Step Into Cryptography

Fundamental

- Digest
- Encryption and Decryption
- Message authentication code
- Authenticated encryption(AEAD)
- Public-key cryptography
- RSA(Rivest–Shamir–Adleman)
- Elliptic curve cryptography(ECC)

Digest

- Hash
- MD5
- SHA1
- SHA2(256/512)
- SHA3

- CRC
- FNV
- Murmur

KDF

- salt
- scrypt
- bcrypt

Attack & Best Practice

- hash collision attack
- rainbow table
- length extension attac k

- Don’t use MD5/SHA1
- Prefer SHA-512/256 or SHA-512/224

Encryption & Decryption

- DES
- AES
- Salsa20(aka chacha20)

- ECB
- CBC
- CFB
- OFB
- CTR

Message Authentication Code

- HMAC
- CMAC

Attack & Best Practice

- KPA/CPA
- RA
- Example: Efail

- Use AEAD
- Timestamp

Authenticated Encryption(AEAD)

- EtM
- E&M
- MtE

- AES-GCM
- Salsa20-Poly1305

Public-key Cryptography

- key exchange
- public key encryption
- public key signature

- DH
- DSA
- RSA
- ECC

Key Exchange

- pa=ka*O
- Get pb
- k=ka*pb
- =ka*(kb*O)
- =(ka*kb)*O

- pb=kb*O
- Get pa
- k=kb*pa
- =kb*(ka*O)
- =(kb*ka)*O
- =(ka*kb)*O

Trapdoor Function

- Abelian group
- y=x*O are easy to calculate
- Reverse function are hard enough(Non-Polynomial)

DH Example

- p=23, g=5
- a=4, pa=5^4mod23=4
- k=10^4mod23=18

- p=23, g=5
- b=3, pb=5^3mod23=10
- k=4^3mod23=18

RSA(Rivest–Shamir–Adleman)

- Generation
- Encryption
- Decryption
- Signing
- Verifying
- Key exchange

RSA Example: Generate

- prime1&prime2
- modulus n=p*q
- φ(n)=(p-1)*(q-1)
- public exponent: chosen a prime coprime with φ(n)
- private exponent: d = e^(φ(n)-1) mod φ(n)

- p=61, q=53
- n=61*53=3233
- φ(n)=60*52=3120
- e=17
- d = 17^(3120-1) mod 3120 = 2753

RSA Example: Encryption & Decryption

- c = (m^e) mod n
- m = (c^d) mod n

- c = 65^17 mod 3233 = 2790
- m = 2790^2753 mod 3233 = 65

RSA Example: Signing & Verifying

- v = (m^d) mod n
- m = (v^e) mod n

- v = 65^2753 mod 3233 = 588
- m = 588^17 mod 3233 = 65

RSA Key Exchange

- RSA
- DH_RSA
- DHE_RSA

Elliptic Curve Cryptography(ECC)

- Generation
- Encryption
- Decryption
- Signing
- Verifying
- Key exchange

- prime field/binary field
- Kx: ECDH
- Enc: ECIES
- Sig: ECDSA
- Sig: EdDSA
- Kx: ECMQV

Attack & Best Practice

- Shor's algorithm
- Time attack
- Random issue

- Use /dev/urandom

Cryptography In OpenSSL

- Speed Test
- Encryption & Decryption
- RSA Key Generate
- RSA Encryption & Decryption
- RSA Signing & Verifying
- ECC Generate
- ECC Signing & Verifying
- ECC Derivation
- DH Generate

Speed Test

- Hash: long block run faster
- Encryption: long block run slower
- Public key algorithm: RSA 2048 ECC 256

Encryption & Decryption

- openssl enc -ciphers
- aes, chacha20
- openssl enc -chacha20 < src > dst
- openssl enc -d -chacha20 < dst
- openssl chacha20 < src > dst
- No AEAD!

RSA Key Generate

- openssl genrsa 2048 > rsa.key
- openssl genrsa 2048 -out rsa.key
- openssl rsa -text -in rsa.key
- openssl rsa -pubout < rsa.key > rsa.pub
- openssl rsa -text -pubin -in rsa.pub
- openssl rsa -aes128 < rsa.key > rsa.enc
- openssl rsa < rsa.enc > rsa.key

RSA Encryption & Decryption

- openssl rsautl -encrypt -pubin -inkey rsa.pub < src > dst
- openssl rsautl -decrypt -inkey rsa.key < dst > src.new

RSA Signing & Verifying

- openssl rsautl -sign -inkey rsa.key < src > dst
- openssl rsautl -verify -pubin -inkey rsa.pub < dst > src.new
- openssl pkeyutl -verify -pubin -inkey rsa.pub -sigfile sig < src
- openssl pkeyutl -verifyrecover -pubin -inkey rsa.pub < sig

ECC Generate

- openssl ecparam -list_curves
- openssl ecparam -genkey -name secp256r1 > ecc.key
- openssl ec -text < ecc.key
- openssl ec -pubout < ecc.key > ecc.pub
- openssl ec -text -pubin < ecc.pub
- openssl pkey -pubout < ecc.key > ecc.pub

ECC Signing & Verifying

- openssl pkeyutl -sign -inkey ecc.key < src > sig
- openssl pkeyutl -verify -pubin -inkey ecc.pub -sigfile sig < src

ECC Derivation

- openssl pkeyutl -derive -inkey ecc1.key -peerkey ecc2.pub -hexdump
- openssl pkeyutl -derive -inkey ecc2.key -peerkey ecc1.pub -hexdump

Quiz

Implement ECIES by OpenSSL

DH Generate

- openssl dhparam -outform PEM -out dhparam.pem 4096
- openssl dhparam -in dhparam.pem -text

Certification In OpenSSL

- Certification
- Certificate Request
- Self-Signed Certification
- Alternative Domain
- Certification Format
- PEM to DER
- PKCS12
- Ciphers
- Ciphers Best Practice
- Harden TLS
- Self-Built CA
- EasyRSA Setup
- EasyRSA Sign
- EasyRSA Sign With Request
- EasyRSA Revoke

Why Certification

- Forward secrecy
- Man-in-the-middle attack
- O(n^2) issue
- HA/performance

- DH
- Mac+DH
- Authentication Center
- Public key algorithm
- Sign chain + pre-installed public key

Certification

- Certification Authority
- Intermediate
- Certification

- DV
- OV
- EV

Wildcard are prohibited in EV certification.

Certificate Request

- openssl req -new -key rsa.key -out shell.csr
- openssl req -text -in shell.csr -noout
- openssl x509 -x509toreq -in shell.crt -out shell_new.csr -signkey rsa.key

Self-Signed Certification

- openssl x509 -req -days 3650 -in shell.csr -signkey rsa.key -out shell.crt
- openssl req -new -x509 -days 365 -key rsa.key -out shell.crt
- openssl req -new -x509 -days 365 -key rsa.key -out shell.crt -subj '/C=CN/L=SH/O=home/CN=*.shell.org/emailAddress=shell@shell.org'

Alternative Domain

- shell.ext: subjectAltName = DNS:*.facebook.com, DNS:facebook.com
- openssl x509 -req -days 365 -in shell.csr -signkey rsa.key -out shell.crt -extfile shell.ext
- openssl x509 -in shell.crt -text -noout

Certification Format

- DER certification: binary, ASN.1
- PEM certification: plain text, base64
- DER private key
- PEM private key
- PKCS7: RFC2315
- PKCS12

PEM to DER

- openssl x509 -inform PEM -in shell.crt -outform DER -out shell.der

PKCS12

- openssl pkcs12 -export -name shell -out shell.p12 -inkey rsa.key -in shell.crt -certfile shell.crt
- openssl pkcs12 -in shell.p12 -out shell.pem -nodes
- openssl pkcs12 -in shell.p12 -nocerts -out shell.key -nodes
- openssl pkcs12 -in shell.p12 -nokeys -clcerts -out shell.crt
- openssl pkcs12 -in shell.p12 -nokeys -cacerts -out shell-ca.crt

Ciphers

- openssl ciphers -v

Ciphers Best Practice

- In short: EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH
- Full: EECDH+AESGCM:EDH+AESGCM:ECDHE-RSA-AES128-GCM-SHA256:AES256+EECDH:DHE-RSA-AES128-GCM-SHA256:AES256+EDH:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES256-GCM-SHA384:AES128-GCM-SHA256:AES256-SHA256:AES128-SHA256:AES256-SHA:AES128-SHA:DES-CBC3-SHA:HIGH:!aNULL:!eNULL:!EXPORT:!DES:!MD5:!PSK:!RC4
- With chacha20: EECDH+CHACHA20:EDH+CHACHA20:EECDH+AESGCM:EDH+AESGCM:EECDH+AES256:EDH+AES256

Harden TLS

- Keep OpenSSL up to date.
- ssl_prefer_server_ciphers on;
- add_header Strict-Transport-Security "max-age=31536000; includeSubDomains";
- openssl dhparam -out dhparam.pem 4096
- ssl_dhparam /etc/ssl/certs/dhparam.pem;

Self-Built CA

- Do it with esay-rsa

EasyRSA Setup

- edit vars
- ln -s openssl-1.0.0.cnf openssl.cnf
- ./clean-all
- ./build-ca

EasyRSA Sign

- ./build-key common_name

EasyRSA Sign With Request

- ./build-req cn
- ./sign-req cn

EasyRSA Revoke

- ./revoke-full cn
- ./list-crl

Thanks
Q&A