Buenos algoritmos con malas implementaciones

María Teresa Avelino Carmona
Estudiante en la Universitat Oberta de Catalunya
mavelino@uoc.edu
maite.avelino@gmail.com

***Resumen*—Este documento presenta algunos de los problemas más comunes a la hora de implementar y usar algoritmos criptográficos robustos. Se explica el porqué de dichos fallos y las consecuencias de los mismos, a la vez que se proponen después las metodologías para prevenirlos y el procedimiento a seguir para garantizar que se ha desarrollado sofware con componentes criptográficos de forma segura.**

***Index Terms*—procolos criptográficos, seguridad de aplicaciones, evaluación criptológica**

I. Introducción

Disponemos en la actualidad de excelentes algoritmos criptográficos, con gran robustez y calidad, debidamente probados y con criterios muy rigurosos para su aceptación, tal es el caso del algoritmo AES, ganador del concurso convocado por el NIST, que buscaba el mejor algoritmo de cifrado simétrico con exigentes cualidades: que fuera óptimo respecto a resistencia al criptoanálisis, que fuera computacionalmente eficiente, que economizase los recursos de la memoria del dispositivo, que pudiese implementarse tanto en software como en hardware, que dispusiese de un diseño lo más simple posible, y que no estuviese sujeto a licencia.

Sin embargo, pese a disponer de excelentes algoritmos, todos los días encontramos fallos de seguridad o debilidades en sus implementaciones posteriores. Casos con mucha repercusión mediática han sido, por ejemplo, el de la vulnerabilidad Heartbleed, que afectaba a diversas versiones de OpenSSL. Las criptomonedas han estado desde siempre en el punto de mira de los hackers, con robos sobre monederos o plataformas aprovechando las vulnerabilidades en sus implementaciones. Los sistemas operativos comerciales tampoco se libran de esta lacra, tal y como evidencia la vulnerabilidad crítica de la librería crypt32.dll de Windows 10.

Pareciera de este modo que entre los algoritmos y el software que los implementa (protocolos de comunicaciones en el caso de SSL, funciones de derivación de clave para generar direcciones en el caso de las criptomonedas o bien librerías en el caso de Windows), existieran fallos o problemas en componentes intermedios que malograsen las excelentes propiedades a priori de dichos algoritmos. Si esto es así, ¿por qué ponemos siempre el foco en lograr algoritmos cada vez más robustos y eficientes y no en conseguir implementaciones a prueba de "esos pequeños fallos"?

El objetivo de este artículo es poner en evidencia la existencia de estos fallos, apuntar a sus posibles causas y efectos y dar una pincelada sobre las posibles soluciones así como la forma de sistematizarlas. Busca de este modo, más que la profundidad técnica (inabordable por lo extenso que sería), la concienciación sobre la importancia de evitar estos fallos y los primeros pasos para sistematizar las soluciones. Analizaremos en primer lugar algunos casos conocidos de implementaciones deficientes de algoritmos para posteriormente abordar una tipología de errores comunes junto con sus causas y consecuencias. Finalmente, se sugerirán algunas soluciones para prevenir o mitigar estos problemas, así como las posibles metodologías, enfatizando en que las implementaciones de software criptográfico deben ser rigurosas, contemplando tanto el riesgo que supondrían estos fallos como el uso que se va a hacer de dicho software. Por último haremos una breve alusión a la evaluación criptológica.

II. Primer caso: Monederos electrónicos

Un caso muy conocido es el problema ocurrido con las aplicaciones de monedero usadas con las criptomonedas. En particular el caso de Bitcoin. Básicamente, para gestionar las identidades de los usuarios de forma anonimizada se usa un sistema de PKI (Public Key Infraestructure). Para verificar quién está autorizado a realizar transacciones se utiliza el algoritmo de firma electrónica ECDSA (Elliptic Curve Digital Signature), más concretamente el parámetro "secp256k1" con una robustez criptográfica equivalente a un RSA 3072, lo cual es a todas luces razonablemente seguro [1]. Así pues, para poder verificar la identidad de los intervinientes en las transacciones electrónicas usando la criptomoneda Bitcoin, se usan claves generadas mediante ECDSA como acabamos de enunciar. La generación de claves para cada usuario se realiza mediante un software denominado Wallet que normalmente se instala en el PC del usuario (también hay versiones en la Nube). Las claves se deben generar de forma aleatoria para evitar que se puedan adivinar (en la Figura 1 se puede ver un ejemplo de generación con el software de bitaddress). Aquí se presenta ya de partida una posible amenaza, ya que el generador de claves utiliza una semilla (seed) que puede ser mal generada, tal y como ocurrió en el año 2013 con los monederos instalados a través de una app para Android, si utilizaban OpenSSL PRNG con versiones de Android anteriores a la 4.4 (KitKat). Además, en la práctica, para enviar o recibir el dinero electrónico, se utiliza un localizador denominado "address"(algo similar al sistema de IBAN bancario). Estas "addresses" son derivaciones a partir de la clave original ECDSA y un parámetro como veremos a continuación.

La derivación de claves precisa de la versión del protocolo de Bitcoin utilizado (que notaremos por "v") y la clave pública de ECDSA generada de forma presuntamente aleatoria (con un generador aleatorio RNG) que notaremos por "pk". La secuencia de generación de la dirección (address) es la siguiente, que se puede consultar en la bibliografía [2]

$$hashkey = v \| RIPEMD160(SHA256(pk)) \quad (1)$$

$$checksum = SHA256(SHA256(hashkey)) \quad (2)$$

Figura 1. Generación de claves para monederos usando bitaddress.

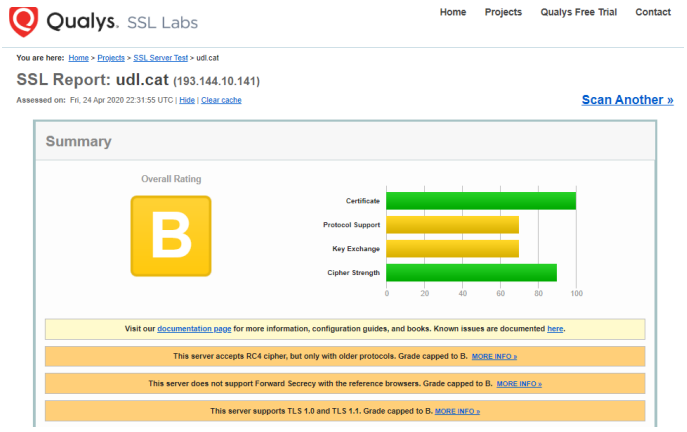

Figura 2. Análisis del estado del SSL del servidor de UDL.

Tomando sólo los 4 primeros bytes del resultado de la ecuación 2 , que llamaremos checkshum4bytes,

$$address = Base58Enc(hashkey||checksum4bytes) \quad (3)$$

De este modo, partiendo de una clave robusta (si está bien generada) como es la "pk", que es una clave ECDSA, hemos derivado un valor para la "address" usando dos trasformaciones que podríamos considerar "débiles" criptográficamente como son la función hash RIPEMD160 (viene a ser equivalente a SHA-1) y, sobre todo, la codificación posterior con Base58 (presenta problemas si se usa como entrada en la generación de tokens seguros). En general, los codificadores de alfabetos pueden presentar problemas en función del alfabeto utilizado y del uso posterior. Así pues, hemos "debilitado" la robustez inicial proporcionada por el algoritmo elegido mediante una derivación de la clave usando métodos menos robustos y como resultado es que obtenemos direcciones (localizadores de transacciones) que podrían llegar a colisionar. Hay que decir, no obstante, que el problema real de Bitcoin no es la probabilidad de colisión sino su posible predicción o incluso deanonimización , pues aun debilitada la clave tras derivarla, es tan ínfima la probabilidad de colisión, que permite a día de hoy todavía un uso razonable (siempre que, como se ha apuntado anteriormente, el RNG de la generación de direcciones no sea además vulnerable). [3]

III. Segundo caso : SSL, Heartbleed y Poodle

En el año 2014, una vulnerabilidad comprometió a medio millón de sitios web en el mundo: Heartbleed [4]. La vulnerabilidad afectaba a OpenSSL, usado para implementar TLS, de modo que se podían comprometer las claves privadas de los certificados de los servidores. TLS, evolución de SSL, permite el cifrado entre servidores y clientes por sesión, negociando las claves mediante el uso de algoritmos para la determinación de protocolos que emplean criptografía de clave asimétrica para intercambio de información inicial, y algoritmos de clave simétrica para establecer posteriormente la comunicación. Tanto los algoritmos a negociar como las claves de sesión han sido siempre objeto de ataques, de modo que algunos algoritmos y protocolos han sido descartados en cuanto se descubrió la vulnerabilidad, como por ejemplo, RC4. Sin embargo, en este caso, la vulnerabilidad no estaba ni en los algoritmos criptográficos ni en la generación de claves ni en su uso, sino en una librería de implementación del protocolo, y el ataque, como muchos ataques al software, consiste en acceder en modo privilegiado a la memoria (habitualmente por desbordamiento de tipo buffer overflown). De este modo, un error ajeno al algoritmo en sí (pues estriba en el cómo se engarza luego el algoritmo en un protocolo) se convierte en un agujero de seguridad. Usando herramientas de análisis de certificados, se puede comprobar que, a menudo, muchos sitios web presentan vulnerabilidades de algún tipo. A continuación, en la Figura 2 se muestra el resultado del análisis del dominio correspondiente a udl.cat usando la herramienta de Qalysis [5] Las categorías que arroja este testeo van graduadas de mejor a peor: A+, A, B, T, F (estas dos últimas categorías indican que los certificados correspondientes no deberían instalarse en ningún servidor por el riesgo alto de compromiso que suponen). Heartbleed fue un error no advertido y descubierto dos años después de validado el RFC donde se incluyó la librería defectuosa, pero el caso de Poodle es aún más nefasto si cabe porque su origen fue un claro ejemplo de cuando se decide sacrificar características de seguridad en aras de una mejor interoperabilidad, funcionalidad o experiencia de usuario. Aun sabiendo que SSLv.3 es susceptible de ataques

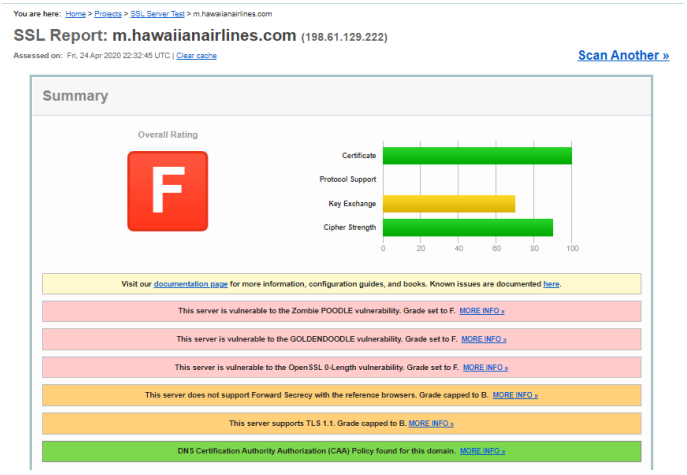

Figura 3. Análisis del estado del SSL del servidor de Hawaian airlines

que pueden exponer las conexiones entre cliente y servidor, se permitió que el cliente pudiese hacer un downgrade a esta versión aun cuando el servidor web inicialmente aconsejase

al usuario cliente el uso de TLS. En vez de forzar por parte del servidor a no permitir el downgrade del lado cliente (utilizando el mecanismo "TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks"), se siguió permitiendo para no excluir potenciales clientes, hasta que la vulnerabilidad fue finalmente explotada (CVE-2014-3566). Aunque residual, todavía está presente en algunos sitios web, como por ejemplo el que se muestra en la Figura 3.

IV. Comunicaciones móviles: GSM y el algoritmo A5/1

En el año 2009, un joven informático de la Universidad de Virginia, Karsten Nohl, anunció en el congreso de hackers "Chaos Communication Congress" que había logrado romper el algoritmo de cifrado A5/A1 para teléfonos móviles GSM. Para ello, Nohl utilizaba un conocido sistema de ataque llamado "ataque por rainbow tables". Este ataque fue más sencillo de realizar de lo habitual porque ya existía un segundo algoritmo similar al anterior, el A5/2, usado para ser exportado fuera de Europa y USA, y menos complejo que el anterior, si bien usando los mismos fundamentos criptográficos. Un análisis por ingeniería inversa del A5/2 dio la pista para romper luego el A5/1. A5/1 es un algoritmo para cifrado de flujo. La trama a cifrar consta de 228 bits, 114 de ida de A a B, y los otros 114 para la respuesta (que comparten de este modo la misma secuencia cifrante). A5/1, para generar la secuencia cifrante, dispone de 3 generadores LFSR con longitudes de registro de 19, 22 y 23 bits. De este modo por XOR conseguimos los 64 bits de clave. A5/1 es inicializado usando una clave de 64 bits junto con un número de secuencia conocido (y público, además) de 22 bits. Una de las debilidades del A5/1 es que 10 de los 64 bits de la clave son siempre cero, con lo que la clave efectiva resultante es de 54 bits (10 menos que lo deseable). Este problema (el debilitamiento de la clave) también se encontró en su día en 3DES) [6]. Sin entrar en más detalles de cómo se abordó la rotura de este algoritmo [7], queremos incidir en algo que todavía sigue ocurriendo hoy en día, y es la combinación de errores tales como debilitación de la eficacia real de un algoritmo eligiendo claves débiles de forma prefijada (para mejorar el rendimiento por ejemplo), generando versiones similares, pero con un algoritmo simplificado y la posibilidad de criptoanalizar textos conocidos. Estas tres vulnerabilidades hacen que un algoritmo en principio perfecto (según Shannon) se convierta en una solución vulnerable. También, el hecho de que se exporten versiones similares, pero una de ellas vulnerable, a terceros países, que es una práctica habitual en virtud de FIPS (Federal Information Processing Standards). Pero este procedimiento, en vez de proteger la criptografía nacional, puede llegar a propiciar su ruptura.

V. La CriptoApi vulnerable de Windows

Concluimos el repaso de malas implementaciones con la vulnerabilidad en la librería crypt32.dll de Windows 10 (CVE-2020-0601) que, aunque pasó con poco eco, y no parece muy fácil explotarla, su impacto podría ser el más alto de todos los ejemplos expuestos. Fue descubierta por la NSA (National Security Agency), quien alertó a Microsoft seguidamente [8]. El problema de esta librería es que permitía firmar certificados

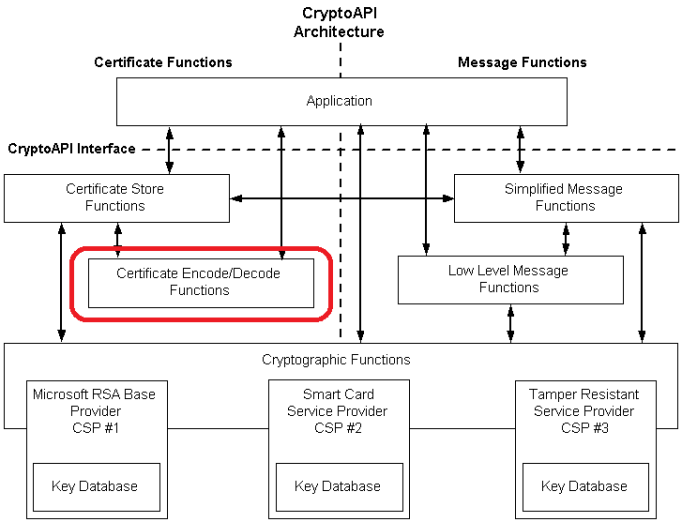

Figura 4. Componente afectado por la vulnerabilidad de la crytoapi

como si fueran de confianza cuando en realidad no lo eran, permitiendo así la descarga de malware. La NSA explica este problema por cómo se entienden los parámetros de los certificados X.509 cuando se generan usando algoritmos de curva elípticas. Este problema afecta, entre otros, al componente "Certificate Encode/Decode Functions". Los parámetros deben especificarse usando notación ASN.1. El problema es que no se especificaban correctamente, provocando que se generasen certificados sin las garantías adecuadas. No es la primera vez que ocurre este problema: En el año 2009 una vulnerabilidad similar (vulnerabilidad en ASN.1 en la CryptoAPI) hubo de ser solventada por el parche MS09-056. ASN.1 es fuente de numerosas vulnerabilidades, de hecho, hay en la actualidad existen 126 vulnerabilidades registradas en la base de datos de MITRE relativas a este "parser" utilizado por el software de Windows [9]. En el caso que nos ocupa, se puede resumir muy brevemente el problema con ASN.1 como un error de interoperabilidad semántica, ya que no fue capaz de interpretar correctamente la semántica de DER (Distinguised Encoding Rules) que se usa para describir las ECC (parámetros de curvas elípticas). La Figura 4 muestra la ubicación de este componente, enmarcado en rojo. Así pues, un problema de interoperabilidad provoca a posteriori una vulnerabilidad.

VI. Clasificación de los errores

Tras hacer este breve repaso a varios fallos conocidos de implementación/uso de algoritmos, podemos comprobar que muchos de ellos son fallos de programación de componentes intermedios (entre el cliente final y el componente de negocio que usa el algoritmo). A menudo, cuando se emplean algoritmos criptográficos se confía ciegamente en que, si el algoritmo es robusto, no es preciso siquiera revisar su pertinencia, su uso o siquiera su idoneidad. Un ejemplo sencillo es el del uso de algoritmos basados en curvas elípticas (ECC). Hay muchas familias de curvas elípticas en función de su comportamiento. Las curvas supersingulares no deben ser utilizadas para criptosistemas de cifrado, y sin embargo son idóneas para la Criptografía Basada en la Identidad [10]. Un error en el uso de la especificación de la curva incorrecta (usar

una curva supersingular para implementar un criptosistema de firma) puede provocar un agujero de seguridad posterior, aunque el algoritmo, en sí mismo, sea totalmente correcto. Hay diversas clasificaciones de las vulnerabilidades criptológicas. Una es la que propone MITRE [11]. A continuación, mostramos un resumen de esta clasificación (más extensa), pero en vez de referenciar la familia de vulnerabilidades de dicha clasificación, lo haremos atendiendo al impacto en las dimensiones de seguridad afectadas.

Errores en la autenticación de la parte cliente o servidora: Uso de contraseñas débiles, mal generadas o reutilizar contraseñas ya caducadas, que propician el éxito en ataques por fuerza bruta o bien de diccionario. Uso de algoritmos obsoletos o vulnerables como RC4 o DES, considerados inseguros, en la negociación de las claves de sesión que propiciaría un posterior ataque "man-in-the-middle". Errores en los certificados de servidor o en su gestión que propicien mala autenticación del usuario. Esto correspondería con el ejemplo visto de la CryptoAPI de Windows.

Errores en la gestión de las identidades a lo largo de las sesiones de usuario. Por ejemplo, suplantación de identidades por uso incorrecto de mecanismos de verificación usando hashes de sesión, como el ataque "Pass the Hash" sobre el protocolo Kerberos, ya que éste, en vez de trabajar con contraseñas, usa los hashes de las mismas para validar la identidad en los tickets de sesión.

Errores en la verificación de la integridad de los datos: Colisión en la generación del hash por usar algoritmos incorrectos, mala o débil derivación de claves, como el primer ejemplo que hemos visto relativo a las direcciones en Bitcoin.

Errores en la gestión de firmas con impacto en el mecanismo de No Repudio: Por ejemplo, usar la misma clave para firmar que la usada para autenticarse, porque el software de portafirmas no gestiona correctamente ambos tipos, firmas sin verificación de la Autoridad de Certificación correspondiente, etc. Uso de aplicaciones de portafirmas con vulnerabilidades que cachean los PINs de los certificados o que permiten la exportación de claves por usuarios no autenticados previamente.

Rotura de la confidencialidad de la información. Este problema suele ser el más visible, ya que a menudo se sigue de exfiltración de la información. Generalmente la rotura viene motivada por un ataque a menudo propiciado por el empleo de claves débiles. (A modo de ejemplo se estima, por ejemplo, que una clave de 8 bytes puede ser rota en minutos por un superordenador y en horas por un PC). Si además la clave está registrada en algún diccionario de uso común como puede ser "rockyou", disponible en Github (incluido en la herramienta de cracking Mimikatz), el ataque puede ser completado en segundos. Otras veces se usan sistemas de SSO(Single Sign On) y lo que se rompe es la clave maestra (ataques históricos al protocolo Kerberos, troyanos, keyloggers, etc.).

En el informe anual de OWASP (Open Web Application Security Project) con el Top 10 de las vulnerabilidades que más afectan a las aplicaciones web a nivel mundial [12], se sitúa en segundo lugar los fallos en la autenticación, y seguidamente, en tercer lugar, el fallo en la protección de los datos sensibles, como, por ejemplo, falta o uso incorrecto del cifrado de dichos datos. Más adelante en el lugar noveno encontramos el uso de componentes con vulnerabilidades conocidas. Año tras año se repiten estos tres tipos de fallos. Y estos tres fallos están conectados con el uso incorrecto de la criptografía.

VII. Metodologías de Desarrollo/Verificación

Si bien existe abundante literatura sobre metodologías y buenas prácticas para el desarrollo seguro de software y/o para su verificación posterior, hay muy pocas en lo tocante a la implementación/verificación del software que usa criptografía. A continuación, se enumeran las que presentan mejor nivel de detalle o mayor difusión. OWASP provee de una metodología muy extendida para verificar la seguridad de las aplicaciones web y, en parte, de los componentes criptográficos. La documentación con las implementaciones recomendadas se ha consolidado en el famoso repositorio de código abierto "Github" donde, además de librerías correctas o software pregenerado, se ofrecen las denominadas "Cheat Sheets" o resúmenes de procedimientos por temas. Entre otras, destacaremos la "Cryptographic Storage Cheat Sheet" [13]. En dicho documento, las recomendaciones se detallan agrupadas en cuatro dominios: Arquitectura y Diseño de alto nivel, Algoritmos, Gestión de Claves y Almacenamiento de Claves. Siguiendo sus recomendaciones encontraríamos la solución parcial para los problemas en el uso de generadores pseudoaleatorios (cómo establecer semillas para monederos de Bitcoin), ya que, por ejemplo, se establecen recomendaciones de uso de funciones y librerías y cuáles no se deben usar ya, para cada lenguaje de programación como por ejemplo las mostradas en la Tabla I para RNGs

Tabla I
Extracto de recomendaciones para RNGs.

Lenguaje	Mét.Inseguro	Mét.Recomendado
C	random(), rand()	getrandom(2)
Java	java.util.Random()	java.security.SecureRandom
PHP	rand(), mt_rand()	random_bytes(), random_int()
.NET/C#	random()	RNGCryptoServiceProvider
Python	random()	secrets()

En segundo lugar, cabe hacer mención a Common Criteria (CC) o ISO-15408, norma que aplica tanto a la certificación de seguridad de un producto (verificando que no tenga fallos y que cumpla con una serie de propiedades deseables) como a que el desarrollo de dicho producto se haga de forma segura. En CC los requisitos de seguridad se agrupan por dominios de productos en lo que se conoce como Perfiles de Protección. Estos perfiles existen para componentes y módulos criptográficos y contienen requisitos funcionarles tales como el requisito "FCS_CKM.2"para el requisito Import Cryptographic key distribution [14] que, por ejemplo, establece que las claves de inicialización que se introduzcan de forma manual deben ir cifradas (se supone que con otro algoritmo más sencillo) o bien divididas en trozos (se supone que, para minimizar el riesgo de interceptación, que es el mayor riesgo que tienen las claves simétricas). OSSTM (Open Source Security Testing Methodology Manual) es un proyecto de Isecom que ofrece una metodología parar verificar controles de seguridad del software desarrollado/aplicación a auditar. En particular, el manual de testeo OSSTMM.3 [15] ofrece controles para la autenticación, la firma, el cifrado y casi todos los usos de la criptografía. Por mencionar un control, por ejemplo, el

11.7.2 está enfocado a la confidencialidad y, por ejemplo, el apartado c) establece que hay que verificar la robustez de las contraseñas o el método para ofuscación elegido. NIST (National Institute of Standars and Technology), dependiente del U.S. Department of Commerce, ofrece también [16] una metodología para la selección apropiada de los componentes criptográficos para su uso en soluciones de software o hardware para el gobierno de los EEUU. Este documento enumera tanto los estándares a considerar (entre los que figuran los de la metodología Common Criteria) como los estándares posibles a adoptar, en particular hace mención especial a FIPS así como las medidas y usos necesarios a adoptar.

VIII. Evaluación criptológica

Como acabamos de repasar, existen, aunque escasas, suficientes metodologías y estándares para desarrollar y verificar la seguridad y funcionalidad del software que hace uso de componentes criptográficos. El siguiente paso en línea con lo anterior sería el de proceder a la evaluación formal de los componentes criptológicos. Este tipo de evaluación de un producto de software criptográfico determina si el grado de cumplimiento de los controles (según la metodología con la que se está evaluando) es suficiente para el nivel de clasificación de la información que va a manejar. Si cumple, entonces el siguiente paso sería la certificación criptológica de los componentes y la acreditación del sistema donde va a ser utilizado. Para manejar información clasificada (OTAN, nacional, UE, etc.), todos estos procesos están debidamente recogidos y documentados. El problema habitual es que, fuera de los sistemas que manejan información clasificada, la confusión abunda. Supongamos el caso de los datos de tipo médico: La legislación de protección de datos los situaría en el tipo de "datos especialmente protegidos" lo cual indica que son, de algún modo, confidenciales, aunque no pertenezcan al ámbito de los secretos nacionales o internacionales y, por tanto, carecemos de directrices para saber si podemos usar la protección de la integridad del fichero con un MD5. ¿Quién nos podría decir si dicho algoritmo es idóneo o no? Si usamos el marco de referencia del Esquema Nacional de Seguridad (ENS) para analizar los controles de seguridad que deben aplicarse a proteger los sistemas y la información en el ámbito de la Administración en España (si consideramos que los datos médicos han sido obtenidos por un organismo de salud de carácter público), entonces tendríamos como apoyo la Guía CCN-STIC-807 relativa a la Criptología de Empleo en el Esquema Nacional de Seguridad. En su página 92 se nos indica que el uso de MD5 no está permitido para información en el ámbito de este ENS, debiendo usar como mínimo SHA-256. A nivel empresarial, muchas organizaciones establecen el uso aceptable de la criptografía, pero sólo para ciertos aspectos, siendo el más común la robustez de las contraseñas usadas en los controles de acceso. Pero no existe una política global del uso de la criptografía y así, podemos encontrar contradicciones como usar un HSM para proteger información irrelevante y no proteger mínimamente con contraseña de apertura en Word un documento estratégico. A nivel doméstico, podemos pecar de infra protección o de sobreprotección, confiar nuestra protección en terceros dudosamente confiables (abusando de aplicaciones con integración SSO) o bien no entendemos el funcionamiento adecuado del software criptográfico, pudiendo incurrir en auténticos desastres (como, por ejemplo, el de perder la clave maestra tras haber cifrado un disco duro entero). ENISA (la agencia de seguridad de la información a nivel UE) empezó a abordar esta tarea hace tiempo, publicando ya en el 2012 una encuesta sobre el uso de la criptografía en los Estados Miembros y concluyendo que es muy difícil encontrar especificaciones criptográficas para la información no clasificada (la que maneja la mayoría de empresas y particulares). Además, el nivel de dichas especificaciones es heterogéneo, dado que mientras los gobiernos y organizaciones oficiales recomiendan ciertos algoritmos y parámetros concretos, otros sólo abordan cuestiones más ambiguas como la longitud de la clave o del hash a emplear. En cualquier caso, las recomendaciones no se siguen en la práctica o se implementan de manera muy pobre. Habitualmente, tampoco se cubre la gestión de claves (Key Management) Por dar una pequeña pincelada sobre la situación en España, INCIBE, mediante su línea Protege tu empresa, proporciona recomendaciones generales sobre el uso de la criptografía, que complementa con una guía de herramientas [17].

IX. Certificación criptológica

La Certificación Criptológica garantiza con alta fiabilidad que un producto que usa algoritmos criptológicos (Hardware o Software) ha pasado favorablemente por un proceso formal de Evaluación Criptológica. En España, el Centro Criptológico Nacional (CCN) es el organismo encargado de realizar esta tarea, para la información Clasificada y para aquella que se encuentra en el ámbito del Esquema Nacional de Seguridad (información en el ámbito de la Ley 39/2015 que rige actualmente los procedimientos administrativos). El problema, no obstante, ya apuntado, es la falta de referentes a nivel privado y los problemas de interoperabilidad de productos o de importación de productos de otros países que incluso han impuesto restricciones a la hora de exportar productos que usan algoritmos criptográficos, tal es el caso del uso de productos de nivel 4 según la certificación FIPS 140-2 de Estados Unidos. En este caso, bajo el paraguas del Reglamento Europeo de Cibsereguridad y con ENISA liderando su ejecución, se puede garantizar el uso certificado de productos de software criptográfico para empresas y usuarios europeos [18].

X. Conclusiones

El camino entre la aprobación de un algoritmo criptográfico y su inclusión en aplicaciones de uso común transita por varios pasos intermedios que deben conocerse y verificarse para asegurar que dicha implementación es "razonablemente" segura para el uso que se precisa. El problema actual es que, si bien hay pautas claras para su uso en información clasificada o gubernamental, no lo hay para su uso empresarial o doméstico si bien a nivel europeo y nacional se está trabajando para conseguir estándares adecuados.

Es vital que tanto los arquitectos de soluciones empresariales como los desarrolladores, así como los CIOs e incluso los CEOs conozcan las limitaciones y características de los algoritmos criptográficos y el software que va a hacer uso de los mismos antes de ponerlos en explotación a fin de evitar

incidentes de seguridad posteriormente. Todas las empresas e instituciones deberían contar con una política de criptografía.

Por otra parte, es preciso que el consumidor final del software criptográfico incluso a nivel de usuario doméstico disponga de documentación y criterios objetivos fácilmente comprensibles para seleccionar dicho software en función de la importancia de la información a proteger y de las opciones de implementación posibles. Gracias a los esfuerzos realizados tanto por ENISA a nivel europeo como por el CCN e INCIBE en España es posible contar con estas recomendaciones para usuarios finales. Hay que recalcar que en el Reglamento Europeo de Ciberseguridad del año 2019 [19] se hace mención expresa a la necesidad/obligación (según el uso) de certificar los productos de software, con especial atención a las dependencias del software de terceros (artículo 11). Si bien todavía queda mucho camino por delante, existe al menos la senda para lograr este objetivo a nivel europeo, siendo ENISA la responsable de esta encomienda.

Por último, y más en línea con el reciente RD 43/2021 que implementa la trasposición que en el 2018 se hizo de la directiva NIS en España (RDL 12/2018), de seguridad de las redes y sistemas de información, en lo tocante a la ciberseguridad, no sólo en sistemas clasificados o afectados por el Esquema Nacional de Seguridad, sino en un ámbito más general, en los sistemas de los prestadores de servicios considerados como esenciales, se deberá observar una serie de medidas de seguridad en base a un análisis de riesgos efectuado así como una declaración de medidas de seguridad a implementar, entre las que, cómo no podría ser de otra forma, las medidas que proporciona la debida implementación de algoritmos criptológicos no podrían faltar. Un referente a tomar en este sentido sería el catálogo de productos STIC (CPSTIC), que ofrece el Organismo de Certificación del Centro Criptológico Nacional.

Agradecimientos

A Criptored y en particular a Jorge Ramió, que me han inspirado para preparar este documento. Al Centro Criptógico Nacional por permitirme el honor de cursar con ellos el Çurso de Especialidades Criptológicas"que me sirvió de impulso. A mis compañeros del grupo de investigación KISON de la UOC por su apoyo. A mi hija Amanda por entender a sus cuatro años para qué sirven las contraseñas y, sobre todo, por pasar a mi lado alguna que otra tarde con paciencia viendo dibujos en la tele o coloreando mientras yo elaboraba este artículo.

Referencias

[1] Jordi Herrera: "Paper: Research and Challenges on Bitcoin Anonymity", en *Data Privacy Management (DPM'2014)At: Wroclaw, Poland*, vol LNCS 8872, 2014

[2] M. Payeras, A.P.Isern , M.Mut: "Aula virtual Crypt4you. Lección 3 del Curso de Sistemas de Pago Electrónico",2014, available online: http://www.criptored.upm.es/crypt4you/portada.html last visited on 02/15/2021

[3] Ziyu Wang, Hui Yu, Zongyang Zhang, Jiaming Piao, and Jianwei Liu: "ECDSA weak randomness in Bitcoin". Elsevier. Septiembre 2019, available online: www.elsevier.com/locate/fgcs last visited on 02/15/2021

[4] MITRE : "Portal de vulnerabilidades conocidas CVE", available on line: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0160 last visited on 02/15/2021

[5] SSLABS, ssltest (ejecutado sobre el sitio web udl.cat), available online https://www.ssllabs.com/ssltest/analyze.html?d=udl.cat last visited on 02/15/2021

[6] Alfonso Muñoz y Jorge Ramió: "Cifrado de las comunicaciones digitales de la cifra clásica al algoritmo RSA 2ª Edición" Editorial Oxword. ISBN: 978-84-09-06917-0

[7] Elad Barkan, Eli Biham, Nathan Keller: "Instant Cyphertext Only Cryptanalysis of Encrypted Communication", available online: http://cryptome.org/gsm-crack-bbk.pdf last visited on 02/15/2021

[8] National Security Agency: "Patch Critical Cryptographic Vulnerability in Microsoft Windows.Cybersecurity Advisory", available online: https://media.defense.gov/2020/Jan/14/2002234275/-1/-1/0/CSA-WINDOWS-10-CRYPT-LIB-20190114.PDF last visited on 02/15/2021

[9] MITRE corporation: "Portal de vulnerabilidades conocidas CVE, consulta con la clave de búsqueda ASN.1", available online: https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=ASN.1 last visited on 02/15/2021

[10] Juan G. Tena: "25 años de Criptografía con Curvas Elípticas", presentado en *conferencia RECSI 2014.* año 2014, available online: https://web.ua.es/es/recsi2014/documentos/papers/25-anos-de-criptografia-con-curvas-elipticas.pdf

[11] MITRE: "Debilidades de software. Common Weakness Enumeration.Cryptographic Issues", available online: https://cwe.mitre.org/data/definitions/310.html last visited on 02/15/2021

[12] OWASP: "Top Ten Web Security Risk. OWAP Project", available online: https://owasp.org/www-project-top-ten/ last visited on 02/15/2021

[13] OWASP: "Github, repositorio de software y guías.Cheat Sheet de OWASP on Cryptographic Storage", available online: https://github.com/OWASP/CheatSheetSeries/tree/master/cheatsheets last visited on 02/15/2021

[14] Common Criteria Portal : "Protection Profile for Cryptographic Modules, Security Level Enhanced", available online: https://www.commoncriteriaportal.org/files/ppfiles/pp0045b.pdf last visited on 02/15/2021

[15] Isecom: "Metolodogía OSSTM, Manual de testeo OSSTMM.3", available online: https://www.isecom.org/OSSTMM.3.pdf last visited on 02/15/2021

[16] Elaine Baker, NIST: "Guideline for Using Cryptographic Standards in the Federal Government:Cryptographic Mechanisms", available online: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175Br1.pdf last visited on 02/15/2021

[17] INCIBE: "Políticas de seguridad para la pyme: uso de técnicas criptográficas", available online: https://www.incibe.es/protege-tu-empresa/herramientas/politicas last visited on 02/17/2021

[18] ENISA: "Security of Personal Data, Cryptographic Protocols and Tools", available online: https://www.enisa.europa.eu/topics/data-protection/security-of-personal-data/cryptographic-protocols-and-tools last visited on 02/15/2021

[19] Parlamento Europeo y del Consejo: Reglamento UE 2019/881 relativo a ENISA y a la certificación de la ciberseguridad de las tecnologías de la información y la comunicación", available online: https://eur-lex.europa.eu , last visited on 02/15/2021