1 Certificates

```
/etc/ssl/certs $ cat GlobalSign_Root_CA.pem
-----BEGIN TRUSTED CERTIFICATE-----
MIIDdTCCAl2gAwIBAgILBAAAAAABFUtaw5QwDQYJKoZIhvcNAQEFBQAwVzELMAkG
A1UEBhMCQkUxGTAXBgNVBAoTEEdsb2JhbFNpZ24gbnYtc2ExEDAOBgNVBAsTB1Jv
b3QgQ0ExGzAZBgNVBAMTEkdsb2JhbFNpZ24gUm9vdCBDQTAeFw05ODA5MDExMjAw
MDBaFw0yODAxMjgxMjAwMDBaMFcxCzAJBgNVBAYTAkJFMRkwFwYDVQQKExBHbG9i
YWxTaWduIG52LXNhMRAwDgYDVQQLEwdSb290IENBMRswGQYDVQQDExJHbG9iYWxT
aWduIFJvb3QgQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDaDuaZ
jc6j40+Kfvvxi4Mla+pIH/EqsLmVEQS98GPR4mdmzxzdzxtIK+6NiY6arymAZavp
xy0Sy6scTHAHoT0KMM0VjU/43dSMUBUc71DuxC73/OlS8pF94G3VNTCOXkNz8kHp
1Wrjsok6Vjk4bwY8iGlbKk3Fp1S4bInMm/k8yuX9ifUSPJJ4ltbcdG6TRGHRjcdG
snUOhugZitVtbNV4FpWi6cgKOOvyJBNPc1STE4U6G7weNLWLBYy5d4ux2x8gkasJ
U26Qzns3dLlwR5EiUWMWea6xrkEmCMgZK9FGqkjWZCrXgzT/LCrBbBlDSgeF59N8
9iFo7+ryUp9/k5DPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNVHRMBAf8E
BTADAQH/MB0GA1UdDgQWBBRge2YaRQ2XyolQL30EzTSo//z9SzANBgkqhkiG9w0B
AQUFAAOCAQEA1nPnfE920I2/7LqivjTFKDK1fPxsnCwrvQmeU79rXqoRSLblCKOz
yj1hTdNGCbM+w6DjY1Ub8rrvrTnhQ7k4o+YviiY776BQVvnGCv04zcQLcFGUl5gE
38NflNUVyRRBnMRddWQVDf9VMOyGj/8N7yy5Y0b2qvzfvGn9LhJIZJrglfCm7ymP
AbEVtQwdpf5pLGkkeB6zpxxxYu7KyJesF12KwvhHhm4qxFYxldBniYUr+WymXUad
DKqC5JlR3XC321Y9YeRq4VzW9v493kHMB65jUr9TU/Qr6cf9tveCX4XSQRjbgbME
HMUfpIBvFSDJ3gyICh3WZlXi/EjJKSZp4DA0MB4GCCsGAQUFBwMEBggrBgEFBQcD
AQYIKwYBBQUHAwMMEkdsb2JhbFNpZ24gUm9vdCBDQQ==
-----END TRUSTED CERTIFICATE-----
```

ASN.1 is the data structure, it can be encoded in PEM format (base64) or DER format (binary). A certificate is encoded via the X.509 Certificate Data Management (see also pkcs7).

Unravelling the ASN.1 structure yields:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            04:00:00:00:00:01:0f:86:26:e6:0d
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: OU=GlobalSign Root CA - R2, O=GlobalSign, CN=GlobalSign
        Validity
            Not Before: Dec 15 08:00:00 2006 GMT
            Not After : Dec 15 08:00:00 2021 GMT
        Subject: OU=GlobalSign Root CA - R2, O=GlobalSign, CN=GlobalSign
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:a6:cf:24:0e:be:2e:6f:28:99:45:42:c4:ab:3e:
                    21:54:9b:0b:d3:7f:84:70:fa:12:b3:cb:bf:87:5f:
                    c6:7f:86:d3:b2:30:5c:d6:fd:ad:f1:7b:dc:e5:f8:
                    60:96:09:92:10:f5:d0:53:de:fb:7b:7e:73:88:ac:
                    52:88:7b:4a:a6:ca:49:a6:5e:a8:a7:8c:5a:11:bc:
                    7a:82:eb:be:8c:e9:b3:ac:96:25:07:97:4a:99:2a:
                    07:2f:b4:1e:77:bf:8a:0f:b5:02:7c:1b:96:b8:c5:
                    b9:3a:2c:bc:d6:12:b9:eb:59:7d:e2:d0:06:86:5f:
                    5e:49:6a:b5:39:5e:88:34:ec:bc:78:0c:08:98:84:
```

```
                    6c:a8:cd:4b:b4:a0:7d:0c:79:4d:f0:b8:2d:cb:21:
                    ca:d5:6c:5b:7d:e1:a0:29:84:a1:f9:d3:94:49:cb:
                    24:62:91:20:bc:dd:0b:d5:d9:cc:f9:ea:27:0a:2b:
                    73:91:c6:9d:1b:ac:c8:cb:e8:e0:a0:f4:2f:90:8b:
                    4d:fb:b0:36:1b:f6:19:7a:85:e0:6d:f2:61:13:88:
                    5c:9f:e0:93:0a:51:97:8a:5a:ce:af:ab:d5:f7:aa:
                    09:aa:60:bd:dc:d9:5f:df:72:a9:60:13:5e:00:01:
                    c9:4a:fa:3f:a4:ea:07:03:21:02:8e:82:ca:03:c2:
                    9b:8f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
                9B:E2:07:57:67:1C:1E:C0:6A:06:DE:59:B4:9A:2D:DF:DC:19:86:2E
            X509v3 CRL Distribution Points:
                URI:http://crl.globalsign.net/root-r2.crl

            X509v3 Authority Key Identifier:
                keyid:9B:E2:07:57:67:1C:1E:C0:6A:06:DE:59:B4:9A:2D:DF:DC:19:86:2E

    Signature Algorithm: sha1WithRSAEncryption
        99:81:53:87:1c:68:97:86:91:ec:e0:4a:b8:44:0b:ab:81:ac:
        27:4f:d6:c1:b8:1c:43:78:b3:0c:9a:fc:ea:2c:3c:6e:61:1b:
        4d:4b:29:f5:9f:05:1d:26:c1:b8:e9:83:00:62:45:b6:a9:08:
        93:b9:a9:33:4b:18:9a:c2:f8:87:88:4e:db:dd:71:34:1a:c1:
        54:da:46:3f:e0:d3:2a:ab:6d:54:22:f5:3a:62:cd:20:6f:ba:
        29:89:d7:dd:91:ee:d3:5c:a2:3e:a1:5b:41:f5:df:e5:64:43:
        2d:e9:d5:39:ab:d2:a2:df:b7:8b:d0:c0:80:19:1c:45:c0:2d:
        8c:e8:f8:2d:a4:74:56:49:c5:05:b5:4f:15:de:6e:44:78:39:
        87:a8:7e:bb:f3:79:18:91:bb:f4:6f:9d:c1:f0:8c:35:8c:5d:
        01:fb:c3:6d:b9:ef:44:6d:79:46:31:7e:0a:fe:a9:82:c1:ff:
        ef:ab:6e:20:c4:50:c9:5f:9d:4d:9b:17:8c:0c:e5:01:c9:a0:
        41:6a:73:53:fa:a5:50:b4:6e:25:0f:fb:4c:18:f4:fd:52:d9:
        8e:69:b1:e8:11:0f:de:88:d8:fb:1d:49:f7:aa:de:95:cf:20:
        78:c2:60:12:db:25:40:8c:6a:fc:7e:42:38:40:64:12:f7:9e:
        81:e1:93:2e
SHA1 Fingerprint=75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
```

2 Using SSH

```
client $ ssh-keygen
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
SHA256:oHoXTyPHvdqOXZkmZoNJDT2khWCSexiY9fSQFAAAyuQ dams@mithrim
The key's randomart image is:
```

```
+---[RSA 2048]----+
|+o.=++B+ .o      |
|= o o=.o.=       |
|.E   +o + o      |
|    o..o + .     |
|    ..o S o      |
|   .   B + . o   |
|  . . . + * =    |
|   . .   B =     |
|        o.+      |
+----[SHA256]-----+

client $ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_ed25519.
Your public key has been saved in id_ed25519.pub.
The key fingerprint is:
SHA256:gH64OdQ/XxQkD55gU6OiH5cqzjT+6QTjp/XgCCarol8 dams@mithrim
The key's randomart image is:
+--[ED25519 256]--+
|        +.= .    |
|     . . = B     |
|    . o . o o    |
|   . + o .   .   |
|    B + S   .    |
|   o B =   .     |
|. o E B o   .    |
|.+ B @ + o .     |
|B.. *o= . .      |
+----[SHA256]-----+


<Add id_rsa.pub or id_ed25519.pub to the server authorized_keys:>
server $ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCy5N7tFP8YSu6YDa8Rt6z4fOyI/
OP4Tix1X64ltBYEWWMP2PCOIYwr74bkL5WIZA3QCPGQnNjfGBUbU3hdwuYiIo4lfL
xW9KOTZg+erne2qJ1fHwn0dX9QGAE+5If7B5et+ciQ4t9XAB5ppNnkhB1xCQ5s1M8
12+hJsST3chJZXKrIknWFclagD03VCAMqC440RH6eHT9uWnDV9s5py+UBfzJQYmTN
zCIvoo+MrkNiPe7SErL/Cc305ss73MXz+A4Z5xNI6TfC5MZghcp8ioAMN22M9n7Z6
FdI/K1rXOdh6aRGO4hlfrRZb8V0vGA9Bz1ffC+G+ft7vzV9pxv5gSbd damien@mithrim

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIMoNrNYhU7CY1Xs6v4Nm1V6oRHs/F
EE8P+XaZOPcxPzz dams@mithrim

client $ ssh server
```

3 Using GPG 2.1

3.1 Generate Key

```
robert $ gpg --gen-key

GnuPG needs to construct a user ID to identify your key.
Real name: Damien Robert
Email address: foo@bar.org
You selected this USER-ID:
    "Damien Robert <foo@bar.org>"
Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.

gpg: key BF8C23DB marked as ultimately trusted
gpg: directory '/tmp/gpg/openpgp-revocs.d' created
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   rsa2048/C2545D77 2015-03-25
      Key fingerprint = 90C0 5A72 1762 7D96 0089  7D12 1A43 D6DF C254 5D77
uid       [ultimate] Damien Robert <foo@bar.org>
sub   rsa2048/27E67089 2015-03-25

couveignes $ gpg --gen-key
GnuPG needs to construct a user ID to identify your key.
Real name: Jean-Marc Couveignes
Email address: ploum@plam.org
You selected this USER-ID:
    "Jean-Marc Couveignes <ploum@plam.org>"
Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
pub   rsa2048/9FBDD7FF 2015-03-25
      Key fingerprint = 61C0 C173 C2EB 16FE 106E  1976 19C9 8B55 9FBD D7FF
uid       [ultimate] Jean-Marc Couveignes <ploum@plam.org>
sub   rsa2048/0ACA39B1 2015-03-25
```

3.2 Export and Import

```
couveignes $ gpg --export -a couveignes
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQENBFUS98UBCADbo32yPewjFpdGVyb4PRNHZpNi07gJvp7ZLRXM1qMU4E3jYyn5
5D6Dw8oACI+HJsa4NBoP/DLUH+am46wkRBGR26Vi9kpybSTe3hJDutLeekpLjuXr
+5Fr0iJz3aPdDp6INmfcWMLmUvu1jQp9j2chKDQGw1RyOmLf0yGu4Y3YpWmmw5h9
iEF86PufvboGMrWfa/7F8YzbED9jYIVrUO+G1+ZxbdkztF9WJYxmRnoy/yTBsd81
kI2EdFEuwLsLEXTsMlkvjAo/5uIk6acF/m/PLRaDfTXLQJixW+iv7O1XZtjLgzlO
+I5f6ajz7fx2DhCFJ6n2SH6xJtgyz7rwNU5NABEBAAG0JUplYW4tTWFyYyBDb3V2
ZWlnbmVzIDxwbG91bUBwbGFtLm9yZz6JATcEEwEIACEFAlUS98UCGwMFCwkIBwIG
FQgJCgsCBBYCAwECHgECF4AACgkQGcmLVZ+91/+TGgf/cE5UN/GiT7WFPZ8de5i+
dCchXgA+RILcqDMATRZPbAnZ1NhXg/KhlunG0YPKxaUVGTmUEuatfL4XplvE+3L0
lk8DMw7uKNkdU3klcQyYWtLnfDddkZXwYEIjyCQRK41RsoJRFgjnSM2lkq+xJOz5
LuPew6mBqYRxpWj4QDmpl99LkxI9rSZD7IssCLC8rEN4e38wr8ArLzGnmWNtFfQV
aAdFrYyYw/Y//uXuhhlaFHOj0Ci1KVNqHZUbN7xJVottYZOZZGBNgQiQxn+MO8ZL
pA2NS7IR9AVQHOsTEmnFE+M1bGB3iiNKCXvf/p+qTIHVhP+fWwksZk+jy8hFWZTD
P7kBDQRVEvfFAQgAwhgBt3q7/i9NY4qK9DV9siGC6l39AgrZKcoyTCQmEdZvTP1B
IRZJNFqMGggPezsYOhgyDjlwhe3I3K29UPEoQY/VxAVTfwUFDOCPwkFGQ++MSoap
9ZK+BpvfLpD2FrYADD0qlPLbui0TPg5Ck3Dd2PuShcAZGUTwrwknH2rxvYNtIr5r
7yYGGDa+9S2nmambM+br9Nu/kWNgZh8rAevElevzigEM3AFpZmFg6QKY+lcPjSip
teI0BwsLBz6QnrLK9NxcqYx1S3S8gHlfYfKJMtFqCYcLvWy20iBAjV9WiJWE8z3E
dsRMj9q087l6NuKpaZMHqJ2xZY+VElgocGAKAQARAQABiQEfBBgBCAAJBQJVEvfF
AhsMAAoJEBnJi1Wfvdf/+tkIALbKwv6GD2V5qYixs0ASQOPjw/G5ebGu6+/RZsMh
aQJ05BEfvbw+UZW7cmagyqNQ0qbF+3VhI2SKWJs8PIxglgaQ8MabawcVDhg/Nkxl
72H1OXeiNnIGA0mCDHTunBeett86p51FbbfQUMJmVR1Xkz+/f3o+AHWMJmtlicy0
WjKnbUYeUrcIKYVo4VEYb537mbLPWBMxe0kcpXCkRDmdJi0U+IieM5hswwSm4Nrf
NpOkXqU2ZUomfzO13vy4FHgwOJ/q4iGXLdnJogs//iuvlphwapqTcTYQd2ph8yYt
XYfbWTrazUud0OJqhQOsXFF6NIu3gYwfhbxW3MUyXoqJyYg=
=fMwt
-----END PGP PUBLIC KEY BLOCK-----

robert $ gpg --import couveignes.asc
gpg: key 9FBDD7FF: public key "Jean-Marc Couveignes <ploum@plam.org>" imported
gpg: Total number processed: 1
gpg:               imported: 1

robert $ gpg -u robert --sign-key couveignes
pub  rsa2048/9FBDD7FF
     created: 2015-03-25  expires: never       usage: SC
     trust: unknown       validity: unknown
sub  rsa2048/0ACA39B1
     created: 2015-03-25  expires: never       usage: E
[ unknown] (1). Jean-Marc Couveignes <ploum@plam.org>

pub  rsa2048/9FBDD7FF
     created: 2015-03-25  expires: never       usage: SC
     trust: unknown       validity: unknown
 Primary key fingerprint: 61C0 C173 C2EB 16FE 106E  1976 19C9 8B55 9FBD D7FF
     Jean-Marc Couveignes <ploum@plam.org>

Are you sure that you want to sign this key with your
key "Damien Robert <foo@bar.org>" (C2545D77)
```

```
Really sign? (y/N) y
<enter passphrase>
```

3.3 Encrypt

```
robert $ echo "RDV demain à 08h15" | gpg -a -e -r couveignes
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2

hQEMA5DMreYKyjmxAQf/ROwtDZQEcZjQ6GVLGdvz6kSX2l4h6hprnUb313NgdUr3
r2nPEOQE8I9+9ehkNSV3HZh+htalFZ8U4Tpd/JDZC+83gi820QQtvQcjSRtXnXfk
hx6wCMTSX+mHHl/Y3ACUQPswQsadsJsTFQkjbuevRyGIQgRyzk2muVfIB3DB4ngn
pn09OAsPAixagOxZ/KiG4ZO+VTPn5+Enp/aAEgHrRamxjkOIVh8FUnaRp6X+DFZf
xSG3ebBFwzMxF66qUPRxdCNULIsWZVdcjjD4rMMIoQwA49v4Gpr4cJuyj44QMrLw
Dxq/R8HNVTlBGs28UXC4On0l1KQ91A2n19dY/zyem9JPAc1QiziBFDL+agUtqUqE
nFfJmYzXe62Kx6TCaAQobHQe73DmTWC7/IgNmjDXYaVuJMq2KbDkxjuPzj9LUVuC
s1LOuXWf4c4nixdOez1DoA==
=Eu4p
-----END PGP MESSAGE-----

couveignes $ gpg -d secret.asc
gpg: encrypted with 2048-bit RSA key, ID 0ACA39B1, created 2015-03-25
      "Jean-Marc Couveignes <ploum@plam.org>"
RDV demain à 08h15
```

3.4 Signing

```
couveignes $ echo "Je dois 1000 à Damien Robert" | gpg -u couveignes --clearsign
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

Je dois 1000 à Damien Robert
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2

iQEcBAEBCAAGBQJVEv3qAAoJEBnJi1Wfvdf/XrMH/R8vmDiJFRAgQomhMuQc2VQe
/IK+yJAho90vIQycnQjmQWCHsrd4bsi53lyXUlJQKMORae7H+0SfXFZNL4tbrTl0
ruPIgRCuAGh9qGEuDds9t06yubICVXbIc+uZq7XLK9XtBKWogz6XvtVP/jRfJuUo
9ge3CsRK5gjcv0wc0jM/5aWFJFsMbGfE1alkXK49wDkIm33YfQq4Nu0WlNh+OjLd
R0Fjp9eunYjZGLaH6ZqNbyZaqvGe9r3EmAXkGdTxmj38t9G8DXVRubvGao0XEphh
wudRutW8vrSN37pT3Azv0kQ0g3iJ5v1v4HNziQEymQ70qaUnE81jFc5Xd0UIFG4=
=dqHC
-----END PGP SIGNATURE-----
# See also gpg -s to generate a binary signature and gpg -b to generate a
# detached signature

robert $ gpg -v message.asc
gpg: armor header: Hash: SHA256
gpg: armor header: Version: GnuPG v2
gpg: original file name=''
gpg: Signature made Wed Mar 25 19:26:50 2015 CET using RSA key ID 9FBDD7FF
gpg: using PGP trust model
```

```
gpg: Good signature from "Jean-Marc Couveignes <ploum@plam.org>" [full]
gpg: textmode signature, digest algorithm SHA256, key algorithm rsa2048

robert $ sed -e s/1000/1000000/ message.asc | gpg -v
gpg: armor header: Hash: SHA256
gpg: armor header: Version: GnuPG v2
gpg: original file name=''
Je dois 1000000 à Damien Robert
gpg: Signature made Wed Mar 25 19:26:50 2015 CET using RSA key ID 9FBDD7FF
gpg: using PGP trust model
gpg: BAD signature from "Jean-Marc Couveignes <ploum@plam.org>" [full]
gpg: textmode signature, digest algorithm SHA256, key algorithm rsa2048
```

Signature in an email:

```
[-- PGP output follows (current time: Wed Mar 25 19:38:52 2015) --]
gpg: Signature made Tue Mar 24 18:52:17 2015 CET using RSA key ID CBC32853
gpg: Good signature from "Damien Robert (Master Public Key)
<foo@bar.org>" [ultimate]
gpg:                 aka "[jpeg image of size 3304]" [ultimate]
gpg:                 aka "Damien Robert (Personal:
http://www.normalesup.org/~robert/perso/) <personal_email>"
[ultimate]
gpg:                 aka "Damien Robert (Work: inria,
http://www.normalesup.org/~robert/pro/) <work_email>" [ultimate]
[-- End of PGP output --]

[-- The following data is signed --]
...
```