Seguridad en sistemas embebidos

Presentation · August 2018

DOI: 10.13140/RG.2.2.24581.81124

1 author:

Javier Alejandro Jorge
UNC Universidad nacional de corodoba
11 PUBLICATIONS **3** CITATIONS

SEE PROFILE

Seguridad en sistemas embebidos

Javier Jorge - Ignacio Moretti - labdei@inti.gob.ar

Cyberphisical systems

Internet of Things (IoT)
Industrial Internet
Smart Cities
Smart Grid
"Smart" Anything (e.g., Cars, Build
Hospitals, Appliances)

By Wilgengebroed on Flickr - Cropped and sign removed from Internet of things signed by the author.jpg, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=32745645

Algunos ejemplos de problemas

Suministro de energía y plantas industriales

- Gusano SUNTEX para interferir en sistemas basados en dispositivos Siemens (para inutilizar una “planta de enrriquecimiento” de uranio en iran).

Computadoras de escritorio

- Gusano Flame para sistemas Windows

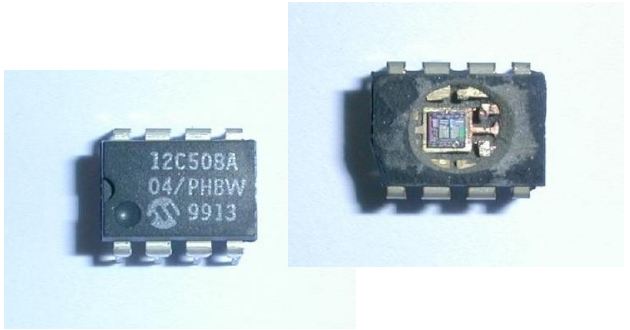
Wikileaks ?
Uso de bugs de software, imposibilidad de actualización de firmware, etc.

Como garantizar el origen de los datos ?

Costo de implementar seguridad en el diseño (tiempo al mercado)

https://www.cnbc.com/id/48147966

Ataques posibles



Clonado de dispositivos

- Ruptura de la protección contra copia

Técnicas

- Microprobing (acceso directo a la superficie del microcontrolador)
- Software attack (uso de interfaces de comunicación normal para explotar bulnerabilidades de seguridad de protocolos, algoritmos o en su implementación particular)
- Eavesdropping techniques (monitorear consumo, emisiones electromagnéticas de la alimentacion el clock o los canales de comunicación usados en el normal funcionamiento).
- Fault generation (inducir estados de error en el procesador para llevarlo a estados poco verificados y escalar en privilegios)

https://www.cl.cam.ac.uk/~sps32/

Soluciones ??

Que tan seguras son las computadoras que intervienen en la mayoría de nuestras actividades, desde la producción de caramelos, hasta las luces de los semáforos o el respirador artificial conectado a la LAN del hospital?

Seguridad basada en Secreto ?

Buenas practicas de programación ?

Uso de Open Source frameworks/OS ?

Criptografía ? Firma digital ?

Calidad en el proceso

Calidad en el proceso permite disminuir el numero de errores de software

Los modelos iterativos posibilitan llegar antes al mercado

Los proyectos mantenibles tienen mas posibilidad de sostenerse en el tiempo

La posibilidad de actualizaciones remotas puede ser muy útil pero ...

Considerar a los requisitos de seguridad en el proceso y tratarlos como req. funcionales.

Ref:Aportes para una Internet de las Cosas Seguras
https://www.researchgate.net/publication/320452544_Aportes_para_una_Internet_de_las_Cosas_Seguras/references

Atributos de seguridad

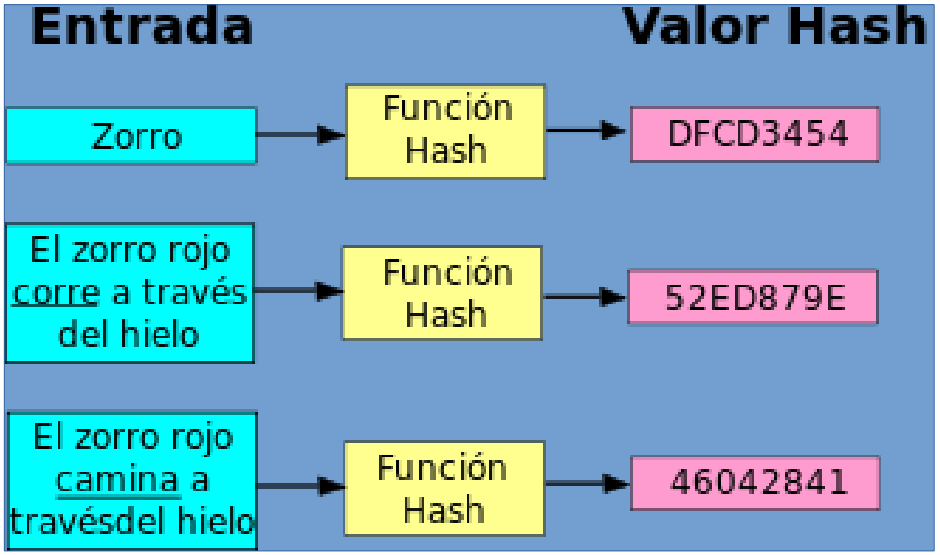
Integridad (intencional o accidental)
Confidencialidad (secreto)
Autenticidad (origen)
No repudio (imposibilidad de negar la creación de un mensaje)

Disponibilidad (DOS)

INTEGRIDAD: Algoritmos de HASH

Algoritmo, que tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte (mapea) en un rango de salida finito, normalmente cadenas de longitud fija.

Es un conjunto de procesos que se aplican en un texto o archivo, generando un código verificador único, asegurando su integridad



Fercufer - Trabajo propio-funcionamiento de función hash con terminología estándard - CC BY-SA 3.0

PROPIEDADES de los algoritmos de hash

Irreversible : deberá ser imposible obtener el texto original desde el resumen obtenido
Rápido: procesos rápidos y de escasos recursos
Salida de longitud constante: de un texto con longitud variable se conseguirá uno de longitud fija
Determinista: el texto original arrojara el mismo resultado
No continuo: modificando al menos 1 bit , el hash tendra que cambiar su salida en mas de un bit.

Algoritmos populares

MD5: es un algoritmo de reducción criptográfico diseñado por el profesor Ronald Rivest del MIT, en el año 1992, mejorando sus antiguas versiones (MD2/MD4), utilizando una salida de resumen de 128 bits

SHA -1: creado por la NSA en el año 1995, utilizando una salida de resumen de 160 bits

SHA-2: es un conjunto de funciones criptográficas de hash (SHA-224, SHA-256, SHA-384, SHA-512) diseñado por la Agencia de Seguridad Nacional (NSA) y publicado en 2001 por el NIST

SHA-3: es el último miembro de la familia de estándares Secure Hash Algorithm , publicado por NIST el 5 de agosto de 2015.

Ejemplo

Palabra “SASE” procesada con diferentes algoritmos
Sha3 512

- C336b8b96e4d5ede04ee181293e1207ab837a8a750eb98f1a951cb113193325d a1492782163cb70d01cfe642f9ec183fd8e4941558c89ad785349e0ecd2de045

SHA2 512

- 20a065c4dead78854b005dad9fd3fe2161cadab1741d885a31d75c45bff060c8c0 03f4bf9a01f186a2bd2ef68b2f8c5a20a22ba43fa78d2d0ca82d0aba1e652d

MD5

- 502689547734d313aee1f9176684caaf

CRC32

- 1e84232f

Ejemplo en arduino

```
#include <MD5.h>

// the setup function runs once when you press reset or power the board
void setup() {
    // Open serial communications and wait for port to open:
  Serial.begin(57600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  Serial.println("Goodnight moon!");
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  // delay 10 milliseconds before the next reading:
  delay(   10);
  unsigned char* hash=MD5::make_hash("hello world");
  //generate the digest (hex encoding) of our hash
  char *md5str = MD5::make_digest(hash, 16);
  //print it on our serial monitor
  Serial.println(md5str);
  //Give the Memory back to the System if you run the md5 Hash generation in a loop
  free(md5str);
  //free dynamically allocated 16 byte hash from make_hash()
  free(hash);
}
```

Ref: https://github.com/tzikis/ArduinoMD5/

Ejemplo en raspberry

```c
#include <stdio.h>
#include <openssl/md5.h>

int main()
{
    unsigned char c[MD5_DIGEST_LENGTH];
    char *filename="file.c";
    int i;
    FILE *inFile = fopen (filename, "rb");
    MD5_CTX mdContext;
    int bytes;
    unsigned char data[1024];

    if (inFile == NULL) {
        printf ("%s can't be opened.\n", filename);
        return 0;
    }

    MD5_Init (&mdContext);
    while ((bytes = fread (data, 1, 1024, inFile)) != 0)
        MD5_Update (&mdContext, data, bytes);
    MD5_Final (c,&mdContext);
    for(i = 0; i < MD5_DIGEST_LENGTH; i++) printf("%02x", c[i]);
    printf (" %s\n", filename);
    fclose (inFile);
    return 0;
}
```

Palabra “SASE” procesada con diferentes algoritmos

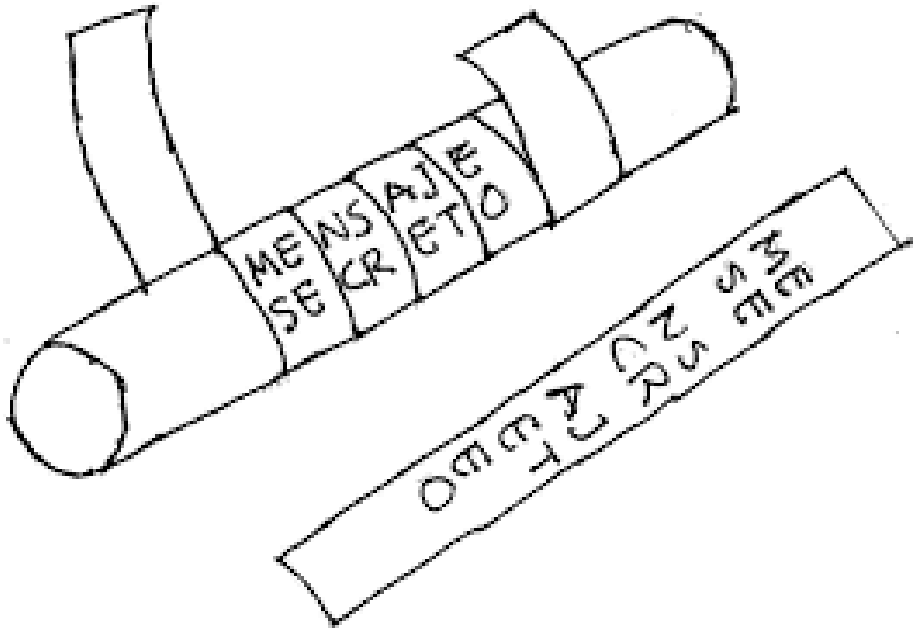
$ echo -n SASE | **md5sum**

502689547734d313aee1f9176684caaf -

Uso de openssl
gcc -o file file.c -lssl -lcrypto

Confidencialidad: Criptografía

Es el proceso que se encarga de alterar un contenido de un mensaje, evitando que para personas no autorizadas el mismo sea ilegible, mediante técnicas de ocultación, sustitución y permutación, al igual que el uso de algoritmos matemáticos, ofreciendo confidencialidad e integridad



La enigma

Alessandro Nassiri - Museo della Scienza e della Tecnologia "Leonardo da Vinci" CC BY-SA 4.0

Contaba con una serie de rotores electromecánicos. Cada uno era un disco circular plano con 26 contactos eléctricos en cada cara, uno por cada letra del alfabeto.

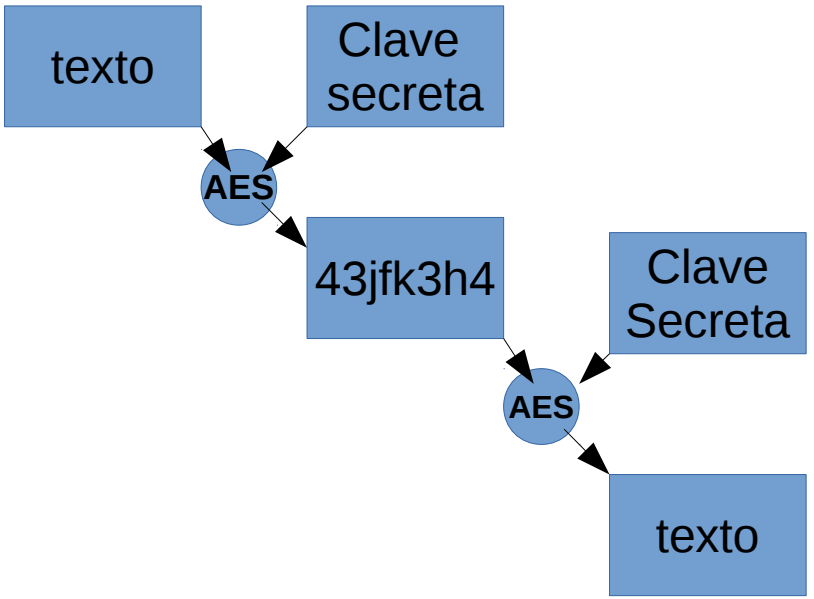
Cada contacto de una cara está conectado o cableado a un contacto diferente de la cara contraria. Por ejemplo, en un rotor en particular, el contacto número 1 de una cara puede estar conectado con el contacto número 14 en la otra cara y el contacto número 5 de una cara con el número 22 de la otra

Encripción simétrica

Existen dos tipos de encipción:

- Simétrica
- Asimétrica

La encripción simétrica utiliza la misma clave secreta para encriptar y desencriptar



Encripción simétrica

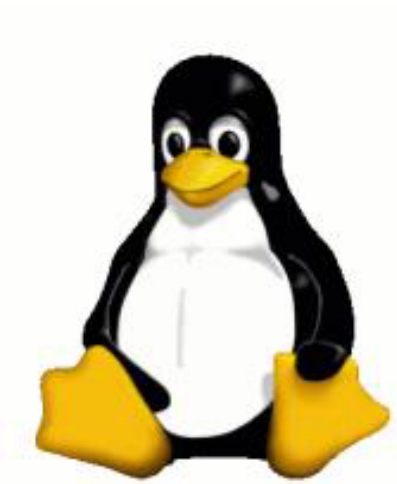
Sólo el emisor y el receptor deben conocer la clave.
Se basan en operaciones matemáticas sencillas, por ello son fácilmente implementados en hardware.
Debido a su simplicidad matemática son capaces de cifrar grandes cantidades de datos en poco tiempo.
Desventajas:
El intercambio/distribución de claves no esta estipulado.
Al ser una clave compartida, no es posible identificar al usuario del otro extremo

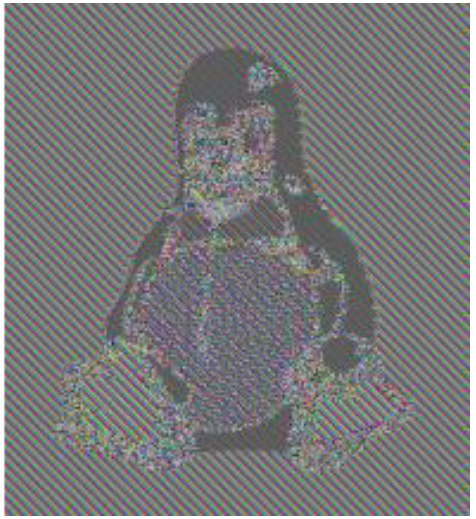
Algoritmos

DES (internal mechanics, Triple DES)
AES

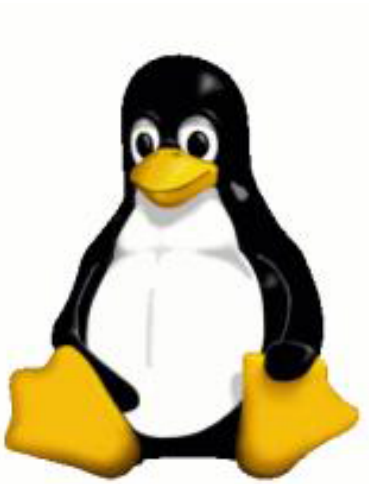
Blowfish
Serpent
Twofish

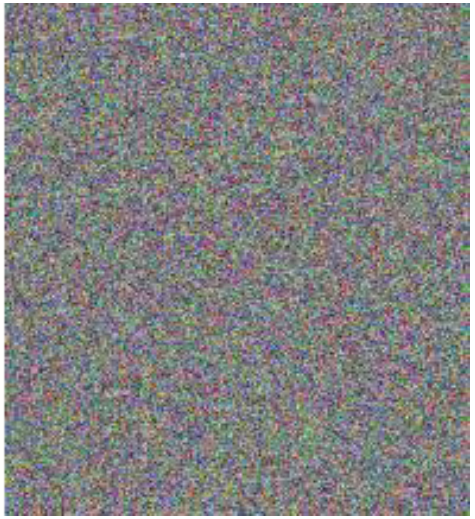
AES block modes





ECB





Cipher Block Chaining (CBC)
(CBC modes are a little slower than ECB)

AES tamaño de la clave

Tamaños utilizados generalmente: 128, 192 and 256

Basicamente con claves de 128 bits es suficiente para cualquier uso, además el uso de claves de 128 bits es mas rápido que utilizar 256.

Ejemplo Arduino

```
#include <AESLib.h>


void setup() {
  // put your setup code here, to run once:

Serial.begin(57600);
}

void loop() {

uint8_t key[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
char data[] = "0123456789012345"; //16 chars == 16 bytes
aes128_enc_single(key, data);
Serial.print("encrypted:");
Serial.println(data);
aes128_dec_single(key, data);
Serial.print("decrypted:");
Serial.println(data);

}
```

```
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
decrypted:0123456789012345
encrypted: %ʕ]ʕ?ʕʕEʕʕ ʕiʕ
```

Ejemplo raspberry

Consola

- openssl **enc** -aes128 -in foto.png -pass pass:contraseña -out foto.png-aesenc
- openssl **enc -d** -aes128 -in foto.png-aesenc -pass pass:contraseña -out foto.png
- Tipos de cifrado:
  - -aes-128-cbc -aes-128-cbc-hmac-sha1 -aes-128-cbc-hmac-sha256
  - -aes-128-ccm -aes-128-cfb -aes-128-cfb1
  - -aes-128-cfb8 -aes-128-ctr -aes-128-ecb
  - -aes-128-gcm -aes-128-ofb -aes-128-xts

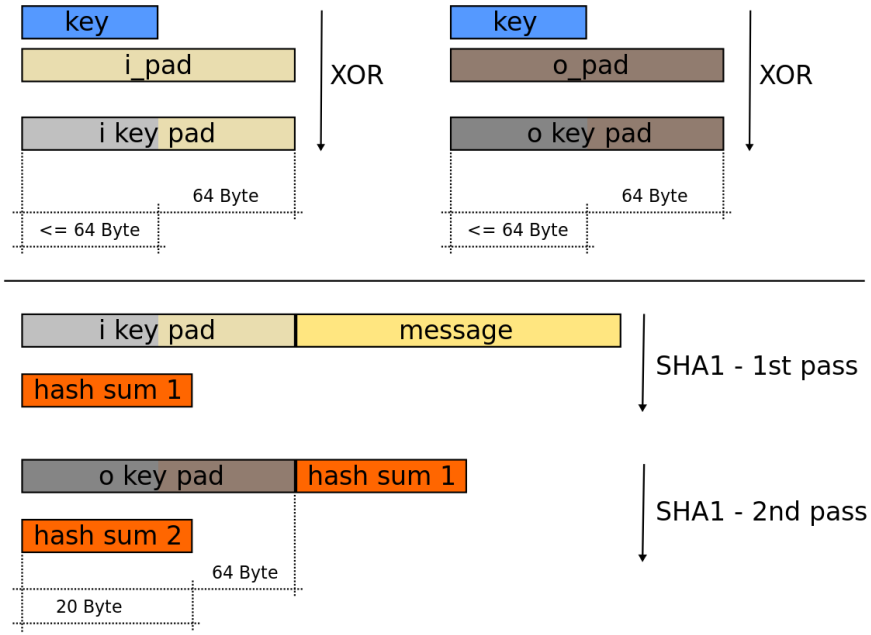
Autentificación simétrica: HMAC

Código de autentificación de mensajes en clave-hash (HMAC) Utiliza una función hash con una clave secreta “Key” (simétrica) HMAC no encripta los mensajes Puede ser utilizado para verificar simultáneamente la integridad de los datos y la autentificación de un mensaje

El mensaje (encryptado o no) se envia junto con el HMAC hash.

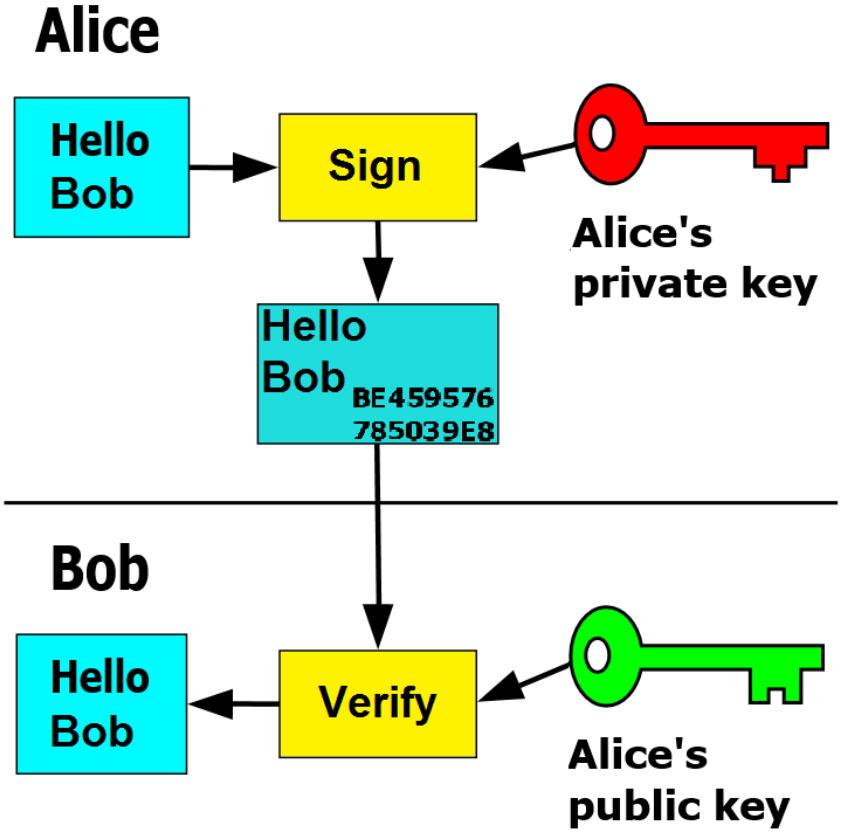
Quien tenga la clave secreta “key” y recibe el mensaje puede generar nuevamente la hmac y si es autentico coincidirá

Ejemplo de uso **Amazon AWS,** on-demand cloud computing platforms



Firma digital (Autenticación, Integridad y No repudio)

Mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente determinar la **entidad originadora** de dicho mensaje, y confirmar que el **mensaje no ha sido alterado** desde que fue firmado por el originador...

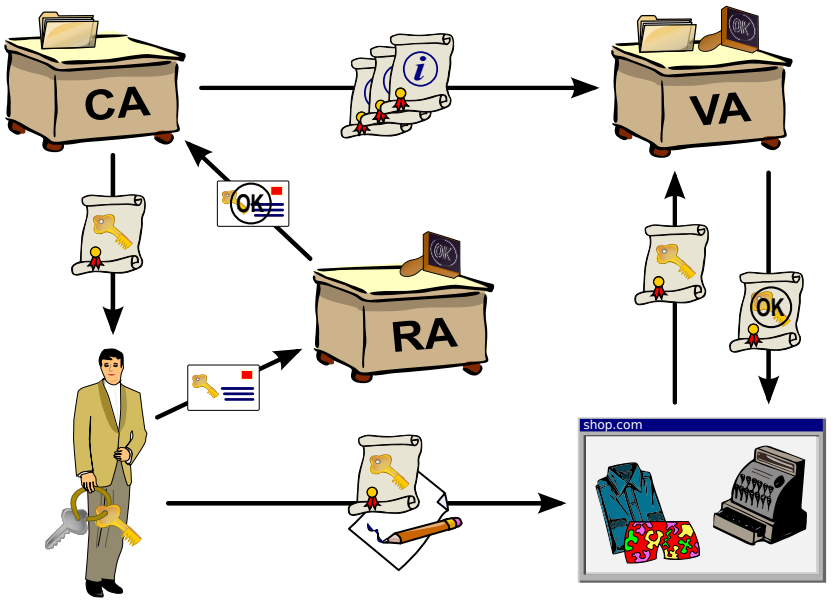


By FlippyFlink [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)], via Wikimedia Commons

PKI - AC

Infraestructura de clave pública:

CA: Autoridad de Certificación ;
VA: Autoridad de Validación ;
RA: Autoridad de Registro.



By Chris 論 ([1] and OpenCliparts.org) [GFDL (http://www.gnu.org/copyleft/fdl.html) or CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)], via Wikimedia Commons

Se usa para...

Asegurar autenticidad e integridad de datos digitales.
Detectar la falsificación y la manipulación del contenido.

Se pueden firmar cualquier documento digital

- email,
- archivo de texto, hojas de cáculo
- pdf,
- Cualquier secuencia de bytes.

Procesos intervinientes

- Proceso de formateo de los datos a firmar
- Algoritmos de generación de firma
- Algoritmo de verificación de firma

La elección de estos algoritmos constituye un esquema de firma digital adoptado.

Características de las firmas digitales

Para garantizar la seguridad de las firmas digitales es necesario que estas sean:

- Únicas
- Infalsificables
- Verificables
- Innegables

Ventajas y desventajas de pki:

- mayor seguridad (no hay intercambio de claves privadas)
- proporciona un método de firma digital
- Proporciona un método de cifrado: desventaja de utilizar el método con clave pública es la velocidad

Algoritmos de firma digital

RSA

- 1977 (bien establecido, ampliamente usado)
- Es relativamente lento (principalmente el desencriptado y la validación de la firma)

Elliptic curve

- Shorter keys are as strong as long key for RSA
- Low on CPU consumption.
- Low on memory usage.

FIPS PUB 186-4
Digital Signature Standard (DSS)

Algoritmos oficiales en argentina

La Infraestructura de Firma Digital de la República Argentina (IFDRA) ha adoptado los siguientes estándares tecnológicos:

Formato de los certificados y de las listas de certificados revocados: ITU–T X509.
Generación de las claves: RSA, DSA o ECDSA.
Protección de las claves privadas de certificadores y suscriptores: FIPS 140.
Políticas de certificación: RFC 5280 y 3739.

Ejemplo firma digital en Arduino

```
    unsigned long a = millis();
    const struct uECC_Curve_t * curve = uECC_secp160r1();
    unsigned long b = millis();
    Serial.print("uECC_secp160r1 in "); Serial.println(b-a);

    Serial.println("Testing 256 signatures\n");
    for (c = 0; c < 1; ++c) {
        for (i = 0; i < 256; ++i) {
            Serial.println(i);
            fflush(stdout);

            Serial.println("uECC_make_key() ");
            a = millis();
            if (!uECC_make_key(publice, privatee, curve)) {
                Serial.println("uECC_make_key() failed\n");
                return 1;
            }
            b = millis();
            Serial.print("uECC_make_key in "); Serial.println(b-a);

            memcpy(hash, publice, sizeof(hash));
            a = millis();
            Serial.println("uECC_sign() ");
            if (!uECC_sign(privatee, hash, sizeof(hash), sig, curve)) {
                Serial.println("uECC_sign() failed\n");
                return 1;
            }
            b = millis();
            Serial.print("uECC_sign in "); Serial.println(b-a);

            Serial.println("uECC_verify()");
            a = millis();
            if (!uECC_verify(publice, hash, sizeof(hash), sig, curve)) {
                Serial.println("uECC_verify() failed\n");
                return 1;
            }
            b = millis();
            Serial.print("uECC_verify in "); Serial.println(b-a);
        }
        Serial.println();

    }
}
```

```
uECC_verify in 1203
2
uECC_make_key()
uECC_make_key in 1112
uECC_sign()
uECC_sign in 2936
uECC_verify()
uECC_verify in 1203
3
uECC_make_key()
uECC_make_key in 5704
uECC_sign()
uECC_sign in 17222
uECC_verify()
uECC_verify in 1247
4
uECC_make_key()
uECC_make_key in 1116
uECC_sign()
uECC_sign in 27357
uECC_verify()
uECC_verify in 1241
5
uECC_make_key()
uECC_make_key in 1215
uECC_sign()
uECC_sign in 1354
uECC_verify()
uECC_verify in 1233
6
uECC_make_key()
uECC_make_key in 1223
uECC_sign()
uECC_sign in 9435
uECC_verify()
uECC_verify in 1212
7
uECC_make_key()
uECC_make_key in 1399
uECC_sign()
uECC_sign in 2390
uECC_verify()
uECC_verify in 1155
8
```

Ejemplo firma digital Raspberry

Consola:

- openssl dgst -sha1 -sign private.key -out signed.sha1 archivo.txt
- openssl dgst -sha1 -verify public.key -signature signed.sha1 archivo.txt

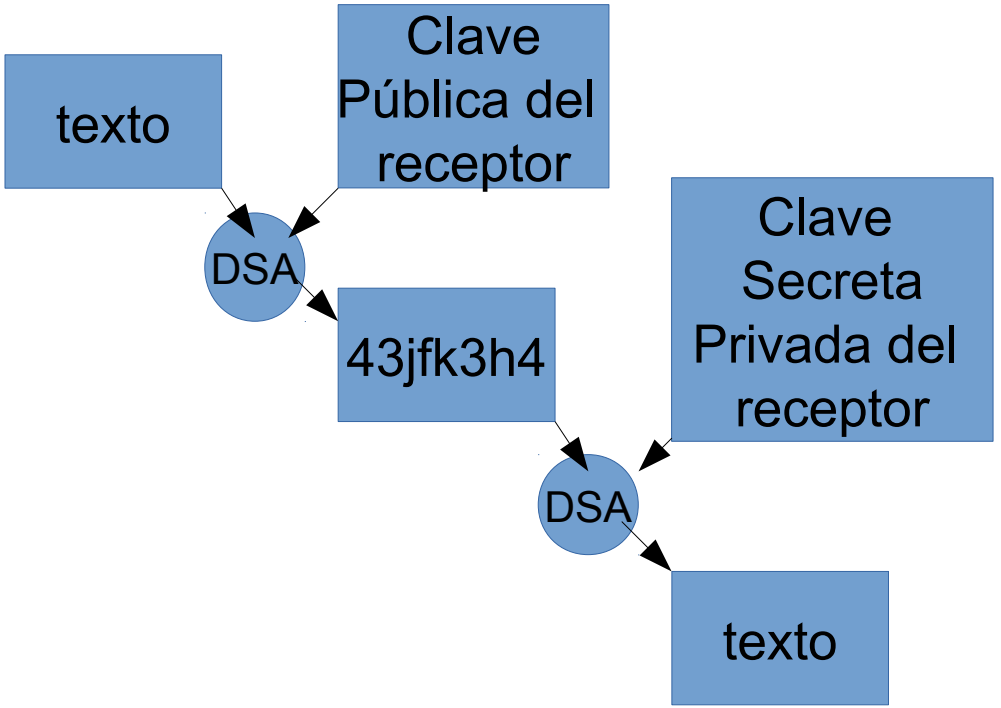
Encripción ASIMÉTRICA:

Se utiliza el esquema de firma digital,
Existen dos tipos de claves una publica y una privada, se encripta con una y se desencripta con la otra.

El emisor emplea la clave pública del receptor para cifrar el mensaje, éste último lo descifra con su clave privada

El emisor del mensaje no puede desencriptarlo.

Se ejecutan de 100 a 1000 veces más lento que los algoritmos simétricos



Encripción ASIMÉTRICA:

Debilidades:

- Es muy lenta y consume muchos recursos.
- Tiene limitaciones en el tamaño del mensaje a encriptar

Cualidades:

- Gestión y manejo de claves: seguridad en el intercambio de claves para abrir la sesión
- Incorpora concepto de firma digital y no repudio

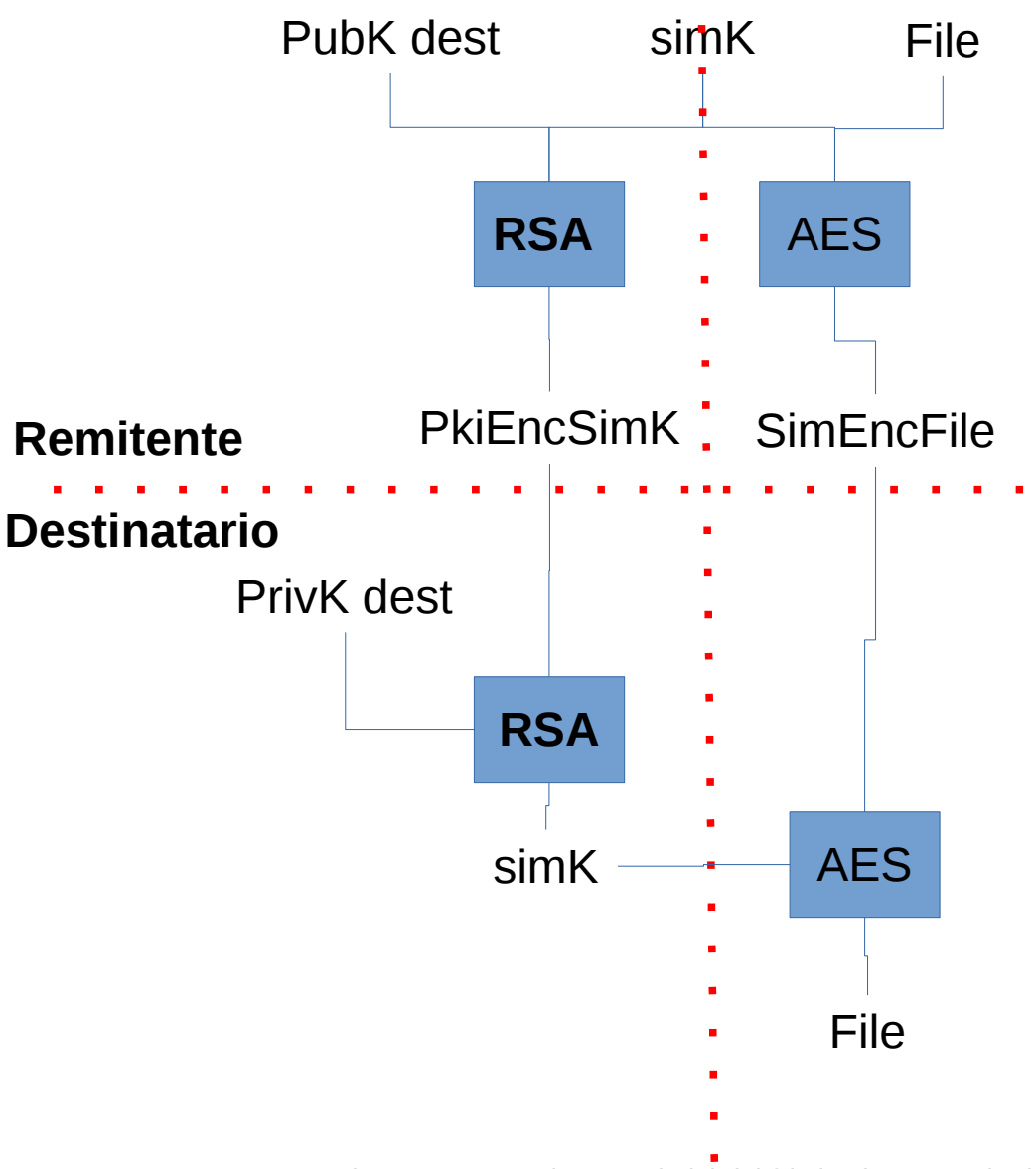
Encripción de grandes archivos
PUENTE SIMÉTRICO

Generación de una clave simétrica compleja (aleatoria)

Encripción de clave simétrica con la clave pública del destinatario

Encripción de grandes archivos con sistemas simétricos

Transmisión de clave simétrica cifrada con pki y archivo cifrado con sistemas simétricos



Ejemplo Raspberry

/**Emisor**/
2)Generar una clave simétrica aleatoria

openssl rand -base64 128 > SimetricKey.bin

3)Encryptar la llave simétrica con la clave publica del destinatario

openssl rsautl -encrypt -inkey ClavePublica.pem -pubin -in SimetricKey.bin -out SimetricKey.bin.enc

4)Encriptar el archivo grande con la clave simétrica

openssl enc -aes-128-cbc -salt -in foto.png -out foto.png.enc -pass file:./SimetricKey.bin

5) enviar SimetricKey.bin.enc y foto.png.enc

/**Receptor**/
5)Desencriptar la llave simétrica

openssl rsautl -decrypt -inkey ClavePrivada.key -in SimetricKey.bin.enc -out SimetricKey.bin

6)Desencriptar el archivo con la clave desencriptada

openssl enc -d -aes-128-cbc -in foto.png.enc -out foto.png -pass file:./SimetricKey.bin

Ejemplos

- Conexiones de red seguras http+ssl = https
  - Creación de claves de sesión simétricas
  - Intercambio de las claves simétricas usando encripción asimétrica
- Transmisión de datos firmados
  - Envío de datos de sistemas autónomos
- Firmware firmado digitalmente

Arranque de firmware firmado digitalmente

Generación del par de claves

Desarrollo del firmware

Calculo del digesto con SHA

Firma del digesto con la clave **privada** ECDSA

Frimware

Hash con su firma digital

Programación del firmware firmado

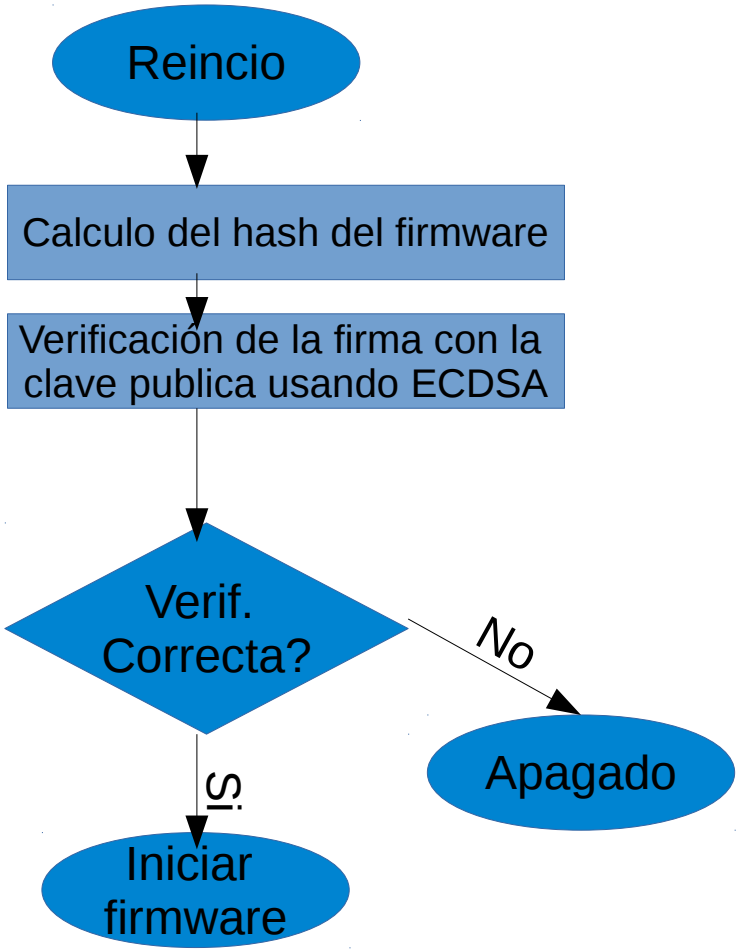
Grabado de la clave publica y el certificado

Dispositivo

Frimware

Certificado con la clave **publica**

Hash del firmware actual firmado



Ref: https://pdfserv.maximintegrated.com/en/an/AN5696.pdf

Librerías

Micro-ecc

- https://github.com/kmackay/micro-ecc

Arduino Library for AES Encryption (source based on avr-crypto-lib)

- https://github.com/DavyLandman/AESLib

WolfSSL

- https://www.wolfssl.com

BearSSL

- https://bearssl.org/

Micro-ecc

Características:

- Resistant to known side-channel attacks.
- Written in C, with optional GCC inline assembly for AVR, ARM and Thumb platforms.
- Supports 8, 32, and 64-bit architectures.
- Small code size.
- No dynamic memory allocation.
- Support for 5 standard curves: secp160r1, secp192r1, secp224r1, secp256r1, and secp256k1.
- BSD.

Ejemplo de uC con herramientas de seguridad

MAX325XX
- Secure Boot Loader with Public Key Authentication
- AES, DES and SHA Hardware Accelerators
- Modulo Arithmetic Hardware Accelerator (MAA)

Supporting RSA, DSA and ECDSA
- Secure Keypad Controller
- Hardware True Random Number Generator
- Die Shield with Dynamic Fault Detection
- 6 External Tamper Sensors with Independent Random Dynamic Patterns
- 256-Bit Flip-Flop Based Nonvolatile AES Key Storage
- Temperature and Voltage Tamper Monitor
- Real-Time External Memory Encryption and Integrity Check

Applications/Uses
- ATM Keyboards
- EMV Card Readers
- HSMs
- Industrial Modules
- PCI Mobile Payment Terminals (mPOS)
- Standalone Smartcard Readers

https://www.maximintegrated.com/en/products/microcontrollers/MAX32565.html

Atributos de seguridad - Revisión

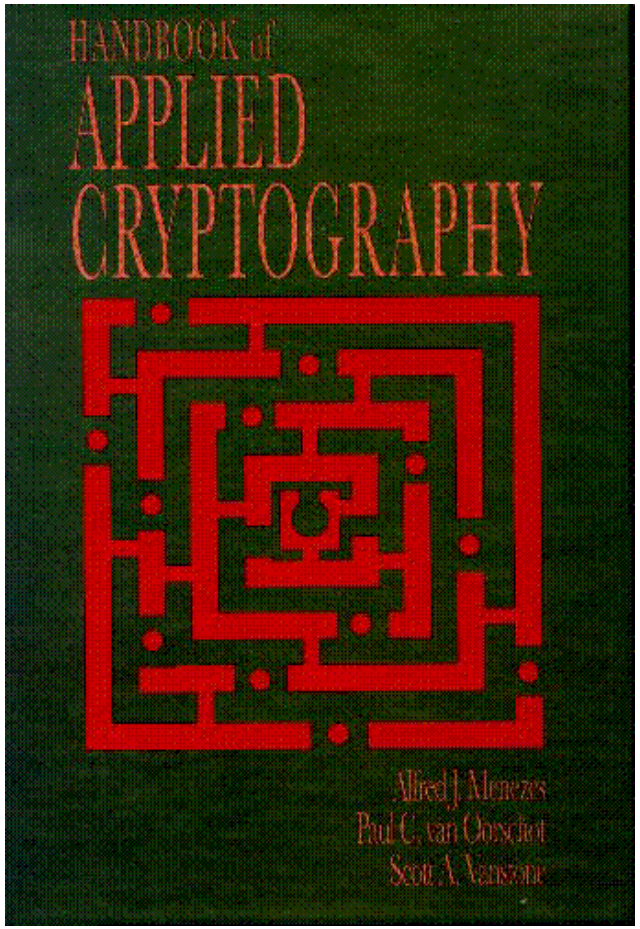
Confidencialidad → encripción (simetrica-asimetrica)

No repudio } firma digital

Autenticidad } hmac

Integridad y → hash

Donde profundizar estos conceptos



Consejos finales

Evaluar la necesidad de seguridad y el posible impacto en caso de un ataque exitoso.
Considerar a la seguridad del sistema desde el principio del proceso de desarrollo

No implementar tus algoritmos propios (basados en secreto)
No implementar versiones propias de algoritmos conocidos
Utilizar librerías abiertas (auditables, menos sensibles a bugs)

Cada dispositivo debería tener su propia clave privada
No guardar las claves en memorias no volatiles
De ser posible generar las claves durante la inicialización del dispositivo/inicio
Si no puede generarlas es necesario enviárla mediante un canal seguro

Preguntas ?

Contacto
labdei@inti.gob.ar
jjorge@inti.gob.ar