(pk, vk) \leftarrow \mathsf{ZK.Setup}(1^\lambda, R)$
$bb \leftarrow (acc, \emptyset, \emptyset, \emptyset, (pk, vk))$
$ssk \leftarrow \mathsf{PKI.AddId}(\text{"AS"})$
return (ssk, bb)
end function

function INITKAL($1^\lambda, id_{KAL}$)
$ssk \leftarrow \mathsf{PKI.AddId}(id_{KAL})$
$st_{KAL} \leftarrow (id_{KAL}, ssk)$
return st_{KAL}
end function

function CREATESAE($priv, id_{KAL}$)
$(u, _, _) \leftarrow priv$
$s \leftarrow \mathcal{X}$
$id_{sae} \leftarrow \mathrm{PRF}_u(s)$
$sae \leftarrow (id_{sae}, id_{KAL}, s)$
end function

function RETRIEVEKEY($st_{KAL}, sae_r, \pi, kid, bb$)
$(k, \vec{a^r}) \leftarrow \mathsf{KMS.KeyRetrieval}(sae_r, kid)$
`assert` $verify(sae_e, \pi, \vec{a^r}, bb)$
$(id_{KAL}, ssk) \leftarrow st_{KAL}$
$(_, C_f, N) \leftarrow \pi$
$bb' \leftarrow updateBB(id_{KAL}, ssk, C_f, N, bb)$
return (k, bb')
end function

function AUTH($sae, priv, \vec{a^r}, bb$)
$(acc, _, _, _, (pk, vk)) \leftarrow bb$
$(id_{sae}, id_{KAL}, s) \leftarrow sae$
$(u, r_i, obj_i) \leftarrow priv$
$C_i \leftarrow \mathrm{PRF}_{r_i}(obj_i)$
$w_{C_i} \leftarrow \mathsf{AC.ProveMem}(acc, C_i)$
$N \leftarrow \mathrm{PRF}_u(obj_i)$
$(_, _, obj_f) \leftarrow \phi(priv, \vec{a^r}, bb)$
$r_f \leftarrow \mathcal{X}$
$C_f \leftarrow \mathrm{PRF}_{r_f}(obj_f)$
$N \leftarrow \mathrm{PRF}_u(obj_i)$
$w \leftarrow (u, w_{C_i}, r_i, r_f, obj_i, obj_f)$
$x \leftarrow (\vec{a^r}, bb, C_f, N, id_{sae}, s)$
$\pi^{auth} \leftarrow \mathsf{ZK.Prove}(pk, x, w)$
$\pi \leftarrow (\pi^{auth}, C_f, N)$
$priv' \leftarrow (u, r_f, obj_f)$
return $(\pi, priv')$
end function

function REQUESTKEY($st_{KAL}, sae_e, sae_r, \pi, \vec{a^r}, bb$)
`assert` $verify(sae_e, \pi, \vec{a^r}, bb)$
$(id_{KAL}, ssk) \leftarrow st_{KAL}$
$(_, C_f, N) \leftarrow \pi$
$bb' \leftarrow updateBB(id_{KAL}, ssk, C_f, N, bb)$
$(k, kid) \leftarrow \mathsf{KMS.KeyRequest}(sae_e, sae_r, \vec{a^r})$
return (k, kid, bb')
end function

Formalización de la Seguridad: Ejemplos

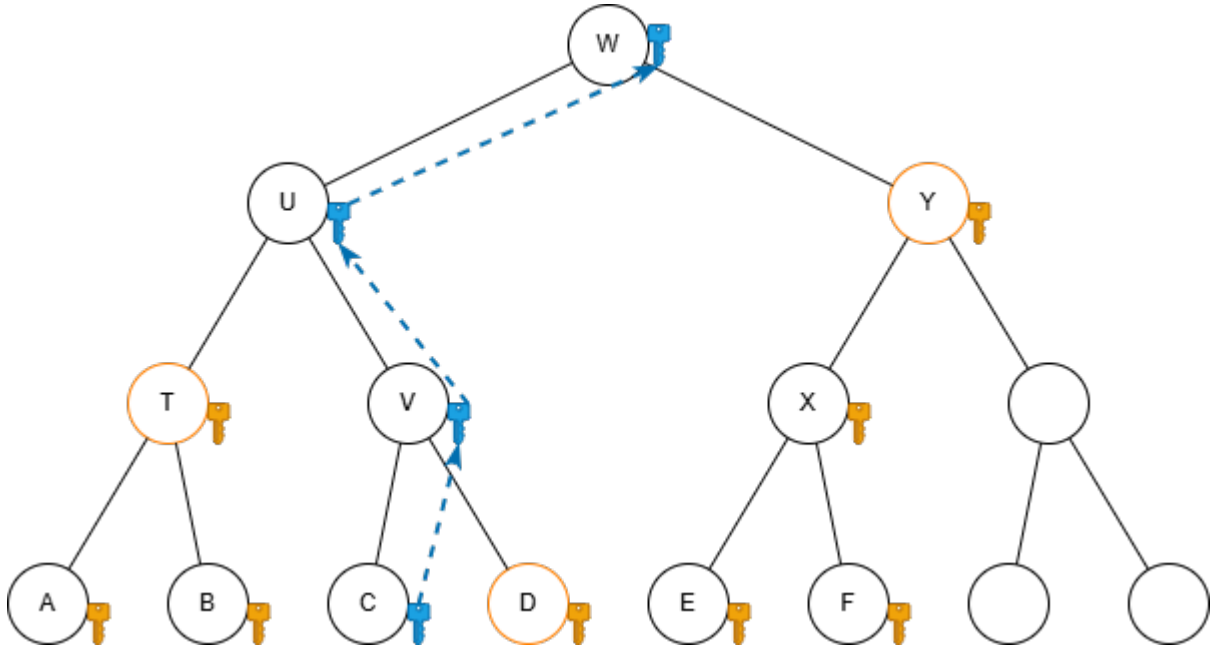
- *Unforgeability*
  - Algoritmo de firma: Clave pública / clave privada
  - Imposible falsificar firmas
  - Certificado firmado por Entidad de Confianza
- *Anonymity*
  - *Pseudo-Random Function* (PRF):
  - Salida indistinguible de cadena aleatoria
  - ¿$ps = PRF(u_1)$ o $ps = PRF(u_2)$?

Continuous Group Key Agreement (CGKA)

- Secreto compartido entre un grupo dinámico
  - Actualización periódica (inserciones, eliminaciones)
  - *Forward Secrecy* (FS), *Post-Compromise Security* (PCS)
- Un Commit aplica múltiples Proposals
- Aplicación a la mensajería grupal

Messaging Layer Security (MLS)

- Utiliza **TreeKEM** como protocolo CGKA
  - El grupo se estructura como un *Ratchet Tree*
  - Par de claves por cada nodo
  - Claves privadas conocidas solo por descendientes
- **Eficiencia logarítmica** en las actualizaciones
- Escala con el tamaño del grupo



Contribuciones

1. Autenticación con atributos en CGKA
2. *Federated Anonymous Blocklisting*

Protocolo 1: CGKA con Autenticación con Atributos

- Autenticación actual en CGKA
  - Basada en identidad: el usuario revela toda su información
  - Claves de firma de larga duración: Linkability
- Alternativa propuesta
  - Basada en atributos: el usuario revela solo los atributos necesarios
  - One-time signature keys: Unlinkability
- Contribuciones
  - CGKA con Autenticación con Atributos (AA-CGKA)
  - Construcción e implementación de AA-CGKA

Herramientas: Attribute-Based Credentials

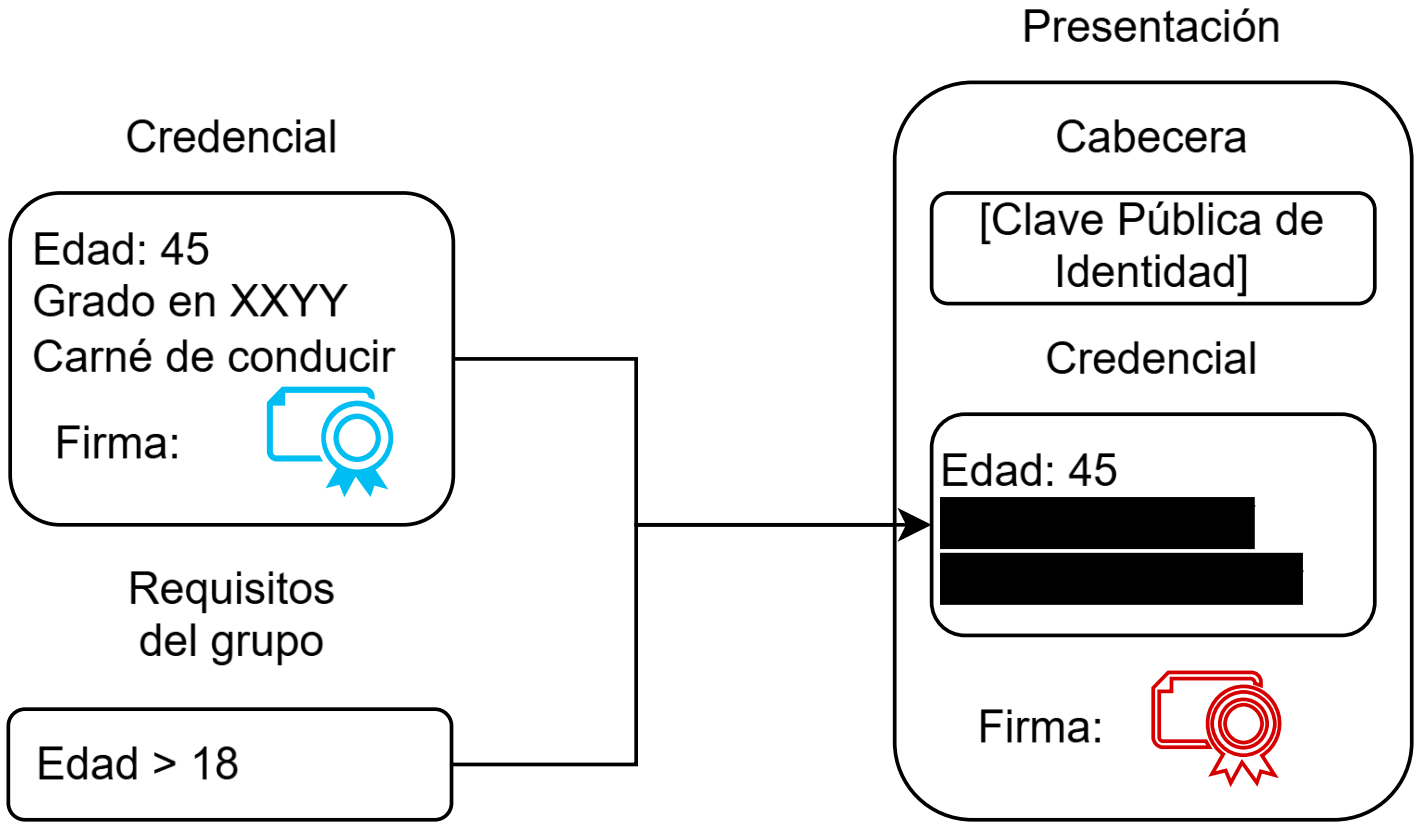
- Contienen un conjunto de *claims* sobre su titular
- Presentaciones
  - Demuestran que el ABC cumple ciertos requisitos
  - Divulgan selectivamente algunos de los *claims*
  - No enlazables con el ABC


```
{
  "type": [
    "VerifiableCredential",
    "ExampleAlumniCredential"
  ],
  "issuer": "did:example:issuer",
  "exp": 1883000000,
  "credentialSubject": {
    "name": "did:example:subject",
    "degree": {
      "university": "Example University",
      "name": "Degree in Computer Science"
    }
  }
}
```

Resumen de AA-CGKA

- Los grupos poseen un conjunto de requisitos
  - Incluidos en GroupInfo
  - Modificados mediante Proposals
- Cambios en autenticación
  - Key Packages -> Presentation Packages
  - La Presentation demuestra que se cumplen los requisitos
  - Claves de firma *frescas* para cada autenticación

Resumen de AA-CGKA

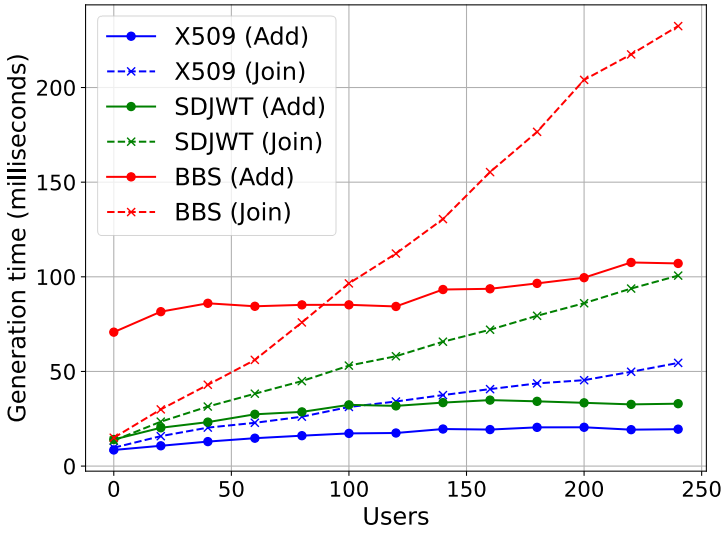


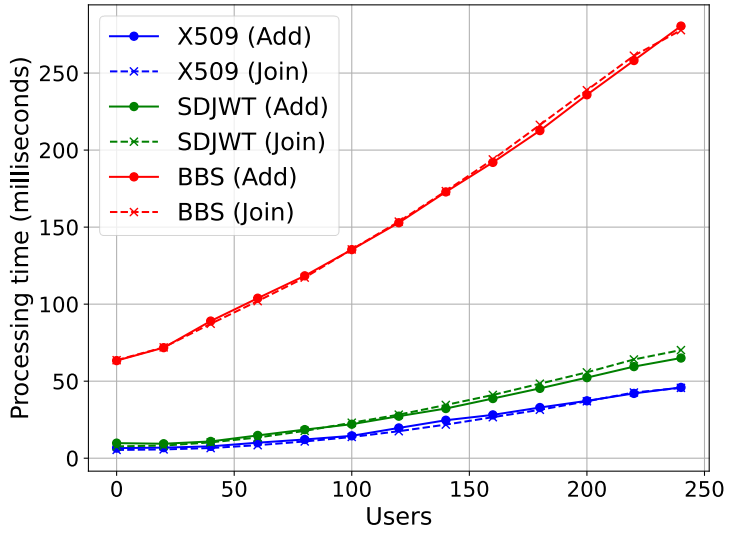
Seguridad

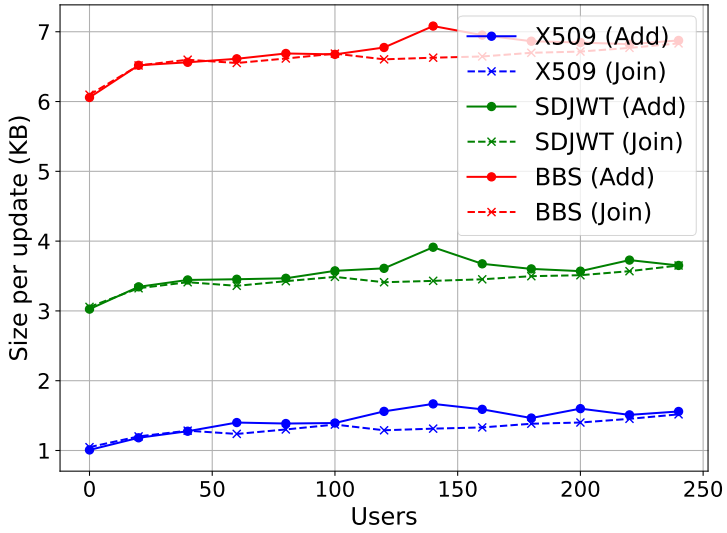
- ***Unforgeability*** sin una credencial que cumpla los requisitos
- ***Unlinkability*** entre Presentaciones
- ***Requirement Integrity:*** solo los usuarios actuales pueden modificar los requisitos

Implementación (Rust)

- Basado en el testbed desarrollado en el capítulo anterior
- Uso de MLS Extensions: Proposals, Credentials y Group Context
- Dos esquemas ABC: BBS y SD-JWT





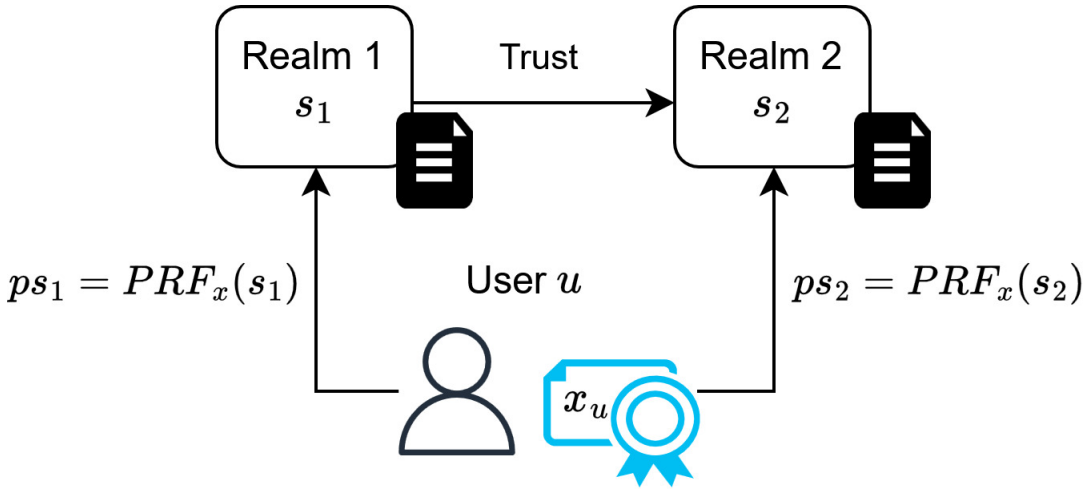


Protocolo 2: Federated Anonymous Blocklisting

- Uso de pseudónimos para aumentar el anonimato
  - ¿Cómo aplicar bloqueos si los usuarios pueden crear nuevos pseudónimos?
- Relacionado: ***Anonymous Blocklisting***
  - Todos los pseudónimos de usuario se derivan del secreto x
  - Los usuarios deben demostrar que no están bloqueados
  - Ineficiente: los usuarios deben procesar cada nuevo bloqueo (sincronización)
- Contribuciones
  - Esquema Federated Anonymous Blocklisting (FAB)
  - Construction e implementación de FAB

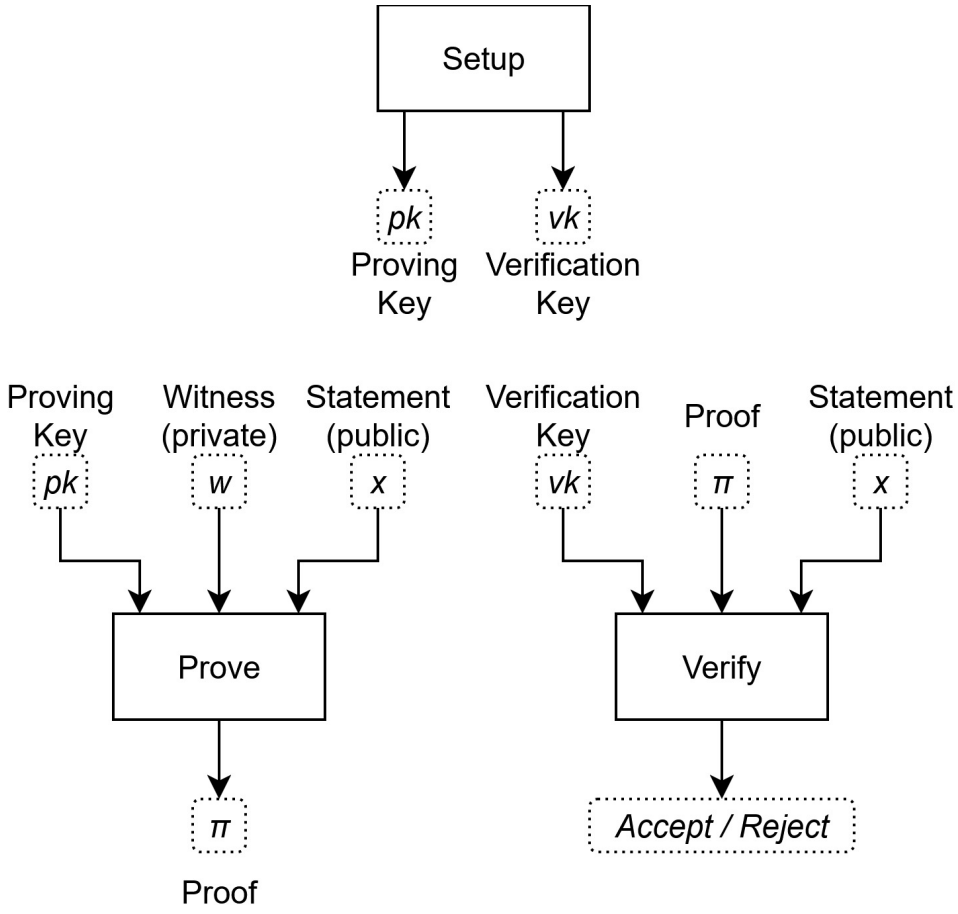
Resumen de FAB

- Grupos
  - Semilla aleatoria s
  - Lista de bloqueos
  - Confianza entre grupos
- Usuarios
  - Certificado de usuario con identidad x
  - Pseudónimo para el grupo: $PRF_x(s)$
- Autenticación: demostrar que no está bloqueado
  - En el grupo objetivo
  - En sus grupos de confianza



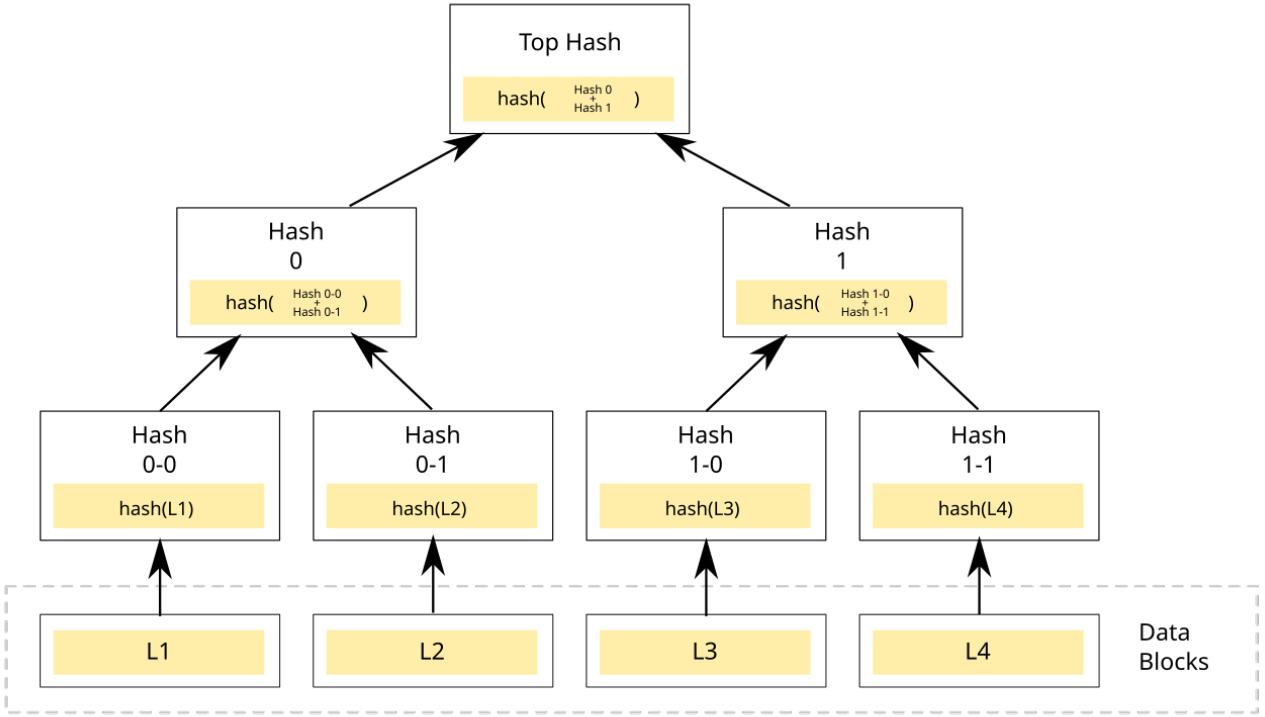
Primitivas criptográficas: ZK-SNARK

- Demostrar **conocimiento** de un valor privado w
  - Private Witness w, Public Statement x, Proof π
  - Relación R entre (x,w)
  - Ejemplo: $x = H(w)$
  - **Zero-Knowledge**: no revela información sobre w
- Construcción
  - Transformar R en un circuito aritmético
  - *Prove* costoso y *Verify* barato



Primitivas criptográficas: Acumuladores

- Conjunto de elementos
  - Añadir, eliminar
- Pruebas de (no) pertenencia
  - Verificación eficiente
  - Combinados con zk-SNARK: **Pruebas de Conocimiento Cero de (no) pertenencia**
- Construcción: ***Árbol Merkle***



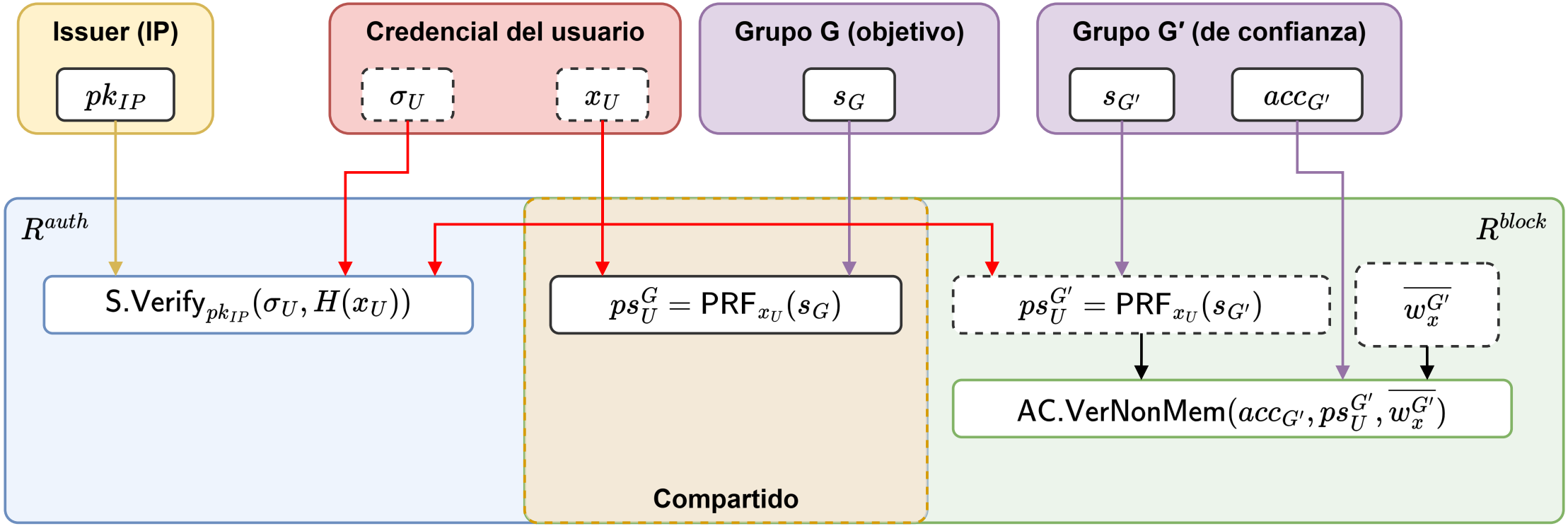
Construcción

- Lista de bloqueos: Acumulador negativo
- 2 tipos de pruebas zk-SNARK
  - R^{auth}: El pseudónimo para este grupo está correctamente calculado
  - R^{block}: El pseudónimo para el grupo de confianza no está en lista de bloqueos
- N grupos de confianza -> 1 prueba R^{auth} + N pruebas R^{block}

$$R^{auth} = \left\{ ((pk_{IP}, s_G, ps_U^G), (\sigma_U, x_U)) : \begin{array}{l} \mathsf{S.Verify}_{pk_{IP}}(\sigma_U, \mathrm{H}(x_U)), \\ ps_U^G = \mathrm{PRF}_{x_U}(s_G) \end{array} \right\}$$

$$R^{block} = \left\{ ((ps_U^G, s_G, s_{G'}, acc_{G'}), (\overline{w_x^{G'}}, x_U)) : \begin{array}{l} ps_U^G = \mathrm{PRF}_{x_U}(s_G) \\ ps_U^{G'} = \mathrm{PRF}_{x_U}(s_{G'}) \\ \mathsf{AC.VerNonMem}(acc_{G'}, ps_U^{G'}, \overline{w_x^{G'}}) \end{array} \right\}$$

Construcción

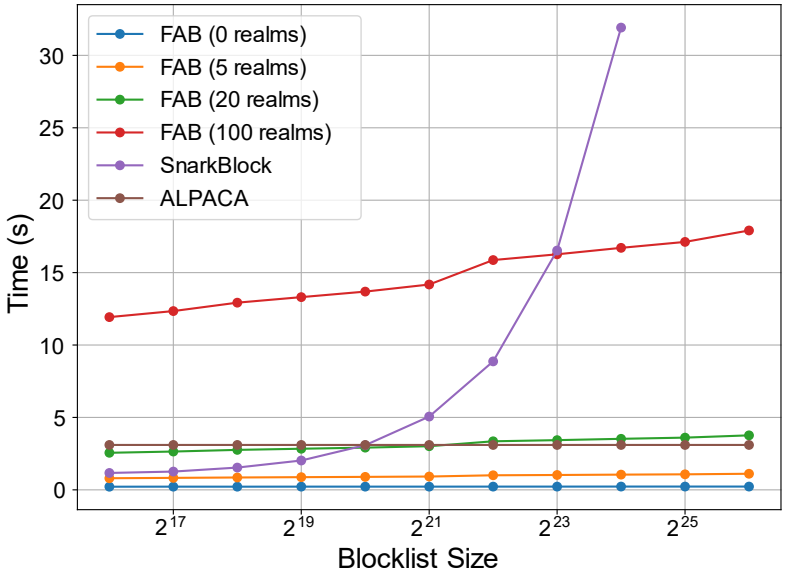


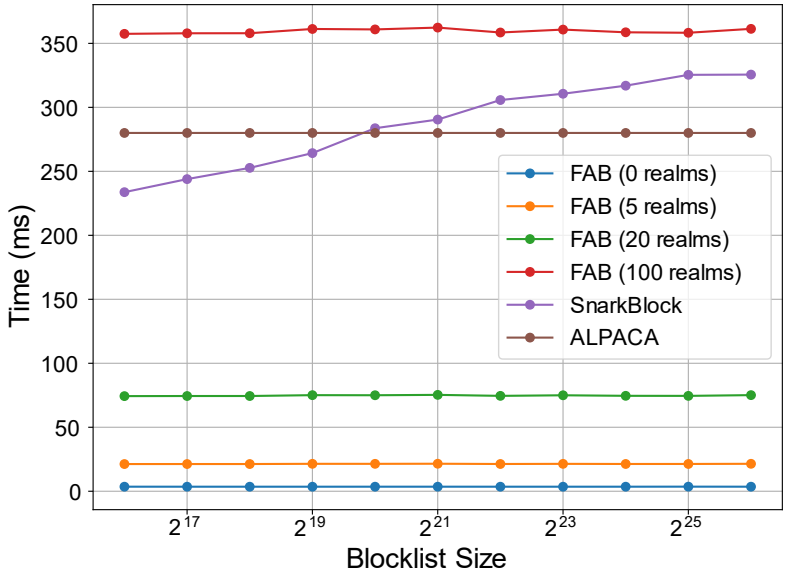
Seguridad

- ***Blocklistability*** impide la autenticación de usuarios bloqueados
- ***Unlinkability*** entre pseudónimos
- ***Unframeability:*** imposible bloquear maliciosamente a un usuario

Implementación (Rust)

- Comparación con el State-of-the-art: SNARKBlock y ALPACA
- Evita la operación más costosa: coste de sincronización
- Introduce un nuevo coste de escalado lineal: grupos de confianza





Conclusión

- Aplicaciones de Autenticación con privacidad reforzada a la mensajería en grupo
- Viabilidad de los esquemas propuestos
- Trabajo futuro
  - Aplicaciones de construcciones modernas de zk-SNARK
  - Estados dinámicos

Autenticación y privacidad en grupos de mensajería: ¿por qué siempre preguntan “¿Quién eres?” y nunca “¿Qué tal estás?”

David Soler García

Desayuno TIC Talk

24 de Noviembre de 2026

