Criptografía y seguridad: bibliotecas y prácticas

Por Gunnar Wolf

EL PASADO MES DE JULIO TUVE NUEVAMENTE LA OPORTUNIDAD DE PARTICIPAR EN EL CONGRESO ANUAL QUE ORGANIZA SG, PRESENTANDO LA CONFERENCIA DESARROLLO DE SOFTWARE Y CRIPTOGRAFÍA [1]. ME DOY CUENTA QUE, SI BIEN SU RELEVANCIA RESULTA CLARA Y HAY INTERÉS EN LA COMUNIDAD DE DESARROLLADORES POR COMPRENDERLO, EL TEMA DE LA CRIPTOGRAFÍA ES VISTO POR MUCHOS COMO UN TEMA CASI MÁGICO O MÍSTICO: HAY UNOS POCOS EXPERTOS, PERO ESTÁ FUERA DEL ALCANCE PARA EL COMÚN DE NOSOTROS, LOS SIMPLES DESARROLLADORES QUE CARECEMOS DE UN SÓLIDO CONOCIMIENTO MATEMÁTICO DE FONDO.

Y sí, no puedo negarlo: El uso correcto de la criptografía requiere de darse una pequeña zambullida en un mar de conceptos nuevos. Sin embargo, me da gusto ver que estamos cada vez más conscientes de la importancia de proteger los datos de nuestras aplicaciones, tanto como sea posible, por criptografía fuerte. Aquí aparece la primer tensión que abordo: ¿Cómo responder a las necesidades de seguridad de información sin tener un sólido conocimiento matemático en cuestiones de criptografía?

Afortunadamente, en ese punto es donde aparece el hilo conductor con el tema de la presente edición de SG: La era de las APIs: Hay varias bibliotecas criptográficas ampliamente disponibles, implementando los diferentes algoritmos y modos de operación. Y no es necesario implementar los protocolos criptográficos, sino únicamente llamar a las funciones indicadas de la biblioteca de nuestra elección.

Y, claro, el hablar de estas principales implementaciones también da pie a abordar del impacto de los no pocos ni triviales problemas que han aquejado a este campo en los últimos meses. Y sí, aunque probablemente muchos asocien en principio al término API con las interfaces presentadas por servicios en la nube como las redes sociales, en realidad toda biblioteca que empleemos para el desarrollo (incluyendo las bibliotecas estándar, parte de la definición de nuestro lenguaje) expone una interfaz de programación de aplicaciones.

Claro está, estamos viviendo en la era de las APIs — Desde hace más de 60 años.

Criptografía: ¿Para qué?

La criptografía nos brinda distintas herramientas para proteger nuestra información — Y si partimos de que la información es el activo más valioso de todos nuestros sistemas, resulta obvio que la criptografía es uno de nuestros mayores aliados.

Si bien la criptografía es casi tan vieja como la escritura misma, apenas fue abordada como una disciplina científica matemática a partir de mediados del siglo XX. El objetivo esencial de la criptografía es brindar confidencialidad a a información valiosa. Sin embargo, hoy en día la confidencialidad representa sólo una de las propiedades de la información que la criptografía nos permite asegurar.

La segunda propiedad que protege la criptografía es la integridad: Partiendo de un texto origen arbitrariamente largo, las llamadas funciones digestoras calculan su hash, una cadena de longitud fija con la propiedad fundamental de ser muy dificil encontrar otro texto que genere la misma salida. Esta definición de integridad posiblemente deje a más de uno insatisfecho. Y sí, por sí sola no sirve para garantizar gran cosa… pero la tercer propiedad hará clara su utilidad.

Hasta mediados de 1970, todos los esquemas de criptografía dependían de una llave única: el equivalente a una contraseña que permitía tanto convertir un texto claro en una representación ininteligible como manipular a dicha representación para obtener de vuelta al texto claro. Entre el breve lapso comprendido 1976 y 1980, el mundo de la criptografía cambió por completo: nació la criptografía de llave pública, con los principales protocolos que siguen siendo utilizados al día de hoy. Esencialmente, en vez de manejar una sola llave, se descubrieron varias funciones matemáticas que permiten llaves compuestas de dos mitades: una privada, que cada individuo debe custodiar celosamente, y una pública, que puede ser ampliamente distribuida. Estas dos llaves actúan como inversas: lo que se cifra con una sólo puede descifrarse con la otra.

Este hecho, que parecería una mera curiosidad, permitió la creación de mecanismos para proteger las tercera propiedad de la información: la autenticación. Si un mensaje cifrado por mi llave privada puede ser descifrado por cualquiera que conozca mi llave pública, y existen directorios de llaves públicas, este mensaje sólo puede haber sido generado por mí.

Ya con estos antecedentes, ¿qué es la cadena que aparece en todas las facturas electrónicas y ocasionalmente incluso en los correos electrónicos?

Tomando un texto origen, se obtiene su hash, y se firma con la llave privada del emisor. ¡Listo! Una firma electrónica con un tamaño constante, estableciendo una relación no repudiable entre un documento y su emisor.

Partiendo de distintas formas de emplear estas tres propiedades pueden derivarse otros varios patrones de validación de documentos; este es sólo el más común y sencillo de muchos de los patrones de uso de criptografía.

Pero... ¿Se puede confiar en la criptografía?

Al hablar de la integridad, mencioné que debía ser muy difícil encontrar dos textos que produzcan el mismo resultado. A pesar de que el término puede no inspirar mucha confianza, sin embargo, debe leerse no como algo que intuitivamente (y sin ser especialista) me parezca difícil, sino definido desde la teoría de la complejidad computacional: computacionalmente difícil significa que es sumamente resistente a ser atacada por búsqueda exhaustiva (o lo que es lo mismo, por fuerza bruta); idealmente, la complejidad crece de forma exponencial con el espacio de búsqueda.

Por ejemplo: un algoritmo de cifrado simétrico ampliamente utilizado como el AES puede operar con llaves de 128 a 256 bits. Esto significa que, con una llave de 128 bits, el espacio de búsqueda es de 2^{128} (del orden de 10^{38}) posibilidades. Si tuviéramos poder de cómputo suficiente para intentar decodificar un billón de llaves por segundo, una búsqueda exhaustiva tomaría sólo 719 millones de veces la edad del universo. En contraste (e ilustrando el crecimiento exponencial), una llave de 64 bits tomaría sólo 213 días.

Los algoritmos que se han desarrollado, es cierto, no son perfectos. Por ejemplo, Alex Biryukov y Dmitry Khovratovich presentaron en 2009 un artículo [2] detallando un tipo de análisis permitiendo reducir dramáticamente (a sólo unas decenas de veces la edad del universo según nuestro cálculo) el espacio de búsqueda. La búsqueda de debilidades en algoritmos criptográficos es un campo vibrante de investigación matemática, con muchos talentosos académicos buscando cómo ganarle un bit más, un pasito más. De encontrarse alguna debilidad, aparecerá sin duda publicada en las más importantes revistas académicas del ramo.

Bibliotecas criptográficas

Pero, dado que nuestro campo es el desarrollo de sistemas y no la criptografía desde un punto de partida matemático, recordemos que hay una gran regularidad entre los parámetros que requieren los diferentes algoritmos. Si empleamos una biblioteca de funciones criptográficas, como la muy conocida OpenSSL, y un algoritmo ampliamente utilizado es descubierto como vulnerable, hacer el cambio para emplear otro algoritmo resultará casi trivial.

Por otro lado, me permito citar la máxima de Adi Shamir, uno de los padres de la criptografía moderna:

La criptografía normalmente es evadida, no penetrada.

Decenas de productos han implementado soluciones criptográficas (generalmente basadas en el firmado) en los últimos años, sólo para encontrar que su mecanismo fue roto al muy poco tiempo de salir al mercado [3]. Todos estos ejemplos derivan de un uso incorrecto de la criptografía: Para hacer un jailbreak, el atacante burla la llamada a la función criptográfica, aprovechando un error de programación para secuestrar el flujo antes de que se lleve a cabo la validación.

Y antes de cerrar esta columna: mencioné a la popular biblioteca OpenSSL. A más de uno de ustedes debe haberle sonado una alarma: ¿En serio está recomendando OpenSSL? ¿Después de Heartbleed?[4] Sí, sí lo recomiendo. Vamos paso por paso.

OpenSSL es la biblioteca criptográfica por excelencia, y su principal fuerza es también su principal desgracia: es una biblioteca con 20 años de historia, y profundamente multiplataforma. Vamos, su herencia multiplataforma ha mantenido el soporte para arquitecturas que desde hace años son consideradas obsoletas, como VMS u OS/2. Además, el estilo del código es de muy difícil comprensión —los desarrolladores justifican esto puesto que, siendo la criptografía tan demandante de recursos de cómputo, se vieron orillados a tomar decisiones motivados más por el rendimiento que por la claridad.

Como sea, el fallo conocido como Heartbleed expuso lo inadecuadas que dichas prácticas son hoy en día. Sin embargo, y analizando el fenómeno desde la perspectiva de un promotor del software libre, la respuesta de la comunidad ha sido de lo más interesante: por un lado, la Linux Foundation (a través de su Core Infrastructure Initiative) está fondeando una auditoría del código de OpenSSL. Y por otro lado, surgieron dos proyectos derivados de OpenSSL, buscando corregir sus defectos de formas más radicales, aunque signifique romper la compatibilidad a nivel API: LibReSSL [5], del equipo de sistema operativo OpenBSD, y BoringSSL [6], de Google. ¿Por qué BoringSSL? En palabras de Shawn Wilden:

Los componentes fundamentales de seguridad deberían ser muy aburridos. No son el lugar para innovar y experimentar, (ni) para el código que demuestra el virtuosismo del autor. Son donde quieres implementaciones simples, directas y aburridas de los algoritmos y protocolos. Cuando se habla de seguridad, lo aburrido es bueno.

Que Heartbleed fue un fallo nefasto nadie lo pone en duda. Sin embargo, ante estos fallos siempre es interesante ver la reacción de las comunidades de desarrolladores. Al igual que cuando otros fallos severos, Heartbleed ha llevado a los desarrolladores (al menos dentro de las comunidades de software libre, cuyo código y cuyas discusiones son claramente visibles desde fuera) a iniciar auditorías y replantearse prácticas que seguramente redundarán en el avance global de la calidad del software.

Referencias

[1] *http://gwolf.org/desarrollo_y_criptografia*

[2] *http://eprint.iacr.org/2009/317*

[3] *La excelente presentación Crypto won't save you either, disponible en http://regmedia.co.uk/2014/05/16/0955_peter_gutmann.pdf, presenta más de 20 casos.*

[4] *¿No supiste acerca de la vulnerabilidad de seguridad más publicitada por lo menos en este lustro? http://heartbleed.com*

[5] *http://www.libressl.org*

[6] *https://www.imperialviolet.org/2014/06/20/boringssl.html*

.BIO

Gunnar Wolf es administrador de sistemas para el Instituto de Investigaciones Económicas de la UNAM y desarrollador del proyecto Debian GNU/Linux.
http://gwolf.org