Programming with cryptography

Chapter 11: Building Secure Software

Lars-Helge Netland

larshn@ii.uib.no

10.10.2005

INF329: Utvikling av sikre applikasjoner

Overview

- Intro: The importance of cryptography
- The fundamental theorem of cryptography
- Available cryptographic libraries
- Java code examples
- One-time pads
- Summary

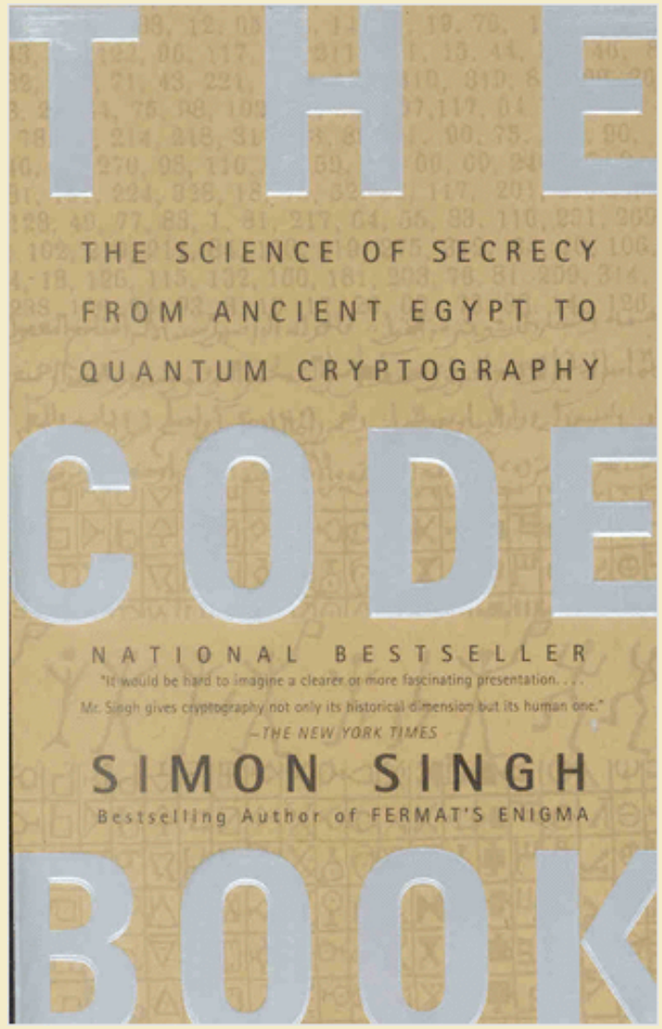
Intro

- Cryptography
  - ‘kryptos’(=hidden)
  - ‘graphein’ (=to write)
- Layman description: the science of encrypting and decrypting text

The Code Book

- Simon Singh
- Highly accessible introduction to cryptography
- Retails at $10

  (free at public libraries)
- Includes crypto challenges



Crypto: Why is it important?(I)

- Basic services:
  - confidentiality
    - encrypted text only viewable by the intended recipient
  - integrity
    - assurance to the involved parties that the text was not tampered with in transit
  - authentication
    - proof of a binding of an identity to an identification tag (often involves trusted third parties)

Crypto: Why is it important?(II)

- Basic services (continued)
  - authorization
    - restricting access to resources (often involves authentication)
  - digital signatures
    - electronic equivalent of hand-written signatures
    - non-repudiation

The fundamental theorem of cryptography

- Do NOT develop your own cryptographic primitives or protocols
- The history of cryptography repeatedly shows how "secure" systems have been broken due to unknown vulnerabilities in the design
- If you really have to: PUBLISH the algorithm

Examples on how ciphers have been broken

- Early substitution ciphers broken by exploiting the relative frequency of letters
- WW2: The enigma machine
- SHA-1: Wang

Overview of cryptographic libraries

- Cryptlib
- OpenSSL
- Crypto++
- BSAFE
- Cryptix
- Bouncy Castle

Cryptlib

- http://www.cs.auckland.ac.nz/~pgut001/cryptlib/

- Very easy to use
- Free for non-commercial use
- Extensively documented, robust, well-written and highly efficient

OpenSSL

- http://www.openssl.org
- Free software license
- Serious lack of documentation
- Not for the faint of heart: hard to use

Crypto++

- http://www.eskimo.com/~weidai/cryptlib.html
- Includes all the significant cryptographic algorithms
- Almost completely undocumented
- Use requires a good understanding of C++
- Free for any use

BSAFE

- http://www.rsasecurity.com
- Most widely deployed commercial library available
- Pricing information not available
- Extremely efficient and of high quality

Cryptix

- http://www.cryptix.org
- Cryptographic package for Java
- Notoriously difficult to use
- Inadequate documentation and highly complex to use
- Free sofware

Bouncy Castle

- http://www.bouncycastle.org
- nice crypto extensions to Java
- very poorly documented
- lightweight API for J2ME; cell phones
- free!
- efficiency??

Cryptography in Java (I)

- Cryptographic services are realized through engine classes
  - an engine class is an abstraction of a particular cryptographic concept
  - e.g., the `signature` engine encapsulates functionality to manage digital signatures
  - each engine has a corresponding Java class with the same name

Cryptography in Java (II)

- Engines are instantiated through its static `getInstance(String algorithm)` method
- Engines can be associated with numerous algorithms
  - e.g. signature algorithms: SHA1withRSA, ECDSA, MD5withRSA

Cryptography in Java (III)

- The Java Cryptography Architecture (JCA) was designed around 2 principles
  - algorithm independence
  - implementation independence
- Creating a message digest:
  - `MessageDigest md = MessageDigest.getInstance("SHA-1")`
  - `security.provider.<n>=<classname>`
  - Mapping contained in the `lib/security/java.security` file

Cryptographic Engines

Signature	Provides the functionality required to create a digital signature, which in turn can be used for authentication or data assurance
MessageDigest	Enables message digests, which are secure one-way hash functions that take an arbitrary length input and produce a fixed length output
SecureRandom	Provides a cryptographically strong Random Number Generator (RNG)
KeyGenerator	Used for generation of symmetric keys
KeyPairGenerator	Used to generate asymmetric key pairs, i.e. public and private key pairs
Cipher	Provides the functionality to create cryptographic ciphers that can encrypt and decrypt data
SSLContext	Used to specify and retrieve SSL implementations
MAC	Provides the functionality of a Message Authentication Code (MAC)

Encryption and decryption (I)

- Symmetric key generation
  - ```
    private SecretKey generateKey () {

        KeyGenerator kg = KeyGenerator.getInstance
                                ("DESede");

        return kg.generateKey(); }
    ```
- Encryption
  - ```
    private byte[] encrypt(byte[] msg) {

        cipher.init(Cipher.ENCRYPT_MODE, secretKey);

        return cipher.doFinal(msg); }
    ```

Encryption and decryption (II)

- Decryption
  - ```
    private byte[] decrypt(byte[] msg) {

      cipher.init(Cipher.DECRYPT_MODE,secretKey);

      return cipher.doFinal(msg);

    }
    ```
    ```

Public Key Cryptography (I)

- Key generation
  - ```
    private KeyPair generateKeyPair () {

        KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");

        return kpg.generateKeyPair(); }
    ```

- Signature
  - ```
    private byte[] sign(byte[] msg) {

        Signature signature = Signature.getInstance("SHA1withRSA");

        signature.initSign(privKey);

        signature.update(msg);

        return signature.sign();}
    ```

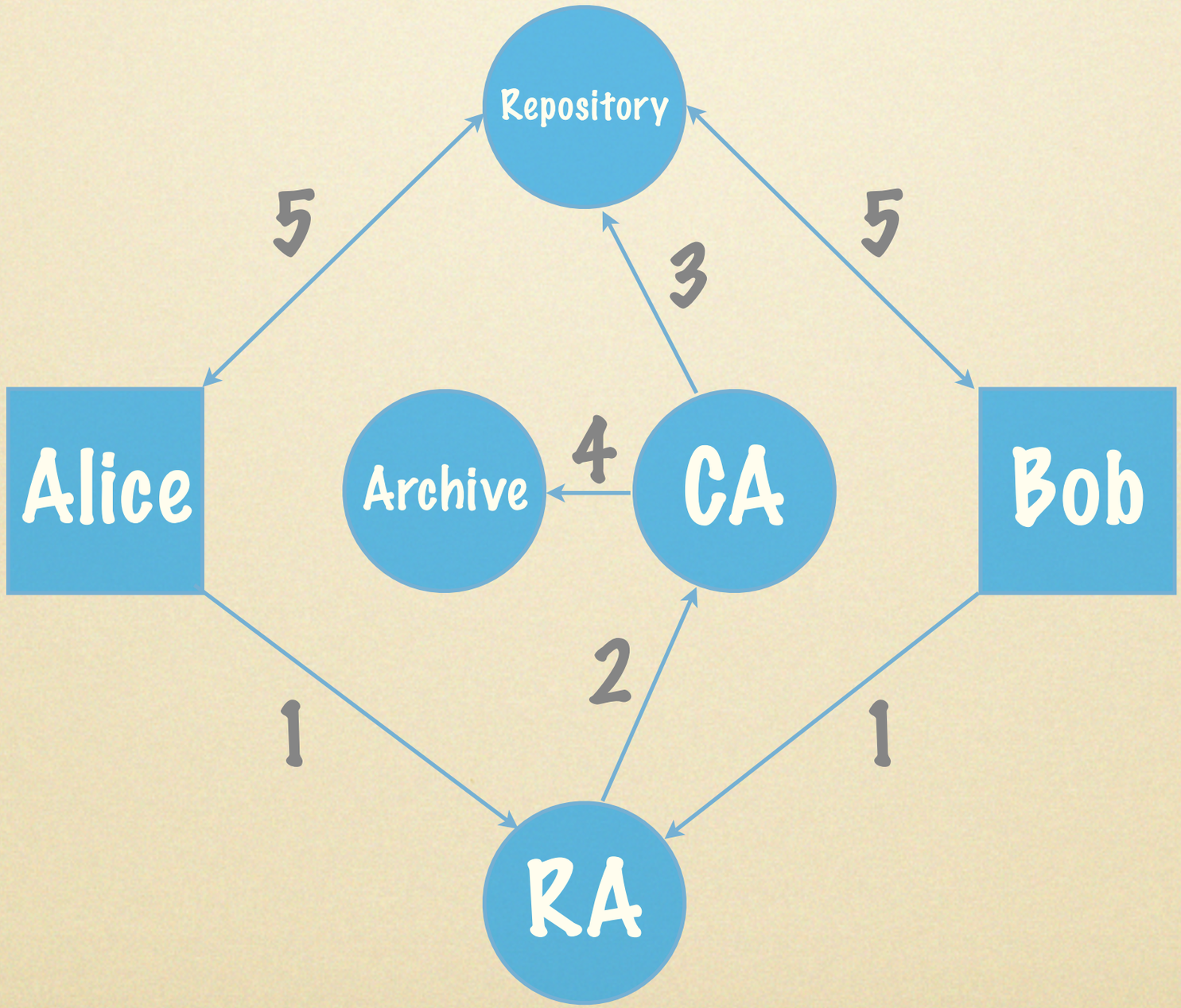
Public Key Cryptography (II)

- Signature verification
- ```
  private boolean verify (byte[] msg, byte[]
                          signature) {
  Signature sign = Signature.getInstance
                      ("SHA1withRSA");
  sign.initVerify(pubKey);
  sign.update(msg);
  return sign.verify(signature); }
  ```
  ```

Public Key Infrastructure (PKI)

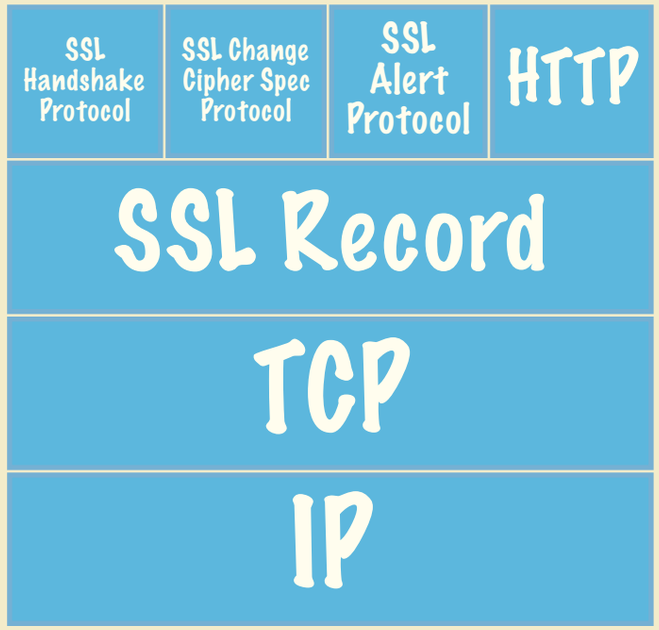
- Certificates
  - identity, public key, issuer, validity window, issuer's digital signature
- Certification Authority (CA)
- Registration Authority (RA)
- Repository
- Archive

PKI entities

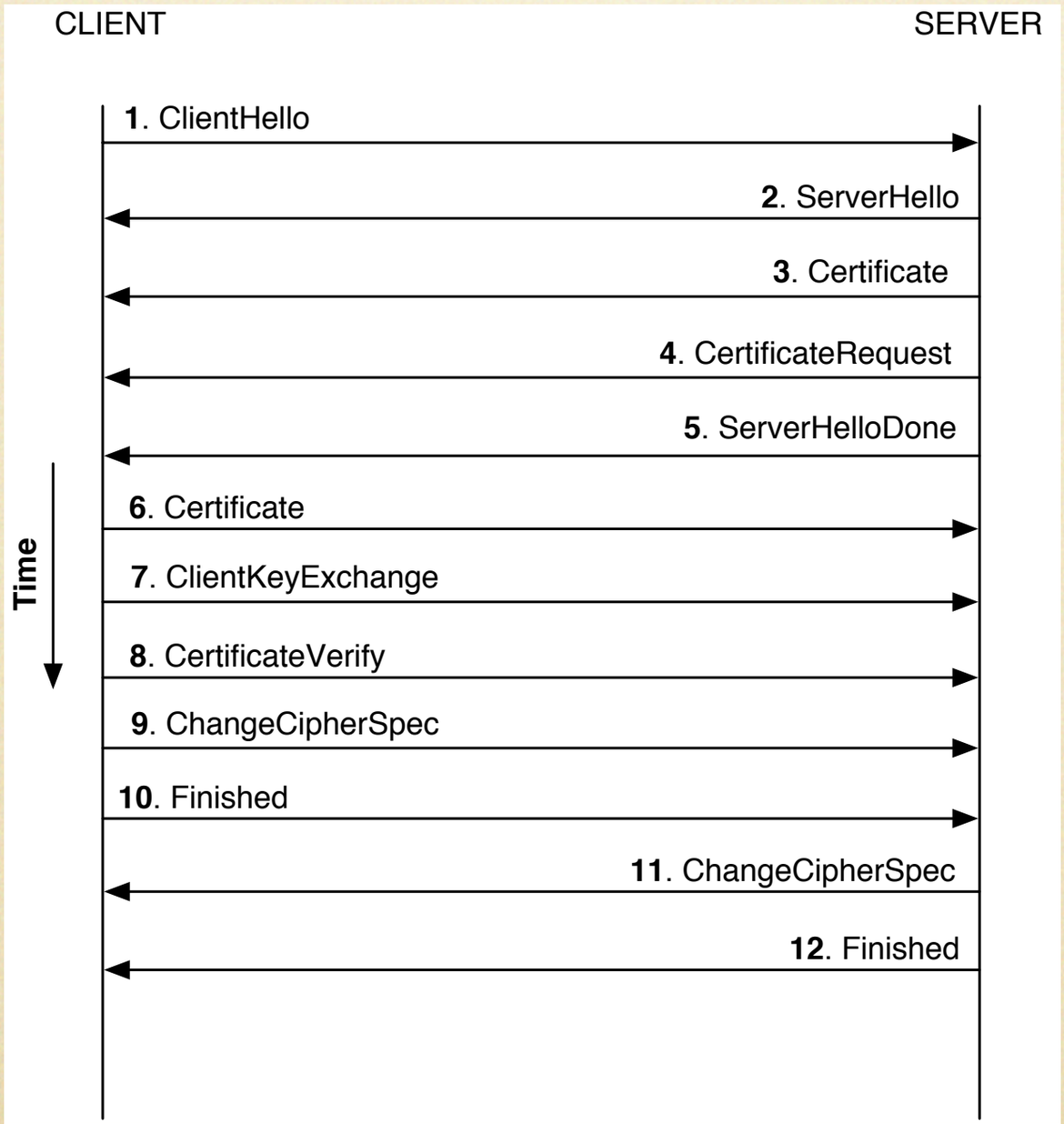


SSL

- Motivation: bring e-commerce to the Internet
- Originated by Netscape in 1994



SSL Handshake Protocol



SSL in Java

- `javax.net.ssl`
- Important classes: SSLContext, SSLEngine, SSLSocket, SSLServerSocket
- Setting up SSL requires the following steps:
  - initialize key and trust material
  - initialize SSLContext
  - create SSLServerSocketFactory & SSLSocketFactory
  - create SSLServerSocket & SSLSocket

Initialize key and trust material (I)

- Use OpenSSL to generate CA and certificates
- locate the script CA.sh in your OpenSSL installation (default: `/usr/local/ssl/misc`)
- Invoke
  - `./CA.sh -newca` to create a CA
  - `./CA.sh -newreq` to generate a certificate request
  - `./CA.sh -sign` to issue a certificate
- The generated certificates can be read into Java using the `CertificateFactory` class.

Initialize SSLContext

- SSLContext sslCon = SSLContext.getInstance(“TLS”);

  KeyManagerFactory kmf = KeyManagerFactory.getInstance
  (“SunX509”);

  KeyStore ks = KeyStore.getInstance(“JKS”);

  char[] passwd = “changeit”.toCharArray();

  ks.load(new FileInputStream(“filename”), null);

  kmf.init(ks, passwd);

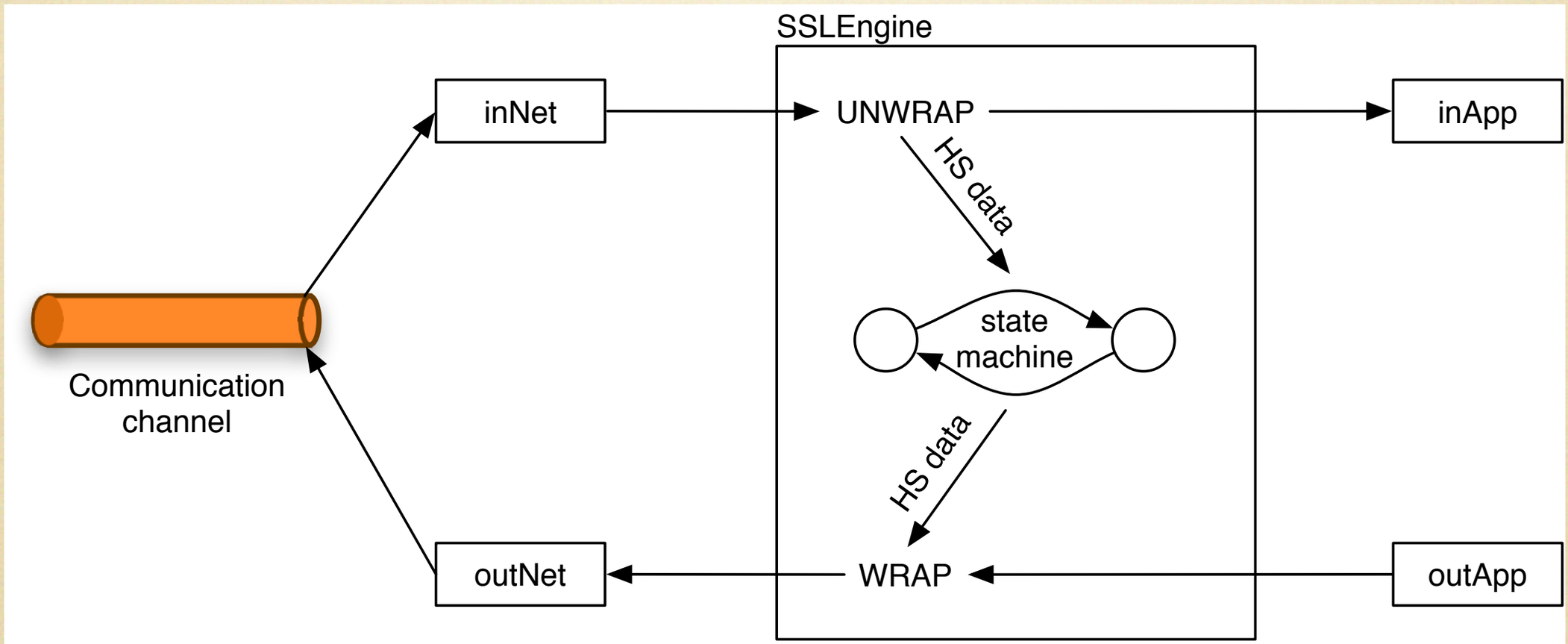
  sslCon.init(kmf.getKeyManagers(), null, null);

Create ServerSocketFactory & ServerSocket

- ```
  ServerSocketFactory ssf =
  sslCon.getServerSocketFactory();

  ServerSocket ss =
  ssf.createServerSocket(8080);
  ```
  ```

SSLEngine


SSLEngine
inNet
UNWRAP
inApp
HS data
state
machine
HS data
Communication
channel
outNet
WRAP
outApp

One-time pad

- Perfect data secrecy!
- Ciphertext = plaintext XOR key
- Plaintext = ciphertext XOR key
- The length of the plaintext and key must be equal

One-time pad (II)

- difficult to distribute keys securely
- requires numbers that absolutely cannot be guessed
- if you must:

I. Get high-quality random output

II. Store those bits on two CDs, both identical

III. Securely distribute and use the CDs

IV. Make sure new CDs are available when more data are needed

V. Destroy the CDs when done

Summary

- `if(!"The Code Book") { enjoy!  }`
- Do not invent your own cryptographic primitives or protocols!
- Use existing crypto libraries
- Java contains a rich and powerful crypto API