Sección de Matemáticas
Universidad de La Laguna

Manuel Toledo Tauler

Aplicaciones de la Teoría de Códigos en Criptografía: Compartición de Secretos y Computación Multiparte

Applications of Coding Theory in Cryptography: Secret Sharing and Multiparty Computation

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Mayo de 2023

Dirigido por
Ignacio García Marco

Ignacio García Marco
Departamento de Matemáticas
Estadística e Investigación
Operativa
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

A mis padres, por darme siempre todo su apoyo y amor de manera incondicional. A ellos les debo todas mis virtudes.

A hermano, por ser la mejor molestia que pudo haberme tocado en la vida.

A Nacho, por transmitir las Matemáticas con tanta ilusión. No pude escoger mejor mentor para acompañarme en esta última etapa.

Y a Nuria, por estar siempre a mi lado, por todo su cariño. Por hacerme mejor matemático y mejor persona.

Gracias.

Manuel Toledo Tauler
La Laguna, 22 de mayo de 2023

Resumen · Abstract

Resumen

Dos retos de la Criptografía son la compartición de secretos y la computación multiparte. La Teoría Algebraica de Códigos aporta soluciones a ambos problemas, haciendo natural su estudio conjunto. Este trabajo desarrolla las herramientas matemáticas de interpolación polinomial y códigos Reed-Solomon y se utilizan para abordar los citados desafíos. Se exploran conceptos clave como privacidad y seguridad, y se presentan diversos protocolos que aumentan en dificultad, comenzando con el esquema privado de Shamir y avanzando hacia protocolos más complejos para terminar con productos seguros. Se utiliza una notación unificada y se reorganizan conceptos para facilitar la comprensión, y se introducen pequeñas mejoras en algunos protocolos existentes.

Palabras clave: *compartición de secretos – computación multiparte – criptografía – códigos lineales – Reed-Solomon – Shamir*

Abstract

Two challenges in Cryptography are secret sharing and multiparty computation. The field of Coding Theory provides solutions to both problems, prompting a natural study of them together. In this paper we develop the mathematical tools of polynomial interpolation and Reed-Solomon codes, and we apply them to solve said issues. We explore key concepts like privacy and security, and we introduce a series of protocols, gradually increasing their complexity. We start with Shamir's secret scheme, and advance to more intricate designs, finishing with private products computation. We use a unified notation and reorganize some concepts to enhance comprehension, while making subtle improvements to some existing protocols.

Keywords: *secret sharing – multiparty computation – criptography – linear codes – Reed-Solomon – Shamir*

Contenido

Introducción

La **Criptografía** es un campo de estudio que surge de la necesidad humana de salvaguardar la información y comunicarse de manera segura. Uno de los primeros desafíos criptográficos en la historia fue la **compartición de secretos**, mantener información oculta hasta que diversas partes acuerden revelarla.

Uno de los registros históricos más antiguos de compartición de secretos se encuentra en la **antigua Grecia**, donde se empleaban mecanismos basados en **tablillas de cera**. Este método implicaba dividir el mensaje en partes y distribuirlo entre mensajeros, de forma que solo se pudiese reconstruir cuando todas las partes se reunían. Aunque este método resultaba rudimentario y presentaba limitaciones significativas en cuanto a seguridad y eficiencia, sentó las bases para futuros avances.

A lo largo de los siglos, se desarrollaron otras técnicas para la compartición de secretos. Destaca, por ejemplo, la **"escritura Lumière"**, introducida en 1889 por los hermanos Lumière. Este sistema criptográfico permitía compartir un secreto entre varias personas. Consistía en el uso de tiras de papel transparente superpuestas, cada una con una parte del mensaje. Al combinar todas las tiras, se revelaba el secreto original. Aunque resulta un ingenioso avance respecto a las tablillas de cera, también presentaba limitaciones y no proporcionaba una solución robusta frente a la traición de uno o más participantes.

No fue hasta 1979 cuando **Adi Shamir** realizó un giro trascendental en la disciplina. Su influyente artículo **"How to share a secret"** [8] presentó un concepto revolucionario para la compartición de secretos mediante el uso de **polinomios**. Este esquema, conocido como el esquema de Shamir, marcó un hito en la disciplina criptográfica.

El esquema de Shamir permite dividir un secreto en múltiples partes asignando coeficientes a un polinomio, de modo que solo se requiere un número mínimo de

partes para reconstruir el secreto original. Aunque revolucionario, este método no proporciona una solución eficiente para detectar alteraciones en las partes compartidas.

Es en este punto donde la **Teoría Algebraica de Códigos** tiene mucho que aportar, proporcionando herramientas y técnicas para abordar las limitaciones del esquema de Shamir y mejorar la seguridad y confiabilidad de los esquemas de compartición de secretos. Al aplicar conceptos como la codificación y decodificación de mensajes con corrección de errores, se logra garantizar la integridad y autenticidad de las partes compartidas. Esto condujo al desarrollo de esquemas de compartición de secretos seguros.

Además, la **computación multiparte** ha surgido como un área de investigación estrechamente relacionada con la compartición de secretos. Permite que múltiples participantes realicen cálculos conjuntos sin revelar su información privada. Esta área se basa en las mismas técnicas de compartición de secretos para preservar la confidencialidad de los datos, y hace uso de la teoría de códigos para garantizar su seguridad. La combinación de la compartición de secretos y la computación multiparte ofrece soluciones más robustas y seguras para la colaboración en entornos distribuidos.

En la actualidad, tanto la compartición de secretos como la computación multiparte han adquirido una **relevancia creciente**. Con la digitalización de datos en aumento, y la aparición de tecnologías blockchain para la computación descentralizada que necesitan de colaboración segura, estas disciplinas han experimentado avances significativos. Investigadores y expertos continúan explorando nuevas técnicas, protocolos y aplicaciones para abordar los desafíos de seguridad en entornos distribuidos y proteger la privacidad de la información.

Este trabajo pretende ofrecer una **introducción autónoma** a los conceptos fundamentales de ambas disciplinas criptográficas. El objetivo es presentar algunos de los protocolos en estas áreas, recopilando en una única lectura toda la teoría matemática necesaria para comprender su funcionamiento y demostrar su validez. Para ello, se ha llevado a cabo un importante esfuerzo de síntesis y reorganización de las ideas que se desarrollan en las diferentes fuentes bibliográficas, a fin de proporcionar una lectura lo más independiente posible, con una complejidad creciente en el desarrollo de la misma.

El primer Capítulo, *"Preliminares: polinomios, códigos de Reed-Solomon y conceptos básicos"*, explora los **fundamentos matemáticos** necesarios. Se examinan los **polinomios** univariados y bivariados sobre un cuerpo y se introduce la teoría de códigos, destacando los **códigos lineales** y, en particular, los códigos

de **Reed-Solomon**. Introduciremos asimismo conceptos como la **privacidad** y la **seguridad** de un protocolo.

En el segundo Capítulo, *"Protocolos multiparte privados"*, estudiamos los protocolos que permiten la **compartición de secretos** y la **computación multiparte privada**. Comenzamos introduciendo el esquema privado de Shamir y el esquema en rampa, que permiten repartir información sensible sin poner en riesgo su privacidad. Se abordan también los protocolos multiparte, que posibilitan efectuar combinaciones lineales y productos de entradas privadas y secretas.

El tercer Capítulo, *"Protocolos multiparte seguros"*, comienza analizando los **problemas de seguridad** de los protocolos privados, y **aportando soluciones** a cada uno de ellos. A continuación, se refuerzan los protocolos privados del capítulo anterior, de forma que garanticen también la seguridad frente a los problemas explorados, obteniendo así protocolos de compartición de secretos y de computación multiparte seguros. Es en este Capítulo donde la Teoría de Códigos juega un papel crucial.

Finalmente, en el Capítulo de conclusiones, se exploran algunas de las limitaciones o posibles puntos de mejora de los protocolos expuesto, y se proponen algunas líneas de estudio que completarían el conocimiento de este trabajo, pero que se escapan del rango de extensión del mismo.

1

Preliminares: polinomios, códigos de Reed-Solomon, y conceptos básicos

Este primer Capítulo recopila las herramientas matemáticas necesarias para el estudio de protocolos de compartición de secretos y computación multiparte de este trabajo. La primera Sección presenta y repasa propiedades básicas de polinomios de una y dos variables [2]. La segunda Sección explora la noción de código lineal, y más concretamente las propiedades de los códigos de Reed-Solomon, tomando los resultados de las referencias [4] y [6], y detallando un decodificador eficiente [7]. Por último, en la tercera Sección se introducen conceptos básicos relacionados con la privacidad y la seguridad.

1.1 Polinomios sobre un cuerpo.

Esta Sección estudia la teoría de los polinomios univariados y bivariados sobre un cuerpo, en la que se basan los protocolos de este trabajo.

1.1.1 Polinomios univariados.

A continuación describiremos brevemente los polinomios univariados a fin de fijar notación, y expondremos algunas de sus propiedades fundamentales de uso recurrente en este trabajo.

Definición 1.1.1 Un *polinomio univariado* sobre un cuerpo $\mathbb{F}_q$ con coeficientes $a_0, a_1, ..., a_t \in \mathbb{F}_q$, $a_t \neq 0$, se define como:

$$a(x) = \sum_{i=0}^{t} a_i x^i \tag{1.1}$$

Diremos que el *grado* del polinomio $a(x)$ es t, y lo denotamos $\deg(a) = t$. Denotaremos al espacio vectorial de polinomios univariados sobre el cuerpo $\mathbb{F}_q$ como $\mathbb{F}_q[x]$, y al subespacio de polinomios de grado menor o igual a t como $\mathbb{F}_q[x]_{\leq t}$.

Proposición 1.1.2 Sean $a(x), b(x) \in \mathbb{F}_q[x]$ dos polinomios sobre $\mathbb{F}_q$. Se cumplen las siguientes propiedades:

1. La combinación lineal de evaluaciones es igual a la evaluación de la combinación lineal: $(\lambda a + \nu b)(x_0) = \lambda a(x_0) + \nu b(x_0)$.
2. El producto de evaluaciones es igual a la evaluación del producto: $a(x_0)b(x_0) = (ab)(x_0)$.

La interpolación polinómica juega un papel crucial en esta memoria.

Proposición 1.1.3 *(Existencia y unicidad del polinomio interpolador)* Dados $n+1$ pares de elementos de $\mathbb{F}_q$, $(x_1, y_1), ..., (x_{n+1}, y_{n+1})$ tales que $x_i \neq x_j$ si $i \neq j$, existe un único polinomio $p(x) \in \mathbb{F}_q[x]_{\leq n}$ que verifica que $p(x_i) = y_i, \forall i \in \{1, ..., n+1\}$.

Por último, introducimos la matriz de Vandermonde, la noción de polinomio truncado, y exponemos algunas de sus propiedades.

Definición 1.1.4 Dado $\vec{x} = (x_1, ..., x_n)$ un vector de n coordenadas o nodos en $\mathbb{F}_q$, con $x_i \neq x_j$ si $i \neq j$, definimos la *matriz de Vandermonde* de $\vec{x}$ como la siguiente matriz $V_{\vec{x}} \in (\mathbb{F}_q)_{n\times n}$:

$$V_{\vec{x}} = \begin{pmatrix} 1 & x_1^1 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^1 & \cdots & x_n^{n-1} \end{pmatrix} \tag{1.2}$$

Es claro que, dado un polinomio $p(x) = \sum_{i=0}^{n-1} a_i x^i$, se puede expresar el vector de evaluaciones del polinomio en $\vec{x}$ como $(p(x_1), ..., p(x_n))^\top = V_{\vec{x}}(a_0, ..., a_{n-1})^\top$.

Proposición 1.1.5 Toda matriz de Vandermonde es invertible.

Proposición 1.1.6 Sea $a(x) = \sum_{i=0}^{t} a_i x^i \in \mathbb{F}_q[x]_{\leq t}$ un polinomio de grado menor o igual a t, y tomemos n evaluaciones del polinomio $a(x_1), ..., a(x_n)$, con $x_i \neq x_j$ si $i \neq j$, y siendo $t < n$. Se tiene que para cualquier $k \in \{1, ..., n\}$, podemos expresar $a(x_k)$ como una combinación lineal del resto de evaluaciones:

$$a(x_k) = \sum_{i=1}^{n} \lambda_i a(x_i) \tag{1.3}$$

Además, estos escalares $\lambda_1, ..., \lambda_n$ quedan definidos por los valores $x_1, ..., x_n$, es decir que no dependen del polinomio $a(x)$.

Demostración. Sea $V_{\vec{x}}$ la matriz de Vandermonde para los elementos $x_1, ..., x_n$, $\vec{x_k} = (1, x_k, ..., x_k^{n-1})$ su columna k-ésima, y sea $\vec{a} = (a_0, a_1, ..., a_t, 0, ..., 0)$ un vector de longitud n que tiene por primeras $t+1$ coordenadas los coeficientes del polinomio $a(x)$. Sabiendo además que la matriz de Vandermonde es invertible, podemos escribir:

$$a(x_k) = \vec{x_k}\,\vec{a}^\top = \vec{x_k}(V_{\vec{x}}^{-1}V_{\vec{x}})\vec{a}^\top = (\vec{x_k}\,V_{\vec{x}}^{-1})(V_{\vec{x}}\,\vec{a}^\top) = (\vec{x_k}\,V_{\vec{x}}^{-1})(a(x_1), ..., a(x_n))$$

Por tanto, podemos expresar $a(x_k)$ como combinación lineal del conjunto de evaluaciones $a(x_1), ..., a(x_n)$. Además, los escalares del vector $(\vec{x_k}\, V_{\vec{x}}^{-1})$ solo dependen de los elementos $x_1, ..., x_n$, siendo independientes del polinomio $a(x)$. □

Proposición 1.1.7 Sea $a(x) = \sum_{i=0}^{t} a_i x^i \in \mathbb{F}_q[x]_{\leq t}$ un polinomio de grado menor o igual a t, y tomemos n evaluaciones del polinomio $a(x_1), ..., a(x_n)$, con $x_i \neq x_j$ si $i \neq j$, y siendo $t < n$. El término independiente del polinomio $a_0 = a(0)$ se puede obtener como combinación lineal de las n evaluaciones:

$$a_0 = \sum_{i=1}^{n} \lambda_i a(x_i) \tag{1.4}$$

Además, estos escalares $\lambda_1, ..., \lambda_n$ quedan definidos por los valores $x_1, ..., x_n$, es decir que no dependen del polinomio $a(x)$.

Demostración. La demostración es inmediata por la Proposición 1.1.6. □

Definición 1.1.8 Dado un polinomio $a(x) = \sum_{i=0}^{m} a_i x^i \in \mathbb{F}_q[x]_{\leq m}$, y dado $t < m$, definimos el *polinomio truncado en* m de $a(x)$ al siguiente polinomio:

$$trunc_t a(x) = \sum_{i=0}^{t} a_i x^i \tag{1.5}$$

Proposición 1.1.9 Sea un polinomio $a(x) = \sum_{i=0}^{m} a_i x^i \in \mathbb{F}_q[x]_{\leq m}$, sea $b(x) = \sum_{i=0}^{t} a_i x^i \in \mathbb{F}_q[x]_{\leq m}$ su polinomio truncado en t, $b(x) = trunc_t a(x)$, y sean $x_1, ..., x_n \in \mathbb{F}_q$ nodos, con $t < m < n$. Entonces, existe una matriz $A \in (\mathbb{F}_q)_{n\times n}$ tal que $(b(x_1), ..., b(x_n)) = A(a(x_1), ..., a(x_n))$.

Demostración. En primer lugar, podemos definir un vector de longitud n con las primeras $m+1$ coordenadas los coeficientes del polinomio $a(x)$, y de resto elementos nulos: $\vec{a} = (a_0, a_1, ..., a_m, 0, ..., 0)$. De este modo, podemos escribir el vector de evaluaciones en $\vec{x} = (x_1, ..., x_n)$ como sigue:

$$(a(x_1), ..., a(x_n))^\top = V_{\vec{x}}\, \vec{a}^\top \tag{1.6}$$

siendo $V_{\vec{x}}$ la matriz de Vandermonde. Además, por ser dicha matriz invertible, podemos escribir lo siguiente:

$$\vec{a}^\top = V_{\vec{x}}^{-1}\, (a(x_1), ..., a(x_n))^\top \tag{1.7}$$

Análogamente, tomando $\vec{b} = (a_0, a_1, ..., a_t, 0, ..., 0)$ vector de longitud n, tenemos:

$$(b(x_1), ..., b(x_n))^\top = V_{\vec{x}}\, \vec{b}^\top \tag{1.8}$$

Por otra parte, tomemos la siguiente matriz de proyección $P = (p_i^j) \in (\mathbb{F}_q)_{n\times n}$, donde $p_i^j = 1$ si $i = j \leq t+1$, y $p_i^j = 0$ en caso contrario. De este modo, podemos obtener el vector $\vec{b}^\top$ multiplicando la matriz P por el vector $\vec{a}^\top$ de la siguiente manera:

$$\vec{b}^\top = P\vec{a}^\top \tag{1.9}$$

Por tanto, combinando las ecuaciones (1.7), (1.8) y (1.9), podemos deducir:

$$(b(x_1), ..., b(x_n))^\top = V_{\vec{x}}\, P\, V_{\vec{x}}^{-1}(a(x_1), ..., a(x_n)) \tag{1.10}$$

Concluyendo así que existe una matriz $A = V_{\vec{x}}\, P\, V_{\vec{x}}^{-1}$ que verifica las condiciones de la proposición.

□

1.1.2 Polinomios bivariados.

Definición 1.1.10 Un *polinomio bivariado* sobre un cuerpo $\mathbb{F}_q$ se define como:

$$B(x, y) = \sum_{i=0}^{t}\sum_{j=0}^{m} b_{ij}x^i y^j \tag{1.11}$$

Diremos que el *grado* del polinomio B para la variable x es t y para la variable y es m, siempre que existan $1 \leq i \leq t, 1 \leq j \leq m$ tales que $b_{tj}, b_{im} \neq 0$ y lo denotaremos por $\deg_x(B) = t$, y $\deg_y(B) = m$. Denotaremos al conjunto de polinomios bivariados sobre el cuerpo $\mathbb{F}_q$ como $\mathbb{F}_q[x, y]$.

Proposición 1.1.11 *(Interpolación polinómica bivariada)* Dados $\{\alpha_1, ..., \alpha_{t+1}\}$ nodos de $\mathbb{F}_q$ y $\{f_1, ..., f_{t+1}\}$ polinomios univariados de grado menor o igual a t en $\mathbb{F}_q[x]$, existe un único polinomio bivariado $B(x, y) \in \mathbb{F}_q[x, y]$ que verifique:

a) $\deg_x(B), \deg_y(B) \leq t$.
b) $B(x, \alpha_k) = f_k(x), \forall k \in \{1, ..., t+1\}$.

Demostración. Comenzamos demostrando la unicidad del polinomio bivariado interpolador. Definimos el siguiente polinomio $B(x, y)$ mediante interpolación de Lagrange:

$$B(x, y) = \sum_{i=1}^{t+1} f_i(x) \prod_{\substack{j=1 \\ j\neq i}}^{t+1} \frac{(y - \alpha_j)}{(\alpha_i - \alpha_j)} \tag{1.12}$$

Podemos definir los polinomios $\pi_i(y)$ de la siguiente forma:

$$\pi_i(y) = \prod_{\substack{j=1 \\ j\neq i}}^{t+1} \frac{(y - \alpha_j)}{(\alpha_i - \alpha_j)} \tag{1.13}$$

Podremos entonces escribir el polinomio $B(x, y)$ como sigue:

$$B(x,y) = \sum_{i=1}^{t+1} f_i(x)\pi_i(y) \tag{1.14}$$

Es claro que se cumple a), es decir, $\deg_x(B) = \deg_y(B) = t$ por la construcción del polinomio. Veamos si se verifica b): estudiando la expresión $\pi_i(y)$ cuando $y = \alpha_k$, para $k \in \{1, ..., t+1\}$, encontramos dos posibilidades:

- **Si $i = k$:**

$$\pi_i(\alpha_k) = \prod_{\substack{j=1 \\ j\neq i}}^{t+1} \frac{(\alpha_k - \alpha_j)}{(\alpha_k - \alpha_j)} = 1 \tag{1.15}$$

- **Si $i \neq k$:**

$$\pi_i(\alpha_k) = \prod_{\substack{j=1 \\ j\neq i}}^{t+1} \frac{(\alpha_k - \alpha_j)}{(\alpha_i - \alpha_j)} = \frac{(\alpha_k - \alpha_k)\prod_{j\neq i,k}(\alpha_k - \alpha_j)}{\prod_{j\neq i}(\alpha_i - \alpha_j)} = 0 \tag{1.16}$$

Es decir, $\pi_i(\alpha_k) = \delta_{i,k}$, siendo $\delta_{i,k}$ la delta de Kronecker de i, k. Por tanto, al evaluar $B(x, y)$ en $y = \alpha_k$, con $k \in \{1, ..., t+1\}$, tenemos que:

$$B(x,\alpha_k) = \sum_{i=1}^{t+1} f_i(x)\delta_{i,k} = f_k(x) \tag{1.17}$$

Veamos ahora la unicidad del polinomio bivariado interpolador. Dados los nodos $\{\alpha_1, ..., \alpha_{t+1}\}$ de $\mathbb{F}_q$, y los polinomios $\{f_1, ..., f_{t+1}\}$ de $\mathbb{F}_q^t[x]$, supongamos que existen $B_1(x, y)$ y $B_2(x, y)$ dos polinomios bivariados que verifican a) y b). Podemos definir el polinomio bivariado $R(x, y) = B_1(x, y) - B_2(x, y)$. Este polinomio cumple que $\deg_x(R), \deg_y(R) \leq t$ por construcción, luego podemos expresarlo de la siguiente forma:

$$R(x,y) = \sum_{i=0}^{t}\sum_{j=0}^{t} r_{i,j}x^i y^j \tag{1.18}$$

Vamos a comprobar que $R(x, y) \equiv 0$. En primer lugar, tenemos que para cualquier $k \in \{1, ..., t+1\}$ se verifica que:

$$R(x,\alpha_k) = B_1(x,\alpha_k) - B_2(x,\alpha_k) = f_k(x) - f_k(x) = 0 \tag{1.19}$$

Aplicando esto en la expresión del polinomio (1.18):

$$R(x,\alpha_k) = \sum_{i=0}^{t}\sum_{j=0}^{t} r_{i,j}x^i\alpha_k^j = \sum_{i=0}^{t}(\sum_{j=0}^{t} r_{i,j}\alpha_k^j)x^i = 0, \forall x \in \mathbb{F}_q \tag{1.20}$$

Por tanto, podemos concluir que:

$$\sum_{j=0}^{t} r_{i,j}\alpha_k^j = 0, \qquad \forall i \in \{0, ..., t\} \tag{1.21}$$

A continuación, podemos definir $t+1$ polinomios univariados de la siguiente forma:

$$h_i(x) = \sum_{j=0}^{t} r_{i,j}x^j, \qquad \forall i \in \{0, ..., t\} \tag{1.22}$$

Cada uno de estos polinomios $h_i(x)$ de grado menor o igual que t se anula al evaluarlo en $t+1$ puntos. Más concretamente, al evaluarlos en $x = \alpha_k$, con $k \in \{1, ..., t+1\}$, por la ecuación (1.21):

$$h_i(\alpha_k) = \sum_{j=0}^{t} r_{i,j}\alpha_k^j = 0 \tag{1.23}$$

Esto nos permite concluir que todos ellos son polinomios nulos, es decir $h_i(x) = \sum_{j=0}^{t} r_{i,j}x^j \equiv 0$. Podemos concluir así que todos sus coeficientes son nulos, $r_{i,j} = 0$, para cualesquiera $i, j \in \{0, 1, ..., t\}$. Al ser estos los $(t+1)^2$ coeficientes del polinomio $R(x, y)$, tal y como se indica en la ecuación (1.18), podemos concluir que $R(x, y) \equiv 0$. Luego $B_1(x, y) = B_2(x, y)$, probando así la unicidad del polinomio bivariado interpolador. □

A continuación, estudiaremos un resultado que es particularmente útil para los protocolos de compartición de secretos y de computación multiparte que se describen en los capítulos siguientes.

Proposición 1.1.12 Dados $n > t \in \mathbb{N}$, los nodos $\{\alpha_1, ..., \alpha_n\} \subset \mathbb{F}_q$, con $\alpha_i \neq \alpha_j$ si $i \neq j$. Si se toman $\{f_1(x), ..., f_n(x)\} \subset \mathbb{F}_q[x]_{\leq t}$ y $\{g_1(y), ..., g_n(y)\} \subset \mathbb{F}_q[y]_{\leq t}$ dos colecciones de n polinomios univariados de grado t de tal forma que verifiquen:

$$f_i(\alpha_j) = g_j(\alpha_i), \qquad \forall i, j \in \{1, ..., n\} \tag{1.24}$$

Entonces existe un único polinomio bivariado $B(x, y) \in \mathbb{F}_q[x, y]$ que verifique:

a) $\deg_x(B), \deg_y(B) \leq t$.
b) $f_i(x) = B(x, \alpha_i), \forall i \in \{1, ..., n\}$.
c) $g_j(y) = B(\alpha_j, y), \forall j \in \{1, ..., n\}$.

Demostración. Primero, tomaremos un subconjunto $\{i_1, ..., i_{t+1}\}$ de $t+1$ elementos del conjunto $\{1, ..., n\}$ para poder definir un polinomio bivariado interpolador. Por la Proposición 1.1.11 sabemos que existe un único polinomio bivariado $B(x, y) \in \mathbb{F}_q[x, y]$ con $\deg_x(B) = \deg_y(B) = t$ tal que $B(x, \alpha_i) = f_i(x), \forall i \in \{i_1, ..., i_{t+1}\}$.

Por otra parte, sabemos que $\forall i, j \in \{1, ..., n\}$, se cumple que $g_j(\alpha_i) = f_i(\alpha_j)$. En particular, esto también se cumple $\forall i \in \{i_1, ..., i_{t+1}\}, \forall j \in \{1, ..., n\}$. Además, por definición de $B(x, y)$, tenemos que $f_i(\alpha_j) = B(\alpha_j, \alpha_i)$. Por tanto, podemos concluir que:

$$g_j(\alpha_i) = B(\alpha_j, \alpha_i), \qquad \forall i \in \{i_1, ..., i_{t+1}\}, \forall j \in \{1, ..., n\} \tag{1.25}$$

Sabiendo que $\deg(g_j) = \deg_y(B) = t$, y que los polinomios $g_j(y)$ coinciden con el polinomio $B(\alpha_j, y)$ en $t + 1$ evaluaciones, podemos garantizar que:

$$g_j(y) = B(\alpha_j, y), \qquad \forall j \in \{1, 2, ..., n\} \tag{1.26}$$

Ahora solo queda garantizar el resultado para los polinomios $\{f_1(x), ..., f_n(x)\}$ (aunque sea trivial para el subconjunto de índices escogido para la construcción del polinomio interpolador, no es inmediato para el resto de índices).
Por la ecuación (1.26), podemos particularizar y garantizar lo siguiente:

$$g_j(\alpha_i) = B(\alpha_j, \alpha_i), \qquad \forall i, j \in \{1, ..., n\} \tag{1.27}$$

Como además sabemos que $g_j(\alpha_i) = f_i(\alpha_j)$, $\forall i, j \in \{1, ..., n\}$, podemos escribir $g_j(\alpha_i) = f_i(\alpha_j) = B(\alpha_j, \alpha_i)$. E igual que antes, tenemos que $\deg(f_i) = \deg_x(B) = t$, y que los polinomios $f_i(x)$ coinciden con el polinomio $B(x, \alpha_i)$ en $t + 1$ evaluaciones, luego podemos garantizar que:

$$f_i(x) = B(x, \alpha_i), \qquad \forall i \in \{1, ..., n\} \tag{1.28}$$

□

1.2 Teoría de códigos, y códigos Reed-Solomon.

Esta Sección introduce los conceptos básicos de códigos lineales, y trata las propiedades específicas de los códigos Reed-Solomon ([4] y [6]), terminando con la descripción del decodificador eficiente de Berlekamp-Welch [7].

1.2.1 Códigos lineales.

Definición 1.2.1 Dado $\mathbb{F}_q$ un cuerpo finito con q elementos, definimos un (n, k)-*código lineal* como un subespacio vectorial C de dimensión k del espacio vectorial $\mathbb{F}_q^n$. Los elementos de este código $\vec{c} \in C$ reciben el nombre de *palabras de código*, o simplemente *palabras*.

- Diremos que n es la *longitud* del código, puesto que cada palabra del código es una secuencia de n elementos del cuerpo.

- Diremos que k es la *dimensión* del código, en consecuencia existen q^k palabras de código.

Un *mensaje* es un vector $\vec{m}$ del espacio vectorial $\mathbb{F}_q^k$. El *proceso de codificación* es una aplicación lineal que transforma el vector del mensaje $\vec{m} \in \mathbb{F}_q^k$ en un vector palabra $\vec{c} \in C \subset \mathbb{F}_q^n$.

Definición 1.2.2 Dado un (n, k)-código lineal sobre $\mathbb{F}_q$, diremos que G es una *matriz generatriz* si se trata de una matriz de dimensión $k \times n$ con elementos en $\mathbb{F}_q$, y sus filas forman una base de C. Dado un mensaje $\vec{m} \in \mathbb{F}_q^k$, diremos que su *codificación* es la palabra del código $\vec{c} \in C$ que se obtiene como $\vec{c} = \vec{m} \cdot G$.

Definición 1.2.3 Dado un (n, k)-código lineal, diremos que $\vec{h}$ es un *vector de paridad* si es un vector de longitud n tal que $G \cdot \vec{h}^\top = 0$. Diremos que H es una *matriz de paridad de* C si se trata de una matriz de dimensión $(n-k) \times n$ y sus filas son vectores de paridad linealmente independientes.

Definición 1.2.4 Dado un (n, k)-código lineal sobre $\mathbb{F}_q$ dado por su matriz de paridad H, y un vector $\vec{y} \in \mathbb{F}^n$, definimos su *síndrome* como:

$$syn(\vec{y}) = H \cdot \vec{y}^\top \tag{1.29}$$

Se observa que un vector $\vec{y} \in \mathbb{F}_q^n$ pertenece al código si y sólo si su síndrome es el vector nulo. En consecuencia, se tiene el siguiente resultado.

Proposición 1.2.5 Dado un vector $\vec{y} \in \mathbb{F}_q^n$ de la forma $\vec{y} = \vec{c} + \vec{e}$, donde $\vec{c} \in C$ es una palabra del código y $\vec{e}$ un vector en $\mathbb{F}_q^n$, se tiene que $syn(\vec{y}) = syn(\vec{e})$.

Definición 1.2.6 Se define el *peso de Hamming* de un vector $\vec{y} \in \mathbb{F}_q^n$, $\omega_H(\vec{x})$, como el número de coordenadas no nulas del vector.

Definición 1.2.7 Diremos que un (n, k)-código lineal es *t-corrector* si para cualesquiera $\vec{c_1}, \vec{c_2} \in C \subset \mathbb{F}_q^n$ palabras de código, y para cualesquiera $\vec{e_1}, \vec{e_2} \in \mathbb{F}_q^n$ tales que $\omega_H(\vec{e_1}), \omega_H(\vec{e_2}) \leq t$, siempre se verifica la siguiente desigualdad:

$$\vec{c_1} + \vec{e_1} \neq \vec{c_2} + \vec{e_2} \tag{1.30}$$

Definición 1.2.8 La *distancia de Hamming* entre dos vectores $\vec{x}$ e $\vec{y}$ de $\mathbb{F}_q^n$ se define como el número de coordenadas en la que difieren. Se denota como $d_H(\vec{x}, \vec{y})$, y coincide con el peso de Hamming de su definición, es decir:

$$d_H(\vec{x}, \vec{y}) = \omega_H(\vec{x} - \vec{y}) \tag{1.31}$$

Proposición 1.2.9 La distancia de Hamming verifica las cuatro propiedades usuales de una función distancia:

1. $d_H(\vec{x}, \vec{y}) \geq 0, \ \forall \vec{x}, \vec{y} \in \mathbb{F}_q^n$.

2. $d_H(\vec{x},\vec{y}) = 0 \iff \vec{x} = \vec{y}$.
3. $d_H(\vec{x},\vec{y}) = d_H(\vec{y},\vec{x})$.
4. $d_H(\vec{x},\vec{y}) \leq d_H(\vec{x},\vec{z}) + d_H(\vec{z},\vec{y})$ *(Desigualdad triangular).*

Definición 1.2.10 La *distancia mínima* de un código d es la mínima distancia de Hamming entre cualesquiera dos palabras diferentes del código.

Proposición 1.2.11 Para un (n,k)-código lineal, la distancia mínima es igual al peso mínimo de una palabra distinta de cero.

Demostración. Sea $\vec{c}$ una palabra de peso mínimo. Para cualquier palabra (y en particular para $\vec{c}$) se verifica la siguiente igualdad:

$$\omega_H(\vec{c}) = d_H(\vec{0},\vec{c}) \tag{1.32}$$

Por tanto, como el peso mínimo coincide con una distancia, podemos garantizar que $d_{min} \leq \omega_{min}$. Por otra parte, sean $\vec{c_1}$ y $\vec{c_2}$ dos palabras a distancia mínima. Se verifica la siguiente igualdad:

$$d_H(\vec{c_1},\vec{c_2}) = \omega_H(\vec{c_1} - \vec{c_2}) \tag{1.33}$$

Por tanto, como la distancia mínima coincide con un peso, podemos garantizar que $\omega_{min} \leq d_{min}$. De esta forma podemos concluir que $\omega_{min} = d_{min}$.

□

Proposición 1.2.12 Un (n,k)-código es t-corrector si y sólo si $t < \frac{d}{2}$.

Demostración. Supongamos que $t < \frac{d}{2}$, y que existen dos palabras $\vec{c_1},\vec{c_2}$ y dos errores $\vec{e_1},\vec{e_2}$, con pesos $\omega_H(\vec{c_1}),\omega_H(\vec{c_2}) \leq t$ tales que se verifica:

$$\vec{c_1} + \vec{e_1} = \vec{c_2} + \vec{e_2} \tag{1.34}$$

Entonces, tenemos que $\vec{c_1} - \vec{c_2} = \vec{e_2} - \vec{e_1}$. Por tanto, tomando pesos de Hamming a ambos lados de la igualdad, $\omega_H(\vec{c_1} - \vec{c_2}) = \omega_H(\vec{e_2} - \vec{e_1})$, o lo que es lo mismo, $d_H(\vec{c_1},\vec{c_2}) = d_H(\vec{e_2},\vec{e_1})$. Por la desigualdad triangular, $d_H(\vec{e_2} - \vec{e_1}) \leq d_H(\vec{e_2},\vec{0}) + d_H(\vec{0},\vec{e_1}) = \omega_H(\vec{c_2}) + \omega_H(\vec{c_1}) \leq 2t < d$. Con todo lo anterior, llegamos a la siguiente cadena de desigualdades de la que se concluye que d no es la distancia mínima:

$$d_H(\vec{c_1},\vec{c_2}) \leq \omega_H(\vec{c_2}) + \omega_H(\vec{c_1}) \leq 2t < d \tag{1.35}$$

Por otra parte, sea $t \geq \frac{d}{2}$, y tomemos una palabra $\vec{c}$ tal que $\omega_H(\vec{c}) = d$. Podemos cambiar t de las posiciones no nulas de $\vec{c}$ por ceros, de forma que obtenemos un vector $\vec{y}$, con $d_H(\vec{0},\vec{y}) \leq d - t \leq t$ y $d_H(\vec{c},\vec{y}) \leq t$. Pero entonces $\vec{0} + \vec{y} = \vec{c} + (\vec{y} - \vec{c})$, luego no es t-corrector. □

Definición 1.2.13 Un *decodificador de mínima distancia* es un algoritmo que recibe $\vec{y} \in \mathbb{F}_q^n$, y devuelve la palabra de código $\vec{c}$ que satisface que $d_H(\vec{y},\vec{c}) < \frac{d}{2}$ si dicha palabra existe, o devuelve *error* en caso contrario.

Observación 1.2.14 *Por la Proposición 1.2.12, dado $\vec{y}$ un vector en $\mathbb{F}_q^n$, como mucho puede existir una palabra de código $\vec{c}$ a una distancia menor que $\frac{d}{2}$.*

Teorema 1.2.15 *(Cota de Singleton)* Sea C un (n,k)-código lineal sobre $\mathbb{F}_q$ con distancia mínima d. Entonces, se verifica la desigualdad $d \leq n-k+1$.

Demostración. En primer lugar, veamos que cualquier subconjunto de r columnas de la matriz de paridad H es linealmente independiente, con $r < d$. Supongamos que existe un subconjunto de r columnas de H linealmente dependientes. Denotemos estas columnas por $\vec{h_{i_1}}, ..., \vec{h_{i_r}}$. Existen por tanto r escalares $\lambda_{i_1}, ..., \lambda_{i_r}$ tales que $\sum_{j=1}^{r} \lambda_{i_j} \vec{h_{i_j}} = \vec{0}$. Si tomamos el vector $\vec{y} = ((y_i)_{i=1}^n)$, donde $y_{i_j} = \lambda_{i_j}$ y el resto de coordenadas son nulas, se tiene que $\omega_H(\vec{y}) = r < d$ y que $H\vec{y}^\top = \vec{0}$. Esto es absurdo, puesto que d es la distancia mínima del código, luego un subconjunto de $r < d$ columnas de H no puede ser linealmente dependiente.

Tenemos por tanto que cualquier subconjunto de $d-1$ columnas de H es linealmente independiente. Esto nos permite escribir que $d-1 \leq \mathrm{rg}(H) = n-k$, concluyendo así que $d \leq n-k+1$. □

1.2.2 Códigos de Reed-Solomon.

Definición 1.2.16 Sean $n \geq q$, $n \geq k$, dado $\vec{\alpha} = (\alpha_1, ..., \alpha_n) \in \mathbb{F}_q^n$ un vector de n elementos distintos, y sea $\mathbb{F}_q[x]_{\leq k-1}$ el espacio vectorial de los polinomios de grado $\leq k-1$ con coeficientes en $\mathbb{F}_q$, un $(n, k, \vec{\alpha}, \mathbb{F}_q)$-*código de Reed-Solomon* consiste en el conjunto de palabras:

$$(p(\alpha_1), ..., p(\alpha_n)), \qquad p(x) \in \mathbb{F}_q[x]_{\leq k-1} \tag{1.36}$$

Este código de Reed-Solomon es un (n,k)-código lineal. En efecto, la matriz generatriz del código Reed-Solomon descrito coincide con la k primeras filas de la matriz de Vandermonde $V_{\vec{\alpha}}$ de la Definición 1.1.4, que tiene rango máximo, $\mathrm{rg}(G) = k$.

Proposición 1.2.17 Para un $(n, k, \vec{\alpha}, \mathbb{F}_q)$-código de Reed-Solomon, la distancia mínima es $d = n-k+1$, es decir que se cumple la cota de Singleton.

Demostración. Tomemos dos palabras de código distintas $\vec{c} = (p(\alpha_1), ..., p(\alpha_n))$ y $\vec{d} = (q(\alpha_1), ..., q(\alpha_n))$, con $p(x), q(x) \in \mathbb{F}_q[x]_{\leq k-1}$. Por ser $\vec{c} \neq \vec{d}$ palabras distintas, sabemos que $p(x) \neq q(x)$. Dos polinomios de grado $\leq k-1$ distintos tienen a lo sumo $k-1$ evaluaciones comunes. En consecuencia, al menos $n-k+1$ evaluaciones en los nodos $\alpha_1, ..., \alpha_n$ serán distintas. Por tanto, la distancia mínima entre dos palabras de código es $d \geq n-k+1$. Además, al tratarse de un (n,k)-código lineal, se verifica la desigualdad de Singleton $d \leq n-k+1$. Como consecuencia de ambas desigualdades, $d = n-k+1$.

□

Una de las ventajas de trabajar con códigos de Reed-Solomon es este último resultado. Conocer la distancia mínima del código permite garantizar que una palabra con menos de $\frac{d}{2}$ errores puede ser corregida. Además, como se tiene igualdad en la cota de Singleton, se tiene la mayor distancia posible para un (n, k)-código. En la práctica es conveniente contar con un algoritmo de decodificación eficiente en tiempo polinómico como el que se presenta a continuación.

Decodificador eficiente de Berlekamp-Welch.

Definición 1.2.18 Se define el *algoritmo de decodificación de códigos Reed-Solomon de Berlekamp-Welch* de la siguente forma:

- **Entrada:** La descripción de un $(n, k, \vec{\alpha}, \mathbb{F}_q)$-código de Reed-Solomon, y un vector $\vec{y} = (y_1, ..., y_n) \in \mathbb{F}_q^n$.
- **Salida:** Un polinomio $p(x) \in \mathbb{F}_q[x]_{\leq k-1}$ tal que $d_H((p(\alpha_1), ..., p(\alpha_n)), \vec{y}) < \frac{n-k+1}{2}$, o *fallo* si no existe tal polinomio.
- **Proceso:** Pasos del algoritmo.
  1. Calcular un polinomo mónico no nulo $E(x)$ de grado exactamente $\frac{n-k+1}{2}$, y un polinomio $Q(x)$ de grado menor o igual a $e + k - 1$ que verifiquen la siguiente ecuación:

$$y_i E(\alpha_i) = Q(\alpha_i),\ 1 \leq i \leq n \tag{1.37}$$

     De no existir, el protocolo devuelve *fallo*.
  2. Si $E(x)$ no divide a $Q(x)$, el protocolo devuelve *fallo*. En caso contrario, computar $p(x) = \frac{Q(x)}{E(x)}$. Si $d_H(\vec{y}, (p(\alpha_1), ..., p(\alpha_n))) > \frac{n-k+1}{2}$, el protocolo devuelve *fallo*. En caso contrario, el protocolo devuelve $p(x)$.

Teorema 1.2.19 Si existe un polinomio $p(x)$ tal que $d_H(\vec{y}, (p(\alpha_1), ..., p(\alpha_n)) \leq \frac{n-k+1}{2}$, entonces el algoritmo descrito devuelve $p(x)$.

Para demostrar este resultado, basta con el estudio de los siguientes lemas.

Lema 1.2.20 Si $d_H(\vec{y}, (p(\alpha_1), ..., p(\alpha_n)) \leq e = \frac{n-k+1}{2}$, entonces existen $E(x)$ polinomio mónico de grado exactamente e, y $Q(x)$ polinomio de grado menor o igual a $e + k - 1$ que verifican la ecuación (1.37) y tales que $p(x) = \frac{Q(x)}{E(x)}$.

Demostración. Definimos el polinomio $E(x)$ de la siguiente forma:

$$E(x) = \prod_{i\,:\,p(\alpha_i) \neq y_i} (x - \alpha_i)\, x^{e - d_H(\vec{y}, (p(\alpha_i))_i)} \tag{1.38}$$

Este es un polinomio localizador de errores, con raíces en los nodos en los que $p(\alpha_i) \neq y_i$. Es mónico y de grado exactamente e. A continuación, definimos el polinomio $Q(x)$ como sigue:

$$Q(x) = E(x)p(x) \tag{1.39}$$

Este polinomio cumple que $\deg(Q) \leq \deg(E) + \deg(p) \leq e + k - 1$. Además, es evidente que $p(x) = \frac{Q(x)}{E(x)}$. Solo queda comprobar que los polinomios $E(x)$ y $Q(x)$ verifican la ecuación (1.37):

- Si $y_i = p(\alpha_i)$, entonces $Q(\alpha_i) = y_i E(\alpha_i)$.
- Si $y_i \neq p(\alpha_i)$, entonces $y_i E(\alpha_i) = E(\alpha_i) = 0 = p(\alpha_i)E(\alpha_i) = Q(\alpha_i)$.

Por tanto, los polinomios $E(x)$ y $Q(x)$ verifican las condiciones del enunciado. □

Lema 1.2.21 Si existen dos soluciones $(E_1(x), Q_1(x)) \neq (E_2(x), Q_2(x))$ distintas a la ecuación (1.37), entonces cumplen que $\frac{Q_1(x)}{E_1(x)} = \frac{Q_2(x)}{E_2(x)}$.

Demostración. Podemos definir el polinomio $R(x) = Q_1(x)E_2(x) - Q_2(x)E_1(x)$. Este polinomo es de grado menor o igual a $2e + k - 1$. Como ambos pares de polinomios verifican la ecuación (1.37), podemos escribir que para cualquier $1 \leq i \leq n$, $R(\alpha_i) = y_i E_1(\alpha_i)E_2(\alpha_i) - y_i E_2(\alpha_i)E_1(\alpha_i) = 0$. Por tanto, el polinomio $R(x)$ es nulo en n evaluaciones distintas. Como $e < \frac{n-k+1}{2}$, podemos escribir $2e + k - 1 < n$, deduciendo así que $R(x) \equiv 0$. De esta forma, $Q_1(x)E_2(x) = Q_2(x)E_1(x)$, y por tanto, $\frac{Q_1(x)}{E_1(x)} = \frac{Q_2(x)}{E_2(x)}$. □

Estos dos lemas demuestran el Teorema 1.2.19 y, por tanto, la corrección del algoritmo de Berlekamp-Welch. Es decir, para una palabra recibida $\vec{y} \in \mathbb{F}_q^n$ con menos de $\frac{n-k+1}{2}$ errores, el decodificador devuelve el polinomio $p(x)$ correcto. Para calcular los polinomios $E(x)$ y $Q(x)$ del paso 1, solo hay que resolver un sistema lineal de ecuaciones, tomando como incógnitas los $2e + k$ coeficientes de ambos polinomios, y como n ecuaciones las que se obtienen de verificar (1.37), sabiendo además que $2e + k \leq n$. Como el primer lema nos garantiza que este sistema tiene solución, se puede resolver con métodos como la eliminación gaussiana de forma eficiente. La división de polinomios univariados del segundo paso también tiene cálculo eficiente. En consecuencia, el algoritmo de Berlekamp-Welch proporciona un decodificador eficiente de los códigos Reed-Solomon.

1.3 Conceptos básicos de protocolos multiparte.

Definición 1.3.1 Definimos un *protocolo multiparte* P como un conjunto de los siguientes elementos:

- **Participantes**: personas que participan en el protocolo.
- **Entradas públicas**: elementos conocidos por todos los participantes.

- **Entradas privadas**: elementos de entrada en el protocolo que uno o varios participantes, o una tercera parte, introducen sin darlos a conocer públicamente.
- **Objetivo**: finalidad con la que se ejecuta el protocolo, que es pública.
- **Procedimiento**: pasos a seguir por los participantes del protocolo.
- **Salida condicionada**: elemento que resulta de llevar a cabo el procedimiento, cuya construcción habrá sido definida en el objetivo.

Diremos que un protocolo de computación multiparte está *bien definido* si la única salida condicionada posible al aplicar el procedimiento, es la establecida en el objetivo.

Definición 1.3.2 En un protocolo multiparte P con $\{P_1, ..., P_n\}$ participantes, diremos que un *adversario*, denotado por Adv, es un subconjunto $\{P_{i_1}, ..., P_{i_t}\}$ de participantes que actúan de forma coordinada.

- Si Adv ejecuta el protocolo tal y como se define en el procedimiento, pero intenta obtener información de las entradas privadas de los jugadores $P - Adv$, o de la salida condicionada sin que se verifique la condición, diremos que es un *adversario semi-honesto*.
- Si Adv ejecuta el protocolo de forma diferente a la descrita en el procedimiento, intentando alterar el resultado de la salida condicionada, diremos que es un *adversario malicioso*.

Definimos el *tamaño de un adversario*, y lo denotaremos por $size(Adv)$, como el número de participantes que actúan de forma coordinada. Es decir, un adversario $Adv = \{P_{i_1}, ..., P_{i_t}\}$ diremos que es de tamaño t, o $size(Adv) = t$.

Definición 1.3.3 La *privacidad* de un protocolo multiparte se define como la imposibilidad de obtención de ninguna información de las entradas privadas, así como de la salida condicionada sin el cumplimiento de la condición, por parte de un adversario semi-honesto. Diremos que un protocolo multiparte es *t-privado*, cuando es inmune a la presencia de adversarios de tamaño menor o igual a t.

Definición 1.3.4 La *seguridad* de un protocolo multiparte se define como la imposibilidad de efectuar el procedimiento de forma incorrecta para alterar la salida condicionada, de forma que no se corresponda con la establecida en el objetivo. Diremos que un protocolo multiparte es *t-seguro*, cuando es inmune a la presencia de adversarios maliciosos cuya suma de tamaños es menor o igual a t.

1.3.1 Compartición de secretos.

La compartición de secretos es un tipo de protocolos de computación multiparte. Su principal objetivo es distribuir una participación del secreto a cada participante, de forma que no se revele ninguna información sobre el mismo,

y solo cuando un número predeterminado de participantes lo acuerden puedan revelarlo. Atendiendo a la definición de protocolo multiparte:

Definición 1.3.5 Un *protocolo de compartición de secretos* es un protocolo multiparte bien definido que cumple las siguientes características:

- Participantes: $\{P_1, ..., P_n\}$ participantes, $n \leq 2$.
- Entradas públicas: La descripción de un cuerpo finito $\mathbb{F}_q$, $\{\alpha_1, ..., \alpha_n\}$ nodos en $\mathbb{F}_q$, asociados a cada uno de los participantes.
- Entradas privadas: Un agente externo, que puede ser uno de los participantes, introduce un secreto $S \in \mathbb{Q}$.
- Objetivo: Cada participante recibe y almacena una participación $A_i \in \mathbb{F}_q$ que no revela información sobre el secreto S. Cuando t participantes así lo acuerdan, pueden recuperar el secreto S.
- Procedimiento: Depende de cada protocolo.
- Salida condicionada: El secreto S.

1.3.2 Computación multiparte.

La computación multiparte es el segundo bloque de protocolos multiparte que estudiaremos. Su principal objetivo es permitir a los participantes efectuar un cálculo acordado sobre unas entradas privadas proporcionadas por dos o más participantes, sin que estas últimas tengan que ser reveladas.

Definición 1.3.6 Un *protocolo de computación multiparte* es un protocolo multiparte bien definido que cumple las siguientes características:

- Participantes: $\{P_1, ..., P_n\}$ participantes, $n \leq 2$.
- Entradas públicas: La descripción de un cuerpo finito $\mathbb{F}_q$, $\{\alpha_1, ..., \alpha_n\}$ nodos en $\mathbb{F}$ asociados a cada uno de los participantes, la descripción de una operación o cálculo *calc*.
- Entradas privadas: $k \geq 2$ participantes introducen secretos $S_{i_1}, ..., S_{i_k} \in \mathbb{F}_q$.
- Objetivo: Los participantes efectúan el cálculo descrito sobre las entradas privadas. Cuando t participantes así lo acuerden, pueden revelar el cálculo.
- Procedimiento: Depende de cada protocolo.
- Salida condicionada: El cálculo $calc(S_{i_1}, ..., S_{i_k})$.

2

Protocolos multiparte privados

La **privacidad** es la propiedad fundamental de los protocolos multiparte. En este segundo Capítulo definiremos protocolos de **compartición de secretos** y de **computación multiparte** que garanticen dicha propiedad. Un protocolo multiparte se considera privado si garantiza que ninguna información secreta puede ser obtenida por un grupo de adversarios semi-honestos de un determinado tamaño t, tal y como se recoge en la Definición 1.3.3 del Capítulo anterior.

En la primera Sección de este Capítulo, introducimos dos protocolos diferentes para compartir secretos, basados en la propuesta de esquema original de Shamir [8] y en su modificación en rampa [5]. En la segunda Sección presentamos protocolos para computación multiparte relativa a la combinación lineal y producto de entradas privadas, basados en la referencia [1].

Todos los protocolos privados que estudiaremos en este Capítulo basan su privacidad en las propiedades fundamentales de los polinomios univariados desarrolladas en la Sección 1.1.

2.1 Compartición de secretos privada.

El primer protocolo de compartición de secretos que vamos a estudiar permite compartir participaciones de un secreto S a n participantes. Este protocolo es t-privado, para $t < n$, lo que quiere decir que el secreto S no podrá ser conocido por ninguno de los participantes hasta que más de t así lo acuerden. A continuación se presenta el esquema de Shamir.

2.1.1 Esquema de Shamir.

Definición 2.1.1 El *esquema de Shamir* es un protocolo de computación multiparte que permite la compartición de un secreto S con un nivel de privacidad $t < n$. Su procedimiento es el siguiente:

1. El repartidor, que conoce el secreto $S \in \mathbb{F}_q$, define un polinomio de la siguiente forma:
$$p(x) = S + \sum_{i=1}^{t} a_i x^i \tag{2.1}$$
donde $p(0) = S \in \mathbb{F}_q$ es el secreto, $\deg(p) \leq t$ es el nivel de privacidad, y $a_1, ..., a_t \in \mathbb{F}_q$ son coeficientes elegidos aleatoriamente en el cuerpo.
2. El repartidor define n participaciones $A_1, ..., A_n$ evaluando en $p(x)$ los distintos nodos públicos, es decir:
$$A_i = p(\alpha_i), \ \forall i \in \{1, ..., n\}. \tag{2.2}$$
3. El repartidor distribuye de forma privada cada participación a cada jugador. Es decir, comparte A_i con P_i, $\forall i \in \{1, ..., n\}$.
4. Cuando $m > t$ participantes acuerden revelar el secreto, hacen una puesta en común de sus participaciones, y efectúan una interpolación de grado t sobre $t + 1$ participaciones cualesquiera en sus respectivos nodos, definiendo así el siguiente polinomio:
$$q(x) = interpol_t\{(\alpha_{i_1}, A_{i_1}), ..., (\alpha_{i_{t+1}}, A_{i_{t+1}})\}(x) \tag{2.3}$$
Para obtener el secreto, basta con evaluar el polinomio $p(x)$ en cero. Es decir, definen como salida del protocolo:
$$\tilde{S} = q(0) \tag{2.4}$$

Proposición 2.1.2 El esquema de Shamir es un protocolo de compartición de secretos bien definido.

Demostración. Para ver que el esquema de Shamir está bien definido, debemos comprobar que la salida condicionada tras aplicar el protocolo $\tilde{S}$ coincide con el secreto S. Tomemos el polinomio $p(x)$ definido en (2.1), y el polinomio $q(x)$ definido en (2.3). Tenemos que $\deg(p) = \deg(q) = t$, y además $p(\alpha_j) = q(\alpha_j)$ para $t + 1$ evaluaciones $\alpha_j \in \{\alpha_{i_1}, ..., \alpha_{i_{t+1}}\}$. Por tanto, por unicidad de polinomios, $p(x) = q(x)$. En particular, $p(0) = q(0)$, siendo por definición $p(0) = S$ y $q(0) = \tilde{S}$. Luego $S = \tilde{S}$, es decir que el protocolo está bien definido.

□

Observación 2.1.3 *El procedimiento para demostrar la privacidad de los distintos protocolos será plantear el peor de los casos posibles (worst-case scenario). Es decir, supondremos que existe un adversario del tamaño máximo permitido por el protocolo, y estudiaremos si con la información a su disposición es capaz de descubrir los elementos secretos. Esto es porque un adversario que controla menos participantes siempre tendrá menos información, y nunca tendrá una posición más ventajosa para lograr su objetivo.*

Proposición 2.1.4 El esquema de Shamir es un protocolo t-privado.

Demostración. Para ver que el esquema de Shamir es t-privado, debemos comprobar si un adversario semi-honesto que controla hasta t participantes es capaz de obtener S. Ese resultado es consecuencia directa de las propiedades de los polinomios: sea $Adv = \{P_{i_1}, ..., P_{i_t}\}$ un adversario semi-honesto, con $size(Adv) = t$. Al hacer el reparto de participaciones, Adv conocerá $\{A_{i_1}, ..., A_{i_t}\}$. Para averiguar el secreto S, el adversario tendría que ser capaz de hallar el polinomio $p(x)$ para poder evaluarlo en cero. Adv solo conoce t evaluaciones del polinomio $p(x)$, $\{(\alpha_{i_1}, A_{i_1}), ..., (\alpha_{i_t}, A_{i_t})\}$, y además el polinomio es de coeficientes aleatorios. Por tanto, al ser $\deg(p) = t$, y como consecuencia de la Proposición 1.1.3, tenemos que para cualquier $\beta \in \mathbb{F}_q$, existe $p_\beta \in \mathbb{F}_q[x]_{\leq t}$ tal que $p_\beta(0) = \beta$ y $p_\beta(\alpha_{i_j}) = A_{i_j}$, $\forall j \in \{1, 2, ..., t\}$. Esto significa que $f(0)$ podría ser cualquier elemento de $\mathbb{F}_q$ de forma equiprobable, concluyendo así que la información de la que dispone el adversario no revela ninguna información sobre el secreto S. □

2.1.2 Esquema en rampa.

En el esquema de Shamir, tanto el secreto S como las participaciones A_i son elementos del cuerpo $\mathbb{F}_q$. Sin embargo, puede interesar que las participaciones tengan un tamaño menor que el secreto. A continuación se propone un esquema privado que permite compartir un secreto $S \in \mathbb{F}_q^k$ mediante participaciones $A_i \in \mathbb{F}_q$, con un nivel de privacidad $t < n - k$.

Definición 2.1.5 El *esquema en rampa* es un protocolo de computación multiparte que permite la compartición de un secreto $\vec{S} \in \mathbb{F}_q^k$ con un nivel de privacidad $t < n - k$. Su procedimiento es el siguiente:

1. El agente externo, que conoce el secreto $\vec{S} = (S_1, ..., S_k) \in \mathbb{F}_q^k$, define un polinomio de la siguiente forma:

$$p(x) = \sum_{i=0}^{k-1} S_{i+1} x^i + \sum_{i=k}^{t+k-1} a_i x^i \tag{2.5}$$

   donde $\deg(p) = t + k - 1$, siendo t el nivel de privacidad, y $a_k, ..., a_{t+k-1} \in \mathbb{F}_q$ son coeficientes elegidos aleatoriamente en el cuerpo.
2. El agente externo define n participaciones $A_1, ..., A_n$ evaluando en $p(x)$ los distintos nodos públicos, es decir:

$$A_i = p(\alpha_i),\ \forall i \in \{1, ..., n\}. \tag{2.6}$$

3. El agente externo reparte de forma privada cada participación a cada jugador. Es decir, comparte A_i con P_i, $\forall i \in \{1, ..., n\}$.

4. Cuando $m > t + k - 1$ participantes acuerden revelar el secreto, hacen una puesta en común de sus participaciones, y efectúan una interpolación de grado $t + k - 1$ sobre $t + k$ participaciones cualesquiera en sus respectivos nodos, definiendo así el siguiente polinomio:

$$q(x) = interpol_{t+k-1}\{(\alpha_{i_1}, A_{i_1}), ..., (\alpha_{i_{t+k}}, A_{i_{t+k}})\}(x) \tag{2.7}$$

   Para obtener el secreto, basta con obtener los k primeros coeficientes del polinomio $q(x) = \sum_{i=0}^{t+k-1} q_i x^i$. Es decir, definen como salida del protocolo:

$$\tilde{S} = (q_0, ..., q_{k-1}) \tag{2.8}$$

Proposición 2.1.6 El esquema en rampa es un protocolo de compartición de secretos bien definido.

Demostración. Para ver que el esquema en rampa está bien definido, debemos comprobar que la salida condicionada tras aplicar el protocolo $\tilde{S}$ coincide con el secreto S. Tomemos el polinomio $p(x)$ definido en (2.1), y el polinomio $q(x)$ definido en (2.3). Tenemos que $\deg(p), \deg(q) \leq t + k - 1$, y además $p(\alpha_j) = q(\alpha_j)$ para $t + k$ evaluaciones $\alpha_j \in \{\alpha_{i_1}, ..., \alpha_{i_{t+k}}\}$. Por tanto, por unicidad de polinomios, $p(x) = q(x)$. En particular, $(S_1, ..., S_k) = (q_0, ..., q_{k-1})$, luego $\vec{S} = \tilde{S}$, es decir que el protocolo está bien definido.

□

Proposición 2.1.7 El esquema en rampa es un protocolo t-privado.

Demostración. Para ver que el esquema en rampa es t-privado, debemos probar que un adversario semi-honesto que controla hasta t participantes no es capaz de obtener información alguna sobre el secreto $\vec{S}$. En este caso, entenderemos los coeficientes del polinomio $p(x)$ como incógnitas de un sistema de ecuaciones $Ax = b$. Como el polinomio $p(x)$ es de grado menor o igual a $t + k - 1$, sabemos que tiene $t + k$ coeficientes $\{p_0, ..., p_{t+k-1}\}$, luego el sistema tiene $t + k$ incógnitas. Por otra parte, cada participación A_i que recibe Adv supone una ecuación $\sum_{j=0}^{t+k-1} \alpha_i^j p_j = b_i$. Como $size(Adv) = t$, el sistema lineal tiene t incógnitas. Podemos concluir por tanto que la matriz A tiene rango $\mathrm{rg}(A) = t$, y por el Teorema de Rouché-Frobenius [3], se trata de un sistema compatible indeterminado, y el conjunto de soluciones forma una variedad lineal de $\mathbb{F}_q^{t+k}$ de dimensión $t + k - \mathrm{rg}(A) = k$. Por tanto, existen q^k posibles soluciones al sistema, que coincide con el número de elementos del cuerpo al que pertenece el secreto $\vec{S} \in \mathbb{F}_q^k$. En consecuencia, Adv no obtiene ninguna información acerca del secreto.

□

Como podemos comprobar, en este esquema el nivel de seguridad t, y el número de participantes a superar para poder revelar el secreto $t + k$ no coinciden. Esto

significa que cualquier número de participantes $t + w$ con $t < t + w < t + k$ obtiene cierta información sobre el secreto. A mayor valor w, más información sobre el secreto extá expuesta. Por eso se conoce como esquema *en rampa*.

Corolario 2.1. Un posible adversario semi-honesto formado por $t + w$ participantes, con $w < k$, no puede obtener el secreto $\vec{S}$, pero sí puede reducir de q^k a q^{k-w} las posibilidades.

Demostración. De forma análoga a la demostración de la privacidad, el sistema de ecuaciones $Ax = b$ tiene ahora w ecuaciones más, luego el rango de la matriz A es $\text{rg}(A) = t+w$, y la variedad lineal de soluciones tiene dimensión $t+k-\text{rg}(A) = k - w$, existiendo por tanto q^{k-w} posibles soluciones. □

2.2 Computación multiparte privada.

En esta Sección se definen protocolos de computación multiparte para efectuar distintos tipos de combinaciones lineales, así como productos, de entradas secretas, garantizando la privacidad de los procedimientos. Las técnicas que se utilizan son las mismas que las empleadas en la Sección sobre compartición de secretos.

2.2.1 Combinaciones lineales privadas.

El primer protocolo de combinación lineal privada que estudiaremos tiene como salida un elemento del cuerpo, que resulta de la combinación lineal previamente definida de n entradas secretas, una por cada participante. Su esquema es bastante sencillo, muy similar al esquema de Shamir. Al igual que este último, se basa fundamentalmente en las propiedades de los polinomios univariados.

Definición 2.2.1 El *esquema de combinación lineal privado* es un protocolo de computación multiparte que permite a n participantes efectuar una determinada combinación lineal $\sum_{i=1}^{n} \lambda_i S_i$ de n elementos $S_1, ..., S_n \in \mathbb{F}_q$ conocidos de forma secreta por $P_1, ..., P_n$ respectivamente. Este protocolo es t-privado, para $t < n$, lo que quiere decir que ni las entradas privadas $S_1, ..., S_n \in \mathbb{F}_q$, ni la salida condicionada $\sum_{i=1}^{n} \lambda_i S_i$ podrán ser conocidas por ninguno de los participantes hasta que más de t así lo acuerden. A continuación se presenta el procedimiento:

1. Cada uno de los participantes P_i, que conoce uno de los secretos $S_i \in \mathbb{F}_q$, define un polinomio de la siguiente forma:

$$p_i(x) = S_i + \sum_{j=1}^{t} a_j^i x^j \tag{2.9}$$

   Tenemos de este modo que $p_i(0) = S_i$, para $i \in \{1, ..., n\}$, con $a_j^i \in \mathbb{F}_q$ coeficientes elegidos aleatoriamente en el cuerpo.

2. Cada participante P_i define n participaciones $A_i^1, ..., A_i^n$ evaluando en $p_i(x)$ los distintos nodos públicos, es decir:

$$A_i^j = p_i(\alpha_j),\ \forall i, j \in \{1, ..., n\}. \tag{2.10}$$

3. Cada participante P_i reparte de forma privada a cada participante P_j la participación A_i^j.
4. Al terminar el reparto, cada participante P_i tiene las n participaciones $\{A_1^i, ..., A_n^i\}$. A continuación, debe efectuar sobre ellas la misma combinación lineal definida para las entradas. Es decir, cada P_i calcula la siguiente participación:

$$B_i = \sum_{j=1}^{n} \lambda_j A_j^i \tag{2.11}$$

5. Por último, cuando $m > t$ participantes acuerden revelar la combinación lineal, hacen una puesta en común de sus participaciones B_i, y efectúan una interpolación de grado t sobre $t+1$ participaciones cualesquiera en sus respectivos nodos, definiendo así el siguiente polinomio:

$$p(x) = interpol_t\{(\alpha_{i_1}, B_{i_1}), ..., (\alpha_{i_{t+1}}, B_{i_{t+1}})\}(x) \tag{2.12}$$

   Para obtener el resultado de la combinación lineal, basta con evaluar el polinomio $p(x)$ en cero. Es decir, definen como salida del protocolo:

$$\tilde{S} = p(0) \tag{2.13}$$

Proposición 2.2.2 El esquema de combinación lineal privado es un protocolo de computación multiparte bien definido.

Demostración. Para ver que el esquema de combinación lineal privado está bien definido, debemos comprobar que la salida condicionada tras aplicar el protocolo $\tilde{S}$ coincide con la combinación lineal definida $\sum_{i=1}^{n} \lambda_i S_i$. Esto se deduce fácilmente de las Proposiciones 1.1.2 y 1.1.3. Con más detalle, tomemos el siguiente polinomio de grado t:

$$q(x) = \sum_{i=1}^{n} \lambda_i p_i(x) = \sum_{i=1}^{n} \lambda_i S_i + \sum_{i=1}^{n} \sum_{j=1}^{t} \lambda_i a_j^i x^j \tag{2.14}$$

La evaluación de este polinomio en cero coincide con la combinación lineal de los secretos, $q(0) = \sum_{i=1}^{n} \lambda_i S_i$. Ahora, veamos que cualquier participación B_i coincide con $q(\alpha_i)$. Según las ecuaciones (2.10) y (2.11):

$$B_i = \sum_{j=1}^{n} \lambda_j A_i^j = \sum_{j=1}^{n} \lambda_j p_j(\alpha_i) = q(\alpha_i) \tag{2.15}$$

Por tanto, podemos reescribir el polinomio $p(x)$ definido en la ecuación (2.12) como:

$$p(x) = interpol_t\{(\alpha_{i_1}, q(\alpha_{i_1})), ..., (\alpha_{i_{t+1}}, q(\alpha_{i_{t+1}}))\}(x) \tag{2.16}$$

Es decir, que $p(x)$ y $q(x)$ son dos polinomios de grado t que coinciden en $t+1$ evaluaciones, luego $p(x) \equiv q(x)$, y $\tilde{S} = p(0) = q(0) = \sum_{i=1}^{n} \lambda_i S_i$, concluyendo que el protocolo está bien definido.

□

Proposición 2.2.3 El esquema de combinación lineal privado es un protocolo t-privado.

Demostración. Para ver que el esquema de combinación lineal privado es t-privado, debemos comprobar en primer lugar si un adversario semi-honesto que controla hasta t participaciones es capaz de obtener alguno de los secretos $S_1, ..., S_n$, y en segundo lugar si es capaz de obtener la combinación lineal $\sum_{i=1}^{n} \lambda_i S_i$. Sea $Adv = \{P_{i_1}, P_{i_2}, ..., P_{i_t}\}$ un adversario semi-honesto, con $size(A) = t$. Al hacer el reparto de participaciones de un secreto S_j, Adv conocerá $\{A^j_{i_1}, A^j_{i_2}, ..., A^j_{i_t}\}$. Para averiguar el secreto S_j, el adversario tendría que ser capaz de hallar el polinomio $p_j(x)$ para poder evaluarlo en cero. Adv solo conoce t evaluaciones del polinomio $p_j(x)$, que además es de coeficientes aleatorios. Por tanto, al ser $\deg(p_j) = t$, por la Proposición 1.1.3, es imposible interpolar un único polinomio, luego el adversario no puede obtener S_j. La privacidad de la combinación lineal, es inmediata, puesto que no se comparte más información en el protocolo. Concluimos así que el protocolo es t-privado.

□

Una pequeña modificación del protocolo anterior nos permite introducir un esquema para efectuar una combinación lineal con las mismas n entradas privadas, pero de forma que solo uno de los participantes P_k conozca el resultado.

Definición 2.2.4 El *esquema de combinación lineal privado para un participante* es un protocolo de computación multiparte que permite a n participantes efectuar una determinada combinación lineal $\sum_{i=1}^{n} \lambda_i S_i$ de n elementos $S_1, ..., S_n \in \mathbb{F}_q$ conocidos de forma secreta por $P_1, ..., P_n$ respectivamente, y de forma que solo uno de los participantes P_k conozca el resultado de la misma. Este protocolo es t-privado, para $t < n$, lo que quiere decir que ni las entradas privadas $S_1, ..., S_n \in \mathbb{F}_q$, ni la salida condicionada $\sum_{i=1}^{n} \lambda_i S_i$ podrán ser conocidas por ninguno de los participantes hasta que más de t así lo acuerden (a excepción de P_k, que claramente conocerá $\sum_{i=1}^{n} \lambda_i S_i$) . A continuación se presenta el procedimiento, planteando las modificaciones al definido en 2.2.1:

1. Se efectuan los cuatro primeros pasos de forma idéntica. De este modo, cada participante P_i tiene la participación B_i de la ecuación (2.11).

2. Cada uno de los participantes P_i comparte con el participante P_k su participación B_i.
3. El participante P_k efectúa una interpolación de grado t sobre $t+1$ participaciones cualesquiera en sus respectivos nodos, definiendo así el siguiente polinomio:

$$p(x) = interpol_t\{(\alpha_{i_1}, B_{i_1}), ..., (\alpha_{i_{t+1}}, B_{i_{t+1}})\}(x) \tag{2.17}$$

Para obtener el resultado de la combinación lineal, basta con que evalúe el polinomio $p(x)$ en cero. Es decir, su salida privada del protocolo es:

$$\tilde{S} = p(0) \tag{2.18}$$

Proposición 2.2.5 El esquema de combinación lineal privado para un participante es un protocolo de computación multiparte bien definido y t-privado.

Demostración. La demostración es inmediata al ser análogo al protocolo de la Definición 2.2.1. □

El protocolo que acabamos de introducir nos permite garantizar que se puede efectuar una transformación lineal sobre un vector de entradas secretas $\vec{S} = (S_1, ..., S_n)$ basado en una matriz $A \in (\mathbb{F}_q)_{n\times n}$ a través de un protocolo multiparte.

Corolario 2.2.6 Existe un protocolo bien definido y t-privado que, dado un vector de n entradas secretas $\vec{S} = (S_1, ..., S_n)$ y una matriz pública $A \in (\mathbb{F}_q)_{n\times n}$ nos permite calcular $\vec{E} = (e_1, ..., e_n) = \vec{S}A$ de forma que cada participante P_i obtenga de forma privada el elemento e_i.

Demostración. El protocolo descrito se consigue simplemente en iterando n veces el protocolo anterior. □

2.2.2 Producto de entradas privado.

Después de estudiar cómo efectuar una combinación lineal de n entradas mediante un protocolo multiparte privado, puede parecer que se deduce de forma inmediata cómo proceder para efectuar el producto de n entradas. Sin embargo, encontramos dos problemas fundamentales con este planteamiento:

- **El aumento del grado del polinomio**. Al hacer el producto de k polinomios de grado m, se obtiene un polinomio cuyo grado puede ser hasta km. Si este grado es mayor o igual que el número de participantes, $km \geq n$, no se pueden interpolar sus participaciones de manera unívoca.

- **La construcción del polinomio**. Al obtener un polinomio de grado km como producto de k polinomios de grado m, los coeficientes del mismo no son aleatorios. Por ejemplo, quedan descartados los coeficientes de polinomios irreducibles. Este detalle puede suponer información añadida que permita a un adversario semi-honesto obtener las entradas privadas o la salida condicionada.

Para resolver los problemas, diseñamos un protocolo que controle el aumento del grado de los polinomios, y que garantice la construcción aleatoria de los mismos. Para lograrlo, requeriremos que los productos se realicen elemento a elemento, y que el nivel de seguridad sea $t < \frac{n}{2}$. Una vez hayamos definido dicho protocolo, podremos iterarlo para permitir el producto de más elementos.

Definición 2.2.7 El *esquema de producto de dos entradas privado* es un protocolo de computación multiparte que permite a n participantes efectuar el producto de dos elementos $a, b \in \mathbb{F}_q$ conocidos de forma secreta por Q_a y Q_b respectivamente (con $Q_a \neq Q_b \in \{P_1, ..., P_n\}$). Este protocolo es t-privado, para $t < \frac{n}{2}$, lo que quiere decir que ni las entradas privadas $a, b \in \mathbb{F}_q$, ni la salida condicionada $ab \in \mathbb{F}_q$ podrán ser conocidas por ninguno de los participantes hasta que más de t así lo acuerden. A continuación se presenta el procedimiento:

1. Los participantes Q_a y Q_b, que conocen las entradas privadas a y b respectivamente, definen cada uno de ellos un polinomio de grado t de la siguiente forma:
$$p_a(x) = a + \sum_{i=1}^{t} a_i x^i \tag{2.19}$$
$$p_b(x) = b + \sum_{j=1}^{t} b_j x^j \tag{2.20}$$
Tenemos de este modo que $p_a(0) = a$ y que $p_b(0) = b$, con $a_1, b_1, ..., a_t, b_t \in \mathbb{F}_q$ coeficientes elegidos aleatoriamente en $\mathbb{F}_q$.
2. Tanto Q_a como Q_b definen n participaciones $\{A_1, ..., A_n\}$ y $\{B_1, ..., B_n\}$ evaluando en $p_a(x)$ y $p_b(x)$ los distintos nodos públicos, es decir:
$$A_i = p_a(\alpha_i),\ \forall i \in \{1, ..., n\}. \tag{2.21}$$
$$B_i = p_b(\alpha_i),\ \forall i \in \{1, ..., n\}. \tag{2.22}$$
3. Tanto Q_a como Q_b reparten de forma privada cada participación a cada participante. Es decir, cada participante P_i recibe el par de participaciones $\{A_i, B_i\}$.
4. Cada participante P_i multiplica sus dos participaciones para definir un nuevo elemento privado $S_i = A_i B_i = p_a(\alpha_i) p_b(\alpha_i)$.

5. A continuación, cada participante P_i escoge $2t$ coeficientes aleatorios en el cuerpo $c_1^i, ..., c_{2t}^i \in \mathbb{F}_q$ para construir el siguiente polinomio $c_i(x)$:

$$c_i(x) = \sum_{j=1}^{2t} c_j^i x^j \tag{2.23}$$

   Estos polinomios $c_i(x) \in \mathbb{F}_q[x]_{\leq 2t}$ son nulos al evaluarlos en cero, $c_i(0) = 0$. Cada participante P_i evalúa su polinomio $c_i(x)$ en cada nodo público α_j, y comparte con el resto de participantes P_j la participación resultante $C_i^j = c_i(\alpha_j)$.
6. Al completar el paso anterior, cada participante P_i tiene en su poder la participación S_i, y el conjunto de n participaciones $\{C_1^i, ..., C_n^i\}$. Ahora P_i puede definir una nueva participación que resulte de la suma de todas las anteriores:

$$D_i = S_i + \sum_{j=1}^{n} C_j^i \tag{2.24}$$

7. El siguiente paso consiste en que los participantes utilicen el protocolo propuesto en el Corolario 2.2.6 para multiplicar de forma privada el vector de participaciones $\vec{D} = (D_1, ..., D_n)$ por la matriz $A = V_{\vec{\alpha}}^{-1} P V_{\vec{\alpha}}$ que se describe en la Proposición 1.1.9. Es decir, se computa de forma privada el vector $\vec{E} = (E_1, ...E_n) = \vec{D}A$, lo que resulta en que cada participante P_i obtiene de forma privada el elemento E_i.
8. Cuando $m > t$ participantes acuerden revelar el producto, hacen una puesta en común de sus participaciones, y efectúan una interpolación de grado t sobre $t + 1$ participaciones cualesquiera en sus respectivos nodos, definiendo así el siguiente polinomio:

$$p_{ab}(x) = interpol_t\{(\alpha_{i_1}, E_{i_1}), ..., (\alpha_{i_{t+1}}, E_{i_{t+1}})\}(x) \tag{2.25}$$

   Para obtener el producto ab, basta con evaluar el polinomio $p_{ab}(x)$ en cero. Es decir, se define como salida del protocolo:

$$\tilde{S} = p_{ab}(0) \tag{2.26}$$

Proposición 2.2.8 El esquema de producto de dos entradas privado es un protocolo de computación multiparte bien definido.

Demostración. Para ver que el esquema de producto de dos entradas privado está bien definido, debemos comprobar que la salida condicionada tras aplicar el protocolo $\tilde{S}$ coincide con el producto ab. Para ello, estudiemos los polinomios que definen las participaciones en cada paso, y veamos su valor en cero.

- En los pasos 1, 2 y 3, se definen los polinomios $p_a(x)$ y $p_b(x)$, dos polinomios de grado t cuyas evaluaciones en cero son a y b respectivamente.

- En el paso 4, aplicando la Proposición 1.1.2, podemos considerar que las participaciones S_i provienen de evaluar en α_i un polinomio $q(x) = a(x)b(x)$, de grado a lo sumo $2t$ y que evaluado en cero resulta en ab.
- En el paso 5 se definen n polinomios $c_i(x)$ de grado $\leq 2t$ con término independiente nulo. Es decir, todos ellos son nulos al evaluarlos en cero.
- En el paso 6, de nuevo por la Proposición 1.1.2, podemos considerar que las participaciones D_i provienen de evaluar en α_i un polinomio $f(x) = q(x) + \sum_{j=1}^{n} c_j(x)$. Por tanto, se trata de un polinomio de grado a lo sumo $2t$ y que evaluado en cero resulta en $f(0) = q(0) + \sum_{j=1}^{n} c_j(0) = ab$.
- En el paso 7, por la Proposición 1.1.9, podemos considerar las participaciones E_i como evaluaciones en α_i del polinomio $trunc_t f(x)$, que es trivialmente de grado t. El término independiente de un polinomio es invariante ante truncamientos, luego $trunc_t f(0) = f(0) = ab$.

Por tanto, $trunc_t f(\alpha_i) = p_{ab}(\alpha_i)$ para $n > t$ evaluaciones distintas, siendo además $\deg(trunc_t f) = \deg(p_{ab}) = t$, luego por unicidad del polinomio interpolador, $trunc_t f(x) \equiv p_{ab}$. En consecuencia, $\tilde{S} = p_{ab}(0) = trunc_t f(0) = ab$, concluyendo así que el protocolo está bien definido.

□

Proposición 2.2.9 El esquema de producto de dos entradas privado es un protocolo t-privado.

Demostración. Para comprobar que el esquema es t-privado, veamos si un adversario semi-honesto que controla hasta t participantes es capaz de obtener las entradas secretas a, b o el producto ab con la información de la que dispone. Sea $Adv = \{P_{i_1}, ..., P_{i_t}\}$ un adversario semi-honesto con $size(Adv) = t$. Estudiemos la información que adquiere el adversario en cada paso:

- Tras los pasos 1, 2 y 3, el adversario conocerá $\{A_{i_1}, ..., A_{i_t}\}$ y $\{B_{i_1}, ..., B_{i_t}\}$ participaciones de los secretos a y b respectivamente. Para averiguar dichos secretos, el adversario tendría que poder hallar los polinomios $p_a(x)$ y $p_b(x)$ y evaluarlos en cero. Adv solo conoce t evaluaciones de los polinomios $p_a(x), p_b(x)$, que además son aleatorios. Por tanto, al ser $\deg(p_a), \deg(p_b) \leq t$, por la Proposición 1.1.3, es imposible interpolar polinomios únicos, luego Adv no obtiene ninguna información de a ni de b.
- Tras el paso 4 no hay información nueva, puesto que lo único que hacen los participantes es multiplicar sus propias participaciones.
- Tras el paso 5, cada participante ha definido un polinomio aleatorio $c_i(x)$ de grado menor o igual a $2t$ con término independiente nulo, y ha repartido participaciones del mismo al resto de participantes. Igual que en el primer punto, el subconjunto de t participaciones conocida por el adversario $\{C^i_{i_1}, ..., C^i_{i_t}\}$ de cualquiera de estos polinomios aleatorios $c_i(x)$ de grado menor o igual a $2t$, no permite interpolación única. Estos son los polinomios que van a garantizar la aleatoriedad del polinomio final.

- Tras el paso 6 no hay información nueva, puesto que lo único que hacen los participantes es una combinación lineal de sus participaciones. El polinomio $f(x)$ que se obtendría de interpolar las participaciones es aleatorio, puesto que es suma de n polinomios aleatorios $c_i(x)$, cada uno de ellos elegido de forma secreta por un participante.
- En el paso 7 se ejecuta el protocolo propuesto en el Corolario 2.2.6, cuya privacidad ya demostramos. Este paso sería problemático si el polinomio $f(x)$ no fuese aleatorio, pues esa información añadida podría permitir al Adv interpolar el polinomio $trunc_t f(x)$ de grado menor o igual a t, con solo t participaciones. Sin embargo, como el polinomio original $f(x)$ es aleatorio, su truncamiento $trunc_t f(x)$ también lo es. Por ser aleatorio y de grado menor o igual a t, el adversario no puede interpolar $trunc_t f(x)$ con sus t participaciones $\{E_{i_1}, ..., E_{i_t}\}$. Es decir, que no puede evaluarlo en cero y en consecuencia, no obtiene ninguna información del producto ab.

Por todo ello, un adversario semi-honesto Adv con $size(Adv) = t$ no puede obtener las entradas a, b, ni el producto ab, luego el protocolo es t-privado. □

Hemos desarrollado un protocolo para el producto de dos entradas. No conviene plantear un protocolo para $k > 2$ entradas de forma conjunta por el problema del aumento del grado de los polinomios que se discute al inicio de la Sección. Sin embargo, se puede efectuar el producto de múltiples entradas privadas de forma iterativa sin necesidad de revelar el resultado de los pasos intermedios, como se detalla a continuación.

Corolario 2.2. Existe un protocolo de computación multiparte bien definido y t-privado que permite el producto de n entradas privadas $S_1, ..., S_n$.

Demostración. El protocolo descrito resulta de la iteración del protocolo para dos entradas de la Definición 2.2.7 con unas leves modificaciones:

1. Se efectúa el protocolo para dos entradas S_1, S_2 hasta el paso 7, sin efectuar el paso 8. Es decir, cada participante conserva E_i de forma privada.
2. Se toma una tercera entrada S_3 y se reparten participaciones como se describe en el paso 1.
3. Ahora cada participante tiene una participación de S_1S_2 y una participación de S_3, luego se puede ejecutar de nuevo el protocolo desde el paso 2 hasta el paso 7, obteniendo así nuevas participaciones E_i' que se corresponden con el producto $S_1S_2S_3$.
4. Iterar hasta agotar las entradas privadas, y por último efectuar el paso 8 con las participaciones finales.

Es claro que esto no pone en riesgo la t-privacidad del protocolo, puesto que en ningún momento se revelan las participaciones intermedias a medida que se efectúan los cálculos. □

3

Protocolos multiparte seguros

En el Capítulo anterior se estudian distintos protocolos multiparte que verifican la propiedad fundamental de la privacidad. Resulta natural que la siguiente propiedad a estudiar sea la **seguridad**. En este tercer Capítulo definiremos protocolos de **compartición de secretos** y de **computación multiparte** que garanticen la seguridad. Un protocolo multiparte se considera t-seguro si garantiza que el resultado del mismo no puede ser manipulado por un número de t participantes que ejecuten el procedimiento de manera errónea, como se recoge en la Definición 1.3.4 del primer Capítulo.

En la primera Sección de este tercer Capítulo se analizan en detalle los posibles problemas de seguridad de los protocolos privados, y se proponen soluciones para todos ellos. En la segunda Sección se recoge la aplicación de estas soluciones en un protocolo de compartición de secretos. Por último, en la tercera Sección se aplican a un protocolo de computación multiparte para la combinación lineal de entradas. Todas estas ideas están basadas en el artículo [1]. Los protocolos que se estudian en este Capítulo están basados en las propiedades de los polinomios univariados y bivariados que se estudian en la Sección 1.1, y en resultados de códigos lineales y de Reed-Solomon vistos en la Sección 1.2.

3.1 Problemas de seguridad de los protocolos privados.

La seguridad de un protocolo es una propiedad más deseable que la privacidad. Es por ello que nos interesa analizar en detalle los protocolos privados para encontrar en qué puntos son vulnerables frente a adversarios maliciosos, y modificarlos para conseguir construir protocolos seguros. Efectuando dicho análisis, llegamos a la conclusión de que son tres los puntos de vulnerabilidad de los protocolos privados: la *veracidad del secreto*, la *puesta en común de las participaciones*, y el propio *reparto de participaciones*. Estos tres problemas se estudian en las tres Secciones que se encuentran a continuación.

3.1.1 Veracidad del secreto.

El primer problema de seguridad es que cualquier secreto S que tenga que ser utilizado por un agente externo o un participante, ya sea al comienzo del protocolo o en un paso intermedio, es susceptible de ser alterado.

Si se trata de un secreto original, es decir, si un agente externo o un participante introduce en el protocolo un secreto que sólo él conoce, el problema no tiene solución, puesto que nada puede impedir que sea alterado. Es por ello que al garantizar la seguridad supondremos que los secretos originales que se comparten siempre son correctos.

Sin embargo, si se trata de un secreto intermedio que se genera en el proceso del protocolo, es decir de una participación, el problema si puede y debe ser abordado. Distinguimos dos casos:

- Si la participación va a ser revelada de forma inmediata, no es necesario abordar el problema, puesto que alterar el secreto en ese caso es equivalente a alterar la puesta en común de las participaciones. Este segundo problema es el que se trata en la Sección 3.1.2, haciendo uso de decodificación de Reed-Solomon.
- Si la participación no va a ser revelada, sino que se va a utilizar en un paso intermedio del protocolo, habrá que solventar el problema. Para ello, también se utilizarán propiedades de decodificación de Reed-Solomon sobre un mensaje, pero sin necesidad de revelarlo públicamente. Como este problema es exclusivo de los protocolos de computación multiparte para el producto, lo estudiaremos en detalle en la Sección 3.3.2.

3.1.2 Puesta en común de participaciones.

El segundo problema de seguridad al que nos enfrentamos es que las participaciones reveladas pueden no ser las correctas. Es decir, en el momento de una puesta en común de participaciones en un protocolo, los participantes pueden compartir elementos incorrectos, alterando así el resultado de la interpolación polinómica.

Este problema se resuelve fácilmente si se interpretan las participaciones compartidas como una palabra de un $(n, t+1)$-código de Reed-Solomon.

Proposición 3.1.1 Consideremos un protocolo multiparte con $p(x) = S + \sum_{i=1}^{t} a_i x^i$ un polinomio aleatorio utilizado para repartir n participaciones $\{P_j = p(\alpha_j) \mid 1 \leq j \leq n\}$ de un secreto $S \in \mathbb{F}_q$. En una puesta en común de participaciones, si $t < \frac{n}{3}$, se puede hallar el secreto correctamente aunque t de las participaciones reveladas por los participantes sean incorrectas.

Demostración. Sea $\vec{c} = (p(\alpha_1), ..., p(\alpha_n)) \in \mathbb{F}_q^n$ el vector formado por las participaciones de los n participantes. Es decir, $\vec{c}$ es un vector que tiene por coordenadas n evaluaciones de un polinomio de grado menor o igual que t. Por tanto, por la Definición 1.2.16, $\vec{c}$ es una palabra del $(n, t+1)$-código de Reed-Solomon sobre $\mathbb{F}_q$. Por la Proposición 1.2.17 sabemos que la distancia mínima de este código es $d = n - t$, y de la Proposición 1.2.12 se deduce que el código es t-corrector para todo $t < \frac{d}{2}$. De todo ello podemos concluir que el código es t-corrector si $t < \frac{n}{3}$. Es decir, que dadas n participaciones, aún si hasta t de ellas son incorrectas, podemos obtener $\vec{c}$ y por tanto el mensaje $\vec{m} = (S, a_1, ..., a_t)$ y con él, el secreto S. Además, podemos garantizar que esta operación se puede efectuar con un algoritmo de decodificación eficiente como el propuesto en la Definición 1.2.18. □

Con la Proposición anterior, podemos resolver el problema de la seguridad en cualquier puesta en común de participaciones. Para permitir esto, hemos requerido que el nivel de seguridad sea $t < \frac{n}{3}$. Veremos que para el resto de problemas no necesitaremos disminuir este umbral de seguridad.

3.1.3 Reparto de participaciones verificable.

El tercer y último problema de seguridad es que el reparto de participaciones puede ser manipulado. En cualquier etapa de los protocolos que requiera de un reparto de participaciones, se puede escoger un polinomio de grado superior a t. Eso supone que el resultado de evaluar en cero el polinomio obtenido puede variar en función del subconjunto de $t+1$ participantes que acuerden revelar el secreto.

Para solucionar este problema, vamos a introducir un protocolo de reparto de participaciones que utiliza las propiedades de los polinomios bivariados estudiadas en la Sección 1.1, de forma que los participantes puedan estar seguros de la honestidad del repartidor. Éste será el primer protocolo t-seguro del Capítulo, y se utilizará en el resto de protocolos cada vez que se necesite hacer un reparto de participaciones.

Observación 3.1.2 *Definiremos los protocolos seguros desde la posición de un participante honesto. Es decir, se describe la forma correcta de proceder de un participante que actúa honestamente, asumiendo que otros participantes pueden estar procediendo de otro modo. En algunos casos necesitaremos referirnos a una igualdad que debería cumplirse, pero cuyo valor puede haber sido modificado por un adversario malicioso. Para ello utilizaremos el símbolo* $\approx$.

Definición 3.1.3 El *esquema de reparto de participaciones seguro* es un protocolo multiparte que permite compartir participaciones de un secreto S con n participantes, con nivel de seguridad $t < \frac{n}{3}$. Su procedimiento es el siguiente:

1. El repartidor, que conoce el secreto $S \in \mathbb{F}_q$, define un polinomio univariado de la siguiente forma:
$$p(x) = S + \sum_{i=1}^{t} a_i x^i \tag{3.1}$$
donde $p(0) = S \in \mathbb{F}_q$ es el secreto, $\deg(p) \leq t$ es el nivel de seguridad, y $a_1, ..., a_t \in \mathbb{F}_q$ son coeficientes elegidos aleatoriamente en el cuerpo.
2. El repartidor utiliza ese polinomio univariado $p(x)$ para definir un polinomio bivariado $B(x, y)$ de la siguiente forma:
$$B(x, y) = S + \sum_{i=1}^{t} a_i x^i + \sum_{i=0}^{n} \sum_{j=1}^{m} c_{ij} x^i y^j \tag{3.2}$$
siendo $c_{ij} \in \mathbb{F}_q$ coeficientes aleatorios en el cuerpo. Es decir, $B(x, y)$ verifica que $\deg_x(B), \deg_y(B) \leq t$, y $B(x, 0) = p(x)$, por tanto $B(0, 0) = p(0) = S$.
3. El repartidor define n participaciones intermedias $(f_i(x), g_i(y))$, cada una de ellas compuesta por dos polinomios univariados de grado menor o igual a t obtenidos evaluando el polinomio bivariado $B(x, y)$ en los distintos nodos públicos de la siguiente forma:
$$\begin{aligned} f_i(x) &= B(x, \alpha_i),\ \forall i \in \{1, ..., n\}. \\ g_i(y) &= B(\alpha_i, y),\ \forall i \in \{1, ..., n\}. \end{aligned} \tag{3.3}$$
4. El repartidor comparte de forma privada cada participación intermedia con cada participante. Es decir, comparte $(f_i(x), g_i(y))$ con P_i, $\forall i \in \{1, ..., n\}$.
5. A continuación, cada par de participantes $\{P_i, P_j\}$ intercambian de forma privada sus polinomios univariados evaluados en el nodo del otro. Es decir, P_i envía $\{f_{ij} = f_i(\alpha_j), g_{ij} = g_i(\alpha_j)\}$ a P_j para cualquier $j \neq i$, y de forma análoga P_i recibe dos elementos $\{\overline{f_{ji}} \approx f_j(\alpha_i), \overline{g_{ji}} \approx g_j(\alpha_i)\}$ de cada participante P_j con $j \neq i$. Recordemos que por construcción de los polinomios:
$$\begin{aligned} f_i(\alpha_j) &= B(\alpha_i, \alpha_j) = g_j(\alpha_i) \\ f_j(\alpha_i) &= B(\alpha_j, \alpha_i) = g_i(\alpha_j) \end{aligned} \tag{3.4}$$
Por tanto, cada participante debe esperar que los dos valores que intercambia con cada uno de los participantes coincidan de forma cruzada. Luego P_i tiene que verificar que se cumpla que $f_{ij} = \overline{g_{ji}}$ y $g_{ij} = \overline{f_{ji}}$ para todo $j \neq i$.
6. Después de todos los intercambios privados y de que cada participante haya realizado sus verificaciones cruzadas, se comienzan a presentar quejas en un canal de comunicación público. Esto es, si el participante P_i comprueba que para un $j \neq i$ la verificación cruzada no se cumple, es decir $f_{ij} \neq \overline{g_{ji}}$ o $g_{ij} \neq \overline{f_{ji}}$, entonces anuncia públicamente la siguiente queja:
$$queja(i, j, f_{ij}, g_{ij}) \tag{3.5}$$
Es decir, si P_i presenta una queja sobre el participante P_j, tiene que revelar sus propios valores.

7. A medida que los participantes presentan quejas, el repartidor debe ir resolviéndolas en el mismo canal público. Cuando se encuentre con una queja de la forma $queja(i, j, \overline{f_{ij}}, \overline{g_{ij}})$ procederá como sigue:
   - Si $\overline{f_{ij}} = B(\alpha_i, \alpha_j)$ y $\overline{g_{ij}} = B(\alpha_j, \alpha_i)$, no da ninguna respuesta.
   - Si alguna de las dos desigualdades falla, revela los dos polinomios del participante P_j de forma pública:
   $$revelado(j, f_j(x), g_j(y)) \tag{3.6}$$
8. Una vez todas las quejas han sido presentadas y atendidas por el agente externo, se realiza una votación en la que cada participante P_i debe decidir si opta por *resolver* el protocolo, o por *no resolver* el mismo. Esta decisión se toma en base a las siguientes comprobaciones:
   a) Si P_i encuentra dos quejas cruzadas no coincidentes, esto es, se han anunciado $queja(j, k, u_1, v_1)$ y $queja(k, j, u_2, v_2)$ tales que $u_1 \neq v_2$ o $v_1 \neq u_2$, y el repartidor no ha publicado ninguna revelación para j ni para k, entonces optará por *no resolver* el protocolo.
   b) Si P_i encuentra sus polinomios revelados, independientemente de si son iguales a los que el agente externo le asignó en un primer lugar o no, entonces optará por *no resolver* el protocolo.
   c) Si P_i encuentra un $revelado(j, \overline{f_j}(x), \overline{g_j}(y))$, debe evaluar los polinomios $\overline{f_j}(x)$ y $\overline{g_j}(y)$ en su nodo público α_i, y comprobar que coinciden con sus polinomios evaluados en α_j. Si $f_i(\alpha_j) \neq \overline{g_j}(\alpha_i)$ o $g_i(\alpha_j) \neq \overline{f_j}(\alpha_i)$, entonces optará por *no resolver* el protocolo.

   Si P_i no se encuentra con ninguna de las situaciones anteriores, entonces optará por *resolver* el protocolo.
9. Después de la votación, si al menos $n - t$ participantes han optado por *resolver* el protocolo, cada participante P_i puede definir y dar por válida su participación final como la evaluación del polinomio $f_i(x)$ en cero.
   $$A_i = f_i(0) \tag{3.7}$$

   En caso contrario, se asume que el repartidor es malicioso y se dan las participaciones por inválidas.

Proposición 3.1.4 El protocolo anterior está bien definido y es t-seguro.

Demostración. Queremos probar que el protocolo está bien definido y es t-seguro. Veamos qué significa esto en función de la honestidad del repartidor.

Caso 1: El repartidor es honesto. Si el repartidor es honesto, el protocolo está bien definido y es t-seguro si garantiza que todos los participantes obtienen y aceptan una participación válida del secreto S, aún con la presencia de un adversario Adv con $size(Adv) = t$. Como el repartidor es honesto, todos los polinomios que reciben los participantes son correctos. La única forma en la que

Adv puede lograr que los participantes consideren sus participaciones inválidas, es haciendo que el número de participantes que opten por *resolver* el protocolo sea menor que $n - t$. El número de participantes maliciosos es a lo sumo t, siendo por tanto el número de honestos como mínimo $n - t$, luego solo hemos de probar que *Adv* no es capaz de conseguir que ninguno de ellos opte por *no resolver*. Atendiendo a las comprobaciones del paso 8, veamos por qué esto no puede suceder:

a) Como el repartidor es honesto, si se presentan dos quejas cruzadas no coincidentes, al menos una de ellas tiene que ser errónea, y por tanto la resolverá. De este modo, un participante no puede encontrar quejas cruzadas sin resolución.
b) Un participante honesto no puede encontrar sus polinomios revelados. La única forma de encontrar $revelado(i, \overline{f_i}(x), \overline{g_i}(y))$ es si P_i presenta una queja sobre otro participante P_j. Si P_i es honesto y presenta una queja $queja(i, j, f_{ij}, g_{ij})$, el repartidor honesto comprobará que $f_{ij} = f_i(\alpha_j)$ y $g_{ij} = g_i(\alpha_j)$, y por tanto no responderá a la queja.
c) Siempre que un participante honesto evalúe algún polinomio revelado, éste será válido. Por ser el repartidor honesto, todos los polinomios revelados $revelado(j, \overline{f_j}(x), \overline{g_j}(y))$ coincidirán con los polinomios de los participantes honestos en las evaluaciones de forma cruzada.

Por tanto, todos los participantes terminan el protocolo con $f_i(0)$ una participación válida del secreto S.

Caso 2: El repartidor no es honesto. Si el repartidor no es honesto y forma parte del adversario, entonces el interés de *Adv* solo puede ser que los participantes honestos den por válidas participaciones que no lo son. Es decir, el protocolo está bien definido y es t-seguro si garantiza que las participaciones se invalidan si son incorrectas, o se fuerza a un reparto de participaciones correctas.

Las participaciones se dan por válidas si $n-t$ participantes optan por *resolver* el protocolo. Suponiendo que *Adv* controla t participantes maliciosos que optarán por *resolver*, necesita el voto de $n - t - t$ participantes honestos. Es decir, al menos $t + 1$ participantes honestos han de optar por *resolver* el protocolo.

A continuación, vamos a suponer que $t + 1$ participantes honestos han optado por *resolver* el protocolo, y comprobaremos: primero, que ello supone que sus polinomios están definidos por un único polinomio bivariado $B(x, y)$ de grado a lo sumo t tal y como se recoge en la ecuación (3.3); y segundo, que esto garantiza que los polinomios del resto de participantes honestos también están bien definidos.

Supongamos que $t + 1$ participantes honestos $\{P_{i_1}, ..., P_{i_{t+1}}\}$ han optado por

resolver el protocolo. Esto quiere decir que han superado las comprobaciones a), b) y c) del paso 8.

- Haber superado la comprobación b) implica que en ningún momento sus polinomios $f_{i_j}(x), g_{i_j}(y)$ han sido revelados.
- Haber superado la comprobación a) implica que todas las quejas cruzadas han sido resueltas por el repartidor.

Estos dos puntos garantizan que cualquier par de participantes honestos $\{P_{i_j}, P_{i_k}\}$ que optaron por *resolver* el protocolo, recibieron en el paso 4 polinomios $f_{i_j}(x), f_{i_k}(x), g_{i_j}(y), g_{i_k}(y)$ que cumplen la validación cruzada de la ecuación (3.4). De no ser así, existirían dos quejas de la forma $queja(i_j, i_k, u_{i_j}, v_{i_j})$ y $queja(i_k, i_j, u_{i_k}, v_{i_k})$, con $u_{i_j} \neq v_{i_k}$ o $u_{i_k} \neq v_{i_j}$. Si el repartidor no hubiese resuelto ninguna de las dos quejas, ninguno de los $t+1$ participantes hubiesen optado por *resolver* el protocolo; y si hubiese resuelto alguna de ellas $revelado(i_j, \overline{f_{i_j}}(x), \overline{g_{i_j}}(y))$, ese participante P_{i_j} no hubiese optado por *resolver* el protocolo. Por tanto, los polinomios recibidos por los participantes $P_{i_1}, ..., P_{i_n}$ en el paso 4 conforman $t+1$ pares de polinomios univariados $\{f_{i_j}(x), g_{i_j}(y)\}$ que verifican que $f_{i_j}(\alpha_{i_k}) = g_{i_k}(\alpha_{i_j})$. Aplicando la Proposición 1.1.12, podemos garantizar que existe un único polinomio bivariado $B(x, y)$ con $deg_x(B), deg_y(B) \leq t$ tal que $f_{i_j}(x) = B(x, \alpha_{i_j})$ y $g_{i_j}(y) = B(\alpha_{i_j}, y)$, para cualquier $i_j \in \{i_1, ..., i_{t+1}\}$.

Veamos ahora que esto último implica que necesariamente los polinomios finales del resto de participantes honestos también se definen a partir del mismo polinomio $B(x, y)$ como se describe en la ecuación (3.3). Supongamos que existe un $m \in \{1, ..., n\}$ tal que P_m es un participante honesto, y que en el paso 4 ha recibido un polinomio $\overline{f_m}(x) \neq B(x, \alpha_m)$ (argumento análogo para $\overline{g_m}(y)$). Por tanto, el polinomio $\overline{f_m}(x)$ coincide con el polinomio $B(x, \alpha_m)$ a lo sumo en t puntos. En particular, podemos garantizar que para un $\alpha_{i_j} \in \{\alpha_{i_1}, ..., \alpha_{i_{t+1}}\}$, se tiene la desigualdad $\overline{f_m}(\alpha_{i_j}) \neq B(\alpha_{i_j}, \alpha_m)$, y por tanto, $\overline{f_m}(\alpha_{i_j}) \neq g_{i_j}(\alpha_m)$. Como los participantes P_{i_j} y P_m son honestos, al comprobar que sus verificaciones cruzadas no coinciden, presentarán las quejas $queja(i_j, m, f_{i_j}(\alpha_m), g_{i_j}(\alpha_m))$ y $queja(m, i_j, \overline{f_m}(\alpha_{i_j}), \overline{g_m}(\alpha_{i_j}))$. Sabiendo que P_{i_j} y otros t participantes honestos optan por *resolver* el protocolo, y volviendo a analizar las comprobaciones del paso 8:

- Haber superado la comprobación a), implica que la queja cruzada fue resuelta.
- Haber superado la comprobación b), implica que los polinomios de P_{i_j} no fueron revelados. Junto con lo anterior, podemos concluir que tuvo lugar $revelado(m, f_m(x), g_m(y))$.
- Haber superado la comprobación c), implica que $t+1$ participantes honestos validaron de forma cruzada los polinomios $f_m(x)$, $g_m(y)$. Es decir, $f_{i_k}(\alpha_m) = g_m(\alpha_{i_k})$ y $g_{i_k}(\alpha_m) = f_m(\alpha_{i_k})$ para todo $i_k \in \{i_1, ..., i_{t+1}\}$.

Por esto último, y de nuevo por la Proposición 1.1.12, podemos concluir que $f_m(x) = B(x, \alpha_m)$ y $g_m(y) = B(\alpha_m, y)$. Por tanto, como el participante P_m fija como participación intermedia los nuevos polinomios $f_m(x)$ y $g_m(y)$, cuando se opte por *resolver* el protocolo, su participación final $f_m(0)$ será correcta.

Con todo esto, podemos afirmar que, de resolverse el protocolo, todos los participantes honestos tendrán una participación final válida, es decir, que cualquier subconjunto de $t+1$ participaciones permite interpolar el mismo polinomio $p(x)$. □

Observación 3.1.5 *Como los protocolos seguros se construyen con el objetivo de fortalecer los protocolos privados, la privacidad no se pone compromiso.*

Estudiados los tres problemas de seguridad y sus soluciones, se construye de forma muy simple un protocolo multiparte para compartir secretos, y un protocolo multiparte para efectuar combinaciones lineales. Simplemente utilizaremos el protocolo de reparto de participaciones seguro de la Definición 3.1.3, y la decodificación final de las participaciones estudiada en la Proposición 3.1.1. Utilizaremos los términos *se reparten participaciones del secreto* S y *se reparten participaciones del polinomio* $p(x)$ de forma análoga, refiriéndonos en este último caso al polinomio de la ecuación (3.1).

3.2 Compartición de secretos segura.

A continuación propondremos un protocolo de compartición de secretos bajo la restricción de la *seguridad*. Este protocolo permite compartir un secreto S, garantizando que las participaciones que se reparten son correctas, e impidiendo que la puesta en común de las participaciones pueda ser manipulada.

Definición 3.2.1 El *esquema de compartición de secretos seguro* es un protocolo de computación multiparte que permite compartir un secreto S con un nivel de seguridad $t < \frac{n}{3}$. Su procedimiento es el siguiente:

1. El repartidor, que conoce el secreto $S \in \mathbb{F}_q$, define y reparte de forma privada participaciones $A_1, ..., A_n$ del secreto a cada participante $P_1, ..., P_n$ como se describe en el protocolo de la Definición 3.1.3.
2. Cada participante P_i recibe la participación verificada A_i. Cuando los participantes acuerden revelar el secreto, hacen una puesta en común de sus participaciones $\overline{A_1} \approx A_1, ..., \overline{A_n} \approx A_n$ y definen el vector $\vec{r} = (\overline{A_1}, ..., \overline{A_n})$, sobre el que efectúan una decodificación de Reed-Solomon como se describe en la Proposición 3.1.1. De este modo, se decodifica un polinomio $q(x) \in \mathbb{F}_q[x]_{\leq t}$, y se define como salida del protocolo su evaluación en cero, es decir:

$$\tilde{S} = q(0) \tag{3.8}$$

Proposición 3.2.2 El esquema de compartición de secretos seguro es un protocolo bien definido y t-seguro.

Demostración. De forma trivial, el paso 1 está bien definido y es t-seguro por la Proposición 3.1.4. Al completar este paso, cada participante tiene una participación verificada A_i del secreto S. En el segundo paso, se revelan estas participaciones $\overline{A_1}, ..., \overline{A_n}$. Aunque t participantes revelen participaciones erróneas, la Proposición 3.1.1 garantiza que serán corregidas en el proceso de decodificación. Por tanto, el polinomio resultante $q(x)$ verifica que $q(0) = S$.

□

3.3 Computación multiparte segura.

En esta Sección se proponen distintos protocolos multiparte para la computación de diferentes cálculos de forma *segura*, es decir, contemplando la posible presencia de adversarios maliciosos de tamaño $t \leq \frac{n}{3}$. Estos protocolos se basan en las soluciones a los problemas de seguridad tratados en la Sección 3.1.

3.3.1 Combinaciones lineales seguras.

A continuación propondremos un protocolo de computación multiparte para el cálculo de combinaciones lineales de entradas privadas, bajo la restricción de la *seguridad*. Este protocolo tiene que permitir calcular una combinación lineal de n secretos $S_1, ..., S_n$. Los problemas de seguridad a solventar siguen siendo garantizar que las participaciones que se reparten son correctas, e impedir que la puesta en común de las participaciones pueda ser manipulada. De nuevo haremos uso de la Definición 3.1.3, y de la Proposición 3.1.1 para obtener un protocolo inmune a la presencia de adversarios maliciosos de tamaño t.

Definición 3.3.1 El *esquema de combinación lineal seguro* es un protocolo de computación multiparte que permite a n participantes efectuar una determinada combinación lineal $\sum_{i=1}^{n} \lambda_i S_i$ de n elementos $S_1, ..., S_n \in \mathbb{F}_q$ conocidos de forma secreta por $P_1, ..., P_n$ respectivamente. Este protocolo es t-seguro, para $t < \frac{n}{3}$. A continuación se presenta el procedimiento:

1. El participante P_i, que conoce el secreto $S_i \in \mathbb{F}_q$, define y reparte de forma verificable participaciones $A_i^1, ..., A_i^n$ del secreto a cada participante $P_1, ..., P_n$ como se describe en el protocolo de la Definición 3.1.3.
2. Al terminar el reparto, cada participante P_i tiene n participaciones verificadas $\{A_1^i, ..., A_n^i\}$. A continuación, debe efectuar sobre ellas la misma combinación lineal definida para las entradas. Es decir, cada P_i calcula la siguiente participación:

$$B_i = \sum_{j=1}^{n} \lambda_j A_j^i \tag{3.9}$$

3. Por último, cuando $k > t$ participantes acuerden revelar la combinación lineal, hacen una puesta en común de sus participaciones $\overline{B_1} \approx B_1, ..., \overline{B_n} \approx B_n$, y definen el vector $\vec{r} = (\overline{B_1}, ..., \overline{B_n})$, sobre el que efectúan una decodificación de Reed-Solomon como se describe en la Proposición 3.1.1. De este modo, se decodifica un polinomio $q(x) \in \mathbb{F}_q[x]_{\leq t}$, y se define como salida del protocolo su evaluación en cero, es decir:

$$\tilde{S} = q(0) \tag{3.10}$$

Proposición 3.3.2 El esquema de combinación lineal seguro es un protocolo bien definido y t-seguro.

Demostración. El paso 1 está bien definido y es t-seguro por la Proposición 3.1.4. Al completarlo, cada participante tiene una participación verificada A_j^i de cada secreto S_j. En el segundo paso, cada P_i define una participación como la combinación lineal $B_i = \sum_{j=1}^n \lambda_j A_j^i$. Como estas participaciones están verificadas, por la Proposición 1.1.2 podemos garantizar que B_i es la evaluación en α_i de un polinomio $p(x) \in \mathbb{F}_q[x]_{\leq t}$ aleatorio que verifica que $p(0) = \sum_{j=1}^n \lambda_j S_j$. Por último, se ponen en común las participaciones $\overline{B_1}, ..., \overline{B_n}$. Aunque t participantes revelen participaciones erróneas, la Proposición 3.1.1 garantiza que serán corregidas en el proceso de decodificación. Por tanto, el polinomio resultante $q(x)$ coincide con $p(x)$, y verifica que $q(0) = \sum_{j=1}^n \lambda_j S_j$.

□

3.3.2 Productos de entradas seguro.

Analizando el protocolo de producto de dos entradas privado de la Definición 2.2.7, encontramos un problema añadido: en el desarrollo del proceso, cada participante genera secretos de forma privada que en ningún momento pasan por una puesta en común. Esto se debe a los pasos añadidos para solventar el aumento de grado y la no aleatoriedad de los polinomios intermedios. Para poder mantener el nivel de seguridad máximo que hemos trabajado en todo el Capítulo, es decir $t \leq \frac{n}{3}$, debemos analizar los pasos en los que se producen estos secretos intermedios, y conseguir garantizar que no se alteran.

Los primeros secretos intermedios del protocolo los encontramos en el paso 4, cuando cada P_i define la participación $S_i = A_i B_i$. Estas participaciones no pueden ser puestas en común, pues se daría más información sobre los secretos a y b. Para solventar esto, la única alternativa es introducir el reparto de subparticipaciones y utilizarlas para efectuar una corrección de Reed-Solomon sin revelar el mensaje a corregir. Esto significa que cada participante P_i, al recibir las participaciones A_i y B_i, tendrá que repartir a su vez subparticipaciones de cada una de ellas.

Introduciremos un subprotocolo, *reparto de subparticipaciones verificable*, que permita efectuar lo descrito. Este subprotocolo necesitará a su vez de un subprotocolo de *transformación lineal segura*.

Con todo ello, en el momento en el que cada P_i efectúe el cálculo de $S_i = A_i B_i$ podrá repartir subparticipaciones del mismo. Estas participaciones no serán verificables por proceder de un polinomio de grado $2t$, pero cada P_j sí que podrá verificar que la subparticipación del producto coincide con el producto de las subparticipaciones anteriores.

Los otros pasos problemáticos son los relativos a la aleatorización del polinomio. Aquí, la lista de problemas es extensa:

1. Los polinomios $c_i(x)$ que define cada P_i son de grado $2t$, luego no permiten un reparto de participaciones seguro.
2. El secreto que esconde cada uno de estos polinomios tiene que ser cero, pero nada impide que a cada P_i alterarlo.
3. El truncamiento también puede ser problemático, puesto que utiliza combinaciones lineales privadas con salida privada. Al no pasar por una decodificación de Reed-Solomon, un único participante malicioso podría introducir su participación errónea, que al combinarse linealmente alteraría la participación final de todos los participantes.

Es por esto que introduciremos un subprotocolo de *participaciones de productos verificables*, que aleatoriza y trunca el polinomio subyacente de forma simultánea. Este protocolo se basará en la compartición de polinomios aleatorios de grado t, de forma que pueda utilizarse el reparto de participaciones verificable. Además la construcción del subprotocolo impide alterar la evaluación en cero de los mismos. También se suprime la necesidad del uso de combinaciones lineales con salida privada. Este subprotocolo a su vez requiere de otro subprotocolo de *revelado de participaciones* que permita exponer participaciones de determinados participantes.

Por tanto, estudiaremos estos cuatro subprotocolos, para posteriormente poder definir un protocolo de producto de dos entradas seguro.

Transformación lineal segura.

El primer subprotocolo que vamos a construir nos permite efectuar transformaciones lineales de un vector de entradas privadas $\vec{x} = (S_1, ..., S_n)$, con cada coordenada S_i conocida de forma secreta por P_i, obteniendo como salida pública un vector $\vec{y} = \vec{x}A$, donde A es una matriz de $(\mathbb{F}_q)_{n\times m}$.

Definición 3.3.3 El *esquema de transformación lineal segura* es un protocolo de computación multiparte que permite multiplicar un vector de entradas secretas $\vec{x} = (S_1, ..., S_n)$ por una matriz pública $A \in (\mathbb{F}_q)_{n\times m}$, para obtener como salida $\vec{y} = (y_1, ..., y_m) = \vec{x}A$. Se trata de un protocolo t-seguro para $t < \frac{n}{3}$. Su procedimiento es el siguiente:

1. Cada participante P_i construye un polinomio aleatorio $p_i(x)$ de grado a lo sumo t, cuya evaluación en cero coincide con su secreto $p_i(0) = S_i$. A continuación hace un reparto de participaciones seguro de su secreto, tal y como se detalla en la Definición 3.1.3.
2. Cada participante P_i recibe de cada P_j participaciones verificadas de cada secreto $A_i^j = p_j(\alpha_i)$. Por tanto, P_i puede definir el vector $\vec{x_i} = (p_1(\alpha_i), ..., p_n(\alpha_i))$.
3. Cada participante P_i multiplica de forma privada su vector $\vec{x_i}$ por la matriz A, definiendo así el vector $\vec{y_i}$:

$$\vec{y_i} = \vec{x_i}A = (p_1(\alpha_i), ..., p_n(\alpha_i))A \tag{3.11}$$

   Detonaremos las coordenadas de este vector como $\vec{y_i} = (Y_1(\alpha_i), ..., Y_m(\alpha_i))$.
4. Cada participante revela su vector $\vec{y_i}$ de forma que se puede obtener la siguiente matriz de forma pública:

$$B = \begin{pmatrix} \longleftarrow & \overrightarrow{y_1} & \longrightarrow \\ & \vdots & \\ \longleftarrow & \overrightarrow{y_n} & \longrightarrow \end{pmatrix} = \begin{pmatrix} \overline{Y_1(\alpha_1)} & \cdots & \overline{Y_m(\alpha_1)} \\ \vdots & \ddots & \vdots \\ \overline{Y_1(\alpha_n)} & \cdots & \overline{Y_m(\alpha_n)} \end{pmatrix} \tag{3.12}$$

5. Se realiza una decodificación de Reed-Solomon, evaluando en cero el polinomio resultante, de cada una de las columnas de la matriz B. Es decir, se corrige y decodifica cada mensaje $b_j = (\overline{Y_j(\alpha_1)}, ..., \overline{Y_j(\alpha_n)})^\top$, y al evaluar el polinomio resultante en cero se define $Y_j(0)$.
6. Se establece como vector de salida pública $\vec{y_0} = (Y_1(0), ..., Y_m(0))$. Además, cada participante conserva como salida privada las participaciones recibidas en el paso 2, $\{p_1(\alpha_i), ..., p_n(\alpha_i)\}$.

Proposición 3.3.4 El protocolo de transformación lineal segura está bien definido y es t-seguro.

Demostración. Queremos garantizar que el vector de salida del protocolo $\vec{y_0} = (Y_1(0), ..., Y_m(0))$ coincide con el vector $\vec{y} = (y_1, ..., y_m) = \vec{x}A$, aún con la posible presencia de un adversario malicioso Adv de tamaño $size(Adv) = t$. Estudiemos cada uno de los pasos:

- En los pasos 1 y 2, cada participante P_i recibe una participación verificada de cada uno de los secretos S_j. Sabemos que el protocolo de la Definición 3.1.3 garantiza que $A_i^j = p_j(\alpha_i)$.

- Tras el paso 3, cada participante P_i tiene una participación $\vec{y_i}$ del vector $\vec{y}$, tal y como garantiza la Proposición 1.1.2. Es decir, cada coordenada $Y_j(\alpha_i)$ es la evaluación de un polinomio Y_j aleatorio de grado menor o igual a t que verifica que $Y_j(0) = y_j$.
- En el paso 4, al construir la matriz B se obtienen n participaciones del vector $\vec{y}$. Es decir, para cada coordenada y_j del vector $\vec{y}$, se tienen n participaciones $\{\overline{Y_j(\alpha_1)}, ..., \overline{Y_j(\alpha_n)}\}$.
- En el paso 5, como el protocolo contempla la existencia de un adversario malicioso Adv de tamaño $size(Adv) \leq t$ y por tanto puede que hasta t de las participaciones sean erróneas, se realizan m decodificaciones de Reed-Solomon. Esto lo garantiza la Proposición 3.1.1 al ser $t < \frac{n}{3}$.
- En el paso 6 se definen como salida pública los elementos $\{Y_1(0), ..., Y_m(0)\}$ resultantes de la decodificación de Reed-Solomon y evaluación en cero de cada polinomio, con la garantía de que $Y_j(0) = y_j$. La salida privada de cada P_i no es más que la colección de participaciones verificadas del paso 2.

Con todo esto, podemos concluir que el protocolo está bien definido y es t-seguro. □

Reparto de subparticipaciones verificable.

La idea del protocolo del reparto de subparticipaciones verificable es simple. Si todos los participantes tienen participaciones de un secreto, al revelarlas se puede efectuar una decodificación de Reed-Solomon para corregir hasta t errores, lo que implicaría primero calcular su síndrome para después obtener su error. Pues bien, este protocolo se basa en la idea de que se puede obtener el síndrome y el error de un mensaje sin necesidad de revelar públicamente el mensaje en sí, a través del reparto de subparticipaciones. De esta manera, sólo si se detecta que una participación es incorrecta, se utilizan las subparticipaciones para revelarla y corregirla.

Definición 3.3.5 El *esquema de reparto de subparticipaciones verificable* es un protocolo de computación multiparte que, dado un secreto S y n participaciones $A_1, ..., A_n$ del mismo, permite repartir subparticipaciones verificables de cada A_i. Esto quiere decir que cada participante P_i puede obtener una participación B_j^i de cada A_j y verificar si es correcta. Además en caso de ser incorrecta permite revelar A_j. Se trata de un protocolo t-seguro. Su procedimiento es el siguiente:

1. Cada participante P_i tiene una participación $A_i = p(\alpha_i)$ del secreto S, donde $\deg(p) \leq t$ y $p(0) = S$.
2. Los participantes ejecutan el protocolo de transformación lineal seguro de la Definición 3.3.3, introduciendo cada uno de ellos su participación A_i como entrada privada, y siendo la matriz a multiplicar $H^\top$ la matriz de paridad

traspuesta con respecto de la matriz generatriz $G = V_{\vec{\alpha}}$ del código de Reed-Solomon.

3. Al término del subprotocolo anterior, se obtiene como salida pública el vector $\vec{s} = (S_1, ..., S_{n-t-1}) = (\overline{A_1}, ..., \overline{A_n})H^\top$, donde $\overline{A_j} \approx A_j$. El vector $\vec{s}$ se corresponde con el síndrome del mensaje $\vec{s} = syn((\overline{A_1}, ..., \overline{A_n}))$. Además, cada participante P_i recibe de forma privada las subparticipaciones $A_1^i, ..., A_n^i$ de cada participación $A_1, ..., A_n$.
4. Cada participante P_i utiliza un decodificador por síndromes para obtener el vector de errores $\vec{e}$ partiendo de $\vec{s}$.
5. Cada participante P_i construye un vector $\vec{b_i} = (B_1^i, ..., B_n^i)$ de n coordenadas, donde cada B_k^i se define en función del siguiente análisis de cada coordenada e_k del vector de errores $\vec{e} = (e_1, ..., e_n)$:
   a) Si $e_k = 0$, cada P_i fija $B_k^i = A_k^i$.
   b) Si $e_k \neq 0$:
      - El conjunto de participantes pone en común las subparticipaciones $\{A_k^j \mid 1 \leq j \leq n\}$.
      - Se obtiene y se evalúa en cero el polinomio $p_k(x)$ mediante decodificación de Reed-Solomon.
      - Entonces, cada P_i define $B_k^i = p_k(0) - e_k = p(\alpha_k)$.
   c) Cada participante P_i define cada B_k^i como subparticipación de A_k.

Observación 3.3.6 *En el paso 4 del protocolo definido, se menciona el uso de un decodificador de Reed-Solomon basado en síndromes. Existen decodificadores eficientes que, dado un vector síndrome, es capaz de obtener el vector de errores. Sin embargo, no se estudian este trabajo debido a la limitación en extensión.*

Proposición 3.3.7 El esquema de reparto de subparticipaciones verificable es un protocolo bien definido y t-seguro.

Demostración. Queremos garantizar que al término del protocolo, cada participante P_i obtiene una participación B_k^i válida de cada participación A_k. Es decir, queremos ver que para cualquier $1 \leq i \leq n$, $B_k^i = q_k(\alpha_i)$, con $\deg(q_k) \leq t$ y $q_k(0) = A_k$. Veamos paso a paso la evolución del protocolo:

- Al completar el paso 3, los participantes tienen de forma privada un vector de subparticipaciones verificadas $(A_1^i, ..., A_n^i)$ de cada participación $\overline{A_k^i} \approx A_k^i$, y se conoce de forma pública el síndrome del vector $(\overline{A_1}, ..., \overline{A_n})$.
- Tras el paso 4, se obtiene el vector de errores de $(\overline{A_1}, ..., \overline{A_n})$ con respecto de la palabra del código $(A_1, ..., A_n)$.
- Veamos que la construcción del paso 5 es correcta:
  - Si el vector de errores $\vec{e}$ tiene su k-ésima coordenada nula, $e_k = 0$, significa que $\overline{A_k} = A_k$. Por tanto, como A_k^i es una subparticipación verificada de $\overline{A_k}$, es correcto tomarla como participación de A_k.

– Si el vector de errores $\vec{e}$ tiene su k-ésima coordenada no nula, $e_k \neq 0$, quiere decir que $\overline{A_k} = A_k + e_k$. Por tanto, al revelar la participación $\overline{A_k}$ mediante las subparticipaciones $A_k^1, ..., A_k^n$ y restar e_k, se obtiene la participación correcta A_k.

De esta forma, podemos garantizar que las subparticipaciones obtenidas de este protocolo son subparticipaciones verificadas de $A_1, ..., A_n$. La t-seguridad está garantizada por hacer uso de subprotocolos t-seguros. □

Revelado de participaciones.

Para el subprotocolo de participaciones de productos verificables, será necesario que entre los participantes puedan revelar la subparticipación de uno de ellos, posteriormente se explicará el motivo. En general, revelar la participación de un participante es sencillo, puesto que al tratarse de la evaluación de un polinomio, puede obtenerse como combinación lineal del resto de participaciones o evaluaciones, tal y como garantiza la Proposición 1.1.6. Por tanto, el subprotocolo que presentamos a continuación no es más que el protocolo de combinación lineal segura, sustituyendo el primer paso por un reparto de subparticipaciones verificable.

Definición 3.3.8 El *esquema revelado de participaciones* es un protocolo de computación multiparte que, dadas n participaciones $A_1, ..., A_n$ de un secreto S (definidas mediante evaluaciones de un polinomio de grado t), permite revelar una participación A_k sin revelar el resto de participaciones. Se trata de un protocolo t-seguro para $t < \frac{n}{3}$. Su procedimiento es el siguiente:

1. Cada participante P_i, que conoce la participación $A_i \in \mathbb{F}_q$ del secreto S, define y reparte de forma verificable subparticipaciones $A_1^i, ..., A_n^i$ a cada participante $P_1, ..., P_n$ como se describe en el protocolo de la Definición 3.3.5.
2. Al terminar el reparto, cada participante P_i tiene n subparticipaciones verificadas $\{A_i^1, ..., A_i^n\}$. A continuación, debe efectuar sobre ellas la combinación lineal, definiendo la siguiente participación:
$$B_i = \vec{\alpha_k} V_{\vec{\alpha}}^{-1}(A_1^i, ..., A_n^i) \tag{3.13}$$
3. Por último, los participantes hacen una puesta en común de sus participaciones $\overline{B_1} \approx B_1, ..., \overline{B_n} \approx B_n$ y definen el vector $\vec{r} = (\overline{A_1}, ..., \overline{A_n})$, sobre el que efectúan una decodificación de Reed-Solomon como se describe en la Proposición 3.1.1. De este modo, se decodifica un polinomio $q(x) \in \mathbb{F}_q[x]_{\leq t}$, y se define como salida del protocolo su evaluación en cero, es decir:
$$\tilde{S} = q(0) \tag{3.14}$$

Proposición 3.3.9 El esquema de revelado de participaciones seguro es un protocolo bien definido y t-seguro.

Demostración. Las participaciones A_i están definidas por la evaluación de un polinomio de grado menor o igual a t en cada nodo α_i, luego $A_k = \vec{\alpha_k} V_{\vec{\alpha}}^{-1}(A_1^i, ..., A_n^i)$ por la Proposición 1.1.6. El protocolo es análogo al esquema de combinación lineal segura de la Definición 3.3.1, cambiando el primer paso por un reparto de subparticipación verificable como se describe en la Definición 3.3.5. Por tanto, podemos garantizar que está bien definido, $q(0) = A_k$, y que es t-seguro. □

Participaciones de productos verificables.

El subprotocolo que se introduce a continuación permite compartir participaciones verificables del producto de dos secretos, de forma que se puede garantizar que están definidas por un polinomio de grado t. Para ello, cada participante conocerá participaciones verificadas A_i y B_i de los secretos a y b. Entonces, el repartidor compartirá participaciones de polinomios de grado t, de forma que se puedan construir participaciones de un polinomio de grado t que evaluado en cero revela ab. Además, para el correcto funcionamiento de este protocolo, es necesario poder forzar el revelado de las participaciones de un participante determinado, puesto que el reparto de secretos verificable no es suficiente para garantizar que el polinomio cumple con la descripción. En este trabajo hemos podido demostrar que el número mínimo de polinomios necesario para hacer esto sin poner en riesgo la privacidad ni la seguridad es $\lceil \frac{t}{2} \rceil$. Se introduce el protocolo, suponiendo que t es par, puesto que facilita su seguimiento, y se puede formular de forma análoga para t impar.

Definición 3.3.10 El *esquema de participaciones de productos verificable* permite a n participantes $P_1, ..., P_n$ que tienen participaciones verificadas A_i y B_i de los secretos a y b, obtener participaciones verificadas de un producto ab con un polinomio de grado t subyacente. Su procedimiento es el siguiente:

1. El participante Q_c, que conoce los secretos a y b, define el polinomio $p(x) = p_a(x)p_b(x)$. Podemos escribir este polinomio como sigue:
$$p(x) = ab + \sum_{i=1}^{2t} p_i x^i \tag{3.15}$$
2. El participante Q_c elige $\{d_j^k \mid 0 \leq j \leq t-2, 0 \leq k \leq \frac{t}{2} - 1\}$ elementos aleatorios del cuerpo, y define los $\frac{t}{2}$ polinomios siguientes:
$$d_k(x) = \sum_{j=0}^{t-2} d_j^k + (p_{2(t-k)-1}x^j + \sum_{i=0}^{k-1} d_{t-2(k+i)-1}^i)x^{t-1} + (p_{2(t-k)}x^j + \sum_{i=0}^{k-1} d_{t-2(k+i)}^i)x^t$$
A continuación, puede define el polinomio:
$$c(x) = p(x) - \sum_{k=0}^{t/2-1} x^{t-k} d_k(x) \tag{3.16}$$

3. El paticipante Q_c evalúa los polinomios para definir las participaciones $C_i = c(\alpha_i)$, $D_i^k = d_k(\alpha_i)$.
4. Cada participante P_i recibe las participaciones $C_i = c(\alpha_i)$, $D_{ik} = d_k(\alpha_i)$. De forma privada, efectúa el siguiente cálculo:

$$E_i = A_i B_i - \sum_{k=0}^{t/2-1} \alpha_i^{t-k} D_{ik} \tag{3.17}$$

5. A continuación, cada P_i comprueba si $C_i = E_i$. En caso de no verificarse la igualdad, presenta una queja $queja(i)$.
6. Tras el periodo de quejas, comienza el momento de resolución de quejas. En este caso, para cada $queja(j)$ presentada, los participantes van a comprobar si se trata de una queja verdadera o falsa. Para ello, van a revelar las participaciones $A_j, B_j, C_j, D_{j1}, .., D_{j\frac{t}{2}}$ haciendo uso del subprotocolo de la Definición 3.3.8.
7. Si alguna de las quejas encontradas resulta ser verdadera, los participantes revelan los secretos a y b mediante el subprotocolo de la Definición 3.2.1, y se calcula públicamente ab. En caso contrario, validan las participaciones E_i.

Proposición 3.3.11 El esquema participaciones de productos verificables está bien definido y es t-seguro.

Demostración. Hagamos un análisis de los pasos del protocolo:

En el paso 2, se definen polinomios con $t-1$ coeficientes aleatorios. Esto garantiza que un adversario malicioso que controla t participantes, y que por conocer t particiones de a y t particiones de b, no puede descubrir ninguno de ellos.

- En el paso 3, se define un polinomio $c(x)$. Este polinomio es de grado t por construcción.
- En el paso 6 se resuelven las quejas. Veamos que:
  - Si el repartidor es honesto, solo podrán encontrarse quejas de participantes maliciosos. Es decir, como mucho se darán t quejas, lo que no pone en riesgo la información.
  - Si el repartidor no es honesto, solo se pone en juego su información privada. Que además solo es revelada si no actúa honestamente.

Por todo lo anterior, y sabiendo que todos los subprotocolos utilizados son seguros, garantizar que el protocolo está bien definido y es t-privado.

Producto de dos entradas seguro.

Una vez estudiados todos los subprotocolos necesarios, resulta sencillo definir la estructura de un protocolo de computación multiparte seguro que permita multiplicar dos entradas privadas.

Definición 3.3.12 El *esquema de producto de dos entradas seguro* es un protocolo de computación multiparte que permite a n participantes calcular el producto de dos entradas secretas, $a, b \in \mathbb{F}_q$ conocidas por Q_a y Q_b respectivamente (con $Q_a \neq Q_b \in \{P_1, ..., P_n\}$). Este producto es t-privado, para $t < \frac{n}{3}$. Su procedimiento es el siguiente:

1. Los participantes Q_a y Q_b conocen las entradas privadas a y b respectivamente. Cada uno de ellos define y reparte participaciones $\{A_1, ..., A_n\}$ y $\{B_1, ..., B_n\}$ de forma verificable, como se describe en la Definición 3.1.3.
2. Cada participante P_i reparte las participaciones A_i y B_i. Para cada una de ellas, cada P_i define y reparte subparticipaciones $A_i^1, ..., A_i^n$ y $B_i^1, ..., B_i^n$ de forma verificable como se describe en el subprotocolo de la Definición 3.3.5. Así, cada P_i recibe las subparticipaciones verificadas $\{A_1^i, B_1^i, ..., A_n^i, B_n^i\}$.
3. Cada participante P_i efectúa el producto de sus participaciones A_iB_i. Utilizando el subprotocolo descrito en la Definición 3.3.10, obtiene una participación C_i del producto de sus participaciones, y reparte subparticipaciones verificables de la misma, $C_i^1, ..., C_i^n$. De este modo, cada participante P_i recibe subparticipaciones verificadas de la participación del producto de participaciones C_j de cada uno de los participantes P_j, es decir $\{C_1^i, ..., C_n^i\}$.
4. Por último, se emplea el protocolo de combinación lineal segura descrito en la Definición 3.3.1 para efectuar el cálculo $\sum_{k=1}^n \lambda_k C_k$, donde $\vec{\lambda} = (\lambda_1, ..., \lambda_n)$ se define como la primera fila de la inversa de la matriz de Vandermonde, $V_{\vec{\alpha}}^{-1}$. En la aplicación de este protocolo no es necesario el primer paso de reparto de participaciones, pues cada P_i utiliza las subparticipaciones verificadas $\{C_1^i, ..., C_n^i\}$ de los secretos $\{C_1, ..., C_n\}$ descritas en el paso anterior.
5. Se define la salida del subprotocolo anterior como salida final, $\tilde{S} = \sum_{k=1}^n \lambda_k C_k$, se define como salida final.

Proposición 3.3.13 El esquema de producto de dos entradas seguro es un protocolo de computación multiparte bien definido y t-seguro.

Demostración. Este protocolo se basa en una serie de subprotocolos que están bien definidos y que son t-seguros. Por ello, podemos garantizar que las subparticipaciones $\{C_1^i, ..., C_n^i\}$ que recibe cada P_i en el paso 3 son correctas, es decir que $interpol_t\{(\alpha_1, C_k^1), ..., (\alpha_n, C_k^n)\}(0) = C_k$ y $interpol_t\{(\alpha_1, C_1), ..., (\alpha_n, C_n)\}(0) = ab$. De forma análoga, garantizamos que $\tilde{S} = \sum_{k=1}^n \lambda_k C_k$, y por la Proposición 1.1.7, tenemos$\sum_{k=1}^n \lambda_k C_k = ab$. Luego el protocolo está bien definido y es t-seguro. □

Corolario 3.1. Existe un protocolo de computación multiparte bien definido y t-seguro que permite el producto de n entradas privadas $S_1, ..., S_n$.

Demostración. Completamente análoga a la demostración del Corolario 2.2, ejecutando cada iteración del protocolo hasta el paso previo a la puesta en común de participaciones del suprotocolo del paso 4. □

Conclusiones

En este capítulo de conclusiones, analizaremos las limitaciones identificadas y las posibles mejoras de la teoría expuesta en este trabajo, proponiendo posibles direcciones futuras de investigación.

En primer lugar, conviene hacer una referencia a la complejidad algorítmica de los protocolos. Algunos de los protocolos presentados en este trabajo, sobre todo los que garantizan la seguridad, involucran múltiples pasos y llamadas a otros subprotocolos. Sin embargo, como garantizan su seguridad por diseño, los participantes conocen que un intento de manipulación del proceso solo resultaría en exponer su condición de adversario malicioso. Por tanto, cabe esperar que los participantes con intenciones maliciosas actúen de forma honesta, y en consecuencia que muchos de los pasos descritos no lleguen a efectuarse en la práctica. Esto hace que para algunos protocolos no preocupe calcular exhaustivamente la complejidad algorítmica. Sin embargo, los protocolos más complejos, como el producto de entradas privado, sí que requieren de un número elevado de intercambios e iteraciones aún cuando no se producen actuaciones maliciosas. De forma intuitiva ningún protocolo expuesto preocupa en términos de eficiencia y rendimiento, pero sería conveniente realizar un estudio pormenorizado de las complejidades algunos de ellos.

Una forma sencilla de disminuir la complejidad de los protocolos sería flexibilizar el nivel de seguridad t. Este trabajo ha tenido por premisa garantizar el máximo nivel de seguridad posible en cada escenario, pero en algunos casos esto se consigue a costa de aumentar la complejidad, lo que se traduce en mayor costo computacional. Sería interesante evaluar la posibilidad de relajar los requisitos de seguridad en ciertos escenarios, permitiendo la creación de protocolos más simples y eficientes. La síntesis fundamental en este aspecto es, *a mayor seguridad, mayor complejidad*. Determinar el compromiso óptimo entre ambas es un desafío constante en Criptografía.

Otro punto a destacar es la limitación en la elección del cuerpo finito $\mathbb{F}_q$ en toda la teoría expuesta. Generalmente, en criptografía es interesante trabajar con cuerpos de orden muy pequeño, idealmente en $\mathbb{F}_2$. Esto se debe a que el costo de las manipulaciones aritméticas y el orden del cuerpo están directamente relacionadas, permitiendo en los cuerpos de orden pequeño simplificar protocolos y reducir la complejidad computacional. Sin embargo, todos los protocolos que se estudian en este trabajo tienen como requisito fundamental asignar a cada participante P_i un elemento distinto del cuerpo o nodo público $\alpha_i \in \mathbb{F}_q$. Se hace evidente por tanto que la elección del cuerpo $\mathbb{F}_q$ quede limitada por la necesidad de que su cardinal sea mayor que el número de participantes $q > n$. Sería conveniente estudiar formas de esquivar esta restricción.

Una línea de ampliación interesante es explorar la adaptación de los protocolos al uso de otros códigos. Los códigos de Reed-Solomon han demostrado ser eficaces en los protocolos de compartición de secretos, pero presentan ciertas limitaciones de rendimiento que motivan el interés de aplicar otros códigos en este contexto. Sin embargo, explorar la aplicación de otros códigos supone un reto, puesto que los códigos Reed-Solomon tienen propiedades muy adecuadas para este contexto, fundamentalmente las basadas en propiedades polinómicas, además de poder ser decodificados de forma eficiente.

Por último, una pequeña mejora que podría ser muy deseable en los protocolos de computación multiparte sería aumentar el nivel de privacidad de las entradas privadas con respecto al de las salidas. Hemos definido de forma teórica los adversarios como grupos de participantes de tamaño determinado y con intereses determinados, pero en la práctica el abanico es mucho más amplio. En computación multiparte, las entradas privadas contienen información sensible y confidencial que los participantes desean proteger, y pueden existir intereses por revelarlas. Por lo tanto, sería conveniente diseñar protocolos y técnicas que garanticen un mayor nivel de confidencialidad y privacidad de las entradas, evitando cualquier posible filtración de información durante la ejecución de los protocolos.

References

[1] ASHAROV, G. y LINDELL, Y. A Full Proof of the BGW Protocol for Perfectly-Secure Multiparty Computation. *Journal of Cryptology*, 2017, vol. 30(1), pp. 58–151. Disponible online en: https://eprint.iacr.org/2011/136.

[2] BORWEIN, P. y ERDÉLYI, T. *Polynomials and polynomial inequalities*, Springer Science & Business Media, 1995.

[3] CASTELLET, M. y LLERENA, I. *Álgebra lineal y geometría*, Reverté, 1991.

[4] JUSTESEN, J y HØHOLDT, T. *A course in error-correcting codes*, European Mathematical Society, 2004.

[5] KRENN, S. y LORÜNSER, T. *An Introduction to Secret Sharing: A Systematic Overview and Guide for Protocol Selection*, Springer International Publishing, 2023.

[6] PELLIKAAN, R.; WU, X.W.; BULYGIN, S. y JURRIUS, R. *Codes, cryptology and curves with computer algebra*, Cambridge University Press, 2017.

[7] RUDRA, A. *Error Correcting Codes: Combinatorics, Algorithms and Applications*, Lecture Notes. Disponible online en: http://www.cse.buffalo.edu/faculty/atri/courses/coding-theory/fall07.html.

[8] SHAMIR, A. How to share a secret. *Communications of the ACM*, 1979, vol. 22(11), pp. 612–613.

Coding Theory in Cryptography: Secret Sharing and Multiparty Computation

Manuel Toledo Tauler
Facultad de Ciencias • Sección de Matemáticas
Universidad de La Laguna
alu01012315525@ull.edu.es

Abstract

Two challenges in Cryptography are secret sharing, which aims at protecting information among multiple parties, and multiparty computation, which aims at performing computations on private information without revealing it.
In this work we explore the fundamental concepts, the underlying mathematical theory based on polynomials and Reed-Solomon codes, and describe various protocols. This poster showcases some of the most important elements.

1. Basic concepts

The following ideas are basic in order to understand the topic of this work.

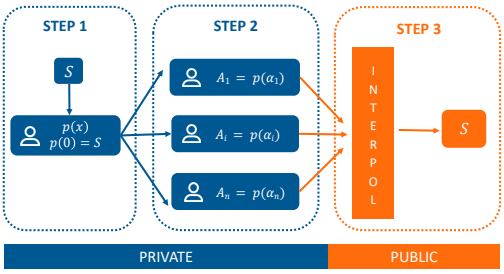
- A **secret sharing protocol** allows an external agent to distribute shares among different participants, ensuring that none of them can know the secret until a certain number of them agree to reveal it.
- A **multiparty computation protocol** enables operations on private inputs from multiple participants, such as linear combinations or products, without any of them having to disclose their secret information.

The two most desirable properties for these protocols are:

- **privacy**, which guarantees that private information cannot be known.
- **security**, which prevents corrupt adversaries from tampering with the processes.

in this work we propose private and secure protocols to tackle secret sharing and multiparty computation, which are based on **polynomial interpolation** and **Reed-Solomon codes**.

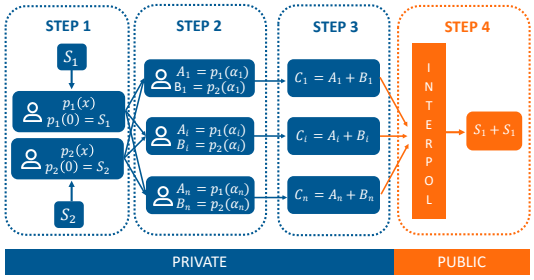
2. Secret sharing



Now we present a simplified version of a secret sharing t-private protocol. Given a field $\mathbb{F}_q$ and participants $P_1, ..., P_n$, $n \geq q$ with elements $\alpha_1, ..., \alpha_n \in \mathbb{F}_q$ assigned to each of them:

1. An external agent who knows a secret $S \in \mathbb{F}_q$, defines a random polynomial $p(x)$ such that $p(0) = S$ and $\deg(p) \leq t$.
2. Each participant is associated with an element of the field α_i. The external agent evaluates $p(x)$ at each of them, and secretly shares the evaluation with each participant, $A_i = p(\alpha_i)$.
3. When more than t participants agree, they reveal all their shares. By interpolating a t-degree polynomial, they obtain $p(x)$, and by evaluating it at zero, they successfully recover S.

3. Multiparty computation



We can define a multiparty computation protocol, similar to the one we just saw, to privately sum two secrets. This protocol utilizes basic properties of polynomials. Given a field $\mathbb{F}_q$ and some participants $P_1, ..., P_n$ with elements $\alpha_1, ..., \alpha_n \in \mathbb{F}_q$ assigned to each of them, the protocol proceeds as follows:

1. Two external agents, each knowing a secret S_1 and S_2 in $\mathbb{F}_q$, respectively, independently define random polynomials $p_1(x)$ and $p_2(x)$ such that $p_1(0) = S_1$ and $p_2(0) = S_2$, both with $\deg(p_1), \deg(p_2) \leq t$.
2. Each participant is associated with an element of the field α_i. The first external agent evaluates $p_1(x)$ at each participant's element and secretly shares the evaluation with them as $A_i = p_1(\alpha_i)$. Similarly, the second external agent evaluates $p_2(x)$ at each participant's element and shares the evaluation as $B_i = p_2(\alpha_i)$.
3. Each participant privately sums both shares to obtain $C_i = A_i + B_i$.
4. Finally, when more than t participants agree, they reveal all their summed shares C_i. By interpolating a t-degree polynomial from the shared values C_i, they obtain $p_1(x) + p_2(x)$. By evaluating $(p_1(0) + p_2(0))$, they successfully recover $S_1 + S_2$.

4. The role of Coding Theory

The two protocols we have just seen, both secret sharing and multiparty computation, ensure the privacy of inputs and outputs. However, they are not secure protocols. This means that some participants and/or the external agents can misbehave, altering the procedure and thereby interfering with the results of the processes. At this point, Coding Theory provides solutions. In this work, we specifically explore how Reed-Solomon codes allow for error correction and prevention of malicious actions.

References

[1] ASHAROV, G. & LINDELL, Y. A Full Proof of the BGW Protocol for Perfectly-Secure Multiparty Computation. *Journal of Cryptology*, 2017, vol. 30(1), pp. 58–151.

[2] SHAMIR, A. How to share a secret. *Communications of the ACM*, 1979, vol. 22(11), pp. 612–613.