声网

Web媒体处理与实时传输标准实践

Practices of Web Media Processing and Real-time Communication Standards

高纯

声网

Outline

1. Background
2. Case1: E2E Encryption
3. Case2: Digital Rights Management
4. Case3: H265 Supporting for RTC
5. Case4: Alpha Video Transmission

声网

Background

New Trends in the RTC Industry

Expanding overseas business

Web is the most important platform overseas

Security and compliance by design is required by foreign laws

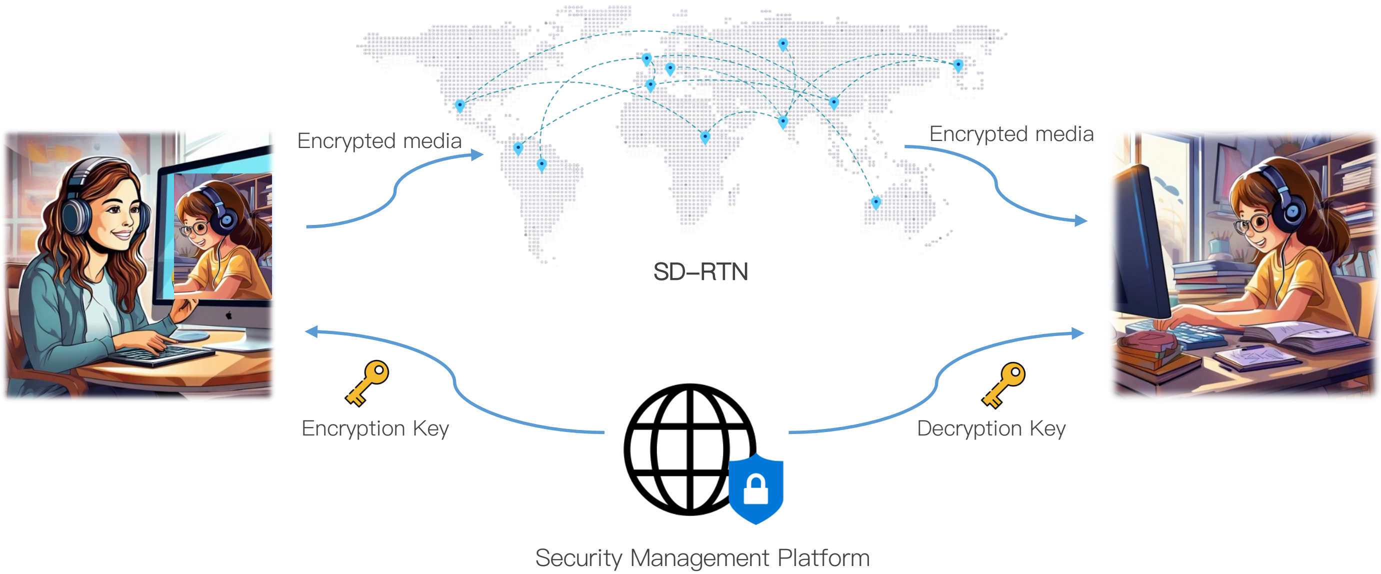
Diverse application scenarios create diverse requirements on web infrastructure

声网

Case 1: E2E Encryption

Providing reliable transmission network

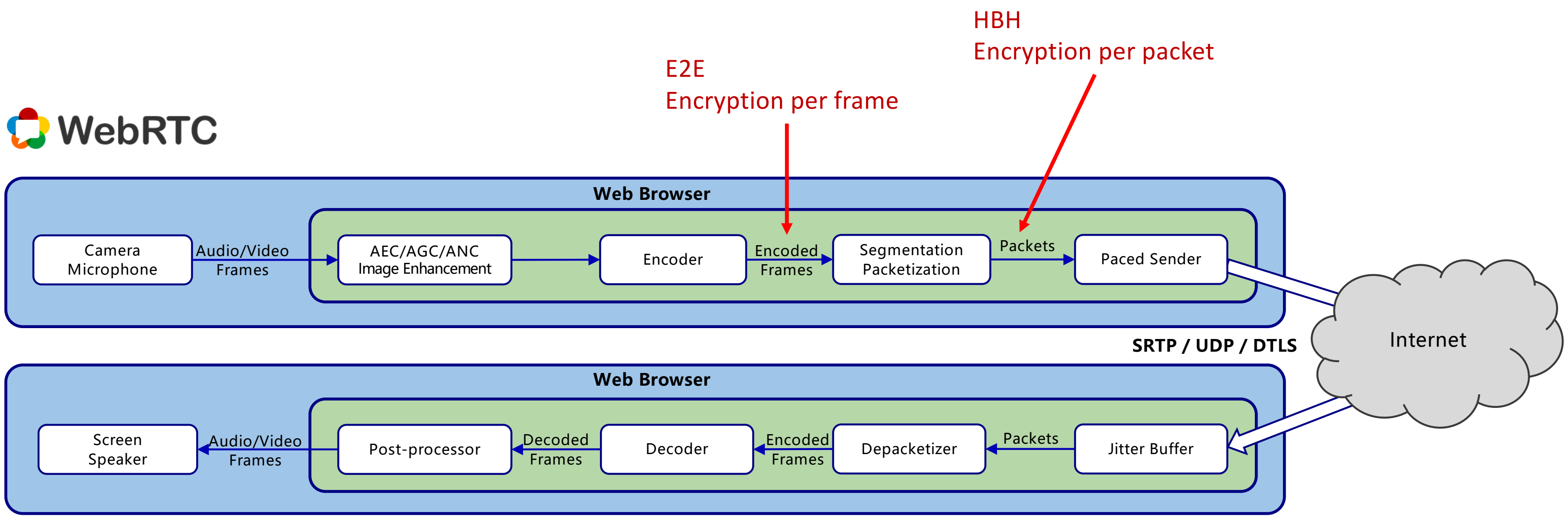
- Encrypt and decrypt media by end device to protect privacy
- User data could not be decrypted by RTC service provider



Case 1: E2E Encryption

Requirements:

- Provide interface to grab encoded media data
- Provide high performance encryption/decryption component
- Provide interface to write back transformed media data



声网

Case 1: E2E Encryption

Secure Frame (SFrame)

- End–to–end encryption and authentication mechanism for media frames
- Compatible with RTP & non–RTP media transport
- Reduce bandwidth overhead by adding encryption overhead only once per media frame, instead of once per packet.

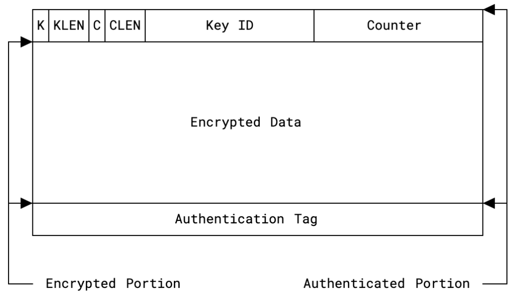
Workgroup: Network Working Group
Internet-Draft: draft-ietf-sframe-enc-04
Published: 22 October 2023
Intended Status: Standards Track
Expires: 24 April 2024
Authors: E. Omara *Apple*, J. Uberti *Google*, S. Murillo *CoSMo Software*, R. L. Barnes, Ed. *Cisco*, Y. Fablet *Apple*

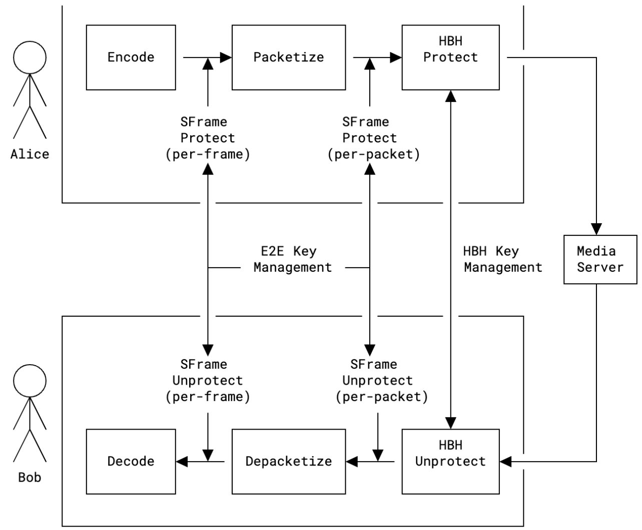
Secure Frame (SFrame)

Abstract

This document describes the Secure Frame (SFrame) end-to-end encryption and authentication mechanism for media frames in a multiparty conference call, in which central media servers (selective forwarding units or SFUs) can access the media metadata needed to make forwarding decisions without having access to the actual media.

The proposed mechanism differs from the Secure Real-Time Protocol (SRTP) in that it is independent of RTP (thus compatible with non-RTP media transport) and can be applied to whole media frames in order to be more bandwidth efficient.





Case 1: E2E Encryption

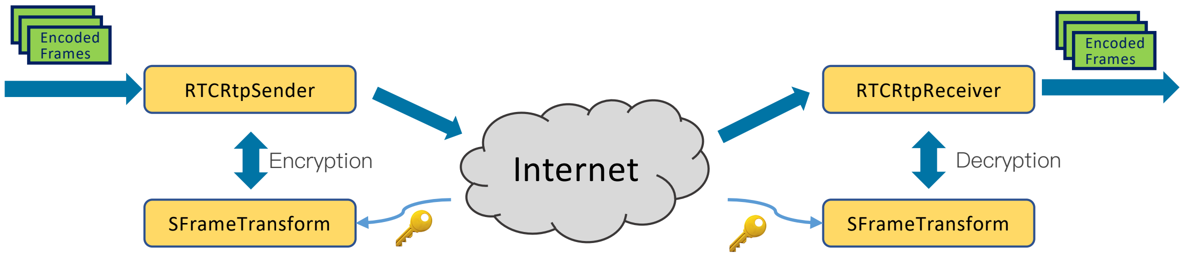
声网

SFrameTransform

Spec: https://www.w3.org/TR/webrtc-encoded-transform/

```
typedef (SFrameTransform or RTCRtpScriptTransform) RTCRtpTransform;

// New methods for RTCRtpSender and RTCRtpReceiver
partial interface RTCRtpSender {
    attribute RTCRtpTransform? transform;
};
partial interface RTCRtpReceiver {
    attribute RTCRtpTransform? transform;
};
```

```
enum SFrameTransformRole {
    "encrypt",
    "decrypt"
};
dictionary SFrameTransformOptions {
    SFrameTransformRole role = "encrypt";
};

typedef [EnforceRange] unsigned long long SmallCryptoKeyID;
typedef (SmallCryptoKeyID or bigint) CryptoKeyID;

[Exposed=(Window,DedicatedWorker)]
interface SFrameTransform : EventTarget {
    constructor(optional SFrameTransformOptions options = {});
    Promise<undefined> setEncryptionKey(CryptoKey key, optional CryptoKeyID keyID);
    attribute EventHandler onerror;
};
```



声网

Case 1: E2E Encryption

web-platform-tests dashboard

Sign in with GitHub

Latest Run | Recent Runs | ✨Interop 2024✨ | Insights | Processor | About

wpt / webrtc-encoded-transform

partly

For information on the search syntax, view the search documentation

Showing 5 tests (11 subtests) in webrtc-encoded-transform from the latest master test runs for chrome[experimental], edge[experimental], firefox[experimental], safari[experimental] LINK EDIT

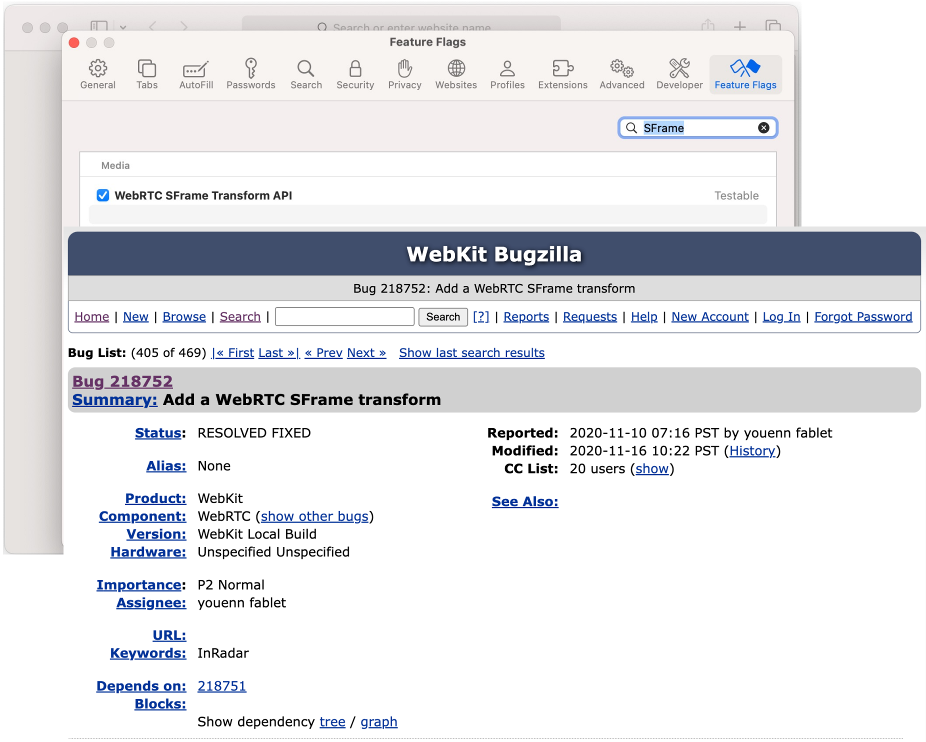
Path	Chrome 126 Linux 20.04 53eba69 May 16, 2024	Edge 126 Windows 10.0 53eba69 May 16, 2024	Firefox 128 Linux 20.04 53eba69 May 16, 2024	Safari 194 preview macOS 13.6 53eba69 May 16, 2024
sframe-keys.https.html	0 / 2 ⚠	0 / 2 ⚠	0 / 2 ⚠	0 / 2 ⚠
sframe-transform-buffer-source.html	0 / 1	0 / 1	0 / 1	0 / 1
sframe-transform-in-worker.https.html	0 / 1	0 / 1	0 / 1	0 / 1
sframe-transform-readable.html	0 / 1	0 / 1	0 / 1	0 / 1
sframe-transform.html	0 / 6	0 / 6	0 / 6	0 / 6
Subtest Total	0 / 11	0 / 11	0 / 11	0 / 11

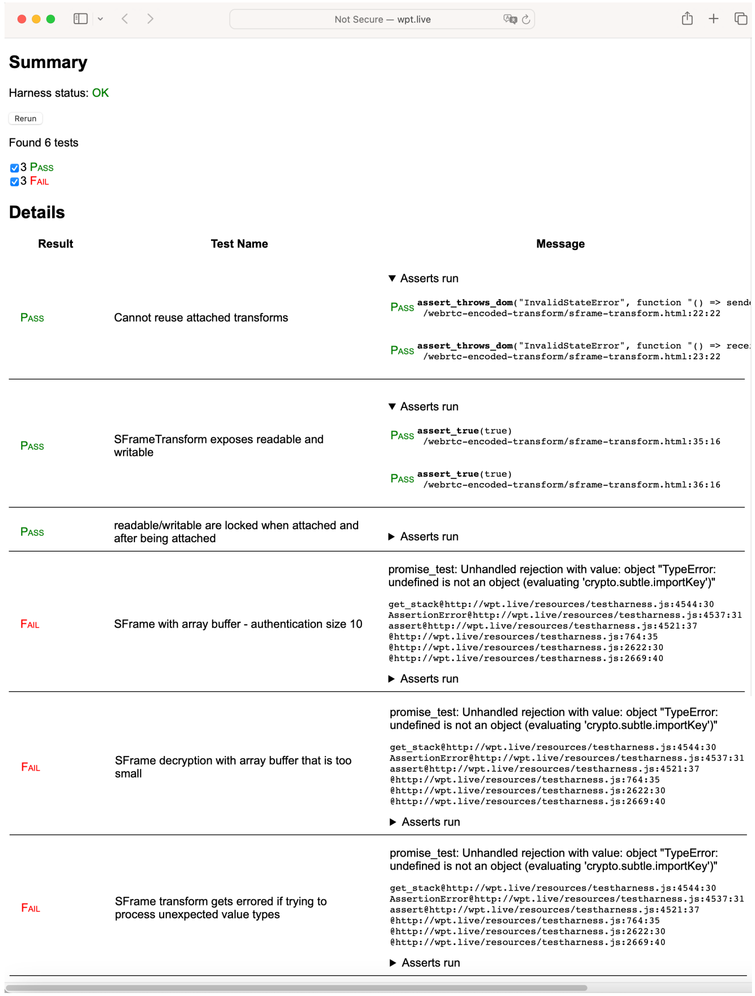
声网

Case 1: E2E Encryption

Partly testable on Safari with feature flag enabled

Functionalities not ready





声网

Case 1: E2E Encryption

RTCRtpScriptTransform

Spec: https://www.w3.org/TR/webrtc-encoded-transform/

```
typedef (SFrameTransform or RTCRtpScriptTransform) RTCRtpTransform;

// New methods for RTCRtpSender and RTCRtpReceiver
partial interface RTCRtpSender {
    attribute RTCRtpTransform? transform;
};
partial interface RTCRtpReceiver {
    attribute RTCRtpTransform? transform;
};
```

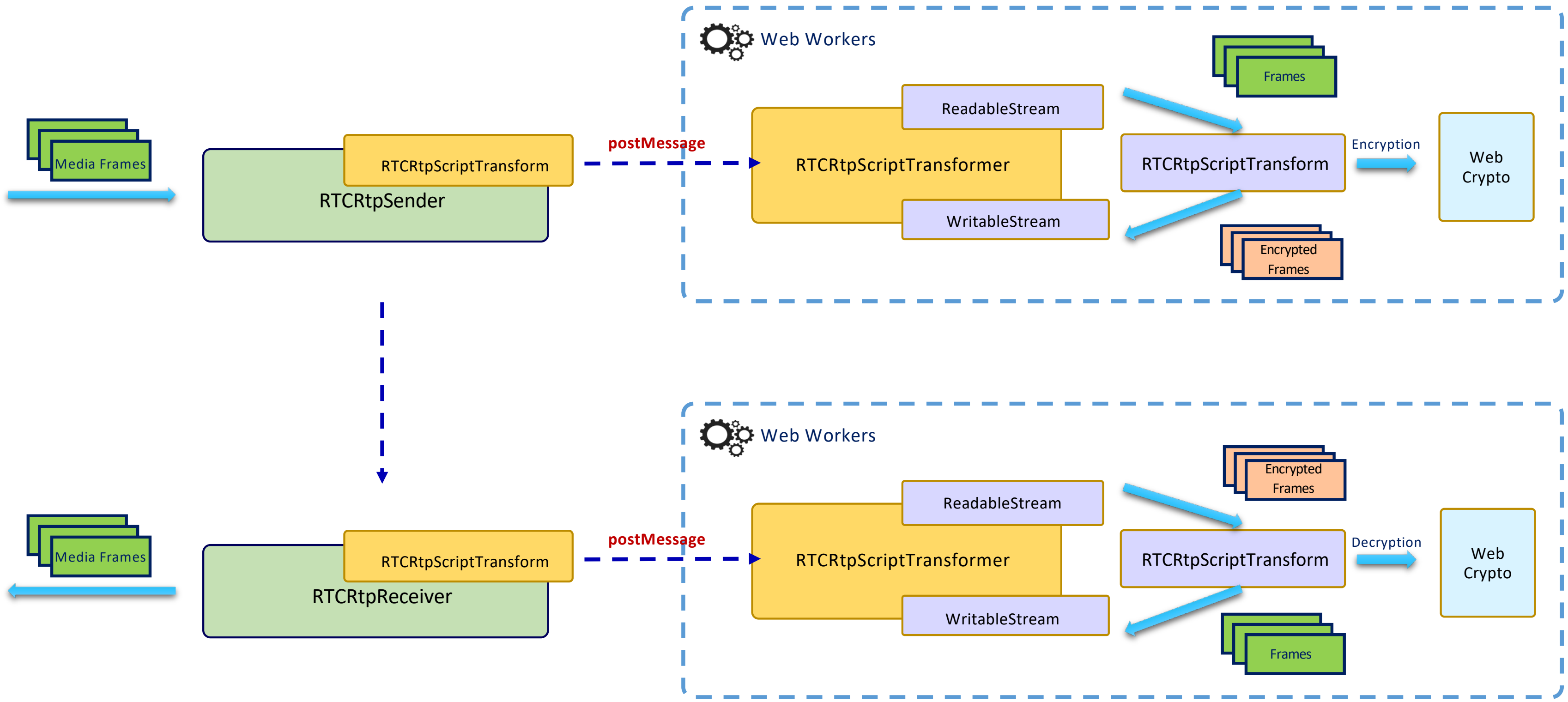
```
partial interface RTCRtpSender {
    Promise<undefined> generateKeyFrame(optional sequence <DOMString> rids);
};
```

```
[Exposed=DedicatedWorker]
interface RTCTransformEvent : Event {
    readonly attribute RTCRtpScriptTransformer transformer;
};

partial interface DedicatedWorkerGlobalScope {
    attribute EventHandler onrtctransform;
};

[Exposed=DedicatedWorker]
interface RTCRtpScriptTransformer : EventTarget {
    // Attributes and methods related to the transformer source
    readonly attribute ReadableStream readable;
    Promise<unsigned long long> generateKeyFrame(optional DOMString rid);
    Promise<undefined> sendKeyFrameRequest();
    // Attributes and methods related to the transformer sink
    readonly attribute WritableStream writable;
    attribute EventHandler onkeyframerequest;
    // Attributes for configuring the Javascript code
    readonly attribute any options;
};

[Exposed=Window]
interface RTCRtpScriptTransform {
    constructor(Worker worker, optional any options, optional sequence<object> transfer);
};
```

声网

Case 1: E2E Encryption

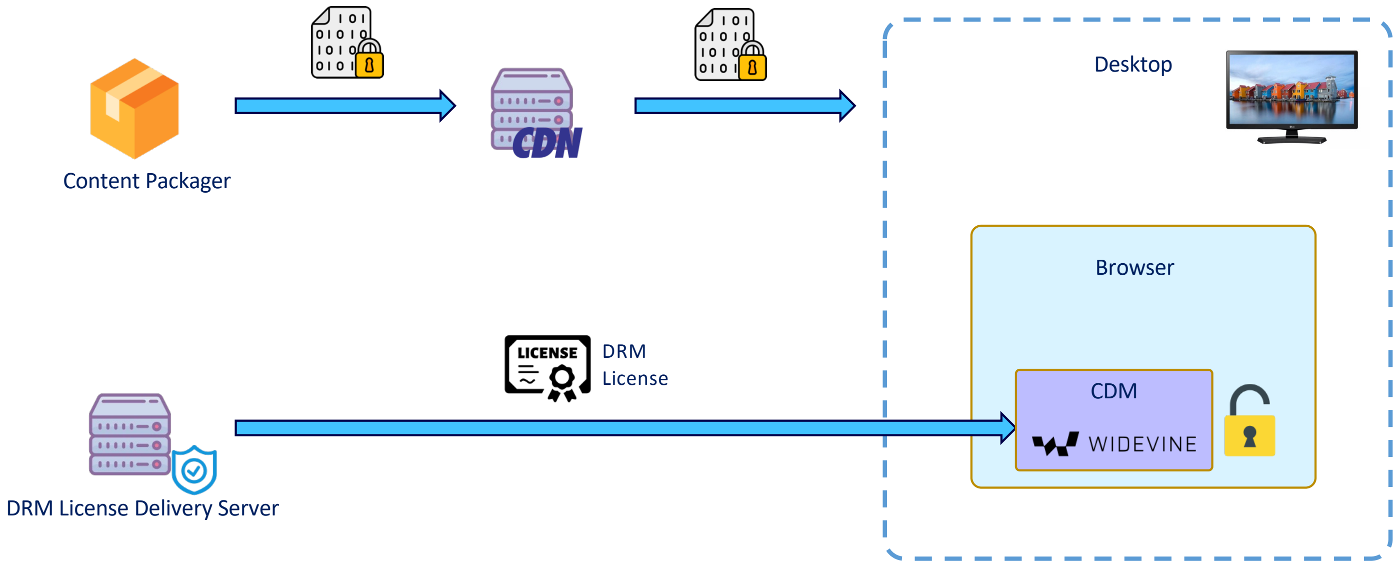
E2E encryption with RTCRtpScriptTransform and Web Crypto API



Case 2: Digital Rights Management

声网

Traditional DRM for CDN one–way media



Case 2: Digital Rights Management

声网

WebRTC Extended Use Cases

Case: Live encoded non–WebRTC media

Requirement ID	Description
N40	An application can create an outgoing WebRTC connection without activating an encoder.
N41	An application can create encoded video frames from encoded data and metadata, and enqueue them on an outgoing WebRTC connection
N42	The WebRTC connection can generate signals indicating the desired bandwidth, and surface those to the application.

Case: Transmitting stored encoded media

Requirement ID	Description
N41	An application can create encoded video frames from encoded data and metadata, and enqueue them on an outgoing WebRTC connection
N42	The WebRTC connection can generate signals indicating the desired bandwidth, and surface those to the application.
N43	The application can modify metadata on outgoing frames so that they fit smoothly within the expected sequence of timestamps and sequence numbers.
N44	The application can signal the WebRTC encoder when resuming live transmission in such a way that generated frames fit smoothly within the expected sequence of timestamps and sequence numbers.

Case: Decoding pre–encoded media

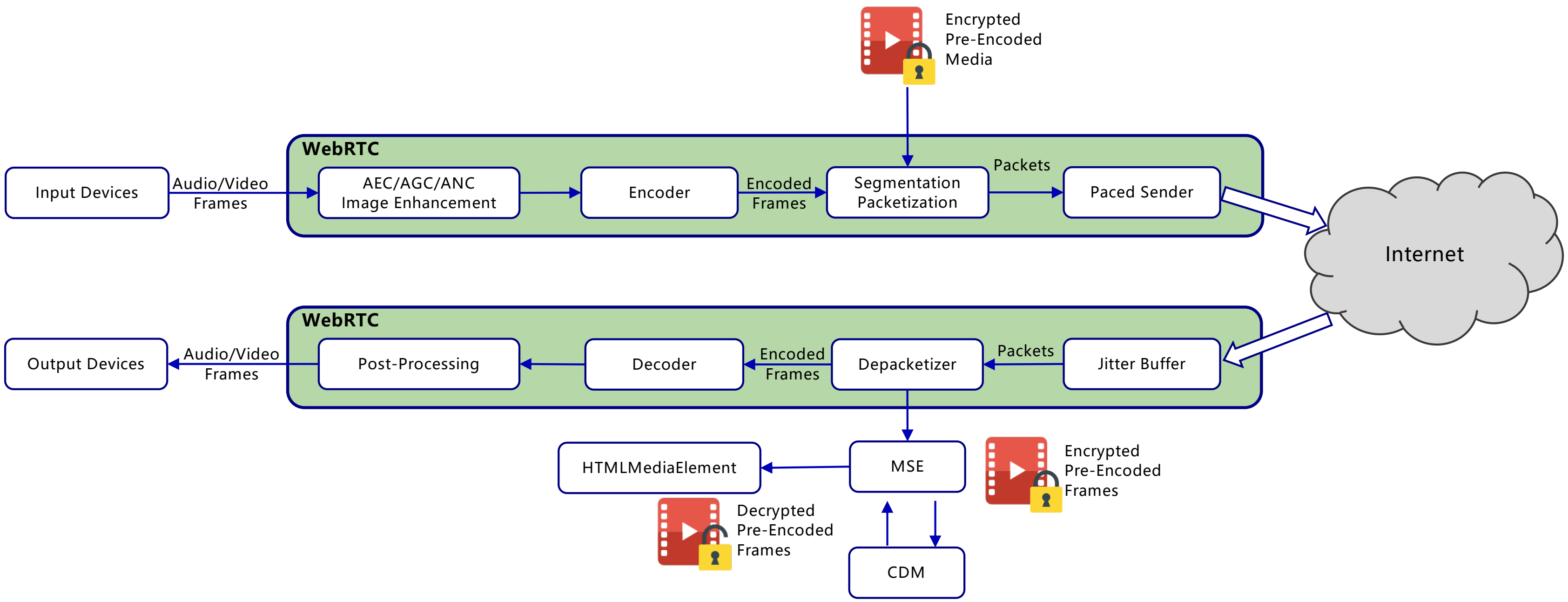
Requirement ID	Description
N45	An application can create an incoming WebRTC connection to accept frames as if they were coming in over RTP, without creating an RTP transprort.
N46	An application can create encoded video frames from encoded data and metadata, and enqueue them on an incoming WebRTC connection.
N47	The WebRTC connection can generate signals indicating demands for keyframes, and surface those to the application.

声网

Case 2: Digital Rights Management

DRM requirements for WebRTC one-way media：

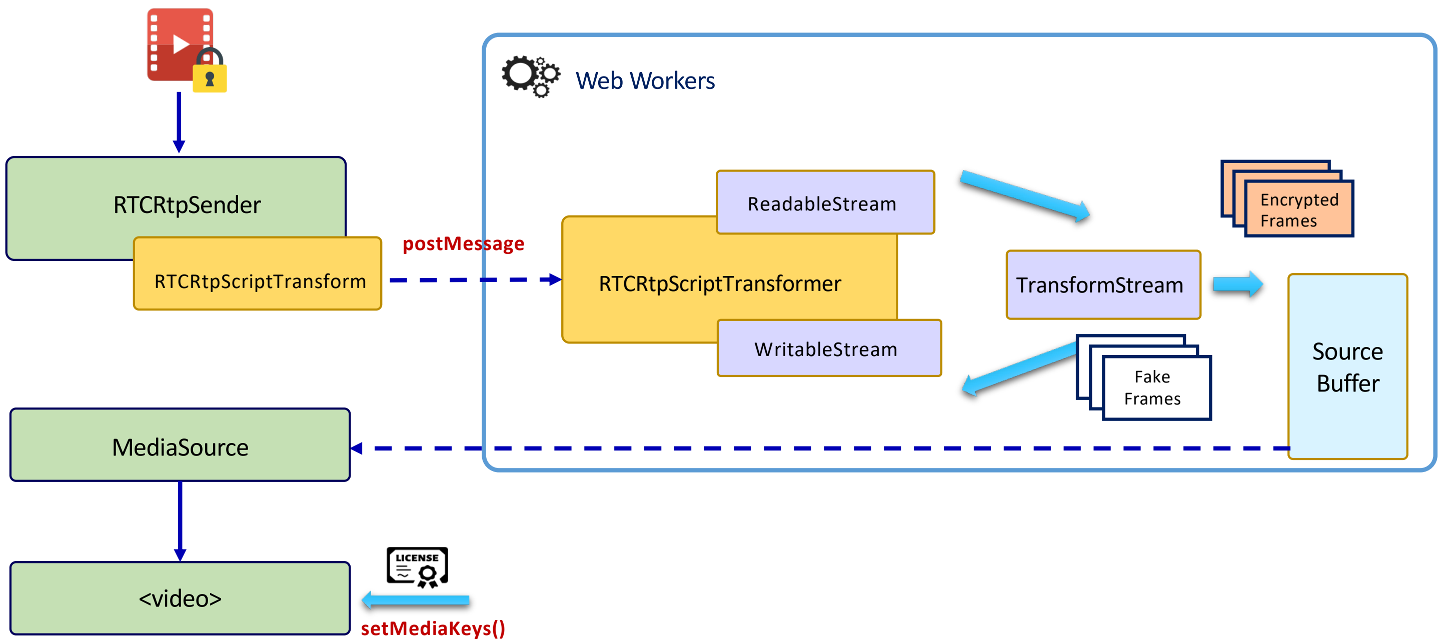
- Transmitting stored pre-encoded media as part of the WebRTC RTP session
- Decrypting media with CDM， and decoding pre-encoded media with MSE.



声网

Case 2: Digital Rights Management

Current DRM solution for WebRTC



声网

Case 3: H265 RTC Supporting

HEVC/H.265 decoding support on Web

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *
		3.1 - 10.1	2 - 119					4		
4 - 106	[1] 12 - 18	[3] 11 - 12.1	[6] 120	10 - 93			3.2 - 10.3	[2] 5 - 20		
[5] 107 - 124	[4] 79 - 123	13 - 17.4	[7] 121 - 125	[5] 94 - 108	6 - 10		11 - 17.4	21 - 23		12 - 12.1
[5] 125	[4] 124	17.5	[7] 126	[5] 109	[1] 11	[5] 124	17.5	24	all	[2] 80
[5] 126 - 128		17.6 - TP	[7] 127 - 129				17.6			

1. Supported only for devices with hardware support
2. Reported to work in certain Android devices with hardware support
3. Supported only on macOS High Sierra or later
4. Supported for all devices on macOS (>= Big Sur 11.0) and Android (>= 5.0) if Edge >= 107, for devices with hardware support on Windows (>= Windows 10 1709) when HEVC video extensions from the Microsoft Store is installed
5. Supported for all devices on macOS (>= Big Sur 11.0) and Android (>= 5.0), for devices with hardware support on Windows (>= Windows 8), and for devices with hardware support powered by VAAPI on Linux and ChromeOS
6. Supported for devices with hardware support (the range is the same as Edge) on Windows in Nightly only. 10-bit or higher colors are not supported.
7. Supported for devices with hardware support (the range is the same as Edge) on Windows only. Enabled by default in Nightly and can be enabled via the media.wmf.hevc.enabled pref in about:config. 10-bit or higher colors are not supported.

声网

Case 3: H265 RTC Supporting

HEVC Encoding support on Web

- Safari 13+ on Mac and IOS
- Chrome for Windows / Mac M109+; Chrome for Android M117+;
  Experimental feature with switch: *--enable-features=PlatformHEVCEncoderSupport*
- Not supported

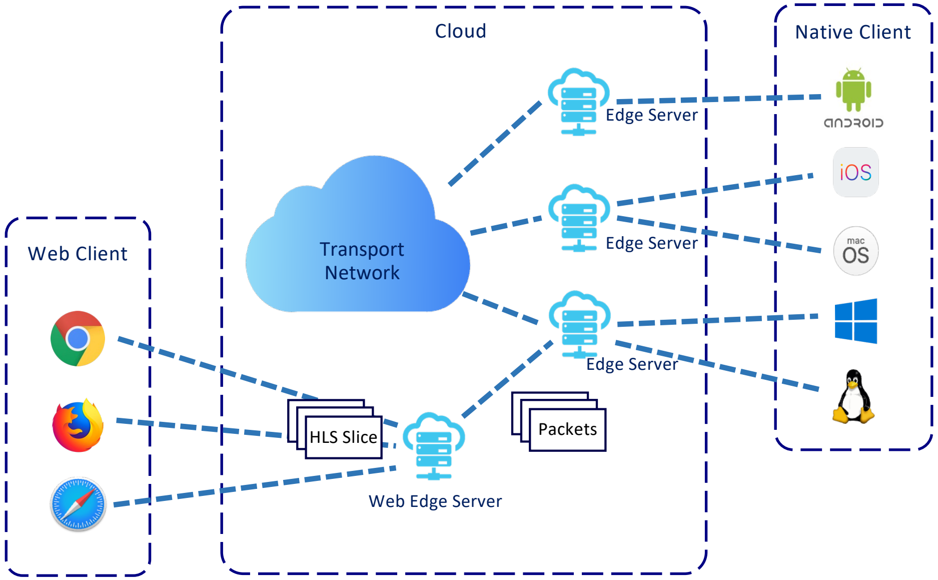
WebRTC HEVC Support

- Experimental feature on Safari 14+
  Not compatible with RFC 7798 Packetization
- Chrome Canary 127
  Experimental feature with switch: *--enable-features=PlatformHEVCEncoderSupport, WebRtcAllowH265Send, WebRtcAllowH265Receive --force-fieldtrials=WebRTC-Video-H26xPacketBuffer/Enabled*
- Not supported

声网

Case 3: H265 RTC Supporting

Solution 1: Remuxing RTP packets to HLS on Server



Pros

- The architecture is relatively simple
- The system has good compatibility

Cons

- Not a real–time system
- Poor resistance to weak network

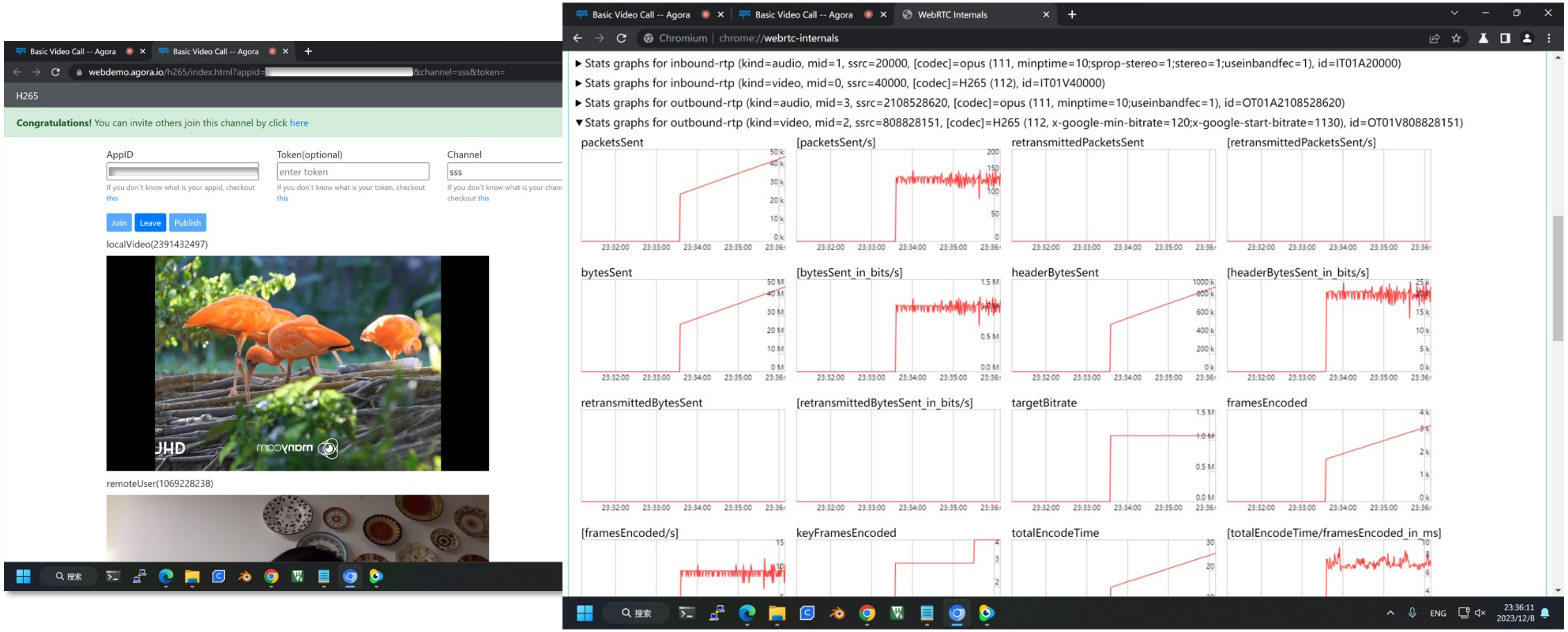
Scenario

Good network quality, do not have strict real–time requirements.

声网

Case 3: H265 RTC Supporting

Solution 2: Customized Browser with WebRTC H265 Support



Case 3: H265 RTC Supporting

Solution 2: Customized Browser with WebRTC H265 Support

Pros

- Pure web technical stack for developer
- Compatible with existed WebRTC apps
- Forward compatibility with future official Chrome browser

Cons

- Higher technical maintenance costs
- Extra efforts for client distribution

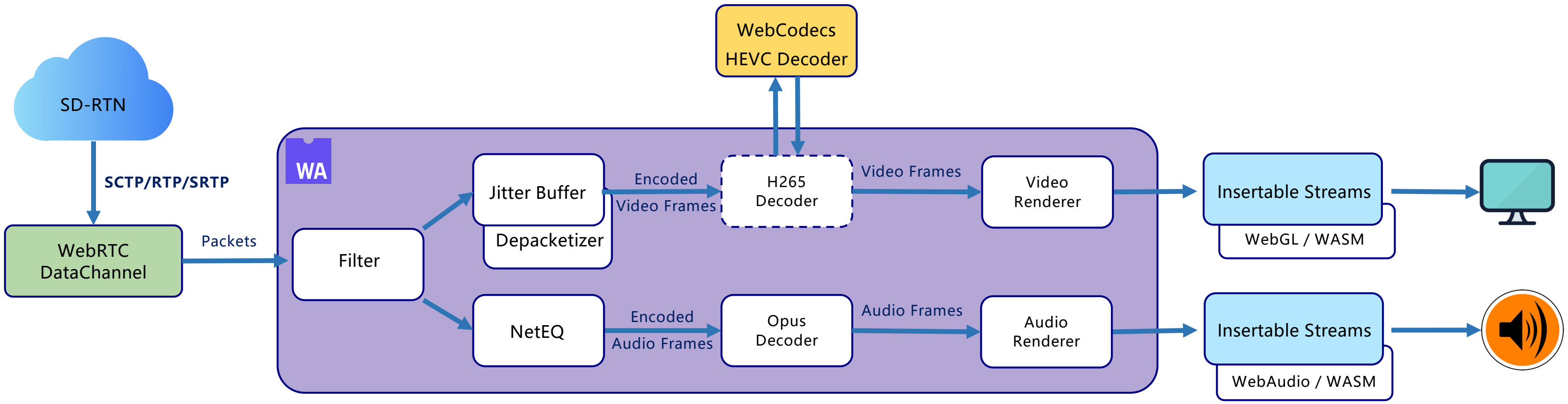
Scenario

Application environment under control in organization like company, bureau, etc…

声网

Case 3: H265 RTC Supporting

Solution 3: Port RTC components to WebAssembly



Features

- Implement full downlink pipeline with WebAssembly. Including bandwidth estimation, jitter buffer, netEQ、video packetizer/depacketizer, media codecs, media renderer.
- Media is transmitted with tuned WebRTC DataChannel

Case 3: H265 RTC Supporting

声网

Protential Future Solution: WebRTC-RtpTransport API

A new proposal being discussed in WebRTC WG

Problem & Motivation

WebRTC APIs is not sufficient, due to:

- Lack of support for custom metadata
- Lack of codec support
- Lack of custom rate control
- Inability to support custom RTCP messages

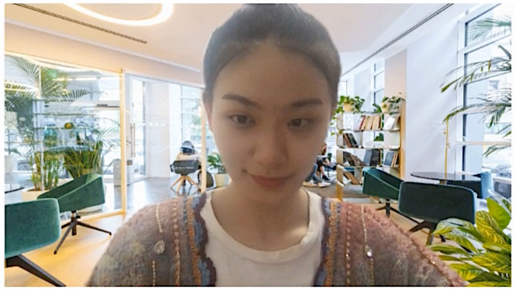
Goal

- Custom rate control (with built-in bandwidth estimate)
- Custom bitrate allocation
- Custom metadata (header extensions)
- Custom RTCP messages
- Custom RTCP message timing
- RTP forwarding
- Custom payloads (ML-based audio codecs)
- Custom packetization
- Custom FEC
- Custom RTX
- Custom Jitter Buffer
- Custom bandwidth estimate

声网

Case 4: Alpha Video Transmission

Scenario: Rendering characters onto virtual background



- Immersive online meeting

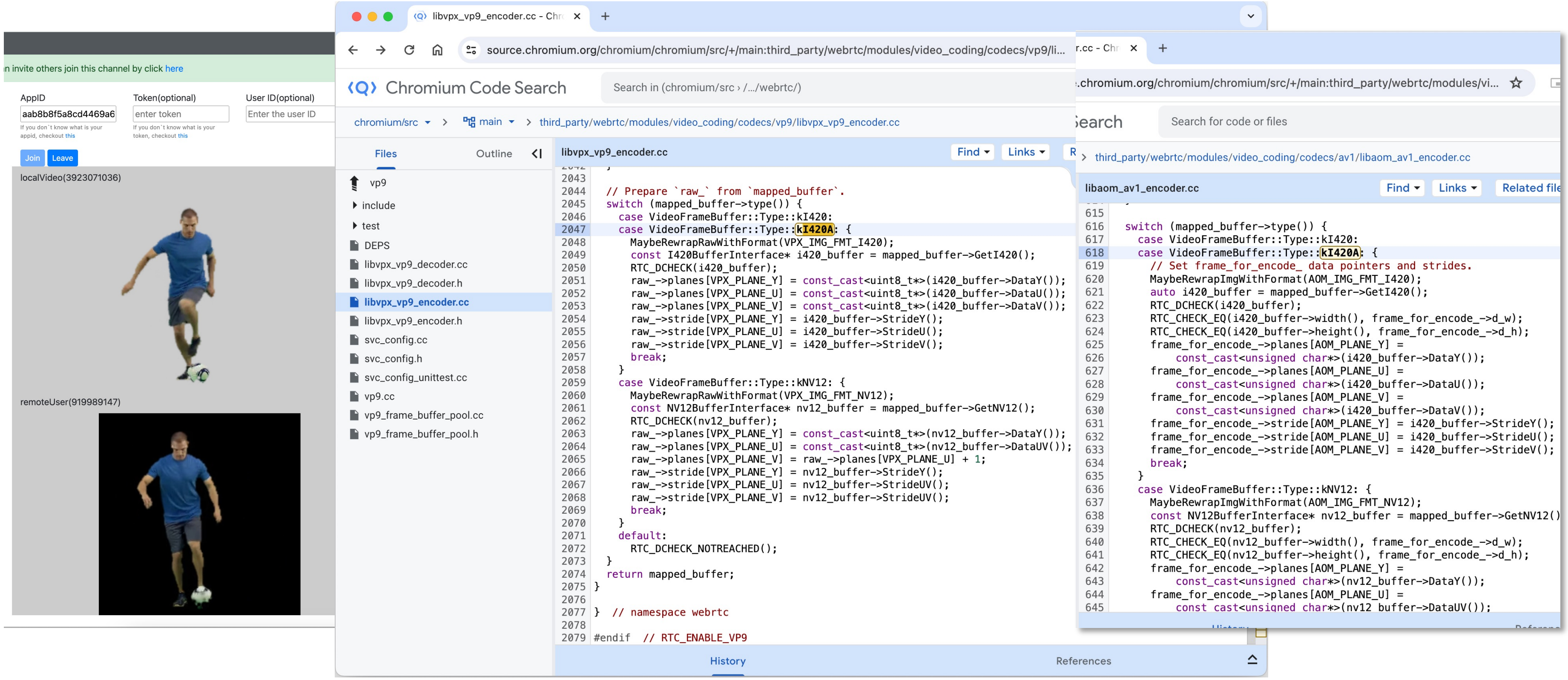
- Picture-in-picture presentation

- Virtual interaction

声网

Case 4: Alpha Video Transmission

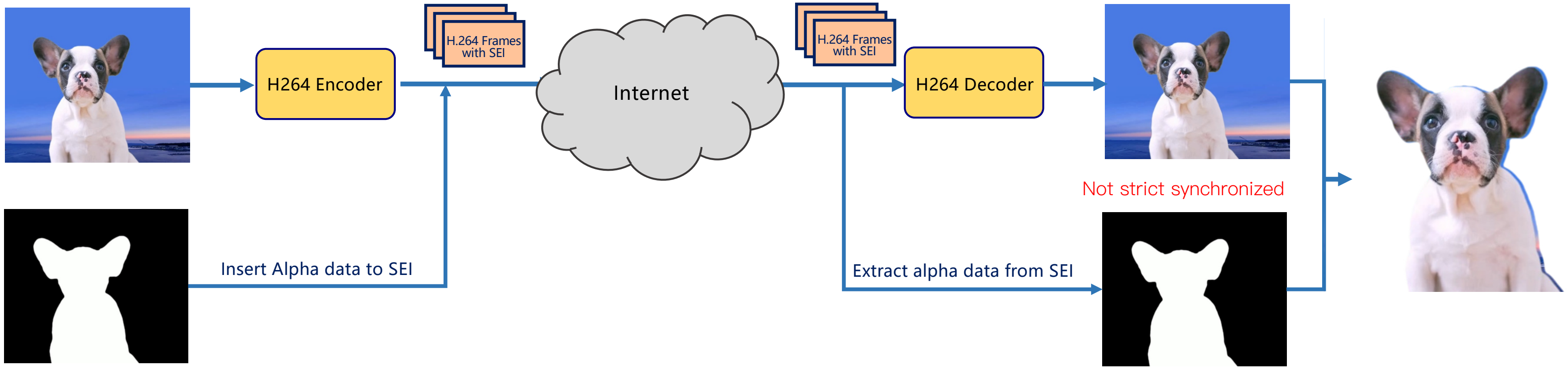
In WebRTC pipeline, alpha plane is ignored even if the encoder support alpha channel



声网

Case 4: Alpha Video Transmission

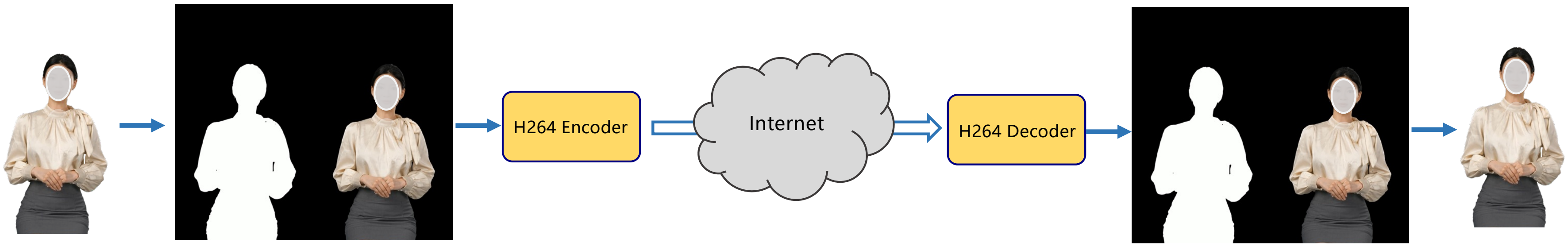
Solution 1: Send alpha data with H.264 SEI



声网

Case 4: Alpha Video Transmission

Solution 1: Store alpha data to the expanded area



- Need extra metadata to record video type (normal or expanded)
- Cost about 20% extra bandwidth

声网

END

Thanks