Criptografía Simétrica

M.I. GONZÁLEZ VASCO / GRADO EN INGENIERÍA DE LA CIBERSEGURIDAD

UNIVERSIDAD REY JUAN CARLOS

Qué vamos a aprender

1. Primitivas útiles: HASH/MACs
2. Cifradores históricos
3. Cifrado en Flujo
4. Cifrado en Bloque

Referencia: Capítulo 3, 5, 7 y 8 del Smart

1. Primitivas

¿Qué es una Función Hash?

- (informal); función H con dominio en {0,1}* y rango en $\{0,1\}^m$ para m un número natural (entero positivo) prefijado.
- Características:
  - H fuertemente no inyectiva
  - H "resume" o "condensa" cadenas de bits
- Propiedades computacionales deseables:
  - Eficiencia, simplicidad
  - Resistencia a colisiones

USOS

- Con frecuencia, se utilizan para almacenar datos y hacer comparaciones rápidas (passwords, credenciales..)
- Son piezas fundamentales para distintos escenarios de clave secreta o clave pública: firma, cifrado, intercambio de clave, etc. Sirven para construir *pruebas de integridad* o *pruebas de conocimiento:*
  - *INTEGRIDAD: las alteraciones a un mensaje transmitido x pueden detectarse si se envía una etiqueta H(x) a contrastar con el hash del mensaje recibido (como veremos, los MACs son diseños específicos para este fin).*
  - *PRUEBA DE CONOCIMIENTO: la publicación de un valor H(x) en un determinado momento t_0 sirve para demostrar conocimiento de x en t_0*

Propiedades: Resistencia a Colisiones (CR)

- Es una propiedad difícil de demostrar, se enuncia en términos probabilísticos
- Informalmente: no debe existir un algoritmo eficiente que resuelva el siguiente problema:
  - Entrada: descripción de H, parámetro de seguridad n
  - Salida: dos valores x y x', distintos, de $\ell(n)$ bits --- ℓ polinomial --- cumpliendo que $H(x) = H(x')$

Resistencia a colisiones (debilitada)

- *Target-collision resistance (TCR):*
  - Entrada: descripción de H, parámetro de seguridad n, x elegido uniformemente al azar en $\{0,1\}^n$
  - Salida: un valor x' (distintos de x), cumpliendo $H(x) = H(x')$

- *Preimage resistance (PR):*
  - Entrada: descripción de H, parámetro de seguridad n, y elegido uniformemente al azar en el rango de H
  - Salida: un valor x cumpliendo $H(x) = y$.

¿Qué es más difícil de conseguir?

- Ejemplo: considera una función Hash que, dada una cadena arbitraria de bits x, sigue los siguientes pasos:
  - Construye un número entero al que esos bits representan, que llamaremos z
  - Calcula $w = z^6 + z^2 - 24$
  - Devuelve la cadena y que forman los "n" bits más altos de w
- ¿Sabrías encontrar una colisión?
- ¿Sabrías, dada una cadena de bits arbitraria x, encontrar un valor x' con igual resumen?
- ¿Sabrías, dada una cadena y, encontrar un x tal que H(x)=y ?

¿Cómo se construyen?

Es una construcción en dos pasos:

1. Construcción de una **función de compresión** (que sería una función hash pero con tamaño de entrada prefijado), H_C con dominio en $\{0,1\}^n$ y rango en $\{0,1\}^m$
2. Diseño de un **mecanismo de extensión** para que la función pueda tener entradas de tamaño arbitrario.

Paso 1: EJEMPLO 1

Fijado un parámetro de seguridad n consideremos (la descripción de) un grupo cíclico $\mathcal{G}$ de orden q (con n bits) y un generador g del mismo. Supongamos que q es primo con probabilidad abrumadora. Seleccionemos h al azar en $\mathcal{G}$.

Para cada par de enteros (x1, x2) en Z_q se define su hash como

$$H(x1,x2)= g^{x1} h^{x2}$$

Ejemplo 1 (cont)

- Como x1 y x2 son elementos de Z_q y q es un número que se escribe con n bits, la definición anterior puede reescribirse para definir el dominio de H como $\{0,1\}^{2(n-1)}$
- De modo similar, podemos explicitar la salida de H como una cadena de bits. La función sólo será una función de compresión válida si los elementos del grupo $\mathcal{G}$ pueden representarse con menos de 2n-2 bits (existen numerosos ejemplos).

Funciones Hash; resumen histórico

- **MD5:** 1991, función hash con 128 bits de salida. Demostrada completamente Insegura desde 2005, puesto que incluso es posible generar colisiones controladas.
- **FUNCIONES SHA:** 1995, estándares del NIST.
  - **SHA-0:** insegura
  - **SHA-1:** salida de 160 bits, sospechosa
  - **SHA-2:** salidas de 256 o 512 bits.

  Se construyen obteniendo una función de compresión a partir de un cifrador en bloque con un método conocido como la construcción de Davies-Meyer. Después se aumentan para recibir entradas de longitud arbitraria usando la transformada de Merkle-Damgaard.

SHA -3

- También llamada Keccac, estándar del NIST desde 2012.
- Muy diferente a sus predecesoras de la familia SHA:
  - No usa la construcción de Davies-Meyer
  - Utiliza una construcción llamada *de esponja* sustituyendo a la transformada de Merkle-Damgaard.

¿Cómo se atacan?
Primer paso: Ataques de cumpleaños *(Birthday Attacks)*

- Por el llamado Principio del Palomar, dada una función hash H con rango en $\{0,2\}^n$ si evaluamos H en $2^n + 1$ entradas distintas encontraremos una colisión → ataque trivial de complejidad $\mathcal{O}(2^n)$
- Objetivo: minimizar t, de modo que si evaluamos H en t entradas elegidas uniformemente al azar, la probabilidad de encontrar una colisión sea lo mayor posible.

Ataques de cumpleaños *(Birthday Attacks) (II)*

- Obviamente, si t = $\boldsymbol{\theta}(2^{n/2})$, la probabilidad de encontrar un colisión es aproximadamente ½. Así, si queremos que encontrar colisiones sea tan difícil como buscar en un conjunto de tamaño 2^{128} tenemos que buscar una función hash con rango de al menos 256 bits
- NOTA: el mismo razonamiento vale si las t entradas son sobre textos DISTINTOS, aunque no se elijan uniformemente al azar.

Ataques de cumpleaños *(Birthday Attacks) (III)*

En conclusión:

- La existencia de ataques de cumpleaños obliga a, si queremos seguridad de N bits, usar funciones hash de rango en $\{0,1\}^{2N}$
- Los BA "directos" obligan a ocupar mucha memoria, pero existen mejoras (ataques de cumpleaños de espacio-pequeño).

MACs

Son herramientas para conseguir INTEGRIDAD, i.e., sirven para detectar si un mensaje ha sido alterado (en la transmisión o mientras estaba almacenado).

MAC

- Un MAC, **message authentication code,** es una terna de algoritmos (**Gen, Mac, Vrfy**):
  1. **Gen:** generación de claves, pptm, entrada 1^n salida una clave k (con menos de n bits)
  2. **Mac:** generación de etiquetas, pptm, entrada k, m $\in \{0,1\}^*$ da como salida un etiqueta t.
  3. **Vrfy:** verificación, determinístico, entrada k, m, t, salida un bit b=1 (válido) o b=0 (inválido).

  Corrección: Para todo n, k, m, t consistentes, se tiene **Vrfy**(k,m,t)=1

MAC: seguridad

Idea: un adversario no debería ser capaz de crear una etiqueta t válida en un mensaje que no ha sido construido por un usuario legítimo (i.e., que tenga la clave secreta)

…incluso aunque se le dé acceso a un oráculo que genera etiquetas, (para la clave correcta)

MAC: seguridad

CUIDADO: Los ataques por reenvío *(replay attacks)* quedan excluídos de esta definición. Para evitarlos es necesario incluir técnicas como contadores o sellos de tiempo

Obviamos otros problemas que tienen que ver con ataques colaterales (side-channels), como los ataques de medición de tiempo o *timing attacks*.

El paradigma *Hash and MAC*

- Es el paradigma de autenticación de mensajes más utilizado en la actualidad (por ejemplo, en el estándar llamado HMAC)
- IDEA: dado un mensaje m de longitud arbitraria, se construye un hash de longitud prefijada usando una función hash H resistente a colisiones. Después, se aplica un MAC sobre el resultado.
- ¿Por qué es seguro? Intuición: si H es resistente a colisiones, autenticar H(m) *es tan bueno como* autenticar m.

Algunas pinceladas de seguridad

- Informe ECRYPT 2018 ;

Table 4.3: Hash Function Summary

Primitive	Output Length	Classification Legacy	Classification Future
SHA-2	256, 384, 512, 512/256	✓	✓
SHA-3	256, 384, 512	✓	✓
SHA-3	SHAKE128, SHAKE256	✓	✓
Whirlpool	512	✓	✓
BLAKE	256, 384, 512	✓	✓
RIPEMD-160	160	✓	✗
SHA-2	224, 512/224	✓	✗
SHA-3	224	✓	✗
MD5	128	✗	✗
RIPEMD-128	128	✗	✗
SHA-1	160	✗	✗

Table 5.2: Symmetric Key Based Authentication Summary Table. When instantiating the primitives they should be selected according to our division into legacy and future use to provide the MAC function with the same level of security.

Scheme	Classification Legacy	Classification Future	Building Block
CMAC	✓	✓	Any block cipher as a PRP
EMAC	✓	✓	Any block cipher as a PRP
AMAC	✓	✓	Any block cipher
HMAC	✓	✓	Any hash function as a PRF
UMAC	✓	✓	An internal universal hash function
GMAC	✓	✗	Finite field operations
Poly1305	✓	✗	Finite field operations