The University of Newcastle
School of Information and Physical Sciences

SENG2250/6250 System and Network Security
Week 5 Lab

Part 1: Digital Signature Algorithm (DSA)

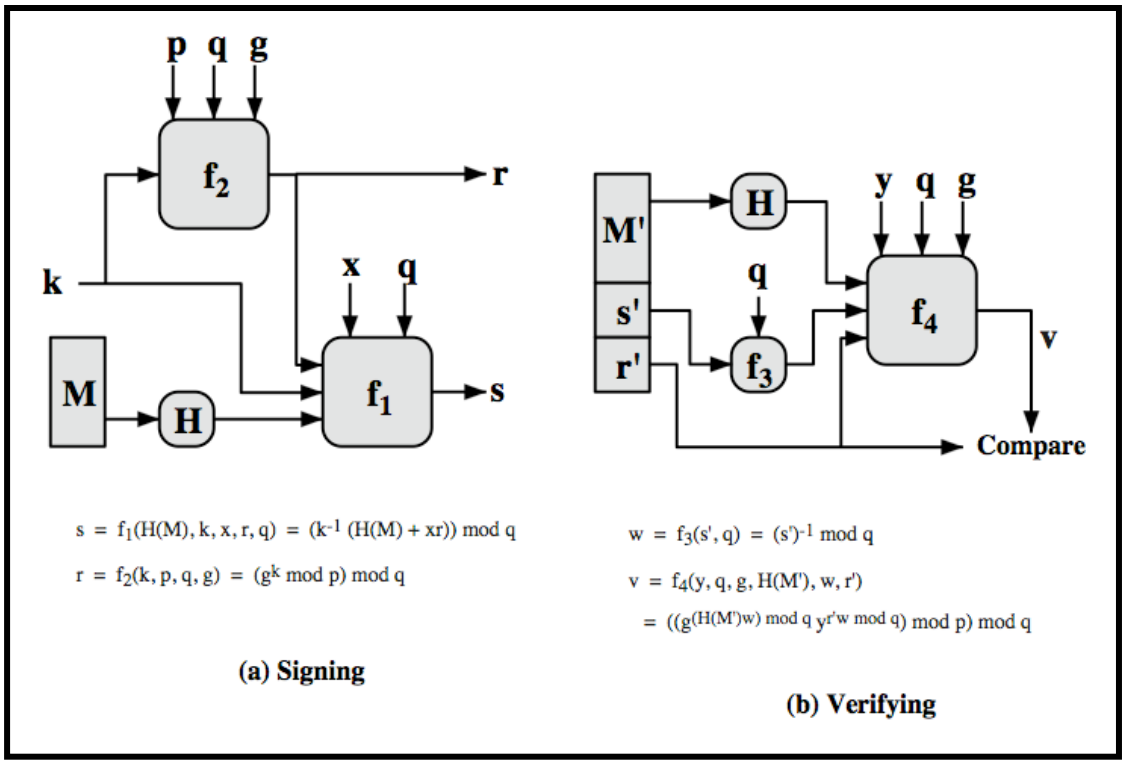


Figure: Digital Signature Algorithm

The Digital Signature Algorithm (DSA) is a Federal Information Processing Standard for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in 1991 and is based on the mathematical concept of modular exponentiation and the discrete logarithm problem.

Here's how DSA works:

Step 1. Key Generation

1.1 Generate Parameters (p, q, g):

a. choose 160-bit prime number q
b. choose a large prime p with $2^{L-1} < p < 2^L$
   - where L is the bit length of p, a multiple of 64 between 512 and 1024 inclusive in the original DSA, and now typically 2048 or 3072
   - such that q is a 160-bit prime divisor of (p-1)
c. choose $g = h^{(p-1)/q}$

- where 1 < h < p-1 and $h^{(p-1)/q}$ mod p > 1

1.2 Generate Keys (x,y):

a. choose random private key: 0 < x < q
b. compute public key: y = g^x mod p

Step 2. Signature Generation (Sender to sign a message M)

2.1 Generate a random signature key k, k < q

2.2 Computes signature pair

a. r = (g^k mod p)mod q
b. s = [k^{-1} ($H(M) + xr$)] mod q

2.3 Sends signature (r, s) with message M

Step 3. Signature Verification (Recipient receives message M and signature (r,s))

3.1 Compute

a. $w = s^{-1}$mod q
b. u_1= [$H(M)w$]mod q
c. u_2= (rw)mod q
d. v = [(g^{u_1} y^{u_2})mod p]mod q

3.2 Verify

a. If $v = r$ then signature is verified

Consider a DSA system where global parameters are p = 11, q = 5 and

g = $h^{(p-1)/q}$ = $2^{(11-1)/5}$ = 4. **Alice chooses a private key $x = 2$.**

Where, h is randomly chosen from {2, …., $p - 2$} and x from {1, …, q-1}.

Q1: Calculate Alice's public key.

Q2: To sign a message M, Alice generates a random number $k = 3$ and hashes her message to get H(M)=3. Calculate the signature pair that Alice adds to her message.

Q3: Verify the authenticity of the message/signature for Bob.

PART 2: Diffie-Hellman Key Agreement

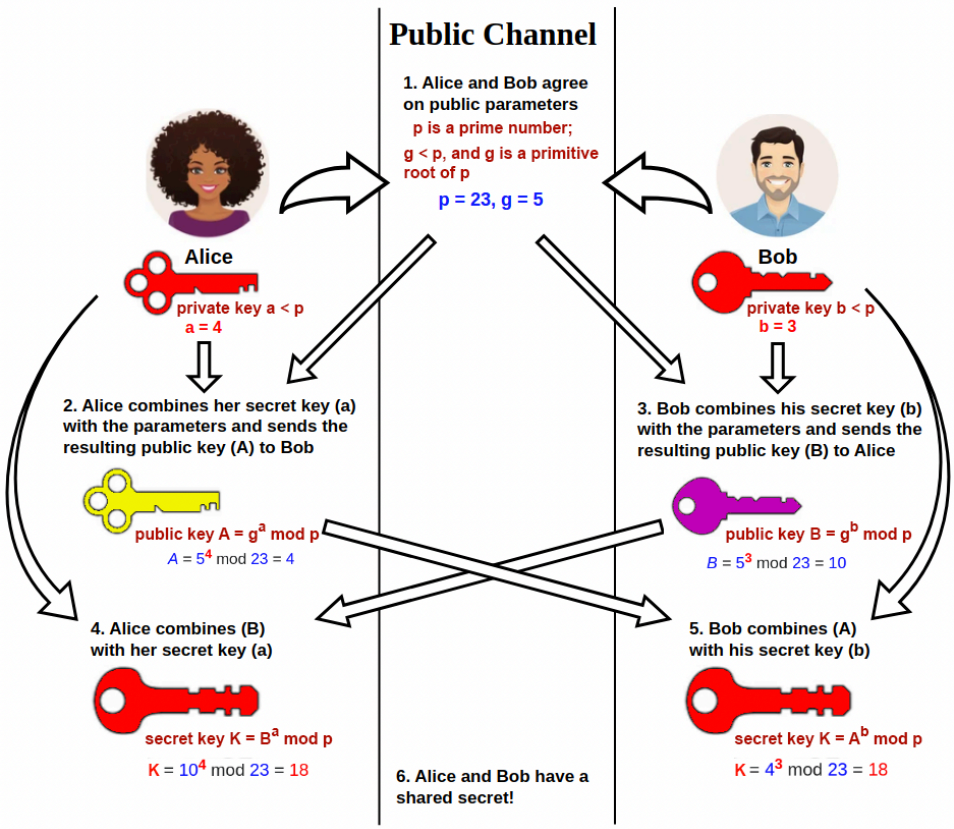
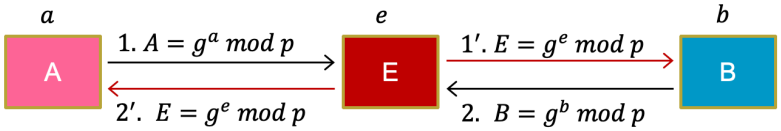


Figure: Diffie-Hellman Key Agreement

Q1: Is 2 a primitive root of 11?

Q2: Let p = 11 and g= 2, compute the public, private and shared secret keys for the communicating parties Alice (A) and Bob (B) based on Diffie-Hellman Key Agreement.



3. $K_{ab} = E^a = g^{ea} \ mod \ p$ 3. $K_{ba} = E^b = g^{eb} \ mod \ p$

Q3: Demonstrate how Man-in-the-Middle Attack may happen with data given for Q2.

PART 3: PKI (X.509) Certificate

A Public Key Infrastructure (PKI) X.509 certificate is a fundamental component of modern digital security, providing a reliable mechanism for authenticating and securing online communications. This certificate, based on the X.509 standard, serves as a digital credential that verifies the identity of an entity, such as a person, organization, or device, in the digital realm. It contains essential information, including the entity's public key, their distinguished name, a digital signature from a trusted Certificate Authority (CA), and a validity period.

The X.509 certificate forms the basis of secure communication protocols like Transport Layer Security (TLS) and is pivotal in enabling encrypted connections between clients and

servers. By utilizing asymmetric cryptography, where different keys are used for encryption and decryption, X.509 certificates ensure the confidentiality, integrity, and authenticity of data exchanged over networks, contributing significantly to the establishment of trust and the prevention of unauthorized access in today's interconnected digital landscape.

Secure Sockets Layer (SSL) is a standardized security technology employed to create an encrypted connection between a server and a client as illustrated in the following figure "SSL Handshake". This connection is commonly established between entities such as a web server (website) and a browser, or a mail server and a mail client (for instance, Outlook).

Open the Chrome browser on your desktop/laptop and enter the following in the address bar: chrome://settings/security. Next, navigate to '*Advanced*' and click on *'Manage certificates*'. Then, click on '*Advanced*', choose '*Export format*' as '*PKCS #7 Certificates*', and ensure that '*Includes all certificates in the certification path*' is checked before clicking '*Ok*'. Within the tab, select '*Trusted Root Certification Authorities*', choose any certificate from the list, and '*View*' the certificate. Click on '*Details*' and proceed to answer the following questions.

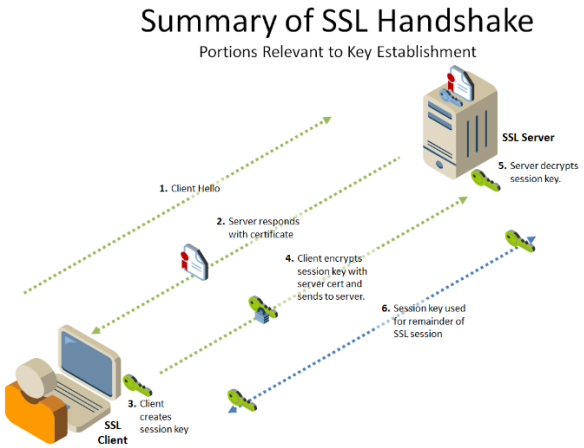


Figure: SSL Handshake

Q1: What is the version number?

Q2: What signature algorithm has been used?

Q3: Who is the issuer? Can we trust them?

Q4: What is the certification period?

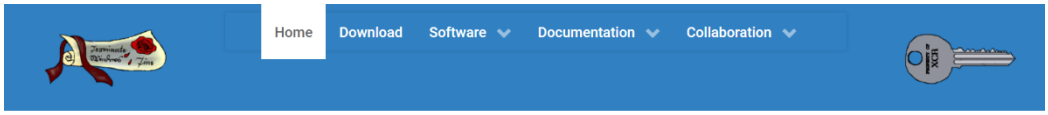
Q5: What is the public key?

Q6: What public key cryptographic algorithm has been sued?

Q7: What are the different uses of this certificate?

PART 4: X Certificate and Key management

Download and install XCA from this website https://hohnstaedt.de/xca/ in your Desktop/Laptop. XCA is intended for creating and managing X.509 certificates, certificate requests, RSA, DSA and EC private keys, Smartcards and CRLs. Everything that is needed for a CA is implemented. All CAs can sign sub-CAs recursively. These certificate chains are shown clearly. For an easy company-wide use there are customizable templates that can be used for certificate or request generation. All cryptographic data is stored in a SQL database – supported are SQLite (Single file), MySQL (MariaDB), PostgreSQL, and Microsoft SQL-Server (via ODBC).

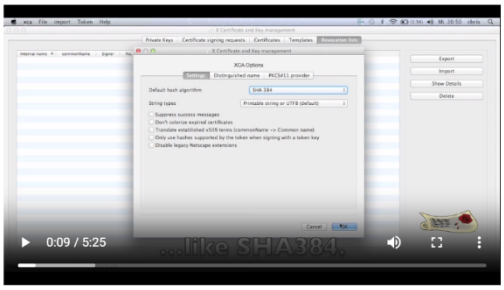


X - Certificate and Key management

This application is intended for creating and managing X.509 certificates, certificate requests, RSA, DSA and EC private keys, Smartcards and CRLs.
Everything that is needed for a CA is implemented.
All CAs can sign sub-CAs recursively. These certificate chains are shown clearly.
For an easy company-wide use there are customiseable templates that can be used for certificate or request generation.

All cryptographic data is stored in a SQL database. Supported are

- SQLite (Single file)
- MySQL (MariaDB)
- PostgreSQL
- Microsoft SQL-Server (via ODBC)



Features

- Start your own PKI and create all kinds of private keys, certificates, requests or CRLs
- Import and export them in any format like PEM, DER, PKCS#7, PKCS#12
- Use them for your IPsec, OpenVPN, TLS or any other certificate-based setup
- Manage your Smart-Cards via PKCS#11 interface
- Export certificates and requests as OpenSSL config file
- Create Subject- and/or Extension- templates to ease issuing similar certs
- Convert existing certificates or requests to templates
- Get the broad support of x509v3 extensions as flexible as OpenSSL but user friendlier
- Adapt the columns to have your important information at a glance

Standards

- PKCS#1 unencrypted RSA key storage format
- PKCS#7 Collection of public certificates
- PKCS#8 Encrypted private key format for RSA DSA EC keys
- PKCS#10 Certificate signing request
- PKCS#11 Security token / Smart card / HSM access
- PKCS#12 Certificate, Private key and probably a CA chain

Homework: Explore XCA for implementing a CA for an organization.