BRNO UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering and Communication

Bitcoin aplicando firmas de Schnorr

Trabajo Fin de Grado

Pablo González de la Lastra González

Brno, 2022

Declaración

Por la presente declaro que este trabajo es de mi autoría y que lo he elaborado por mi cuenta. Todas las fuentes, referencias y bibliografía utilizadas durante la elaboración de este trabajo están debidamente citadas y enumeradas con la referencia completa a la fuente.

Pablo González de la Lastra González

Revisora: M. Sc. Sara Ricci, Ph.D.

Agradecimientos

Me gustaría dar las gracias a mi asesora, Sara Ricci, por su orientación y los conocimientos que me ha proporcionado durante la preparación de este proyecto, y por introducirme en el ámbito de las criptomonedas y los protocolos seguros multipartitos. También me gustaría agradecer a mi familia su apoyo a lo largo de mis estudios.

Resumen

Este trabajo de fin de grado investiga los protocolos multipartitos implícitos basados en el esquema de firma Schnorr y sus beneficios para el ecosistema Bitcoin. En primer lugar, el trabajo presenta el esquema de firma Schnorr, posteriormente, se describen varias construcciones multipartitas que producen firmas Schnorr, y se discuten las formas propuestas de adopción de las firmas Schnorr en Bitcoin.

La solución se basa en un esquema de multi-firma MuSig, y puede ser utilizado como medio de autenticación de transacciones de múltiples factores.

Palabras clave

Multifirma, Esquema multifirma Schnorr, Bitcoin

CONTENIDOS

1 Introducción

Como cada vez son más los aspectos de la vida cotidiana que se trasladan al mundo digital, no es de extrañar que el dinero también se una a esta tendencia. En las últimas décadas, los investigadores intentaron diseñar una plataforma segura y fiable que pudiera servir para este propósito. Estos diseños requerían el empleo de la criptografía para garantizar propiedades como la autenticidad y la propiedad.

A lo largo de los años, los criptógrafos han diseñado varias plataformas de dinero digital, pero los diseños carecían de una propiedad crucial para su confianza: la descentralización. Para evitar el doble gasto, un problema por el que una parte podría crear simultáneamente dos pagos a diferentes partes con el mismo recurso, los criptógrafos recurrieron a utilizar una autoridad central para compensar los pagos. Esta autoridad central tiene un control total sobre la validez de las transacciones, por lo que se convierte en un punto débil de la plataforma.

Un gran avance en el diseño del dinero digital se produjo en el año 2008, cuando alguien, bajo el seudónimo de Satoshi Nakamoto, presentó el famoso libro blanco [1], que introducía una solución descentralizada al problema del doble gasto; y también describía el diseño de la moneda digital conocida como Bitcoin, que revolucionó el mundo. Desde entonces, se diseñaron muchas monedas digitales basadas en principios similares.

La propiedad del dinero digital en el sistema necesita ser probada cuando un propietario desea transferirlo. La prueba se consigue mediante la creación de una firma digital, una conocida operación criptográfica, utilizando información secreta específica ligada al dinero. El secreto debe ser conocido sólo por el propietario; por lo tanto, sólo él debe poder gastar el dinero.

En algunos casos, es deseable poder especificar condiciones de gasto más complejas que la mera emisión de una única señal digital. Para facilitar estos casos de uso, Bitcoin se complementa con un lenguaje de scripting especializado que permite especificar varias condiciones de gasto.

Para gastar dinero vinculado a un script, éste debe publicarse junto con la transacción. El script aumenta el tamaño de la transacción, lo que se traduce en mayores tasas de transacción y en la reducción del ancho de banda del sistema.

Además, la privacidad del gastador disminuye, ya que el script de gasto especializado simplifica la huella digital de la transacción. Para solucionar estos problemas, la comunidad de Bitcoin comenzó a investigar esquemas de firma digital alternativos.

Bitcoin actualmente utiliza el algoritmo de firma ECDSA para sus firmas, pero ha demostrado ser inflexible en el diseño de varios protocolos que podrían ser utilizados para expresar algunas condiciones de gasto especializadas implícitamente, dentro de una sola firma. El esquema de firma alternativo más destacado con las propiedades deseadas es el esquema de firma Schnorr. El esquema de firma Schnorr se basa en operaciones y supuestos criptográficos similares a los de ECDSA, pero presenta muchas ventajas y no tiene prácticamente ningún inconveniente. En particular, las firmas Schnorr presentan una rica estructura algebraica adecuada para la construcción de protocolos multipartitos. Las desventajas están relacionadas principalmente con la falta de estandarización y la no disponibilidad de implementaciones en las principales bibliotecas criptográficas.

El objetivo de este proyecto es investigar los protocolos multipartitos implícitos y sus beneficios para el ecosistema Bitcoin. Para ello, primero introduzco el esquema de firma Schnorr, describo varias construcciones multipartitas que producen las firmas Schnorr, y discuto las formas propuestas de adopción de las firmas Schnorr en Bitcoin.

2 Esquema de las firmas de Schnorr

El esquema de firma de Schnorr [2] es un esquema de firma digital basado en el problema del logaritmo discreto (DLP) [3], derivado del esquema de identificación de Schnorr utilizando la heurística de Fiat-Shamir [4].

El esquema produce firmas eficientemente computables y verificables de corta longitud [5]. Hasta febrero de 2008, el esquema estaba cubierto por una patente estadounidense, lo que impedía una mayor adopción de las firmas Schnorr en ese momento en favor de esquemas de firma digital alternativos como ECDSA (Algoritmo de Firma Digital de Curva Elíptica) [6].

Se ha analizado la seguridad del esquema. Se ha demostrado que no se puede falsificar contra ataques de mensajes elegidos de forma adaptativa en el modelo de oráculo aleatorio (ROM) [7] por Pointcheval y Stern [8].

El esquema está parametrizado por una selección de:

· un grupo G de orden primo q;

· un elemento generador G del grupo G;

· una función hash criptográfica H : {0, 1}* → Zq.

El algoritmo aleatorio 2.1 describe el procedimiento de generación de pares de claves del esquema de firma Schnorr. Genera un par de claves que consiste en un número secreto x muestreado al azar de Zq y un elemento de grupo público X obtenido por la multiplicación del generador G por el número secreto x.

Algorithm 2.1: Schnorr Gen

Output: A pair of private and public key (x, X)

1 $x \leftarrow \mathbb{Z}_q$
2 $X = xG$
3 **return** (x, X)

El número secreto x, también denominado clave privada, puede introducirse en el algoritmo 2.2 junto con un mensaje m, para producir una firma de m verificable con la clave pública X. El algoritmo muestrea primero un aleatorio r de Zq y calcula un elemento de grupo R = rG. El aleatorio actúa como una clave privada temporal, y el grupo R es su correspondiente clave pública. A continuación, una función hash H vincula el elemento de grupo R con el mensaje m para obtener un reto e = H(R, m). El último paso finaliza la firma calculando s = r ex (mod q), que

introduce una dependencia de la clave privada. La firma resultante es un par de números (e, s).

Algorithm 2.2: Schnorr Sign

Input: A private key x, and a message m
Output: A signature $\sigma = (e, s)$

1 $r \leftarrow \mathbb{Z}_q$
2 $R = rG$
3 $e = H(R, m)$
4 $s = r - ex \pmod{q}$
5 **return** (e, s)

El algoritmo 2.3 especifica el procedimiento de verificación. El algoritmo toma una clave pública X, un mensaje m y una firma σ = (e, s). Intenta reconstruir un elemento de grupo R′ a partir de la ecuación $R' = sG \oplus eX$. Si la clave pública X, el mensaje m y la firma σ son legítimos, el verificador es capaz de reconstruir el elemento original R, que fue calculado en el algoritmo de firma. Véase la siguiente prueba.

Prueba.

$$\begin{aligned} R' &= sG \oplus eX \\ &= sG \oplus e(xG) \\ &= sG \oplus (ex)G \\ &= (r - ex)G \oplus (ex)G \\ &= rG \ominus (ex)G \oplus (ex)G \\ &= rG \\ &= R \end{aligned}$$

La validez de la firma se decide por comparación de e con el hash H(R′, m).

Algorithm 2.3: Schnorr Verify

Input: A public key X, a message m, and a signature $\sigma = (e, s)$
Output: An indication of validity of signature σ

$R' = sG \oplus eX$
if $e = H(R', m)$ **then**
  return VALID
else
  return INVALID

2.1 Formulación alternativa

El esquema de firma Schnorr puede formularse de forma diferente cambiando la salida del algoritmo de firma y el procedimiento de verificación posterior. Esta formulación devolvería un par (R, s) en la última línea del algoritmo 2.2, y el procedimiento de verificación comprobaría en su lugar si sG = $R \ominus H(R, m)X$ es válido*. Efectivamente, también es válido, véase la siguiente prueba.

Prueba.

$$\begin{aligned} sG &= (r - ex)G \\ &= rG \ominus (ex)G \\ &= rG \ominus e(xG) \\ &= R \ominus eX \\ &= R \ominus H(R, m)X \end{aligned}$$

Esta versión tiene la ventaja de soportar la verificación por lotes ya que no hay operaciones de curva elíptica dentro del hash [9]. Ambas versiones son utilizadas habitualmente por los esquemas derivados [10-13]. En este trabajo, las firmas producidas por esta variante del esquema se denominan firmas Schnorr de tipo R, mientras que las firmas de la formulación anterior se denominan firmas Schnorr de tipo e.

* Cuando se utiliza esta variante, el cálculo de s suele cambiarse a s = r + ex, lo que a su vez hace que la ecuación de verificación sG = R ⊕ H(R, m)X

2.2 Firmas de Schnorr de clave prefijada

En la versión de tipo R del esquema, un atacante puede convertir una firma (R, s) verificable con la clave X en una firma (R, s aH(R, m)) para la clave aG⊕X y el mismo mensaje m, donde a Zq. La solución a este problema son las firmas Schnorr de clave prefijada [9].

Las firmas Schnorr de clave prefijada se crean mediante algoritmos similares. La única diferencia es que el reto e se calcula como e = H(R, X, m), donde X es la clave pública cuya correspondiente clave privada se utilizó para firmar el mensaje. Esencialmente, al mensaje m se le añade el prefijo X antes de pasarlo a la función hash H. Esto garantiza que el atacante no pueda seguir calculando la firma derivada debido a la propiedad unidireccional de la función hash.

Además, Bernstein [14] defiende el uso de la variante con prefijo de clave por otra razón. La mayoría de los análisis de seguridad de los esquemas de firma se centran en el problema de la falsificación de mensajes para una clave pública específica. Sin embargo, en las aplicaciones del mundo real, un atacante potencial ve muchas claves públicas, y puede estar satisfecho con falsificar un mensaje para cualquiera de ellas. Actualmente no se conocen ataques a la versión de múltiples claves más rápidos que a la versión de una sola clave. Sin embargo, Bernstein proporcionó una prueba que afirma que si el esquema de firma clásico de Schnorr es seguro en el caso de una sola clave, entonces la variante de clave prefijada es segura en el entorno de múltiples claves [14].

El coste de añadir la clave pública a la función hash es insignificante, y debido a los beneficios mencionados, la variante de clave prefijada es comúnmente utilizada por los esquemas derivados [10, 12].

2.3 Verificación en lotes

Las firmas Schnorr de tipo R admiten la verificación por lotes [9]. La verificación por lotes es un proceso de verificación de varias firmas a la vez. La verificación tiene éxito si todas las firmas verificadas son válidas. Si al menos una de las firmas no es válida, se permite que la verificación tenga éxito con una probabilidad insignificante.

Un algoritmo ingenuo de verificación por lotes para firmas Schnorr puede basarse en la siguiente observación. Para una firma Schnorr de tipo R válida, se cumple la ecuación de verificación $sG = R \oplus eX$. Si añadimos dos ecuaciones para diferentes firmas, entonces $s1G \oplus s2G = R1 \oplus e1X1 \oplus R2 \oplus e2X2$ se mantiene. Utilizando el homomorfismo del elemento de grupo por multiplicación escalar, podemos sumar primero los componentes de si, y sólo después multiplicar el generador $(s1 + s2)G = R1 \oplus e1X1 \oplus R2 \oplus e2X2$. En el caso de que la suma modular se pueda calcular más rápido que el elemento del grupo por multiplicación escalar, el proceso de verificación se ha acelerado. Mediante una mayor generalización, se puede obtener un algoritmo para un número arbitrario de firmas.

Sin embargo, este algoritmo no es seguro. Una parte maliciosa podría producir una firma no válida, que complementaría o anularía otra firma (posiblemente no válida), haciendo así que la verificación por lotes tuviera éxito incluso para firmas no válidas. Para evitar este tipo de situaciones, Wuille [9] sugirió multiplicar cada firma a punto de ser verificada por un factor aleatorio ai antes de su verificación. La aleatorización impide al atacante calcular el número exacto necesario para llevar a cabo el ataque. El algoritmo propuesto [9] se muestra en el algoritmo 2.4.

Algorithm 2.4: Schnorr BatchVerify

Input: A list of public keys $X_1, \ldots, X_n$, a list of messages $m_1, \ldots, m_n$, and a list of signatures $(R_1, s_1), \ldots, (R_n, s_n)$

Output: An indication of validity of all n signatures

$a_1 = 1$
for $i \in \{2, \ldots, n\}$ **do**
  $a_i \leftarrow Z_q$
for $i \in \{1, \ldots, n\}$ **do**
  $e_i = H(R_i, m_i)$
if $(\sum_{i \in \{1,\ldots,n\}} a_i s_i)G = (\bigoplus_{i \in \{1,\ldots,n\}} a_i R_i \oplus a_i e_i X_i)$ **then**
  return VALID
else
  return INVALID

Obsérvese que el primer coeficiente a1 se fija en 1. Esto elimina la necesidad de la multiplicación de los primeros componentes de la firma, lo que reduce reduce la sobrecarga computacional de la técnica necesaria para la prevención de terceros malintencionados.

La prevención del ataque añadió n-1 elemento de grupo adicional mediante multiplicaciones escalares. En consecuencia, el número de elementos de grupo necesarios por multiplicaciones escalares es el mismo que en el caso de la verificación individual de todas las firmas. A pesar de ello, se puede conseguir una mejora. En la forma actual de la ecuación de verificación se puede utilizar un algoritmo dedicado a la suma de multiplicaciones por elementos como el algoritmo de Pippenger [15]. Estos algoritmos calculan el resultado más rápido que la suma ingenua de elementos individuales del grupo por multiplicaciones escalares escalar, logrando así la aceleración de la verificación por lotes.

3 Construcciones basadas en firmas de Schnorr

Las firmas Schnorr tienen una rica estructura algebraica, lo que las hace un bloque de construcción adecuado para otras aplicaciones criptográficas. En el capítulo anterior, se describió un método de verificación de lotes de firmas que utilizaba una propiedad lineal de las firmas Schnorr; que una suma de firmas Schnorr también puede considerarse una firma Schnorr.

Las propiedades algebraicas de las firmas Schnorr pueden utilizarse también para construir otras primitivas criptográficas como las firmas múltiples, las firmas de umbral, las firmas ciegas o las firmas de adaptador, todas las cuales tienen interesantes aplicaciones, especialmente en las tecnologías blockchain que están surgiendo últimamente. Estas construcciones se exploran en este capítulo.

3.1 Multifirma

Los esquemas de multifirma permiten la descentralización del proceso de firma al requerir la cooperación de múltiples partes para producir una firma válida. Los cosignatarios individuales no revelan su información privada durante este proceso, lo cual es una propiedad característica del SMPC (Secure Multi-Party Computation) [16].

Se puede construir un esquema trivial de multifirma utilizando un esquema de firma digital arbitrario de la siguiente manera. Los firmantes generan sus pares de claves, firman individualmente el mensaje y concatenan sus firmas. A continuación, se procede a la verificación obteniendo todas las claves públicas de los firmantes y verificando las firmas concatenadas individualmente [10].

Es evidente que este esquema no es óptimo en muchos aspectos. La longitud de la multifirma resultante es lineal en el número de firmantes. Todas las claves públicas de los firmantes individuales tienen que ser conocidas por el verificador, con lo que se filtra su identificación. Y las firmas individuales pueden ser eliminadas de la multifirma. Todos estos problemas pueden evitarse utilizando una construcción más elaborada.

3.1.1 Propiedades del esquema multifirma

En la subsección anterior se introdujo un esquema de multifirma que produce una multifirma de longitud constante en lugar de lineal en el número de firmantes. Se dice que los esquemas que tienen esta propiedad producen firmas compactas. La ventaja de tener una multifirma de longitud constante, aparte de las implicaciones aparentes de un tamaño menor es una verificación más rápida de la multifirma. El tiempo de verificación de la multifirma puede llegar a ser incluso constante si se cumple otro requisito, descrito en el siguiente párrafo, se satisface.

Algunos esquemas multifirma presentan una propiedad de agregación de claves. Se trata de una propiedad que permite la agregación de las claves públicas de los firmantes individuales en una única clave pública común de grupo, que es suficiente para garantizar la seguridad. Esta propiedad es beneficiosa por dos razones. La primera es que el procedimiento de verificación no necesita procesar todas las claves públicas de los participantes, sino sólo la clave de grupo, lo que resulta en una computación más rápida y en menores requerimientos de memoria. La segunda razón es la mejora de la privacidad, ya que el verificador no necesita conocer la identidad (clave pública) de los firmantes individuales, sólo la clave de grupo.

Si un esquema produce firmas compactas y tiene la propiedad de agregación de claves, surge la posibilidad de la indistinguibilidad de Schnorr.

La indistinguibilidad de Schnorr requiere que la multifirma resultante tenga la forma de una firma estándar y sea verificable mediante el procedimiento estándar de verificación de Schnorr. Con un esquema de multifirma de este tipo el verificador no necesita saber que una firma determinada es una firma múltiple, ya que no se distingue de la versión estándar de firma única. Las dos propiedades mencionadas son un requisito necesario, pero no suficiente. El esquema tiene que estar especialmente construido para permitir esta funcionalidad. Otra cosa que hay que tener en cuenta a la hora de seleccionar un esquema de firma múltiple es el modelo de su uso. Los modelos múltiples ya se introdujeron en la sección anterior. Los anteriores esquemas multifirma asumían el modelo KOSK o requerían alguna interacción previa antes de permitir que los firmantes formen un grupo y firmen conjuntamente un mensaje.

Algunos de los últimos trabajos eliminaron la necesidad de rondas de comunicación preliminares utilizando el modelo KV, que requería que las claves públicas fueran acompañadas por una prueba de conocimiento. Los esquemas multifirma desarrollados recientemente intentan conseguir un funcionamiento seguro en el modelo PPK. Los esquemas adecuados para el modelo PPK no imponen ningún requisito adicional a los firmantes que no sean habituales en la criptografía de clave pública estándar [10].

Aunque existe cierta controversia en torno a la seguridad demostrable [21], tiene sus beneficios y se está convirtiendo en una norma para los nuevos protocolos criptográficos estandarizados. Por lo tanto, contar con esquemas de seguridad demostrable multifirma es cada vez más importante. Los protocolos criptográficos demostrables en el modelo estándar suelen ser demasiado ineficientes para su uso práctico [3]. Por esta razón, la mayoría de los esquemas multifirma se están probando en un modelo que proporciona una función hash idealizada idealizada necesaria para construir las pruebas, llamado modelo de oráculo aleatorio [3]. Es el mismo modelo en el que se demostró la seguridad del Schnorr; por lo tanto, su uso en los esquemas de firmas múltiples basados en Schnorr no introduce ninguna suposición adicional. Las pruebas reales suelen realizarse mediante una reducción del problema difícil subyacente, que en el ámbito de las multifirmas basadas en Schnorr es el problema del logaritmo discreto (DLP) o alguna de sus variantes. He realizado un estudio de los esquemas de multifirma publicados. De los esquemas estudiados [10-13, 17, 20, 22, 23], sólo tres, a saber, CoSi [11], Myst [13] y MuSig [12], tienen la propiedad de indistinguibilidad de Schnorr de Schnorr. Los resultados del estudio se resumen en la tabla 3.1.

Scheme	Group Key		Signature		Properties				Proof	
	Rounds	Domain	Rounds	Domain	Comp	Agg	Sch	Req	Model	Prob
MOR [20]	2	$\mathbb{G}^n$	3	$\mathbb{G} \times \mathbb{Z}_q$	✓	✗	✗	DKPG	ROM	DL
BN [10]	0	$\mathbb{G}^n$	3	$\mathbb{G} \times \mathbb{Z}_q$	✓	✗	✗	PPK	ROM	DL
BCJ1 [17]	0	$\mathbb{G}^n$	2	$\mathbb{G} \times \mathbb{Z}_q^3$	✓	✗	✗	PPK	—	—
BCJ2 [17]	0	$\mathbb{G}$	2	$\mathbb{G}^2 \times \mathbb{Z}_q^3$	✓	✓	✗	KV	—	—
MWLD [23]	0	$\mathbb{G}^n$	2	$\mathbb{Z}_q^3$	✓	✗	✗	PPK	—	—
CoSi [11]	0	$\mathbb{G}$	2	$\mathbb{Z}_q \times \mathbb{Z}_q$	✓	✓	✓	KOSK	—	—
mBCJ [22]	0	$\mathbb{G}$	2	$\mathbb{G}^2 \times \mathbb{Z}_q^3$	✓	✓	✗	KV	ROM	DL
MuSig [12]	0	$\mathbb{G}$	3	$\mathbb{G} \times \mathbb{Z}_q$	✓	✓	✓	PPK	ROM	DL
Myst [13]	2	$\mathbb{G}$	2	$\mathbb{Z}_q \times \mathbb{Z}_q$	✓	✓	✓	DKPG	ROM	OMDL

Tabla 3.1: La tabla es un resumen de las propiedades de los esquemas multifirma basados en el problema del logaritmo discreto. La primera columna contiene el nombre del esquema. Las dos columnas siguientes indican el número de rondas de comunicación necesarias para realizar el cálculo de la clave de grupo y el dominio de la clave de grupo, respectivamente. Del mismo modo, las dos columnas siguientes indican el número de rondas de comunicación necesarias para crear una firma múltiple y su dominio. Las cuatro columnas siguientes describen las propiedades de los esquemas: si el esquema produce una firma compacta, si el esquema tiene la propiedad de agregación de claves, si el esquema tiene la propiedad de indistinguibilidad de Schnorr, y qué modelo de uso requiere el esquema. Las dos últimas columnas indican si el esquema tiene una prueba de seguridad y, si la tiene, en qué modelo y cuál es su problema subyacente.

3.1.2 Esquema MuSig

MuSig es un esquema multifirma diseñado por Maxwell et al. [12]. El esquema fue diseñado originalmente para operar en dos rondas de comunicación, pero Drijvers et al. [22] señaló que tal operación era insegura. En respuesta a su hallazgo, los autores de MuSig presentaron otra versión del esquema, que conserva sus propiedades únicas pero realiza tres rondas de comunicación durante la fase de firma en lugar de dos. La modificación permitió demostrar que el esquema es seguro en el modelo de oráculo aleatorio a costa de aumentar la comunicación entre los firmantes [12]. La versión de tres rondas se describe enesta sección.

Además de la notación habitual, las descripciones también distinguen entre las funciones hash utilizadas. En particular, Hcom se utiliza para calcular los compromisos, Hagg se utiliza para calcular los coeficientes en la fase de agregación de la clave, y Hsig se utiliza para calcular un desafío en la fase de la clave. En la fase de agregación de claves, y Hsig se utiliza para calcular un reto en el procedimiento de firma. En la práctica, las diferentes funciones hash pueden ser realizadas como variantes etiquetadas de una única función hash. El algoritmo de generación de pares de claves 3.1 es el mismo que el algoritmo estándar de generación de pares de claves Schnorr 2.1. Es decir, un número secreto aleatorio xi se extrae de Zq, y el generador de grupos G se multiplica por este número secreto para obtener una función por este número secreto para obtener Xi = xiG. El par de claves par de claves es (xi,Xi).

Obsérvese que no hay interacción entre los firmantes durante la generación del par de claves. Es uno de los requisitos de diseño cruciales necesarios para obtener un esquema multifirma que permita un funcionamiento seguro en el modelo PPK [12].

Algorithm 3.1: MuSig Gen

Output: A pair of private and public key (x_i, X_i)

1 $x_i \leftarrow \mathbb{Z}_q$
2 $X_i = x_i G$
3 **return** (x_i, X_i)

La protección contra el ataque de la clave falsa se consigue mediante una nueva técnica utilizada en el algoritmo de agregación de claves 3.2. En lugar de calcular la clave de grupo como una mera suma de las claves públicas de los firmantes, cada clave se multiplica antes por un factor ai que depende del conjunto de claves públicas de los firmantes y también de la propia clave multiplicada. De este modo,

ya no es posible realizar el ataque de la clave falsa, ya que cambiar cualquier clave da lugar a un cambio imprevisible de la clave final del grupo debido al efecto de avalancha de la función hash criptográfica.

Algorithm 3.2: MuSig Agg

Input: A multi-set of public keys $L = \{X_1, \ldots, X_n\}$
Output: The group public key X

1 **for** $i \in \{1, \ldots, n\}$ **do**
2 $\quad a_i = H_{agg}(L, X_i)$
3 $X = \bigoplus_{i=1}^{n} a_i X_i$
4 **return** X

El algoritmo de firma multipartita 3.3 refleja la construcción de claves construcción de la clave. Cada firmante i inicia el algoritmo con un conjunto múltiple de claves públicas L = {X1, . . . , Xn} y su clave privada xi. El algoritmo calcula entonces la ai del firmante específico y la clave agregada del grupo X = Agg(L). Los participantes muestrean un nonce ri de Zq y multiplican el generador de grupo por este número para obtener Ri = riG. Una vez que tienen Ri cada firmante calcula su compromiso ti=Hcom(Ri). Los compromisos se distribuyen a las demás partes. Al recibir todos los compromisos tj los firmantes distribuyen los suyos Ri a las demás partes. Cuando se reciben todos los Rj, cada firmante comprueba si los valores recibidos son son coherentes con los compromisos.

En caso de que se encuentre una incoherencia, el proceso debe ser abortado. En caso contrario, el algoritmo continúa sumando los Rj para obtener el elemento nonce del grupo R . Ahora los firmantes tienen que calcular e = Hsig(X, R, m) y sus firmas parciales si = ri + eai*xi (mod q). A continuación, las firmas parciales se distribuidas a otras partes, y los participantes calculan s . La multifirma resultante es el par (R,s).

Algorithm 3.3: MuSig Sign

Input: A message to be signed m, a multi-set of public keys $L = \{X_1, \ldots, X_n\}$, and a private key x_i

Output: A signature $\sigma = (R, s)$

$a_i = H_{agg}(\langle L \rangle, X_i)$
$X = \mathsf{Agg}(L)$
$r_i \leftarrow \mathbb{Z}_q$
$R_i = r_i G$
$t_i = H_{com}(R_i)$
send t_i to other parties
wait for t_i from other parties
send R_i to other parties
wait for R_i from other parties
if $H_{com}(R_i) \neq t_i$ *for any* i **then**
  abort
$R = \bigoplus_{i=1}^{n} R_i$
$e = H_{sig}(X, R, m)$
$s_i = r_i + e a_i x_i \pmod{q}$
send s_i to other parties
wait for s_i from other parties
$s = \sum_{i=1}^{n} s_i \pmod{q}$
return (R, s)

El algoritmo de verificación 3.4 es el mismo que el algoritmo de verificación de una firma Schnorr estándar de tipo R. Por lo tanto, el esquema tiene la propiedad de indistinguibilidad de Schnorr. Dado un mensaje m, una clave pública X y una firma σ = (e,s), el verificador calcula e =Hsig(X, R, m) y comprueba si la ecuación sG = R ⊕ eX se cumple. En caso de que si se cumpla, la firma es válida; en caso contrario, no lo es.

Algorithm 3.4: MuSig Verify

Input: A message m, a group public key X, and a signature $\sigma = (R, s)$

Output: An indication of validity of signature σ

$e = H_{sig}(X, R, m)$
if $sG = R \oplus eX$ **then**
  return VALID
else
  return INVALID

Como se mencionó al principio de esta sección, la seguridad de MuSig de tres rondas se demostró en el modelo de oráculo aleatorio. La prueba se construyó mediante una reducción del problema subyacente del logaritmo discreto.

3.2 Firmas Treshold

Al igual que los esquemas de firmas múltiples, los esquemas de firmas de umbral permiten la descentralización del proceso de firma al requerir que varias partes cooperen para producir una firma válida. Sin embargo, en este caso, la cooperación de todas las partes interesadas no es necesaria.

Un subconjunto de al menos t firmantes de un conjunto de n firmantes es suficiente para crear una firma umbral, pero ningún subconjunto de tamaño inferior a t es capaz de producir una firma válida. Dicha firma umbral se denomina firma t-de-n de n. Las firmas umbral pueden considerarse una generalización de las multifirmas, que son un caso especial de firma n-de-n.

Las firmas Schnorr admiten la generación de claves distribuidas (DKG) [24], con la que se pueden construir esquemas de firma umbral interactivos. Desgraciadamente, hasta ahora, la utilidad de estos esquemas sigue siendo limitada. La mayoría de los esquemas actuales sólo han demostrado ser seguros para t < n/2 en un entorno no concurrente, o requieren un mecanismo de difusión fiable. [9]

Los esquemas de firma de umbral basados en Schnorr se basan en el procedimiento DKG para dos propósitos. El primero es la generación inicial de la clave de grupo y las acciones privadas correspondientes. El segundo es la generación del elemento nonce R y los correspondientes nonces compartidos ri al crear una firma.

Últimamente, hasta donde yo sé, no se han producido muchos avances en el área de los esquemas de firma de umbral basados en Schnorr. Dos notables esquemas de firma de umbral basados en Schnorr son los esquemas descritos por Stinson y Strobl [25], y Gennaro et al. [26].

3.3 Firma de adaptadores

Las firmas adaptadoras son un concepto introducido por Poelstra como parte de soluciones de scripts sin guión para blockchains compatibles con Schnorr. Los scripts sin guión son una técnica para hacer cumplir la semántica de algunos contratos inteligentes sin necesidad de utilizar scripts en las aplicaciones de blockchain [27]. Dada una firma Schnorr de tipo R (R,s), una firma adaptadora es una tripleta (R,s',T), donde $t \in Zq$ es un valor del adaptador, $T = tG$ es un elemento adaptador, y $s' = s + t \pmod q$ es una firma ajustada [28]. Las siguientes ecuaciones, en las que $e = H(R, m)$ es el reto hash y X es una clave pública, son válidas.

$$sG = R + eX \qquad (3.1)$$

$$s'G = T + R + eX \qquad (3.2)$$

La firma del adaptador no es una firma Schnorr válida, ya que $(R \oplus T,s')$ no satisface la ecuación 3.1 debido a que el elemento del adaptador no se considera en el cálculo del hash e. Sin embargo, se puede verificar que es una firma adaptadora consistente utilizando la ecuación 3.2. Si tenemos una firma Schnorr (R,s) y una firma adaptadora correspondiente correspondiente (R,s′T), el logaritmo discreto de T puede calcularse restando s de s′. Véase la siguiente demostración, que comienza restando 3.1 de 3.2.

Prueba:

$s'G - sG = T + R + eX - R - eX$ (3.2) - (3.1)

$s'G - sG = T$

$(s' - s)G = tG$

$s' - s = t \pmod q$

Alternativamente, si tenemos una firma del adaptador (R,s ′ ,T) y el valor del adaptador t, podemos calcular la firma original de Schnorr (R,s) por sustracción modular del valor del adaptador de la firma ajustada s ′ , lo que se deduce directamente de la definición de la firma del adaptador. Por lo tanto, dada una firma adaptadora (R,s ′ , T), el conocimiento de la correspondiente firma Schnorr es equivalente al conocimiento del logaritmo discreto de T [28]. En combinación con las multifirmas, esta propiedad se utiliza en las soluciones, que se describen en el capítulo 4.

3.4 Firmas ciegas

r ∈ Zq y envía el nonce público R′ = r′G al usuario. El usuario elige dos números aleatorios α, β ∈ Zq y ciega el nonce público R = R′ ⊕ αG ⊖ βX, donde X es la clave pública del firmante. A continuación, el usuario calcula un desafío ciego e‘= H(R, m) + β mod q y lo envía al servidor. El servidor firma este reto y devuelve s′ = r + e′x. Esta firma ciega se puede descifrar calculando s = s′ + α y e=e′ - β. La (e,s) es entonces una firma Schnorr válida para el mensaje m verificable con la clave pública X. Alternativamente, (R,s) puede salir, en caso de que se necesite una firma de tipo R. La siguiente prueba muestra que el componente s construido utilizando el servidor de firma es efectivamente una firma válida para la clave pública X.

Prueba:

s = s ′ + α

= r ′ − (H(R ′ ⊕ αG ⊕ βX, m) + β)x + α

= r ′ − H(R ′ ⊕ αG ⊕ βX, m)x − βx + α

= (r ′ + α − βx) − H(R ′ ⊕ α ⊕ βX, m)x

= r − ex

Algorithm 3.5: Schnorr Blind Signing Server

1 **wait** for signing request
2 $r' \leftarrow \mathbb{Z}_q$
3 $R = r'G$
4 **send** R' to user
5 **wait** for e'
6 $s' = r - e'x$
7 **send** s' to user

Algorithm 3.6: Schnorr Blind Signing User

send signing request
wait for R'
$\alpha \leftarrow \mathbb{Z}_q$
$\beta \leftarrow \mathbb{Z}_q$
$R = R' \oplus \alpha G \ominus \beta X$
$e' = H(R, m) + \beta \mod q$
send e' to server
wait for s'
$s = s' + \alpha$
$e = e' - \beta$
return (e, s)

Schnorr también proporcionó una prueba de seguridad contra la falsificación de un signo más del esquema en el ROM y el modelo de grupo genérico (GGM) bajo el supuesto de la intratabilidad de los problemas OMDL (One More Discrete Logarithm) y ROS (Random inhomogeneities in an Overdetermined, Solvable system of linear equations) [31]. Más tarde, Wagner [32] ha demostrado que el problema ROS es resoluble en tiempo subexponencial. Fuchsbauer et al. [30] ha presentado la prueba de seguridad en un entorno diferente, donde utilizó el modelo de grupo algebraico (AGM) en lugar del GGM. También sugirió una modificación del esquema, que evita la dependencia del problema ROS, solucionable subexponencialmente.

La modificación requiere que el firmante abra al menos dos sesiones de firma con el firmante. El firmante completa sólo una de las firmas y falla la otra mediante una elección aleatoria. La aleatoriedad impide que el atacante emplee el algoritmo de Wagner, ya que para utilizarlo con éxito, tendría que adivinar correctamente cuál de las consultas de firma tendrá éxito. Como el ataque de Wagner funciona sólo para un gran número l de sesiones paralelas, la aleatorización hace que sea exponencialmente inviable [30].

Esta modificación ha demostrado ser segura contra un ataque de ataque de falsificación en los modelos AGM+ROM bajo el supuesto de la intractabilidad del problema OMDL y del problema ROS modificado, que ellos conjeturan que es difícil [30].

4 Bitcoin

Bitcoin es un sistema descentralizado entre pares diseñado para servir como una moneda digital o, más exactamente, una criptomoneda [33], descrita por Satoshi Nakamoto en su libro blanco [1]. Fue la primera criptomoneda ampliamente adoptada, que sigue gozando de una creciente popularidad. Por ejemplo, en el momento que empecé este proyecto, un solo BTC valía aproximadamente 40.000 USD.

Los diseños de monedas digitales anteriores tenían problemas para resolver el problema del doble gasto. El doble gasto significa que una parte maliciosa podría crear dos transacciones inconsistentes que transfieran los mismos fondos a dos partes diferentes. Para resolver este problema, los diseñadores de monedas digitales a menudo recurren al uso de una autoridad centralizada, que compensaría las transacciones, lo que daría lugar a un sistema con un único punto de fallo [33].

Nakamoto [1] propuso una solución novedosa para el problema del doble gasto problema del doble gasto. Sugirió utilizar una red descentralizada de nodos, que comprobaría las transacciones suministradas según las reglas de consenso y registrarlas en un libro de contabilidad público de manera que fuera prácticamente inviable cambiar el registro más tarde. Las modificaciones en el libro de contabilidad sólo son apéndices añadidos a través de un concurso de nodos de verificación llamados mineros. Los mineros intentan resolver un rompecabezas criptográfico, que requiere el empleo de una cantidad significativa de poder computacional, por lo que sirviendo así de prueba de trabajo [34, 35]. Si un nodo minero consigue resolver el rompecabezas, difunde su solución. Los demás mineros suelen aceptar esta solución y comienzan a resolver otro problema sobre la solución recién descubierta.

En Bitcoin, la solución al rompecabezas es un bloque específico. El bloque es una estructura de datos que contiene una colección de transacciones que se añaden al libro mayor, un identificador único del bloque anterior, un nonce, y otra información. La dependencia entre los bloques forma una cadena. Pueden existir varias cadenas, pero sólo la de más esfuerzo computacional de cálculo colectivo, la más larga, se considera válida.

En caso de que varios nodos encuentren una solución al rompecabezas aproximadamente al mismo tiempo, la cadena puede bifurcarse. La cadena bifurcada da lugar a un estado incierto de blockchain, en el que existen múltiples cadenas más largas.

Una parte de los mineros puede considerar un bloque como el más reciente, mientras que otra parte de los mineros uno diferente. Sin embargo, algunos de los mineros eventualmente crean un siguiente bloque encima de uno de ellos, creando así la cadena más larga, a la que la red se adapta [33].

Para duplicar el gasto en un entorno de este tipo, la parte maliciosa tendría que bifurcar la cadena y crear una nueva cadena más larga. Este ataque requeriría una enorme capacidad de cálculo, pero podría ser posible (el llamado ataque del 51% [36]). Sin embargo, cuantos más bloques sean construidos sobre el que contiene la transacción que el atacante quiere gastar dos veces, más difícil será el ataque desde el punto de vista computacional. Por lo tanto, se aconseja a los receptores de pagos que esperen a que se creen unos cuantos bloques antes de confiar en una transacción.

Este mecanismo de prevención del doble gasto sería insostenible sin un incentivo para que los nodos mineros consuman sus recursos en la resolución del puzzle. Cada vez que un nodo minero consigue resolver el rompecabezas, el nodo puede añadir una transacción especial de coinbase que crea nuevas monedas. El resultado de esta transacción sirve como recompensa al minero y es la oferta monetaria del sistema. Aparte de la transacción coinbase, los mineros cobran comisiones por las transacciones. Las tasas se entregan a los nodos mineros como recompensa por los usuarios del sistema, que desean que sus transacciones se añadan al libro mayor. La cantidad de dinero que un minero puede adquirir de la transacción de coinbase está disminuyendo. Cada 210.000 bloques, la cifra se reduce a la mitad, y una vez 6,93 millones de bloques se minan, el suministro de dinero coinbase disminuye por completo. En ese momento, las comisiones se convierten en el principal incentivo para los mineros [33].

Una transacción de Bitcoin consta de entradas y salidas. Las salidas determinan a qué dirección se transfiere el dinero. Las entradas corresponden a salidas aún no gastadas de transacciones anteriores. Una vez que la transacción se publica en la cadena de bloques, las entradas se gastan, y las salidas pueden utilizarse posteriormente como entradas no gastadas para nuevas transacciones.

La dirección es una transformación única de una información secreta, que sirve como prueba de propiedad. Cualquiera que desee gastar fondos bloqueados a una determinada dirección necesita proporcionar una prueba utilizando esta información secreta. Las pruebas de propiedad son en su mayoría basadas en firmas digitales. En este caso, la dirección es un valor vinculado a una clave pública, y para gastar los fondos bloqueados a ella, el gastador debe proporcionar una firma digital de la transacción dada con la correspondiente clave privada.

Los mineros comprueban la validez de las pruebas antes de añadir una transacción a un bloque. Sin embargo, no sólo comprueban las pruebas. También comprueban otras propiedades de las transacciones como que la suma de salidas de una transacción debe ser menor o igual que la suma de sus entradas, o que las entradas deben estar sin gastar en la cadena de bloques. El conjunto de todas las reglas se llama consenso. Es crucial que todos los nodos de la red tengan las mismas reglas de consenso; de lo contrario, los nodos se forzarían y crearían blockchains divergentes [33].

La tecnología de firma digital utilizada en Bitcoin es ECDSA [6] (Algoritmo de Firma Digital de Curva Elíptica) sobre una curva elíptica secp256k1, definida en [37]. Últimamente, la comunidad de Bitcoin empezó a investigar las firmas Schnorr como una alternativa al menos flexible ECDSA [9]. Las sugerencias actuales de adopción de Schnorr y sus beneficios se discuten en la siguiente sección. La siguiente sección utiliza construcciones del capítulo 3 para presentar algunas de sus aplicaciones a Bitcoin.

4.1 Adopción de firmas de Schnorr

El esquema de firma Schnorr se basa en supuestos y operaciones criptográficas similares a las de ECDSA, lo que permite que el esquema sea adaptado para reutilizar una parte importante del código actual. Schnorr puede adaptarse para utilizar la curva elíptica de Bitcoin secp256k1. Del mismo modo, la función hash utilizada en el algoritmo de firma Schnorr puede ser instanciada por SHA256 (Secure Hash Algorithm), que también se utiliza actualmente en Bitcoin. Por lo tanto, para cambiar a firmas Schnorr, no es necesario introducir primitivas criptográficas adicionales en el sistema. Además, los mecanismos de generación de pares de claves pueden seguir siendo los mismos, lo que es importante para soluciones como los monederos HD (Hierarchical Deterministic) [38]. Wuille et al. propusieron la BIP 340 (Propuesta de Mejora de Bitcoin) [9] como una especificación de firmas Schnorr aplicadas a la curva secp256k1, a efectos de las transacciones de Bitcoin. Su sugerencia aporta mejoras adicionales al sistema Bitcoin, que no son específicas de las firmas Schnorr, pero que pueden acoplarse a la actualización. Esas mejoras son:

· Codificación de firmas de 64 bytes;

· Codificación de puntos de curva elíptica de 32 bytes;

· Verificación por lotes.

La codificación de la firma de 64 bytes mejora la codificación de longitud. Para codificar un punto de curva elíptica, se utiliza codificación implícita de coordenadas y, lo que resulta en una reducción de tamaño de un byte. Seleccionan el tipo de firmas, que pueden utilizarse para realizar verificación por lotes, a diferencia de la versión de ECDSA utilizada en Bitcoin.

La adopción real de las firmas Schnorr en Bitcoin requeriría cambios en el consenso. Actualmente, existen múltiples sugerencias y una propuesta publicada [39, 40].

La forma más directa de añadir soporte de firma Schnorr a Bitcoin es la simple aceptación de las firmas Schnorr como un método de firma. En este caso, el principal reto a resolver es cómo diferenciar entre los métodos de firma. Una posible solución es añadir otro código de operación al script de Bitcoin, que determinaría el tipo de firma. Alternativamente, el tipo de firma podría determinarse implícitamente, por ejemplo, por la longitud de la firma o intentando verificar ambas opciones. Sin embargo, esto último supondría una ralentización computacional y podría plantear problemas de seguridad.

Independientemente del método de diferenciación de firmas seleccionado, esta solución sólo requeriría una modificación mínima de las reglas de consenso. Sin embargo, Maxwell presentó dos ideas intrigantes [41,42], que se están considerando. Las ideas combinan las firmas Schnorr con mejoras adicionales de Bitcoin, para aumentar la eficiencia y la privacidad de la cadena de bloques.

La primera de esas ideas se llama Taproot [41]. Taproot se basa en un esquema implícito de firmas múltiples y en MAST (Merkelized Abstract Syntax Tree) para representar diferentes condiciones de gasto.

Cada escritura de Bitcoin puede representarse como una disyunción lógica de condiciones individuales de gasto. Cuando se gasta utilizando una determinada condición, sólo el script correspondiente necesita ser visible para los nodos verificadores. Las demás condiciones de gasto pueden permanecer ocultas, aumentando así la privacidad. La ocultación de las condiciones de gasto no utilizadas puede ser realizada por el MAST. Representan las condiciones de gasto individuales como nodos de un árbol de Merkle, cuya raíz está codificada en una dirección. Para gastar desde la dirección, el gastador tiene que proporcionar una ruta de hashes al script utilizado, el propio script y los datos que lo satisfacen.

La idea principal de Taproot se basa en la suposición de que la mayoría de las transacciones contienen una regla de gasto que permite su liquidación en caso de que todas las partes interesadas estén de acuerdo. Si la suposición es correcta, el caso de uso más común del convenio colectivo puede combinarse con las vías de gasto alternativas de forma eficiente a través de una técnica similar al concepto de pago por contrato [43].

Para crear una dirección Taproot, las partes generan una clave pública X (multifirma) y un MAST de sus rutas de gasto alternativas. Toman la raíz t de dicho árbol y compensan su clave pública X con este valor, obteniendo X′ = X + tG, que utilizan como dirección. El gasto de una dirección Taproot puede lograrse de dos maneras. O bien proporcionando una firma para X′ lo que significa crear una firma para la colectiva de la clave privada x compensada por t; o revelando la clave pública original y la ruta de gasto en el MAST a la condición de gasto deseada, y proporcionando la prueba que satisface la condición de gasto dada.

Taproot fue formalizado en un BIP 341 [39] por Wuille et al. Los autores afirman que su solución daría lugar a una comunidad de Bitcoin menos fracturada, en la que todo el mundo apoya esta propuesta por completo o no la apoyan en absoluto. En caso de adopción gradual de los mismos cambios, la comunidad sería más diversa, lo que simplificaría la huella de las transacciones [39].

La segunda idea de Maxwell se llama Graftroot [42]. Amplía la hipótesis de Taproot. En particular, el gasto de una dirección determinada puede ser causado por el acuerdo colectivo de todas las partes interesadas o utilizando condiciones de gasto alternativas, todas previamente acordadas.

Las direcciones de Graftroot consisten en una clave pública (multipartita) generada por las partes implicadas. De nuevo, la forma predeterminada de gastar desde una dirección es proporcionar una firma verificable por la clave pública, exactamente como en Taproot. Las condiciones de gasto alternativas se representarían como scripts de Bitcoin simples complementados con una firma de su contenido verificable por la misma clave pública. Al invocar una condición de gasto alternativa, que todas las partes interesadas han acordado previamente, el gastador enviaría el script, su firma y los datos de prueba.

La principal ventaja de este método es su escalabilidad. Se puede crear cualquier número de condiciones de gasto alternativas sin ninguna sobrecarga adicional en la cadena. Además, las condiciones de gasto alternativas se pueden crear incluso después de que la transacción de financiación se haya añadido a la cadena de bloques.

Estos beneficios se consiguen con una comunicación interactiva. Para crear condiciones de gasto, todas las partes interesadas tienen que cooperar. Otro coste es la sobrecarga de tamaño. Con cada condición de gasto alternativa, su firma debe ser publicada, lo que requiere añadir más bytes a la blockchain que la condición de gasto alternativa de la capa superior de Taproot. Por otro lado, si el gasto de Taproot utiliza una estructura de árbol más profunda, la ruta Merkle puede superar la sobrecarga de Graftroot y ser más costoso.

Hasta ahora, la solución más descrita y desarrollada es Taproot. Ya se han publicado sus correspondientes PBI con detalles de implementación elaborados. Sin embargo, en este momento, sólo podemos especular qué solución será aceptada, o si alguna de ellas lo será.

4.2 Aplicaciones en Bitcoin

Hasta ahora, hemos discutido varios aspectos de las firmas Schnorr, los fundamentos de Bitcoin, y hemos sugerido formas de adopción de la firma Schnorr a Bitcoin. En esta sección, examinamos algunas aplicaciones habilitadas por la adopción de las firmas Schnorr en Bitcoin. Hay que tener en cuenta que la mayoría de las aplicaciones pueden realizarse incluso sin soporte de firma Schnorr; sin embargo, en general, las soluciones basadas en Schnorr son más escalables y privadas.

4.2.1 Monederos multifirma

Los monederos multifirma permiten la descentralización de las transacciones delegando la creación de firmas en múltiples participantes, cada uno de los cuales posee una clave privada diferente. Los participantes no necesariamente tienen que corresponder a personas individuales. Por ejemplo, pueden ser diferentes dispositivos de firma de la misma persona, que los utiliza como método de autenticación multifactorial.

Ya existen varios monederos con múltiples firmas, pero en su mayoría se basan en el uso del script de Bitcoin para proporcionar la multifirma. Debido a esto, no se escalan bien, ya que requieren una firma estándar para cada cosignatario, lo que resulta en una dependencia lineal en el tamaño de la multifirma. Además, el número de participantes y sus claves públicas son visibles en la cadena de bloques, lo que proporciona pistas para huellas dactilares, y reduce la privacidad.

Lindell [44] inventó un algoritmo para la creación de firmas múltiples de dos partes utilizando ECDSA. El algoritmo podría utilizarse para construir un monedero multifirma con beneficios similares a los de la multifirma basada en Schnorr. Sin embargo, un monedero de este tipo se no escalaría más allá de las dos partes.

En la sección 3.1, mencionamos varios esquemas de multifirma compatibles con las firmas Schnorr, que podrían utilizarse para la construcción de un monedero multifirma. Cada esquema tiene sus ventajas y desventajas, lo que hace que algunos de ellos sean más adecuados en diferentes aplicaciones. Sin embargo, todos comparten las ventajas de la escalabilidad, tamaño de la firma, y privacidad sobre la solución basada en scripts.

4.2.2 Intercambio atómico (atomic swap)

Un swap atómico es una técnica de intercambio de criptodivisas sin la necesidad de un tercero de confianza, de forma atómica. Es decir, si una parte recibe las monedas, la otra también lo hace. Puede utilizarse para el intercambio de diferentes criptodivisas o para el intercambio de Bitcoin con fines de privacidad.

La solución actual de los swaps atómicos implica el HTLC (Hash Time Locked Contract) [45]. Se trata de un tipo especial de escritura de Bitcoin que utiliza dos vías de gasto distintas: una requiere proporcionar una imagen previa a una función hash, y la otra sólo puede utilizarse después de que haya pasado un tiempo determinado. La primera se utiliza para realizar el intercambio atómico, y la segunda es una lógica de reembolso en caso de que una de las partes no coopere.

En la solución basada en HTLC [45], el intercambio atómico se realiza mediante una parte (Alice) comprometiendo los fondos que quiere transferir a una segunda parte (Bob) en su blockchain. Ella conoce la preimagen del hash utilizado en el HTLC, pero Bob no. Bob compromete sus fondos para Alice en su cadena utilizando HTLC con la misma imagen hash que utilizó Alice. Alice puede retirar monedas de la transacción de Bob ya que conoce la preimagen. Una vez que ella hace eso, Bob llega a conocer la imagen previa y puede retirar las monedas de Alice.

Como hemos comentado en la sección 3.3, el conocimiento de una firma de adaptador y la correspondiente firma Schnorr es equivalente al conocimiento de la firma del adaptador y el offset t. En un entorno Schnorr , esto puede utilizarse para sustituir la parte hash de HTLC para realizar un intercambio atómico de la siguiente manera [28]. Sin embargo, hay que tener en cuenta que la lógica de reembolso todavía requiere el uso de un script.

En primer lugar, ambas partes comprometen sus fondos intercambiados en su dirección común multifirma. Acuerdan puntos colectivos de nonce R1, R2 para las firmas. Alice elige un valor aleatorio t y calcula un punto correspondiente T = tG. A continuación, produce un adaptador parcial (R1,s'1A, T),(R2,s′2A, T) para las transacciones y las envía a Bob. Puede hacerlo ya que no son firmas Schnorr válidas; por lo tanto, no pueden ser publicadas en la cadena de bloques. Bob necesita verificar que ambas firmas del adaptador son coherentes y que utilizan el mismo T. Si la verificación tiene éxito, puede producir su firma parcial para la transacción que transfiere monedas de él a Alice (R2,s2B), y enviarle la transacción.

Una vez que reciba la parte parcial, puede completar la multifirma (R2,s2A + s2B) y publicarla. Una vez que la transacción llega a la blockchain, Bob se entera del valor t = s′2A - (s2A + s2B) - s2B, y puede corregir la firma del adaptador de Alice y completar la multifirma de la transacción que transfiere las monedas de Alice a Bob (R1,s′1A - t + s1B).

El alcance de los beneficios del intercambio atómico basado en Schnorr sobre el basado en HTLC depende del método de adopción de Schnorr elegido, porque sería desconsiderado comprometer fondos en la blockchain sin ninguna lógica de reembolso, y la lógica de reembolso tiene que lograrse a través de un script. En el caso de Taproot o Graftroot, un intercambio atómico exitoso en blockchain es indistinguible de una transacción estándar, con lo que se obtienen beneficios de privacidad y la reducción de la huella de blockchain. Cuando se utiliza la lógica de reembolso, el script tiene que ser publicado, lo que es observable en la blockchain, pero sigue teniendo una huella menor. Si se toma el camino de adopción de Schnorr el tamaño del script se reduce en comparación con la solución basada en HTLC, porque no se utiliza la imagen hash, pero la ganancia de privacidad desaparece.

4.2.3 Canales de pago multisalto

Con el crecimiento del uso de Bitcoin, sus problemas de escalabilidad se hacen evidentes [46]. Estos problemas han llevado a la aparición de varios protocolos para mantener las transacciones fuera de la blockchain de Bitcoin; no tener todos los nodos de la red para verificarlas, pero manteniendo garantías similares de seguridad y privacidad. Se basan en la noción de canales de pago. Un canal de pago es un canal virtual a través del cual los fondos pueden fluir entre dos partes sin necesidad de comprometer cada una de las transacciones en la cadena de bloques. Se han presentado múltiples diseños de canales de pago [45, 47, 48]. Los diseños difieren en las capacidades de los canales de pago y las primitivas en las que se basan.

Los canales suelen iniciarse con dos partes que depositan algunos de sus fondos en una dirección multifirma de dos partes. Esta combinación de depósito de ambas partes es un límite superior de los fondos transferidos a través de ese canal. Inicialmente, las partes acuerdan una transacción que devuelve los fondos depositados a sus propietarios mediante su firma. La idea clave de los canales de pago es que no tienen que publicar esta transacción inmediatamente. Pueden mantener esta transacción fuera de blockchain como un crédito, con el conocimiento de que pueden ejecutarla en cualquier momento. Si una parte desea retirar los fondos, la parte puede publicar independientemente la transacción y cerrar el canal. Hasta que el canal se cierre, las partes pueden realizar pagos dentro del canal ajustando su saldo. El saldo se ajusta creando y firmando otra transacción con con el nuevo estado del canal acordado e invalidando el anterior.

Sin embargo, los canales de pago sólo pueden transferir fondos entre dos partes. Abrir demasiados canales es caro, ya que requiere comprometer fondos en el canal, por lo que esta solución no es escalable. Para resolver este problema, los canales de pago se combinan en un gráfico que forma una red de canales de pago. La red de canales de pago permite transferir fondos entre partes que no están

directamente conectadas a través de nodos intermedios. Los nodos intermedios transmiten los fondos en un camino formado por un pago determinado. Sin embargo, debe realizarse con precaución, ya que puede provocar el robo de fondos en caso de que alguna de las partes decida no retransmitir su parte. Por lo tanto las transferencias individuales deben tener éxito sólo si todas ellas lo hacen.

La solución original utilizada en Lightning Network [45] (la primera propuesta de red de canales de pago) se basaba en HTLC, al igual que los intercambios atómicos. Alice, que quiere pagar a Bob a través de un canal de pago, se lo comunica primero a Bob. Bob genera entonces un número aleatorio t como preimagen de una función hash H resistente a las colisiones. Calcula H(t) y la envía a Alice. Alice inicia la cadena de pagos creando un contrato HTLC comprometido con el valor H(t) de Bob con el siguiente nodo de la ruta de pago. Los siguientes nodos de la ruta actúan de forma similar. Al recibir el contrato de la parte anterior, cada uno de ellos crea un HTLC con la parte siguiente. Una vez que la cadena HTLC llega a Bob, éste puede cerrar el contrato y retirar sus monedas del contrato de la parte anterior. Al hacerlo, publicará el valor t en la cadena de bloques y las partes en la ruta de pago pueden empezar a retirar las suyas también. Alternativamente, si desean que los canales se mantengan abiertos, Bob puede revelar el valor t en la dirección opuesta de la de pago, y cada parte puede verificar que realmente es la preimagen de H(t); de este modo, su transacción puede ser ejecutada en la blockchain.

5 Conclusión

El objetivo de este proyecto era investigar los protocolos multipartitos implícitos basados en Schnorr y sus beneficios para el ecosistema Bitcoin. Estudiamos cuatro tipos de protocolos multipartitos implícitos basados en Schnor; las firmas múltiples, las firmas de umbral, las firmas de adaptador y las firmas ciegas. Aunque la mayoría de sus aplicaciones son realizables incluso sin soporte de firma Schnorr usando Bitcoin, su uso aporta ventajas significativas sobre la solución basada en el script. En general, la solución implícita presenta un menor tamaño de transacción y una mayor privacidad. La primera es el resultado de que el contrato sea implícito en una firma. La segunda se consigue al no tener que registrar los datos de la escritura en la blockchain.

Otros beneficios, no necesariamente exclusivos de las firmas Schnorr, dependen de la forma real de adopción de la firma Schnorr a Bitcoin. La especificación propuesta BIP 340 mejora la actualmente utilizada ECDSA mediante el uso de una codificación de firma de longitud fija y una codificación de puntos de curva elíptica más corta, y el soporte de la verificación por lotes. Además, las ideas Taproot y Graftroot hacen que la coexistencia de script y opciones de gasto de firma sean más concisa y privada, porque las unen en un solo tipo de dirección y ocultan las vías alternativas de gasto, a menos que se utilicen realmente.

6 Bibliografía

[1] Satoshi Nakamoto et al. Bitcoin: A Peer-to-Peer Electronic Cash System 2008.

[2] Claus-Peter Schnorr. “Efficient Signature Generation by Smart Cards”. In: Journal of Cryptology 4.3 (1991), pp. 161–174.

[3] Jonathan Katz and Yehuda Lindell. Introduction to Modern Cryptography. Chapman and Hall/CRC, 2014.

[4] Amos Fiat and Adi Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: Conference on the theory and application of cryptographic techniques. Springer. 1986, pp. 186–194.

[5] Yannick Seurin. “On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model”. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer. 2012, pp. 554–571.

[6] Don Johnson, Alfred Menezes, and Scott Vanstone. “The Elliptic Curve Digital Signature Algorithm (ECDSA)”. In: International Journal of Information Security 1.1 (2001), pp. 36–63.

[7] Mihir Bellare and Phillip Rogaway. “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols”. In: Proceedings of the 1st ACM conference on Computer and communications security. 1993, pp. 62–73.

[8] David Pointcheval and Jacques Stern. “Security Arguments for Digital Signatures and Blind Signatures”. In: Journal of cryptology 13.3 (2000), pp. 361–396.

[9] Pieter Wuille, Jonas Nick, and Anthony Towns. Schnorr Signatures for secp256k1. 2020. url: https://github.com/bitcoin/bips/blob/master/bip0340.mediawiki (visited on 03/11/2022).

[10] Mihir Bellare and Gregory Neven. “Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma”. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. ACM. 2006, pp. 390–399.

[11] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. “Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning”. In: 2016 IEEE Symposium on Security and Privacy (SP). IEEE. 2016, pp. 526–545.

[12] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. “Simple Schnorr Multi-Signatures with Applications to Bitcoin”. In: Designs, Codes and Cryptography (2018), pp. 1–26.

[13] Vasilios Mavroudis, Andrea Cerulli, Petr Svenda, Dan Cvrcek, Dusan Klinec, and George Danezis. “A Touch of Evil: High-Assurance Cryptographic Hardware from Untrusted Components”. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM. 2017, pp. 1583–1600.

[14] Daniel J. Bernstein. Multi-User Schnorr Security, revisited. Cryptology ePrint Archive, Report 2015/996. https://eprint.iacr.org/2015/996. 2015.

[15] Daniel J Bernstein. Pippenger’s Exponentiation Algorithm. 2002.

[16] Oded Goldreich. “Secure Multi-Party Computation”. In: Manuscript. Preliminary version 78 (1998).

[17] Ali Bagherzandi, Jung-Hee Cheon, and Stanislaw Jarecki. “Multisignatures Secure under the Discrete Logarithm Assumption and a Generalized Forking Lemma”. In: Proceedings of the 15th ACM conference on Computer and communications security. ACM. 2008, pp. 449–458.

[18] Ali Bagherzandi and Stanisław Jarecki. “Multisignatures Using Proofs of Secret Key Possession, as Secure as the Diffie-Hellman Problem”. In: International Conference on Security and Cryptography for Networks. Springer. 2008, pp. 218–235.

[19] Thomas Ristenpart and Scott Yilek. “The Power of Proofs-of-Possession: Securing Multiparty Signatures Against Rogue-Key Attacks”. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer. 2007, pp. 228–245.

[20] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. “Accountable-Subgroup Multisignatures”. In: Proceedings of the 8th ACM conference on Computer and Communications Security. ACM. 2001, pp. 245–254.

[21] Neal Koblitz and Alfred Menezes. “Critical Perspectives on Provable Security: Fifteen Years of "Another Look" Papers”. In: Advances in Mathematics of Communications 13.4 (2019), pp. 517–558.

[22] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. “On the Security of Two-Round Multi-Signatures”. In: 2019 IEEE Symposium on Security and Privacy (SP). IEEE. 2019, pp. 1084–1101.

[23] Changshe Ma, Jian Weng, Yingjiu Li, and Robert Deng. “Efficient Discrete Logarithm Based Multi-Signature Scheme in the Plain Public Key Model”. In: Designs, Codes and Cryptography 54.2 (2010), pp. 121–133.

[24] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. “Secure Distributed Key Generation for Discrete-Log Based Cryptosystems”. In: International Conference on the Theory and Applications of Cryptographic Techniques. Springer. 1999, pp. 295–310.

[25] Douglas R Stinson and Reto Strobl. “Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates”. In: Australasian Conference on Information Security and Privacy. Springer. 2001, pp. 417–434.

[26] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. “Revisiting the Distributed Key Generation for Discrete-Log Based Cryptosystems”. In: RSA Security 2003 (2003).

[27] Andrew Poelstra. Scriptless Scripts. 2018. url: https://download.wpsoftware.net/bitcoin/wizardry/mwslides/2018-05-18-l2/slides.pdf (visited on 03/06/2022).

[28] Andrew Poelstra and Nick Jonas. Adaptor Signatures and Atomic Swaps from Scriptless Scripts. 2019. url: https://github.com/ElementsProject/scriptlessscripts/blob/master/md/atomic-swap.md (visited on 03/06/2022).

[29] David Chaum. “Blind Signatures for Untraceable Payments”. In: Advances in Cryptology. Springer. 1983, pp. 199–203.

[30] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. “Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model”. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer. 2020, pp. 63–95.

[31] Claus Peter Schnorr. “Security of Blind Discrete Log Signatures Against Interactive Attacks”. In: International Conference on Information and Communications Security. Springer. 2001, pp. 1–12.

[32] David Wagner. “A Generalized Birthday Problem”. In: Annual International Cryptology Conference. Springer. 2002, pp. 288–304.

[33] Andreas M Antonopoulos. Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O’Reilly Media, Inc., 2014.

[34] Markus Jakobsson and Ari Juels. “Proofs of Work and Bread Pudding Protocols”. In: Secure Information Networks. Springer, 1999, pp. 258–272.

[35] Adam Back. Hashcash – A Denial of Service Counter-Measure. 2002.

[36] Ittay Eyal and Emin Gün Sirer. “Majority is not Enough: Bitcoin Mining is Vulnerable”. In: International Conference on Financial Cryptography and Data Security. Springer. 2014, pp. 436–454.

[37] Daniel RL Brown. “SEC 2: Recommended Elliptic Curve Domain Parameters”. In: Certicom Research, Certicom Corp (2010).

[38] Pieter Wuille. Hierarchical Deterministic Wallets. 2012. url: https://github.com/bitcoin/bips/blob/master/bip0032.mediawiki (visited on 04/13/2020).

[39] Pieter Wuille, Jonas Nick, and Anthony Towns. Taproot: SegWit Version 1 Spending Rules. 2020. url: https://github.com/bitcoin/bips/blob/master/bip0341.mediawiki (visited on 03/11/2020).

[40] Pieter Wuille, Jonas Nick, and Anthony Towns. Validation of Taproot Scripts. 2020. url: https://github.com/bitcoin/bips/blob/master/bip0342.mediawiki (visited on 03/11/2022).

[41] Gregory Maxwell. Taproot: Privacy Preserving Switchable Scripting. 2018. url: https://lists.linuxfoundation.org/pipermail/bitcoindev/2018-January/015614.html (visited on 03/13/2022).

[42] Gregory Maxwell. Graftroot: Private and Efficient Surrogate Scripts under the Taproot Assumption. 2018. url: https://lists.linuxfoundation.org/pipermail/bitcoindev/2018-February/015700.html (visited on 03/13/2022).

[43] Ilja Gerhardt and Timo Hanke. “Homomorphic Payment Addresses and the Pay-to-Contract Protocol”. In: arXiv preprint arXiv:1212.3257 (2012).

[44] Yehuda Lindell. “Fast Secure Two-Party ECDSA Signing”. In: Annual International Cryptology Conference. Springer. 2017, pp. 613–644.

[45] Joseph Poon and Thaddeus Dryja. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. 2016.

[46] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. “On Scaling Decentralized Blockchains”. In: International conference on financial cryptography and data security. Springer. 2016, pp. 106–125.

[47] Christian Decker, Rusty Russell, and Olaoluwa Osuntokun. eltoo: A Simple Layer2 Protocol for Bitcoin. 2018. url: https://blockstream.com/eltoo.pdf

[48] Christian Decker and Roger Wattenhofer. “A Fast and Scalable Payment Network With Bitcoin Duplex Micropayment Channels”. In: Symposium on Self-Stabilizing Systems. Springer. 2015, pp. 3–18.

[49] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability. Cryptology ePrint Archive, Report 2018/472. https://eprint.iacr.org/2018/472. 2018.

[50] Vasilios Mavroudis and Petr Svenda. “Towards Low-level Cryptographic Primitives for JavaCards”. In: arXiv preprint arXiv: 1810.01662 (2018).