GRAFOS EXPANSORES EN CRIPTOGRAFÍA

Expanding graphs in cryptography

DARÍO ALEJANDRO GARCÍA[1]

Recibido:17 de abril de 2018. Aceptado:28 de mayo de 2018

DOI: http://dx.doi.org/10.21017/rimci.2018.v5.n10.a47

Resumen

¿Cuál es la forma más segura de mantener en secreto un mensajé La respuesta obvia a esta pregunta es, por supuesto, dar el mensaje única y directamente al destinatario. Pero, ¿qué sucede si es necesario transmitir el mensaje a través de un medio en presencia de terceros? ¿Cómo podemos asegurarnos que el destinatario reciba el mensaje y, al mismo tiempo, prevenir que cualquier otra persona lo descubrá

Palabras clave: Criptografía, matemáticas, ciencias de la computación, seguridad.

Abstract

What is the surest way to keep a message secret? The obvious answer to this question is, of course, to give the message uniquely and directly to the recipient. But, what happens if it is necessary to transmit the message through a medium in the presence of third parties? How can we ensure that the recipient receives the message and, at the same time, prevent anyone else from discoveringit?

Keywords: Cryptography, mathematics, computer science, security.

I. Introducción

La criptografía es un área de las matemáticas y las ciencias de la computación cuyo objetivo es dar respuestas a las anteriores preguntas, haciendo que la intercepción de mensajes de parte de terceros no sea un problema. Para esto, el mensaje original es transformado a través de un proceso que se conoce como *encriptado*, convirtiéndolo en un mensaje que no guarda ningún tipo de relación con el mensaje original, y así transmitirlo al destinatario. El destinatario por su parte recibe el mensaje encriptado, y a través de un proceso de *desencriptado* podrá saber cuál era el mensaje original. Así, si el mensaje es interceptado por un tercero, o un adversario, este verá únicamente el mensaje encriptado, que no tiene ningún valor si no se conoce la técnica de desencriptado.

Consideremos el siguiente escenario. Supongamos que Tomania[2] y Krakozhia[3] son países limítrofes. Tomania, cansada de la neutralidad de Krakozhia, decide declararle la guerra. Al comando central krakozhiano le gustaría poder transmitir mensajes al frente sin que los tomaneses puedan ver su contenido. Por lo tanto, los krakozhianos encriptan sus mensajes antes de transmitirlos al frente, de tal suerte que, si estos son interceptados, los tomaneses no puedan conocer el contenido de los mensajes sin antes descifrarlos. De esta forma, el ejército krakozhiano puede transmitir planes de ataques, desplazamiento de tropas, y otros movimientos tácticos sin que los tomaneses puedan saberlo con suficiente anticipación para prevenirlos.

Este escenario de guerra ha sido utilizado en varias formas a través de la historia. Uno de los

1 Matemático de la Universidad Nacional de Colombia, Magíster en Matemáticas y Doctorado en Matemáticas de la Universidad de los Andes. Postdoctorado/Estancia postdoctoral UNIVERSITE CLAUDE BERNARD LYON 1 Mathématiques - Institut Camile Jordan. Postdoctorado/Estancia postdoctoral UNIVERSITY OF LEEDS. Marie Curie Fellowship. Correo electrónico: dagarciar@gmail.com

2 Nombre tomado de la película The Great Dictator, 1940, protagonizada por Charles Chaplin.

3 Nombre tomado de la película The Terminal, 2004, protagonizada por Tom Hanks.

ejemplos más antiguos de encriptado es conocido como el cifrado de César, usado por Julio César para transmitir mensajes a sus generales a los límites del imperio romano. Más recientemente, un proceso de encriptado sistemático fue usado con gran éxito por los alemanes en los primeros años de la segunda guerra mundial, y fue precisamente el descubrimiento de su técnica de desencriptado lo que abrió un camino a la victoria por parte de los aliados.

Por cada algoritmo de encriptado, viene alguien que trata de descrifrar el código. Si un adversario intercepta los mensajes continuamente, el proceso de encriptado puede romperse si se tiene el tiempo suficiente. El tiempo que toma desencriptar un mensaje puede tornarse muy largo, por lo que diseñar ataques más ingeniosos y específicos se vuelve una necesidad. El estudio de cómo romper procesos de encriptado se conoce como *criptoanálisis*, y como el lector puede imaginar existe una tensión constante entre criptoanálisis y criptografía: cuando una nueva técnica de encriptado es desarrollada por un criptógrafos, los criptoanalistas tratan de usar diferentes ataques para romperla. En respuesta a estos ataques, el proceso de encriptado se hace más fuerte, o se inventan nuevos procesos de encriptado, lo que hace que los criptoanalistas traten de crear nuevos y mejores ataques, y el ciclo se repite sucesivamente.

El encriptado de mensajes para propósitos militares todavía es muy importante, pero con el advenimiento de la internet, procesos de encriptados se han convertido en una necesidad esencial para todos aquellos que envían mensajes o realizan transacciones online. Por ejemplo, las personas que revisan sus cuentas bancarias o realizan compras con sus tarjetas de crédito a través de la red se benefician de procesos de encriptado porque esto les permite realizar dichas transacciones sin riesgos de robo o fraude.

En este artículo exploraremos el uso de una clase específica de grafos (conocidos como *grafos expansores*) en técnicas recientes de encriptado.

II. Nociones Fundamentales de la Teoría de Grafos

Para introducir el concepto de grafos expansores, es necesario explicar primero algunos conceptos del campo conocido como teoría de grafos. Los primeros pasos en esta área fueron dados por Leonhard Euler con su famosa solución al problema de los puentes de Kónigsberg, y desde ese entonces esta área de las matemáticas ha crecido y actualmente cuenta con aplicaciones en ciencias de la computación, física, química, e incluso linguística.

A. Conceptos básicos

Recordando las definiciones descritas en [2], un *grafo* es un par $G = (V, E)$ donde V es un conjunto de vértices y E es un conjunto de aristas, esto es, una colección de pares de vértices de la forma $e = \{v1,v2\} \subseteq V$. Dado un vértice $v \in V$, el grado de v (degv) es el número de aristas en E que contienen el vértice v, o equivalentemente, el número de vértices en V que son adyacentes a v.

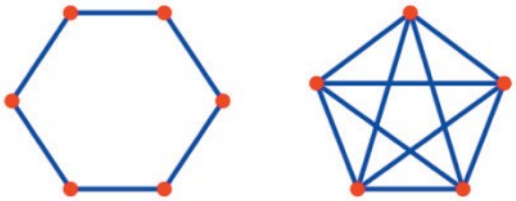

Fig. 1. Ciclo *C6* y grafo completo *K5*.

Definición 2.1. Un grafo $G = (V, E)$ se dice *k-regular* si cada vértice en V tiene grado k. Esto es, si cada vértice v tiene exactamente k vértices que son adyacentes a v.

Ejemplo 2.1. Por ejemplo, un ciclo como el de la Fig. 1 es 2-regular, mientras que un grafo completo Kn es (n - 1)-regular.

El uso de grafos k-regulares simplifica una relación importante en la teoría de grafos expansores. Adicionalmente, más que trabajar con grafos individuales, trabajaremos con *familias de grafos* que son simplemente colecciones de grafos que comparten ciertas características comunes. En términos de la conectividad de grafos, decimos informalmente que un grafo con muchas aristas es *denso*, mientras que un grafo con una cantidad relativamente pequeña de aristas se conoce como *esparso*.

Definición 2.3. Considere una colección de grafos $\{Gn = (V_n, E_n)\}n \in N$ donde el tamaño $|V_i|$ crece a medida que *n* tiende a infinito.

1. Decimos que $\{G_n\}_{n\in N}$ es una *familia de grafos densos* si existe una constante $C > 0$ tal que $|E_n| \geq C \cdot |V_n|^2$.
2. Decimos que $\{G_n\}_{n\in N}$ es una *familia de grafos esparsos* si existe una constante $c > 0$ tal que $|E_n| \leq c \cdot |V_n|$.

Note que dado un grafo con n vértices tiene a lo más $\binom{n}{2} = \frac{n^2-n}{2} \sim \frac{1}{2}n^2$ aristas. Por otra parte, la familia de grafos cíclicos $C = \{C_n = (V(C_n), E(C_n)) : n \in N\}$ y cualquier familia de árboles $T = \{T_n : n \in N\}$ son familias de grafos esparsos: tenemos que $|E(C_n)| = |V(C_n)|$ y $|E(T_n)| = |V(C_n)| - 1$, por lo que podemos tomar $c = 1$ en ambos casos.

Otro ejemplo de grafos esparsos viene dado por las familias de grafos k-regulares:

Proposición 2.4. *Para todo $k \in N$, cualquier familia $G = \{G_n : n \in N\}$ de grafos k-regulares es una familia de grafos esparsos.*

Demostración. Si cada grafo G_n es k-regular, entonces por el handshaking lemma (cf. Lema2.3 en [2]) tenemos que $2|E(G_n)| = \sum_{v\in V} \deg v = k \cdot |V|$, por lo que tomando $c=k/2$ podemos concluir que G es una familia de grafos esparsos.

En términos más simples, el número de aristas en un grafo esparso debería ser del orden del número de vértices, mientras que el número de aristas en un grafo denso es del orden del número de vértices al cuadrado.

A continuación introduciremos el concepto de caminatas a través de los diferentes vértices de un grafo.

Definición 2.5.

1. Un camino en un grafo es una sucesión $\langle v0, v1, \ldots, vn\rangle$ tal que para todo v_{i-1} es adyacente a v_i para todo i = 1,2,...,n. Decimos que la camino es cerrado si el vértice final es igual al vértice inicial, esto es, $v_0 = v_n$.
2. Dados dos vértices *u, v,* definimos la *distancia* entre *u y v* como la mínima longitud de todos los caminos que van de *u a v*, o ∞ si no existe dicho camino.
3. El *diámetro* de un grafo *G* se define como el máximo de las distancias entre cualesquiera par de vértices *u, v* de *G*.
4. La *cintura* (en inglés *girth)* es la longitud del ciclo de menor longitud en el grafo.

Por ejemplo, en los grafos C_6 y K_5 usados anteriormente, y más generalmente para el ciclo C_n y el grafo completo K_n, tenemos lo siguiente:

	C_6	K_5	C_n	K_n
Diámetro	3	1	$[\frac{n}{2}]$	1
Girth	6	3	n	3

B. Grafos de Cayley

Un grafo de Cayley es un grafo que contiene la estructura de un grupo.

Definición 2.6. Dado un grupo **G** y un subconjunto $S \subseteq G$, el *grafo de Cayley* $C_{G,S}$ se construye como sigue:

- Conjunto de vértices: por cada elemento $g \in G$, añadimos un vértice v_g.
- Por cada par de vértices v_{g1}, v_{g2}, E contiene una arista dirigida (v_{g1}, v_{g2}) si y sólo si existe $s \in S$ tal que $g_2 = g_1 * s$.

En este caso, denotamos los elementos de *S* como los *generadores* del grafo $C_{G,S}$ [8].

Ciertas propiedades del conjunto *S* determinan propiedades del grafo de Cayley $C_{G,S}$, como lo enuncia el siguiente resultado.

Proposición 2.7.

1. El grafo $C_{G,S}$ es conexo si y sólo si los elementos de S generan todo el grupo *G*.
2. El grafo $C_{G,S}$ es k-regular si y sólo si $|S| = k$.
3. El grafo es no dirigido cuando *S* es simétrico, esto es, si *S* contiene los inversos de sus propios elementos.

Demostración. (1) Supongamos que $C_{G,S}$ es conexo. Para mostrar que S genera todo el grupo G, es necesario mostrar que cada elemento $g \in G$ puede escribirse como un producto de elementos de S. Sea e el elemento neutro de G, y considere los vértices v_g, $v_e \in C_{G,S}$. Como $C_{G,S}$ es conexo, existe un camino $x_0 = v_e$, $x_1 = v_{g1},\ldots,x_n = v_{gn} = v_g$. Por definición, esto significa que existen elementos $s_1,\ldots,s_n \in S$ tales que $g_{i+1} = g_i * s_{i+}1$ para todo $i=0,\ldots,n-1$, y así, tenemos que $g=g_{n-1}*s_n = g_{n-2}*s_{n-1}*s_n = \cdots = e*s_1*\cdots*s_n = s_1 * \cdots * s_n$, esto es, g se puede escribir como producto de elementos de S.

Supongamos ahora que S es un conjunto de generadores de G, y tomemos dos vértices v_g, $v_h \in C_{G,S}$. Por hipótesis, existen elementos $s_1,\ldots,s_n \in S$ tales que $h^{-1} * g = s_1 * s_2 * \cdots * s_n$. Esto es, $g = h*s_1 * s_2 * \cdots * s_n$ y por la definición del grafo de Cayley tendríamos que v_h, v_{h*s1} , $v_{h*s1*s2}$, . . . , $v_{h*s1*s2*\cdots*sn} = v_g$ es un camino conectando los vértices v_g y v_h.

(2) Cada vértice $v_g \in C_{G,S}$ está conectado con los vértices en $N(v_g) = \{v_{g*s} : s \in S\}$. De esta manera, $\deg(v_g) = |S|$ para todo vértice v_g. Así, todo grafo de Cayley es $|S|$- regular, y en particular $C_{G,S}$ es k-regular si y sólo si $|S| = k$.

(3) Si S es un conjunto simétrico y $(v_g, v_h) \in E(C_{G,S})$, entonces $h = g * s$ para algún $s \in S$, por lo cual tendríamos $g^{-1} * h$-$s \Rightarrow g^{-1} = s*h^{-1} \Rightarrow g=(s*h^{-1})^{-1} = h*s^{-1}$, y dado que S es simétrico, $s^{-1} \in S$ y tenemos que $(v_h, v_g) \in E(C_{G,S})$. Esto muestra que el grafo $C_{G,S}$ es no dirigido.

Proposición 2.8. Los grafos de Cayley son transitivos, esto es, dados dos vértices v_{g1}, v_{g2} existe un *automorfismo* φ de $C_{G,S}$ tal que $\varphi(v_{g1}) = v_{g2}$.

Demostración. Dados v_{g1}, v_{g2}, considere el mapa $\varphi : C_{G,S} \to C_{G,S}$ dado por $\varphi(vx) = v_{g2*g1^{-1}x}$. Para mostrar que φ es un automorfismo de grupos, tenemos que probar que φ es una función biyectiva y que preserva aristas y no aristas.

- *El mapa φ es una función biyectiva: La función φ es sobreyectiva, ya que para todo v_g, tenemos que $\varphi(v_{g1*g^{-1}*g}) = v_{g2*g^{-1}*(g1*g^{-1}*g} = v_g$. Para ver que φ es inyectiva,* note que $\varphi(v_x) = \varphi(v_g) \Leftrightarrow g_2 * g_1^{-1} * x = g_2 * g_1^{-1} * y \Leftrightarrow x = y$.
- El mapa φ preserva aristas y no aristas: Por definición, $(\varphi(v_x),\varphi(v_y))$ es una arista si y sólo si existe $s \in S$ tal que $g_2 * g_1^{-1} * y = g_2 * g_1^{-1} * x * s$, que es equivalente a $y = x * s$.

Para finalizar, note que $\varphi(vg_1) = v_{g2*g1*g^{-1}} = v_{g2*e} = v_{g2}$, como queríamos.

Los grafos de Cayley también poseen propiedades interesantes con respecto a su diámetro. En particular, es posible mostrar que para grupos finitos no abelianos el diámetro de su grafo de Cayley es usualmente pequeño. De hecho, esto se sigue directamente de la conjectura de Bábai:

Conjetura 2.9 (Bábai). Existe una constante $c > 0$ tal que para cualquier grupo finite simple no abeliano G y para cualquier subconjunto simétrico $S \subseteq G$ que genere G,

$$\operatorname{diam}(C_{G,S}) < (\log |G|)^c.$$

III. Fundamentos en Criptografía

Antes de hablar de grafos expansores, nos concentraremos en introducir la noción de funciones resumen en criptografía. Esto será necesario para nuestra posterior discusión de funciones resumen y grafos expansores. Las funciones resumen son muy importantes en criptografía moderna, formando parte de muchos sistemas criptográficos de autenticación y con aplicaciones en generadores de números pseudoaleatorios, verificación de archivos y contraseñas.

A. Funciones resumen

Una *función* resumen es una función que transforma mensajes de tamaño arbitrario en va-

lores resumen que son textos de longitud corta y constante. Usando la codificación usual en sistema binario, podemos denotar una función resumen como una función de la forma:

$$H : \{0,1\}^* \rightarrow \{0,1\}^{\lambda}.$$

El uso de funciones resumen en criptografía se basa en el siguiente principio: *en casi todo alfabeto*, *la mayoría de los posibles mensajes no tienen sentido.*

Para explicar esto, podemos usar el idioma español, cuyo alfabeto tiene 27 letras y su diccionario contiene aproximadamente 88.000 palabras[4], la más larga de las cuales es electroencefalografista con 23 letras. Para homogeneizar, podemos suponer que hay 28 símbolos (incluyendo un nuevo símbolo *), y asumir que todas las palabras tienen longitud 23. Así, la palabra murciélago se traduciría como murciélago**********. Con esta información, tendríamos que hay un total de $28^{23} \approx 2^{110}$ posibles palabras (secuencias de letras de longitud 23), de las cuales sólo $88{,}000 \approx 2^{17}$ tienen significado real. En otras palabras, la proporción entre posibles palabras y palabras de verdad es de 2^{100}, que es comparable con el ¡número de granos de arena que hay en el planeta Tierra! Así, una función de resumen para el idioma español podría convertir una secuencia de 110 bits en una de 17 bits, manteniendo una traducción fiel del mensaje.

Muchas veces, especialmente cuando definimos protocolos de seguridad, estaremos interesados no sólo en funciones resumen, sino en familias de funciones resumen $\{H_n\}_{n\in N}$ parametrizadas por un *parámetro de seguridad n:*

$$H_n : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda(n)}$$

para alguna función λ(n).

Adicionalmente, puede ser útil definir una familia afinada de funciones resumen:

$$H_n : \{0, 1\}^{\kappa(n)} \times \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda(n)}$$

para funciones dadas *κ(n) y λ(n).* Si estamos trabajando con mensajes de longitud fija, la función de resumen se conoce como una *función resumen de longitud fija,*

$$H_n : \{0, 1\}^{\kappa(n)} \times \{0, 1\}^{\mu(n)} \rightarrow \{0, 1\}^{\lambda(n)}$$

para algunas funciones *κ(n), μ(n), y λ(n).*
Mientras que las funciones resumen comparten todas las características que mencionamos antes, estas pueden diferir en términos de las funciones usadas y en su construcción. Existen muchas posibles funciones resumen, no todas ellas fáciles de usar: estas pueden variar de fácil de construir pero inútil (como una función constante), o muy útil pero difícil o prácticamente imposibles de construir. Para procedimientos criptográficos, estamos interesados únicamente en funciones resumen que son fáciles de construir y criptográficamente útiles. Esto es, queremos una función resumen para la cual sea fácil de calcular el valor de resumen de un mensaje dado, y aún así ser criptográficamente seguro.

Los requerimientos de seguridad más comunes para funciones de resumen son los siguientes:

- *Resistencia de preimagen*: Para casi todos los valores de salida pre-especificados, es computacionalmente inviable encontrar un valor de entrada cuya función resumen es el valor original, esto es, dado cualquier valor resumen *h* es difícil encontrar cualquier preimagen *m'* tal que *H(m') = h.*
- *Resistencia de preimagen de segundo orden*: Dado un valor input y su correspondiente valor de resumen, es computacional inviable encontrar otro input con el mismo valor de resumen. Esto es, dado un input *m* es difícil encontrar $m' \neq m$ tal que *H(m) = H(m').*

4 158.000 palabras si incluímos las diferentes variantes en el diccionario de americanismos.

- *Resistencia a colisiones:* Es computacionalmente inviable encontrar dos inputs *m, m'* tal que *H(m) = H(m')*.

Es claro que resistencia a colisiones implica resistencia de preimagen y resistencia de preimagen de segundo orden.

B. Seguridad computacional

Una vez definidos los requisitos de un sistema de seguridad, la siguiente tarea es determinar el significado de que un proceso sea algo "computacionalmente inviable". Para dar una idea de este concepto, consideremos las funciones de resumen resistentes a colisiones que describimos a continuación:

- *Algoritmo* **A1** construye una base de datos de parejas $(m_i, H(m_i))$. Hasta que se encuentre una colisión, **A1** escoge un mensaje aleatorio m_i, calcula $h_i = H(m_i)$, chequea la base de datos buscando una ocurrencia previa de h_i (en cuyo caso se habrá encontrado una colisión). Si hi no ha aparecido, añadimos la pareja (m_i, h_i) a la base de datos. Si se encuentra una colisión, se devuelve los puntos de colisión.
- Algoritmo **A2** escoge dos mensajes aleatorios *m y m'* y devuelve *(m, m')* si *H(m) = H(m')*, ó devuelve ⊥ en otro caso. Esto es, ⊥ indica que los dos mensajes no tienen el mismo valor resumen.

El algoritmo **A1** siempre encuentra una colisión después de $2^{\lambda} + 1$ cómputos de funciones resumen, y en promedio demorará $2^{\lambda/2}$ cómputos, mientras que el algoritmo **A2** produce colisiones con probabilidad al menos $1/2^{\lambda}$. Podemos ver entonces que si λ se vuelve lo suficientemente grande, estos algoritmos se vuelven inviables ya que el algoritmo **A1** requerirá un tiempo muy largo y una cantidad excesiva de memoria, mientras que el algoritmo **A2** tiene una probabilidad de éxito muy baja ó casi insignificante. En la práctica, nos mantenemos en el contexto de criptografía con restricciones, por lo que algoritmos como los anteriores son desechados.

A continuación se presenta dos ejemplos algorítmicos que reflejan los principales enfoques en seguridad computacional, que trataremos de identificar y formalizar:

- *Enfoque concreto:* Un protocolo será *(ε, t, m)-seguro* si cualquier algoritmo que use una cantidad de tiempo menor a *t* y una cantidad de memoria menor que *m* tiene una probabilidad de éxito menor que *ε*.
- *Enfoque asintótico*: La familia de funciones resumen *{Hn}* es segura si cada H_n es *(ε(n),t(n),m(n))-segura*, las funciones *t(n) y m(n)* no crecen muy rápido con respecto a *n*, y la función *ε(n)* decrece rápido con respecto a *n*.

Definición 3.1.
Un algoritmo *A se dice eficiente ó tiempo polinomial probabilístico or PPT*[5] si puede ser solucionado en tiempo polinomial por una máquina de Turing probabilística. En otras palabras, si existe un polinomio *p* tal que para cualquier input $x \in \{0, 1\}^*$, el cómputo de *A(x)* termina en máximo *p(|x|)* pasos, donde *|x|* denota la longitud del input *x*.

Una función *f* se dice *insignificante* si para cualquier polinomio *p* existe un natural N tal que para todo entero $n \geq N$ se cumple que $f(n) < 1/p(n)$.

5 probabilistic polynomial tyme

Una función *f* se dice *f* se dice *notable* si existe un polinomio *p* tal que para todo entero suficientemente grande *n* se cumple que $f(n) > 1/p(n)$.

Es importante notar que un algoritmo PPT no puede ser usado en una máquina con una cantidad de memoria mayor a polinomial, y las nociones de insignificancia y notabilidad no son opuestos: hay funciones que no son ni insignificantes ni notables. Esto nos permite introducir una forma general de seguridad asintótica. Es así como muchos teoremas en criptografía tienen la siguiente forma:

Si la hipótesis computacional *X* se cumple, entonces el esquema criptográfico *Y* es seguro en el sentido computacional *Z*.

Como consecuencia, la pregunta de si el esquema *Y* es seguro se reduce a saber si la hipótesis *X* es cierta o no. Los siguientes son ejemplos ampliamente usados de hipótesis computacionales en criptografía:

- *Problema de Factorización Entera*: dado un entero grande $n = p \cdot q$ donde *p, q* son primos, es computacionalmente difícil encontrar *p y q*. Dicho algoritmo tendrá al menos $\sqrt{n}$n multiplicaciones, lo que se vuelve poco práctico para grandes valores de n.
- *Problema del Logaritmo Discreto*: dado un primo *p*, un elemento *g* de F_p de orden primo grande, y un elemento g^k mód *p* para algún entero *k* seleccionado aleatoriamente, es computacionalmente difícil encontrar el valor de *k.*
- *Problema de Logaritmo Discreto en Curvas Elípticas*: Dada una curva elíptica *E* definida sobre un campo primo F_p, un punto racional $P \in E$, y el punto $Q = k \cdot P$ para algún entero *k* seleccionado aleatoriamente, es computacionalmente difícil encontrar el valor de *k*.

Muchos algoritmos criptográficos basan su seguridad en el hecho de que no existen algoritmos que puedan resolver los problemas anteriores en una cantidad viable de tiempo.

C. La transformación de Merkle-Damgård

Tradicionalmente, las funciones criptográficas de resumen se construyen con dos componentes principales: se comienza con una función de resumen que comprime mensajes de una longitud fija en mensajes de otra longitud fija mucho más pequeña, y luego se usa una *extensión de dominio* que usa la función de compresión original y una división de bloques para construir funciones de resumen para mensajes arbitrariamente largos.

La transformada de Merkle-Damgård es un ejemplo común de una transformación de extensión de dominio y es utilizada para construir una variedad de familias de funciones resumen. Considere una función de compresión resistente a colisiones *f* que tiene como input una clave *s* y un mensaje de longitud $\mu+\lambda$, y retorna una cadena de bits de tamaño λ. La tranformada de Merkle-Damgård de *f* tiene como input una clave *s* y un mensaje *m* de longitud $m < 2^{\lambda}$, retornando una cadena de bits de longitud λ.

Para obtener resistencia a colisiones, usamos un proceso de fortalecimiento en nuestro mensaje *m*. Este proceso produce N+1 cadenas de bits $m_0,\ldots,m_N$ de tamaño μ, donde $N = \lceil L/\mu \rceil + 1$. El mensaje *m* se descompone primero en *N* bloques de μ bits consecutivos. Si *L* no es múltiplo de μ, el último bloque se completa con ceros. Añadimos también un bloque adicional que contiene una representación binaria de *L* en λ bits.

Sea h_0 un valor inicial fijo. La transformada de Merkle-Damgård de *f* se define como $H_f(s,m) = h_{N+1}$, donde $h_i = f(s,h_{i-1} \hat{} m_{i-1})$ y x ^ y denota la

concatenación de x and y. La transformada de Merkle-Damgård satisface la siguiente propiedad:

Teorema 3.2 (Merkle-Damgård). *Si (Gen,f) es una función de resumen resistente a colisiones de longitud fija, entonces (Gen,H) es una función de resumen resistente a colisiones.*

Demostración. Supongamos que un adversario encuentra m, m' tales que $H_f(s, m) = H_f(s, m')$. Si escribimos m_i, m'_i para los bloques correspondientes de m, m′, y h_i, h'_i para los valores intermedios en el cómputo de H_f (s, m) y H_f (s, m′). Entonces existen $i \leq n+1$ tales que $h_i = h'_i$ pero $h'_{i-1}\|m'_{i-1} \neq h'_{i-1}\|m'_{i-1}$. Por lo tanto, el adversario había encontrado una colisión $(h_{i-1}\|m_{i-1}, h'_{i-1}\|m'_{i-1})$ en la función de compresión, lo que es inviable ya que (G_{en},f) es una función resistente a colisiones.

La transformada de Merkle-Damgård es una transformada de extensión de dominio que preserva resistencia a colisiones: este proceso convierte funciones resumen de longitud fija resistentes a colisiones en fuciones resumen de longitud arbitraria resistentes a colisiones. Sin embargo, es importante notar que este proceso podría no preservar resistencia de preimagen de primer o segundo grado, y para obtener estar propiedades será necesaria cambiar otros aspectos en su construcción.

D. *Criptoanálisis de funciones resumen*

Un ataque a un protocolo criptográfico es una prueba de que el protocolo no satisface las propiedades de seguridad que se pensaban. Los ataques a funciones resumen tienden a contradecir sus propiedades de resistencia a preimágenes o a colisiones. En el contexto asintótico, un ataque teórica contra la resistencia a colisiones de una función resumen es un algoritmo PPT que encuentra colisiones para valores asíntoticamente grandes del parámetro de seguridad. En la práctica, sin embargo, muchas funciones están definidas únicamente en un conjunto pequeño de valores para su parámetro de seguridad.

La viabilidad de un ataque depende en gran medida de los recursos que el atacante está dispuesto a invertir en el ataque, y usualmente se define en términos de la complejidad de tiempo y memoria, a la cual añadimos la longitud de los mensajes resumidos. Para estándares DES del 2014, los ataques que corran en tiempo mayor a 2^{80} se consideran inviables, y ataques con requerimientos de memoria de 2^{60} ó 2^{70} bytes se consideran inviables por costos de almacenamiento. Por ejemplo, si un hard-drive de un Terabyte (= 2^{20} bytes) cuesta aproximadamente $250.000, para obtener 2^{30} bytes en almacenamiento necesitaríamos invertir 250.000 billones de pesos.

Como el codominio de cualquier función de resumen concreta es un conjunto finito, existen ataques genéricos inevitables que pueden ser mitigados únicamente escogiendo valores suficientemente grandes de los parámetros. En la práctica, un ataque de preimagen o un ataque de colisión se considera exitoso si calcula preimágenes o colisiones más rápido que los ataques genéricos. Para funciones de resumen iteradas, tales como las basadas en transformadas de Merkle-Damgård, existen ataques eficientes. A continación identificaremos diferentes ataques en funciones de resumen iteradas.

1) *Ataques genéricos*

Ataques de fuerza bruta. Dado un valor h de un mensaje seleccionado aleatoriamente, un adversario puede encontrar una preimagen m tal que $H(s,m) = h$ (donde la llave es conocida por el adversario) tratando valores sucesivos $m = 1, 2, \ldots$ hasta encontrar una preimagen. Si el output tiene tamaño λ, entonces se espera que este ataque tenga éxito en tiempo 2^{λ}.

El ataque de la paradoja del cumpleaños. Este ataque se basa en la paradoja probabilística que

describe la probabilidad de que dos personas tengan el mismo cumpleaños en un conjunto de n personas. Sabemos que por el principio del palomar, cuando $n = 367$ la probabilidad de que dos personas tengan el mismo cumpleaños será del 100 %. Sin embargo, se sabe que cuando $n = 23$ esta probabilidad es de aproximadamente 50.73 %, y cuando $n = 57$ la probabilidad es aproximadamente 99.01 %[6].

De forma similar, si el codominio de una función resumen tiene tamaño 2^λ, un adversario puede encontrar colisiones después de $2^{\lambda/2}$ cómputos de resúmenes de mensajes aleatorios. Si $N = 2^\lambda$ es el número de posibles resúmenes, la probabilidad de encontrar colisiones después de $N' = 2^{\lambda/2}$ ensayos aleatorios será de

$$\mathbb{P}[col] = 1 - \prod_{i=0}^{N'-1} \frac{N-i}{N} = 1 - \prod_{i=0}^{N'-1} \left(1 - \frac{i}{N}\right)$$

Usando la aproximación de Taylor de primer orden $e^x \approx 1 + x$ obtenemos

$$\mathbb{P}[col] \approx 1 - e^{\sum_{i=1}^{N'-1} \frac{i}{N}} \approx 1 - e^{\frac{(N')^2}{2N}} = 1 - \frac{1}{\sqrt{e}} \approx 0{,}39$$

Este tipo de ataque requiere memoria $2^{\lambda/2}$. Sin embargo, trasladando el problema de colisión en el problema de detectar ciclos en un mapa iterado, el requerimiento de memoria se vuelve insignificante mientras que el requerimiento de tiempo permanece casi igual. En esta versión modificada del ataque de cumpleaños, en vez de escoger los mensajes aleatoriamente, el adversario los escoge determinísticamente evaluando en un conjunto finito, que eventualmente se repetirá, resultando en ciclos. La principal ventaja de este enfoque es que no hay necesidad de almacenar los valores resúmenes.

2) Ataques a funciones resumen iteradas

Ataque de punto medio. Si la función de compresión es invertible, las pre-imágenes pueden calcularse en tiempo aproximadamente $2^{\lambda/2}$ extendiendo el ataque del cumpleaños de la siguiente manera: aplique la función de compresión a $2^{\lambda/2}$ mensajes aleatorios y calcule la inversa a otros $2^{\lambda/2}$ mensajes aleatorios. Por la paradoja del cumpleaños, existe una probabilidad alta de que el adversario encuentre un valor común "en el medio". Este ataque también tiene una versión libre de memoria, pero no es viable si la función de compresión es resistente a preimágenes.

Ataque de punto fijo. Este ataque busca un valor intermedio h_{i-1} en un bloque de mensaje m_i tal que $f(s,h_{i-1}\hat{}m_i) = h_i$. El ataque permite insertar cualquier número de bloques mi sin cambiar el valor de resumen. En una construción Merkle-Damgård, este ataque se vuelve muy práctico si el valor inicial puede ser seleccionado por el adversario.

Ataque de colisión múltiple. El ataque de la paradoja de cumpleaños permite encontrar una colisión a una función de compresión en tiempo $2^{\lambda/2}$. Repitiendo una búsqueda de colisión $\lambda/2$ veces, un adversario puede encontrar bloques de mensajes

$$m_1, m_1', m_2, m_2', \ldots, m_{\frac{\lambda}{2}-1}, m_{\frac{\lambda}{2}-1}', m_{\frac{\lambda}{2}}$$

Tales que

$$\begin{aligned} h_1 &:= f(s, h_0 \hat{} m_1) = f(s, h_0 \hat{} m_1'), \\ h_2 &:= f(s, h_1 \hat{} m_2) = f(s, h_1 \hat{} m_2'), \\ &\vdots \\ h_{\frac{\lambda}{2}-1} &:= f\left(s, h_{\frac{\lambda}{2}-1} \hat{} m_{\frac{\lambda}{2}}\right) = f\left(s, h_{\frac{\lambda}{2}-1} \hat{} m_{\frac{\lambda}{2}}'\right) \end{aligned}$$

Estos bloques de mensajes pueden combinarse en $2^{\lambda/2}$ mensajes de $\lambda/2$ bloques que tienen el mismo valor de resumen. Encontrar estas colisiones múltiples requiere únicamente un tiempo de $\lambda/2 \cdot 2^{\lambda/2}$ mientras que en una función ideal se requerirá un tiempo de $2^{\lambda\,(2\lambda/2-1)/2\lambda/2} \approx 2^\lambda$.

Esta observación puede ser usado para mejorar el ataque de paradoja de cumpleaños a una clase de funciones de resumen. Supongamos que G y

6 Un argumento similar explica por qué es casi imposible terminar una colección sin recurrir al intercambio con otros coleccionistas.

H son dos funciones de resumen con resistencia a colisiones ideal (esto es, se espera que el mejor ataque tenga un tiempo estimado de $2^{\lambda/2}$). En este caso, la función *F (s, m)* := *G(s, m)^H(s, m)* tendría una resistencia a colisiones ideal (el mejor ataque tiene un tiempo estimado de 2^{λ}). Sin embargo, si *G* es una función de resumen iterada, un adversario puede construir $2^{\lambda/2}$ colisiones para *G* en tiempo $\lambda/2 \cdot 2^{\lambda/2}$. Por la paradoja del cumpleaños, es probable que estos $2^{\lambda/2}$ mensajes nos den una colisión para *H*, y por lo tanto también para *F*.

Referencias

[1] Béla Bollobás. Random graphs. Cambridge Studies in Advance Mathematics. Cam- bridge University Press, 1985.

[2] Darío García. Grafos aleatorios y sus aplicaciones. Corporación Universitaria Republicana. 2016.

[3] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their appli- cations. Bulletin of the American Mathematical Society 43, no. 4, 439–561. 2006.

[4] Emanuel Kowalski. An introduction to expander graphs. ETH Zúrich. Noviembre 2017

[5] Alexander Lubotzky, Ralph Phillips, Peter Sarnak. Ramanujan graphs. Combinatorica 8, No. 3, 261-277. 1988.

[6] Abreu, A., Abreu, J., Iglesias, R., & Navarro, I. Interfaz gráfica en matlab para el cálculo de criterios de bondad de ajuste. Revista Ingeniería, Matemáticas Y Ciencias de La Información, 13–21. 2016.

[7] Franco, J. A. Estado del arte de la (in) seguridad VOIP. Rev. Ingeniería, Matemáticas y Ciencias de la Información, 77-98. 2013.