

Tutorial 1

A Small Subset of the Java Crypto API

Sandeep Kumar

skumar@comp.nus.edu.sg

Course Administrivia

- **Textbooks:** Security in Computing by Charles Pfleeger 2nd ed.
Computer Security—Art and Science by Matt Bishop.
- **Office Hours:** TBD.
- **Course Web Page:** [here](#).
- **Grading:** (for my portion of the course)
 - Mid-term: 10%.
 - Projects: 30% (3-4). Start selecting a paper to read and present.
 - Final: 10% + 30% Chaoping.

Course Administrivia...

- Get your computer accounts on *sunfire*.
- Everyone subscribe to the mailing list **cs4236-l** by sending an e-mail to **cs4236-l-request@comp** with the subject **subscribe**.
- Don't print slides yet! It'll use too much ink.
- Use `/home/course/cs4236/src/jdk/jdk.map` with emacs to view JDK source code.

Some *key*-related classes in Java

java.security.Key—top-level interface for all *opaque* keys.

- *String getAlgorithm()* — name of the algorithm of the key.
- *byte[] getEncoded()* — the raw bytes of the key representation.
- *String getFormat()* — the format of the encoded key, for e.g., PKCS#8, X.509.

KeySpec

java.security.spec.KeySpec—an interface that denotes the “transparent” (user-visible) representation of the key material that constitutes a key.

- Contains no methods or constants.
- Class **SecretKeySpec** implements **KeySpec** & **SecretKey** (more directly relevant for us).
 - **SecretKey** is an interface that extends **Key**. So it can be used with **Cipher**.
 - To generate a DES key for `0x0102030405060708`, use:

```
byte[] key = new byte[8] {0x01, ...};  
desKey = new SecretKeySpec(key, "DES");
```

Cipher

javax.crypto.Cipher—a class that provides the functionality of encryption and decryption.

- *public static Cipher getInstance(String tx)*. For e.g.,

```
Cipher.getInstance("DES/CBC/PKCS5Padding");
```

- Tx or transformation \approx algorithm/mode/padding.
- A cipher can be initialized with
 - *cipher.init(int Cipher.ENCRYPT_MODE, Key key, AlgorithmParameterSpec IV)*.
 - *cipher.init(int Cipher.ENCRYPT_MODE, Key key)*. This generates its own IV which can be retrieved with *cipher.getIV()*.

Cipher...

To encrypt a byte stream, use

```
byte[] encryptedBytes = cipher.update(buffer, 0, b_read
```

and end with

```
byte[] encryptedBytes = cipher.doFinal();
```

IV

javax.crypto.spec.IvParameterSpec—a class that specifies an IV. Implements *AlgorithmParameterSpec*.

- *public IvParameterSpec(byte[] iv)*, for e.g., create an eight byte array and initialize it with the IV and create an *IvParameterSpec*.
- Can be used in *Cipher.init(...)* because it's an *AlgorithmParameterSpec*.

Hashes

To calculate the MD5 checksum of a byte stream, use

```
MessageDigest md5 = MessageDigest.getInstance("MD5");  
md5.update(buffer, 0, b_read);  
byte[] digest = md5.digest();
```

X.500 Names

Loosely, an X.500 name is hierarchical and consists of the following attributes:

- Country: *SG*.
- State or Province: *Singapore*.
- Locality: *Clementi*.
- Organization: *National University of Singapore*.
- Organizational Unit: *School of Computing*.
- Common Name: *Sandeep Kumar*.
- Email Address: *skumar@comp.nus.edu.sg*.

Base 64 encoding

Look here for more information. There must be other references.

- Encode a sequence of octets using the characters [A-Za-z0-9+/] to represent 6 bits each.
- Use the character '=' for trailing padding.
- 6 bits of input represented as one printable character of 8 bits \Rightarrow 33% expansion.

Ex: 0x1F is Hw==. Is it?

Administrivia

- Need your group members and paper selections today.
- Will probably use the last class and the last several tutorials for presentations.
- Programming project 2 is out. Groups of 1, 2, or 3 are fine.
- Feel free to discuss the project with your classmates. However, the coding should be yours.
- Haven't started grading your programming assignment yet.
- Run `tut1.pl` for modular exponentiation example.

Why use MD5 when roiling?

Encryption alone does **not** provide integrity. Usually provided with

m $h(m)$

Client-side certificates

What were we doing in Q2 of Tutorial 1? Consider a simplified SSL protocol exchange.

$C \xrightarrow{r_1} S$	Supp cipher suites, 28B randomness
$C \xleftarrow{r_2} S$	Chosen cipher suite, 28B randomness
$C \longleftarrow S$	Server cert
$C \longrightarrow S$	$E_{servPubKey}(\text{Client selected 48B PMS})$

Both parties compute $MS \approx h(pms || r_1 || r_2)$

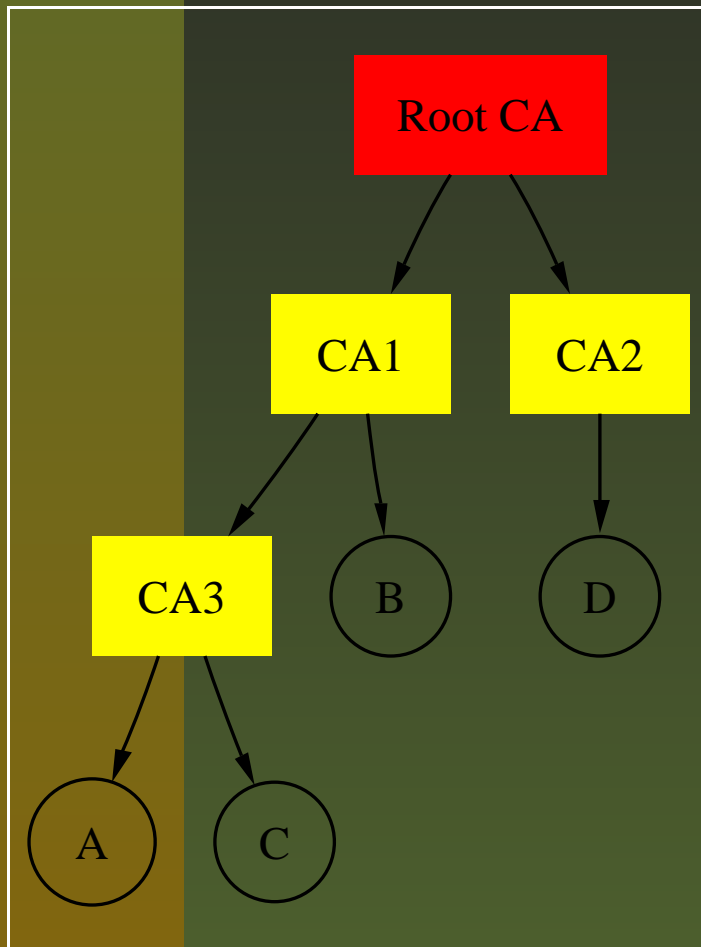
Encryption & MAC keys derived from MS

Certificates

A certificate binds a name to a key. From [Tho00, Appendix A.1]:

Version
Serial Number
Algorithm Id
Issuer
Validity
Subject
Subject's Public Key
Issuer Unique Id (optional)
Subject Unique Id (optional)
Extensions (optional)
Signature

Certificate Authority



- Everyone has root's certificate and trusts it.
- Trust flows from root → leaf.
- Is trust transitive?
- Getting a person's public key in order to communicate with them, say `scott.mcnealy@sun.com`? Use IBE.

Data Encoding

- **Explicit.** Each field is tagged with type & size.
- **Implicit.** Type is implicit in context, with tags for variant records. For e.g., RPC records are defined by the procedure-id.

ASN.1—DER

Data types are encoded as (Type, Length, Value).

- **Type** is usually one byte =



- **Length** encodes the length in bytes of the data.
- **Value** are the actual data bytes.

DER Type

- **C** is 2 bits \equiv *universal context* (00) or *context-specific* (10).
- **P** is 1 bit (primitive/constructed).
 - 0 for atomic types such as integer, bit string...
 - 1 for container types such as SEQUENCE, SET...
- **V** is 5 bits and denotes the actual tag for the type if it's < 31 . [*Unusual to find it > 30 .*]

See page 4 of Layman's guide for primitive types.

DER Length

See Section 3.1 of Layman's guide.

- Encoded in 1 byte (MSb = 0) if length ≤ 127 .
- Else, MSb set to 1 and the remaining bits indicate *the size of length field in octets*.
- Consider that many succeeding octets to encode the bit pattern of the integer as if it were a large integer.

Encoding integers

All the data bytes encode the 2's complement bit pattern of the integer.

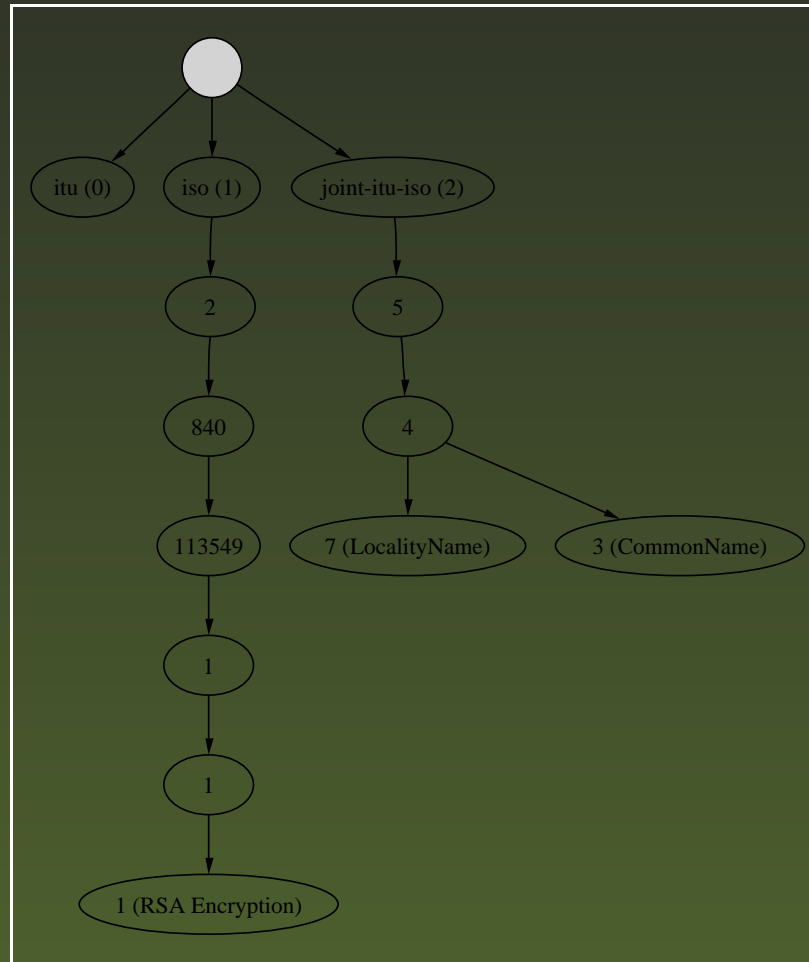
Encoding IA5String, PrintableString

- IA5String \approx ASCII string.
- Printable String \approx isprint().
- Each byte represents a character of the string.

Encoding a BitString

- One byte for number of “unused bits” followed by the bit string.
- The unused bits are the trailing bits in the last byte.

What is an OID?



Encoding an OID

- Hierarchical way to unambiguously name objects.
- BER encoded, except for the first two ints that are combined into one as $40 \times first + second$.

A BER int is a sequence of bytes with the high bit set in each byte, followed by a byte with the high bit unset.

DER Examples

- **NULL** 05 00. Universal context, primitive, length = 0.
- **INTEGER** 0 \Rightarrow 02 01 00. Universal context, primitive, length = 1, data = 0.
128 \Rightarrow 02 02 00 80. Integers are *signed*.
- **IA5STRING** “test1@rsa.com” \Rightarrow 16 0d ‘t’ ‘e’ ... ‘m’.
- **OID** 1.2.840.113549 \Rightarrow 06 06 2A 86 48 86 F7 0D. Universal context, primitive, length = 6 bytes, 2A = $40 \times 1 + 2$.

References

References

- [Tho00] Stephen A. Thomas. *SSL and TLS Essentials: Securing the Web*. John Wiley and Sons, 2000.