May 6, 1998

INTRODUCTION

United States export law prohibits the export of certain kinds of cryptographic software. My book, *Java Cryptography*, contains examples that are not exportable. Although it is customary to put the example from our books online, we can't do this with the non-exportable examples, as this would make them easily available outside the United States. The purpose of this document is to record the process I followed in deciding which examples are exportable and which are not.

There are two conflicting motivations at work here. First, we would like to put the examples from *Java Cryptography* online as a convenience to our customers. Second, we don't want to get in trouble with the United States government. Since the regulations are not well defined, this document exists to show that we did our best to comply with the law.

EXPORT LAW

Current export law prohibits or restricts software with these capabilities:

- Encryption using a key larger than 40 bits (this is called "strong" encryption)
- Drop-in or plug-in encryption capabilities, that is, any software that can easily be modified to include strong encryption. This is the primary grey area.

Cryptography used for authentication (proving identity) is not limited by export law.

OTHER ASSUMPTIONS

I'm assuming I can chop up my examples roughly along the same lines that Sun was forced to chop up their own software. Java's cryptographic capabilities are actually shipped in two products. The first is the Java Development Kit (JDK), which is exportable. The second piece is the Java Cryptography Extension (JCE), whose distribution is limited to the United States and Canada. I therefore assume, conservatively, that the concepts and implementations in the JCE are not exportable. Specifically, I assume that my examples covering key agreement protocols are not exportable.

THE NON-EXPORTABLE EXAMPLES

I weeded out the examples based on what is not allowed by export law. I have therefore grouped them below based on the offending cryptographic capability:

ENCRYPTION WITH KEYS GREATER THAN 40 BITS

- *SecretWriting* uses DES (which has a 56-bit key) to encrypt or decrypt data.
- *Cloak* uses DES to encrypt or decrypt a file.
- *PBE* uses a passphrase as a DES key to encrypt or decrypt data.
- *BlockCipher* extends Java's Cipher class to make it easy to write block ciphers, using another cipher with any key size.
- *CBCWrapper* and *CFBWrapper* implement the CBC and CFB mode algorithms on top of a block cipher with any key size.
- *ElGamalCipher* implements the ElGamal cipher algorithm, which can be used with arbitrarily large keys.

- *IDEACBCPKCS5* shows how to wire together a IDEA cipher (if you happen to have one) with the CBCWrapper class from earlier. IDEA is non-exportable.

USE OF KEY AGREEMENT PROTOCOLS

- *Skip* encapsulates some constants used in the Diffie-Hellman key agreement protocol.
- *SkipClient* and *SkipServer* are a matched client and server that perform the Diffie-Hellman key agreement protocol.
- *Skipper* is a multi-party Diffie-Hellman example.

DROP-IN CRYPTOGRAPHIC CAPABILITIES

- *SafeTalk*, *Session*, *SessionServer*, and *Receiver* make up the SafeTalk application, which uses ElGamal and DES to protect network communication.
- *CipherMail*, *Message*, *POP3*, *SMTP*, and *Composer* make up the CipherMail application, which uses ElGamal and DES for authenticated, encrypted email messaging.

SUMMARY

The examples that cannot be exported, and therefore will not appear online, are as follows:

- *BlockCipher*
- *CBCWrapper*
- *CFBWrapper*
- *CipherMail*
- *Cloak*
- *Composer*
- *ElGamalCipher*
- *IDEACBCPKCS5*
- *Message*
- *PBE*
- *POP3*
- *Receiver*
- *SafeTalk*
- *SecretWriting*
- *Session*
- *SessionServer*
- *Skip*
- *SkipClient*
- *Skipper*
- *SkipServer*
- *SMTP*

Jonathan Knudsen