Avances en la función hash Tangle

Rafael Álvarez
Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante
Email: ralvarez@dccia.ua.es

Francisco Ferrández
Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante
Email: ferrande@dccia.ua.es

Julia Sánchez
Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante
Email: jsanchez@dccia.ua.es

Antonio Zamora
Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante
Email: zamora@dccia.ua.es

***Resumen*—Tangle es una función hash iterativa basada en el esquema Merkle-Damgard que fue sometida a la competición del NIST para el estándar SHA-3 y aceptada en primera ronda, mostrando un buen rendimiento respecto a otras propuestas. Desafortunadamente, se encontraron colisiones lo que provocó que no avanzara a la siguiente ronda.**

Presentamos el progreso realizado para mejorar la seguridad de Tangle y evitar los ataques conocidos manteniendo un nivel de rendimiento equivalente. Estas modificaciones se centran en las funciones de expansión, de ronda y de salida y se basan en los resultados obtenidos tras el criptoanálisis de la versión original de Tangle.

I. Introducción

Tangle es una función hash unidireccional basada en el esquema Merkle-Damgard (véase [3]) con la particularidad de contar con una función de ronda dependiente del mensaje en combinación con una caja de sustitución 8×8 (véase [1]) y una expansión de mensaje que utiliza un generador pseudoaleatorio matricial.

La función de compresión acepta de forma nativa un bloque de 4096 bits de longitud como entrada y produce, como salida, un resumen de 1024 bit. Soporta seis tamaños de resumen (224, 256, 384, 512, 768 y 1024 bits) mediante truncamiento; todas comparten la misma función de compresión pero difieren en el número de rondas y los valores iniciales.

Tangle soporta el mismo interfaz que SHA-2 (véase [7]), aceptando mensajes de hasta 2^{128} bits de longitud y rellenando de forma similar para obtener un mensaje con una longitud que sea múltiplo de 4096 bits.

A pesar de haber sido diseñado, principalmente, para microprocesadores de 32 bits con organización de memoria *little-endian*, ya que eran el escenario común cuando fue diseñado, permite ser implementado de forma satisfactoria en múltiples arquitecturas.

Fue sometido a la competición para el estándar SHA-3 del NIST (véase [8]) y aceptado para su evaluación en primera ronda, mostrando un alto rendimiento en relación a muchas de las otras propuestas. Desafortunadamente, se encontraron colisiones ([9]) lo que, a pesar de no ser un ataque demasiado útil en las aplicaciones prácticas de una función hash, fue considerado suficiente para no continuar a la segunda ronda.

Hemos trabajado desde entonces para mejorar el diseño de Tangle con el objetivo de evitar las debilidades encontradas durante su evaluación en la competición SHA-3. El esfuerzo se ha concentrado en tres áreas principales: las funciones de expansión del mensaje, de ronda y de salida. Detallamos los avances logrados en la mejora de Tangle, como una función de expansión que evita los ataques encontrados hasta la fecha y posibles mejoras en el diseño de la función de ronda y de salida.

II. Función de expansión del mensaje

Se describe a continuación la función de expansión del mensaje original de Tangle. Está basada en un generador pseudoaleatorio matricial simple que utiliza el mensaje de entrada como semilla y crea una secuencia de palabras de la longitud requerida por la función de compresión (4096 bits). Se puede encontrar una descripción más detallada de Tangle en la documentación oficial ([2]).

II-A. Expansión de mensaje original

La función de expansión expande las 128 palabras del bloque de mensaje M a $2R$ palabras (dos palabras por cada ronda de la función de compresión). Se utiliza un pequeño generador pseudoaleatorio basado en matrices para este cometido.

Se usan las siguientes funciones no lineales en la especificación:

$$\begin{aligned} F_1(x,y,z) &= (x \wedge (y \vee z)) \vee (y \wedge z) \\ F_2(x,y,z) &= (\neg y \wedge (x \vee z)) \vee (x \wedge z) \\ FR_1(x) &= \text{rotl}(x,3) \oplus \text{rotl}(x,13) \oplus \text{rotl}(x,29) \\ FR_2(x) &= \text{rotl}(x,5) \oplus \text{rotl}(x,19) \oplus \text{rotl}(x,27) \\ S_{box}(x) &= \text{inverso multiplicativo de } x \text{ en } GF(2^8) \end{aligned}$$

II-A1. Descripción del generador: El generador tiene 8 palabras de estado $(X_0, X_1, \ldots, X_7)$, y se itera siguiendo $X^i = A'X^{i-1}$ y $A' = PAP^{-1}$, donde

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix},$$

$$P = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$P^{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Como la matriz A se construye de forma que sea una matriz asociada a un polinomio primitivo, está garantizado que su período es $2^8 - 1$ o 255 iteraciones (véase [6]).

II-A2. Establecimiento de la semilla: El estado inicial del generador, X_j^0, con $j = 0, 1, \ldots, 7$, es una función de todo el bloque de mensaje M (con palabras de M_1 a M_{128}) y de las variables temporales del hash $h_0, h_1, \ldots, h_{31}$.

Para $j = 0$ a 7:

$$\begin{aligned} g_1 &= h_j + h_{j+8} \\ g_2 &= M_j + M_{j+8} + M_{j+16} + M_{j+24} \\ g_3 &= M_{j+32} + M_{j+40} + M_{j+48} + M_{j+56} \\ g_4 &= h_{j+16} + h_{j+24} \\ g_5 &= M_{j+64} + M_{j+72} + M_{j+80} + M_{j+88} \\ g_6 &= M_{j+96} + M_{j+104} + M_{j+112} + M_{j+120} \\ X_j &= FR_1(g_1 + g_2 + g_3) + FR_2(g_4 + g_5 + g_6) \end{aligned}$$

II-A3. Iteración y extracción: El generador se itera $R/2$ veces (siendo R el número de rondas necesarias para cada tamaño de resumen) siguiendo la expresión que se detalla a continuación. En cada iteración del generador, se producen cuatro palabras de mensaje expandido (W_t) que se extraen mediante una función de salida no lineal. K_i es la constante i como se define en la especificación original de Tangle (véase [2]).

Para $k = 1$ a $R/2$:

$$\begin{aligned} X_0^k &= X_0^{k-1} \oplus X_3^{k-1} \oplus X_6^{k-1} \oplus X_7^{k-1} \\ X_1^k &= X_0^{k-1} \oplus X_1^{k-1} \oplus X_3^{k-1} \oplus X_6^{k-1} \\ X_2^k &= X_0^{k-1} \oplus X_1^{k-1} \oplus X_3^{k-1} \oplus X_4^{k-1} \oplus X_5^{k-1} \\ X_3^k &= X_4^{k-1} \oplus X_5^{k-1} \oplus X_6^{k-1} \oplus X_7^{k-1} \\ X_4^k &= X_0^{k-1} \oplus X_2^{k-1} \oplus X_7^{k-1} \\ X_5^k &= X_2^{k-1} \oplus X_5^{k-1} \oplus X_6^{k-1} \oplus X_7^{k-1} \\ X_6^k &= X_2^{k-1} \oplus X_3^{k-1} \oplus X_4^{k-1} \oplus X_6^{k-1} \oplus X_7^{k-1} \\ X_7^k &= X_0^{k-1} \oplus X_4^{k-1} \oplus X_6^{k-1} \end{aligned}$$

$$\begin{aligned} t &= 4(k-1) \\ W_t &= F_1(X_0^k, X_1^k, K_{(t \bmod 256)}) \\ &\quad + M_{(t \bmod 128)} \\ W_{(t+1)} &= F_2(X_2^k, X_3^k, K_{(t+1 \bmod 256)}) \\ &\quad + M_{(t+1 \bmod 128)} \\ W_{(t+2)} &= F_1(X_4^k, X_5^k, K_{(t+2 \bmod 256)}) \\ &\quad + M_{(t+2 \bmod 128)} \\ W_{(t+3)} &= F_2(X_6^k, X_7^k, K_{(t+3 \bmod 256)}) \\ &\quad + M_{(t+3 \bmod 128)} \end{aligned}$$

II-B. Criptoanálisis

Este esquema de expansión está basado en un generador pseudoaleatorio y presenta los siguientes defectos en términos de seguridad:

- No contiene el mensaje original. La mayoría de las funciones hash incluyen el mensaje original dentro del mensaje expandido; de esa forma, es seguro que dos mensajes distintos producirán expansiones de mensaje diferentes.
- Presenta una avalancha muy pobre. El número de bits que cambian en la expansión del mensaje al cambiar un único bit en el mensaje de entrada es muy bajo. Un esquema de expansión de mensaje apropiado debería provocar que, de media, cambien un 50 % de los bits de salida cuando se varía un bit de la entrada.
- No aprovecha el hecho de que las diferencias al principio de la expansión del mensaje causan más cambios (avalancha) que aquellos al final; principalmente porque afectan a más rondas.

Estos tres criterios son, claramente, muy importantes puesto que pueden crear un camino para la existencia de ataques de colisiones satisfactorios.

- Si un esquema de expansión no incluye el mensaje original, entonces dos mensajes distintos pueden generar la misma expansión de mensaje, creando una colisión de forma automática.
- Por otra parte, una expansión de mensaje con una avalancha pobre permite un control mayor sobre el mensaje expandido, simplificando la tarea de provocar una colisión en la función de compresión.

- Finalmente, un esquema de expansión debe colocar las diferencias más significativas al principio de forma que sean procesadas por el mayor número de rondas posible maximizando, de esta forma, la avalancha de la función de compresión.

Estos fallos fueron observados por Esmaeili [4] y Thomsen [9] encontró una pareja de mensajes que colisionaban para todos los tamaños de resumen con un coste computacional relativamente pequeño.

El esquema de mensaje mejorado, descrito en el apartado siguiente, está motivado por estos mismos criterios: incluir el mensaje original en el mensaje expandido y optimizar la avalancha colocando las diferencias más significativas al principio de la expansión del mensaje.

III. Expansión de mensaje mejorada

III-A. Descripción

Tangle divide el mensaje de entrada en bloques de 4096 bits, consistiendo en 128 palabras de 32 bit cada una (M). La función de ronda usa dos palabras expandidas por ronda por lo que requiere tantas palabras de mensaje expandido (W) como el doble del número de rondas que utiliza cada tamaño de resumen. Con el objetivo de maximizar las diferencias en mensajes similares, se incluyen dos palabras pseudo-CRC (similar en concepto a un código de redundancia cíclica) de 32 bit que actúan como mini-hashes dificultando los ataques de colisiones. De esta forma, el mensaje expandido está formado por:

$$\text{PCRC}_1|\text{PCRC}_2|W_0|W_1|\ldots|W_n|M_0|M_1|\ldots|M_{127}$$

III-B. Los pseudo-CRC

El primer pseudo-CRC de 32 bit se calcula usando un código de suma ponderada aplicado a la detección de errores, empleando el algoritmo descrito para WSC-1 por McAuley [5]. Es similar a un CRC normal pero más rápido, actuando como un pequeño resumen del mensaje:

$$\text{PCRC}_1 = \text{WSC-1}(M_0 \ldots M_{127}).$$

El segundo pseudo-CRC es una variación del esquema de expansión del mensaje y se calcula junto con el mensaje expandido:

$$A = \text{PCRC}_1.$$

Para obtener cada palabra expandida (W_i con $i = n$ a 0) es necesario hacer:

$$\begin{aligned} W_i &= \text{ROTL}((W_{i+5} \oplus W_{i+9} \oplus W_{i+17} \oplus W_{i+128}), 1) \\ &\quad \oplus K_{S_{box}(A\oplus(A>>8)\oplus(A>>16)\oplus(A>>24))}, \\ A &= A \oplus W_i. \end{aligned}$$

Las palabras expandidas, W_i, se generan hacia atrás, de la última a la primera; y, al final, PCRC_2 es la última A. El símbolo $\oplus$ denota el XOR binario, el símbolo $>>$ denota el desplazamiento de bits a la derecha y ROTL significa rotación de bits a la izquierda. La expresión $K_{S_{BOX}()}$ implica que la constante K se elige en función de la salida de la caja de sustitución (SBOX).

Resumen	Rondas	Expansión	Ratio	Avalancha
224	72	144(14)	10.9 %	49 %
256	80	160(30)	23.4 %	50 %
384	96	192(62)	48.4 %	50 %
512	112	224(94)	73.4 %	50 %
768	128	256(126)	98.4 %	50 %
1024	144	288(158)	123.4 %	50 %

Tabla I
Resultados de avalancha para el esquema de expansión mejorado

III-C. Resultados

Este esquema de expansión proporciona resultados muy positivos en términos de avalancha, como se puede ver en la tabla I, y evita los ataques conocidos que rompen el diseño original de Tangle.

Donde *Expansión* es el tamaño total del mensaje expandido en palabras de 32 bits, mientras que el número de palabras expandidas W está en paréntesis; *ratio* es el porcentaje de palabras expandidas respecto al tamaño del mensaje; y *avalancha* significa el número de bits que cambian en el mensaje expandido cuando se varía un único bit en el mensaje original. La avalancha esperada es del 50 % y no se ha tenido en cuenta las palabras del mensaje original que se incluyen en el mensaje expandido para el cálculo de la misma.

En términos de rendimiento, la función de expansión de mensaje propuesta no añade un coste computacional significativo a la versión original, obteniéndose resultados igualmente satisfactorios, como se muestra en la tabla II.

Resumen	Rendimiento
224	10.79 cpb
256	12.53 cpb
384	15.94 cpb
512	19.36 cpb
768	22.69 cpb
1024	26.06 cpb

Tabla II
Rendimiento en ciclos por byte

IV. Función de ronda

IV-A. Función de ronda original

Cada ronda utiliza los siguientes elementos:

- Dos palabras expandidas $W_{(2r)}$ y $W_{(2r+1)}$.
- Un valor de 8 bits s, dependiente del mensaje, y sus nibbles superior (q) e inferior (p).
- Una constante K_s.
- Las variables temporales del hash h_j, con $j = 0, 1, \ldots, 31$.

En el inicio de cada ronda, se obtiene s al comprimir ambas palabras expandidas ($W_{(2r)}$ y $W_{(2r+1)}$) y emplearlas como entrada para la caja de sustitución. Posteriormente, se obtienen los nibbles p y q a partir de s.

A continuación, las palabras temporales A y B se obtienen mediante la aplicación de las funciones no lineales descritas en

la sección II-A. La función empleada depende de la ronda: F_1 se usa cuando el número de ronda es par y F_2 cuando es impar. Además, los valores utilizados dependen de s (o p, q) por lo que la transformación, efectivamente, depende también del mensaje. Finalmente, se modifican dos variables temporales del hash en función de las palabras A y B recién calculadas. Estas palabras dependen del número de ronda ($h_{(r \mod 32)}$ and $h_{(r+16 \mod 32)}$) de forma que todas las palabras son modificadas según progresa el algoritmo.

La función de ronda se define de la forma siguiente, con $r = 0, 1, \ldots, R-1$:

$$\begin{aligned} C &= W_{(2r)} + W_{(2r+1)} \\ s &= s \oplus S_{box}(C \oplus (C >> 8) \\ &\quad \oplus (C >> 16) \oplus (C >> 24)) \\ p &= s \mod 16 \\ q &= (s >> 4) \mod 16 \end{aligned}$$

$$\begin{aligned} A &= F_{(r \mod 2)+1}(h_p, h_{(r \mod 32)}, \\ &\quad FR_1(h_{q+16})) + W_{(2r)} + K_s \\ B &= F_{(r \mod 2)+1}(h_q, h_{(r+8 \mod 32)}, \\ &\quad FR_2(h_{p+16})) + W_{(2r+1)} \\ h_{(r \mod 32)} &= h_{(r \mod 32)} + B \\ h_{(r+16 \mod 32)} &= h_{(r+16 \mod 32)} \oplus (A + B) \end{aligned}$$

IV-B. Posibles mejoras en la función de ronda

En el ataque de Thomsen (véase [9]), se hace uso de la siguiente expresión en la función de ronda para provocar la colisión:

$$h_{(r+16 \mod 32)} = h_{(r+16 \mod 32)} \oplus (A + B).$$

El problema consiste en que $A + B$ es conmutativo, por lo que es una vía sencilla de conseguir que una diferencia entre dos mensajes quede anulada. Si bien la nueva función de expansión evita que el atacante tenga el nivel de control necesario para que este ataque sea satisfactorio, resulta necesario un rediseño de esta función de forma que maximice la posibilidad de que una diferencia entre dos mensajes sobreviva a lo largo de todas las rondas.

V. Función de salida

V-A. Función de salida original

Una vez se han procesado todos los bloques del mensaje con la función hash, el hash final para el mensaje M se corresponde con la concatenación de las palabras de hash para el último bloque N. Cuando el tamaño del resumen es menor de 1024 bits, el hash correspondiente es una versión truncada del hash completo de 1024 bits, como sigue:

$$\text{Tangle}(M') = H_0^N \mid H_1^N \mid H_2^N \mid \ldots \mid H_{\frac{D}{32}-2}^N \mid H_{\frac{D}{32}-1}^N.$$

Cada palabra H_j^N se escribe en formato *Little-Endian*.

V-B. Posibles mejoras en la función de salida

Como evidenció el trabajo criptonalítico de Thomsen [9] y Esmaeili [4], la función de salida por truncamiento produce el problema de que las últimas rondas quedan inutilizadas en los tamaños de resumen más pequeños, perdiendo su efecto en el resumen final.

Para evitar este defecto, se debe rediseñar la función de salida de forma que el resumen final, independientemente del tamaño, sea una función de todas las palabras del resumen de 1024 bits producido por la función de compresión.

VI. Conclusiones

Se han presentado los avances realizados sobre el diseño original de la función hash Tangle con el objetivo de corregir los fallos descubiertos en los ataques de colisiones sufridos durante la competición SHA-3. Estas mejoras se centran en eliminar los problemas de seguridad en las áreas de las funciones de expansión, de ronda y de salida encontrados tras un criptoanálisis de la versión original de Tangle.

El esquema de expansión de mensaje mejorado se basa en optimizar la avalancha general de la función, colocando las diferencias más significativas al principio del mensaje expandido para que sean procesadas por un mayor número de rondas. Para ello, se incorporan dos palabras pseudo-CRC que actúan como mini resúmenes del mensaje. Este esquema es un rediseño completo del esquema original, presentando mejores propiedades de seguridad y manteniendo un alto nivel de rendimiento.

Si bien el esquema de expansión modificado evita los ataques conocidos sobre Tangle, también resulta conveniente una labor de rediseño de las funciones de ronda y de salida con el objetivo de evitar el criptoanálisis diferencial y mejorar la seguridad global de Tangle.

Agradecimientos

Este trabajo ha sido realizado bajo el marco de los proyectos GRE09-02 y GRE10-34 de la Universidad de Alicante y GV/2011/001 y GV/2012/111 de la Generalitat Valenciana.

Referencias

[1] Daemen, J., Rijmen, V.: The Design of Rijndael: AES–the Advanced Encryption Standard. Information Security and Cryptography XVII. Springer, 2002.

[2] Alvarez, R., McGuire, G., Zamora, A.: The Tangle Hash Function. NIST SHA-3 Competition submission, 2008.

[3] Coron J., Dodis Y., Malinaud C.: Merkle-Damgard revisited: How to construct a hash function. Advances in Cryptology (CRYPTO 2005), vol. 362, pp.430–448, 2005.

[4] Esmaeili, Y.: Some observations on Tangle. Observations on Tangle, NIST SHA-3 Competition, 2008.

[5] McAuley, A. J.: Weighted Sum Codes for Error Detection and Their Comparison with Existing Codes IEEE/IACM Transactions on Networking, vol. 2-1, pp.16–22, 1994.

[6] Odoni, R. W. K., Varadharajan, V., Sanders, P. W.: Public Key Distribution in Matrix Rings. Electronic Letters, vol. 20, pp.386–387, 1984

[7] National Institute of Standards and Technology: Secure Hash Standard (with change notice). Federal Information Processing Standards Publication FIPS-180-2, 2002.

[8] NIST SHA-3 Competition Resources. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

[9] Thomsen, S. S.: Untangled. Observations on Tangle, NIST SHA-3 Competition, 2008.