“Understanding tools for a more secure internet” 2nd cyber-security week @ CIC-IPN.

Luis J. Dominguez Perez
CONACyT. CIMAT

luis.dominguez@cimat.mx

CONACYT

Centro de Investigación en Computación
Instituto Politécnico Nacional

CIMAT

Public Key Cryptography

- In 1976, Whitfield Diffie, and Martin Hellman published their famous article: "New Directions in Cryptography"

- A bit before, Ralph Merkle invented a public key construction for his lectures: "Secure communication over insecure channels" in 1982

Public Key Cryptography

- In 1976, Whitfield Diffie, and Martin Hellman published their famous article: "New Directions in Cryptography"

- A bit before, Ralph Merkle invented a public key construction for his lectures: "Secure communication over insecure channels" in 1982

- Originally, the concepts were discovered by James Ellis; however, these were kept secret as the were classified information by the GCHQ from 1969 to 1997.

- Additionally, Malcolm Williamson, and Clifford Cocks from the GCHQ, also discovered the Diffie-Hellman key exchange, and the RSA encryption

- Before “New Directions. . .”, all of the research on cryptography in the USA was under the National Security Agency (NSA)

- Up to mid 1990’s, exporting cryptographic algorithms was considered treason (in the USA).

- Before "New Directions...", all of the research on cryptography in the USA was under the National Security Agency (NSA)

- Up to mid 1990's, exporting cryptographic algorithms was considered treason (in the USA).

- Afterwards, the prohibition was only kept for high security algorithms if they were readable by machines (source code, binary executable file, etc.).

- Before "New Directions...", all of the research on cryptography in the USA was under the National Security Agency (NSA)

- Up to mid 1990's, exporting cryptographic algorithms was considered treason (in the USA).

- Afterwards, the prohibition was only kept for high security algorithms if they were readable by machines (source code, binary executable file, etc.).

RSA

Designed in 1977 by Ron Rivest, Adi Shamir, and Leonar Adleman.

- Let p y q be two different random large prime numbers
- The modulus n is the product of p, and q
- The function $\Phi(n) = (p-1)(q-1)$
- Choose $1 < e < \Phi(n)$, such that $\text{GCD}(e, \Phi(n)) = 1$; $e = 2^{16} + 1$ typically
- Compute $d \equiv e^{-1} \bmod \Phi(n)$

The public key is (e, n). The private key is (d, p, q).

RSA encryption, and decryption

Given a message $M < n$

- **Encryption**. $C = M^e \bmod n$
- **Decryption**. $M = C^d \bmod n$

Example

- $p = 11,\ q = 13$
- $n = p \cdot q = 11 \cdot 13 = 143$
- $\Phi(n) = (p-1)(q-1) = 10 \cdot 12 = 120$
- $\text{GCD}(e, \Phi(n)) = \text{GCD}(e, 120) = 1;\ e = 17$
- $d = e^{-1} \bmod \Phi(n) = 17^{-1} \bmod 120 = 113$

- Public Key $= (e, n) = (17, 143)$
- Private Key $= (d, p, q) = (113, 11, 23)$

... example

- Message $M = 50$
- **Encryption:**
  $C = M^e \bmod n = 50^{17} \bmod 143 = 85$
- **Decryption:**
  $M = C^d \bmod n = 85^{113} \bmod 143 = 50$

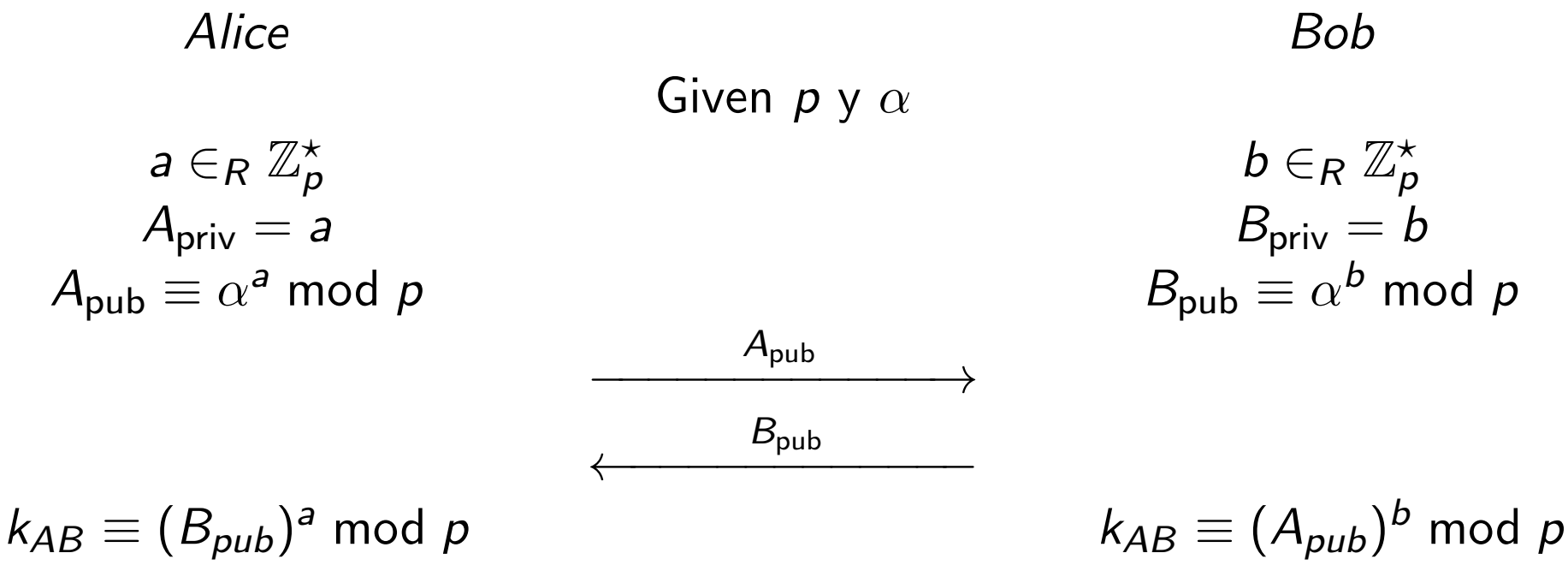
seems easy; however, observe the 85^{113}, what would happen with very large numbers?

DiffieHellman Key Exchange
(DHKE)

- The basic idea behind DHKE is that the exponentiation in $\mathbb{Z}_p^\star$, a p-prime, is a one-way function, and the exponentiation is commutative:

$$x \equiv (\alpha^x)^y \equiv (\alpha^y)^x \bmod p$$

DHKE Diagram

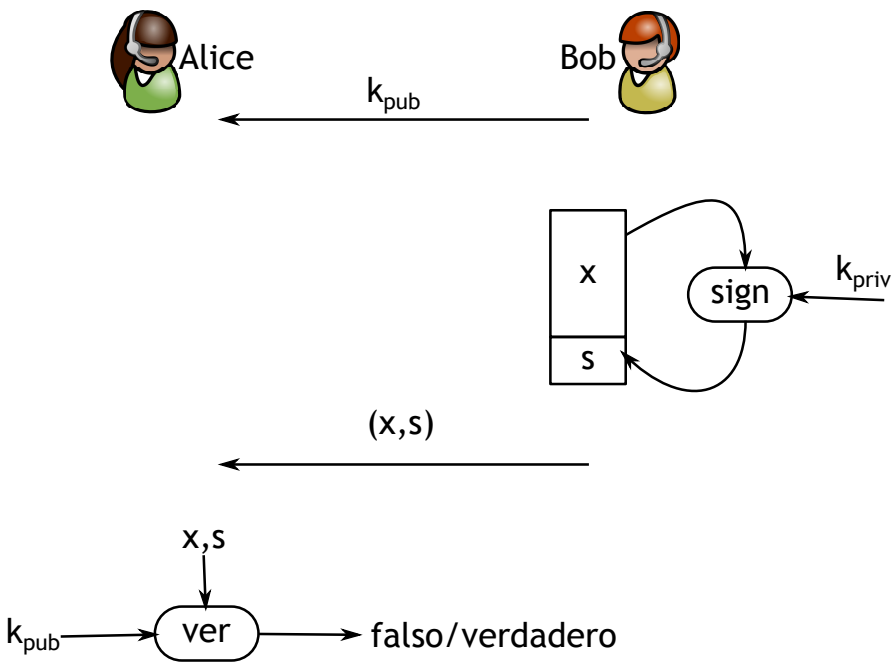


Digital Signatures

- Demonstrating that certain person generated a message is critical so some applications.
- In the "analog" world, we use hand-written signatures (in some countries any way).
- Only the person who created the signature can reproduce it.

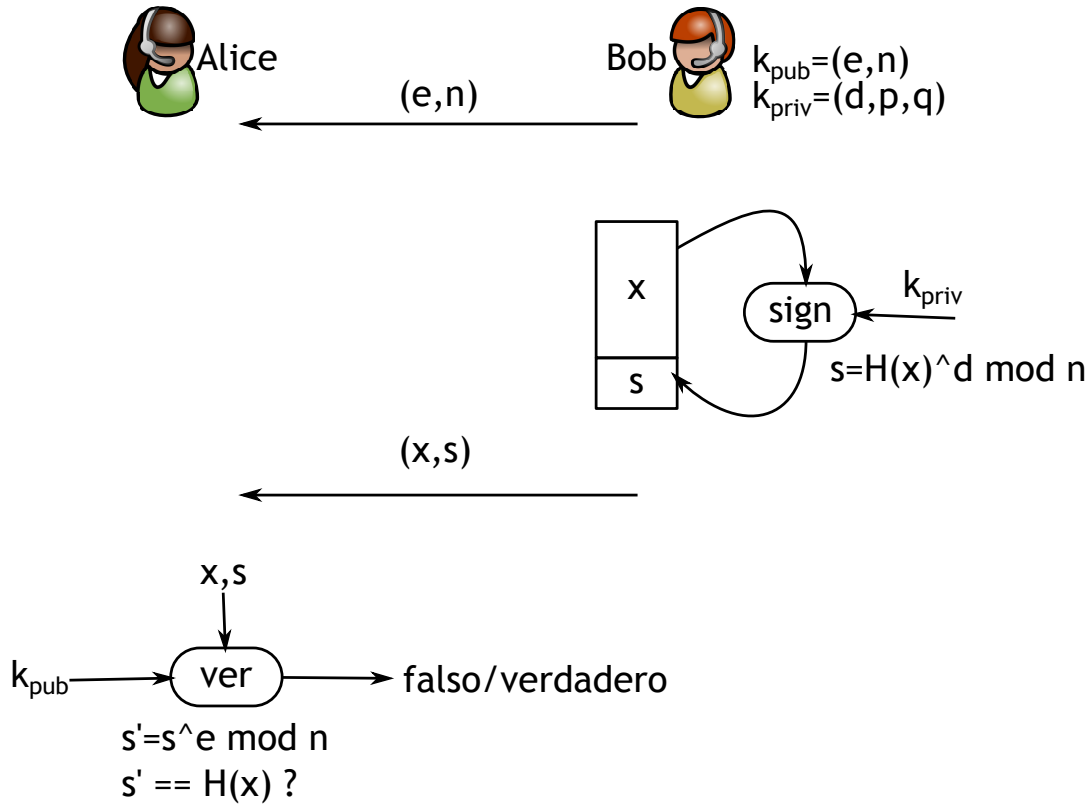
Diagram

In the digital world this is possible by using public key cryptography. The signatory signs with her private key, and addressee uses the corresponding public key to verify.



General diagram for RSA signature

Basic RSA signature



ElGamal

- The Elgamal encryption was proposed by Taher Elgamal in 1985.
- It can be seen as an extension of the Diffie-Hellman Key Exchange (DHKE)

Elgamal signature

- Key generation:
  - Prime number generation p
  - Find an element $\alpha \in \mathbb{Z}_p^\star$
  - Choose a random element d, with $2 < d < p-2$
  - Compute $\beta = \alpha^d \bmod p$

- Message signing:
  - Given a message M
  - Choose an ephemeral key k_E, with $0 < k_E < p-2$, with $\mathrm{GCD}(k_E, p-1) = 1$
  - Compute $r \equiv a^{k_E} \bmod p$
  - Compute $s \equiv (M - d \cdot r) k_E^{-1} \bmod p-1$

Elgamal signature

- Key generation:
  - Prime number generation p
  - Find an element $\alpha \in \mathbb{Z}_p^\star$
  - Choose a random element d, with $2 < d < p-2$
  - Compute $\beta = \alpha^d \bmod p$

- Message signing:
  - Given a message M
  - Choose an ephemeral key k_E, with $0 < k_E < p-2$, with $\mathrm{GCD}(k_E, p-1) = 1$
  - Compute $r \equiv a^{k_E} \bmod p$
  - Compute $s \equiv (M - d \cdot r) k_E^{-1} \bmod p-1$

- The signature of M is (r, s)

Elgamal signature

- Key generation:
  - Prime number generation p
  - Find an element $\alpha \in \mathbb{Z}_p^\star$
  - Choose a random element d, with $2 < d < p-2$
  - Compute $\beta = \alpha^d \bmod p$

- Message signing:
  - Given a message M
  - Choose an ephemeral key k_E, with $0 < k_E < p-2$, with $\mathrm{GCD}(k_E, p-1) = 1$
  - Compute $r \equiv a^{k_E} \bmod p$
  - Compute $s \equiv (M - d \cdot r) k_E^{-1} \bmod p-1$

- The signature of M is (r, s)

Elgamal Signature Verification

- Signature Verification:
  - Compute $t \equiv \beta^r \cdot r^s \bmod p$

- If $t \equiv \alpha^x \bmod pq$, the signature verifies.

Example, sign M

$p = 29,\ \alpha = 2$

$d = 12$

$\beta = \alpha^d \equiv 7$

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

$k_E = 5$

$(5, 28) = 1$

$x = 26$

$r = 2^5 \equiv 3$

$s = -10 \cdot 7 \equiv 26$

$(26,(3,26))$ ←

$t = 7^3 \cdot 3^{26} \equiv 22$

$\alpha^x \equiv 2^{26} \equiv 22$

$t \equiv \alpha^x \Rightarrow \text{OK}$

DSA Signature

The standard signature DSA has the following steps:

- Key Generation:
  - Find Prime number p, with $2^{1023} < p < 2^{1024}$
  - Find a prime number q: $2^{159} < q < 2^{160}$
  - Find an element α, of order q
  - Choose a random number d, with $1 < d < q$
  - Compute $\beta = \alpha^d \bmod p$

- The key are:
  - Public: (p, q, α, β)
  - Private: d

DSA signature of a message

- Message signature:
  - Given a message M
  - Choose an ephemeral key k_E, with $0 < k_E < q$
  - Compute $r \equiv (a^{k_E} \bmod p) \bmod q$
  - Compute $s \equiv (SHA(M) + d \cdot r)k_E^{-1} \bmod q$

- The signature of M is (r, s)

DSA signature verification

- Signature verification:
  - Compute $w \equiv s^{-1} \bmod q$
  - Compute $u_1 \equiv w \cdot SHA(M) \bmod q$
  - Compute $u_2 \equiv w \cdot r \bmod q$
  - Compute $v \equiv (\alpha^{u_1} \cdot \beta^{u_2} \bmod p) \bmod q$

- If $v \equiv r \bmod q$, the signature verifies

Example, sign message M

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

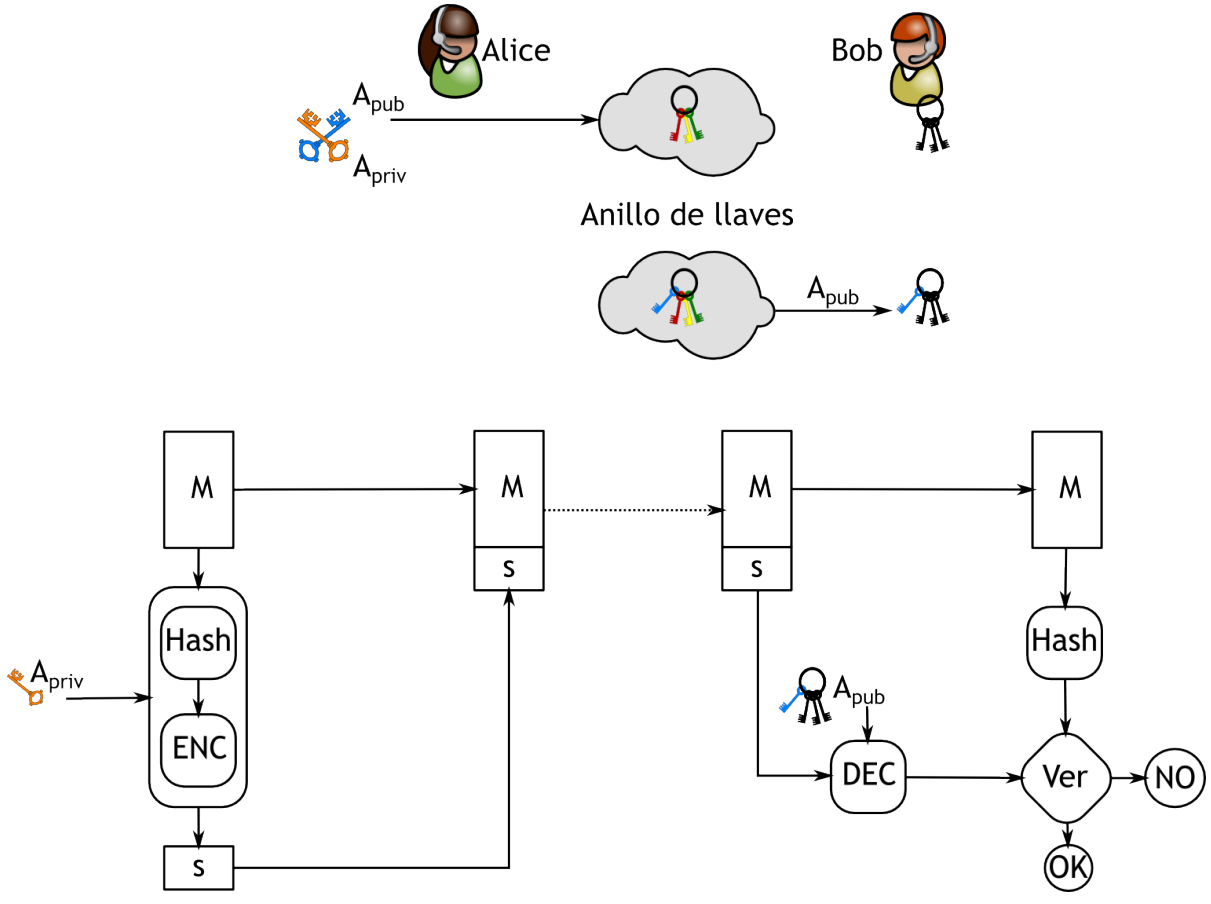
$k_{\text{pub}}(p,q,\alpha,\beta)=(59,29,3,4)$ ⟵

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

$(M,(r,s))$ ⟵

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

Digital Signature



Digital certificate

Is a document in which the digital signature of a trustworthy entity, whose public key is previously stored, associates the public key to a given entity: name, organization, address, email, RFC, CURP, etc.

The certificate serves to warranty that a given public key belongs to the owner of its corresponding private key.

These certificate are granted by a trustworhty entity, a Certificate Authority.. perhaps, in practice, we delegate who to trust to Mozilla, Microsoft, Apple, BlackBerry.

Digital certificate

Is a document in which the digital signature of a trustworthy entity, whose public key is previously stored, associates the public key to a given entity: name, organization, address, email, RFC, CURP, etc.

The certificate serves to warranty that a given public key belongs to the owner of its corresponding private key.

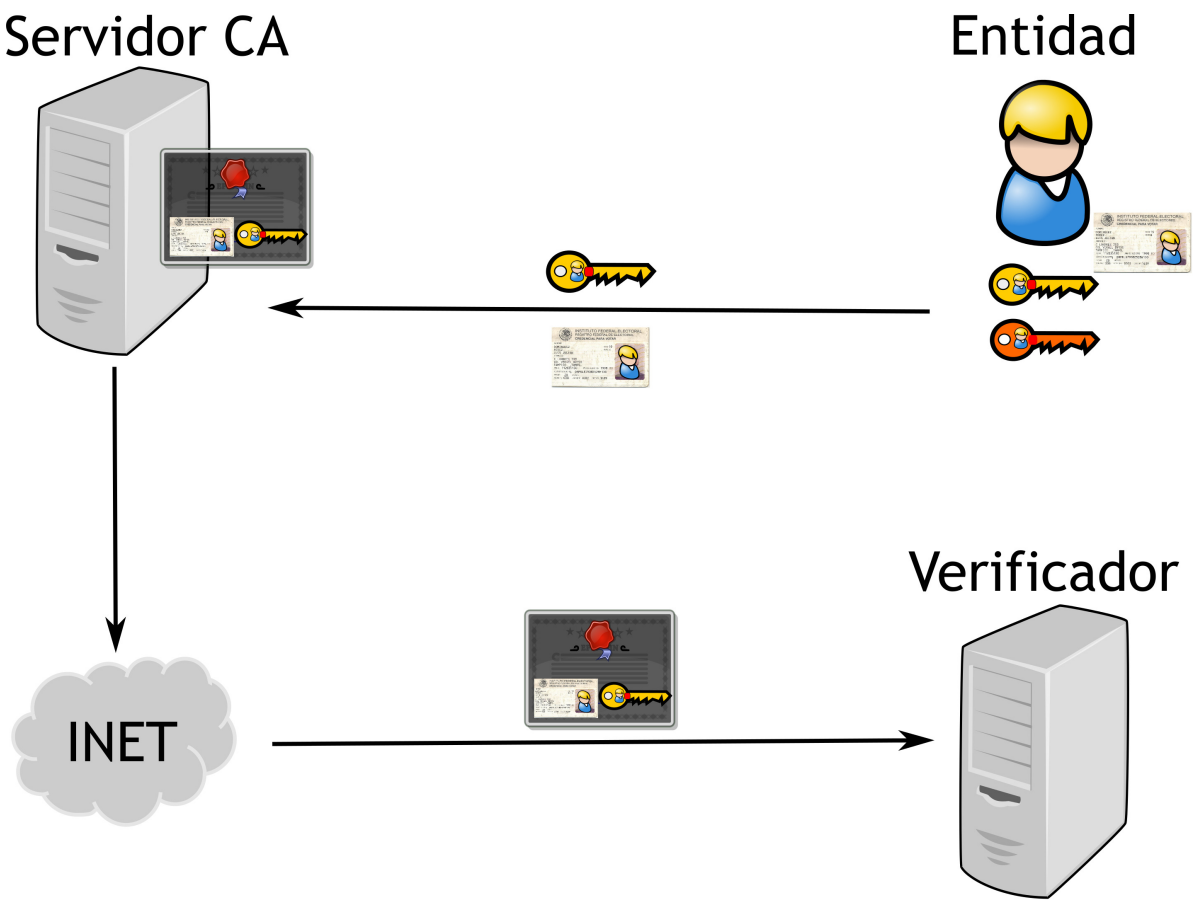
These certificate are granted by a trustworhty entity, a Certificate Authority.. perhaps, in practice, we delegate who to trust to Mozilla, Microsoft, Apple, BlackBerry.

Responsabilities of a CA

The basic responsabilities are:

- Key generation (Secure exchange)
- Certificate Emission
- CRL publication

Certificates



Contents of a certificate

The X.509 certificate sets the ASN1 format for the digital certificates, which contain::

- Serial number (which is no longer consecutive)
- Subject: Person, or entity to identify
- Digital Signature Algorithm
- Digital Signature
- Emitter
- Range of dates of validity
- Public Key allowed usage: encription, signature, certificate emission
- Public Key
- Hashing algorithm
- Hash

Example

```
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 Ass
        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
        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:

    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.m
         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/DigiCertHighAssura

    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
```

Hands-on

Certificate creation

- DSA parameter generation

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

- Key generation

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

- Self-signed certificate generation

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

- Review the certificate

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

Cliente side hands-on

- Generate a certificate for the client

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

- Certificate emission

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

- Revieweing the certificate

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

- Certificate Verification

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

Apache configuration

- Copy the certificates files to the server
- Find the apache config file
- Identify the “VirtuaHost” block to configure

```
<VirtualHost 192.168.0.1:443>
DocumentRoot /var/www/html2
ServerName www.yourdomain.com
SSLEngine on
SSLCertificateFile /path/to/your_domain_name.crt
SSLCertificateKeyFile /path/to/your_private.key
SSLCertificateChainFile /path/to/DigiCertCA.crt
</VirtualHost>
```

Apache configuration

- Test your apache configuration
  - apachectl configtest
- Restart your apache server
  - apachectl stop
  - apachectl start

ngix configuration

- You need the CA's certificate
- Copy the certificates files to the server
- Concatenate the primary certificate and intermediate certificate
  - cat your_domain_name.crt rootcert.pem ¿¿ bundle.crt
- Edit ngix configuration file:

ngix configuration

```
server {
listen   443;
ssl    on;
ssl_certificate    /etc/ssl/your_domain_name.pem; (or bundle.c
ssl_certificate_key    /etc/ssl/your_domain_name.key;
server_name your.domain.com;
access_log /var/log/nginx/nginx.vhost.access.log;
error_log /var/log/nginx/nginx.vhost.error.log;
location / {
root   /home/www/public_html/your.domain.com/public/;
index  index.html;
}
}
```

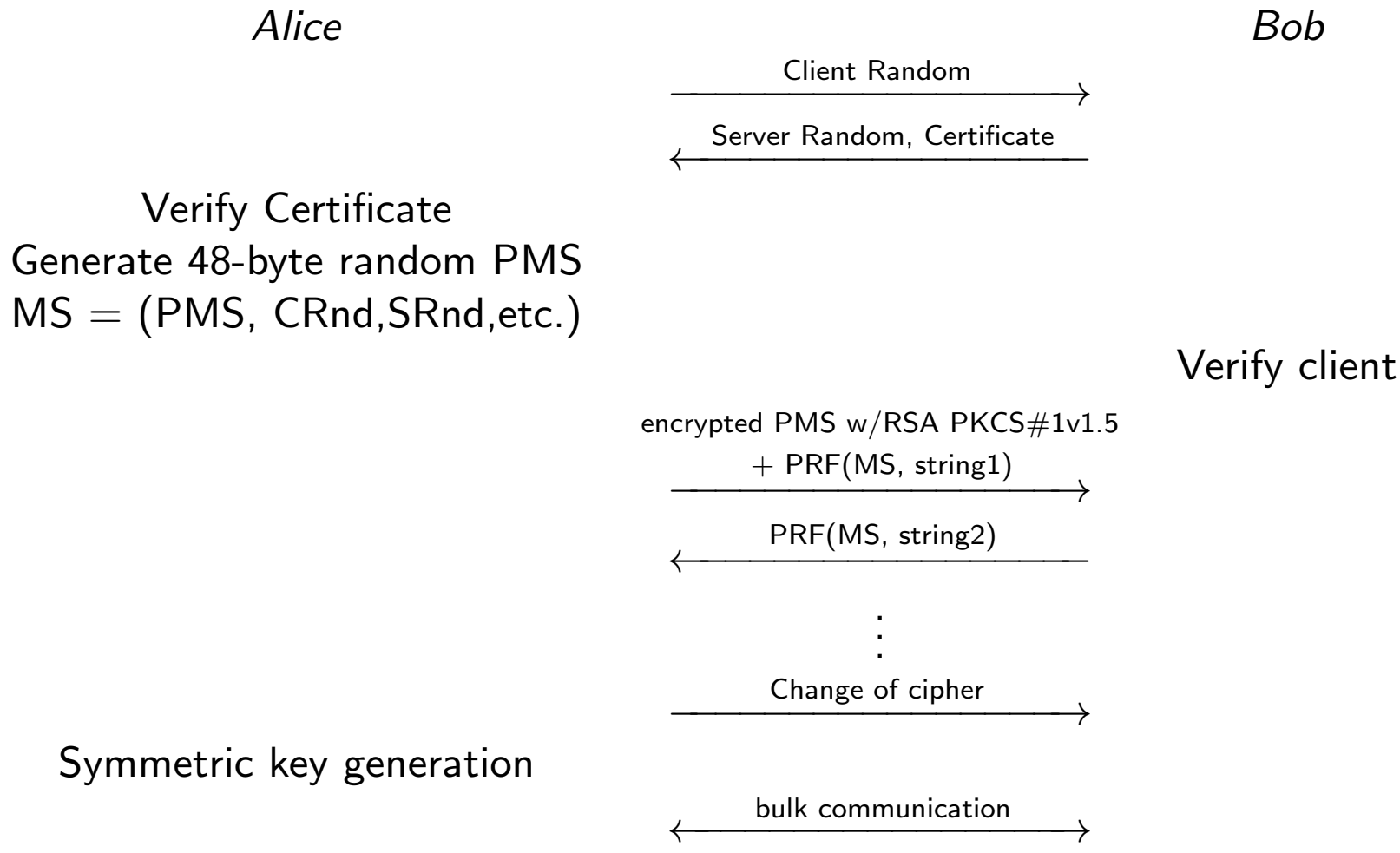
- Restart the ngix server
  - /etc/init.d/nginx restart

IIS configuration

SSL - Definition

- SSL (Secure Sockets Layer) is the security standard to establish an encrypted link between a web server, and an internet browser (perhaps, the protocol could be used for something else).
- This links ensures the data travels between the server, and the client, and that it mantains its privacy, and integrity
- This is the standard for online transactions.

SSL - RSA communication



SSL - overview

- We exchange the problem of verifying the public key of Bob (and everybody else), by the one of verifying the public of a Certificate Authority.

- Despite the are a lot of Certificate Authorities, the number of webserver is substantially larger...

SSL - overview

- We exchange the problem of verifying the public key of Bob (and everybody else), by the one of verifying the public of a Certificate Authority.
- Despite the are a lot of Certificate Authorities, the number of webserver is substantially larger...
- The solution to this problem is transparent to the user, since the Public Keys of the Certificate Authorities are installed together with the Operating System, or when, for example, we install an internet browser. The Application provider does this for the user.

SSL - overview

- We exchange the problem of verifying the public key of Bob (and everybody else), by the one of verifying the public of a Certificate Authority.

- Despite the are a lot of Certificate Authorities, the number of webserver is substantially larger...

- The solution to this problem is transparent to the user, since the Public Keys of the Certificate Authorities are installed together with the Operating System, or when, for example, we install an internet browser. The Application provider does this for the user.

SSL - overview

- SSL/TLS is reduced to the problem of verifying the public keys of the other end-point (using the CA)

- It makes use of Revocation Lists to ensure no one is using a no longer valid certificate (perhaps, until recentrly, this was rarely done)

Email with PGP

There are two schemes to protect email communication:

- Secure/Multipurpose Internet Mail Extension (S/MIME)
- Pretty Good Privacy (PGP)

S/MIME from RSA will emerge as the industry standard for commercial usage

PGP is also on the standardization track, but will go for personal usage. I will talk about PGP

Email with PGP

There are two schemes to protect email communication:

- Secure/Multipurpose Internet Mail Extension (S/MIME)
- Pretty Good Privacy (PGP)

S/MIME from RSA will emerge as the industry standard for commercial usage

PGP is also on the standardization track, but will go for personal usage. I will talk about PGP

PGP

PGP consists of the following services:

- Authentication
- Confidentiality
- E-mail compatibility
- Segmentation

In a nutshell:

- Generates a session key, and encrypt it
- Signs the message
- Compress the message
- Encrypts the message
- (prepend the encrypted key to the message)

PGP

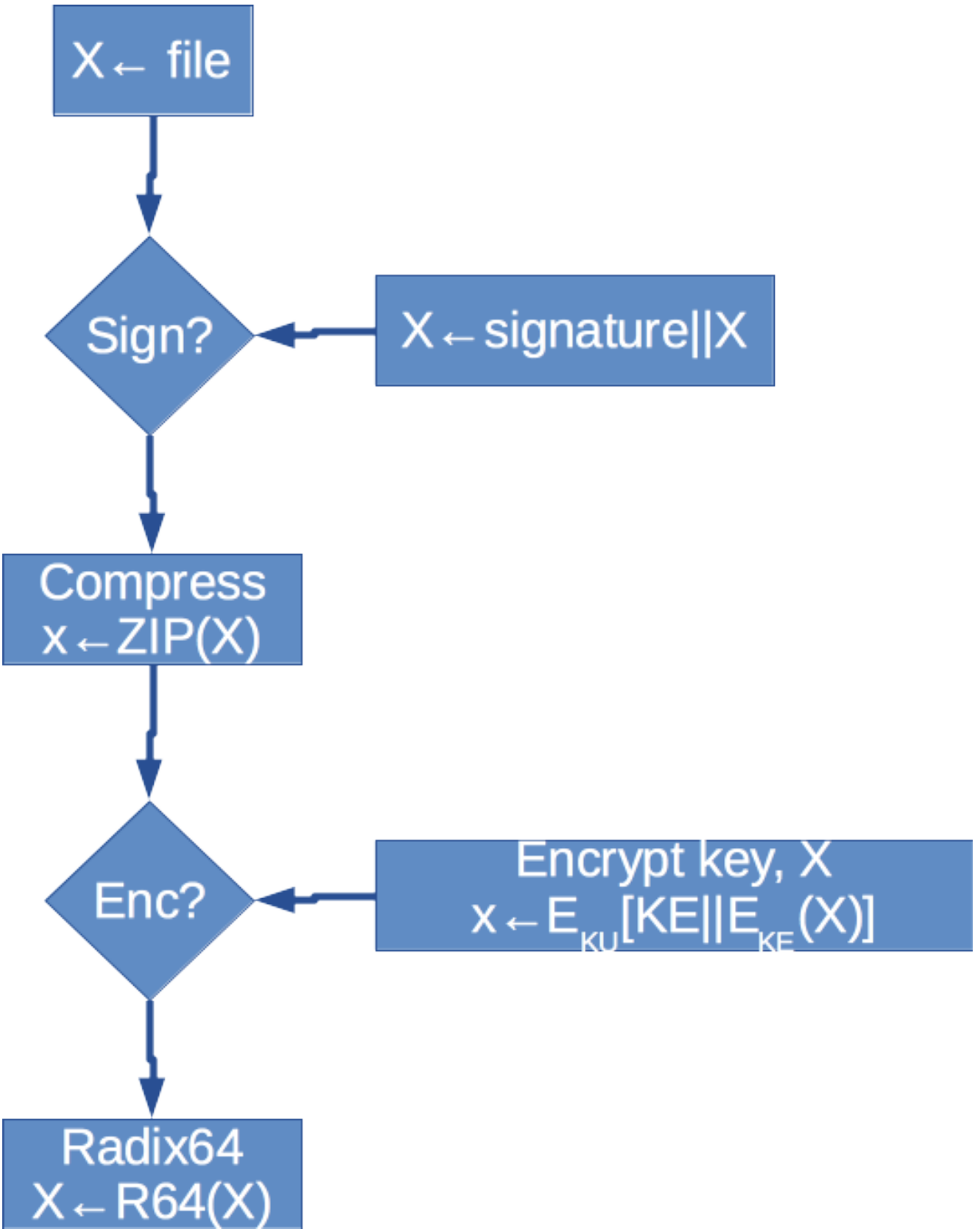
PGP consists of the following services:

- Authentication
- Confidentiality
- E-mail compatibility
- Segmentation

In a nutshell:

- Generates a session key, and encrypt it
- Signs the message
- Compress the message
- Encrypts the message
- (prepend the encrypted key to the message)

PGP Diagram


X ← file
Sign?
X ← signature||X
Compress
x ← ZIP(X)
Enc?
Encrypt key, X
x ← E_KU[KE||E_KE(X)]
Radix64
X ← R64(X)

Algoritmhs used

- Digital Signature (DSS/SHA or RSA/SHA)
  Uses SHA-1(!) for the message, and uses either DSS, or RSA using sender's private key
- Encryption (CAST, IDEA, or Triple DES, with Diffie-Hellman, or RSA)
  The message is encrypted with CAST, IDEA, or Triple DES, with a session key using Diffie-Hellman, or RSA with the public key of the recipient

Algoritmhs used

- Digital Signature (DSS/SHA or RSA/SHA)
  Uses SHA-1(!) for the message, and uses either DSS, or RSA using sender's private key
- Encryption (CAST, IDEA, or Triple DES, with Diffie-Hellman, or RSA)
  The message is encrypted with CAST, IDEA, or Triple DES, with a session key using Diffie-Hellman, or RSA with the public key of the recipient
- Compression (ZIP)
  For transmission purposes, and for removing some of the statistic properties of the message (before encryption)

Algoritmhs used

- Digital Signature (DSS/SHA or RSA/SHA)
  Uses SHA-1(!) for the message, and uses either DSS, or RSA using sender's private key
- Encryption (CAST, IDEA, or Triple DES, with Diffie-Hellman, or RSA)
  The message is encrypted with CAST, IDEA, or Triple DES, with a session key using Diffie-Hellman, or RSA with the public key of the recipient
- Compression (ZIP)
  For transmission purposes, and for removing some of the statistic properties of the message (before encryption)
- Email compatibility (Radix 64 conversion)
  For compatibility purposes

Algoritmhs used

- Digital Signature (DSS/SHA or RSA/SHA)
  Uses SHA-1(!) for the message, and uses either DSS, or RSA using sender's private key
- Encryption (CAST, IDEA, or Triple DES, with Diffie-Hellman, or RSA)
  The message is encrypted with CAST, IDEA, or Triple DES, with a session key using Diffie-Hellman, or RSA with the public key of the recipient
- Compression (ZIP)
  For transmission purposes, and for removing some of the statistic properties of the message (before encryption)
- Email compatibility (Radix 64 conversion)
  For compatibility purposes
- Segmentation
  Depending on the application, it may break the message as needed

Algoritmhs used

- Digital Signature (DSS/SHA or RSA/SHA)
  Uses SHA-1(!) for the message, and uses either DSS, or RSA using sender's private key
- Encryption (CAST, IDEA, or Triple DES, with Diffie-Hellman, or RSA)
  The message is encrypted with CAST, IDEA, or Triple DES, with a session key using Diffie-Hellman, or RSA with the public key of the recipient
- Compression (ZIP)
  For transmission purposes, and for removing some of the statistic properties of the message (before encryption)
- Email compatibility (Radix 64 conversion)
  For compatibility purposes
- Segmentation
  Depending on the application, it may break the message as needed

Usage

- Create Key
  - gpg –gen-key
  - gpg –armor –output pubkey.txt –export 'Your Name'
  - gpg –send-keys 'Your Name' –keyserver hkp://subkeys.pgp.net
- Encrypting / Decrypting
  - gpg –encrypt –recipient 'Your Name' foo.txt
  - gpg –output foo.txt –decrypt foo.txt.gpg
- Encrypting for Recipient
  - gpg –search-keys 'user1@example.org' –keyserver hkp://subkeys.pgp.net
  - gpg –import key.asc
  - gpg –list-keys
  - gpg –encrypt –recipient 'user1@example.org' foo.txt

Usage

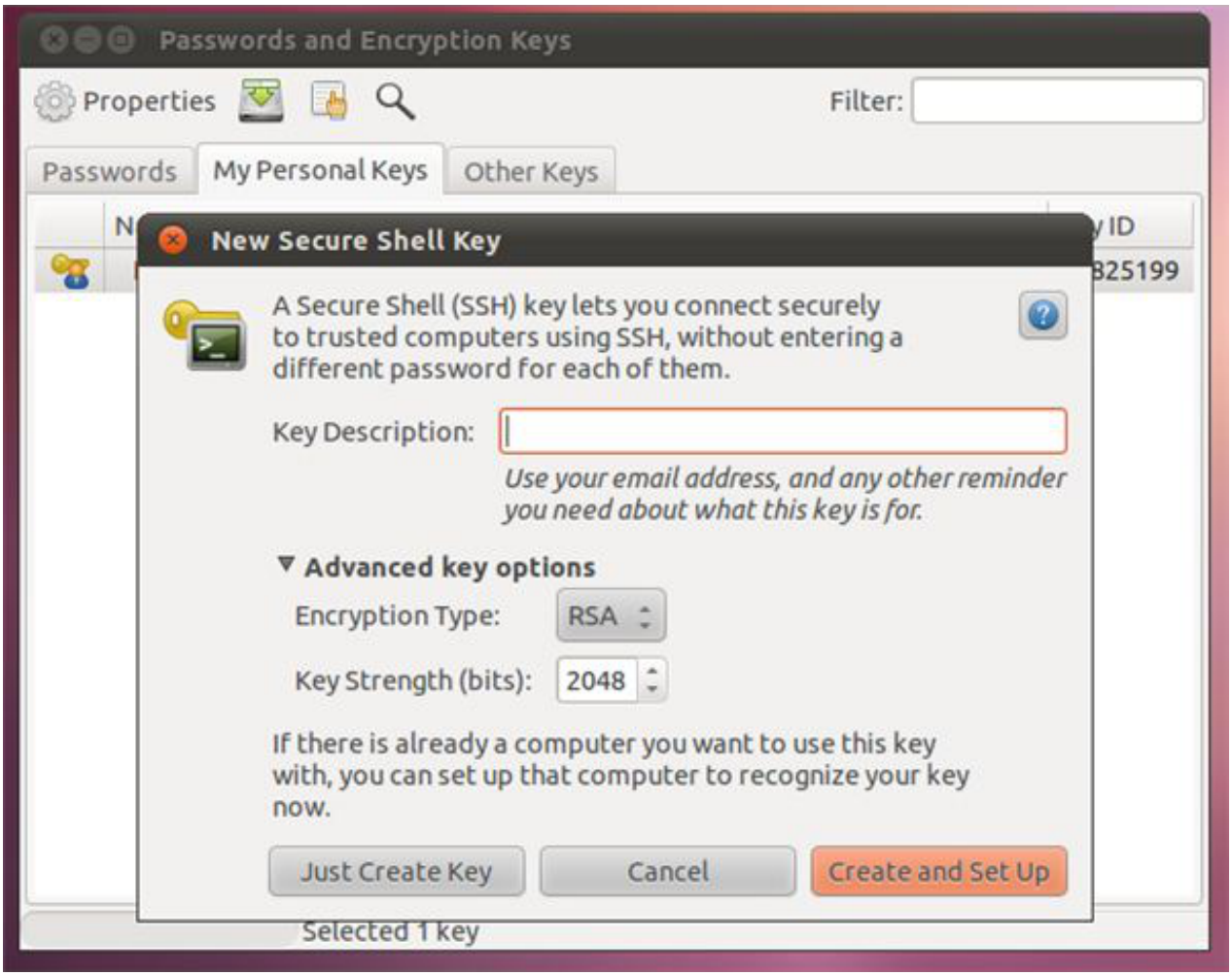
- Decrypting
  - gpg –output foo.txt –decrypt foo.txt.gpg

- Signatures
  - gpg –verify crucial.tar.gz.asc crucial.tar.gz
  - gpg –armor –detach-sign your-file.zip

Linux tools

For Linux

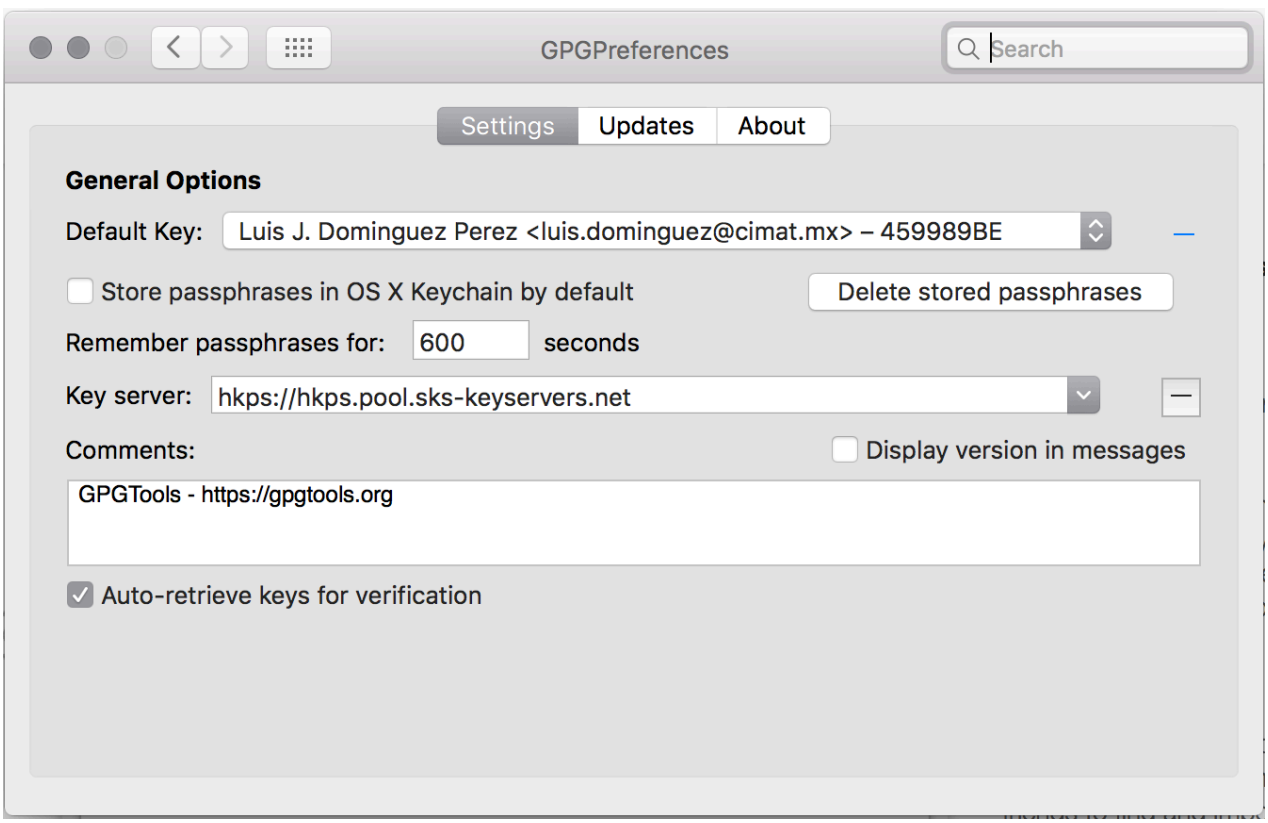
- We use GPG tools
- It has differences between the PGP official implementation
- It provides command line interface
- There are GUI tools, such as Seahorse



OSX Tools

GNU tools

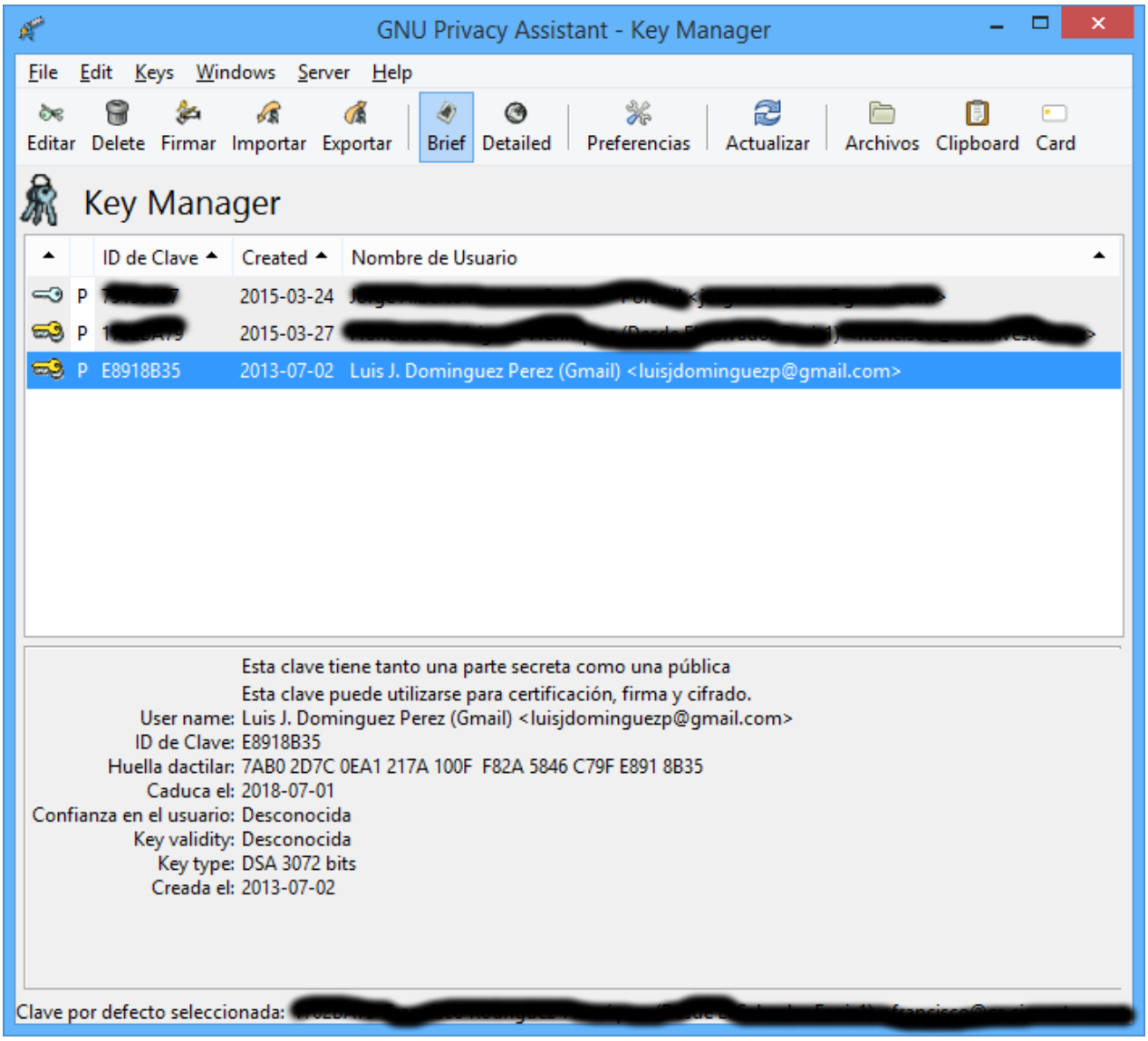
- Works for El Capitan (OS X 10.11
- GPG for Mail (Apple email)
- GPG keychain
- GPG services
- MacGPG



Windows Tools, etc.

For Windows, you have

- PGP by Symantec
- GPG 4 Win (for Outlook)
- GNU tools with Cygwin



Mailvelope

- Is a Chrome, and Firefox plug-in for using PGP encryption in your webmail clients.
- Uses: OpenPGP.js, email.js, DOMpurify, Bootstrap, jQuery, Oxygen icons

Supports:

- Key Management
- Key Generation
- Key Import/Export
- Encrypt/Decrypt, and signing

The keys are stored in your profile. Only Firefox provides a strong encryption for password, but I am not sure if also for the whole profile.

Mailvelope

- Is a Chrome, and Firefox plug-in for using PGP encryption in your webmail clients.
- Uses: OpenPGP.js, email.js, DOMpurify, Bootstrap, jQuery, Oxygen icons

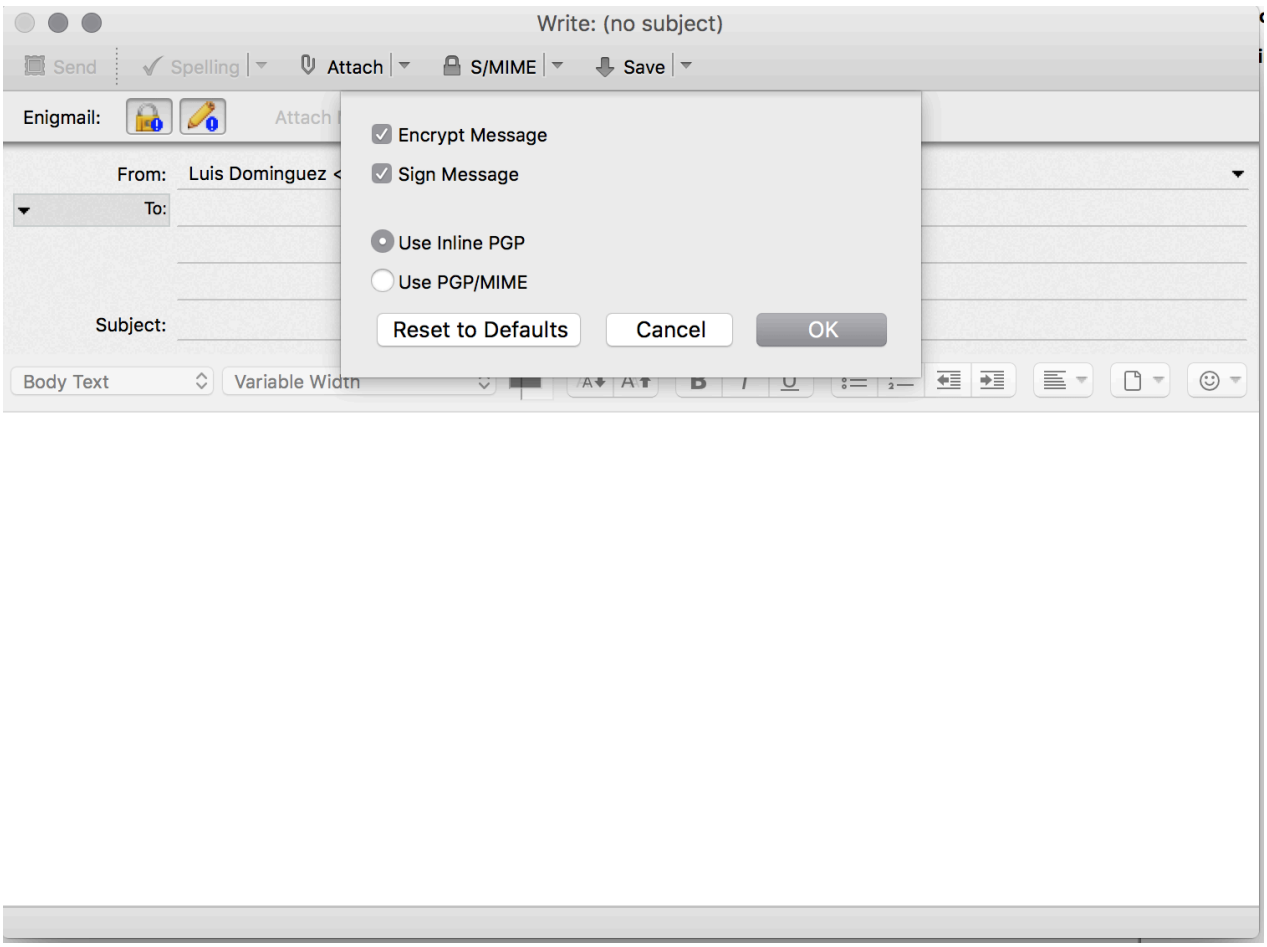
Supports:

- Key Management
- Key Generation
- Key Import/Export
- Encrypt/Decrypt, and signing

The keys are stored in your profile. Only Firefox provides a strong encryption for password, but I am not sure if also for the whole profile.

Multiplataform

- A cross platform solution is Enigmail, a plug-in for Thunderbird
- It connects to your GPG installation



Browsing

- Private mode browsing is now easy to activate on most browsers:
  - No saving cookies
  - No tracking
  - No history
  - No new passwords
  - No cache
  - Yes you can save bookmarks
  - Yes you can undo closing tabs
  - Yes, everybody could detect where you are browsing
  - Yes, you ID can be linked if you login into your account

That's nice, but not enough for many...

Browsing

- Private mode browsing is now easy to activate on most browsers:
  - No saving cookies
  - No tracking
  - No history
  - No new passwords
  - No cache
  - Yes you can save bookmarks
  - Yes you can undo closing tabs
  - Yes, everybody could detect where you are browsing
  - Yes, you ID can be linked if you login into your account

That's nice, but not enough for many...

Tor network

"Tor is free software and an open network that helps you defend against traffic analysis, a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security."

- It helps to anonymize your browsing experience from application providers
- Used by journalists, and people in general for free-speech
- Also used for illegal traffic, and terrorists
- Helps on testing network issues
- Military uses it for information gathering

Does not forbids you to leak your information (if you login into facebook, you are no longer anonymous)

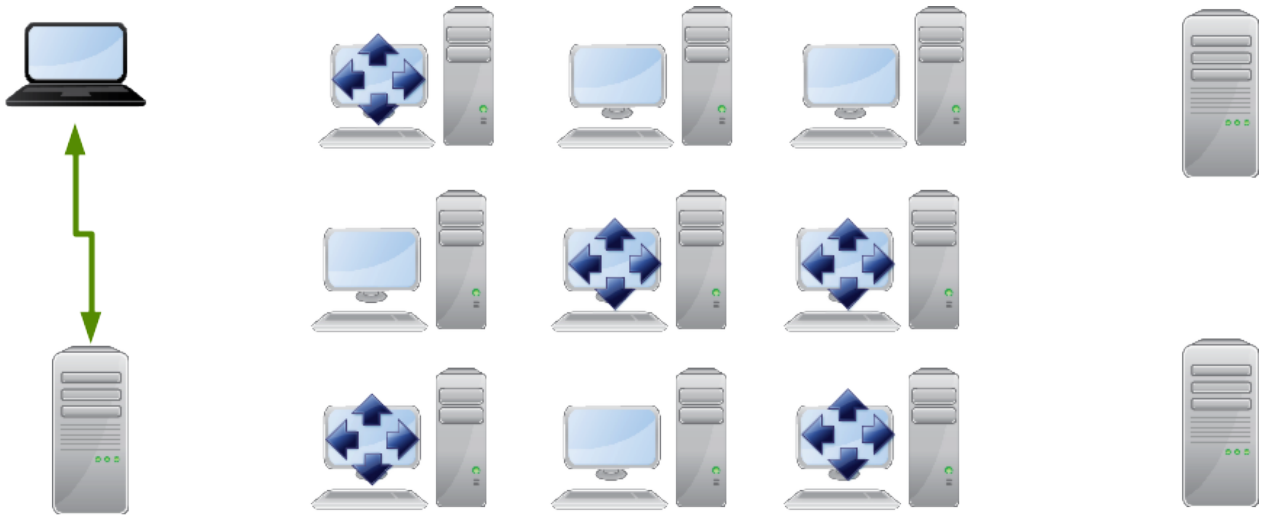
Tor network

"Tor is free software and an open network that helps you defend against traffic analysis, a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security."

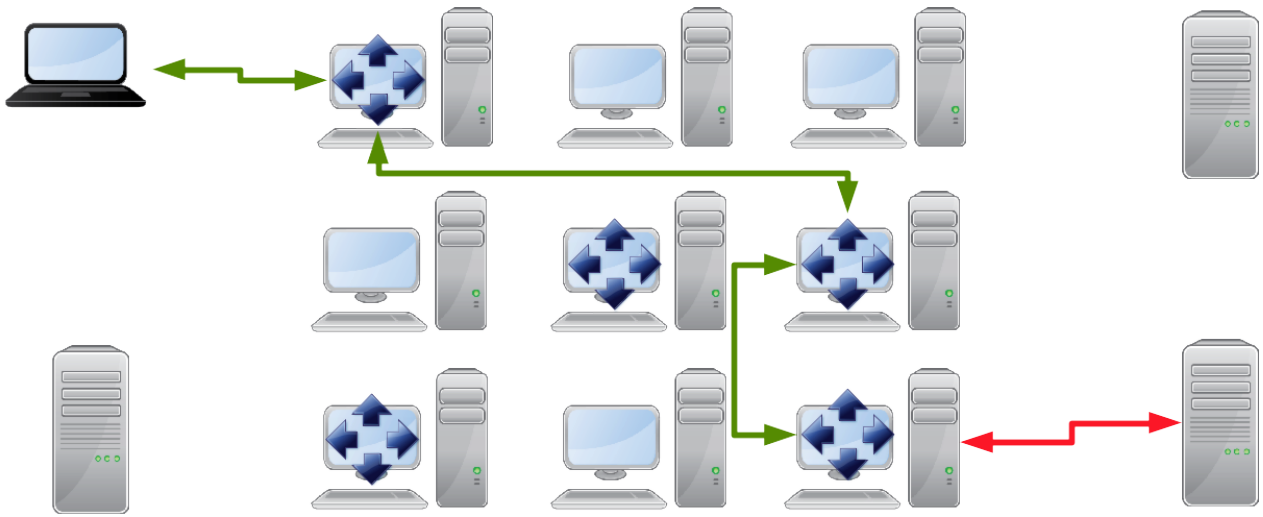
- It helps to anonymize your browsing experience from application providers
- Used by journalists, and people in general for free-speech
- Also used for illegal traffic, and terrorists
- Helps on testing network issues
- Military uses it for information gathering

Does not forbids you to leak your information (if you login into facebook, you are no longer anonymous)

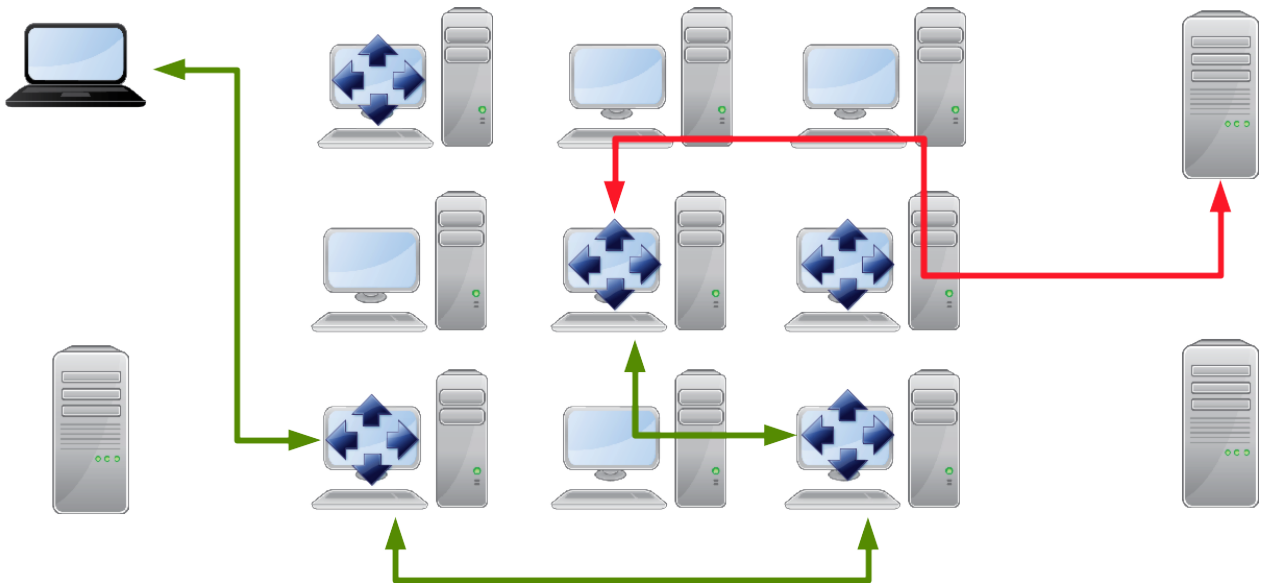
Tor diagram



Tor diagram



Tor diagram



Chat

Instant message conversations are also susceptible to be read on the internet

For mobile, we have very few solutions:

- Telegram - multiplataform, uses "perfect forward secrecy" (100 times, or weekly) for its secure chat mode... but it's an *in-house* protocol
- BBM using a BES uses Triple DES encryption, as it is the recommended standard in FIPS (they would change it accordingly)
- iMessage uses some sort of encryption, but it has a bad record... perhaps it uses AES-128 until you download your messages
- Another good alternative is Cryptocat, which also works on iPhone... they use OTR, with all the flashes

Chat

For servers we have:

- An XMPP server could be configured to use PGP for the messages
- Microsoft has an enterprise solution for chatting, and there are a few mobile applications... protects communication, but I am not sure how they store the messages until you receive them.

Future

We need to work on usability

but also more protocols, and primitives are needed

Future

We need to work on usability

but also more protocols, and primitives are needed