Privacidad en la distribución clásica de claves cuánticas mediante zk-SNARKs

David Soler, Carlos Dafonte y Francisco Nóvoa
CITIC, Universidade da Coruña, A Coruña
Email: {david.soler,carlos.dafonte,fjnovoa}@udc.es

M. Fernández-Veiga y A. Fernández Vilas
atlanTTic, Universidade de Vigo, Vigo, Spain
Email: {mveiga,avilas}@det.uvigo.es

***Resumen*—Los números aleatorios de alta entropía son una parte esencial de la criptografía, y los generadores cuánticos de números aleatorios (QRNG) son una tecnología emergente que puede proporcionar claves de alta calidad para algoritmos criptográficos, pero lamentablemente son de difícil acceso en la actualidad. Las soluciones existentes de *Entropía como Servicio* requieren que los usuarios confíen en la autoridad central que distribuye el material de claves, lo que no es permisible en un entorno de alta privacidad. En este trabajo, presentamos un novedoso protocolo de transmisión de claves que permite a los usuarios obtener material criptográfico generado por un QRNG de forma que el servidor no pueda identificar qué usuario recibe cada clave. Esto se consigue con la inclusión de zk-SNARKs, una primitiva criptográfica que permite a los usuarios demostrar el conocimiento de algún valor sin necesidad de revelarlo. También proporcionamos una implementación del protocolo que demuestra su funcionalidad y rendimiento, utilizando NFC como canal de transmisión de la clave QRNG.**

***Index Terms*—QRNG, transmisión de clave, zk-SNARK, protoco de comunicación, Autenticación privada, NFC**

Tipo de contribución: *Investigación en desarrollo*

I. Introducción

Los números aleatorios siempre han sido un elemento fundamental en la criptografía. Todos los algoritmos criptográficos, desde los de cifrado simétrico como AES hasta los asimétricos como RSA, requieren números aleatorios como material de clave o parámetros de inicialización. Las fuentes más comunes de aleatoriedad para fines criptográficos son los generadores de números pseudoaleatorios (PRNG), que son procesos deterministas que se inicializan a partir de una semilla específica. Además, los PRNG pueden contener vulnerabilidades en sus implementaciones [1], [2] que los hacen vulnerables a ataques. Por ello, los PRNG suelen producir material clave de baja entropía, lo que puede comprometer todo el esquema criptográfico en el que se utilizan: no importa lo teóricamente seguro que sea un algoritmo de cifrado, podría descifrarse fácilmente si un atacante consigue obtener la clave secreta aprovechando su generación insegura. No se trata sólo de un problema teórico: la generación de claves de baja entropía se ha utilizado como punto de entrada para romper la confidencialidad en sistemas criptográficos de uso habitual [3], [4], [5].

Los Generadores Cuánticos de Números Aleatorios (QRNG) [6], [7], [8] son un tipo diferente de Generadores de Números Aleatorios que utilizan el comportamiento intrínsecamente impredecible de las partículas cuánticas como fuente de entropía. Por ello, el material clave que generan es de mayor calidad que el generado por un PRNG, y su uso podría mejorar la seguridad de los algoritmos criptográficos actuales. Por desgracia, la tecnología aún no está totalmente desarrollada y los QRNG tienen un coste elevado, por lo que no es viable distribuirlos de forma masiva y los usuarios finales tienen un acceso muy limitado a ellos. En contextos que requieran un alto nivel de privacidad, como las comunicaciones privadas, los usuarios se beneficiarían de tener acceso a claves criptográficas de alta entropía, de tal forma que sea más difícil para los atacantes adivinar la clave que se ha utilizado. Además, facilitar el acceso a este material de claves podría animar a los usuarios a tomarse en serio su privacidad y mejorar la seguridad general de sus comunicaciones.

El objetivo de este trabajo es proporcionar un método para que los usuarios finales obtengan material criptográfico de mayor calidad mediante la definición de un protocolo de transmisión de claves en el que un servidor con acceso al QRNG distribuye claves aleatorias de forma segura a los usuarios que las solicitan. Como parte del proceso, el servidor exigirá a los clientes que se autentiquen antes de obtener el material criptográfico generado por el QRNG.

Sin embargo, las soluciones existentes de *Entropía como Servicio* (*EaaS*) [9] requieren que los usuarios confíen en su servidor de distribución de claves, ya que este servidor puede rastrear qué claves ha proporcionado a cada usuario. Un servidor malicioso o comprometido podría utilizar las claves que ha transmitido para descifrar las comunicaciones de los usuarios o incluso suplantar su identidad. De hecho, el requisito de confiar en el servidor es una preocupación común en los protocolos de distribución de claves. Este trabajo pretende dar solución a ese problema, diseñando un protocolo de transmisión de claves que busca un equilibrio entre exigir autenticación (es decir, solicitar información privada a los usuarios de forma que el acceso quede restringido a aquellos que puedan demostrar su identidad) y proteger la privacidad de los usuarios. Este requisito se consigue con la inclusión de *Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge* (zk-SNARK) como pruebas de autenticación, una primitiva criptográfica que permite a los usuarios demostrar que conocen algún valor sin necesidad de revelarlo, de forma que cualquier verificador podría convencerse de ello.

Nuestro protocolo se divide en dos fases principales: en la primera, un usuario realiza la autenticación y, en la segunda, el usuario solicita la clave QRNG presentando una prueba zk-SNARK. La inclusión de zk-SNARKs nos permite desacoplar claramente las dos fases, de forma que la presentación de la prueba zk-SNARK en la segunda fase no revela ninguna información (ni siquiera al servidor) sobre las credenciales utilizadas en la primera fase. Así, los usuarios pueden demostrar que se han autenticado previamente, de tal forma que el servidor no podría distinguir al usuario de cualquier otro

ISBN:978-84-09-62140-8

que haya realizado la autenticación alguna vez. Esto impide que el servidor de distribución de claves asocie una clave a un usuario concreto, lo que permitiría al servidor descifrar cualquier mensaje que utilice esta clave. Nuestro protocolo también ofrece protección contra los ataques de repetición: si un atacante intenta reutilizar la prueba zk-SNARK de otro usuario, el servidor detectará el intento y abortará la ejecución.

Dado que los números aleatorios son necesarios en todos los algoritmos criptográficos habituales, las claves QRNG distribuidas por el protocolo podrían ser útiles en una amplia gama de aplicaciones. Por ejemplo, en situaciones en las que los usuarios necesiten establecer canales de comunicación con un alto nivel de confidencialidad, podrían emplear estos valores aleatorios como material de clave de alta calidad para algoritmos de cifrado simétrico. Otra aplicación es la autenticación multifactor: la posesión de un número aleatorio generado por este protocolo también demuestra que su propietario ha realizado la autenticación con éxito, por lo que un servidor podría exigir a los usuarios que ejecutaran este protocolo y presentaran el valor generado por el QRNG para proporcionarles acceso.

En resumen, nuestro trabajo presenta las siguientes aportaciones:

1. Un Protocolo de Transmisión de Claves para la distribución de claves QRNG. El protocolo requiere dos interacciones distintas entre clientes y el servidor: una para la autenticación y otra para la obtención de la clave QRNG.
2. Un método de autenticación que preserva la privacidad para el Protocolo de Transmisión de Claves. La autenticación se define de tal forma que es imposible para el servidor rastrear quién recibe cada clave, evitando así que el servidor espíe las comunicaciones de los clientes que utilizan estas claves. Esto permite a los usuarios ejecutar el protocolo aunque no confíen en el servidor que genera las claves. Además, el protocolo está diseñado para evitar que se puedan falsificar pruebas sin haber realizado la autenticación, incluyendo protección contra ataques de repetición. Denominaremos a nuestra propuesta *Protocolo de Transmisión de Claves que Preserva la Privacidad* o, por sus siglas en inglés, *PPKTP*.
3. Una implementación del Protocolo de Transmisión de Claves propuesto. La implementación simula un caso de uso real, en el que el servidor se encuentra en algún lugar específico al que los usuarios tienen que acudir. Parte de la comunicación se realiza a través de NFC y la clave se almacena en el teléfono móvil de los usuarios. Se emplean dos librerías zk-SNARK diferentes para la generación y validación de pruebas, y el servidor tiene la opción de utilizar dos QRNG para la generación de claves.

El resto del documento está organizado de la siguiente manera: La Sección II revisará trabajos relacionados y los comparará con nuestra contribución. La Sección III presentará al lector el fondo técnico requerido. La Sección IV explicará en detalle todos los pasos involucrados en la ejecución del protocolo, así como las entidades que participan en él. La Sección V analizará la implementación de prueba de concepto de todas las entidades involucradas en el protocolo. Finalmente, la Sección VI concluirá este documento y se discutirán futuras mejoras.

II. Trabajo Relacionado

Muchas de las mejoras recientes en fuentes de entropía se han centrado en los *Distributed Random Beacons* (DRB) [15], [16]. En estos esquemas, un grupo de usuarios puede generar un valor aleatorio público que puede ser verificado por todos los participantes y otras partes, asegurando que ningún usuario haya introducido sesgo en el cálculo. Si bien la aleatoriedad pública verificable es útil en múltiples escenarios, todavía existe la necesidad de fuentes de entropía que distribuyan los números aleatorios generados de manera privada.

Existen varios servicios que proporcionan aleatoriedad de alta entropía a pedido, como *random.org* [17]. NIST ha diseñado una arquitectura para la generación y distribución de números aleatorios [9], que requiere que los clientes presenten un token de autenticación. En estos esquemas, el servidor sabe qué cliente está recibiendo cada valor aleatorio, y por lo tanto, cualquier material criptográfico que se genere a través de ellos. Esto representa un problema de privacidad para los clientes, que quizá no confíen en el servidor que proporciona la fuente de entropía.

El objetivo de la autenticación generalmente entra en conflicto con la privacidad de los usuarios: el servidor de autenticación puede requerir más información de la que el cliente está dispuesto a proporcionar. Las Pruebas de Conocimiento Cero son una familia de herramientas criptográficas mediante los cuales un demostrador puede demostrar a un verificador que conoce cierta información privada sin necesidad de revelarla. Su uso en protocolos de autenticación permite a los usuarios demostrar su identidad sin revelar información sensible sobre sí mismos. En variantes no interactivas de Pruebas de Conocimiento Cero, el usuario que se autentica y el verificador no necesitan establecer comunicación directamente: en su lugar, el usuario publica una prueba que puede ser verificada más tarde por cualquier otra entidad.

Los autores de [10] definen un esquema en el que los usuarios pertenecientes a ciertos dominios de confianza pueden autenticarse entre sí y establecer sesiones de comunicación. Se define un método para registrar nuevos usuarios, pero la defensa contra ataques de repetición es débil, ya que utiliza marcas de tiempo: no protege contra repeticiones que ocurran en un corto período de tiempo.

El esquema definido en [11] permite que las instituciones educativas presenten el diploma de un alumno en una cadena de bloques pública. Se utiliza una Prueba de Conocimiento Cero personalizada en múltiples ocasiones, como proteger la identidad del alumno o atestiguar sus calificaciones. El protocolo no proporciona ninguna protección directa contra ataques de repetición, pero repetir una prueba no es suficiente para suplantar a un alumno legítimo: un atacante también necesitaría conocer la clave secreta de ese alumno.

Entre los algoritmos de Pruebas de Conocimiento Cero, los zk-SNARKs son una alternativa popular que también se pueden usar en protocolos de autenticación. Los autores de [14] proponen un protocolo de autenticación para entornos de atención médica en el que un conjunto de clientes prerregistrados se autentican con un zk-SNARK validado con

Scheme	Purpose	Authentication Method	Client Registration	External Infrastructure	Replay protection
[10]	Key Agreement	Discrete Logarithm	Yes	No	Timestamp
[11]	E-learning records	Zero Knowledge Proof	Yes	Yes (Blockchain)	Secret key
[12]	Voting	zk-SNARK	No	Yes (Blockchain)	Yes
[13]	Identity Management	zk-SNARK	Yes	Yes (Blockchain)	No
[14]	Key Agreement	zk-SNARK	No	Yes (Ethereum)	Yes
Ours	Key Transmission	zk-SNARK/Merkle Tree	Yes	No	Yes

Tabla I
COMPARACIÓN ENTRE OTROS TRABAJOS QUE UTILIZAN AUTENTICACIÓN NO INTERACTIVA DE CONOCIMIENTO CERO Y EL PROTOCOLO PROPUESTO.

un *Smart Contracts* con el objetivo de establecer un canal de comunicación seguro. En [13], se inserta una autoridad de certificación en el esquema para validar los parámetros de los usuarios, que luego pueden proporcionar pruebas. Sin embargo, no presenta defensa contra atacantes que podrían interceptar la prueba de otro usuario y luego presentarla como propia. En [12], el enfoque se centra en definir un sistema que puede proporcionar Rastreabilidad y Trazabilidad para identificar usuarios malintencionados, mientras protege la privacidad de los usuarios honestos. Dado que los autores orientan su trabajo hacia una aplicación de votación, la lista de usuarios autorizados está predefinida y no puede cambiarse.

El concepto de *Commitments* y *Nullifiers*, que se utilizan extensamente en este trabajo, se introduce en [18]. Al igual que en el protocolo propuesto, en ZCash los usuarios pueden crear un Commitment cuando tienen el derecho de realizar una acción y deben publicar un Nullifier para consumirlo, acompañado de un zk-SNARK que verifica la relación entre estos hashes.

La Tabla I compara el protocolo que proponemos con los trabajos mencionados en esta Sección. Como se muestra, nuestro protocolo es capaz de introducir nuevos usuarios dinámicamente en el sistema, no requiere infraestructura externa fuera del control del esquema y proporciona protección contra ataques de repetición.

III. FUNDAMENTOS TEÓRICOS

Para conseguir nuestro protocolo PPKTP, necesitaremos desacoplar los pasos de Autenticación y Solicitud de Clave, de forma que el servidor no pueda asociar a los clientes que realizan el segundo con las credenciales que utilizaron en el primero. Para ello, empleamos la primitiva criptográfica zk-SNARK, que se define formalmente en la Sección III-A.

En este trabajo utilizamos zk-SNARKs para demostrar la pertenencia a un conjunto de usuarios válidos. Para acelerar la generación de pruebas zk-SNARK, el conjunto mencionado se estructura como un árbol Merkle, en el que demostrar la pertenencia a un grupo solo conlleva $\log_2(N)$ operaciones, donde N es el número total de usuarios. Por último, empleamos criptografía asimétrica para transmitir de forma segura la clave QRNG al usuario en el último paso del protocolo.

III-A. zk-SNARKs

Los *Argumentos de Conocimiento Cero Sucintos No Interactivos* (o *zk-SNARK*, por sus siglas en inglés) son un subconjunto de las pruebas de conocimiento cero no interactivas. Un zk-SNARK se construye a partir de una relación R entre una *afirmación* pública x y un *testigo* privado w. Si la relación entre w y x se mantiene, entonces $(x, w) \in R$. De la descripción de R puede derivarse un parámetro de seguridad λ. Formalmente, un esquema zk-SNARK Π para una relación R se compone de la tupla $(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$, definida como sigue:

- $\mathsf{Setup}(R) \rightarrow (pk, vk)$: Toma una relación y devuelve el *Common Reference String* (*CRS*), que se divide en la Clave de Prueba pk y la Clave de Verificación vk.
- $\mathsf{Prove}(pk, x, w) \rightarrow \pi$: A partir de pk, un testigo w y una afirmación x, se genera una prueba π.
- $\mathsf{Verify}(vk, x, \pi) \rightarrow Accept/Reject$: A partir de vk, una prueba π y la correspondiente afirmación x, se obtiene $Accept$ o $Reject$. En el contexto de este trabajo, un *prueba válida* se refiere a cualquier (x_v, π_v) tal que $\mathsf{Verify}(vk, x_v, \pi_v) = Accept$.

Un esquema zk-SNARK también debe cumplir los siguientes requisitos de seguridad [19]:

Completeness: La probabilidad

$$\Pr\left[\begin{array}{c}(pk, vk) \leftarrow \mathsf{Gen}(R) \\ \pi \leftarrow \mathsf{Prove}(pk, x, w) \\ (x, w) \in R\end{array} : \neg\mathsf{Verify}(vk, x, \pi)\right]$$

es despreciable. Intuitivamente, esto significa que un usuario honesto es capaz de convencer a un verificador de que $(x, w) \in R$.

Knowledge Soundness: Para cada adversario eficiente A, existe un extractor eficiente Ext_A con acceso al estado interno de A tal que la probabilidad

$$\Pr\left[\begin{array}{c}((pk, vk), aux) \leftarrow \mathsf{Gen}(R) \\ (x, \pi) \leftarrow \mathsf{A}(R, aux, (pk, vk)) \\ w \leftarrow Ext_A(R, aux, (pk, vk))\end{array} : \begin{array}{c}(x, w) \notin R \wedge \\ \mathsf{Verify}(vk, x, \pi)\end{array}\right]$$

es despreciable, donde aux es una entrada auxiliar producida por Gen. Intuitivamente, esto significa que un usuario deshonesto no puede generar una prueba válida si no conoce w.

Zero-knowledge: Para cada adversario A que actúa como caja negra y $(x, w) \in R$, existe un simulador $\mathsf{Sim}((pk, vk), aux, x)$ tal que se cumple la siguiente igualdad:

$$\Pr\left[\begin{array}{c}((pk, vk), aux) \leftarrow \mathsf{Gen}(R) \\ \pi \leftarrow \mathsf{Prove}(pk, x, w)\end{array} : \mathsf{A}((pk, vk), aux, x, \pi) = 1\right]$$

$$\approx$$

$$\Pr\left[\begin{array}{c}((pk, vk), aux) \leftarrow \mathsf{Gen}(R) \\ \pi \leftarrow \mathsf{Sim}((pk, vk), aux, x)\end{array} : \mathsf{A}((pk, vk), aux, x, \pi) = 1\right]$$

donde aux es una entrada auxiliar producida por Gen. Intuitivamente, esto significa que un atacante no puede averiguar nada sobre un testigo w a partir de una prueba π, una afirmación x y un par de claves (pk, vk).

Weak Simulation-Extractability: Para cada adversario eficiente A con acceso a un oráculo O, existe un extractor eficiente Ext_A con acceso al estado interno de A tal que la probabilidad

$$\Pr\left[\begin{array}{c|c} ((pk,vk),aux) \leftarrow \mathsf{Gen}(R) & x \notin Q \\ (x,\pi) \leftarrow A^{O(aux)}(R,aux,(pk,vk)) & \wedge (x,w) \notin R \\ w \leftarrow Ext_A(R,aux,(pk,vk)) & \wedge \mathsf{Verify}(vk,x,\pi) \end{array}\right]$$

es despreciable, donde aux es una entrada auxiliar producida por Gen y Q es la lista de afirmaciones x_i solicitada al oráculo. Esto significa que A no puede generar una prueba válida para un enunciado que no se haya obtenido del oráculo. Nótese que existe una noción más fuerte de Simulation-Extractability que requiere que tanto la sentencia x como la prueba π no estén incluidas en Q. Sin embargo, para este trabajo sólo requerimos la versión débil de Simulation-Extractability.

En la práctica, los esquemas zk-SNARK pueden construirse mediante diferentes técnicas criptográficas. Los diseños más populares, incluido el de Groth16 [20] (que utilizaremos en este trabajo) implican la transformación de un cálculo en un circuito aritmético. El rendimiento de un esquema zk-SNARK, especialmente durante la configuración y la generación de pruebas, depende en gran medida del número de restricciones de este circuito [21], que vienen determinadas por el cálculo que se va a verificar.

III-B. Árboles Merkle

Los árboles Merkle [22] son una estructura de datos en forma de árbol binario. Cada elemento se identifica como $h_{i,j}$, donde i es el nivel del árbol y j la posición dentro de dicho nivel. Un Árbol Merkle de profundidad n contiene como máximo $l = 2^n$ nodos hoja $h_{n,j}$ que contienen un hash, y los nodos intermedios toman el valor $h_{k,j} = H(h_{k+1,m} \parallel h_{k+1,m+1})$, donde $\parallel$ es el operador de concatenación, H es una función hash y $m = 2j$. El nodo $h_{0,0}$ se denomina *raíz*, y su valor está determinado por todos los demás nodos del Árbol.

La principal ventaja de los Árboles Merkle es que comprobar si un elemento e existe dentro de la lista de nodos hoja es solo $\mathcal{O}(\log l)$, mientras que la misma operación para una lista sería $\mathcal{O}(l)$. Para ello, debe proporcionarse una lista de validación $\{h_{n-1,k_{n-1}}, h_{n-2,k_{n-2}}, ..., h_{1,k_1}\}$ y el nodo raíz $h_{0,0}$ del Árbol Merkle. Si es posible reconstruir la raíz proporcionada a partir de $H(e)$ y la lista de validación, entonces se demuestra que e está dentro del Árbol de Merkle. Este método requiere que la función hash H utilizada para construir el Árbol de Merkle sea resistente a colisiones [23]: de lo contrario, un atacante podría demostrar que un elemento falso e_f está dentro del Árbol Merkle cuando no lo está.

Para el ámbito de este trabajo, se definen las siguientes operaciones para los Árboles Merkle:

1. $\mathsf{EmptyTree}(n) \rightarrow T$: Inicializa un Árbol Merkle vacío de profundidad n, con todos sus nodos hoja establecidos a $H(0)$.
2. $\mathsf{AddLeaf}(T,h) \rightarrow T'$: Recibe un Árbol Merkle T y un hash h y devuelve un árbol modificado, en el que el nodo cuyo valor es $H(0)$ que esté situado más a la izquierda es sustituido por h. Los nodos intermedios y raíz se actualizan en consecuencia.
3. $\mathsf{GetRoot}(T) \rightarrow R$: Recibe un Árbol Merkle T y devuelve la raíz $h_{0,0}$.
4. $\mathsf{GetIndexOf}(T,h) \rightarrow index$: Recibe un Árbol Merkle T y un hash h y devuelve i tal que $h_{n,i} = h$ para algún $h_{n,i}$ en el Árbol.
5. $\mathsf{ValidationList}(T,i) \rightarrow Val = \{h_{n-1,k_{n-1}}, h_{n-2,k_{n-2}}, ..., h_{1,k_1}\}$: Recibe un Árbol Merkle T y un índice i y devuelve la lista de validación necesaria para validar la hoja $h_{n,i}$.
6. $\mathsf{IsLeafOfTree}(h,i,Val,R) \rightarrow true/false$: Recibe un hash h, su índice i, una lista de validación $Val = \{h_{n-1,k_{n-1}}, h_{n-2,k_{n-2}}, ..., h_{1,k_1}\}$ y una raíz R y devuelve $true$ si h es la hoja con índice i del árbol con raíz = R, y $false$ en caso contrario.

III-C. Mecanismo de Encapsulación de Claves

Un mecanismo de encapsulación de claves [24], [25] (o *KEM*, por sus siglas en inglés) es un método de distribución de claves entre dos partes que utiliza criptografía asimétrica. El emisor utiliza la clave pública del receptor para cifrar el material clave de entrada de forma que sólo el receptor pueda descifrarlo. Ambas partes deben acordar qué algoritmo de cifrado asimétrico se utilizará para la encapsulación. En el Cifrado Híbrido de Clave Pública [26], el material de clave se introduce posteriormente en una KDF para obtener una clave simétrica.

Un KEM se compone de las siguientes operaciones:

- $\mathsf{KeyGen}(\lambda) \rightarrow (pk, sk)$: Toma un parámetro de seguridad λ y genera un par de claves simétricas.
- $\mathsf{Encap}(ikm, pk) \rightarrow c$: Cifra una clave de entrada ikm con una clave pública pk.
- $\mathsf{Decap}(c, sk) \rightarrow km$: Descifra un texto cifrado con una clave secreta sk para obtener el material de claves.

IV. Protocolo de Transmisión de Clave que Preserva la Privacidad (PPKTP)

IV-A. Visión General

El protocolo requiere dos interacciones diferentes entre los Usuarios y el Servidor: la Autenticación y la Solicitud de Clave. La principal aportación de nuestro protocolo es la capacidad de desacoplar ambas interacciones, que se consigue mediante el uso de zk-SNARKs: al solicitar una clave, los Usuarios deben demostrar que han realizado previamente la autenticación.

Durante la autenticación, los Usuarios deben presentar credenciales válidas que certifiquen su identidad, como un certificado digital. Si estas credenciales son correctas, se les permite publicar un *Commitment*, que el Servidor incluye en una estructura de Commitments válidos (que, por razones de eficiencia, es un Árbol Merkle). Así, un Commitment identifica a un Usuario concreto, porque se publica junto con la información de autenticación. También representa el derecho del Usuario a obtener una clave QRNG en pasos posteriores.

Los Usuarios deben generar un *Nullifier*, que es un valor que está criptográficamente relacionado con su Commitment: se generan a partir de las mismas entradas, pero sólo el Usuario que ha creado ambos sabe que están relacionados. También generan una prueba zk-SNARK que certifica la

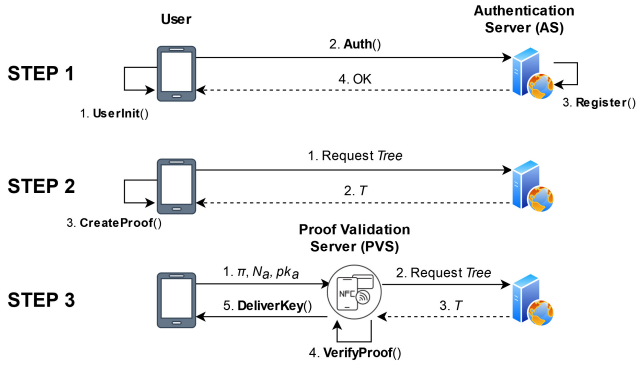

Figura 1. Diagrama del protocolo propuesto.

relación entre el Commitment y el Nullifier, pero sin revelar el valor del primero de ambos. La prueba también certifica que el Commitment está dentro de la estructura de Commitments válidos.

Para realizar la Solicitud de Clave, los Usuarios envían el Nullifier y la prueba zk-SNARK antes mencionados al Servidor, que valida dichos valores. Si la prueba es válida, el Servidor (1) sabe que existe un Commitment relacionado con este Nullifier específico dentro de la estructura de Commitments válidos, pero (2) no sabe de cuál de todos los Commitments válidos se trata. Cabe destacar que (1) demuestra que el Usuario ha realizado el intercambio de Autenticación, mientras que (2) garantiza que ni el Servidor ni ninguna otra entidad puedan conocer la identidad del Usuario. Por último, el Servidor almacena el Nullifier para evitar ataques de repetición: cualquier otra Solicitud de Clave que contenga el mismo Nullifier será rechazada.

IV-B. Arquitectura

En el protocolo PPKTP participan las siguientes entidades:

- *Usuario (U)*: ejecuta el protocolo e inicia la comunicación para ambos Pasos. En el Paso de Autenticación, los Usuarios deben proporcionar su certificado digital y generar un Commitment. A continuación, generan un Nullifier y un zk-SNARK vinculándolo al Commitment.
- *Servidor de Autenticación* (AS): recibe las solicitudes de autenticación de los Usuarios. Cuando el certificado de un usuario se valida correctamente, el AS inserta el Commitment proporcionado en el árbol Merkle de Commitment y actualiza los nodos y la raíz necesarios. El AS también debe proporcionar una copia del Árbol Merkle a cualquiera que lo solicite.
- *Servidor de Validación de Pruebas* (PVS): recibe pruebas zk-SNARK de los Usuarios y las valida. El PVS almacena la lista de todos los Nullifiers publicados, por lo que rechaza cualquier prueba que vaya acompañada de un Nullifier que ya haya sido publicado. El PVS necesita información sobre el Árbol Merkle para validar las pruebas zk-SNARK, por lo que debe comunicarse con el AS para obtener esta estructura de datos. Es la única entidad del protocolo que puede acceder al QRNG, que utiliza para generar material de claves para los Usuarios que presenten pruebas zk-SNARK válidas.

El AS y el PVS son entidades lógicas, por lo que pueden implementarse tanto como uno como dos programas diferentes. Toda la información almacenada por el AS y el PVS (es decir, el Árbol Merkle de Commitments y la Lista de Nullifier) está expuesta para que los Usuarios puedan acceder libremente a ella sin comprometer la seguridad del protocolo.

La figura 1 muestra un diagrama de la interacción entre el Usuario, el Servidor de Autenticación y el Servidor de Validación de Pruebas, tal y como se esboza en IV-A.

Name	Description
crs	Common Reference String del esquema zk-SNARK
pk_{crs}	Clave de Prueba del esquema zk-SNARK
vk_{crs}	Clave de Verificación del esquema zk-SNARK
T	Árbol Merkle de Commitments
T_{old}	Lista de anteriores raíces de T
L_N	Lista de Nullifiers ya publicados
$cert$	certificado digital del Usuario
ck	Clave privada de $cert$

Tabla II
LISTA DE NOTACIÓN DEL PROTOCOLO PROPUESTO.

IV-C. Definición del Protocolo

El protocolo PPKTP utiliza como primitivas una función hash H, un esquema zk-SNARK *ZK = (ZK.Setup, ZK.Prove, ZK.Verify)*, un esquema de Árbol Merkle *M = (M.EmptyTree, M.AddLeaf, M.GetRoot, M.GetIndexOf, M.ValidationList, M.IsLeafOfTree)* y un KEM *K = (K.KeyGen, K.Encap, K.Decap)*. La Tabla II indica la notación utilizada para referenciar a los elementos que participan en el protocolo.

Inicialización de parámetros: Antes de la ejecución del protocolo, es necesario inicializar algunos parámetros. La operación que se ejecuta en este paso inicial es:

- $\mathsf{ServerSetup}(R, n) \rightarrow ((pk_{crs}, vk_{crs}), T, T_{old}, L_N)$: Se inicializan las claves y las estructuras de datos que se utilizarán durante la ejecución del protocolo. Se inicializa el esquema zk-SNARK a partir de su relación R, y se crea un Árbol Merkle vacío de profundidad n.

La relación con la que se inicializa el esquema zk-SNARK es la siguiente:

$$R = \left\{ \begin{matrix} (x, w) = & C = H(sk \,\|\, \rho), \\ ((N, root), & : N = H(pk \,\|\, \rho), \\ (\rho, pk, sk, C, i_C, Val)) & \mathsf{M.IsLeafOfTree}(C, i_C, Val, root) \end{matrix} \right\} \quad (1)$$

Intuitivamente, la relación R demuestra que el Commitment y el Nullifier se generan a partir de un par de claves (pk, sk) y un valor secreto ρ, que sólo conoce el Usuario que creó ambos hashes. También se demuestra que el Commitment está dentro de un Árbol Merkle con raíz $root$. Esta misma relación se muestra en la Figura 2.

Paso 1: Autenticación: Para realizar el primer paso del protocolo, un usuario U debe crear primero un *Commit Note*, que contiene un valor secreto y un par de claves. A continuación, debe autenticarse ante el AS. Si lo consigue, el AS añadirá el Commitment C_U a T. El Paso de Autenticación requiere la ejecución de las siguientes operaciones:

- $\mathsf{UserInit}(\lambda) \rightarrow note$: U crea un *Commit Note* $note_U$ que contiene el valor secreto ρ_U y un par de claves de cifrado asimátrico (pk_U, sk_U). A continuación, almacena $note_U$ de forma segura. El tamaño de estos valores depende del parámetro de seguridad λ.

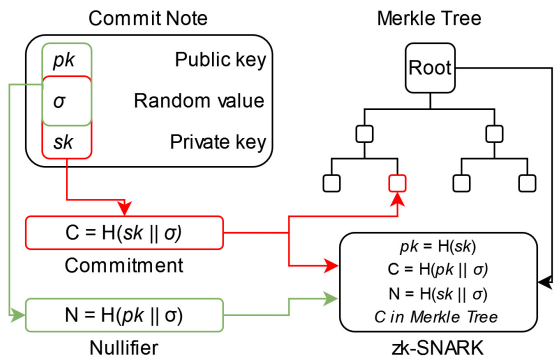

Figura 2. Diagrama de la relación entre Commitments y Nullifiers.

- $\mathsf{Auth}(cert, ck, note) \rightarrow (C, \sigma_C)$: U utiliza $note_U$ para generar un Commitment $C_U = H(pk_U \parallel \rho)$. donde "$\parallel$".[es] el operador de concatenación. A continuación, firma C_U con la clave privada de su certificado ck_U.
- $\mathsf{Register}(cert, C, \sigma_C, T, T_{old}) \rightarrow (true/false, T, T_{old})$: El AS comprueba la validez del certificado $cert_U$ presentado por U. A continuación, verifica la firma σ_U con la clave pública de $cert_U$. Si es válida, el AS inserta C_U en el árbol Merkle T.

Cabe destacar que, aunque el método de autenticación del protocolo que presentamos implica el uso de certificados, el Paso de Autenticación es lo suficientemente flexible como para permitir otros tipos de autenticación o atestación. Se podrían implementar diferentes AS que soliciten otra información a U (por ejemplo, contraseñas, datos biométricos, ...), siempre y cuando inserten C_U en T después de una autenticación exitosa.

Paso 2: Generación de Prueba: Después de una autenticación exitosa, el Commitment C_U de U se almacena en T. En este Paso, U demuestra que conoce algún Commitment dentro de T generando una prueba zk-SNARK. Para ello, U debe descargar primero una copia del Árbol de Merkle del AS. A continuación, U puede utilizar el Árbol y $note_U$ para formar el Testigo zk-SNARK $w = (\rho_U, pk_U, sk_U, C_U, i_C, Val_U)$, necesario para crear la prueba zk-SNARK. Se ejecuta la siguiente operación:

- $\mathsf{CreateProof}(note, T, pk_{crs}) \rightarrow (\pi, x)$: U recalcula su Commitment C_U y genera el Nullifier N_U, ambos a partir de los valores dentro de $note_U$. A continuación, utiliza el esquema zk-SNARK para crear una prueba π_U que demuestre que C_U está dentro de T (sin revelar qué elemento es) y que C_U y N_U se generaron a partir del mismo *Commit Note*.

Paso 3: Solicitud de Clave: En este último Paso, U envía la prueba de que ha realizado con éxito el Paso 1 al PVS, que es la única entidad del protocolo con acceso al QRNG. El mensaje de U al PVS incluye π_U y la afirmación $x = (N_U, \text{raíz})$. Si la información proporcionada por U se valida correctamente, el PVS responderá a U con material de claves generado por el QRNG.

Las operaciones realizadas por U y el PVS en el Paso de Solicitud de Clave son las siguientes:

- $\mathsf{VerifyProof}(\pi, x, T, T_{old}, L_N, vk_{crs}) \rightarrow (true/false, L_N)$: El PVS verifica la prueba zk-SNARK π_U generada por U en la fase anterior. También comprueba que la raíz $root_U$ que forma parte de la afirmación zk-SNARK x_U se corresponde con la raíz del Árbol Merkle actual o cualquiera de sus versiones pasadas y que el Nullifier N_U no se ha presentado todavía. Si todas las comprobaciones tienen éxito, N_U se incluye en una lista de "Nullifiers utilizados", de forma que no se puede volver a utilizar.
- $\mathsf{DeliverKey}(t, pk) \rightarrow enc$: El PVS genera t bytes de material clave con su QRNG y luego lo cifra con la clave pública pk de U.

V. Implementación

Proporcionamos una implementación de nuestro protocolo para demostrar su aplicabilidad. Representamos un caso de uso real en el que las claves QRNG se transmiten a través de NFC al dispositivo móvil del usuario, de forma que estas claves podrían utilizarse para cifrar de forma segura la información del usuario dentro de su teléfono.

Nuestra implementación se compone de los siguientes elementos:

- Tres programas en el lenguaje de programación Java que representan al Usuario, al Servidor de Autenticación y al Servidor de Validación de Pruebas. Ambos servidores están ubicados en la misma máquina, y tienen acceso local a sus archivos compartidos. El AS es una API REST implementada con Spring Boot. Dado que los servidores no necesitan mantener ninguna información privada, el AS permite a los Usuarios solicitar cualquiera de sus estructuras de datos: el Árbol Merkle, la ista de Nullifiers y la lista de raíces de antiguos Árboles Merkle.
- Una librería encargada de la implementación del esquema zk-SNARK, incluyendo la definición de circuitos y la generación y validación de pruebas.
- Un dispositivo QRNG para la generación del material clave.
- Un túnel NFC para la comunicación entre el Usuario y el PVS. Para ello, hemos implementado una App Android que emplea el código del Usuario y actúa como una Smart Card NFC. El PVS tiene acceso a un Lector NFC *ACR 122U USB* para recibir las peticiones de los Usuarios.

V-A. *zk-SNARKs*

Todas las operaciones relacionadas con la generación y validación de zk-SNARK en este trabajo se han implementado con dos librerías diferentes: libsnark [27] y ZoKrates [28], a las que nos referiremos como *backends de zk-SNARK*. Ambas librerías requieren definir las sentencias que el zk-SNARK debe probar, que en este protocolo corresponden a las sentencias que se especificaron en la Sección IV.

Ambos backends de zk-SNARK requieren un proceso similar para la definición del esquema zk-SNARK. En primer lugar, la relación R (en este caso, la relación mostrada en la Ecuación 1 debe codificarse en el lenguaje específico del backend de zk-SNARK. A continuación, se compila este código y se genera un circuito aritmético que representa el cálculo. Para ambos backends de zk-SNARK, los usuarios

pueden interactuar con el programa compilado a través de la línea de comandos.

Una vez compilado, se puede generar el Common Reference String (CRS) a partir del circuito aritmético. El CRS está compuesta por la Clave de Prueba y la Clave de Verificación. Para ejecutar Prove, los usuarios deben proporcionar tanto la Afirmación $x = (\text{raíz}, N)$ y el Testigo $w = (\rho, pk, sk, C, i_C, Val)$. Para verificar una prueba π, sólo se requiere la Declaración.

Hemos elegido Groth16 [20] como esquema zk-SNARK, ya que consigue los tiempos más rápidos de generación y validación de pruebas [21]. Además, satisface todas las propiedades mencionadas en la Sección III-A [29].

V-B. QRNG

En este trabajo hemos utilizado dos QRNG diferentes:

- *"Quantis QNRG USB"de IDQuantique*: Este dispositivo es accesible a través de una interfaz USB, y sus manufacturadores proporcionan una API para múltiples lenguajes de programación. Su velocidad de generación de claves es de aproximadamente 4 Mb/s.
- *Módulo QRNG del Centro de Supercomputación de Galicia (CESGA)*: Este supercomputador tiene infraestructura nativa de QRNG que puede generar hasta 400 Mb/s de material de clave. Se requiere una conexión remota SSH para acceder al dispositivo, lo cual introduce un coste adicional para la obtención de material de clave.

V-C. Túnel NFC

Near Field Communication (NFC) [30], [31], [32], [33] es una tecnología que permite la comunicación inalámbrica entre dos dispositivos físicamente cercanos: un lector que inicia la comunicación y una Smart Card que responde a las peticiones del lector. Los mensajes NFC se encapsulan en Application Protocol Data Units (APDUs).

NFC se considera más seguro que otras tecnologías de comunicaciones inalámbricas como Bluetooth debido a su pequeña área de transmisión (que impide que atacantes puedan capturar las APDUs) y su rápido proceso de inicialización de la conexión. La mayoría de dispositivos móviles soportan NFC, e incluyen la capacidad de emular una Smart Card para crear APDUs de manera dinámica para responder a lectores NFC. Esto hace que NFC sea un método cómodo para transmitir material de clave a los usuarios de nuestro protocolo, ya que este se almacenaría en sus teléfonos móviles.

En nuestra implementación, toda la comunicación entre Usuarios y el PVS en el Paso 3 se realiza mediante NFC. El lector NFC del PVS interactúa con el teléfono móvil de U, que contiene una App que ejecuta nuestro protocolo. Dado que es el lector NFC el que debe iniciar la comunicación, se requiere un método *ad hoc* para transmitir la prueba zk-SNARK del Usuario al PVS.

La Figura 3 muestra el método definido para esta transmisión. El PVS debe primero asegurarse de que el dispositivo móvil de U contiene la App específica que ejecuta nuestro protocolo. Si es así, el PVS pregunta la longitud del mensaje a enviar, y entonces realiza tantas peticiones como sean necesarias para recibir la prueba zk-SNARK completa. La prueba se divide en segmentos para poder encapsularse en APDUs, cuyo tamaño está limitado a 255 bytes. a continuación, el

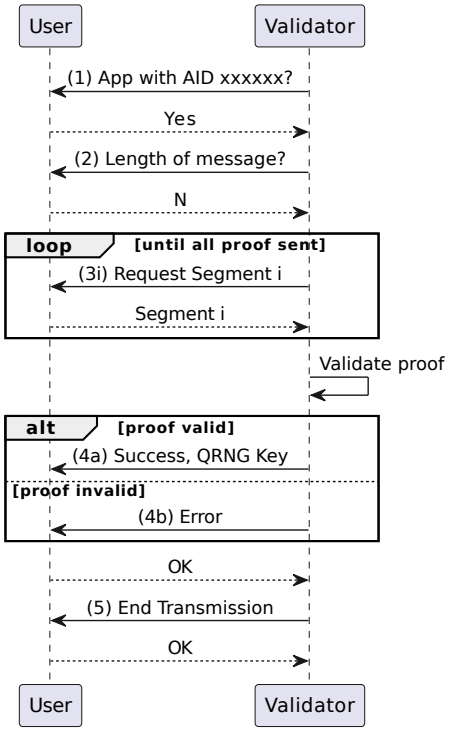

Figura 3. Diagrama de secuencia del túnel NFC.

PVS comprueba la validez de la prueba y en caso afirmativo devuelve la clave QRNG. En caso contrario, devuelve un error. Finalmente, el PVS envía un mensaje de "Fin de la Transmisión"que señaliza al dispositivo móvil de que puede cerrar la conexión.

VI. Conclusión y Trabajo Futuro

En este trabajo hemos presentado un Protocolo de Transmisión de Clave que Preserva la Privacidad para distribuir material criptográfico generado por QRNG, en el que se utilizan zk-SNARKs para asegurar que el servidor no puede identificar cuál de los usuarios está recibiendo cada clave. Por lo tanto, los usuarios no necesitan confiar en el servidor para solicitar estas claves. El protocolo puede ser utilizado para proporcionar material de clave de alta calidad a usuarios finales, que podrán incorporar a su pila de protocolos de comunicación para aumentar su seguridad.

Nuestra implementación del protocolo demuestra su aplicabilidad a un caso de uso real, en el que las claves generadas por QRNG son enviadas al dispositivo móvil de los usuarios a través de NFC, utilizando un método personalizado para transmitir información arbitraria encapsulada en APDUs. En nuestra implementación utilizamos dos librerías diferentes de generación y validación de zk-SNARKs. Continuaremos con este trabajo proporcionando demostraciones formales de la seguridad de nuestro protocolo y midiendo el rendimiento de nuestra implementación bajo diferentes parámetros.

Reconocimiento

Este trabajo está financiado por el Plan Complementario de Comunicaciones Cuánticas, Ministerio de Ciencia e Innovación(MICINN), Plan de Recuperación NextGeneration, Unión Europea (PRTR-C17.I1, CITIC Ref. 305.2022), y la Xunta

de Galicia (Agencia Gallega de Innovación, GAIN, CITIC Ref. 306.2022) D.S. reconoce el apoyo de Xunta de Galicia y de la Unión Europea (European Social Fund - ESF), beca [ED481A-2023-219].

Este trabajo es parte de los proyectos TED2021-130369B-C31 y TED2021-130492B-C21, financiados por MCIN/AEI/10.13039/501100011033 y por "ERDF A way of making Europe".

Trabajo desarrollado gracias al acceso concedido por el Centro de Supercomputación de Galicia a la infraestructura basada de tecnologías cuánticas de la información que permita impulsar I+D+I en Galicia. Esta infraestructura fue financiada por la Unión Europea, a través del FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER), como parte de la respuesta de la Unión Europa a la pandemia de la COVID-19.

Referencias

[1] K. Bhattacharjee and S. Das, "A search for good pseudo-random number generators: Survey and empirical studies," *Comput. Sci. Rev.*, vol. 45, no. C, aug 2022.

[2] A. Saini, A. Tsokanos, and R. Kirner, "Quantum randomness in cryptography—a survey of cryptosystems, RNG-based ciphers, and QRNGs," *Information*, vol. 13, no. 8, p. 358, 2022.

[3] N. Heninger, Z. Durumeric, E. Wustrow, and J. Halderman, "Mining your Ps and Qs: detection of widespread weak keys in network devices," in *21st USENIX Security Symposium (USENIX Security 12)*, 08 2012, pp. 35–35.

[4] D. Antonioli, N. O. Tippenhauer, and K. B. Rasmussen, "The KNOB is broken: Exploiting low entropy in the encryption key negotiation of bluetooth BR/EDR," in *28th USENIX Security Symp. (USENIX Security 19)*, Santa Clara, CA, Aug. 2019, pp. 1047–1061.

[5] D. Kaplan, S. Kedmi, R. Hay, and A. Dayan, "Attacking the Linux PRNG on Android: Weaknesses in seeding of entropic pools and low boot-time entropy," in *Proc. of the 8th USENIX Conference on Offensive Technologies*, ser. WOOT'14, USA, 2014, p. 14.

[6] D. Hurley-Smith and J. Hernandez-Castro, "Quantum leap and crash: Searching and finding bias in quantum random number generators," *ACM Trans. Priv. Secur.*, vol. 23, no. 3, pp. 1–25, jun 2020.

[7] M. Stipčević, "Quantum random number generators and their use in cryptography," in *2011 Proceedings of the 34th International Convention MIPRO*, 2011, pp. 1474–1479.

[8] M. M. Jacak, P. Jóźwiak, J. Niemczuk, and J. E. Jacak, "Quantum generators of random numbers," *Scientific Reports*, vol. 11, no. 1, p. 16108, Aug 2021.

[9] A. Vassilev and R. Staples, "Entropy as a service: Unlocking cryptography's full potential," *Computer*, vol. 49, no. 9, pp. 98–102, 2016.

[10] S. Liu, L. Chen, H. Yu, S. Gao, and H. Fang, "BP-AKAA: Blockchain-enforced privacy-preserving authentication and key agreement and access control for iiot," *Journal of Information Security and Applications*, vol. 73, p. 103443, 2023.

[11] H. An and J. Chen, "Elearnchain: A privacy-preserving consortium blockchain system for e-learning educational records," *Journal of Information Security and Applications*, vol. 63, p. 103013, 2021.

[12] P. Li, J. Lai, and Y. Wu, "Event-oriented linkable and traceable anonymous authentication and its application to voting," *Journal of Information Security and Applications*, vol. 60, p. 102865, 2021.

[13] J. Lee, J. Y. Hwang, J. Choi, H. Oh, and J. Kim, "Sims : Self sovereign identity management system with preserving privacy in blockchain," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1241, 2019.

[14] D. A. Luong and J. H. Park, "Privacy-preserving blockchain-based healthcare system for iot devices using zk-snark," *IEEE Access*, vol. 10, pp. 55 739–55 752, 2022.

[15] A. Kavousi, Z. Wang, and P. Jovanovic, "Sok: Public randomness," Cryptology ePrint Archive, Paper 2023/1121, 2023, https://eprint.iacr.org/2023/1121. [Online]. Available: https://eprint.iacr.org/2023/1121

[16] M. Raikwar and D. Gligoroski, "Sok: Decentralized randomness beacon protocols," in *Australasian Conference on Information Security and Privacy*. Springer, 2022, pp. 420–446.

[17] M. Haahr, "Random. org: True random number service," *School of Computer Science and Statistics, Trinity College, Dublin, Ireland. Website (http://www. random. org). Accessed*, vol. 10, 2010.

[18] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," *GitHub: San Francisco, CA, USA*, vol. 4, p. 220, 2016.

[19] H. Ko, I. Lee, S. Lee, J. Kim, and H. Oh, "Efficient verifiable image redacting based on zk-snarks," Cryptology ePrint Archive, Paper 2020/1579, pp. 213–226, 2020.

[20] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology – EUROCRYPT 2016*, M. Fischlin and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 305–326.

[21] K. Baghery, Z. Pindado, and C. Ràfols, "Simulation extractable versions of groth's zk-snark revisited," in *Cryptology and Network Security*, S. Krenn, H. Shulman, and S. Vaudenay, Eds., 2020, pp. 453–461.

[22] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 369–378.

[23] C. Coronado, "On the security and the efficiency of the Merkle signature scheme," *IACR Cryptol. ePrint Arch.*, vol. 2005, p. 192, 2005.

[24] M. Campagna and A. Petcher, "Security of hybrid key encapsulation," Cryptology ePrint Archive, Paper 2020/1364, 2020.

[25] R. Barnes, K. Bhargavan, B. Lipp, and C. Wood, "Hybrid public key encryption," RFC Editor, RFC 9180, 5 2022.

[26] J. Alwen, B. Blanchet, E. Hauck, E. Kiltz, B. Lipp, and D. Riepel, "Analysing the hpke standard," Cryptology ePrint Archive, Paper 2020/1499, 2020.

[27] S. Lab, "libsnark: a C++ library for zk-SNARK proofs," https://github.com/scipr-lab/libsnark, 2020.

[28] J. Eberhardt and S. Tai, "Zokrates - scalable privacy-preserving off-chain computations," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1084–1091.

[29] K. Baghery, M. Kohlweiss, J. Siim, and M. Volkhov, "Another look at extraction and randomization of groth's zk-snark," Cryptology ePrint Archive, Paper 2020/811, 2020.

[30] V. Coskun, B. Ozdenizci Kose, and K. Ok, "A survey on near field communication (NFC) technology," *Wireless Personal Communications*, vol. 71, 08 2013.

[31] E. Haselsteiner and K. Breitfuss, "Security in near field communication (NFC)," *Workshop on RFID Security*, 01 2006.

[32] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger, "NFC devices: Security and privacy," in *2008 Third International Conference on Availability, Reliability and Security*, 2008, pp. 642–647.

[33] M. M. Mahinderjit Singh, K. Adzman, and R. Hassan, "Near field communication (NFC) technology security vulnerabilities and countermeasures," *International Journal of Engineering and Technology*, vol. 7, pp. 298–305, 12 2018.