

Aplicaciones de la Criptografía Seminario

Verificación de contraseñas

Autenticación

- La autenticación, en una aplicación cualquiera, es lo que determina quién es quién en la misma
- Y con ello los datos a los que puede acceder y sus privilegios
- No parece, en absoluto, algo a tratar a la ligera y sin embargo sí se trata a la ligera
- Los robos de identidad están a la orden del día
- Y muchos se podrían evitar usando el sentido común
- Vamos a ver cómo hacerlo correctamente
- Y cómo revisar (como pentesters) que nuestros clientes lo hacen también

Fortaleza del mecanismo de passwords

- El mecanismo de autenticación más usado en la actualidad es el uso de una password
- Rápido, sencillo, barato. Pero ¿es seguro?
- Single Point Of Failure: Una clave es lo que separa a cualquiera de tener acceso a tu información privada
- Y aún así, con la importancia que tiene, mucha gente descuida este aspecto enormemente
- Las razones son todas muy coherentes:
 - “Es que es muy difícil recordar una clave larga”
 - “Es que nunca me acuerdo de cuál es y la apunto en papeles”
 - “Es que no voy a recordar muchas claves, así que uso una o dos sólo”
 - ...

Fortaleza del mecanismo de passwords

- En 2013 se sustrajeron 150 millones de nombres y passwords de Adobe
- El análisis de las mismas encontró la siguiente distribución de contraseñas:

1. 123456: 1.911.938 usuarios	9. photoshop: 83.411 usuarios	17. azerty: 48.850 usuarios
2. 123456789: 446.162 usuarios	10. 123123: 82.694 usuarios	18. iloveyou: 47.142 usuarios
3. password: 345.834 usuarios	11. 1234567890: 76.910 usuarios	19. aaaaaa: 44.281 usuarios
4. adobe123: 211.659 usuarios	12. 000000: 76.186 usuarios	20. 654321: 43.670 usuarios
5. 12345678: 201.580 usuarios	13. abc123: 70.791 usuarios	
6. qwerty: 130.832 usuarios	14. 1234: 61.453 usuarios	
7. 1234567: 124.253 usuarios	15. adobe1: 56.744 usuarios	
8. 111111: 113.884 usuarios	16. macromedia: 54.651 usuarios	

Fortaleza del mecanismo de passwords

Por otro lado, la lista de peores contraseñas de 2014 según SplashData es (<http://splashdata.com/press/worst-passwords-of-2014.htm>)

- | | | |
|--------------|--------------|--------------|
| 1. 123456 | 10. football | 19. master |
| 2. password | 11. 1234567 | 20. michael |
| 3. 12345 | 12. monkey | 21. superman |
| 4. 12345678 | 13. letmein | 22. 696969 |
| 5. qwerty | 14. abc123 | 23. 123123 |
| 6. 123456789 | 15. 111111 | 24. batman |
| 7. 1234 | 16. mustang | 25. Trustno1 |
| 8. baseball | 17. access | |
| 9. dragon | 18. shadow | |

Fortaleza del mecanismo de passwords

http://www.abc.es/tecnologia/moviles/aplicaciones/abci-ataque-masivo-usuarios-espanoles-netflix-para-robar-cuentas-201801111808_noticia.html

Ataque masivo a usuarios españoles de Netflix para robar sus cuentas

- La compañía de seguridad Panda Security ha alertado sobre una acción de «phishing» en España y Estados Unidos por la que los ciberdelincuentes se hacen pasar por la plataforma de vídeos en un email para obtener las claves del servicio

«El verdadero riesgo es que los delincuentes **revenden todas estas cuentas en el mercado negro**. Además, pueden llegar a llevar a cabo ataques a mayor escala, ya que la mayoría de usuarios reutiliza sus contraseñas y, seguramente a través de los datos robados, otros piratas informáticos **puedan llegar a sus cuentas de correo y redes sociales**. Detrás de estos ataques hay bandas organizadas de ciberdelincuentes que van a por dinero»

Verificación del mecanismo de autenticación

Como pentesters debemos recordar que:

- Cada usuario de una empresa puede ser un potencial objetivo de este tipo de test
- Porque cada usuario de una empresa es responsable de su clave y SIEMPRE habrá alguno que no tenga el cuidado necesario por algún motivo
- Por tanto, el objetivo del test es determinar la resistencia del sistema/aplicación protegido mediante password, examinando las claves de sus usuarios, contra mecanismos de averiguación de claves:
 - Uso de diccionarios con palabras comunes (fuerza bruta)
 - Uso de datos extraídos de redes sociales
 - Búsqueda de patrones de claves

Verificación del mecanismo de autenticación

- Es decir, ¿el sistema de autenticación de la aplicación se ha diseñado para “cuidar” las claves que se le permiten introducir?
- Porque a (todos) los usuarios no los podemos educar

Verificación del mecanismo de autenticación

- Todo lo que hagamos debe estar dirigido a recabar información de la política de passwords. Por ejemplo:
- ¿Qué longitud mínima / máxima se permite?
- Si no hay límites, puede ser un gran problema
- 8+ es lo más aconsejable (más = mejor)
- ¿Se permite dejar las claves vacías?
- ¿Se obliga a que tenga un mínimo de complejidad?
- ¿Qué es una clave compleja (o fuerte)?
 - Fijemos cuatro categorías de caracteres: Mayúsculas, minúsculas, n°s, símbolos (@, #,)
 - Elige 3 (mejor las 4)
 - Compón una cadena mezclando caracteres de las categorías elegidas, de un tamaño suficiente: El7DS3pti3mbr3, Si1K@niTeB@sil@hLoAsimil@h, ..

Verificación del mecanismo de autenticación

- ¿Se obliga a cambiar la password cada X tiempo?
- Si no se obliga a ello, asumámoslo, casi nadie lo hará
- Y la gente no reutiliza claves nunca 😊
- Y que por tanto si se compromete un servicio se comprometen todos (al tener la misma clave y probablemente nombre de usuario)
- Si se obliga, ¿se permite reutilizar claves antiguas?
- Si es así, ¿cuántas?
- ¿Se generan automáticamente las claves de la aplicación?
- Si es así, ¿Siguen un patrón predecible?
- ¿Se bloquea el acceso si introduces claves erróneas?

Verificación del mecanismo de autenticación

- Hay herramientas de averiguación de claves por fuerza bruta, que pueden usarse para comprobar la fortaleza de las claves existentes en el sistema
- Estas herramientas necesitan el fichero de claves del SO
- Se pueden usar para hacer auditoría de fortaleza de claves de usuarios de una empresa
- El usuario que “caiga” no tiene una buena clave
- Se puede establecer un % de claves malas para nuestro informe, usuarios con problemas, usuarios que no siguen políticas de claves de empresa, etc

Password cracking

- La fuerza bruta es el término por el que se conoce la forma de averiguar contraseñas a base de prueba y error (masivas cantidades de prueba y error.)
- Ya que no se conocen las contraseñas, se prueban un nº abismalmente grande de ellas a ver si se da con la buena
- No obstante, raras veces este procedimiento consiste realmente en aplicar sólo “fuerza bruta”
- Las herramientas que lo hacen tienen cierta “inteligencia”
- ¿Cómo puede tener éxito una técnica así?
- Por que, como ya hemos visto, una gran cantidad de usuarios no demuestra mucha inteligencia a la hora de elegir sus claves

<http://www.tecnopasion.com/noticias/las-20-contrasenas-mas-estupidas-que-puedes-elegir-2188/>

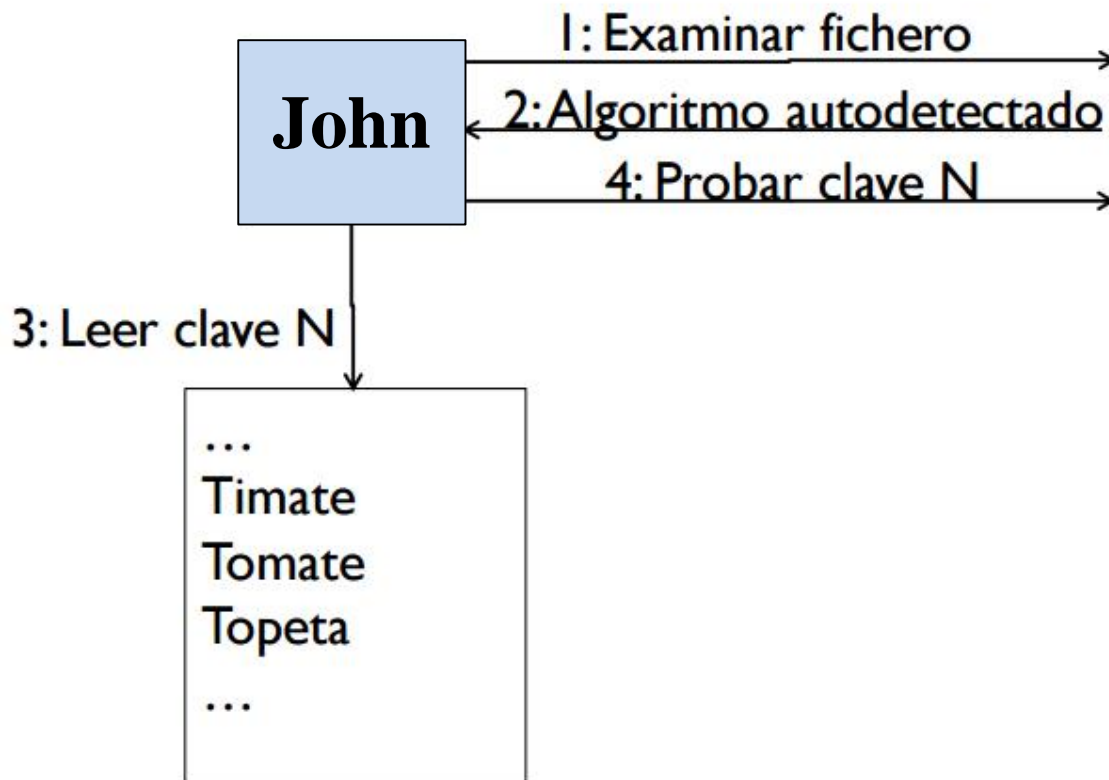
Password cracking

- Probar esta técnica para averiguar passwords “a mano” es una locura
- Para eso existen herramientas
 - Probablemente la más famosa sea John the Ripper (<http://www.openwall.com/john/>)
 - Ophcrack es también muy conocida: <http://ophcrack.sourceforge.net/>
- Dos versiones: la original y la community-enhanced version (Jumbo patch)
- La segunda es mucho más versátil y potente
- Puede crackear passwords de más sistemas operativos, encriptadas con muchos más algoritmos, con funcionalidades más avanzadas
- Otra herramienta de potencia equivalente con los mismos fines es hashcat: <http://hashcat.net/oclhashcat-plus/>

Password cracking

- Romper claves en cualquier sistema sin la autorización (contrato) del propietario de un sistema es ILEGAL
- Es un proceso que además puede lleva a un sistema a un DoS
- John necesita muchos recursos
- Por tanto para empezar a trabajar necesitamos:
- Estar seguros de que podemos usar John legalmente
- Un fichero de passwords sobre el que trabajar, cuyas claves estén cifradas en un formato admitido por el programa
- Una máquina de CPU potente, idealmente dedicada a esta tarea

Password cracking



The screenshot shows a terminal window titled "Ubuntu SSW [Corriendo] - Oracle VM VirtualBox". The terminal output is as follows:

```
Máquina - Ver - Dispositivos - Ayuda
Graph this data and manage this system at https://landscape.canonical.com/
malmuser@ubuntu:~$ cat /etc/shadow
cat: /etc/shadow: Permisos denegados
malmuser@ubuntu:~$ sudo cat /etc/shadow
[sudo] password for malmuser:
root:$1$419:0:99999:7:::
daemon:$1$419:0:99999:7:::
bin:$1$419:0:99999:7:::
lpr:$1$419:0:99999:7:::
games:$1$419:0:99999:7:::
man:$1$419:0:99999:7:::
lp:$1$419:0:99999:7:::
mail:$1$419:0:99999:7:::
news:$1$419:0:99999:7:::
uucp:$1$419:0:99999:7:::
printing:$1$419:0:99999:7:::
www-data:$1$419:0:99999:7:::
backup:$1$419:0:99999:7:::
list:$1$419:0:99999:7:::
irc:$1$419:0:99999:7:::
gnats:$1$419:0:99999:7:::
nobody:$1$419:0:99999:7:::
libbsd:$1$419:0:99999:7:::
nslcd:$1$419:0:99999:7:::
landscape:$1$419:0:99999:7:::
malmuser:$6$Q8fcdp0:24268:30attfqt_0pBX4s0w46LS2wQYow/E982xY5He0IuXJ2128FkeJ3y
D:821uG2r8gntIeKfrt2F1Q4U/1:15419:0:99999:7:::
malmuser@ubuntu:~$
```


Password cracking

- Las mejores herramientas (como John) son capaces de autodetectar el tipo de cifrado usado para las claves en el fichero estudiado
- Por tanto, sólo es necesario especificar un modo para generar passwords en texto plano
- Se hashea/encrpta cada posible password con el algoritmo autodetectado que se usó para generar las claves en el sistema investigado
- Se compara el resultado de la encriptación con la existente en el fichero de passwords

Password cracking

- Tienen muchos modos de funcionamiento
- Y varios de ellos dotan de “inteligencia” a la herramienta
- No se trata solo de generar cadenas y probar
- Acota la cantidad de posibles claves generadas, reduciendo el tiempo y aumentando el éxito al encontrar claves
- Pero al final, una vez configurado, básicamente se arranca y se deja funcionar durante horas/días/semanas

Password cracking

- Existen muchos ejemplos de uso de John en:
<http://www.openwall.com/john/doc/EXAMPLES.shtml>
- Antes de empezar hay que tener un fichero con el que trabajar
- Si el sistema usa shadow password se consigue con el programa unshadow (distribuido con John):
unshadow /etc/passwd /etc/shadow > mypasswd
- Es necesario que el fichero generado sea accesible para cuentas no root
- La forma de conseguir el fichero de trabajo varía según el tipo de encriptación de las passwords:
- Para passwords encriptadas con Kerberos AFS, se debe usar la herramienta unafs
- Para passwords de Windows, se debe usar la herramienta pwdump (<http://www.openwall.com/passwords/pwdump>)

Password cracking

- Toda la configuración (fichero de palabras a usar, comportamiento de los modos de crackeo, reglas de manipulación de claves, etc.) se hace editando el fichero .conf de John
- (<http://www.openwall.com/john/doc/CONFIG.shtml>)
- Hecho esto, simplemente se lanza el programa y se deja funcionar un tiempo preestablecido
- Al finalizar el mismo, tendremos un conjunto de claves de usuario que no son lo suficientemente fuertes, con el que ya podemos hacer nuestro informe.

Password cracking

- El modo por defecto (sin opciones) empieza con el modo “single crack”, luego lista de palabras con reglas y termina con “incremental”
 - `umask 077`
 - `unshadow /etc/passwd /etc/shadow > mypasswd`
 - `john mypasswd`
 - `john --show mypasswd`
- Cuanto mejor y más grande sea la lista de palabras más opciones habrá de romper las claves. Se venden listas de palabras enormes.
 - `john --wordlist=listadepalabras.lst mypasswd`

Verificación de contraseñas

Rainbow Tables

Contraseñas en Windows

- Las contraseñas en sistemas windows se almacenan de manera local en un fichero denominado **SAM** ubicado en **windows\system32\config**.
- Sistemas operativos como Windows XP utilizan la autenticación LM (**L**an **M**anager) para almacenar las contraseñas.
- Versiones posteriores han utilizado la autenticación NTLM aunque sigue teniendo debilidades.
- La autenticación LM utiliza un método débil para almacenar el hash de una contraseña.
- Esto hace que las contraseñas puedan ser averiguadas en pocos segundos utilizando **Tablas Rainbow** o en varias horas mediante ataques de **fuerza bruta**.

Debilidades de Lan Manager

- Las principales debilidades del protocolo LM y de sus hashes radica en lo siguiente:
 - **Las contraseñas no son case-sensitive:** Todas las contraseñas son convertidas a mayúsculas antes de generar la función hash. Además los caracteres de la contraseña están limitados a un subconjunto de caracteres del código ASCII.
 - **La longitud de las contraseñas está limitada a un máximo de 14 caracteres.**
 - **Cada contraseña es almacenada en dos mitades de 7 caracteres:** Sobre cada una de estas mitades se calcula el hash de manera separada. Esta forma de calcular el hash lo hace exponencialmente más fácil de crackear.

Debilidades de Lan Manager (y II)

- Las principales debilidades del protocolo LM...:
 - **Si la contraseña tiene 7 caracteres o menos, la segunda mitad del hash siempre producirá el mismo valor constante (0xAAD3B435B51404EE).** Una contraseña de 7 caracteres o menos puede identificarse visiblemente casi sin utilizar herramientas de crackeo.
 - **La contraseña es enviada por la red sin “salar” (salting).**

Formas de crackear la contraseña

- Una forma sería mediante fuerza bruta:
 - Este método, realiza una comprobación una a una de todas las claves posibles, lo que por norma general (dependiendo de lo bien elegida que esté la contraseña) hace que el proceso tarde demasiado.
 - Por eso siempre se insiste en utilizar letras mayúsculas y minúsculas, cifras y letras, y una longitud de más de 11 caracteres, como mínimo.
 - Con este método se podría necesitar varias horas para crackear la contraseña.

Formas de crackear la contraseña

- Otra forma sería mediante **Tablas Rainbow**:
 - Las **Tablas Rainbow** están pregeneradas y permiten ahorrar tiempo de ejecución sacrificando espacio en disco y memoria.
 - Se utilizan de apoyo para realizar la fuerza bruta de una manera más eficiente y mucho más rápida, reduciendo el tiempo de ejecución considerablemente.
 - Existen **Tablas Rainbow** (generalmente gigantes, pueden ocupar muchos gigas o incluso teras) para cada algoritmo (LM, MD5, SHA1, NTLM, MD2, MD4, RIPEMD160, etc.) previamente computadas.

Tablas Rainbow

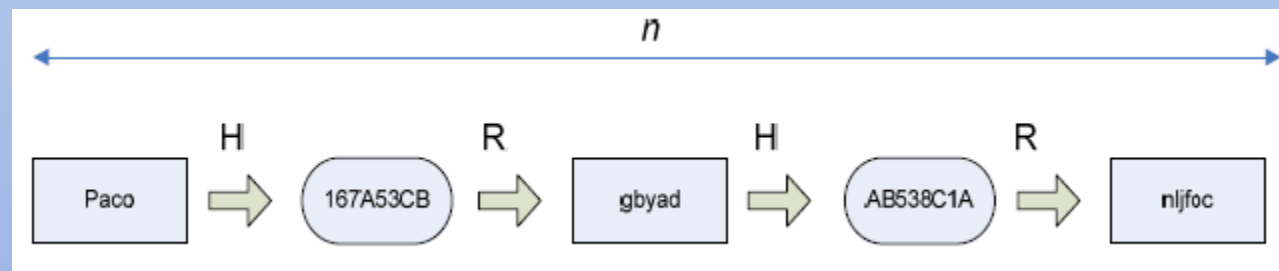
- Funcionamiento:
 - Las Tablas Rainbow son tablas que se generan previamente para obtener una contraseña con una determinada longitud y conjunto de caracteres.
 - El problema y dificultad para obtener las contraseñas, es el disponer de las tablas suficientemente grandes para romper cualquier tipo de contraseña.

Tablas Rainbow

Conjunto de Caracteres	Tamaño en Disco	Tiempo de Ejecución
0123456789	859.375 Kb	7,357 Segundos
ABCDEFGHIJKLMNOPQRSTUVWXYZ	610,35 Mb	14,86 Horas
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789	5,96 Gb	6,19 Días
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !@#\$%^&*()-_+=	59,6 Gb	61,9 Días
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !@#\$%^&*()-_+=~`[]{} \:;'"<>,.?/	521,54 Gb	541 Días

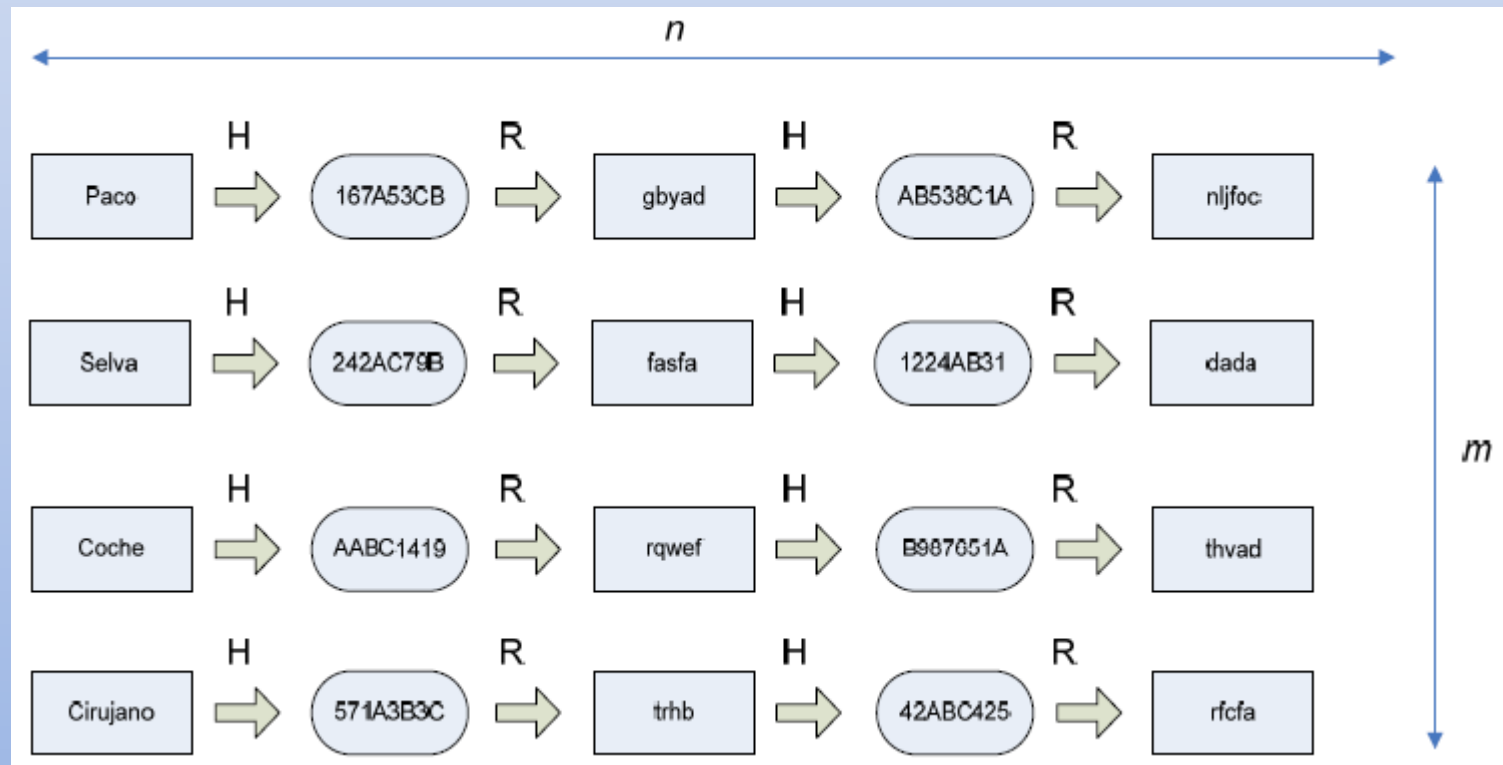
Tablas Rainbow

- Funcionamiento...:
 - Las **Tablas Rainbow** tienen su predecesor en las cadenas de Hash inventadas por Hellman Martin.
 - Estas cadenas se basan en la utilización de dos funciones básicas:
 - **Función Reductora (R)**: Es una función encargada de convertir un hash en una cadena de menor tamaño.
 - **Función Hash (H)**: Es una función encargada de realizar los hash a cada cadena en texto plano.
 - Mediante el uso combinado de ambas funciones se generan cadenas de Hash de longitud n .



Tablas Rainbow

- Funcionamiento...:
 - Para generar una Tabla de Hashes de tamaño $n*m$ habrá que repetir el proceso anterior m veces.



Tablas Rainbow

- Funcionamiento...:
 - Posteriormente se guarda el primer y último valor de cada fila en un archivo y se ordena cada fila en orden alfabético de acuerdo al valor final.

Valor Inicial	Valor Final
Selva	dada
Paco	nljfoc
Cirujano	rfcfa
Coche	thvad

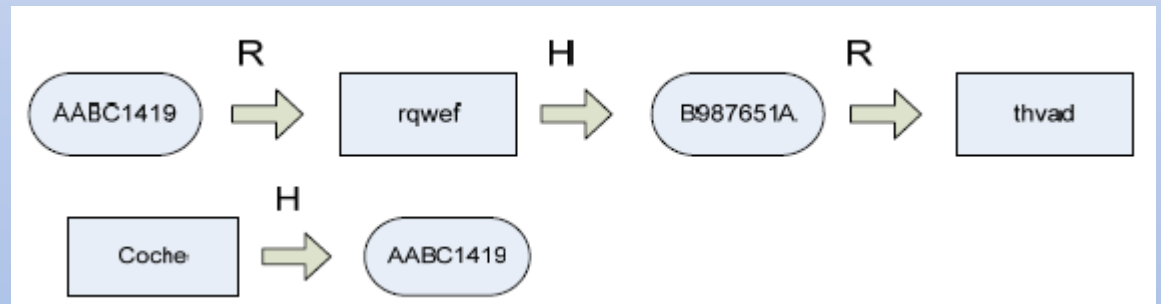
Tablas Rainbow

- Funcionamiento...:
 - La ordenación es imprescindible ya que gracias a ellas se pueden realizar búsquedas binarias que optimicen el trabajo de ruptura del hash.
 - **La tabla una vez generada se puede utilizar para romper múltiples hash.**
 - Para romper un hash hay que ir creando cadenas de hash a partir del hash dado hasta encontrar un valor final que coincida con alguno de los valores finales de la tabla.

Tablas Rainbow

- Funcionamiento....:
 - Ej.: ¿Cuál es la clave correspondiente al hash AABC1419?

Valor Inicial	Valor Final
Selva	dada
Paco	nljfoc
Cirujano	rfcfa
Coche	thvad



Tablas Rainbow: Algoritmo

1. Comprobamos si el hash buscado está en la lista de hashes finales. Si es así saltamos a 4.
2. Si no está en la lista de hashes finales se le aplica una función de reducción y se vuelve a calcular el hash de esa reducción.
3. Saltamos a 1
4. Si el hash se corresponde con uno de los hashes finales obtenemos la cadena correspondiente a ese hash de la primera columna.
5. Aplicamos un hash a la cadena obtenida.
6. Si su hash es igual al original hemos encontrado la cadena buscada. Saltamos a 8.
7. Se aplica una función de reducción al hash obtenido en el paso anterior y se obtiene una nueva cadena. Saltamos a 5.
8. Devolvemos la última cadena a la que le hemos aplicado el hash.

Criptografía Asimétrica

Criptografía Asimétrica

- **Ventajas**

- El principal problema que se encuentra al establecer una comunicación cifrada es establecer una clave de forma secreta.
- Con un esquema de criptografía asimétrica (o de clave pública y privada) se puede transmitir la clave pública por un entorno no seguro sin comprometer la seguridad de la comunicación

- **Desventajas:**

- Para el mismo tamaño de clave y mensaje se necesita mayor tiempo de proceso que en la criptografía simétrica.
- Las claves deben ser de mayor tamaño que en la simétrica
- Los mensajes encriptados suelen ocupar mayor tamaño que en la simétrica

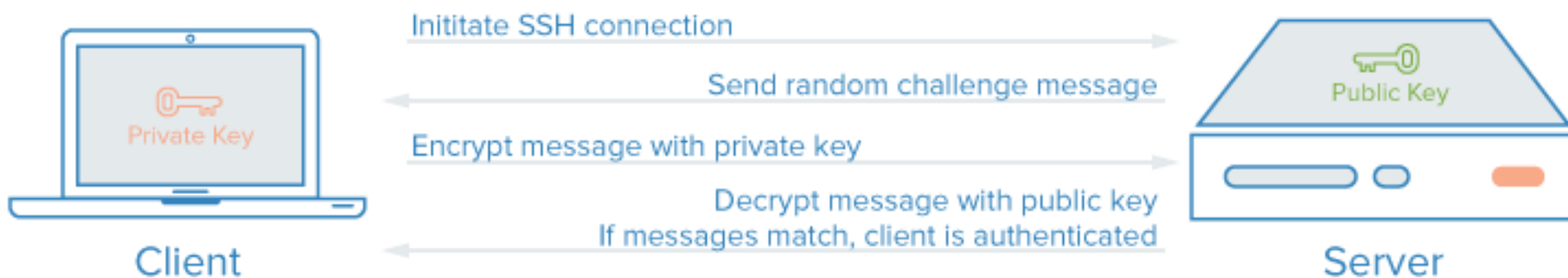
Criptografía Asimétrica

- Debido a estas ventajas e inconvenientes muchos sistemas utilizan un modelo híbrido, por ejemplo ssh:
 - Mediante criptografía asimétrica cliente y servidor intercambian información para ponerse de acuerdo en una clave.
 - Utilizan esa clave en un algoritmo de criptografía simétrica, menos costoso computacionalmente y que genera mensajes más cortos (lo que redundaría en una mayor eficiencia)

SSH mediante Public Key

- Es posible establecer una conexión SSH sin utilizar contraseñas mediante el uso de claves públicas y privadas.
 - Es más seguro que el utilizar contraseñas (suelen ser cortas, no muy buenas, fácilmente atacables con la potencia de hoy en día)
- Para ello es necesario que previamente el servidor reconozca nuestra clave pública como de confianza.

SSH Key Authentication



SSH mediante Public Key

- El primer paso es generar nuestras claves publica y privada
 - `mkdir ~/.ssh`
 - `chmod 700 ~/.ssh`
 - `/usr/bin/ssh-keygen -t rsa`
- Con esto se habrán generado la clave pública y privada (en el directorio .ssh)
 - `id_rsa` (clave privada)
 - `id_rsa.pub` (clave pública)

SSH mediante Public Key

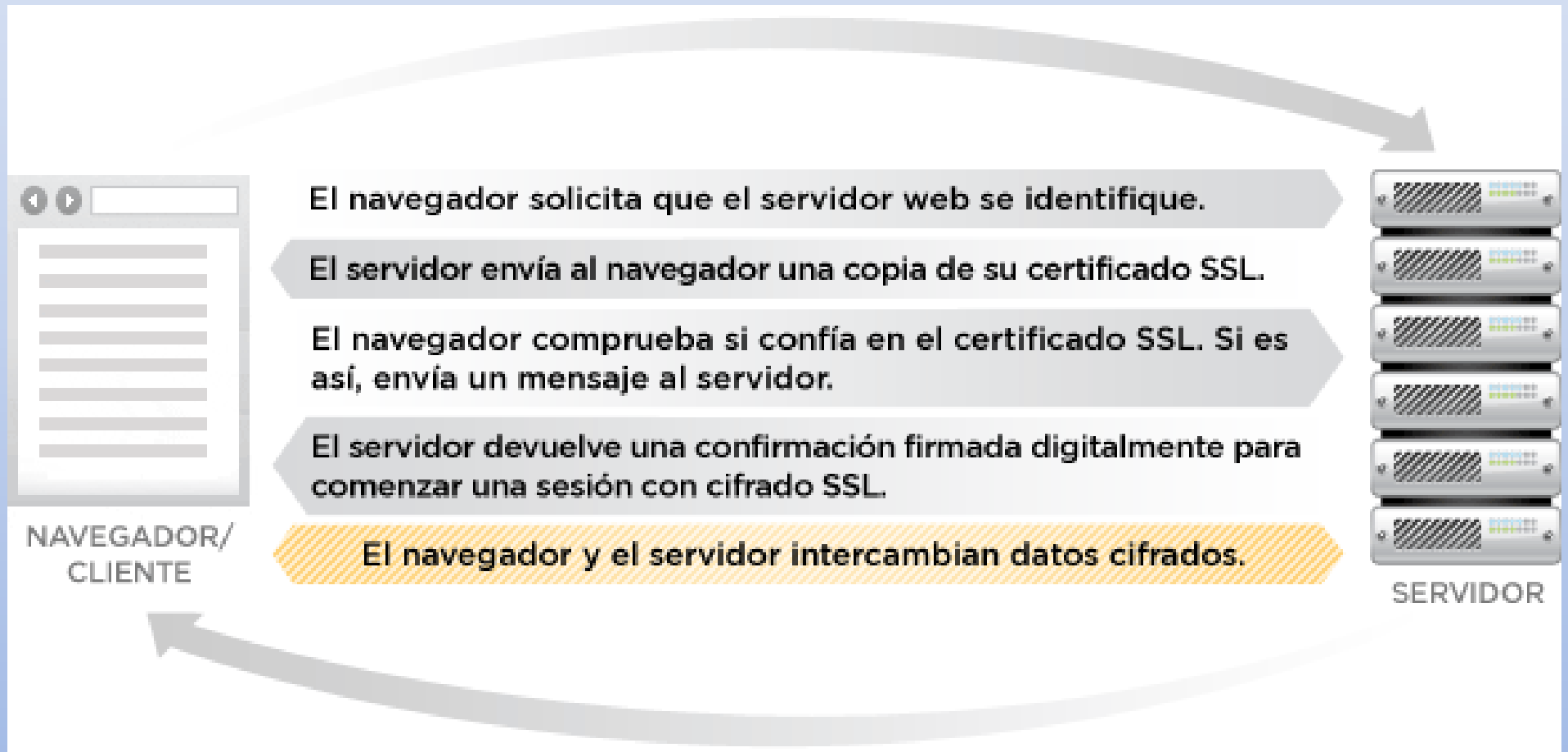
- Para poder acceder al servidor sin contraseña deberemos añadir la clave pública al fichero `~/.ssh/authorized_keys` del usuario remoto al que nos vamos a conectar.
 - `cd ~/.ssh`
 - `cat id_rsa.pub >> authorized_keys`
 - `scp authorized_keys SERVIDOR:/home/USUARIO/.ssh/`

Certificados Digitales

Garantías que aportan los certificados

- **¿Qué es un certificado digital?**
 - El certificado digital es un fichero digital, no modificable e intransferible, emitido por una entidad certificadora (CA) de confianza, que asocia a una entidad (persona, servidor, etc.) una clave pública.
- **Los certificados garantizan....:**
 - **Autenticación del servidor:** Se garantiza la autenticidad de la clave pública asociada al certificado.
 - **Confidencialidad:** La información transmitida en la comunicación entre el cliente y el servidor sólo sea legible por estas dos entidades.
 - **Integridad:** Asegurando que la información transmitida en la comunicación entre el cliente y el servidor no haya sido alterada en su viaje por la red.

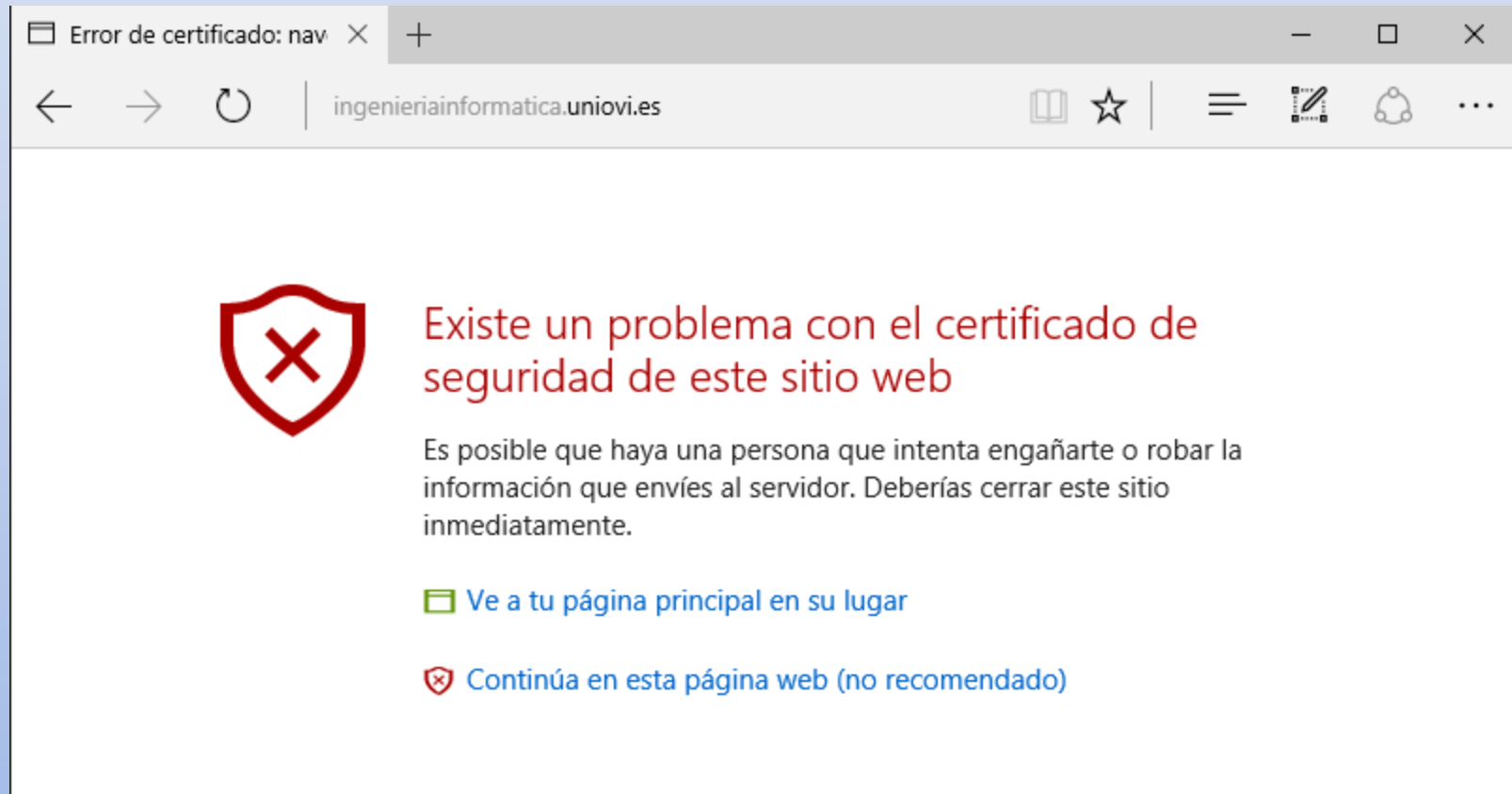
Proceso de validación del certificado



Proceso de validación del certificado

- Para que la conexión segura se realice sin problemas entre el servidor Web y el navegador, este deberá comprobar:
 - **Que haya sido emitido por una CA reconocida por el navegador.** Los certificados raíz de las principales CA están incluidos, por defecto, en los navegadores.
 - **Que el nombre del certificado coincida con el nombre del web al que el navegador se está conectando,** es decir, que el certificado haya sido emitido para el nombre de servidor al que el navegador se está conectando.
 - **Que no haya sido emitido para una fecha posterior o que no esté caducado.** Los certificados tienen un periodo de vida útil asignado por la autoridad de certificación (normalmente 1 ó 2 años). En el momento de la conexión la hora actual tiene que estar entre ambas fechas, es decir, en el periodo de vida útil del certificado.

Proceso de validación del certificado



Proceso de validación del certificado

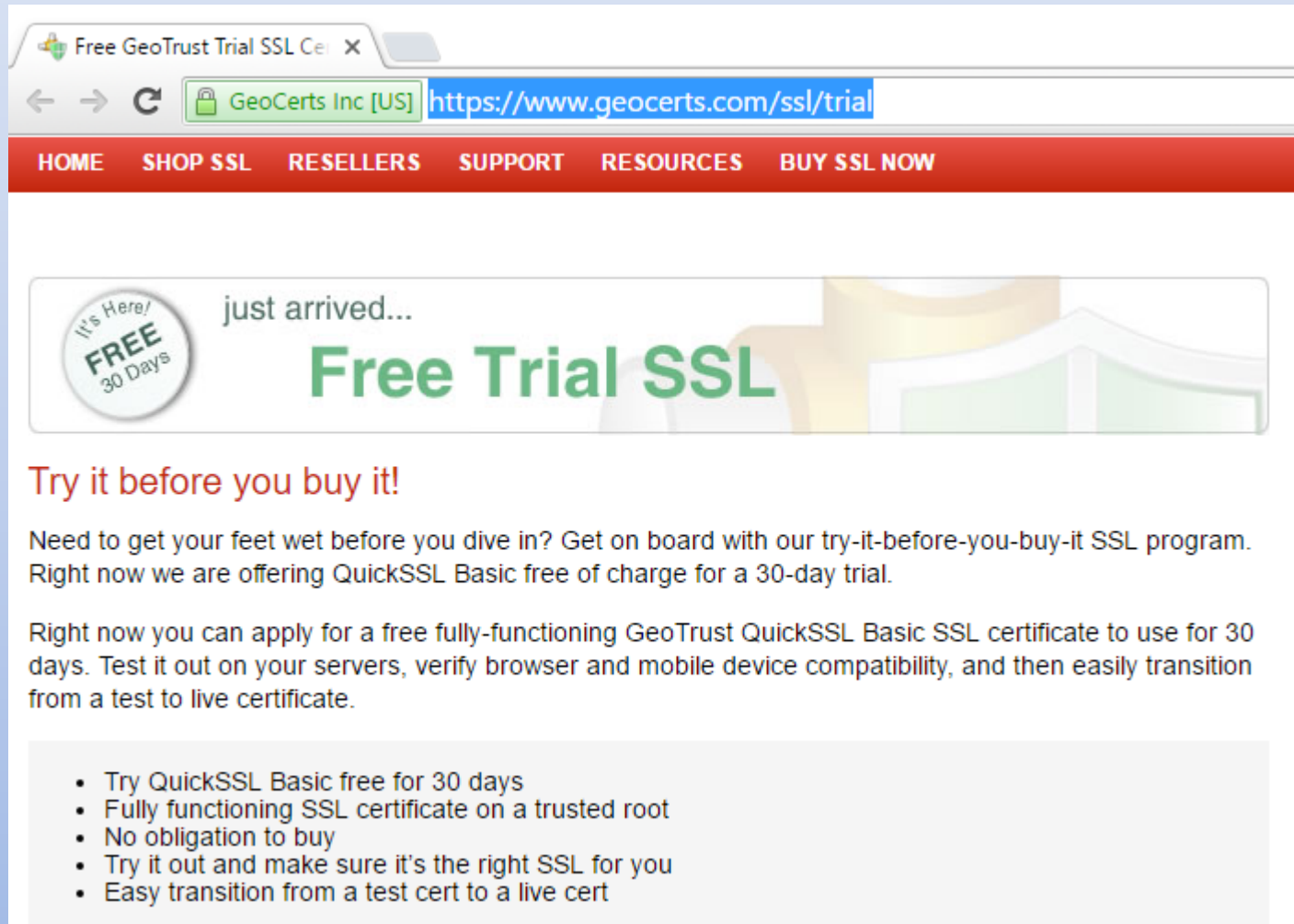
- **Comprobación de la validez del certificado....:**

- Aunque la mayoría de las veces, el navegador comprueba sin problemas la validez del certificado digital, hay ocasiones en las que, aunque queramos acceder a un sitio que sabemos es de confianza, el navegador nos dice que no es confiable.
- Esto sucede, habitualmente, porque el navegador no dispone del certificado raíz de la autoridad que ha creado el certificado digital con el que se está identificando el sitio web.
- En ese caso debemos indicar al navegador, de forma manual, que sí confiamos en el sitio y en la autoridad que lo certifica.
- También podemos añadir un certificado como de confianza para que desde ese momento se confíe en él.

Proceso de validación del certificado

- El error antes descrito es debido a que la máquina utiliza un certificado *autofirmado*. Es muy común, porque son gratuitos (cualquiera puede crearlo), mientras que los certificados emitidos por una autoridad certificadora de confianza suelen comprarse.
- Para pruebas, pueden solicitarse certificados reconocidos.
- O desde hace poco tiempo puede crearse un certificado gratuito reconocido.

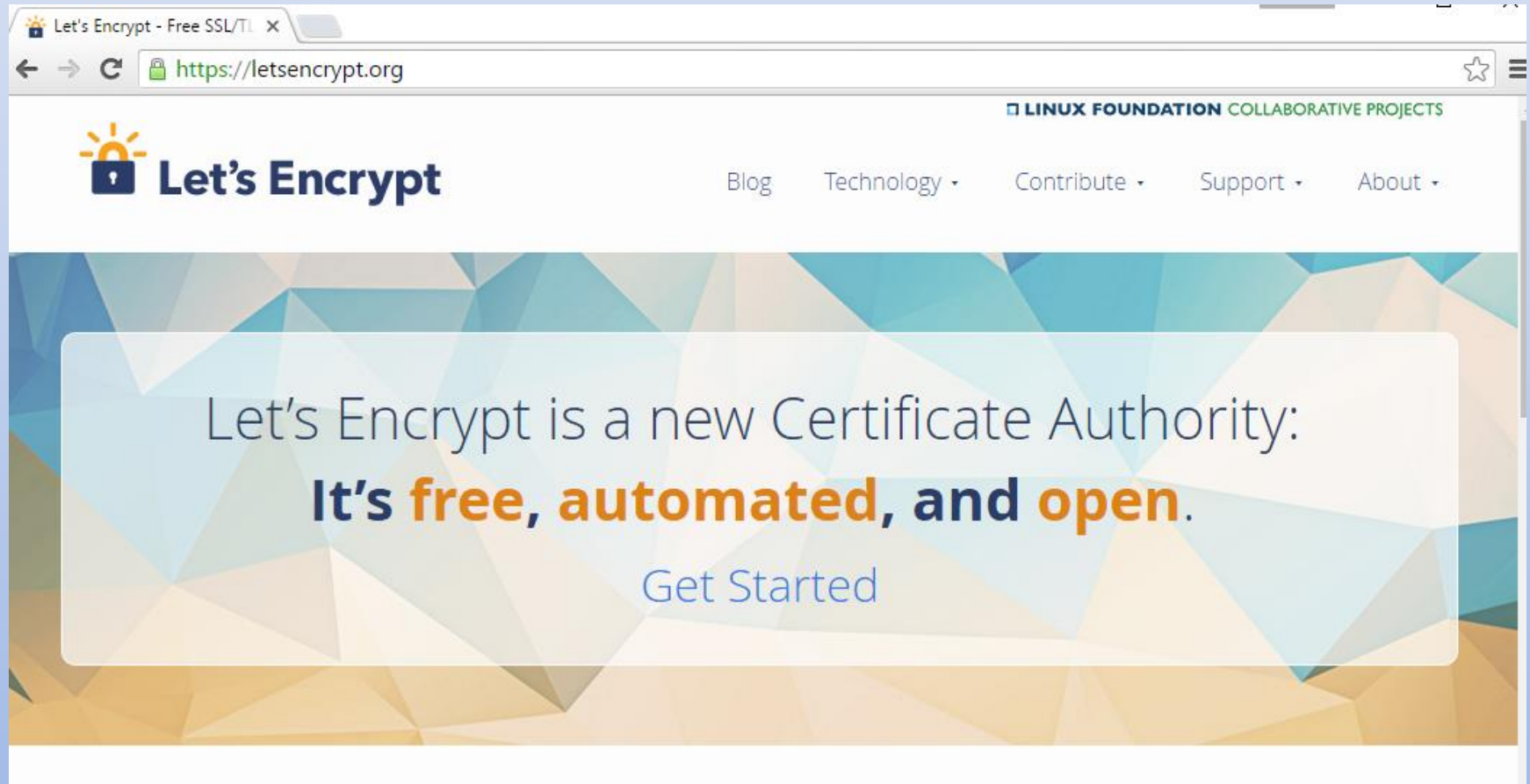
Creación de certificados: adquisición



The screenshot shows a web browser window with the address bar displaying "https://www.geocerts.com/ssl/trial". The page features a red navigation bar with links: HOME, SHOP SSL, RESELLERS, SUPPORT, RESOURCES, and BUY SSL NOW. Below the navigation bar is a large banner with a circular badge that says "It's Here! FREE 30 Days" and the text "just arrived... Free Trial SSL". The main content area includes the heading "Try it before you buy it!" and a paragraph explaining the offer: "Need to get your feet wet before you dive in? Get on board with our try-it-before-you-buy-it SSL program. Right now we are offering QuickSSL Basic free of charge for a 30-day trial." Below this, another paragraph states: "Right now you can apply for a free fully-functioning GeoTrust QuickSSL Basic SSL certificate to use for 30 days. Test it out on your servers, verify browser and mobile device compatibility, and then easily transition from a test to live certificate." At the bottom, a list of bullet points details the offer:

- Try QuickSSL Basic free for 30 days
- Fully functioning SSL certificate on a trusted root
- No obligation to buy
- Try it out and make sure it's the right SSL for you
- Easy transition from a test cert to a live cert

Creación de certificados: “adquisición”



Creación de certificados: “adquisición”

- Let's Encrypt es una propuesta de Linux Foundation que permite obtener certificados reconocidos de manera gratuita.
- El proceso es un poco laborioso pero al final se obtiene un certificado firmado por una autoridad certificadora confiable y de manera gratuita.
- La solicitud y obtención del certificado se realiza ejecutando en la máquina donde se instalará el certificado un software específico.

Niveles de seguridad SSL

- La seguridad que proporciona SSL se basa principalmente en la fuerza de las claves.
- Se mide la fuerza del protocolo en el número de bits que utilizamos para codificar la clave.
- Las primeras implementaciones de SSL utilizaban claves de 40 bits de longitud (criptografía simple).
- No es muy difícil crackear claves de este tamaño utilizando la potencia de cálculo de los ordenadores actuales.
- Por ello pronto se vio la necesidad de aumentar la fuerza de la encriptación aumentando el tamaño de la clave (por ejemplo en 2013 Google pasó a usar 2048 bits).
- El algoritmo siempre es el mismo, para aumentar su fortaleza se aumenta el tamaño de la clave. Implica más tiempo de computación en las operaciones.

Niveles de seguridad SSL

```
ssh-keygen -t rsa
```

Genera por defecto una clave de 2048 bits

```
ssh-keygen -t rsa -b 768
```

Genera una clave de 768 bits (no recomendable).

Otros tipos pueden ser

- dsa
- ecdsa
- ed25519 rsa
- rsa1

Niveles de seguridad SSL

- En caso de que el cliente soporte criptografía simple y el servidor criptografía robusta la comunicación se realizará conforme a las características de criptografía simple que soporte el cliente. (siempre y cuando se configure el servidor para comportarse de esta manera)