# Seguridad en Sistemas de Información

Curso en Tamaulipas [Q2 2014]


Luis J. Dominguez Perez

*Cinvestav*, Junio 02 de 2014 - L5

# Contenido, sección 1

# Servicios Básicos de seguridad

Los servicios más importantes son:

- Confidencialidad
- Integridad
- Autenticación de mensajes
- No repudiación

# Otros servicios de seguridad

Existen otros servicios opcionales que dependen de la aplicación:

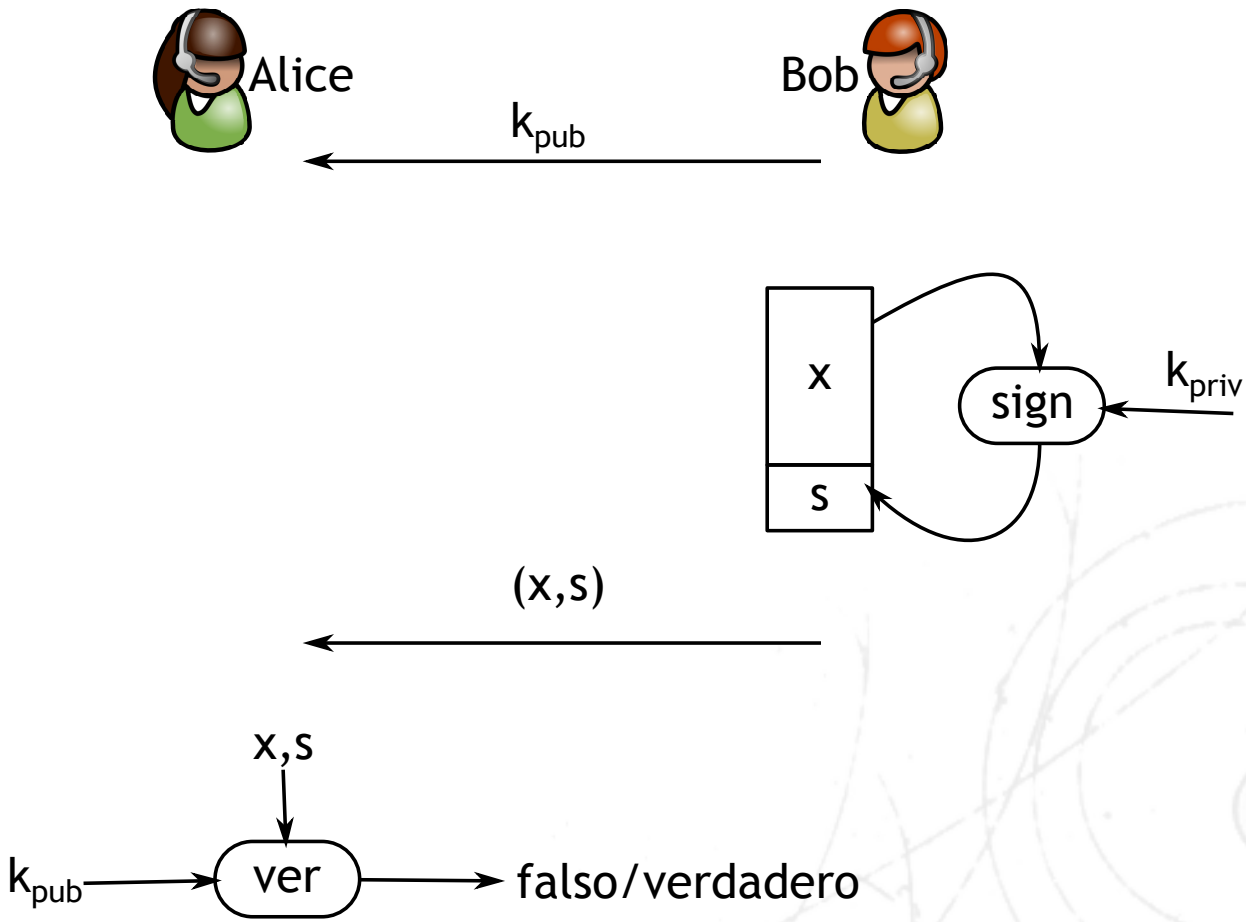
- Identificación o autenticación de entidades
- Control de acceso
- Disponibilidad
- Auditoría
- Seguridad física
- Anonimato

# Firmas digitales

- La propiedad de demostrar que cierta persona generó un mensaje es crítica en muchas aplicaciones.
- En el mundo "analógico", se utilizan firmas a mano sobre papel.
- Sólo la persona que crea la firma, puede reproducirla.

# Esquema

Esto es posible mediante criptografía de clave pública. El signatario firma utilizando su clave privada, el receptor utiliza la clave pública para verificar.



# RSA

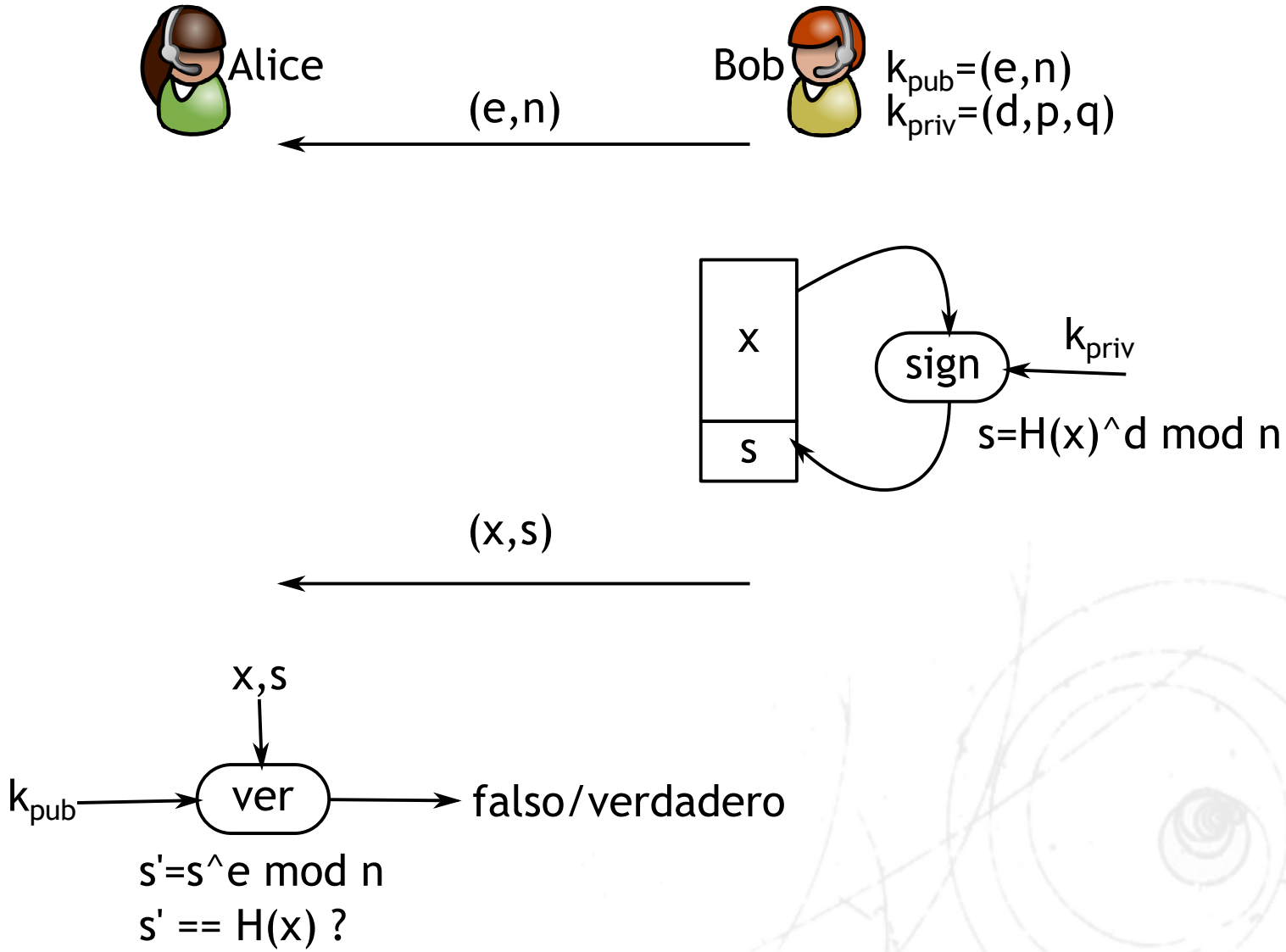
Diseñado en 1977 por Ron Rivest, Adi Shamir, y Leonar Adleman.

- Sean $p$ y $q$ dos diferentes grandes números primos aleatorios
- El módulo $n$ es el producto de $p$ y $q$
- La función $\Phi(n) = (p-1)(q-1)$
- Seleccionamos $1 < e < \Phi(n)$, tal que el MCD$(e, \Phi(n)) = 1$; $e = 2^{16} + 1$ típicamente
- Se calcula $d \equiv e^{-1} \bmod \Phi(n)$

La clave pública es $e$, y $n$. La clave privada es $d$, y los primos $p$ y $q$.

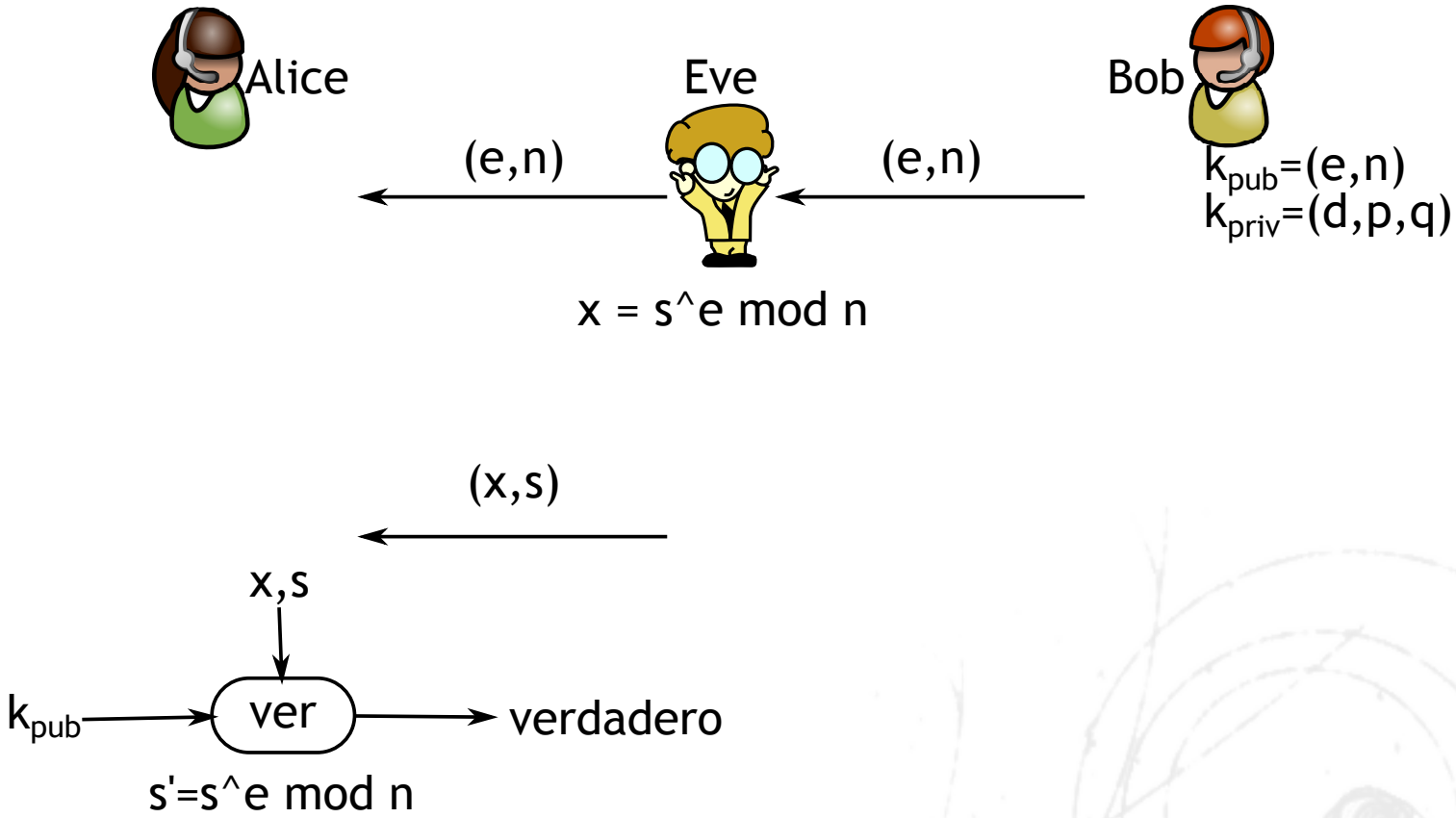
# Firma RSA

## Firma RSA básica



# Ataque a Firma RSA

Sobre validación de firmas ficticias



# ElGamal

- El cifrado Elgamal fue propuesto por Taher Elgamal en 1985.
- Es una extensión del intercambio de llaves de Diffie-Hellman (DHKE)

# Firma Elgamal

- Generación de llaves:
  - Generar un primo $p$
  - Encontrar un elemento $\alpha \in \mathbb{Z}_p^\star$
  - Seleccionar un elemento aleatorio $d$, con $2 < d < p - 2$
  - Calcular $\beta = \alpha^d \bmod p$

# Firma Elgamal de mensaje

- Firma de mensaje:
  - Dado un mensaje $M$
  - Seleccione una llave efímera $k_E$, con $0 < k_E < p - 2$, con MCD$(k_E, p - 1) = 1$
  - Calcule $r \equiv a^{k_E} \bmod p$
  - Calcule $s \equiv (M - d \cdot r)k_E^{-1} \bmod p - 1$

- La firma de $M$ es $(r, s)$

# Verificación de Firma Elgamal

- Verificación de firma:
  - Calcule $t \equiv \beta^r \cdot r^s \bmod p$

- Si $t \equiv \alpha^x \bmod pq$, la firma verificó.

# Ejemplo, $M$ a firmar

$$p = 29,\ \alpha = 2$$
$$d = 12$$
$$\beta = \alpha^d \equiv 7$$

$$k_{\text{pub}}(p,\alpha,\beta)=(29,2,7) \longleftarrow$$

$$k_E = 5$$
$$(5, 28) = 1$$
$$x = 26$$
$$r = 2^5 \equiv 3$$
$$s = -10 \cdot 7 \equiv 26$$

$$(26,(3,26)) \longleftarrow$$

$$t = 7^3 \cdot 3^{26} \equiv 22$$
$$\alpha^x \equiv 2^{26} \equiv 22$$
$$t \equiv \alpha^x \Rightarrow \text{OK}$$

# Firma DSA

La firma estándar DSA contiene los siguientes pasos:

- Generación de claves:
  - Generar un primo $p$, con $2^{1023} < p < 2^{1024}$
  - Encontrar un primo $q$ divisor de $p$, con $2^{159} < q < 2^{160}$
  - Encontrar un elementos $\alpha$, cuyo orden sea igual a $q$
  - Seleccionar un elemento aleatorio $d$, con $1 < d < q$
  - Calcular $\beta = \alpha^d \bmod p$

- Las claves son:
  - Pública: $(p, q, \alpha, \beta)$
  - Privada: $d$

# Firma DSA de mensaje

- Firma de mensaje:
  - Dado un mensaje $M$
  - Seleccione una llave efímera $k_E$, con $0 < k_E < q$
  - Calcule $r \equiv (a^{k_E} \bmod p) \bmod q$
  - Calcule $s \equiv (SHA(M) + d \cdot r)k_E^{-1} \bmod q$

- La firma de $M$ es $(r, s)$

# Verificación de Firma DSA

- Verificación de firma:
  - Calcule $w \equiv s^{-1} \bmod q$
  - Calcule $u_1 \equiv w \cdot SHA(M) \bmod q$
  - Calcule $u_2 \equiv w \cdot r \bmod q$
  - Calcule $v \equiv (\alpha^{u_1} \cdot \beta^{u_2} \bmod p) \bmod q$

- Si $v \equiv r \bmod q$, la firma verificó.

# Ejemplo, $M$ a firmar

$$p = 59,\ q = 29$$
$$\alpha = 3,\ d = 7$$
$$\beta = \alpha^d \equiv 4$$

$$\xleftarrow{k_{\text{pub}}(p,q,\alpha,\beta)=(59,29,3,4)}$$

$$k_E = 10$$
$$r = (3^{10} \bmod 59)$$
$$\equiv 20 \bmod 29$$
$$s = (26 + 7 \cdot 20) \cdot 3$$
$$\equiv 5 \bmod 29$$

$$\xleftarrow{(M,(r,s))}$$

$$w = 5^{-1} \equiv 6 \bmod 29$$
$$u_1 = 6 \cdot 26 \equiv 11 \bmod 29$$
$$u_2 = 6 \cdot 20 \equiv 4 \bmod 29$$
$$v = 20 \equiv (3^{11} \cdot 4^4 \bmod 59) \bmod 29$$
$$v \equiv r \bmod 29 \Rightarrow \text{OK}$$

# MIRACL

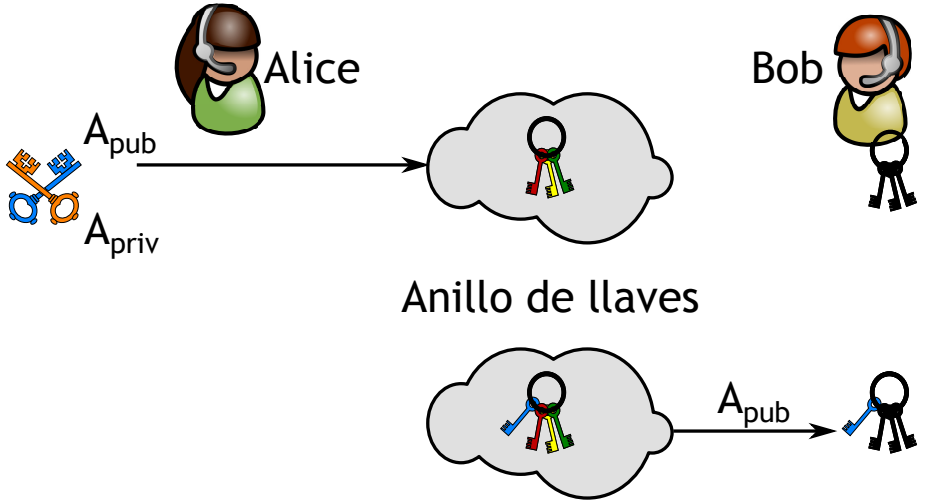
- dssetup.c
  - Generan $p$, $q$, y $g$
  - Archivo *common.dss*
- dssgen.c
  - Generan llaves pública y privada
  - Archivos *public.dss* y *private.dss*
- dssign.c
  - Firman archivo
  - Archivo nombre.dss
- dssver.c
  - Verifica nombre.dss

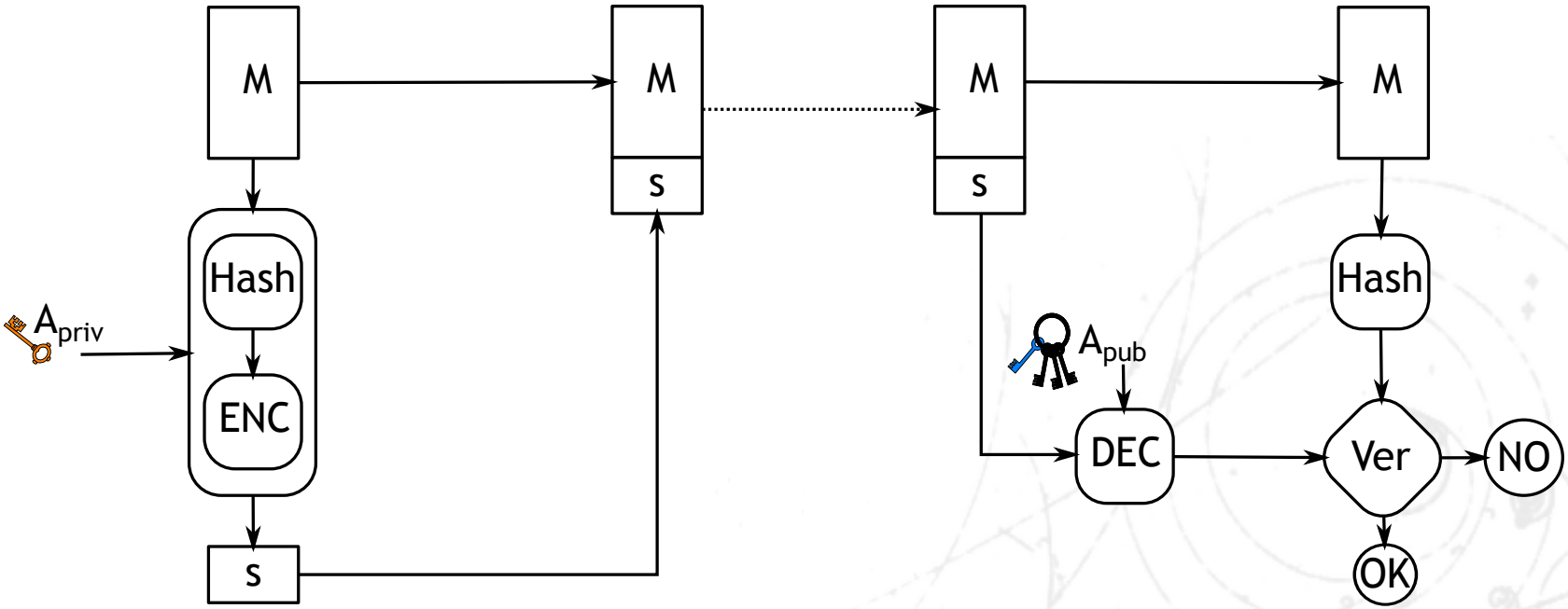
```
gcc -O2 -o dssetu dssetup.c miracl.a
```

# Otras firmas

- Un método particularmente eficiente en aplicaciones en donde el consumo eléctrico o el espacio es restringido son las firmas cortas basadas en curvas elípticas.
- La criptografía basada en curvas elípticas es un tipo de criptografía de clave pública que requiere menos espacio de la clave que sus contrapartes.
- La criptografía de curvas elípticas es mucho más elaborada, pero permite la implementación eficiente de protocolos de seguridad más interesantes, o a un menor costo.

# Firma Digital



# Formalizando un protocolo

- Los protocolos de seguridad contienen generalmente varios pasos para definirlos *formalmente*.

- Los pasos incluyen:
  - Setup
  - Key generation
  - Encryption
  - Decryption
  - Key delegation
  - Key revocation
  - . . .

Los pasos dependen del protocolo en particular

# Certificados digitales

Es un documento que mediante una firma digital de una entidad de confianza, previamente almacenada en el equipo solicitante, asocia una clave pública con una identidad: nombre de la persona, organización, dirección, etc.

El certificado sirve para garantizar que una clave pública en particular pertenece al que dice ser el poseedor de la contraparte privada.

Los certificados son emitidos por una entidad de confianza, una Autoridad Certificadora.

# Certificados digitales

Es un documento que mediante una firma digital de una entidad de confianza, previamente almacenada en el equipo solicitante, asocia una clave pública con una identidad: nombre de la persona, organización, dirección, etc.

El certificado sirve para garantizar que una clave pública en particular pertenece al que dice ser el poseedor de la contraparte privada.

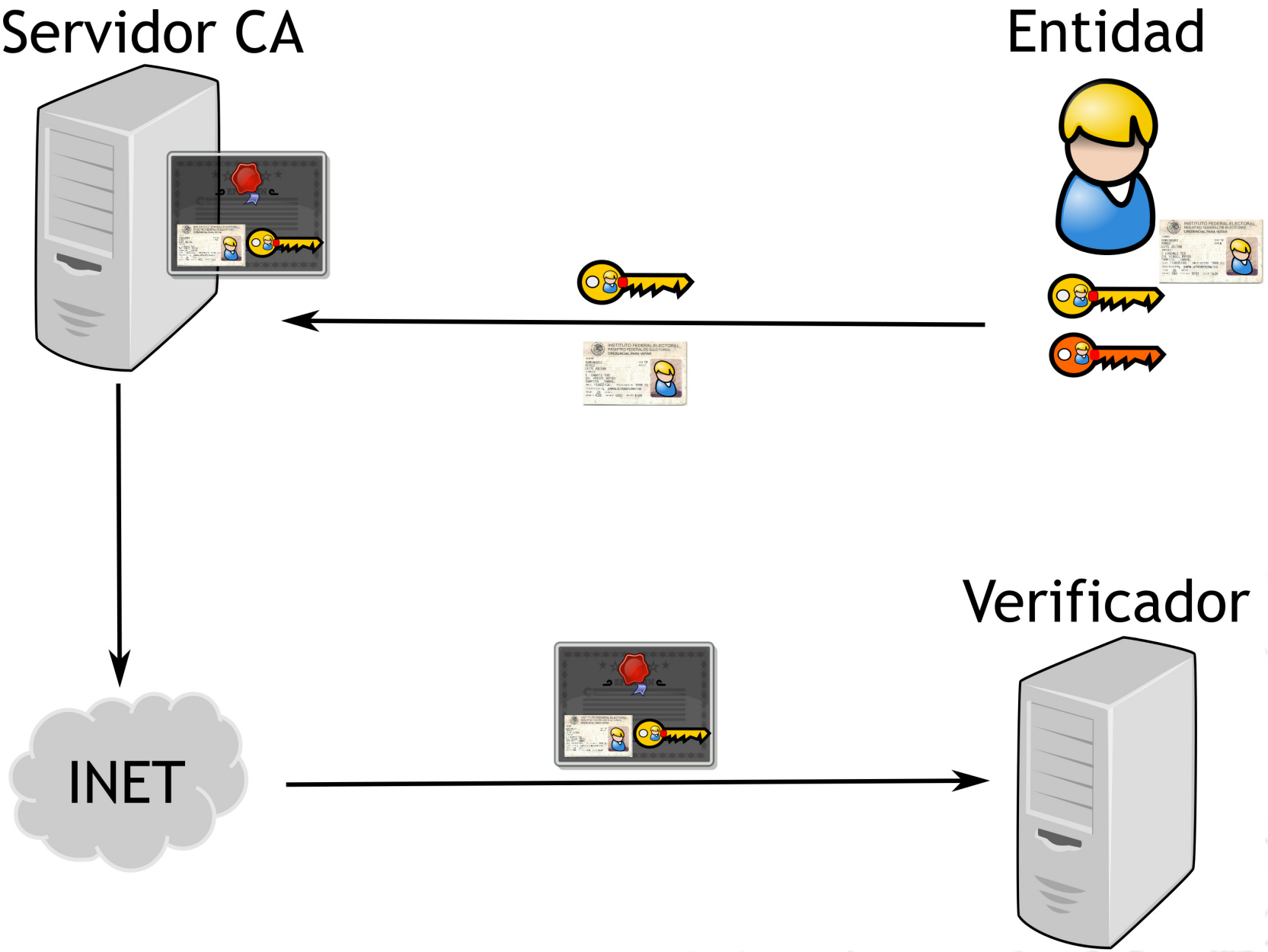
Los certificados son emitidos por una entidad de confianza, una Autoridad Certificadora... aunque en la práctica la relación de confianza se delega a Mozilla, Microsoft, Apple.

# Responsabilidades de una CA

Las responsabilidades básicas son:

- Generación de llaves (Intercambio seguro)
- Emisión de Certificados (¿Qué son?)
- Emisión de CRL's (¿Para qué sirven?)

# Certificados



# Contenido de un certificado

El estándar X.509 establece el formato ASN1 para los certificados digitales, que contienen:

- Número serial
- Sujeto: Persona o entidad identificada
- Algoritmo de firma digital
- Firma digital
- Emisor
- Inicio validez
- Fin validez
- Propósito de la llave: cifrado, firma digital, firma de certificados
- LLave pública
- Algoritmo de huella digital
- Huella digital

# Ejemplo

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            07:23:53:8d:87:6d:b6:27:fc:1e:08:aa:49:96:d9:60
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=DigiCert Inc, OU=www.digicert.com, CN=DigiCert High Assurance CA-3
        Validity
            Not Before: Oct  8 00:00:00 2012 GMT
            Not After : Dec 16 12:00:00 2015 GMT
        Subject: C=MX, ST=Distrito Federal, L=Mexico, O=Centro de Investigacion... CN=*.cinvestav.mx
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:d8:dc:9d:1a:7e:d4:6f:49:5b:7a:95:6a:57:6c:
                    05:8a:c1:0b:3f:b1:03:e0:1a:53:e5:22:8f:bd:6c:
                    c1:59:ec:13:68:5e:f2:6f:44:55:21:36:8c:82:d9:
                    84:4a:e7:97:55:84:f2:cf:71:ad:e4:e5:a6:73:5c:
                    be:5c:23:2d:ab:3b:5d:b7:c3:de:2f:0a:35:74:84:
                    46:23:39:20:78:d4:8b:47:eb:e1:d4:b4:c2:ab:59:
                    8d:7d:33:98:b3:f7:bf:3a:07:c0:64:8a:4f:a6:78:
                    55:87:13:a5:54:b5:e7:be:15:dc:da:9d:61:8c:06:
                    1f:e6:29:01:1e:ab:61:5d:bf:06:cb:ec:48:89:b0:
                    88:6f:e5:b0:4b:bf:83:bd:a0:58:bf:ff:33:0d:f8:
                    c7:73:ff:00:0b:64:f2:2b:9a:69:3f:d5:74:d3:12:
                    0f:e9:15:70:f8:7c:f1:2b:5c:70:d4:49:ce:01:c9:
                    65:47:5f:a2:8f:8f:fa:af:2a:00:c9:ec:20:fd:33:
                    90:12:5c:1c:46:2b:44:24:04:77:44:82:98:26:93:
                    d3:f3:53:a1:5e:a0:f5:f0:1f:f5:6b:22:27:94:a9:
                    2a:45:7d:73:6d:68:39:cf:d2:d2:60:3a:fd:6a:89:
                    2b:a5:22:06:22:46:c2:90:a6:8b:dd:95:61:7b:89:
                    b6:a7
```

```
        Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Authority Key Identifier:
        keyid:50:EA:73:89:DB:29:FB:10:8F:9E:E5:01:20:D4:DE:79:99:48:83:F7

    X509v3 Subject Key Identifier:
        37:92:15:14:C3:5C:87:5F:C4:63:E2:F3:20:C1:8F:0C:92:B7:BC:7D
    X509v3 Subject Alternative Name:
        DNS:*.cinvestav.mx, DNS:cinvestav.mx, DNS:www.tamps.cinvestav.mx,
         DNS:webmail.tamps.cinvestav.mx, DNS:noc.tamps.cinvestav.mx
    X509v3 Key Usage: critical
        Digital Signature, Key Encipherment
    X509v3 Extended Key Usage:
        TLS Web Server Authentication, TLS Web Client Authentication
    X509v3 CRL Distribution Points:

        Full Name:
          URI:http://crl3.digicert.com/ca3-g15.crl

        Full Name:
          URI:http://crl4.digicert.com/ca3-g15.crl

    X509v3 Certificate Policies:
        Policy: 2.16.840.1.114412.1.1
          CPS: http://www.digicert.com/ssl-cps-repository.htm
          User Notice:
            Explicit Text:

    Authority Information Access:
        OCSP - URI:http://ocsp.digicert.com
        CA Issuers - URI:http://cacerts.digicert.com/DigiCertHighAssuranceCA-3.crt

    X509v3 Basic Constraints: critical
        CA:FALSE
```

```
Signature Algorithm: sha1WithRSAEncryption
     89:72:14:45:fc:52:d2:46:12:ff:fa:f4:c5:4f:fd:7b:0e:e4:
     a7:d9:a1:6d:d4:4e:09:aa:c0:30:2f:1a:92:eb:0c:5b:6a:8f:
     58:26:59:bc:95:d7:73:28:36:47:d1:14:6e:e5:95:d1:ae:35:
     57:3d:2e:c2:9e:86:9f:08:47:a4:31:61:5d:4b:d6:3f:0a:60:
     0d:e4:f3:11:aa:69:9d:c1:6b:ed:ea:53:82:e0:b3:f7:cd:c4:
     d2:b5:5e:60:ef:35:d2:bb:19:68:84:c9:c0:82:8d:e1:80:e8:
     e8:0a:d0:d4:b0:b7:13:4f:43:24:e6:6f:37:4d:8b:f0:b9:0e:
     af:3c:d7:61:89:24:6b:8a:88:88:82:7e:de:4c:12:8a:64:2b:
     75:ca:18:e9:11:8f:7a:c4:0a:55:2a:d6:6a:a8:84:2e:6d:d9:
     f9:f5:fc:48:96:bf:e3:87:2c:02:41:ab:1a:6b:ce:e3:16:65:
     0a:08:56:a2:be:28:ea:47:d2:03:bb:28:ab:f1:b4:ec:62:44:
     cd:c4:14:5d:2c:13:21:6a:d0:6e:6c:29:ba:80:9c:08:a2:50:
     bb:7c:ac:56:41:c0:64:3e:2a:c3:e1:44:38:a0:31:2a:68:4b:
     43:02:27:eb:a5:87:71:e6:79:09:51:a6:82:83:28:30:0f:9a:
     d7:3d:5f:c6
```

# Demo

Ejercicio de creación de certificados.

- Generación de Parámetros DSA

```
openssl dsaparam 2048 -out dsaparams.pem
```

- Generación de Llaves

```
openssl gendsa -out dsarootkey.pem dsaparams.pem
```

- Generación de certificado raíz auto-firmado

```
openssl req -newkey dsa:dsaparams.pem -keyout
  dsarootkey.pem -new -x509 -days 365 -out
  rootcert.pem
```

- Examinando el certificado

```
openssl x509 -text -in rootcert.pem | more
openssl asn1parse -in rootcert.pem | more
```

- Generando certificado para el cliente

```
openssl req -newkey dsa:dsaparams.pem -keyout
  dsakey.pem -new -days 365 -out dsareq.pem
```

- Expedición del Certificado

```
openssl x509 -days 180 -CA rootcert.pem -CAkey
  dsarootkey.pem -req -CAcreateserial -CAserial
  ca.srl -in dsareq.pem -out newcert.pem
```

- Examinando el certificado emitido

```
openssl x509 -text -in newcert.pem | more
openssl asn1parse -in newcert.pem | more
```

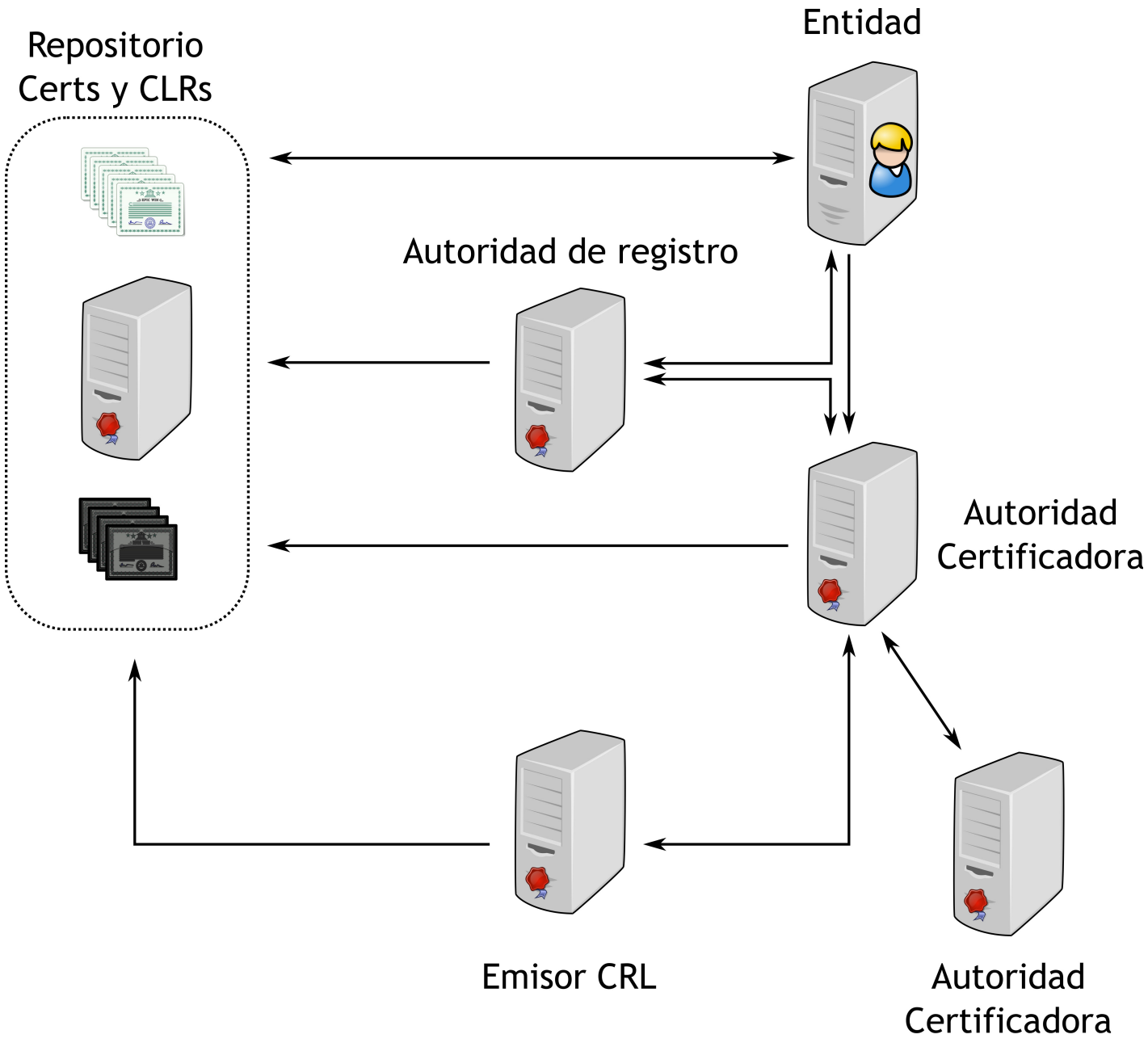
- Verificación del Certificado

```
openssl verify -CAfile rootcert.pem newcert.pem
```

# PKI

- La *Infraestructura de Llave Pública (PKI)* es una combinación de software, tecnologías de cifrado, y servicios que permiten proteger la seguridad de las transacciones de información en un sistema distribuido.
- PKI integra (mas bien lo intenta) certificados digitales, criptografía de llave pública y autoridades de certificación en una arquitectura de seguridad unificada.

# Diagrama PKI



# Elementos en la PKI

- **Entidad final**. Término genérico para denotar a los usuarios finales o cualquier entidad que pueda ser identificada (personas, servidores, compañías, etc.) mediante un certificado digital expedido por una Autoridad Certificadora.

- **Autoridad Certificadora (AC)**. La AC es la entidad que expide los certificados digitales, así como la lista de revocación (CRL). Adicionalmente puede soportar funciones administrativas, aunque generalmente éstas son delegadas a una o varias Autoridades de Registro.

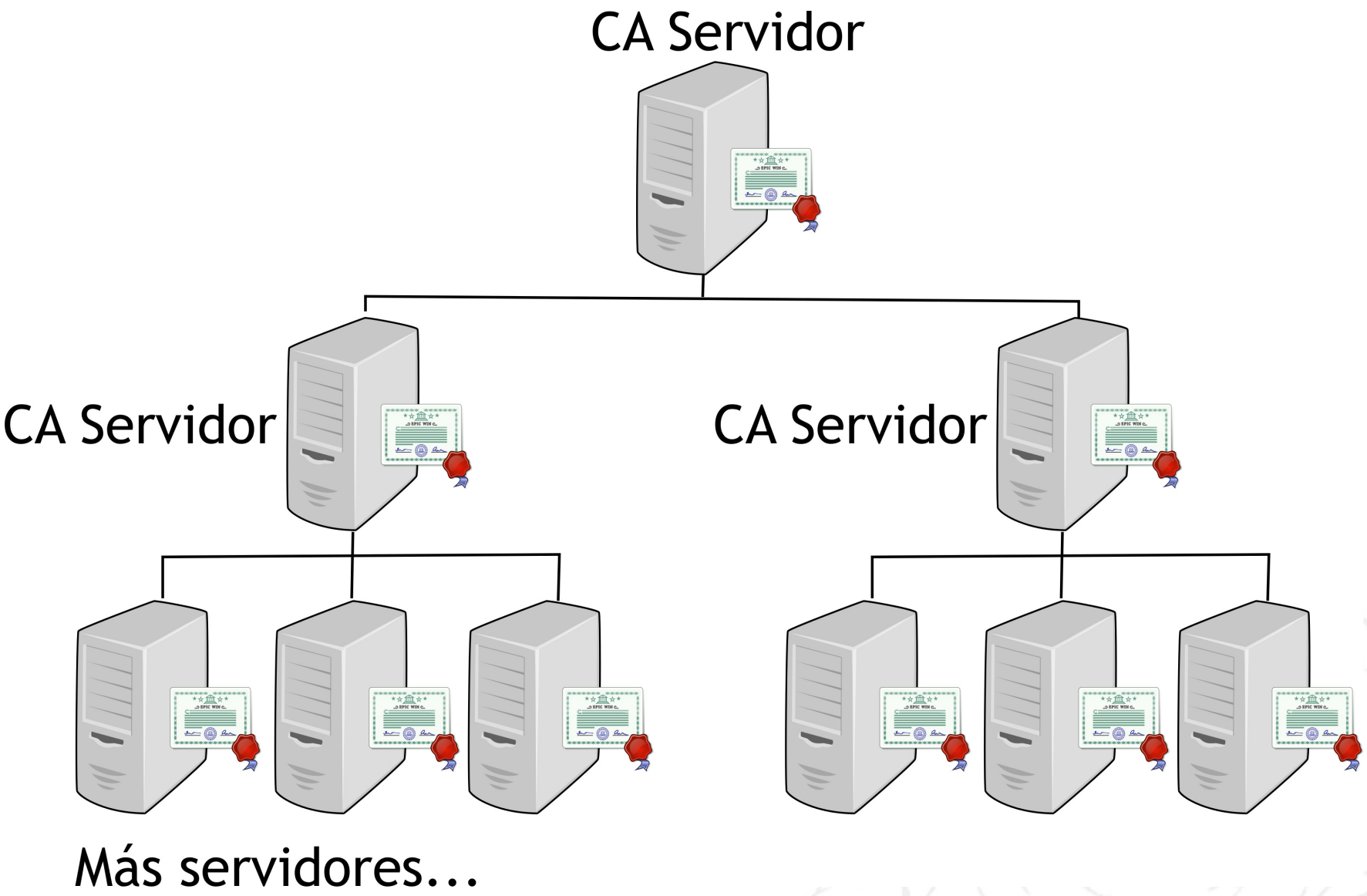
# Elementos en la PKI

- **Autoridad de Registro (AR)**. Una AR es componente opcional que puede asumir funciones administrativas de la CA.

- **Repositorio**. El repositorio es el término genérico utilizado para denotar cualquier método para almacenamiento de certificados y listas de revocación (CRLs) que permita el acceso por parte de las entidades finales a dichos documentos.

- **Emisor CRL**. El emisor CRL es un componente opcional el cual puede ser utilizado por una AC para delegar las tareas de publicación de las listas de revocación.

# Delegación de Autoridades Certificadoras



# Contenido, sección 2

Firmas y Certificados Digitales
- Firmas digitales
- Certificados digitales

SSL
- Intro
- PPF
- Ataques recientes

# SSL

SSL (Secure Sockets Layer) / TLS (Transport Layer Security)

# SSL - Definición

- SSL (Secure Sockets Layer) es el estándar de seguridad para el establecimiento de un enlace cifrado entre un servidor web y un navegador (aunque se puede utilizar para otros fines).
- Este enlace asegura que todos los datos que pasen entre el servidor y el cliente mantengan su privacidad e integridad
- Este es el estándar para la protección de transacciones comerciales en línea.

# SSL - Historia

- **Secure Network Programming API**. Los trabajos iniciales incluían la API Secure Network Programming (SNP). En 1993 exploraron el tener una capa de transporte segura que se pareciera a los sockets Berkeley, para facilitar la compatibilidad.
- **SSL 1.0**. El protocolo SSL fué originalmente desarrollado por Netscape, precursor del Navegador Firefox. Esta versión no se hizo pública.

# SSL - Historia

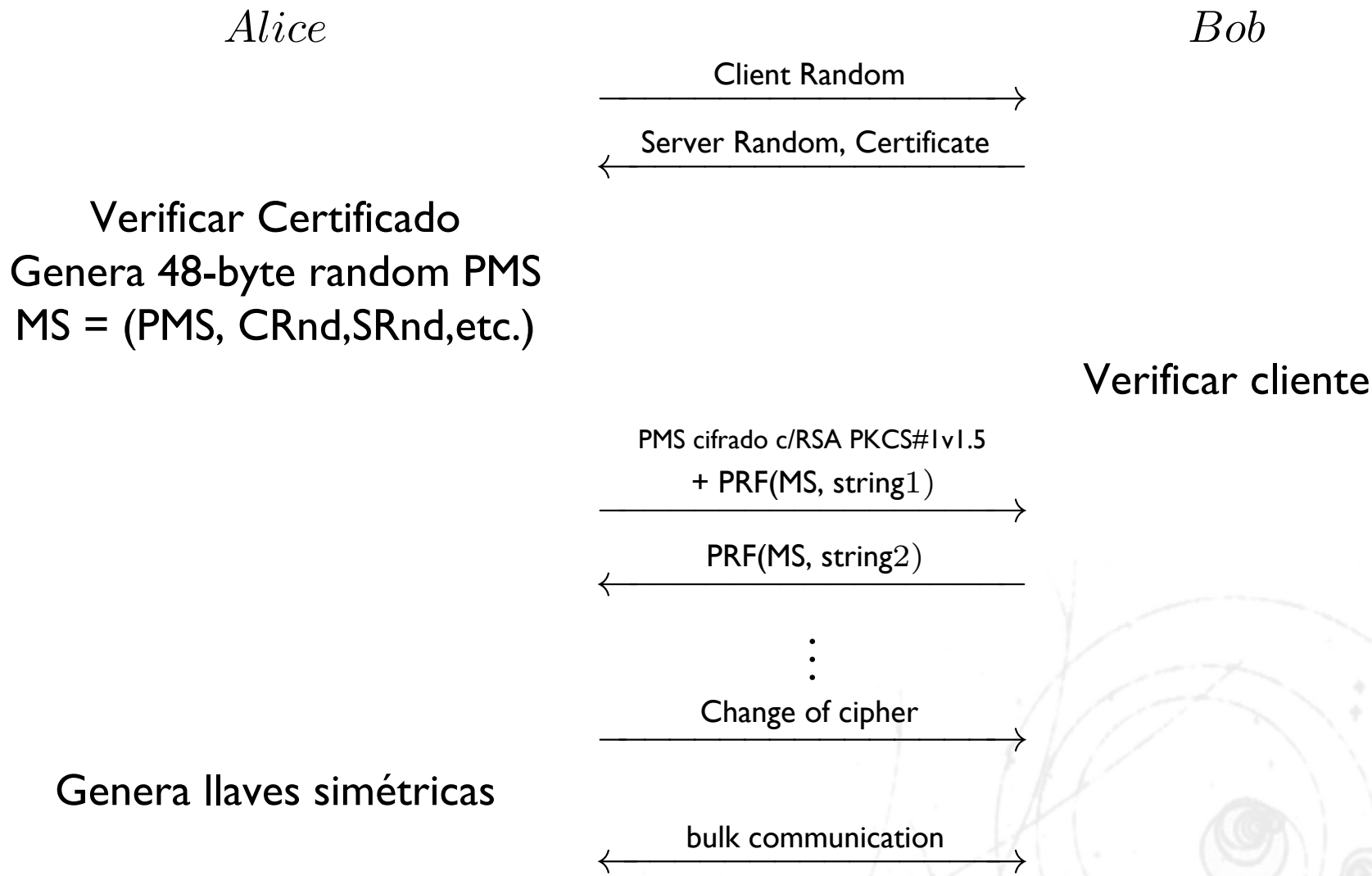
- **SSL 2.0**. Liberada en febrero de 1995, tenía un número de vulnerabilidades de software que propiciaron el desarrollo de la versión 3.0.
- **SSL 3.0** Liberada en 1996, es un rediseño por parte de Kocher, Karilton y Freier. El RFC 6101 se realizó como documento histórico.

El algoritmo básico fue escrito por Elgamal, quien trabajó para Netscape.

# SSL - Historia

- **TLS 1.0**. Se definió en el RFC 2246 en enero de 1999. Es una mejora menor a SSL 3.0, pero que afecta la compatibilidad. Contiene soporte para negociar versiones de SSL.
- **TLS 1.1** Se definió en el RFC 4346 en abril de 2006, y tiene mejoras de seguridad.
- **TLS 1.2** Fué definido en el RFC 5246 en agosto de 2008. Incluye soporte para funciones actualizadas como AES y SHA-256.

# SSL - Comunicación RSA



# SSL - Funcionamiento

- Se cambia el problema de verificar la llave pública de Bob, por verificar la llave pública de la Autoridad Certificadora.
- Aunque son varias Autoridades Certificadoras, habría mucho más servidores web que verificar.
- El problema termina siendo transparente para el usuario, ya que las Autoridades Certificadores se instalan con el Sistema Operativo, o cuando se instala un navegador. El proveedor de la aplicación hace el mantenimiento por el usuario.

# SSL - Funcionamiento

- SSL/TLS se reduce al problema de verificar las llaves públicas del otro extremo (mediante la CA)
- Se apoya el proceso con las listas de revocación (aunque se dice que en la práctica esto rara vez sucede)

¿Qué pasa si no se utilizan listas de revocación?

# Perfect Forward Secrecy

- Si alguna de las llaves utilizadas en la comunicación entre dos partes se conoce, se presenta una catástrofe, ya que se puede descifrar la comunicación futura entre las partes.

- Sin embargo, si un escucha estuvo guardando la comunicación cifrada entre las partes, dicha comunicación puede ser descifrada, y se tendría acceso a la información anterior.

# Perfect Forward Secrecy 2

- Un protocolo criptográfico tiene **perfect forward secrecy** si al comprometer llaves de uso a largo plazo (bulk communication/cifrado para almacenamiento), no se compromete la comunicación de las llaves de sesión anterior.
- La idea es que la llave utilizada para proteger la transmisión de datos no se utilice para derivar llaves adicionales. Si la llave se derivó de otro material procedente de una llave, ese mismo material no debe ser utilizado para derivar más llaves.

# Generación de llaves de sesión e intercambio

Cada conexión SSL pasa por un *SSL handshake* en donde se acuerda lo siguiente:

- Se comunican las habilidades de cada parte
- Se autentica
- Se acuerdan las *llaves de sesión* (intercambio de llaves)

La idea del intercambio de llaves es acordar las llaves de manera segura

# Generación de llaves de sesión e intercambio 2

El método más común es RSA, sin embargo, cualquiera con acceso a la llave privada del servidor, puede obtener las llaves de sesión, y descifrar la comunicación

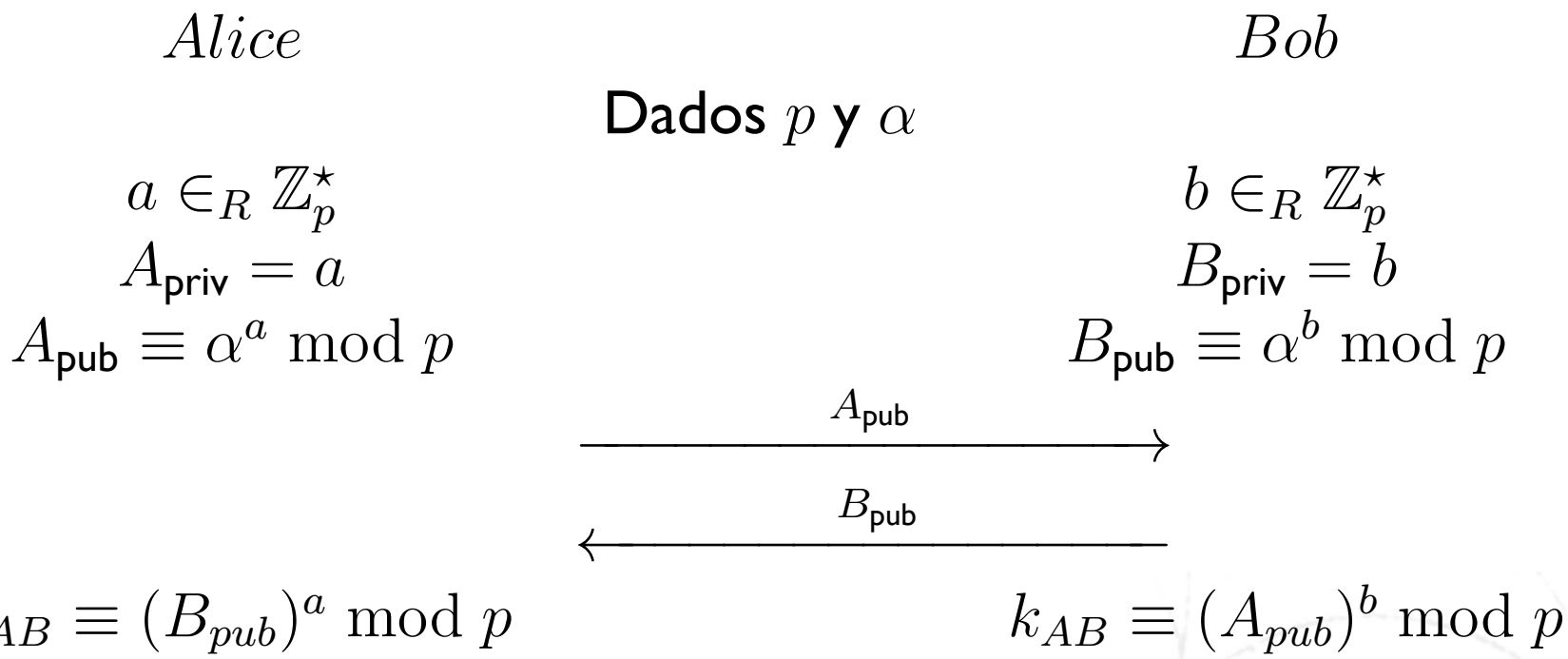
- Un dispositivo de seguridad podría descifrar la transmisión y analizarla en busca de malware
- Un adversario podría guardar la transmisión, y leerla después

# SSL y PFS

SSL soporta *forward secrecy* con el Diffie-Hellman Exchange (DHE), y con el Elliptic Curve Diffie-Hellman Exchange (ECDHE):

- DHE es muy lento, por lo que los operadores de sitios web lo deshabilitan
- ECDHE también es lento, pero más rápido que el DHE

# Diagrama DHE




Dado que siempre se utilizan valores aleatorios nuevos, a este algoritmo se le conoce como Ephemeral Diffie-Hellman (EDH, o simplemente DHE).

# Diffie-Hellman con curvas elípticas

- El DHE puede ser acelerado si utilizamos su modalidad con curvas elípticas. En lugar de utilizar el problema del logaritmo discreto sobre la exponenciación modular, se utiliza la estructura algebraica de las curvas elípticas.

- Se puede obtener el mimo nivel de seguridad de RSA con llaves mucho menores...

# Niveles de seguridad

| Familia | Criptosistema | Nivel de seguridad 128 | 192 | 256 |
|---|---|---|---|---|
| Factorización entera | RSA | 3072 bit | 7680 bit | 15360 bit |
| Logaritmo discreto | DH, DSA, Elgamal | 3072 bit | 7680 bit | 15360 bit |
| Curvas elípticas | ECDH, ECDSA | 256 bit | 384 bit | 512 bit |
| Clave simétrica | | 128 bit | 192 bit | 256 bit |

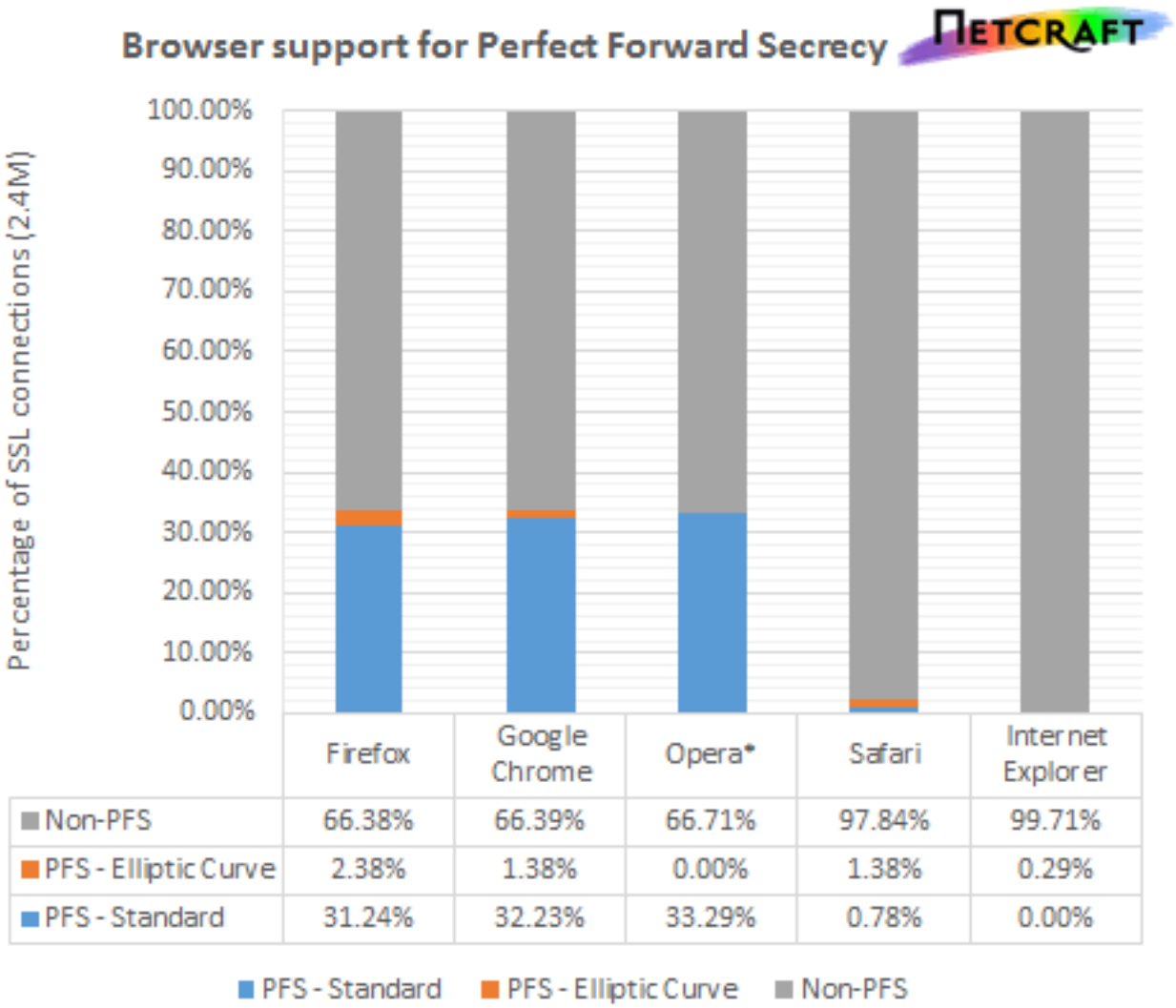
# Diffie-Hellman con curvas elípticas 2

- En lugar de definir $p$ y $\alpha$, se define una curva elíptica de la siguiente forma: $y^2 = x^3 + \alpha x + \beta$, un primo $p$, y un punto generador $G$.
- El RFC 4492 establece el uso de curvas elíptica en el TLS
- Existen unas curvas estandarizadas por el NIST: $p$-256, $p$-384, y $p$-521
- En lugar de exponenciaciones modulares, se utiliza la llamada multiplicación escalar-punto

La comunidad académica está pujando por la integración en estándares de las siguientes curvas:
`http://safecurves.cr.yp.to/`

# Uso de PFS, navegador vs. sitio web

Netcraft hizo prueba de conectividad con los 5 navegadores principales, y 2.4 millones de sitios web con SSL. El gráfico muestra qué algoritmo de conexión utilizaron (2013):




| | Firefox | Google Chrome | Opera* | Safari | Internet Explorer |
|---|---|---|---|---|---|
| Non-PFS | 66.38% | 66.39% | 66.71% | 97.84% | 99.71% |
| PFS - Elliptic Curve | 2.38% | 1.38% | 0.00% | 1.38% | 0.29% |
| PFS - Standard | 31.24% | 32.23% | 33.29% | 0.78% | 0.00% |

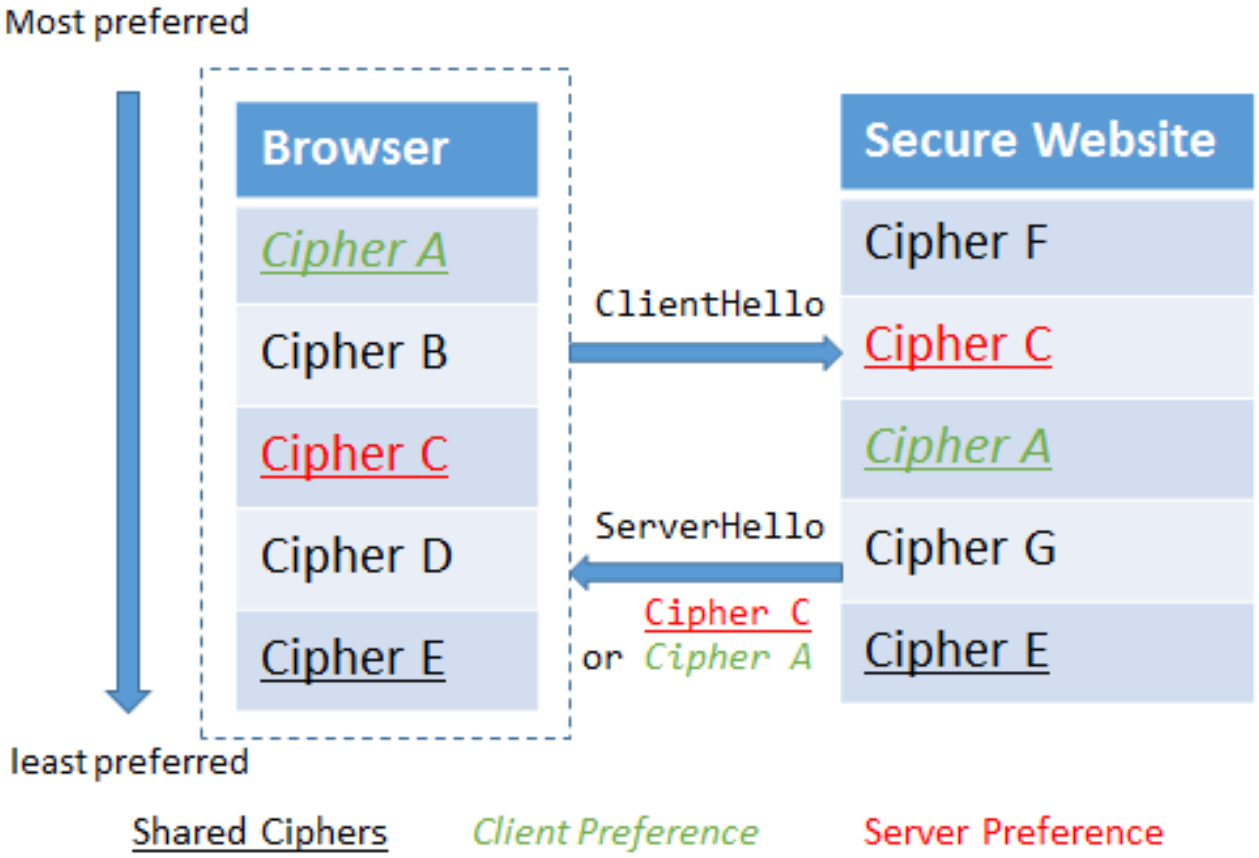
# Desglose del Internet Explorer

Netcraft presenta el uso de lo cifradores en el Internet Explorer al conectare a lo 2.4 millones de sitio web SSL más populares (2013):

| Prioridad | Método | Uso |
|---|---|---|
| 1 | AES128-SHA | 63.52% |
| 2 | AES256-SHA | 2.21% |
| 3 | RC4-SHA | 17.12% |
| 4 | DES-CBC3-SHA | 0.41% |
| 5 | ECDHE-RSA-AES128-SHA | 0.08% |
| 6 | ECDHE-RSA-AES256-SHA | 0.21% |
| 7 | ECDHE-ECDSA-AES128-SHA | 0.00% |
| 8 | ECDHE-ECDSA-AES256-SHA | 0.00% |
| 9 | DHE-DSS-AES128-SHA | 0.00% |
| 10 | DHE-DSS-AES256-SHA | 0.00% |
| 11 | EDH-DSS-DES-CBC3-SHA | 0.00% |
| 12 | RC4-MD5 | 16.46% |

# Algoritmos de conexión

Dramatización de la negociación de los algoritmos de cifrado



# Heartbleed bug

http://heartbleed.com/

- Heartbleed es una seria falla de seguridad en el OpenSSL que no requiere de cuestiones criptografía interesantes para explotarla.
- Es una debilidad que permite el robo de información protegida bajo condiciones normales
- Permite a cualquier persona en internet leer la memoria de los sitemas que son protegidos por alguna versión vulnerable de OpenSSL.

# Heartbleed bug

- Es una pequeña falla en una pieza de código que se relaciona con una implementación del mecanismo conocido como *heartbeat* (RFC 6520).
- La falla está presente en las versiones de OpenSSL 1.0.1 a la 1.0.1f
- Versiones anteriores, o más recientes no presentan esta falla; sin embargo, son las versiones más utilizadas (ya que soportan TLS 1.2).

# Heartbleed bug

Sistemas afectados:

- Servidores web con Apache
- Sistemas Linux y Apple que se conectan a red (incluyendo el iOS, y el Android)
- Sistemas con OpenSSL de la serie 1.0.1 a la 1.0.1.f
- Algunos sistemas Cisco, Firefox, Safari, Chrome

Sistemas no afectados:

- Sistemas con OpenSSL de la serie 0.9.8
- Sistemas Windows en general (incluyendo WP)
- Teléfonos BlackBerry
- Sistemas AS/400 y superior
- Internet Explorer, Opera

# Heartbleed

- Cuando un mensaje *HeartbeatRequest* se recibe y enviar un *HeartbeatResponse* no está prohibido, el receptor DEBE enviar el mensaje correspondiente de *HeartbeatResponse* conteniendo una copia exacta del paquete recibido en el message *HeartbeatRequest*;

# Heartbleed

- Cuando un mensaje *HeartbeatRequest* se recibe y enviar un *HeartbeatResponse* no está prohibido, el receptor DEBE enviar el mensaje correspondiente de *HeartbeatResponse* conteniendo una copia exacta del paquete recibido en el message *HeartbeatRequest*;

sin embargo, `http://imgs.xkcd.com/comics/heartbleed_explanation.png`

# Heartbleed

## Estructura de datos de un heartbeat:

```
1   struct {
2         HeartbeatMessageType type;
3         uint16 payload_length;
4         opaque payload[HeartbeatMessage.payload_length];
5         opaque padding[padding_length];
6     } HeartbeatMessage;
```

# Heartbleed - Código problemático

```
int dtls1_process_heartbeat(SSL *s) {
unsigned char *p = &s->s3->rrec.data[0], *pl;
unsigned short hbtype;
unsigned int payload;
unsigned int padding = 16; /* Use minimum padding */

/* Read type and payload length first */
hbtype = *p++; n2s(p, payload); pl = p;
if (s->msg_callback)
s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT, &s->s3->rrec.data[0], s->s3->rrec.
      length,s, s->msg_callback_arg);

if (hbtype == TLS1_HB_REQUEST) {
unsigned char *buffer, *bp; int r;

/* Allocate memory for the response, size is 1 bytes
* message type, plus 2 bytes payload length, plus
* payload, plus padding */
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;

/* Enter response type, length and copy payload */
*bp++ = TLS1_HB_RESPONSE; s2n(payload, bp);
memcpy(bp, pl, payload); bp += payload;
/* Random padding */
RAND_pseudo_bytes(bp, padding);
r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
if (r >= 0 && s->msg_callback)
        s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding,
             s, s->msg_callback_arg);
```

# Heartbleed

## Ajuste:

```
1 /* Read type and payload length first */
2 if (1 + 2 + 16 > s->s3->rrec.length)
3 return 0; /* silently discard */
4 hbtype = *p++;
5 n2s(p, payload);
6 if (1 + 2 + payload + 16 > s->s3->rrec.length)
7 return 0; /* silently discard per RFC 6520 sec. 4 */
8 pl = p;
```

# Heartbleed

## Ajuste:

```
1 /* Read type and payload length first */
2 if (1 + 2 + 16 > s->s3->rrec.length)
3 return 0; /* silently discard */
4 hbtype = *p++;
5 n2s(p, payload);
6 if (1 + 2 + payload + 16 > s->s3->rrec.length)
7 return 0; /* silently discard per RFC 6520 sec. 4 */
8 pl = p;
```

## Test code: https://gist.github.com/sh1n0b1/10100394

# Bonus

- Report a medium to critical bug in OpenSSL, or similarly widely used security-related software.

# Heartbleed "2.0"

- Recientemente se encontró otra vulnerabilidad en el OpenSSL
- En este caso, al negociar el método de cifrado, se puede deshabilitar el mismo, quedando la transmisión en texto en claro.
- Versiones afectadas:
  - OpenSSL 1.0.1 a 1.0.1g
  - OpenSSL 1.0.0 a 1.0.0l
  - OpenSSL anteriores a 0.9.8y
- Versiones no afectadas:
  - OpenSSL 1.0.1h
  - OpenSSL 1.0.0m
  - OpenSSL 0.9.8za

# Dual_EC_DRBG

- **Dual_EC_DRBG** es un generador de números pseudo-aleatorios promovido por el NIST en el *NIST SP 800-90A*, y creado por la NSA.
- Este algoritmo es problemático porque se hizo mandatorio en la norma FIPS (Federal Information Processing Standards).
- Actualmente, la norma FIPS-140-2 ya no lo hace obligatorio, pero se sigue utilizando

# Dual_EC_DRBG

- En el 2007, Dan Shumow y Niels Ferguson de Microsoft mostraron que el algoritmo Dual_EC_DRBG podría tener una *backdoor* `http://rump2007.cr.yp.to/15-shumow.pdf`

- En el 2004, una patente daba una variante para evitar un posible backdoor `http://www.freshpatents.com/Elliptic-curve-random-number-generation-dt20070816pta.php`

# Dual_EC_DRBG - Funcionamiento

- Este PRNG funciona como sigue:
  - Toma una semilla
  - La introduce en una función hash
  - Se utiliza para los puntos $P$ y $Q$ en una curva elíptica
  - Se hace una transformación final

# Dual_EC_DRBG - Backdoor

- El ataque consiste en la relación de algunos bits de la secuencia, dejando hasta 17 bits ambigüos para una búsqueda a fuerza bruta de un punto dado.
- Dicho punto nos puede dar un valor en la secuencia de salida (el aleatorio)
- Finalmente, se pueden calcular el resto de los valores a partir de este valor.

`http://blog.0xbadc0de.be/archives/155`

# NSA take on SSL

- Las agencias de gobierno van a la cabeza en seguridad y criptografía; sin embargo, cada año la distancia entre las agencias y la academia/empresas varía.
- En fechas recientes, la NSA ha lanzado una campaña contra el cifrado, o mas bien, se han descubierto una parte de sus estrategias de espionaje en la era digital.
- La NSA en particular busca garantizar sus objetivos de seguridad, aunque pareciera que por lo general terminan violentando alguna que otra ley de un tercer país.

# NSA take on SSL

- Algunos documentos filtrados en internet muestran que la NSA ha estado teniendo una especie de *guerra* en contra del cifrado, y de la seguridad informática
- La NSA ha trabajado secretamente en la elaboración de estándares de cifrado, con la finalidad de hacerlos más débiles. En algunos casos de manera inocente, en otros a través de un tercero, con resultados mixtos.
- En otros casos, ha promovido la estandarización de métodos de cifrado débiles, o que es sabido que pueden tronar.

# NSA take on SSL

- http://www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html?_r=1&