Cryptography & The JCE

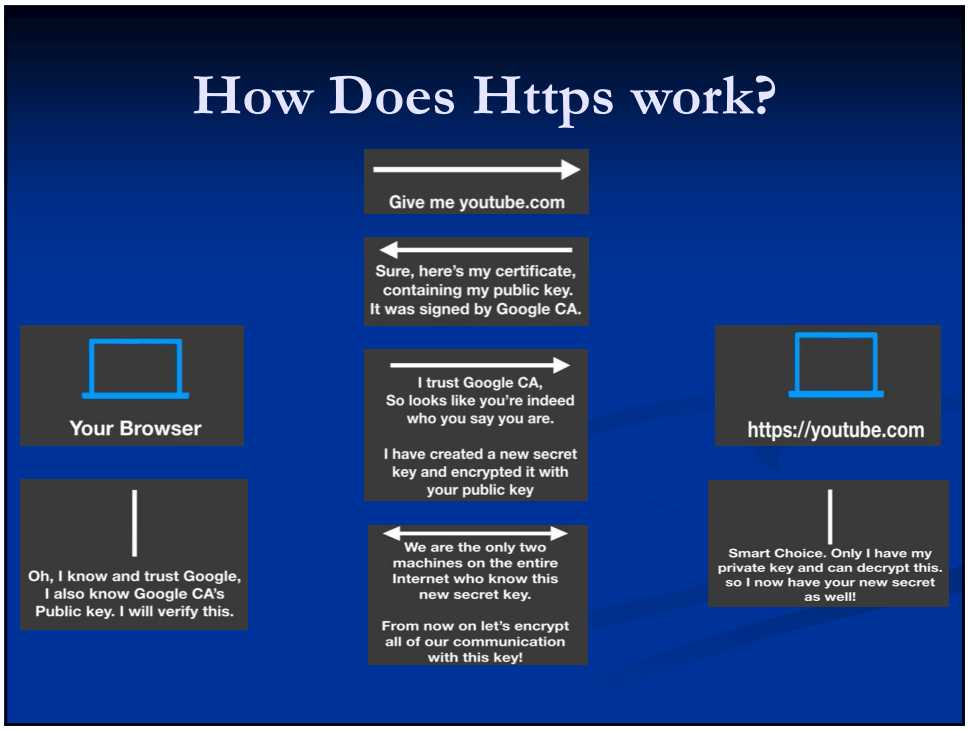
Ameya Joshi

1

Cryptography

The science of securing information.

2


How Does Https work?
Give me youtube.com
Sure, here's my certificate, containing my public key. It was signed by Google CA.
Your Browser
https://youtube.com
I trust Google CA, So looks like you're indeed who you say you are.
I have created a new secret key and encrypted it with your public key
Oh, I know and trust Google, I also know Google CA's Public key. I will verify this.
We are the only two machines on the entire Internet who know this new secret key.
From now on let's encrypt all of our communication with this key!
Smart Choice. Only I have my private key and can decrypt this. so I now have your new secret as well!

3


What is Certificate Authority?
How is a certificate signed?
I have decided, I want to use HTTPS from now on.
I have created a Certificate Signing Request with my key pair. Can you please sign it?
youtube.com web server
Google CA
Sure. I have signed your request with my private key.
Anyone who has my public key can verify that it was actually me who signed it.
Private Key
Public Key
Private Key
Public Key
A Browser

4

Cryptography

- Definitions
- The Setup
- Symmetric Systems
- Hash Functions
- Message Authentication Codes (MAC)
- Asymmetric Systems
- Hybrid Systems
- Electronic Signatures

5

Definitions

- Secret Key – shared piece of secret information used to protect a larger set of data.
- Encrypt –scramble data with a secret key into a hard-to-understand format.
- Decrypt – scramble encrypted data into readable using a secret key.
- Cryptographic algorithm – Description of how a secret key is utilized to scramble information.

6

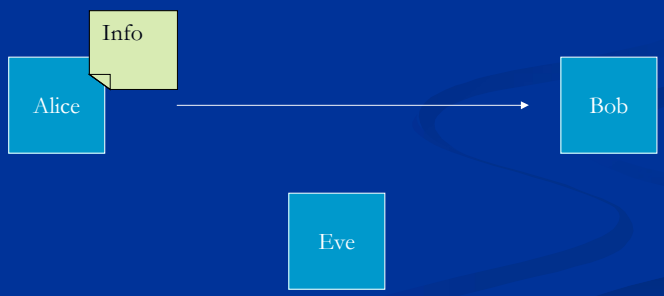
Definitions cont'd...

- Plaintext (aka Cleartext) – The information to be secured.
- Ciphertext – The scrambled/unreadable information after an encryption process is performed.

7

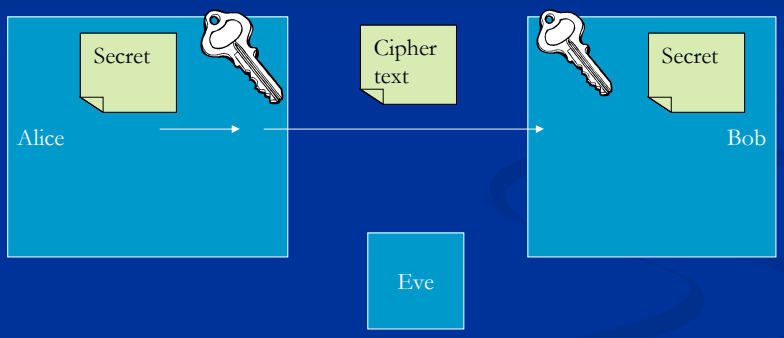
The Setup

1. Alice wants to securely send Bob a secret
2. Bob wants to be sure information came from Alice



8

Symmetric Cryptography



9

Security provider architecture

```
import java.util.*;
import java.security.*;

public class ExamineSecurity {

  public static void main(String args[]) {
    try {
      Provider p[] = Security.getProviders();
      for (int i=0; i < p.length; i++) {
        System.out.println(p[i]);
        for (Enumeration e = p[i].keys(); e.hasMoreElements();)
          System.out.println("\t"+e.nextElement());
      }
    } catch (Exception e) {
      System.out.println(e);
    }
  }
}
```

10

Symmetric Algorithms

- Substitution and transposition using a secret key to obscure the plaintext into ciphertext.
- Fast to implement in software and hardware
- Problem: Secret key used for encryption and decryption must be known.
- Examples: RC5, DES, 3DES, Blowfish, AES

11

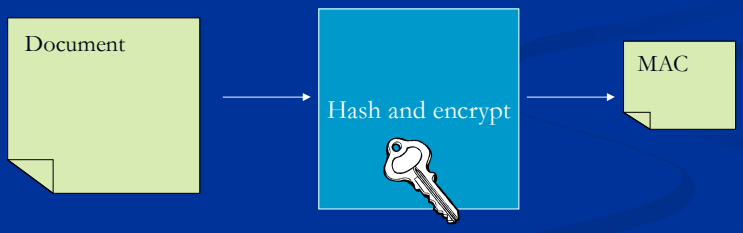
Hash Functions

- One way operation on information that results in smaller set of data, called a *message digest*.
- MD5 and SHA-1 are hash functions.
- Considered secure when it is computationally infeasible to find two input data with the same message digest.
- Secure hash functions are used in electronic signatures.

12

MACs

- Message Authentication Codes provide an authentication scheme in symmetric-based cryptographic protocols.



13

MACs cont'd...

- Produces an encrypted message digest with a secret key.
- Alice sends Bob a document as well as a MAC. Bob can authenticate who sent the document by performing the same MAC on the document and comparing his MAC to the one that Alice sent. If they match, he knows that Alice sent the document.
- Problem: Secret key must be established and known only to Alice and Bob.

14

Example: message digests

```
import java.io.*;
import java.security.*;

public class Send {

  public static void main(String args[]) {
    try {
      FileOutputStream fos = new FileOutputStream("test");
      MessageDigest md = MessageDigest.getInstance("SHA");
      ObjectOutputStream oos = new ObjectOutputStream(fos);
      String data = "Martins message";
      byte buf[] = data.getBytes();
      md.update(buf);
      oos.writeObject(data);
      oos.writeObject(md.digest());
    } catch (Exception e) {
      System.out.println(e);
    }
  }
}
```

- use class to get instance of algorithm

15

Asymmetric Cryptography

- Utilizes two keys: One private to an individual, and another public to the world.
- An individual shares his public key to a Trusted Third Party (TTP)
- Alice can securely send Bob information by encrypting it with Bob's public key retrieved from the TTP. Only Bob's private key will decrypt the information.
- Useful for establishing secure channels in an insecure environment: PGP & SSL.
- Examples: RSA, ElGamal, and ECC

16

Asymmetric Cryptography cont'd…

- Based on 'hard' math problems
- Sharing public keys require a public-key infrastructure (PKI) – retrieving, adding and revoking keys
- Trust is paramount
- Asymmetric keys must be much larger than symmetric keys

17

Hybrid Systems

- Asymmetric cryptosystems are used for establishing secure channels
- With an established secure channel, Alice can exchange a symmetric secret key with Bob and engage in a secure conversation using a symmetric cipher.

18

Electronic Signatures

- Alice can sign a document by using her private key. Bob can authenticate her signature by using her public key.
- Alice signs a document by first hashing it using a secure hash function (SHA-1).
- The Digital Signature Standard (DSS) is a standard means of signing documents

19

Java Cryptography Extension

- JCE bundled with the SDK in 2002.
- Subject to US export restrictions.
- Built on top of java.security and javax.crypto
- The JCE is a pluggable technology – allowing different implementations from many providers.
- Useful classes are:
  - SecretKeyFactory
  - Cipher
  - SealedObject
  - KeyGenerator
  - KeyAgreement
  - Mac
  - SecureRandom

20

JCE Providers

- Open source providers are Cryptix and Bouncy Castle.
- Plugging-in
  - modifying java.security file.
  - Use code to add a provider

Example:

```
import cryptix.jce.provider.CryptixCrypto;
Provider cryptix_provider = new CryptixCrypto();
int result=Security.addProvider(cryptix_provider);
```

21

JCE - SecretKeyFactory

- Generates SecretKey instances for use with a symmetric cipher.
- Useful when the secret key has already been established.
- Supported SecretKey instances are dependent on the ones offered by the installed JCE providers.
- Example:

```
byte[] secretKey = "SecrtKey".getBytes();
DESKeySpec desKeySpec = new DESKeySpec( secretKey );
SecretKeyFactory factory = SecretKeyFactory.getInstance("DES");
SecretKey sk = factory.generateSecret( desKeySpec );
```

22

JCE – Cipher

- Cipher does the work of encryption and decryption
- A Cipher is instantiated using the Cipher.getInstance factory method
- Associated with a transformation name in the format, *algorithm/mode/padding*
- Can operate within four modes: encrypt, decrypt, key wrap, key unwrap.
- Must be initialized using a specified mode, and secret key information.
- Example:

```
Cipher c = Cipher.getInstance("DES");
c.init( Cipher.ENCRYPT_MODE, secretKey );
byte[] plaintext = "The time has come for action.".getBytes();
byte[] ciphertext = c.doFinal ( plaintext );
```

23

JCE - SealedObject

- Great for securely persisting objects which can be serialized.
- Instantiated with a Cipher object and a serializeable object.
- Any algorithm parameters used by the Cipher object are stored in the SealedObject for easy decryption.
- Unsealing requires either the same Cipher object used for sealing or the associated secret key.

24

JCE - KeyGenerator

- The KeyGenerator class solves the problem of Alice or Bob having to come up with their own secret key. It will create one for them.
- Symmetric algorithms have their own specific weak keys. Users who use weak keys open their communication to known exploits. For example, a weak key for DES is:

  0000000 FFFFFFF
- Uses a random number generator, a key size, and a target cryptographic algorithm (like 'DES') to generate an acceptable key for the developer.
- Example:

```
KeyGenerator kg = KeyGenerator.getInstance("DES");
kg.init(56);
SecretKey sk = kg.generateKey();
```

25

Java support for cryptography

- Keys
- Certificates
- Key management
- Message digests
- Secure message digests
- Digital signatures
- Encryption & decryption

26

Keys & certificates: recap

- Two kinds of keys:
  - secret (symmetric)
  - public/private (asymmetric)
- Certificates can be used to authenticate public keys:
  - Public keys usually transmitted as part of a certificate

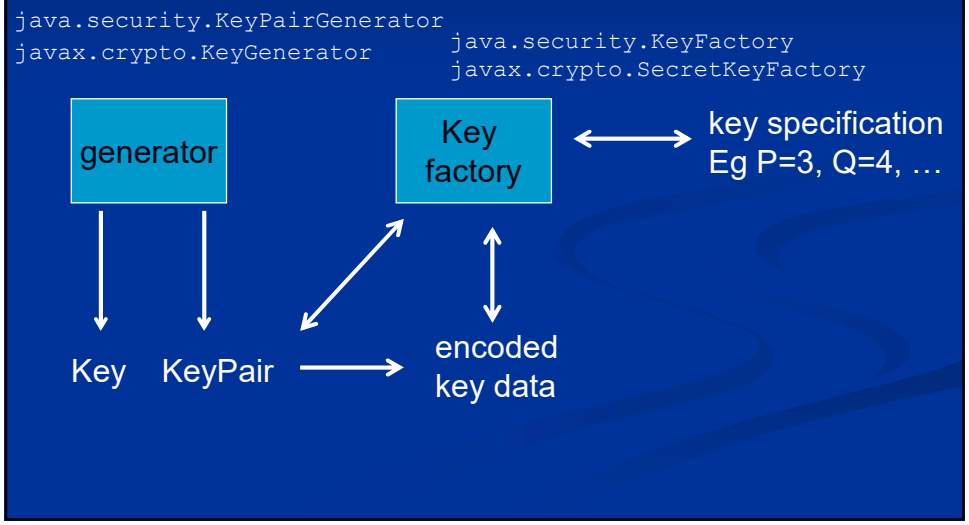
27

Issues

- Key management and storage
- Self-certification?
- Hierarchy of trust

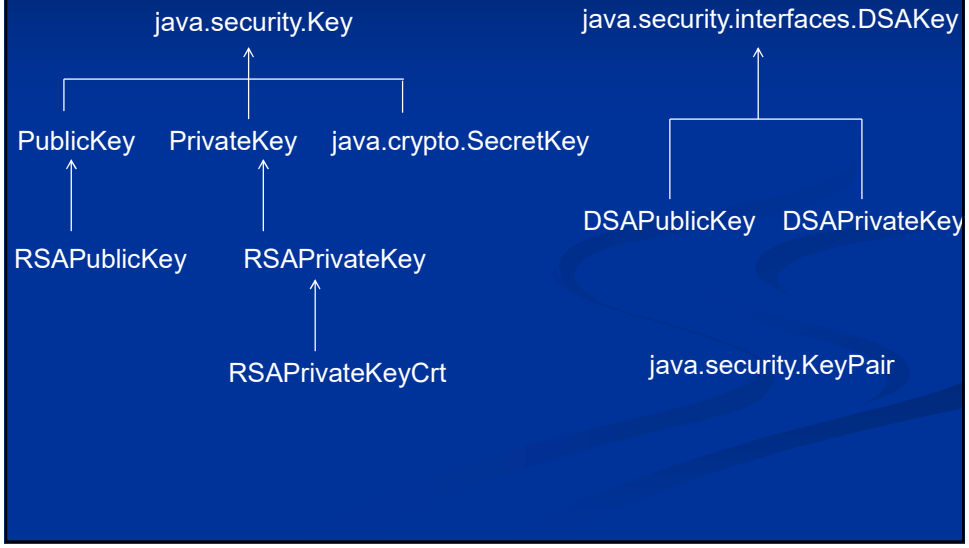
28

Generation and import/export of keys



29

The Key class hierarchies: a partial view



30

Why so many?

- Certain algorithms require methods to access key generation parameters for export
  - DSAKey: methods getP(), getQ(), getG()
- Certain algorithms have specific roles
  - DHKey: Diffie-Hellman key exchange

31

Example: generate/export key pair

```
import java.io.*;
import java.security.*;
import java.security.spec.*;

public class Export {

  public static void main(String args[]) {
    try {
      KeyPairGenerator kpg = KeyPairGenerator.getInstance("DSA");
      kpg.initialize(512,new SecureRandom());
      KeyPair kp = kpg.generateKeyPair();
      Class spec = Class.forName("java.security.spec.DSAPrivateKeySpec");
      KeyFactory kf = KeyFactory.getInstance("DSA");
      DSAPrivateKeySpec ks =
        (DSAPrivateKeySpec) kf.getKeySpec(kp.getPrivate(),spec);
      FileOutputStream fos = new FileOutputStream("exportedKey");
      ObjectOutputStream oos = new ObjectOutputStream(fos);
      oos.writeObject(ks.getX());
      oos.writeObject(ks.getP());
      oos.writeObject(ks.getQ());
      oos.writeObject(ks.getG());
    } catch (Exception e) {
      System.out.println(e);
    }
  }
}
```

32

Encryption Example

- Generate random SecretKey

```
KeyGenerator gen = KeyGenerator.getInstance("DES");
SecretKey key = gen.generateKey();
```

- Create and initialize a Cipher

```
Cipher cipher = Cipher.getInstance("DES", "SunJCE");
cipher.init( Cipher.ENCRYPT_MODE, key);
```

- Perform encryption

```
byte[] plaintext = "the time has come".getBytes();
byte[] ciphertext = c.doFinal( plaintext );
```

33

JCE - KeyAgreement

- Lets Alice and Bob establish a secret key in an insecure environment.
- Utilizes an asymmetric system. A developer must choose the key agreement algorithm. (i.e. Diffie-Hellman)
- The 'generateSecret' method returns the established secret key
- The 'doPhase' method performs the exchange
- Example:

```
KeyAgreement ka = KeyAgreement.getInstance("DH");
ka..init( alicePrivateKey );
ka..doPhase( bobPublicKey, true );
byte[] secret = ka.generateSecret();
```

34

JCE - SecureRandom

- Random numbers are important to security
- {JRE}\lib\security\java.security names the default random number generator URL,

  file:/dev/random

35

Implementation

- Follow standards and recommend key sizes blessed by the cryptographic community.
- Peer review a design and its implementation.
- Avoid writing protocols from scratch
- JCE offers no silver bullet.

36

Implementation

- Java makes no guarantee when an object is released from memory, even when calling System.gc()
- Minimize copies of the sensitive information
- Wipe your StringBuffer instances
- The paranoid ought to consider JNI

37

Java keytool

- The *Java Keytool* is a command line tool which can generate public key / private key pairs and store them in a **Java KeyStore**.
- The Keytool executable is distributed with the Java SDK (or JRE), so if you have an SDK installed you will also have the Keytool executable.

38

Keytool syntax

- keytool -genkeypair
  - -alias mykeystore
  - -keyalg RSA
  - -keysize 2048
  - -keypass 123456
  - -validity 100
  - -storetype JKS
  - -keystore mykeystore.jks
  - -storepass 123456

39

A Few Interesting Books

- General Cryptography
  - Applied Cryptography 2nd Edition, Bruce Schneier.
- Mathematical
  - Cryptography: Theory and Practice, Douglas Stinson.
- Security in General
  - Information Warfare and Security, Dorothy E. Denning

40

Useful Internet Resources

- JCE Providers
  - Cryptix http://www.cryptix.org
  - Bouncy Castle http://www.bouncycastle.org
- URLs
  - Sun's Online Developer Community
    - http://java.sun.com/
  - Sun Crypto Reference Guide
    - http://java.sun.com/j2se/1.4.2/docs/guide/security/CryptoSpec.html
  - Sun's JCE Reference Guide
    - http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html
  - Schneier.com – http://schneier.com
- Newgroups
  - sci.crypt

41