Conteúdo

- Secure Sockets Layer/Transport Layer Security (SSL/TLS)
- OpenSSL - o canivete suíço da criptografia
  - `https://openssl.org/docs/manmaster/apps/`
- GPG
- Além da disciplina

SSL/TLS

- Família SSL/TLS são protocolos de segurança para transmissão de dados com o TCP
  - $\pm$ "camada de apresentação"
  - SSL estão descontinuados (RFC 7568)
  - TLS 1.2 é atual, TLS 1.3 está em desenvolvimento (Jan. 2016)
    - `https://tlswg.github.io/tls13-spec/`
- Pai do SSL: Dr. Taher Elgamal (Netscape Communications)

Aplicação SSL/TLS

- Uso: emprego entre TCP e HTTP
  - Atacante poderá obter somente o IP, porta, quantidade de dados transmitida, tipo de encriptação usada e método de compressão
  - Atacante não saberá qual URL exatamente é transmistido (apenas o domínio pelo DNS)
- 2 fases
  - Protocolo de handshake
  - Protocolo de registro

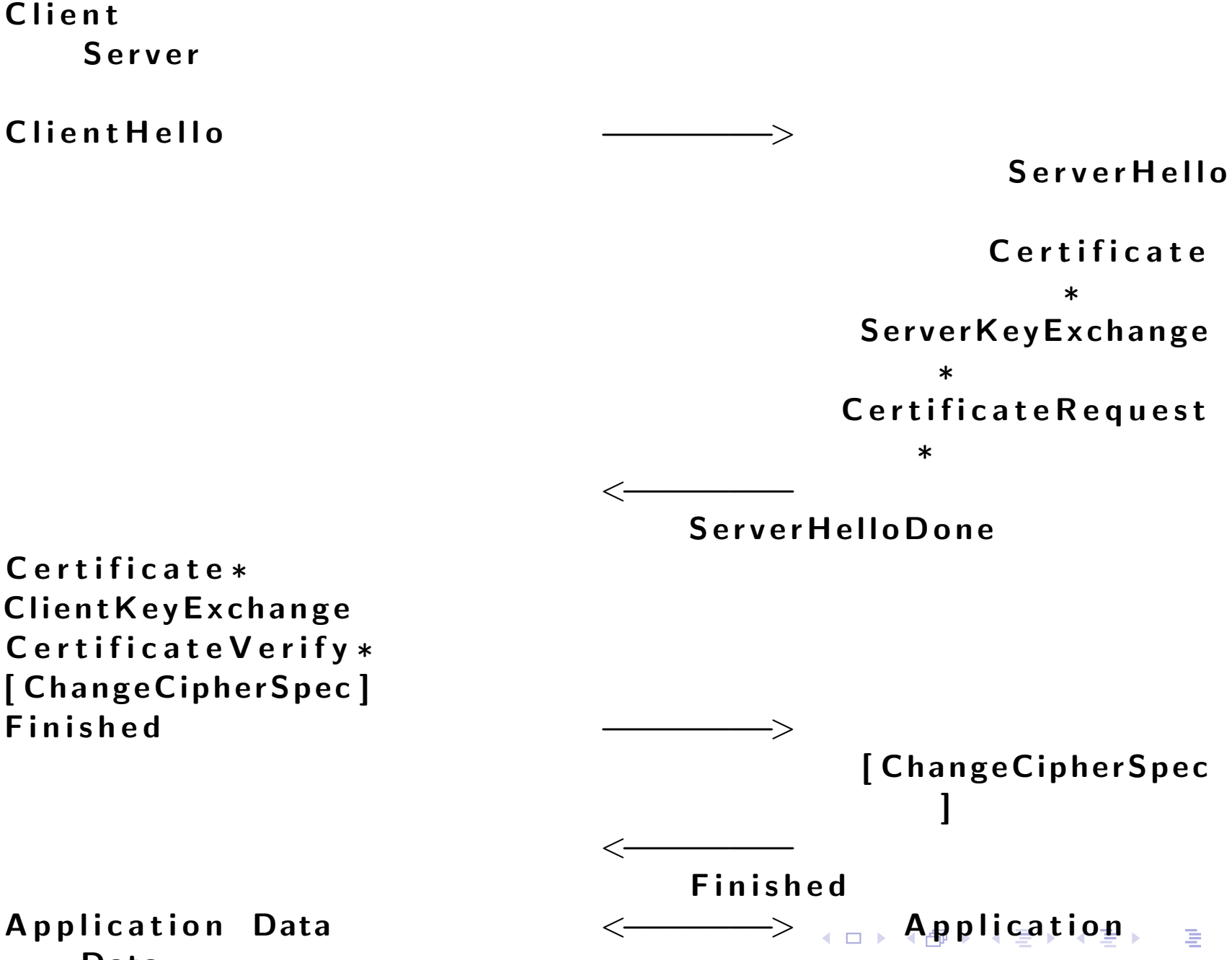
SSL handshake

- Depois de estabelecer conexão TCP, cliente inicia handshake SSL com
  - versão SSL/TLS usada
  - conjunto de primitivas criptográficas desejadas
  - método de compressão (opcional)
- Servidor responde com
  - versão mais alta de SSL/TLS em comum
  - primitiva criptográfica aceita
  - certificado próprio
- Cliente deve confiar no certificado do servidor

SSL handshake: troca de chaves

- Após o cliente confiar no certificado, ocorre a troca de chaves
- Opções, de acordo com a primitiva escolhida:
  - chave pública
  - `PreMasterSecret` (RSA com detalhes de implementação)
- Após chave trocada, inicia-se encriptação simétrica
  - usando MAC encrypt-then-mac (tipo HMAC) para encriptação autenticada
  - em blocos (CBC,GCM) ou stream (RC4)

RFC 5246: Message flow for a full handshake



RFC 5246: ClientHello

```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites<2..2^16-2>;
    CompressionMethod compression_methods<1..2^8-1>;
    select (extensions_present) {
        case false:
            struct {};
        case true:
            Extension extensions<0..2^16-1>;
    };
} ClientHello;
```

RFC 5246: ServerHello

```
struct {
      ProtocolVersion server_version;
      Random random;
      SessionID session_id;
      CipherSuite cipher_suite;
      CompressionMethod compression_method;
      select (extensions_present) {
          case false:
              struct {};
          case true:
              Extension extensions<0..2^16-1>;
      };
} ServerHello;
```

RFC 5246: extensão de assinatura

Extensão "signature_algorithms" indica par de algoritmos de assinatura e hashing usado em assinatura digital.

```
enum {
      none(0), md5(1), sha1(2), sha224(3), sha256(4), sha384
          (5),
      sha512(6), (255)
} HashAlgorithm;

enum { anonymous(0), rsa(1), dsa(2), ecdsa(3), (255) }
   SignatureAlgorithm;

struct {
        HashAlgorithm hash;
        SignatureAlgorithm signature;
} SignatureAndHashAlgorithm;

SignatureAndHashAlgorithm
   supported_signature_algorithms <2..2^16-2>;
```

Protocolos de troca de chaves

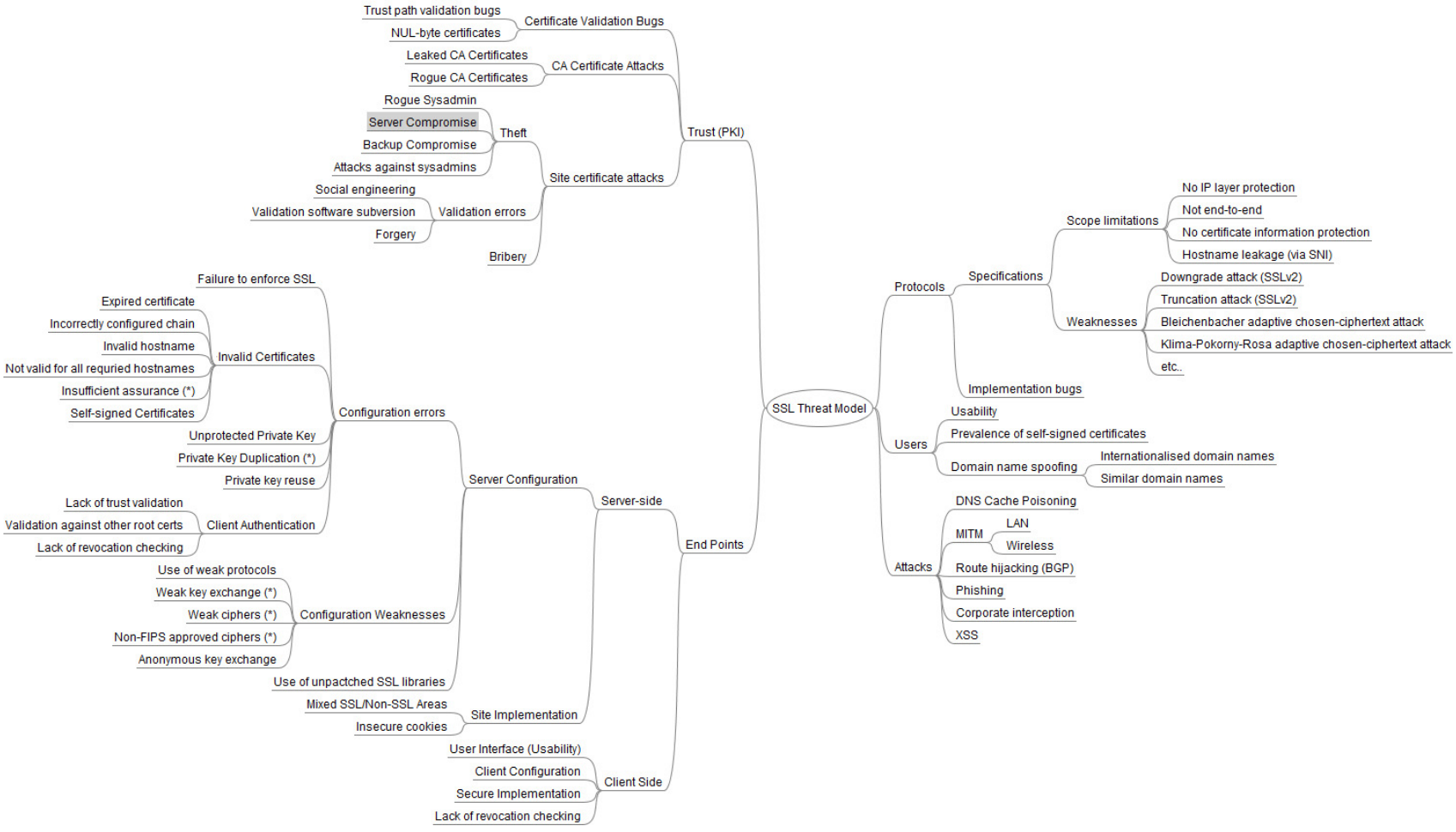
- RSA
  - Cliente deve gerar valor aleatório e concatenar com versão do SSL e criar `PreMasterSecret`
  - Servidor não tem `ServerKeyExchange`
- DHE_RSA
  - Chave do servidor é RSA mas apenas para assinatura.
  - Servidor prepara parâmetros de Diffie-Hellman para cliente
  - Servidor assina `ServerKeyExchange`
  - Cliente gera chave efêmera e envia um `ClientKeyExchange`
- DHE_DSS
  - Como DHE_RSA, mas somente para assinaturas
- ECDHE
  - Troca similar a DHE_RSA mas com curvas elípticas
- PSK
  - Chave secreta pré-compartilhada
- DH_anon
  - Inseguro pois é DH sem certificado

Protocolo da Camada de Registro

- Camada de registro: *Record Layer*
- Partes compartilham k_C, k'_C, k_S, k'_S
- Partes usam múmeros de sequência que previnem contra ataques replay
- Cliente usa k_C, k'_C para encriptar e autenticar todas as mensagens que ele envia
- Servidor usa k_S, k'_S para encriptar e autenticar todas as mensagens que ele envia
  - Previne ataque de reflexão

Ataques ao SSL

- Induzir o usuário a ignorar avisos que certificado está incorreto
- Version rollback (mudar para versão mais antiga com problemas)
- Uso de conjunto de primitivas fracas
  - `TLS_RSA_WITH_NULL_SHA`
    - Sem criptografia, somente integridade SHA
- Oráculo de Padding de Bleichenbacher
  - Modo de troca de chaves com RSA
- Oráculo de Padding de Vaudenay (2002) contra registros
  - E de Rizzo and Duong (2010)
- BEAST attack contra cliente ao forçar uso de RC4
- CRIME attack contra compressão
- Oráculo de Padding tipo Poodle
- ... outros contra implementações...
- `http://blog.ivanristic.com/SSL_Threat_Model.png`


SSL Threat Model
Trust (PKI)
Certificate Validation Bugs
Trust path validation bugs
NUL-byte certificates
CA Certificate Attacks
Leaked CA Certificates
Rogue CA Certificates
Site certificate attacks
Theft
Rogue Sysadmin
Server Compromise
Backup Compromise
Attacks against sysadmins
Validation errors
Social engineering
Validation software subversion
Forgery
Bribery
End Points
Server-side
Server Configuration
Configuration errors
Failure to enforce SSL
Invalid Certificates
Expired certificate
Incorrectly configured chain
Invalid hostname
Not valid for all requried hostnames
Insufficient assurance (*)
Self-signed Certificates
Unprotected Private Key
Private Key Duplication (*)
Private key reuse
Client Authentication
Lack of trust validation
Validation against other root certs
Lack of revocation checking
Configuration Weaknesses
Use of weak protocols
Weak key exchange (*)
Weak ciphers (*)
Non-FIPS approved ciphers (*)
Anonymous key exchange
Use of unpatched SSL libraries
Site Implementation
Mixed SSL/Non-SSL Areas
Insecure cookies
Client Side
User Interface (Usability)
Client Configuration
Secure Implementation
Lack of revocation checking
Protocols
Specifications
Scope limitations
No IP layer protection
Not end-to-end
No certificate information protection
Hostname leakage (via SNI)
Weaknesses
Downgrade attack (SSLv2)
Truncation attack (SSLv2)
Bleichenbacher adaptive chosen-ciphertext attack
Klima-Pokorny-Rosa adaptive chosen-ciphertext attack
etc..
Implementation bugs
Users
Usability
Prevalence of self-signed certificates
Domain name spoofing
Internationalised domain names
Similar domain names
Attacks
DNS Cache Poisoning
MITM
LAN
Wireless
Route hijacking (BGP)
Phishing
Corporate interception
XSS

OpenSSL - Cifras simétricas

```
openssl enc options are
-in <file>      input file
-out <file>     output file
-pass <arg>     pass phrase source
-e              encrypt
-d              decrypt
-a/-base64      base64 encode/decode, depending on encryption fla
-k              passphrase is the next argument
-kfile          passphrase is the first line of the file argument
-md             the next argument is the md to use to create a ke
                from a passphrase.  One of md2, md5, sha or sha1
-S              salt in hex is the next argument
-K/-iv          key/iv in hex is the next argument
-[pP]           print the iv/key (then exit if -P)
-bufsize <n>    buffer size
-nopad          disable standard block padding
-engine e       use engine e, possibly a hardware device.
```

OpenSSL - Cifras simétricas

```
> openssl enc -a -p -e -in "m.txt" -des-cbc -out "c.txt"
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
salt=B2E338BDE7A7C3D2
key=1DDA6D6073C26A85
iv =D688F0E3C18C2FD6
>cat c.txt
U2FsdGVkX1+y4zi956fD0ksDpwtmFAxc6y4/npi4+Xs=
```

OpenSSL: Chaves RSA

```
> openssl genrsa -out chave.pem 1024
Generating RSA private key, 1024 bit long modulus
.....................++++++
..++++++
```

- Ponto · indica que candidato a primo um teste rápido (sieve) de primalidade
- Mais + indica teste de Miller-Rabin

OpenSSL: Chaves RSA

```
> cat chave.pem
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC4m9V7gtWjpYHoycnuZBT9rKOGQpi4KhX9GibRaYZbxfY
A3YhWAmr9BiS9pOIRiT+IR5bbhWOf8OgONL3lY5ajKsOI7fuZcBOSRTyb/s
XtEQjboO5VASsZQaHgYNvA1UOcdgo8JTlwvxNSA1M3b+xGzpq37AtGwjAQI
AoGABtvrY6ppy6nDmOWedhgL25OFemCREg6mQIAvPwqfrls9mGsJ7tRtGDI
11OIWp1YlISgfx8o9+FxuqfB1tbXBvLrqvKo1h4uIV6oAfhMRU/UWua8y9+
MJRjOn8mOOpHy9Q991azazd5HreCYNs9M9AC42WetM96hAOCQQD1QLFS7ml
Pe9WhK/XKvy1iNtddih/NdrEkvheVpD6ehL5HmNeqQoWDjTDgUxENPuuFj1
DngbSVVfAkEAwLLSVpVxbdVsOe5I/z63Q1B34+Aj99HadTO/NgvGeQvRygB
/35DjzNyGnilgXelAfR2ZdHbTTQ1BiADnwJAQbtWC3Eku31CYwSOig7EBjp
We9aSUYI3Vi6wqYW81sWjLhNT8ikkGa8U43q//jnX7BMQakT3OLo4/9SpwJ
v6pOQafuuCH2gFIdHsI161K2FFse5J0984oVHGmOhUehI2MObnR5ciIDORz
BC5h9V9GZygLgf4lbXcCQBBgBNq4e9vqYpArPToEKh+Gvo8Dipx5gULGIvZ
t9bK7rnc95aPOsXWTUOOheYkvgqMagWx7t1RfxW2hUw=
-----END RSA PRIVATE KEY-----
```

OpenSSL: Chaves RSA – detalhes

```
> openssl rsa -in chave.pem -text
Private-Key: (1024 bit)
modulus:
    00:b8:9b:d5:7b:82:d5:a3:a5:81:e8:c9:c9:ee:64: (encurtado)
publicExponent: 65537 (0x10001)
privateExponent:
    06:db:eb:63:aa:69:cb:a9:c3:98:e5:9e:76:18:0b: (encurtado)
prime1:
    00:f5:40:b1:52:ee:69:44:0c:45:16:3d:ef:56:84: (encurtado)
prime2:
    00:c0:b2:d2:56:95:71:6d:d5:6c:39:ee:48:ff:3e: (encurtado)
exponent1:
    41:bb:56:0b:71:24:bb:7d:42:63:04:b4:8a:0e:c4: (encurtado)
exponent2:
    00:9f:12:bf:aa:74:41:a7:ee:b8:21:f6:80:52:1d: (encurtado)
coefficient:
    10:60:04:da:b8:7b:db:ea:62:90:2b:3d:3a:04:2a: (encurtado)
```

OpenSSL: guardar chave privada RSA encriptadas por AES

```
> openssl rsa -in chave.pem -aes256 -out RSAcomAES.pem
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

> cat RSAcomAES.pem
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,076D147D5E6C5A46F091B3979F94E0EF

ORyLDJCa9S9zIM1+mSHsG6UXJHq46eGBQx6F41r3RpTUkUSmqEfPJjC1z17
m1svVfrCtJG8mjKPW8Byj5a37zgVbuKfvL+V6Oo7WZ1k73K/3+Os8ZfwOhh
EaeT+5pteIqxkiBTRXOiwg==
-----END RSA PRIVATE KEY-----
```

OpenSSL: obter chave pública para divulgação

- `openssl genrsa -out chave.pem 128`

```
-----BEGIN RSA PRIVATE KEY-----
MGICAQACEQCvBByE9OjGfz5O3/2vehVFAgMBAAECEGhfn1J7bEbTpyC
CQDX/5hiA/lxdQIJAM9tjsmJMUqRAgkAsMxCY4ATTfOCCGxqaOOxhc/
uhCbzw==
-----END RSA PRIVATE KEY-----
```

- `openssl rsa -in chave.pem -pubout`

```
writing RSA key
-----BEGIN PUBLIC KEY-----
MCwwDQYJKoZIhvcNAQEBBQADGwAwGAIRAK8EHIT06MZ/Pk7f/a96FUU
-----END PUBLIC KEY-----
```

OpenSSL: usando chaves RSA

```
openssl rsautl
  Usage: rsautl [options]
  -in file        input file
  -out file       output file
  -inkey file     input key
  -keyform arg    private key format - default PEM
  -pubin          input is an RSA public
  -certin         input is a certificate carrying an RSA public ke
  -ssl            use SSL v2 padding
  -raw            use no padding
  -pkcs           use PKCS#1 v1.5 padding (default)
  -oaep           use PKCS#1 OAEP
  -sign           sign with private key
  -verify         verify with public key
  -encrypt        encrypt with public key
  -decrypt        decrypt with private key
  -hexdump        hex dump output
  -engine e       use engine e, possibly a hardware device.
  -passin arg    pass phrase source
```

OpenSSL: encriptação com RSA

- `openssl rsautl -encrypt m.txt -inkey chave.pem`

```
openssl rsautl -encrypt -hexdump -in m.txt -inkey chave.pem
0000 - 21 7b 3d 5a d1 41 20 77-f3 be 5b 77 ad 1d 26 31   !
0010 - 38 af f0 2d ef ae a6 8a-36 b5 d1 d2 70 53 ad 22   8
0020 - 33 4e 4f 29 c6 b8 1e c5-a8 c8 75 f1 b6 f9 b7 03   3
0030 - 28 47 2b 53 e4 f5 61 6f-36 5c 71 cd 9a 4f 01 50   (
0040 - b0 9e 7b eb 33 46 e4 ea-12 6a 2b ee 78 fa f9 ed   .
0050 - 10 5d 3e 27 b0 9c 03 51-7d 1a 3e f1 34 4b 49 8d   .
0060 - 15 34 53 15 48 11 b5 52-8b 22 5a 0a 32 6f 4c 83   .
0070 - cf 22 dc df 7a 71 16 d7-34 2e 26 cc 30 c3 7f 4c   .
```

OpenSSL: assinatura digital com RSA

- `openssl rsautl -sign -in m.txt -inkey chave.pem`

```
openssl rsautl -sign -hexdump -in m.txt -inkey chave.pem
0000 - 42 b2 4e f3 d1 b3 87 75-15 e8 77 83 3f f2 82 5b   B.
0010 - 69 b6 82 9a d8 67 af 3c-d6 c4 80 49 5e 00 ed a2   i.
0020 - 64 d7 59 e7 ea 70 3b ba-cb 6b 32 a3 d8 37 dd b2   d.
0030 - 2a a6 3e 6f 18 2a 14 75-30 91 c7 21 28 cc 03 1c   *.
0040 - 68 5b 0a f7 20 10 4d a8-b0 3c 82 f8 03 98 f9 cf   h
0050 - 26 7d b8 4a d7 28 06 3d-2f 82 8c 82 17 fc 2c f3   &]
0060 - 52 ff 3d 72 68 cf 77 32-d8 ef 16 e8 26 c6 a0 23   R.
0070 - 33 f4 aa 06 25 3f 70 73-03 d1 d3 61 56 cf 1e a3   3.
```

OpenSSL: funções hashes

- sha1,sha256,sha512,md5,whirlpool etc.

```
> openssl sha512 m.txt
SHA512(m.txt)= 1098c97c69c1f681ea04739dfd86c27648b202b706f4

> openssl dgst -whirlpool m.txt
 whirlpool(m.txt)= 8ecfac94e6a8e1323b3d73399ba02790bf7cc3d2
```

OpenSSL: HMAC

- Message Authentication Code (MAC) com Hash

```
> openssl dgst -whirlpool -hmac chave.pem m.txt
HMAC-whirlpool(m.txt)= 7f2c27a7a6f068b5412a8bba90dacb4751
```

OpenSSL: assinatura digital com RSA usando Hash

- É mais eficiente usar hash do que RSA puro

```
> openssl genrsa -out chave.pem 8096
> openssl dgst -whirlpool -out h.txt m.txt
> openssl rsautl -hexdump -sign -in h.txt -inkey chave.pem
0000 - b1 30 ad 04 49 af b0 2f-6c e2 09 c7 a9 e5 3e 65   .0
0010 - b4 0a 5c 7e 37 de 60 19-94 54 04 dc a1 a8 36 51   .
0020 - 33 06 55 b7 28 d9 25 ac-66 59 84 94 e2 7e 4c 2c   3
0030 - 79 bc 33 90 98 5f 10 12-15 e4 fd 65 50 08 bb a6   y
0040 - fa a6 c4 b1 32 19 d1 c9-62 ff a6 7f ba 61 ea c0   .
  ... (continua)

> openssl rsautl -out sig -sign -in h.txt -inkey chave.pem
> openssl rsa -in chave.pem -pubout -out kpub.pem
> openssl rsautl -verify -in sig -inkey kpub.pem -pubin
```

Requisição de certificados

- Criação da requisição: Certificate Signing Request CSR
- Passo 1:

```
openssl req -newkey rsa:1024 -nodes -keyout k.pem -out req.csr
```

- Passo 2: enviar requisição para CA

Certificados auto-assinados

- Usar certificados auto-assinados somente em servidores privados

```
openssl req -new -x509 -days 365 -newkey rsa:1024 -keyout k.pem -out cert.crt
```

- Usando chave prévia:

```
openssl req -new -key site.key -nodes -x509 -days 300 -out domain.crt
```

- Mostrar dados de certificados

```
openssl x509 -text -in domain.crt
```

GPG

- GnuPG: GNU Privacy Guard
- `https://vimeo.com/56881481`
  - Criptografia assimétrica
  - Assinaturas
- `http://www.ietf.org/rfc/rfc2440.txt`

Criação e revocação de chaves

- `gpg --gen-key`
- `gpg --full-gen-key`
- `gpg --output revoke.asc --gen-revoke mykey`

Geração de chaves

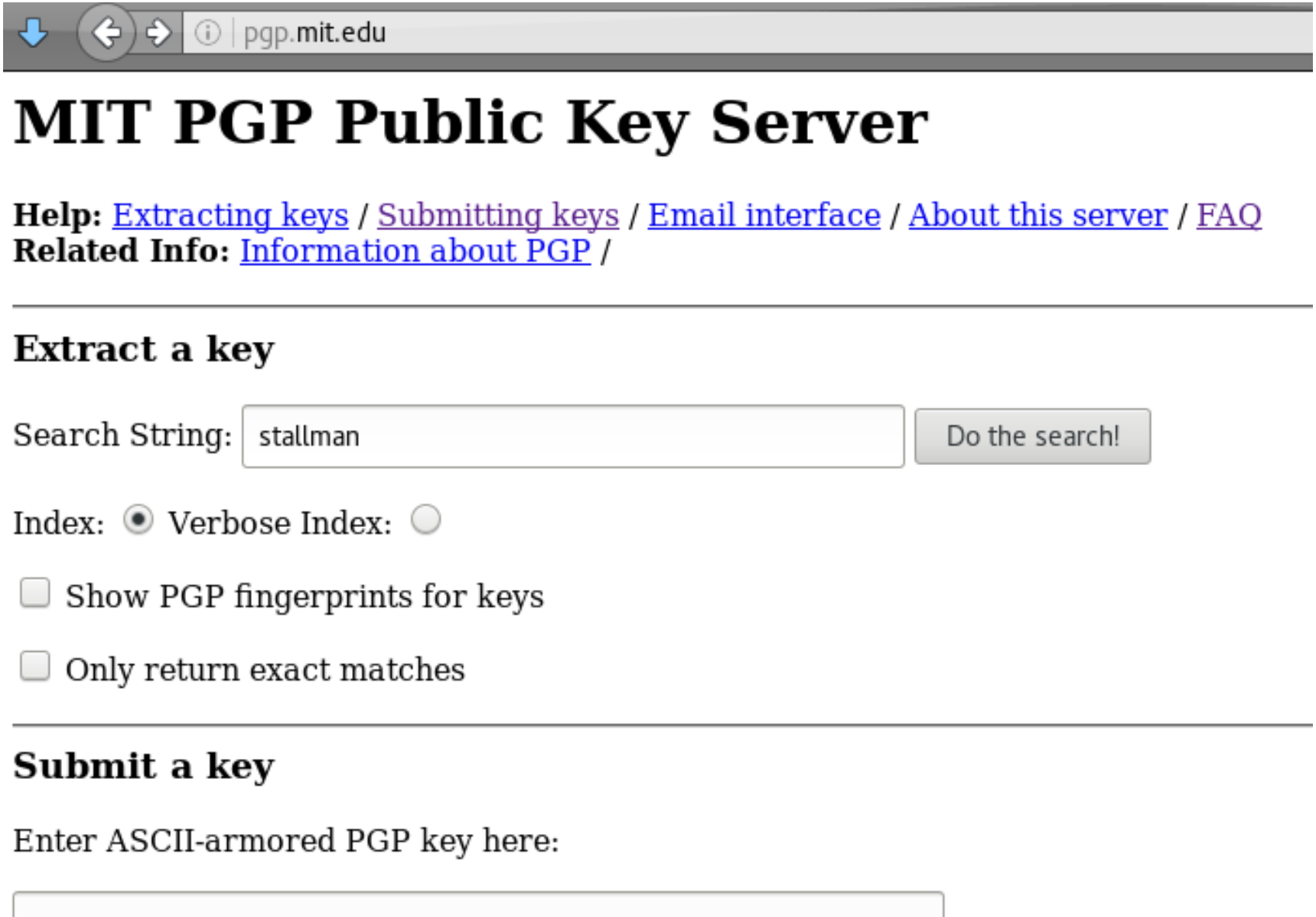
```
> gpg --gen-key
Nome completo: masace
Endereço de correio eletrónico: mas@sma.com
Você selecionou este identificador de utilizador:
    "masace-<mas@sma.com>"

gpg: key 3BB432FC marked as ultimately trusted
gpg: directory '/home/mas/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/mas/.gnupg/
    openpgp-revocs.d/
    B185FF7F2C445DCE239022CF2D2F52723BB432FC.rev'
chaves pública e privada criadas e assinadas.

gpg: a verificar a base de dados de confiança
gpg: marginals needed: 3  completes needed: 1  trust model
    : pgp
gpg: depth: 0  valid:  1  signed:  0  trust: 0-, 0q, 0n,
     0m, 0f, 1u
pub   rsa2048/3BB432FC 2016-06-20 [S]
      Key fingerprint = B185 FF7F 2C44 5DCE 2390  22CF 2
          D2F 5272 3BB4 32FC
uid          [ultimate] masace <mas@sma.com>
sub   rsa2048/C76FC817 2016-06-20 []
```

Troca de chaves

```
$ gpg --list-keys
/home/mas/.gnupg/pubring.gpg
------------------------------------------------
pub   rsa2048/3BB432FC 2016-06-20 [SC]
uid                [ultimate] masace <mas@sma.com>
sub   rsa2048/C76FC817 2016-06-20 [E]
```

Busca e distribuição de chaves

- Fazer `gpg --armor --export mas@sma.com`
- Usar `pgp.mit.edu` para enviar chave pública



Adicionando chaves

- Consultar em `http://pgp.mit.edu/pks/lookup?op=get&search=0x2574BF172A8E4C02`
- `gpg --armor --import rms.pubkey`

```
gpg: key 2A8E4C02: public key "Richard Stallman <rms@gnu.
    org>" imported
gpg: Número total processado: 1
gpg:                importados: 1
gpg: marginals needed: 3  completes needed: 1  trust model
    : pgp
gpg: depth: 0  valid:  1  signed:  0  trust: 0-, 0q, 0n,
     0m, 0f, 1u
```

Adicionando chaves

- `gpg --keyserver subkeys.pgp.net --recv-keys 6092693E`
- `gpg --keyserver hkp://pgp.mit.edu --recv-keys 2A8E4C02`
- `gpg --keyserver hkp://pgp.mit.edu:80 --keyserver-options http-proxy=http://proxy.ufu.br:3128 --recv-keys 2A8E4C02`
- `gpg --keyserver hkp://pgp.mit.edu:80 --keyserver-options http-proxy=http://proxy.ufu.br:3128 --search-key "Linus Torvalds"`

Assinando assinaturas de terceiros

- `gpg --edit-key stallman`

```
pub  rsa4096/2A8E4C02
     created: 2014-06-16  expires: never       usage: SCEA
     trust: unknown       validity: full
[  full  ] (1). Richard Stallman <rms@gnu.org>

sign
```

Trocando mensagens com terceiros

- Encriptando

```
gpg --output doc.gpg --encrypt --recipient rms@gnu.org doc
```

- Decriptando

```
gpg --output doc --decrypt doc.gpg
```

Assinatura integrada ao arquivo

- Assinatura de arquivos

```
gpg --output h.txt.sig  --clearsign h.txt
```

- Verificação de assinatura

```
gpg --output hh.txt --decrypt h.txt.sig
```

Assinatura separada do arquivo

```
> gpg --output h.txt.sig   --detach-sign h.txt
> gpg --verify h.txt.sig h.txt
gpg: Signature made Seg 20 Jun 2016 18:54:03 BRT using RSA
     key ID 7986369A
        gpg: Good signature from "masace-<mas@sma.com>" [
            ultimate]
```

Programando com GPG

- GPGME: biblioteca para usar GPG em códigos C
  - `http://www.nico.schottelius.org/docs/a-small-introduction-for-using-gpgme/`
- GPG-crypter: interface
-

SSH

- SSH = Secure Shell
- Gerar chave
  - `ssh-keygen`
- Associar identidade à chave gerada
  - `ssh-add`
- Transmitir identidade
  - `ssh-copy-id git@github.com`
- Realizar conexão
- `ssh -o VisualHostKey=yes ...`

Além da disciplina

- Garantias e análise de segurança
  - Provas de segurança
  - Criptografia com premissas mínimas
- Criptoanálise
  - Criptanálise quântica
  - Algoritmos para fatorar e computar logaritmos discretos
- Desenvolvimento de novos métodos para primitiva existentes
  - Criptografia em hardware
  - Cifras streams eficientes
  - Princípios modernos de projeto para cifras e funções hash
  - Algoritmos baseados em teoria de números
  - Criptografia pós-quântica
- Novas primitivas
  - Computação segura com múltiplos interessados
    - Votação eletrônica
    - Leilões eletrônicos
    - Moedas criptográficas
  - Prova de conhecimento (sem divulgá-lo)
  - Comunicação anônima
  - Computação anônima