FairPlay Streaming Overview

Developer

Contents

Figures

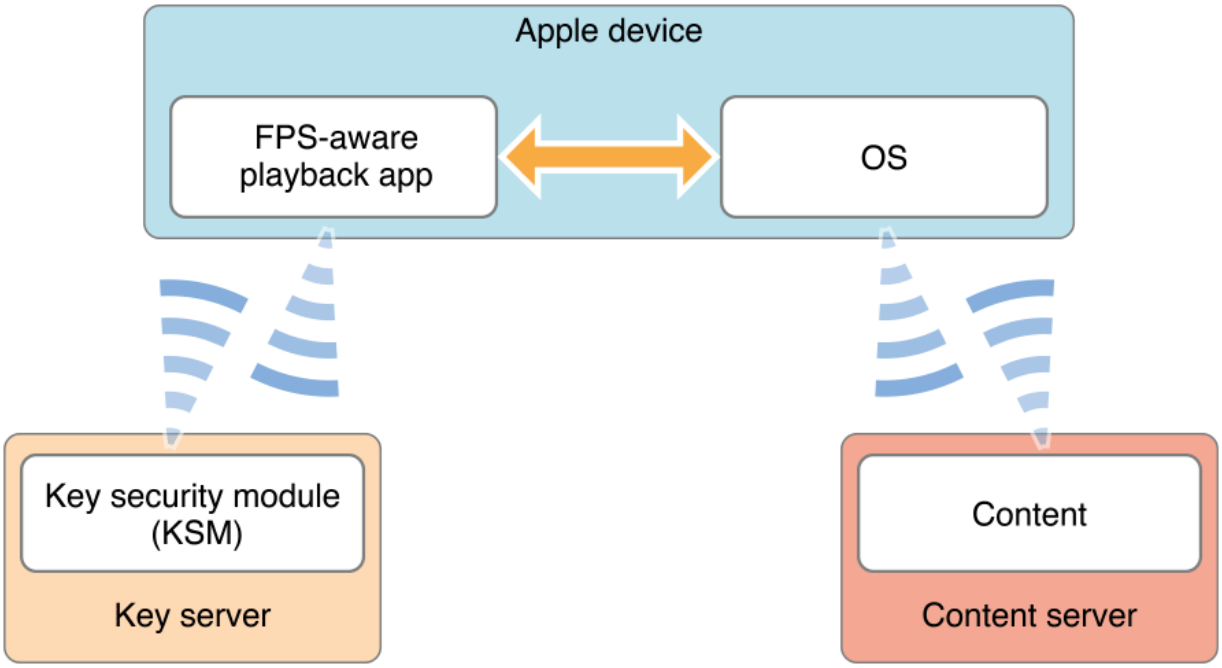
FairPlay Streaming

This overview provides a high-level description of Apple's FairPlay Streaming (FPS) specification. FPS securely delivers keys to Apple mobile devices, Apple TV and Safari on OS X, which enables playback of encrypted video content. This content is delivered over the Web using HTTP Live Streaming (HLS). FPS allows mobile devices and Apple TV to stop playback based on expiration information sent with the content key. Additionally, a constant FPS device identifier is sent to the server in a server playback context (SPC) message, allowing the server to anonymously and privately identify the device. FPS is integrated into the device operating systems, with native support on iOS and Apple TV. Safari on OS X enables FPS using EME interface support.

Key Delivery Process

FPS allows a content provider to securely deliver an AES 128-bit content key from the provider's key server. The content provider encrypts the H.264 video content with the content key. Then, FPS decrypts the encrypted video content with the content key. Figure 1-1 shows the key and content exchange.

Figure 1-1 FPS exchanges



- The FPS framework initializes the key delivery process to create a session with the content provider key server.
- The content provider key server wraps the 128-bit AES content key with the session key and an anti-replay mechanism.
- The key delivery process implements a triple-protection solution (AES, RSA, and derivation functions).
- The content provider encrypts the H.264 video content on a per frame basis using AES-CBC mode with the content key and the initialization vector.

- The content provider fully encrypts the audio content on a per sample basis using AES-CBC mode with the content key and the initialization vector.
- FPS supports the H.264 video codec and AAC-LC, HE-AACV1-2, AC-3, and EC-3 audio codecs. Audio codecs may vary for Safari's FPS.
- The key handling and the content decryption occur on the kernel of the iOS device. In other words, the content as well as the content key are kept on the device kernel for the decryption.
- FPS always enforces HDCP for each protected content block.
- FPS supports persistence of the security material for offline playback.
- The content provider manages the content key, initialization vector database associating keys, and the encryption mode with the content.
- FPS supports AirPlay streaming to Apple TV.
- FPS provides anonymous identification of the device requesting the content key.
- The secure lease mechanism that allows the content providers to manage simultaneously streams per account.
- FPS supports content key expiration for movie rental on iOS devices and Apple TV.

Content Ingestion and Formatting

Video content ingestion is not part of the FPS solution. However, some specific requirements have to be fulfilled to match the key delivery and the protection of the video content.

The content needs to be authored for delivery using HLS. More information on HLS, including a detailed specification, tools, and best practices, can be found on the Apple developer page at http://developer.apple.com/streaming.

The content provider encrypts the H.264 video against the content key on a per frame basis during the ingest process. The initialization vector (IV) has a unique value per encryption, and it is transported along with the content key. The content provider handles the content ID that uniquely identifies the content, the content key, and the initialization vector.

Key Request

At the time of playback, the content provider's application initiates the content key request in FPS. This first phase creates a server playback context (SPC). It prepares a graphic crypto-context on the client — to later unwrap the content key and the initialization vector — that includes the session key, an anti-replay seed, integrity verification, and server authentication elements. FPS protects the SPC with the content provider's RSA public key. A number of verifications are performed by FPS to check the content provider's certificate before exchanging encrypted media.

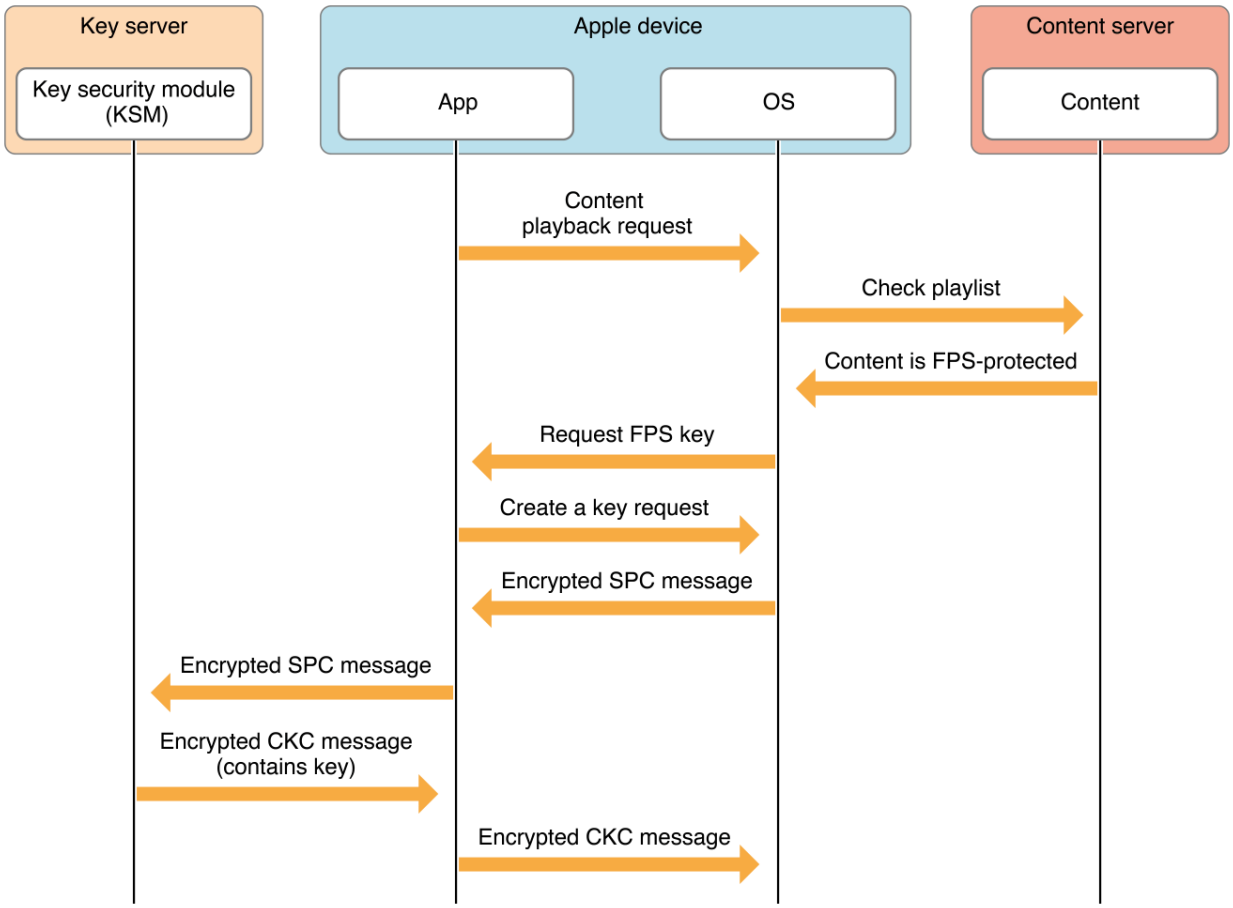
The content provider's application sends both the content ID and the SPC to the content provider's server. The server uses the content ID to fetch the appropriate content key and initialization vector. The content provider opens the SPC to extract the session key, the anti-replay seed, the integrity information, and the authentication materials. Additionally, the SPC includes a secured version of the content ID as provided by the application for best practice server verification.

The next step wraps the content key and the initialization vector according to the FPS specification. The integrity verification is optional, but it is part of the best practice requirements. Next, the server creates a

content key context (CKC) in response to the application that contains the content key and the initialization vector. The client application then provides CKC to FPS to request a play context.

The transaction with the SPC is valid for a single request. In other words, the anti-replay information prevents an attacker from invoking FPS more than once with the same server response. The FPS transaction process is shown in FairPlay Streaming.

Figure 1-2 FPS communication sequence



Device Identification

Identifying the device that requests the content key is a major privacy challenge. The FPS server playback context (SPC) provides an anonymous device identification value. The playback device preserves user privacy by deriving a unique device identifier and adding the content provider's application certificate in the SPC. The content provider uses the SPC to distinguish the playback device from other devices, and no user information is exchanged.

Content Key Expiration

Starting with iOS 9, FPS allows the server to send an expiring content key to Apple mobile devices and Apple TV. Creating the time-limited key begins with the playback device sending its time reference to the server in the SPC message. The server manages the key's expiration according to the playback device's context, independently of the server time location. The server's content key context (CKC) response contains the expiration time along with the content key.

FPS's content key expiration creates two modes of time sensitive exchange: video rental and secure lease. These modes can be used separately or together.

Video Rental

The content key is a rental type. FPS does not start the decryption if the content key has expired. However, FPS continues the user experience if the content key expires during the playback. Only at the start of the next playback request does the client decline the playback.

Secure Lease

The content key is a lease type. It is typical that a content provider policy would restrict the number of simultaneous playbacks (slots) for a user account. The server associates a slot to a device, and the server delivers the content key with expiration that represents the lease. The client may request that the key be renewed by the server before the lease expires. The server provides a new expiration time for the content key, and playback continues uninterrupted. If the content key is not renewed, the client stops the playback when the lease expires. The server recognizes playback has stopped and the server frees the device slot.

This design ensures that a device is not orphaned (in a stale state) based on time rather than messaging and garbage collection. The expiration triggers a server event to securely release the device slot. The server knows playback stopped and frees the device slot as soon as the content key expires and the play content is discarded. Using the secure lease and the device identification, the server can implement a robust solution for the management of simultaneous streams, maintaining a seamless user experience.

Key Management Cryptography

The protection uses AES-128. It protects the key during delivery from the server to the device. The encryption implements a transient and random session key (SK). In other words, the value changes for every request, even for the same content. SK is a nonce value that FPS selects at the key request start and FPS shares SK with the server at the runtime.

FPS protocol protects the SK exchange with an advanced cryptographic solution that implements a first layer of protection using an asymmetrical algorithm, the second using a key derivation and finally symmetrical encryption that contains the integrity verification.

The content key protection relies on AES-128 with a double protection that implements the anti-replay protection.

Content Playback

Once the content key and the initialization vector have been unwrapped and the play context created, then the application can process the decryption of the content on the per frame basis for video or audio samples. When the frame or the audio sample is decrypted, the decryption process directly passes the frame or the sample to the relevant decoder.

Context Persistence

For iOS, the FPS framework may persist the play context for offline playback. The server indicates the nature of the Content Key type (i.e., transient or persistent) in the CKC response. When the Content Key type needs persistence, FPS creates a persistent play context that the application manages. The play context is cryptographically resistant and FPS links the play context with the iOS device.

The application may save the play context within the iOS file system. If the Content Key has an expiration time, the play context includes the expiration. However, the play context never expires for a Content Key without an expiration time. The application has the full flexibility to manage a play context that does not expire.

Application Authentication

The FPS framework does not provide any authentication solution for the application to be authenticated by its key server. The content provider's application and the server can implement any authentication method to prevent unauthorized access to their service.

However, FPS verifies during the key delivery that the server that wraps the content key is server provisioned with FPS credentials. The framework authenticates the server using RSA combined with symmetrical algorithm and several challenge values.

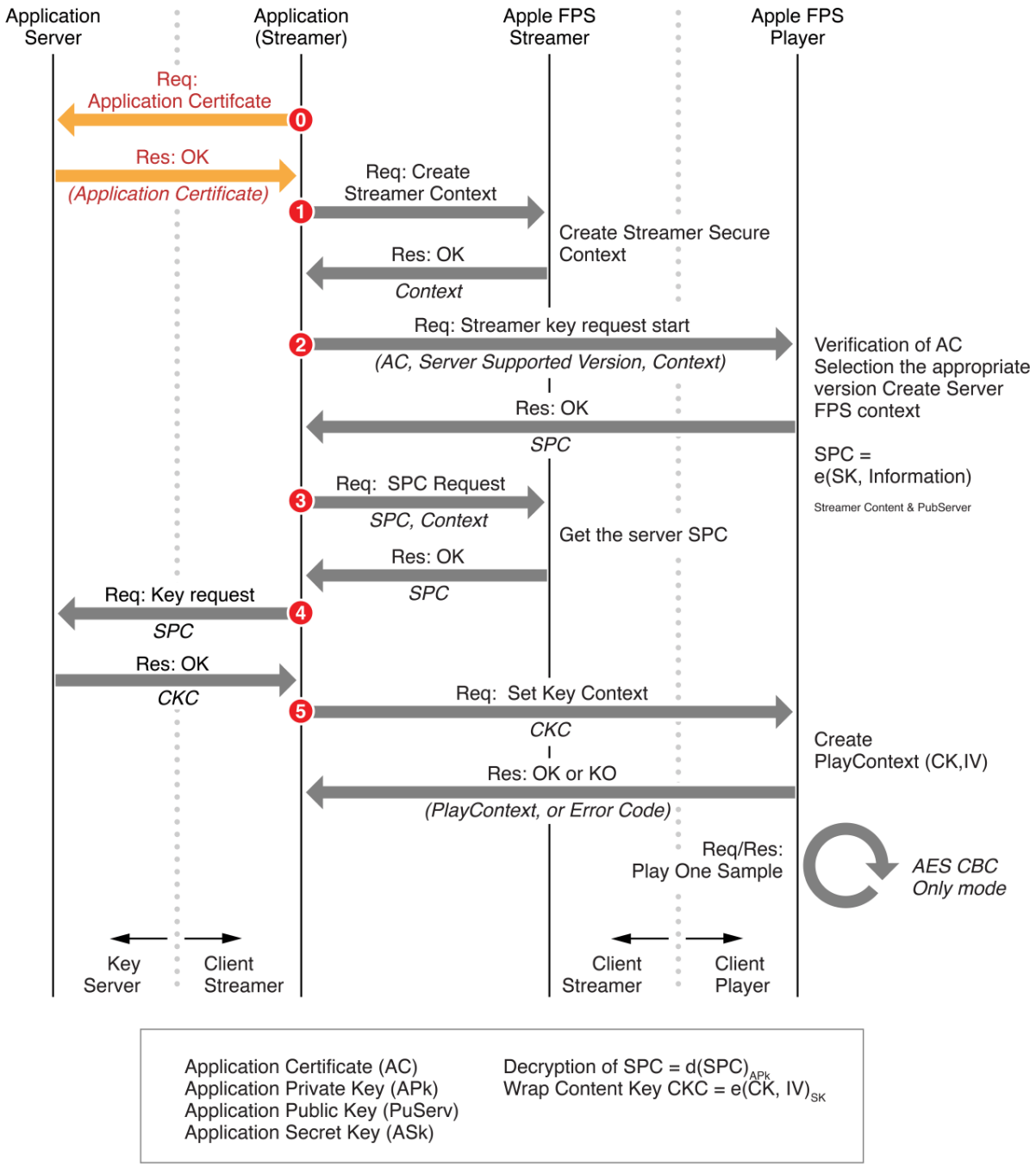
Versioning Management

FPS uses the latest compatible and most advanced version of security available on the platform. The selection is based on the supported version(s) of the content provider's servers as advertised in the application during the content key request. This feature ensures that the most secure solution is used to transport the content key. FPS protocol implements a protection mechanism that avoids an attack to declassify the server request in order to get an older version of the security scheme.

FPS Over AirPlay

Apps on an iOS device (the streamer) can send FPS content securely over AirPlay to Apple TV (the player). The streamer facilitates transmission over AirPlay and sends the FPS encrypted content and key to Apple TV for decryption. Figure 1-3 shows this process.

Figure 1-3 AirPlay content key request protocol



Application Certificate (AC)
Application Private Key (APk)
Application Public Key (PuServ)
Application Secret Key (ASk)

Decryption of SPC = $d(SPC)_{APk}$
Wrap Content Key CKC = $e(CK, IV)_{SK}$

- The key request starts from the device that produces the streamer context. The streamer context restricts the process to the device.
- The Apple TV produces a specific SPC that includes the streamer context.
- After a round trip to the key server, the streaming device proxies the wrapped CKC to the Apple TV to access the content key and the initialization value.

FPS over AirPlay implements the same level of security for the key delivery than FPS. The player handles the key, decrypts and plays the content independently of the communication paradigms. For more information, read *FairPlay Streaming Programming Guide*.

Document Revision History

This table describes the changes to *FairPlay Streaming Overview*.

Date	Notes
2016-06-24	Revised for Offline HLS support with FairPlay Streaming.
2015-06-10	Added information about Context Persistence.
2015-06-06	New document that summarizes FairPlay Streaming encryption for HTTP Live Streaming.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010