Our DRM scheme is designed to allow users to play their favorite songs, while simultaneously allowing the "owners" of these songs to bill the users for it.

1 Use case

The client browses the list of songs on zStore.com. He finds his favorite song and decides to buy it – even though the song is immoderately expensive. [1]

In order to listen to the song, the client will first have to purchase our hardware player. In the process of buying the song, the zTunes client will interface with the hardware device and send its identification – along with the client's credit card number – to the zStore servers.

Once zStore servers verify the completion of the monetary transaction, a license for the song bound to the client's hardware player will be generated. The client will be supplied with the license and a link to download the data package containing the song.

The client can then "enjoy" his limited number of plays.

2 Technical description

The DRM system consists of three components – the data server which provides the encrypted data packages containing the actual songs, the license server which issues licenses based on clients requests, and one or more clients.

2.1 Certificates

There are several certificates issued in the system.

- The license server is issued a key and a certificate, which is stored on the client's device and allows the client to verify the origin of licenses.
- Each client has a key and a certificate. This is used by the license server to bind the license to that particular client.
- Client certificates are signed by a root certificate, which allows the license server to ensure that the license is bound to a trusted client device.

2.2 Data server

The data server converts the file containing the song into a data packages, which is merely AES-CBC-encrypted version of a PCKS7-padded song. The encryption key is generated randomly.

In other words, the data server receives a file containing the song, and produces the encrypted file, and a description file containing

[1]Note that the target audience does not consist merely of stupid people; In fact, we expect many otherwise intelligent people to be peer-pressured to pay for data.

1. the hex-encoded SHA-512 hash of the resulting encrypted file, and
2. the hex-encoded key used to encrypt the file.

The former file can be distributed freely, whereas the latter must be kept secret and transferred to the license server in a secure manner.

2.3 License server

The license server generates licenses given the package description file (generated by the data server) and the client's certificate. First, the certificate is checked for being signed by the root certificate. Then a license is generated – a XML file containing among other things a unique license number, the hash of the encrypted data package, the encryption key the data package was encrypted with, and information on the how the package may be used (e.g., limits on the number of plays).

The XML license is then AES-CBC-PKCS7 encrypted by a random key, the key is then encrypted by the clients public key and prepended to the license. This effectively binds the license to the target client. The resulting blob is then SHA-512 hashed and signed by the license private key.

The license can be tranferred to the client through an unsecure connection.

2.4 Client

The client receives the license and attempts to install it. The installation process consists of verifying that the license originates from the license server (using the license server's certificate), and of decrypting the license (using the client's private key). The unique license number is used to verify that the license is being installed for the first time. The client will then store the encryption key and update metrics information (i.e. the remaining number of plays) and associates them with the package hash. Both the key and metrics data reside on a secure storage.

Given the data package, the client is then able to play the song. It first computes the hash of the package and looks up the associated encryption key and metrics. It verifies that the metrics allow for the song to be played. It then updates the metrics and decrypts the package using the looked-up key.