INFO0045: Introduction to Computer Security
DVD Manufacturer

B. Donnet, Y. Vanaubel
Université de Liège

1 Overview

For the first assignment, you will be implementing a secure content distribution system similar to AACS[1], which is used for HD-DVDs. The DVD manufacturers have content that they would like to distribute widely yet retain control over. Only those DVD players that have an appropriate set of keys should be able to recover the content. The manufacturer should be able to revoke the privileges of a player whose keys were compromised. This revocation policy must support disabling of individual players.

For the purposes of this project, there are a maximum of 2^{32} players. The players can be viewed as the leaves of a binary tree of height 33. Each node in the tree contains an AES key, all of which are kept secret from consumers. A specific player will contain (in a file) the 33 keys associated with the path from the root to leaf nodes. The manufacturer should be able to create content that cannot be decrypted by any revoked player.

Conceptually, the system works in three phases:

- In the *key generation phase*, a master AES key (K_{AACS}) is used to generate node keys. So when a new DVD player is "created", a file is generated with the 33 player keys. This key file is protected with a password-based key.
- In the *manufacturing phase*, the DVD manufacturer encrypts content using a set of keys that is derived from the key tree and the revocation list.
- In the *playback phase*, the DVD player attempts to decrypt the content using its player keys. If the player has not been revoked, the original content should be recovered.

The required security features are:

- Security storage (password-protected) of player keys.
- Encryption of all content with AES in the *counter mode* (CTR – see Sec. 2.3).
- Integrity check for the protected content using Message Authentication Codes (MACs).

We will examine each of these features in detail in the following section.

2 Security Features

This section describes, in details, the security features of the DVD manufacturer.

2.1 Secure Generation and Storage of Player Keys

Before a player can be used, its set of 33 keys must be created and securely stored in a file. The key file will be generated by a command-line utility, that takes a 32-bit player number and a pass- word. This utility will then generate the 33 keys representing a path from the root to the leaf, corresponding

[1] *Advanced Access Content System*. See `http://en.wikipedia.org/wiki/Advanced_Access_Content_System` for details

to the given player number. The keys will then be stored in an encrypted file. The file encryption key will be generated using the provided password.

Note that the full tree of all keys is never stored in one place, for security and storage reasons. So, you will have to derive all player keys, as needed, from K_{AACS}. We will provide you a representa- tion of the unique IDs for each tree node, but you will need to come up with a way to derive an AES key from a node ID. We will also provide a method that you can use to convert a password into an AES key.

2.2 Content Encryption

Content must be encrypted by the DVD manufacturer using AES in CTR mode, with a new, random *title key* for each title encrypted. In order to ensure that revoked players cannot decrypt content, a revocation list must be available to the manufacturer. Then, the *title key* K_t will be encrypted under an appropriate subset (*cover*) of the keys in the tree, and stored along with the encrypted content. This content header should also contain a *content title* string, which is descriptive metadata about the content. The content header plus the encrypted content must have an associated MAC.

Note that the key used for encrypting content, K_{enc}, and the key used for MAC'ing K_{mac} must be derived from the title key:

- $K_{enc} = HMAC(K_t, \text{“}enc''})$
- $K_{mac} = HMAC(K_t, \text{“}mac'')$

Along with each encrypted *title key*, the content header should contain an encoded node ID, for efficient decryption. Otherwise, the player would have to try every combination of (player key, encrypted *title key*). Your decryption solution should run in time proportional to $k + l$ as opposed to $k \times l$, where k is the number of keys in the player and l is the number of encrypted copies of the *title key* in the content header.

2.3 Counter Mode (CTR)

Counter mode of encryption effectively turns a block cipher into a stream cipher. Pseudorandom sequence is generated by encrypting successive values of a counter. Formally, encryption of a message $(m_0, m_1, \ldots, m_n)$ is $(IV, E(k, IV) \oplus m_0), E(k, IV + 1) \oplus m_1, \ldots, E(k, IV + n) \oplus m_n)$. Similarly, as in other modes of encryption, new IV should be chosen randomly each time.

Fig. 1 illustrates encryption (Fig. 1(a)) and decryption (Fig. 1(b)) modes of CTR.

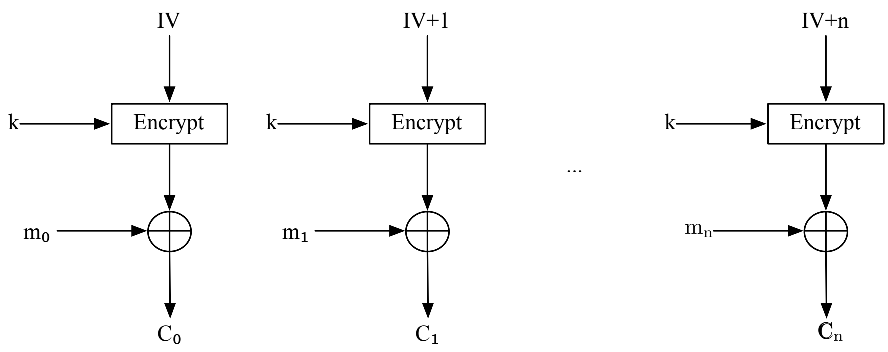
2.4 Integrity Check using MACs

The key used to generate MACs for the content should be derived from the *title key.* This will enable detection of an attacker tampering with the protected content.

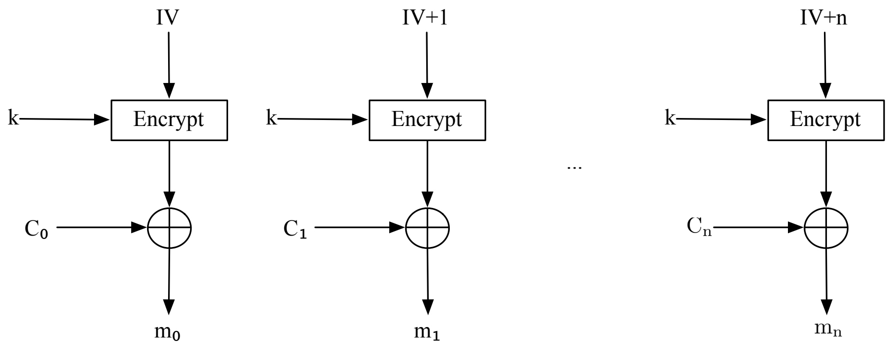
2.5 Content Decryption

Before attempting to decrypt content, the DVD player must access its player keys from an encrypted file. Then it will try to determine the *title key* for the encrypted content. There are three possible outcomes:

- If none of the player keys can be used to recover the *title key*, then this player has been revoked, and an appropriate error message should be output.
- If the MAC is invalid, then the protected content has been tampered with, and an appropriate error message should be output.
- Otherwise, the original unencrypted content should be output.



(a) Encryption



(b) Decryption

Figure 1: CTR Mode illustration

3 Components

This section describes the various components and how they interact between themselves. The name given to each component is the one of the corresponding Java class (see Sec. 4.2). This section describes thus the various mechanisms you will have to implement.

3.1 PlayerKeys

PlayerKeys implements functionality related to the generation of the player keys. The main purpose is to provide a command line utility for generating an encrypted file containing keys for a given player.

In generating player keys, the PlayerKeys utility will take the *master AACS password* (used to derive K_{AACS}), a player number n and a *player password* (used to encrypt the player key file). The output filename is `player_n.key`. The same *player password* will be used by a DVDPlayer to decrypt its key file. Note that all nodes in the key tree are derived from the master key K_{AACS}. Rather than persistently storing K_{AACS} you will derive it from the *master password* provided on the command line. The *same master* password must be used for all operations on a particular key tree.

- PlayerKeys will generate an AES key, K_{enc}, and a MAC key, K_{mac}, from the *player password* shared with player n.
- Then it will encrypt and MAC the player keys for DVD player n as follows:

$$\text{E[} K_{enc}, \text{ player\_keys] || MAC[} K_{mac}, \text{ E[} K_{enc}, \text{ player\_keys]]}$$

- Note that the list of player keys should include some identifying information for each key. This will make determining the appropriate player key to use (in the decryption step) more efficient.
- Then PlayerKeys will write all of this to the given output file and exit.

3.2 KeyTree

The KeyTree class implements all functionality querying of the tree of all node keys. The main functions provided for you are:

- An API (`getPathNodes()`) for retrieving a list of nodeIDs starting from the root and ending at a leaf (i.e., a player).
- An API (`getCoverSet()`) that takes as input a list of revoked player numbers and returns a list of keys (a cover) needed to encrypt a *title key* under. This will be used by the DVDManufacturer.
- An API (`createAESKeyMaterial()`) for converting a password into a byte array that can be used as an AES key.

3.3 DVDManufacturer

The DVDManufacturer will be run as a command line utility that takes a filename (of some content) and a content title (a metadata string). As with the PlayerKeys utility, you will derive K_{AACS} from a master password provided on the command line, and use it to derive any needed node keys. The revocation list file is expected to be at `revoke.lst`, and the output file name is the input filename with a `.enc` extension.

Before encryption is started, a new random *title key* K_t is generated, and encryption/MAC keys K_{enc} and K_{mac} are derived from it (as described in Sec. 2.2). First, the content must be encrypted with K_{enc} using AES in CTR mode. Next, the KeyTree is queried for a list of nodes under whose keys K_t will be encrypted. From this list you will derive a set S of AES keys. Then, the content header is generated and the header and content are MAC'd using K_{mac}.

Finally, an output file will be generated as follows:

[content_header] || E[K_{enc}, content] ||MAC[K_{mac}, [content_header] || E[K_{enc}, content]]

Where the *content_header* consists of K_t encrypted under each key in S, as well as some identifying information for each encrypted key and a *content title*. This identifying information should allow a player to determine which of its keys to use to try and decrypt K_t.

3.4 DVDPlayer

The DVDPlayer will be run as a command line utility that takes the player key file password, and the filename of some encrypted content. The player keyfile is expected to be `player_n.key` and the output file (if generated) will be the encrypted filename without a `.enc` extension. Before decryption is started, the player key file is read in, decrypted, and the MAC checked. Then, the protected content is read in and the player attempts to recover the *title key* K_t from the content header. Given K_t, the MAC will be verified and the content decrypted.

4 Implementation

You will be using the JCE (Java Cryptographic Extensions) while programming for this assignment. You should spend some time getting familiar with the provided framework.

4.1 Getting the Code

A skeletton of the code is provided. It is available on the course web site (see `http://www.montefiore.ulg.ac.be/~bdonnet/info0045/documentation.html`).

Once uncompressed, the source tree for the assignment will be created and a Makefile will be provided.

4.2 Description of the Code

Here is a brief description of the files we provide.

- **Makefile** Makefile for the project.
- **revoke.lst** Revocation list file.
- **info0045/PlayerKeys.java** Used to generate player key file.
- **info0045/KeyTree.java** Key tree utilities.
- **info0045/DVDManufacturer.java** The encryption program.
- **info0045/DVDPlayer.java** The decryption program.

4.3 Running the Code

To build the project, simply type `make`. To run the system, follow these steps:

1. Run the PlayerKeys utility to generate a set of keys for a player n. You will also specify a password for deriving the AACS Master Key and a keyfile password. The output filename will be `player_n.key`.

```
$> java info0045/PlayerKeys <AACSPwd> n <keyfile pwd>
```

2. Encrypt some content in a file. You will specify a metadata string (e.g., a title) and a content file name. You will also specify a password for deriving the AACS Master Key. Note: the revocation list file is in `revoke.lst` and the output file will be content file `name.enc`.

```
$> java info0045/DVDManufacturer <AACSPwd> <content title> <content file name>
```

3. Decrypt the encrypted content file. You will specify a player number, the keyfile password, and an encrypted content file. The player key file is expected to be `player_n.key` and the output file name will be the encrypted file name without the `.enc` extension.

```
$> java info0045/DVDPlayer n <keyfile pwd> <encrypted content file name>
```

4.4 Crypto Libraries and Documentation

Java security and cryptography classes are divided into two main packages:

- java.security.*
- javax.crypto.*

The following are some links to useful documentation:

- JCE Reference Guide: `http://docs.oracle.com/javase/1.4.2/docs/guide/security/jce/JCERefGuide.html`.
- Chapter 6 from Java Cryptography by Jonathan Knudsen: `http://oreilly.com/catalog/javacrypt/chapter/ch06.html`.

Some classes/interfaces you may want to look at:

- javax.crypto.KeyGenerator
- javax.crypto.SecretKey
- javax.crypto.IvParameterSpec
- javax.crypto.Mac

- javax.crypto.Cipher
- javax.crypto.CipherInputStream
- javax.crypto.CipherOutputStream
- javax.crypto.SecretKeyFactory
- java.security.MessageDigest
- java.security.SecureRandom
- java.math.BigInteger

5 Assignment Rules

5.1 Submission

The submission of your assignment is subject to the following rules:

1. you must give back an archive, with format `tar.gz` (this is the only accepted format). The archive will be named as follows: `Group-XX.tar.gz`, where `XX` refers to your group ID.
2. your archive will include the following items:
   - a Makefile for compiling your code. Normally, the one we provide you should be enough.
   - a PDF file (this is the only accepted format) with the name `Groupe-XX.pdf` (where `XX` refers to your group ID). The PDF document will contain a clear, accurate, and concise explanation regarding your code. In particular, we will look at the different steps of your reasoning and any element required to understand your solution (figures, etc.).
   - the directory `info0045` will contain all Java classes required by the assignment.
   - if you are more comfortable with adding new classes and packages, do not hesitate. But this should be clearly justified in your PDF report.
3. your archive must be uploaded on the submission platform (see `http://submit.run.montefiore.ulg.ac.be`)
4. the deadline is **March 27th, 2015, 08:00AM**. The deadline is strict. The assignment cannot be uploaded after the due date. This means that, after the due date, assignments not uploaded will receive a zero.

5.2 Gradings

We will grade your assignment as follows:

- Coding style. This assignment is not given in a programming course context. However, at this step of your studies, we assume you are able to provide elegant solutions to complex problems. The coding style (and, consequently, your solution elegance) will be part of the grading.
- Documentation. Java comes with a powerful tool for documenting code: Javadoc. We ask you to fully document your code with respect to Javadoc standards.
- Report. Your solution description in your PDF report will be also part of your grading.
- Code behaviour. It is obvious your code must meet all security features described in this document.

In case of conflict when correcting your program regarding compiling and running of your code, computers from the algorithmic lab will be used as baseline.