

SafeCast: End-to-End Security for Ultra-High-Bitrate Broadcast Media

Arman Kolozyan
MPI-SP

Christian Knabenhans
EPFL

Zayd Maradni
MPI-SP

Carmela Troncoso
MPI-SP & EPFL

Abstract

To address the scalability and flexibility limitations of dedicated point-to-point cabling in broadcast production, the broadcasting community created a suite of standards for carrying media over IP multicast. This shift removes the inherent physical protection that dedicated cabling provides, exposing streams to tampering and eavesdropping. Such threats have been widely reported and pose major risks to broadcasters and their audiences. Despite almost a decade of deployment, the security of IP-based broadcast has received little academic attention.

We collaborate with the European Broadcasting Union (EBU) to establish the security and deployment requirements of this environment. We introduce SafeCast, a system to secure the transmission of ultra-high-bitrate multicast media. SafeCast builds on established and standardized protocols (the MLS group key agreement protocol and the SRTP media-protection protocol), while extending and adapting them to the demanding setting of broadcast media. SafeCast provides strong confidentiality and integrity guarantees, and exploits the precise clock synchronization of broadcast networks to achieve cheap per-sender authentication. We evaluate SafeCast against the stringent deployment requirements of production facilities (throughput, latency, large dynamic groups) and find that it can sustain the most common media formats with ample headroom.

1 Introduction

Historically, broadcast facilities relied on point-to-point coaxial or fiber cables for media transport [1, 3]. These point-to-point links offer low and predictable delay, which is valuable in live production [100]. From a security perspective, intercepting or tampering with the signal requires physical access to the cable, thereby ruling out remote attackers [16]. However, cable-based systems scale poorly [4]. Capacity expansion generally requires additional router ports and cabling. For example, implementing a higher-bandwidth format like

4K may require four times as many cables and router capacity. Cabled infrastructures are also less adaptable to future formats because they are constrained by the capabilities of the installed hardware. In the past decade, the industry has therefore been moving to IP-based transport, standardized as ST 2110 [5, 7]. The standard carries video, audio, and ancillary data such as captions as separate streams over IP multicast (i.e., one transmission to all subscribed receivers). A production can then easily switch between video streams while the commentators' audio continues unchanged. Alongside ST 2110, the Networked Media Open Specifications (NMOS) provide the IP-based control plane for discovering devices and managing their connections [18, 21].

No protection for media streams. An ST 2110 deployment is split between a control and a media plane. The control plane is secured by a dedicated security framework [31] (including TLS encryption [29, 30, 108] and per-request authorization [32]). However, ST 2110 explicitly states that securing video and audio transport is not in its scope [29]. This means that there is no mechanism to verify that the packets have not been tampered with, or to hide their content. With no media-plane protection in place, production teams resort to producing shows behind closed doors and binding staff by NDAs [70, 77], while facilities rely on network segmentation, isolating the media network (e.g., from office IT), even though broadcast networks are rarely truly isolated [81]. Our work focuses on providing means to protect the media plane itself.

Integrity threats. Attackers have repeatedly shown interest in manipulating broadcast content by airing political messages [47, 53], adult content [96], and false emergency warnings [126]. With IP-based transport, tampering with media no longer requires physical access. For example, a European Broadcasting Union (EBU) study of 68 broadcast devices found hundreds of vulnerabilities, which an attacker can exploit to enter the network [63, 81].

Confidentiality threats. Eavesdropping is a serious threat in broadcast settings. Some content is shot well ahead of airing, so its leakage can carry financial as well as legal conse-

quences. For instance, rehearsal footage of Eurovision 2025 was leaked online. Containing the spread required involving lawyers and issuing takedown requests across platforms to preserve the surprise for millions of viewers [91]. Similar incidents occurred with Olympic ceremony shots [48] and New Year’s Eve shows [92]. Further, raw streams carry everything the cameras and microphones capture, including unblurred faces, off-camera moments, and “hot mic” recordings. Leaks of such material have had severe consequences, including the cancellation of a long-running television series [65, 109]. Finally, the ST 2110 standard is spreading beyond television production into military [110], government [54, 94], health-care [22, 61], and corporate [71] environments, where the stakes are inherently high.

Availability threats. While confidentiality and integrity threats apply broadly to networked systems, availability is particularly critical in live broadcast. Live events are time-bound: lost moments cannot be re-produced, and failures are immediately visible to millions of viewers, causing significant reputational and financial damage. On the media plane, an attacker can cause a denial of service (DoS) by injecting forged packets [113, 114] or replaying previously recorded ones [122]. Availability also limits the set of applicable countermeasures: mechanisms for confidentiality and integrity must have negligible influence on latency and keep up with the ultra-high-bitrate media throughput to not be disruptive. This poses two challenges: keys must be efficiently established and updated, and every packet must then be protected under those keys at media rates.

Contributions. Our work is a firm first step towards protecting broadcast production environments, which have received little attention from the security community. Informed by interactions with the EBU and the Internet Protocol Media Experience (IPMX) working group [24], as well as publicly reported cyberattacks against broadcast organizations, we define a threat model for the IP-based broadcast ecosystem (§3.1), and derive functional, security, and performance requirements for a media-plane security mechanism (§3). We show that the only standardized protocol for protecting ultra-high-bitrate broadcast media [26] falls short of meeting most of these requirements. We then introduce SafeCast (§5), which builds on the Messaging Layer Security (MLS) group key agreement protocol and the Secure Real-time Transport Protocol (SRTP) media-protection protocol – the most suitable building blocks for our scenario (§4) – and extends and adapts them to the demanding setting of broadcast media. SafeCast includes within-epoch keying and cheap delayed authentication tailored to the broadcast setting, thereby achieving provable confidentiality, forward secrecy, post-compromise security, integrity, and per-sender authentication. We implement SafeCast, evaluate it on realistic data streams, and show that it protects uncompressed 4K video on a single CPU core with ample headroom (§6).

2 Background on IP-Based Broadcast

An end-to-end IP-based broadcast workflow, depicted in Fig. 1, consists of three stages: capture, production, and distribution. At the event venue, capture devices such as cameras and microphones record the raw signals and pass them over dedicated links to IP-capable equipment such as camera control units (CCUs). In newer cameras, this IP capability may also be integrated into the camera itself. Production then takes place in one of two settings. Either at the event venue in mobile production control rooms, known as outside broadcast (OB) trucks [40, 93]; or by sending the media over long-distance links to a central facility of the broadcaster, where the production team operates while only a small crew remains at the venue, known as the *remote integration (REMI) model* [95, 124]. We refer to either production environment as a *facility*, where a set of media devices is connected to a shared local network. In both settings, the media travels through the facility’s network to devices such as the sound mixer, which combines the audio streams (e.g., a commentator’s voice over the stadium sound). A gateway at the production facility then sends the produced program to the broadcaster’s headquarters, which distributes it to viewers.

Point-to-point links between facilities use specialized transport protocols [19, 116], and delivery to viewers uses general streaming protocols [39, 58, 74, 102] – all of these, depicted as green arrows in Fig. 1, come with established protection mechanisms. During production, the media travels as streams specified by *ST 2110* [7], a suite of standards for transporting media over managed IP networks. These streams (depicted as red arrows) are unprotected, and hence the focus of our work.

Parties. An ST 2110 broadcast system consists of media devices that exchange *streams*, which are sequences of packets that each carry one media component, such as video or ancillary data (e.g., subtitles) [27]. Whereas these components were traditionally bundled, ST 2110 transports them as separate streams, with a dedicated specification per media type [14, 17, 20]. A device exposes a *sender* endpoint for each stream it produces and a *receiver* for each stream it consumes: a microphone, for example, exposes an audio sender, while a monitor exposes a video receiver. Connections between senders and receivers are managed by a *controller* – software running on a PC [34, 64, 98, 118] – which discovers them through a *registry* (also software) that maintains a global view of all registered devices and their endpoints [18]. A broadcast engineer typically configures the controller before each event, and the production crew operates it during the broadcast.

Discovery and Connection Management. Devices register themselves and their endpoints with the registry via a registration API [18] defined by the Networked Media Open Specifications (NMOS). The controller queries the registry to discover these devices and their endpoints. To set up a connection, the controller configures a receiver with the information it needs to receive a sender’s stream [21]. To this end, each

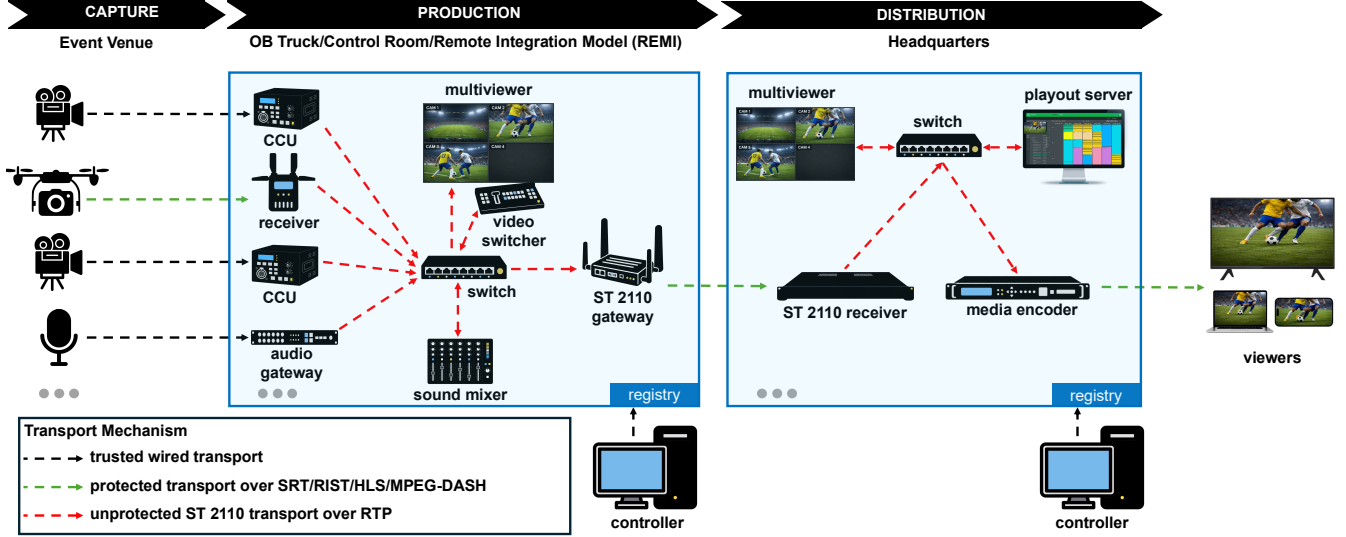


Figure 1: End-to-end IP-based broadcast workflow. Media is captured at the event venue, mixed into the final program in an OB truck or (remote) control room, and distributed to viewers from the broadcaster’s headquarters. Within the production and distribution facilities, media travels as unprotected ST 2110 streams (depicted in red). The links between sites and to viewers are protected (in green).

sender describes its stream in a *Session Description Protocol (SDP) transport file* [42], which the controller retrieves from the sender and passes to the receiver.

Media Distribution over IP Multicast. All media devices in a facility connect through switches to a local IP network dedicated to media. This network is managed by the facility, shared by all media streams, and ideally isolated from the facility’s other networks (e.g., office IT). While our work is agnostic to the network’s internal topology, we describe a typical one in App. A. Streams travel over IP multicast: rather than sending every receiver its own copy, the sender transmits each packet once, and the switches replicate it to all interested receivers. More specifically, each stream has its own *multicast address*; a receiver subscribes to a stream by sending its switch a join message for the multicast address it obtained from the controller [49]. This subscription is unauthenticated: any device on the network can subscribe to any stream, and the sender does not know the subscribers.

Transport Protocols. Senders transmit their streams using the Real-time Transport Protocol (RTP) [115], the standard transport protocol for real-time media over IP networks. RTP runs over UDP [107], which supports multicast and low-latency delivery, but provides no delivery guarantees (i.e., packets may be lost or arrive out of order). Each RTP packet carries a header with the information receivers need for playback, such as a sequence number and a timestamp. The RTP Control Protocol (RTCP) [115] accompanies each RTP stream with control traffic: receivers periodically report reception quality (e.g., packet loss) for monitoring, while messages such as BYE announce that a sender is leaving the session. Neither RTP nor RTCP provides any built-in security.

Time Synchronization. Since ST 2110 carries video and audio separately, receivers need a common clock to align their playback. The Precision Time Protocol (PTP) [12, 13] provides sub-microsecond synchronization via a grandmaster that distributes a shared clock to all devices on the network.

3 Threat Model and Requirements

We define a threat model for IP-based broadcast ecosystems and derive functional, security, and performance requirements for a media-plane security mechanism. Our analysis draws on the security needs of the different environments in which the ST 2110 standard is deployed, publicly reported incidents, and interactions with the European Broadcasting Union (EBU) and its member organizations, as well as engineers in the Internet Protocol Media Experience (IPMX) working group.

3.1 Threat Model

Since media travels over IP multicast, an attacker who gains access to the network may be able to subscribe to streams, redirect them, inject unauthorized traffic, or disrupt delivery without ever touching a physical media cable.

Adversary Capabilities. We consider an adversary with access to the facility network. The adversary may record RTP and RTCP traffic to recover media content or to learn sensitive metadata [44, 115], and interfere with traffic by injecting, modifying, or replaying RTP or RTCP packets. Additionally, the adversary may corrupt endpoint devices such as senders and receivers, thereby learning the key material they hold; and may compromise switches, thereby gaining the ability

to observe and modify the traffic they forward. Note that we assume administrators and the controllers they operate are not malicious. These parties hold privileged control over provisioning and connection management, so any protection collapses if they act dishonestly.

Scope. We do not consider the security of the control plane, as it consists of client-server connections covered by standard transport security, and is already standardized [29–32] and implemented [123]. We also leave volumetric denial of service (DoS) out of scope. On the internet, saturation is handled by congestion control, which makes honest senders reduce their transmission rate. The UDP guidelines permit traffic without congestion control only in controlled environments over provisioned capacity [60], which is the setting of IP broadcast facilities. Here, a new stream is admitted only when sufficient capacity exists, with each endpoint limited to its authorized bandwidth [6, 55]. Network-level DoS protection is thus already a routine operational mitigation.

3.2 Functional Requirements

We identify the following functional requirements for a production-side broadcast environment. This setting differs from typical consumer media streaming in fundamental ways, including the network infrastructure and delivery mechanisms (see App. B for a detailed comparison), which constrain the security mechanisms that can be used.

F1 Support for IP multicast. The ST 2110 standard uses multicast as its transport model, which means that the network replicates the stream to all subscribed receivers. This one-to-many delivery must be preserved, so that every receiver can decrypt and verify the same packets.

F2 Support for mid-event device changes. The broadcast environment is inherently dynamic. The control plane enables dynamic discovery and connection management, and the system automatically removes devices that stop sending periodic heartbeats to the registry [18]. This dynamism extends to the physical level, as cameras, microphones, or even outside broadcast (OB) trucks join for the duration of an event and disconnect afterwards – and might be replaced mid-event in case of failure. There is hence a need for efficient granting and revocation of access to the media streams during live operation.

F3 Compatibility with dual-path redundancy. To protect against failures, IP broadcast deployments commonly transmit two identical copies of the same stream over separate network paths [10]. The receiver then takes, for every packet, whichever copy arrives first [43]. A security mechanism must therefore protect both copies identically, so that receivers can still match them.

F4 Tolerance to packet loss and reordering. RTP runs over UDP and therefore provides no delivery guarantees. In a live broadcast network, packets may be lost and may arrive out of order due to jitter and delay differences between the two redundant paths (F3). A security mechanism must tolerate both conditions.

F5 Deployability. The mechanism should rely on mature building blocks whose specifications are finalized, and should fit into existing broadcast infrastructure without hard prerequisites. This rules out mechanisms that require installing keys on every device in advance (e.g., pre-shared keys (PSKs), whose rotation or revocation means touching every device again), as well as mechanisms that carry secret keys in control-plane messages, which would tie media confidentiality to the protection of the control plane.

3.3 Security Requirements

A mechanism must provide the following security properties:

S1 Integrity. It must not be possible for an attacker to modify or forge RTP packets without being detected. Integrity protection for RTCP is equally critical, because RTCP controls the behavior of RTP sessions. Without it, an attacker could, for example, inject a forged RTCP BYE packet to trick receivers into believing the sender has left the session, causing them to stop playback.

S2 Confidentiality. Media content must be accessible only to authorized devices: an attacker observing network traffic must not be able to recover it. The same holds for the metadata that RTCP carries, such as participant identifiers and quality metrics.

S3 Replay protection. It must not be possible for an attacker to replay legitimate packets without detection.

S4 Forward secrecy. If an attacker obtains the current key material, they must not be able to recover media transmitted before the compromise. This is a realistic concern: as explained in §2, any device on the network can subscribe to a multicast address and passively record encrypted traffic. If such recorded traffic is later combined with a key compromise, all past media would be exposed without forward secrecy.

S5 Post-compromise security. After an adversary’s access to a compromised device’s state ends, the system must be able to recover, so that the adversary cannot access media transmitted afterwards.

S6 Device exclusion security. When a device is removed (e.g., after theft or failure), it must lose access to all subsequent media.

3.4 Performance Requirements

The security mechanism must also meet the following stringent performance constraints:

- P1 Production-rate throughput.** In a broadcast production, media is transported uncompressed [14] as compression algorithms introduce unacceptable latency for live operation. Encryption and decryption must keep up with the data rate of the media streams to not disrupt availability. According to data provided to us by the EBU, these rates can reach 2.6 Gbps for a single uncompressed 1080p video stream at 60 fps, or 40 Gbps for 8K. Audio streams are comparatively small (typically around 20 Mbps), so video dominates the throughput requirements. Additionally, devices need to handle multiple streams in parallel (e.g., cameras send a few streams [101], while multiviewers consume tens of them at once [66]). Encryption and decryption must support the aggregate rate of all streams a device handles.
- P2 Low added latency.** In live production, every packet must meet its playout deadline. A security mechanism must therefore add negligible latency for every packet, not only on average.
- P3 Low packet expansion.** RTP packets in the broadcast setting are sized close to the Maximum Transmission Unit (MTU), the largest packet a link can carry without fragmentation: UDP datagrams may take up to 1460 bytes on standard Ethernet links [13, 14, 57], and up to 8960 bytes on links supporting jumbo frames [2]. To avoid increasing the number of packets, and thus the consumed bandwidth, a security mechanism must add only a minimal number of bytes to each packet.
- P4 Scalability to large deployments.** Production facilities vary widely in size. Our interactions with broadcasters revealed deployments ranging from small studios with a handful of devices to large broadcast centers with hundreds of devices. Public reports also reflect this scale. Large broadcast events commonly involve multiple OB trucks, from a few for a single event [85] to over fifty for multi-venue events such as the Olympics [103]. A single truck can contain well over 100 pieces of equipment, including monitors, replay servers, and audio consoles [40, 93]. These trucks process streams from on-site capture devices, which can range from dozens to several hundred for a single large event [11, 119], and reach thousands across multi-venue events [68, 69]. In particular, both granting or revoking devices’ access to streams (F2, S6) and refreshing keys for forward secrecy (S4) and post-compromise security (S5) happen during live operation, so these operations must complete within milliseconds even at such scales.

4 Protocols for Secure Multicast at Scale

We identify suitable protocols for building a system that meets the requirements of the previous section.

Limitations of existing broadcast security systems. To the best of our knowledge, the Privacy Encryption Protocol (PEP) [26] is the only standardized security protocol applicable to broadcast media. It was developed as part of Internet Protocol Media Experience (IPMX), a suite of standards for audio and video over IP that builds on ST 2110. We learned from communication with the specification authors that PEP was designed with media confidentiality as its primary goal, under two constraints that shaped many of its design choices: targeting the professional audio-visual market (e.g., conference rooms and lecture halls), and maintaining backward compatibility with existing IPMX devices. The first constraint requires PEP to be implementable on devices that are far cheaper and less capable than broadcast equipment (e.g., hundreds of dollars, compared to tens of thousands for broadcast equipment). The second constraint ties PEP’s design to High-bandwidth Digital Content Protection (HDCP), a scheme that encrypts video links such as HDMI to prevent unauthorized copying during transmission [8]. As a result, IPMX devices already implement HDCP’s cipher, AES-128-CTR, in hardware – which PEP reuses [25]. To obtain authentication, PEP computes a CMAC via a second AES pass over each packet, forcing the encryption hardware to run at twice the media rate (as duplication would increase the cost of target devices). In PEP, integrity protection is optional and replay protection is absent. RTCP is left entirely unprotected, allowing forged control messages such as spoofed BYE packets and exposing metadata to passive attackers. Key management relies on statically provisioned pre-shared keys (PSKs): there is no revocation mechanism or post-compromise security, and forward secrecy is offered only for peer-to-peer settings. Moreover, PSK provisioning is left to vendor-specific mechanisms. Finally, PEP’s optional authentication uses a MAC-then-encrypt construction, which is more error-prone than standard AEAD designs [82]. Thus, PEP does not fulfill the requirements in §3. We give a full security analysis of PEP in App. D.

Towards a secure broadcast system. Since in multicast a stream is transmitted once and received by all its subscribers (F1), secure broadcasting requires secure group communication among hundreds of devices. Pairwise key establishment, as used in point-to-point secure communication [111], does not scale to groups of this size. To design our own scheme tailored to the broadcasting scenario, we decompose the problem into a *key establishment* layer that establishes, updates, and revokes the cryptographic keys held by the group; and a *transport security* layer that uses these keys to encrypt, authenticate, and transmit the media over the network. In the following, we survey existing relevant protocols for each layer (described in detail in App. C) against the requirements of §3, and select the two best-fitting building blocks.

4.1 Key Establishment

We consider standardized protocols used to establish keys for real-time media (SDS [38], DTLS [112], MIKEY [97], and ZRTP [50]), the mechanism defined for the ST 2110 standard (PEP) [26], and the MLS protocol for secure group key management [41]. We summarize in Table 1 how these protocols fare against the requirements relevant to key establishment (§3). DTLS, SDS, and ZRTP lack the group support required for multicast. MIKEY and PEP support groups but offer forward secrecy only for peer-to-peer connections, provide no post-compromise security or efficient dynamic membership, and require manual handling for device exclusion. Only MLS meets all of our requirements.¹

Selected protocol: MLS. MLS is a group key establishment protocol designed for large, dynamic groups with up to thousands of members [41]. MLS achieves efficient group updates through a ratcheting mechanism that organizes participants in a binary tree [46]. The group advances through a sequence of *epochs*, where each *commit* message applies changes such as adding or removing a member and derives fresh key material for the new epoch. MLS provides both forward secrecy (FS) and post-compromise security (PCS). The protocol relies on two infrastructure services: an authentication service (AS) that issues and validates credentials (trusted), and a delivery service (DS) that serves as a directory for initial keys and routes protocol messages (untrusted) [45]. MLS also exposes a key exporter interface that allows derived keying material to be used by external protocols.

4.2 Media Transport

We consider the Secure Real-time Transport Protocol (SRTP), the standard for protecting RTP media, alongside two more recent alternatives (RTP over QUIC [62] and SFrame [99]) and the mechanism defined for the ST 2110 standard (PEP). We summarize in Table 2 how these protocols fare against the requirements relevant to media transport (§3). RTP over QUIC cannot use IP multicast and remains an unfinished draft. SFrame and PEP support multicast but leave RTCP unprotected and provide no replay protection, and PEP provides only confidentiality. Only SRTP meets all of our requirements.

Selected protocol: SRTP. SRTP provides confidentiality, integrity, and replay protection for RTP and RTCP traffic [52]. SRTP derives cryptographic keys from a master key and a master salt (a random value used during key derivation to make precomputation attacks harder). SRTP also supports authenticated encryption modes such as AES-GCM [87]. Each device maintains per-stream cryptographic state, including keys and a replay-protection sliding window that tracks seen packet indices. SRTP protects each packet independently, so a receiver can process a packet even when neighboring packets are lost

or reordered (F4). This protection takes no path-specific input, so the redundant copies of a packet carry identical ciphertext (F3). SRTP optionally supports a Master Key Identifier (MKI) appended to each packet, which tells the receiver which key was used to protect that packet. This is particularly useful when multiple master keys are valid at the same time, for example during a rekeying transition where some in-flight packets are still encrypted under a previous key (F4). Note that SRTP leaves the fields of the fixed RTP header (e.g., sequence number) in the clear; this does not conflict with our requirements, as these fields are considered minimally sensitive, and receivers need them for correct processing [52, 115] and dual-path matching (F3). Existing header-encryption mechanisms hence cover only header extensions [86, 120]. SRTP does protect their integrity by including them in its authenticated data. They can additionally be encrypted between network hops with link-layer encryption (e.g., MACsec [9]).

5 End-to-End Protection for Broadcast Media

We introduce SafeCast, a system to protect broadcast media end to end, from the sending device to every receiver. It builds on MLS for group key management and SRTP for media-plane protection. MLS’s messaging layer is tailored to discrete messages and does not fit our continuous stream of media. Thus, we use MLS solely for key establishment and management, a functionality close to what is known as continuous group key agreement (CGKA) [36, 37] but that also includes an exporter interface crucial for our application. When we refer to MLS, we mean the protocol without this messaging layer. The MLS *group* in our setting consists of the devices with access to the production’s media, as well as the controller, which manages the membership. In a real-world deployment, one could form finer-grained groups, each with its own controller (e.g., one per outside broadcast (OB) truck); we consider a single group covering the whole production, a lower bound for scalability. We combine this key agreement with SRTP, which protects both RTP and RTCP traffic over multicast with authenticated encryption. We make two adjustments to SRTP for the broadcast setting. First, SRTP was not designed for frequent master key changes: implementations rekey by rebuilding the stream, which resets security-relevant state such as the replay window. We instead adapt the SRTP layer to swap keys in place. Second, SRTP permits shortened authentication tags to save bandwidth [52], but at a broadcast’s packet rate of millions per second, such short tags would allow for successful forgery within hours; SafeCast therefore uses the full 16-byte AES-GCM tag.

To strengthen the forward secrecy guarantees, we extend this composition with a *within-epoch key ratchet* (§5.1), advancing the media key many times within one epoch. Finally, MLS only provides group authentication, and traditional per-sender authentication is too expensive for the broadcasting setting. SafeCast provides an optional per-sender authentica-

¹The one advantage of the keying mechanism of PEP over MLS is its standardized integration with the broadcast control plane [33].

Table 1: Comparison of key-management schemes for establishing and distributing keys. A full circle (●) means the property is supported, and an empty circle (○) means it is not. “P2P only” means the property holds only for peer-to-peer sessions. “N/A” indicates that the property does not apply because the protocol lacks group support. “Mode-dep.” indicates that the property holds only in certain operational modes. “No pre-shared secrets” indicates whether the protocol can establish keys without all parties sharing an out-of-band secret in advance. “Independent of control plane” indicates whether the protocol’s security holds even when the signaling channel is not already integrity- and confidentiality-protected (e.g., via TLS).

Category	Property	DTLS	SDS	ZRTP	MIKEY	PEP	MLS
Broadcast fit	F1: Group support	○	○	○	●	●	●
	F2: Efficient dynamic membership	N/A	N/A	N/A	○	○	●
Key security	S4: Forward secrecy	Mode-dep.	○	●	P2P only	P2P only	●
	S5: Post-compromise security	○	○	○	○	○	●
	S6: Device exclusion	N/A	N/A	N/A	Manual	Manual	●
Deployment	F5: No pre-shared secrets	Mode-dep.	●	●	Mode-dep.	○	●
	F5: Independent of control plane	○	○	●	●	○	●

Table 2: Comparison of data-plane protection mechanisms for RTP/RTCP. A full circle (●) means yes, an empty circle (○) means no, and a half-filled circle (◐) means that the property is supported but not mandatory.

Category	Property	RTP over QUIC	SFrame over RTP	PEP (data transport)	SRTP
Broadcast fit	F1: Multicast support	○	●	●	●
Content protection	S1: RTP integrity	●	●	◐	●
	S1: RTCP integrity	●	○	○	●
	S2: RTP confidentiality	●	●	●	●
	S2: RTCP confidentiality	●	○	○	◐
Traffic protection	S3: Replay protection	●	○	○	●
Key agility	F4: Per-packet key identification	○	●	●	●
Transport resilience	F3: Dual-path compatibility	○	●	●	●
	F4: Packet-loss tolerance	●	●	●	●
Standardization	F5: Specification finalized	○	●	●	●

tion layer via delayed key disclosure (§5.2).

Fig. 2 gives an overview of SafeCast. MLS rekeys the group every epoch and exports a fresh per-stream secret. SafeCast’s within-epoch key ratchet lets devices turn this secret into a sequence of short-lived SRTP keys key_g for the current *generation* g . A sender encrypts each packet under key_g . The receiver decrypts it under the matching key (and applies SRTP’s replay check). With per-sender authentication enabled, each packet additionally carries a MAC under the sender’s current interval key K_i and discloses the key K_{i-d} from d intervals earlier. We describe the full protocol flow of SafeCast in §5.3, and we formally prove SafeCast’s security in §5.4.

5.1 Within-Epoch Keying

To shrink the media that could be exposed in case one epoch key is compromised, SafeCast feeds the output of the MLS exporter through a key ratchet before it reaches SRTP. The

key advances (i.e., one HKDF ratchet step) per frame or per packet, depending on the configuration.

Per-stream seed. Each epoch, the MLS exporter provides SafeCast with a seed, the root secret we use to derive all of the stream’s keys for that epoch,

$$S_0 = \text{MLS.Exporter}(\text{"SRTP ratchet seed"}, SSRC, 32),$$

using as context the stream’s SSRC, an identifier in the stream header used to tell concurrent streams apart [115]. Different streams yield different seeds, so senders never encrypt under the same (key, nonce) pair.

Per-generation keys. From S_0 , an HKDF ratchet produces one SRTP key per generation $g = 0, 1, \dots$. On entering generation g , SafeCast derives two values from the secret S_g :

$$key_g \parallel salt_g = \text{HKDF.Expand}(S_g, \text{"srtp key"}, 28),$$

$$S_{g+1} = \text{HKDF.Expand}(S_g, \text{"next generation"}, 32).$$

The first output is generation g ’s SRTP key material, a 16-byte encryption key key_g and a 12-byte salt $salt_g$ that is mixed

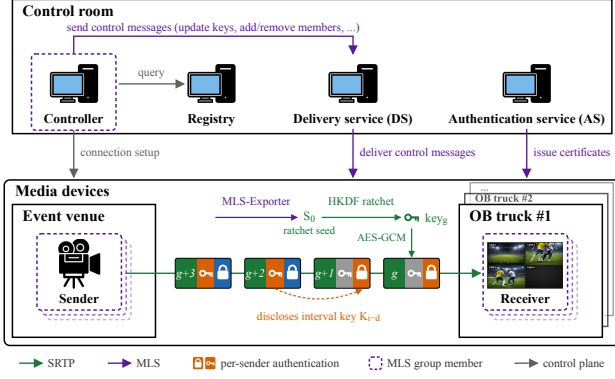


Figure 2: Overview of SafeCast. The controller manages the group by sending MLS control messages through the delivery service, while the authentication service issues certificates. Each epoch, every member derives a fresh per-stream ratchet seed S_0 through the MLS exporter. An HKDF ratchet then advances it into per-generation keys key_g , under which the sender encrypts each packet’s payload via AES-GCM. The optional per-sender authentication layer adds a MAC to each packet and later discloses the corresponding key. Some arrow directions are omitted for simplicity.

into each packet’s IV to prevent precomputation attacks. The second output is the next chain secret. Each member deletes S_g after advancing, so a compromise exposes only the current chain secret. As HKDF.Expand is one-way and the two labels keep the SRTP keys independent of the chain, this yields forward secrecy at the granularity of a generation.

Keying granularity. How long a generation lasts is a deployment choice that trades granularity of forward secrecy against derivation cost, which we quantify in §6.2. SafeCast supports three granularities, leveraging existing header fields to avoid increasing the per-packet expansion with new values such as a Master Key Identifier (MKI). *Epoch-only* holds a single generation for the whole epoch. *Frame-level* exploits the fact that all packets of one video frame share the same RTP timestamp ts , setting $g = (ts - ts_0)/T_{\text{frame}}$. Here, ts_0 is the epoch’s first timestamp and T_{frame} is the clock rate divided by the frame rate, two parameters the stream advertises in its Session Description Protocol (SDP) transport file [14]. *Packet-level* gives each packet its own key, setting $g = idx - idx_0$, where idx is the packet index and idx_0 is the epoch’s first index. It also generalizes to intervals of N packets, setting $g = \lfloor (idx - idx_0)/N \rfloor$. The receiver keeps the last K keys in a sliding window, so that reordering due to jitter does not break decryption. A packet whose generation is in the window decrypts from the cached key. If the packet is ahead of the window, the ratchet moves forward to reach it; if it is behind, the packet is dropped, as its key has been deleted for forward secrecy. With packet-level keying, the keying window matches the replay window, to avoid keeping keys of packets that the replay check would reject anyway. To stop an unauthenticated packet from forc-

ing unbounded work or corrupting the ratchet, this forward catch-up runs on a clone of the ratchet, capped in its number of steps.

5.2 Per-Sender Authentication

SRTP computes its authentication tag under the group key. Thus, the tag can only prove that *some* group member sent a packet. A single compromised device can inject packets, impersonating every other sender. Per-sender authentication addresses this problem, but the typical per-packet public-key signatures are too large and too slow for our throughput, threatening availability and opening the door to DoS attacks [105]. More lightweight per-sender authentication protocols such as TESLA [51, 104–106] replace the signature’s asymmetry with an asymmetry in time. Each packet carries a MAC under a key known only to the sender. The sender discloses the key once receivers no longer accept packets tagged under it. However, these protocols require precise time coordination between sender and receiver to function, which has historically hindered their deployment. SafeCast overcomes this limitation by leveraging additional resources in the broadcasting ecosystem: senders and receivers have synchronized clocks with sub-microsecond precision (preventing an attacker from forging packets under an already-disclosed key).

SafeCast adds an optional per-sender authentication layer based on the TESLA protocol, integrated with the Precision Time Protocol (PTP). SafeCast tolerates packet loss (F4), and the sender’s overhead does not grow with the number of receivers (P4).

Key chain and tagging. Time is sliced into equal *intervals* numbered $1, \dots, N$, each with its own key. A sender picks a random key K_N and hashes backward to build a one-way chain $K_i = \text{MAC}(K_{i+1})$ for a message authentication code MAC. Initially, the sender reveals the anchor K_0 . One-wayness lets a receiver confirm any later key by hashing it down to a key it already trusts (while also recovering lost ones), but an attacker cannot reach undisclosed keys. Senders tag packets sent during interval i under a key derived from K_i . SafeCast offers two MAC algorithms, whose cost we compare in §6.4.

Disclosure. A sender does not reveal K_i until d intervals later, and does so by adding it to a later packet (i.e., each packet carries the key K_{i-d}). More specifically, SafeCast appends 34 bytes to each packet, containing the interval number (4 bytes), the disclosed key K_{i-d} (20 bytes), and the MAC (10 bytes). We use intervals of 1 ms and $d = 2$, so a packet has at least 1 ms to arrive before its key becomes public. This comfortably covers even packets rescued over the slower dual path, which arrive at most 200 μs late, as we discuss in §6.3.

Verification. A packet tagged for interval i can authenticate its sender only if it arrived while K_i was still secret. On arrival, the receiver computes the latest interval the sender can be in, allowing for a clock offset of at most D_t (sub-microsecond,

thanks to the use of the PTP), and checks that this interval lies before $i + d$, the moment the sender starts disclosing K_i . Receivers discard packets that do not pass this check. This does not delay playout, as the receiver decrypts and delivers packets immediately, and only the source-authentication verdict lags. At disclosure, when K_i arrives d intervals later, the receiver first checks the key itself by hashing it down to its trusted anchor, with a cap on the hash steps so that a forged interval number cannot force unbounded work. It then recomputes the MAC of every buffered packet of interval i in one batch. Packets whose MAC matches are confirmed as coming from the sender; the rest are flagged as forged. The check hence completes d intervals (2 ms in our instantiation) after media playout. Note that only a compromised group member can get this far, as packets from outsiders already fail SRTP’s group authentication.

Bootstrapping. The scheme cannot authenticate its own anchor K_0 and the interval schedule (i.e., start, length, and disclosure delay d). The sender bundles both with the chosen MAC algorithm and identifiers that prevent reuse for any other stream, and signs the resulting bootstrap message with its MLS leaf key. While a signature on every packet is far too slow, this single signature per stream and epoch is negligible, and it enables further symmetric verification.

5.3 Protocol Flow

Parties. The protocol flow of our system involves the media devices and the broadcast controller with its registry, as well as MLS’s two infrastructure services (§4.1): the authentication service (AS), which issues and validates credentials, and the delivery service (DS), which stores initial key material and routes protocol messages. The role of the AS is played by the facility’s public key infrastructure (PKI), which the control-plane security framework already mandates [30]. Its certificates serve as the devices’ MLS credentials.

Messages. Key agreement is driven by three types of MLS messages: a *KeyPackage* is a signed bundle of a device’s public encryption and signature keys, with which others can add the device to MLS groups. A *commit* updates keys, changes membership, and starts a new epoch. In SafeCast, only the controller issues commits, mirroring the control-plane architecture, where the controller already orchestrates all connections. A *welcome* message provides a new member with the group state needed to enter the current epoch.

Stages. Fig. 3 illustrates the four stages in our protocol:

1. **Device onboarding.** Each device obtains a certificate from the AS, registers itself and its endpoints with the registry [18], and publishes its KeyPackage to the DS.
2. **Group setup.** The controller creates an MLS group for the stream, fetches the sender’s KeyPackage from the DS, and adds the sender to the group by issuing a commit that contains the sender’s KeyPackage. The stream’s SDP

transport file, which already describes the stream (§2), additionally advertises the URI of the DS and the group identifier, so that any prospective receiver knows where to obtain the group’s key material.

3. **Receiver join.** The controller connects the receiver via the control plane and adds it to the MLS group by issuing a commit with the receiver’s KeyPackage. The receiver processes the *welcome* message and initializes its epoch state. Since the commit starts a new epoch, every member then derives the new ratchet seed S_0 from the exporter secret, and, when per-sender authentication is enabled, the sender publishes its signed bootstrap message. Media transport then begins, with each device advancing the ratchet from S_0 to derive a fresh key per generation.
4. **Per-epoch rekeying.** The controller issues a commit that refreshes the keys or removes a group member. Every remaining member processes the commit, advances to the new epoch, and reseeds its per-stream ratchet with a fresh S_0 . When per-sender authentication is enabled, the sender builds a fresh key chain and publishes a new signed bootstrap message, as the old one is bound to the previous epoch. Rekeying is also required before a stream’s 48-bit SRTP packet index, the counter that numbers its packets [52], wraps around, since a repeated index under the same key would reuse an AES-GCM nonce.

5.4 Security Analysis

The security of SafeCast rests on three layers: MLS establishes and refreshes the group secrets, the within-epoch ratchet turns them into short-lived SRTP keys, and SRTP protects each packet with authenticated encryption. The following two lemmas state the confidentiality and integrity guarantees of this composition. Both hold under a safety predicate that rules out trivial attacks. The safety predicate ensures that the adversary never learns the target stream’s ratchet state or SRTP keys, and that after any compromise, the MLS group has healed (i.e., every compromised member has been removed or has had its own keys refreshed by a commit). We formalize this predicate and prove the lemmas in App. E via a sequence of game hops that progressively replace each layer’s key material with uniform random values, reducing each hop to a standard cryptographic assumption. The lemmas do not cover the optional per-sender authentication layer, whose guarantees rest on TESLA’s security analysis [76, 106].

Lemma 1 (Confidentiality). *Let $\mathcal{A}$ be an adversary against the confidentiality of media payloads in SafeCast, and let $\text{Adv}_{\text{SafeCast}}^{\text{conf}}(\mathcal{A})$ be its advantage over random guessing. Let $g_{\max}$ be the number of ratchet generations in an epoch, and let $\epsilon_{\mathcal{A}}^x$ be the maximum advantage of any efficient adversary*

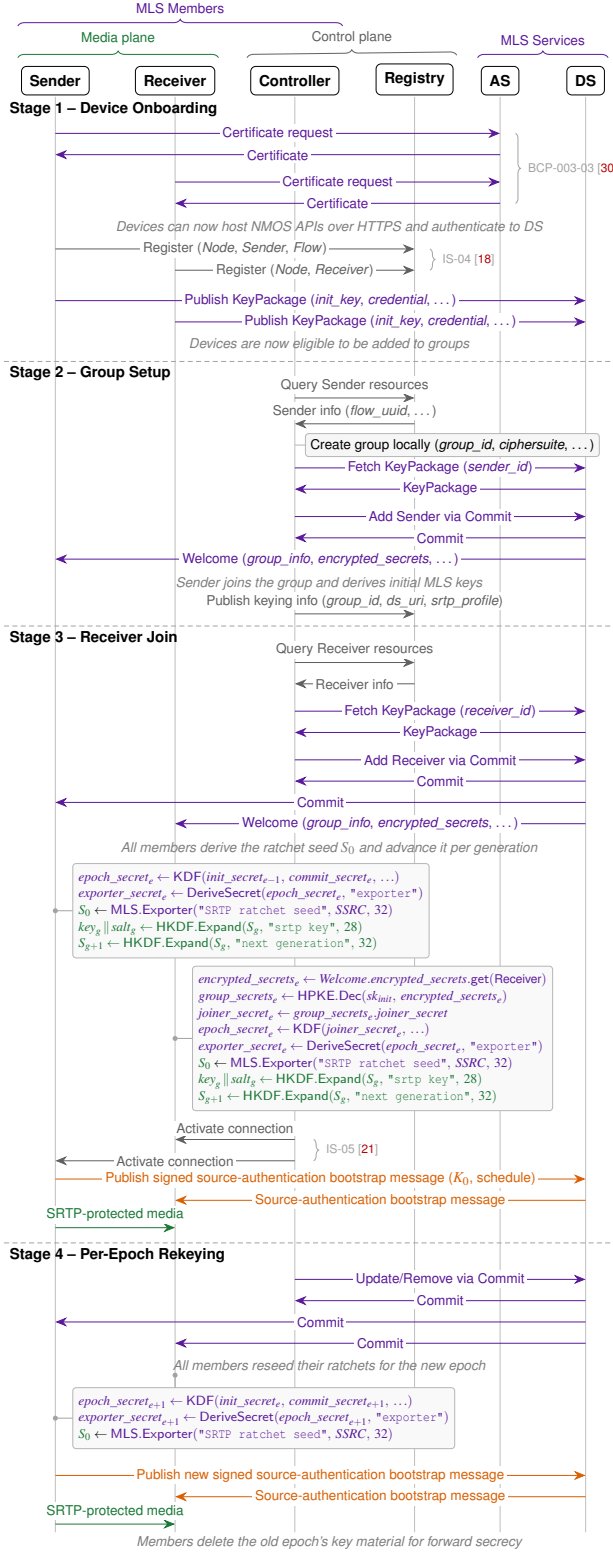


Figure 3: Protocol sequence diagram showing the interactions between sender, receiver, controller, registry, authentication service (AS), and delivery service (DS). Colors match those of Fig. 2.

against property $\times$ of primitive $\mathcal{Y}$. Under the safety predicate:

$$\text{Adv}_{\text{SafeCast}}^{\text{conf}}(\mathcal{A}) \leq \epsilon_{\text{MLS}}^{\text{PCS}} + (1 + g_{\max}) \epsilon_{\text{HKDF}}^{\text{prf}} + g_{\max} \epsilon_{\text{SRTP-KDF}}^{\text{prf}} + \epsilon_{\text{AES-GCM}}^{\text{aead-conf}}.$$

Each term is negligible under the post-compromise security (PCS) of MLS, the PRF security of HKDF (covering the exporter and the within-epoch ratchet), the PRF security of the SRTP KDF, and the IND-CPA security of AES-GCM, respectively. The total advantage is thus negligible, so $\mathcal{A}$ learns essentially nothing about the media content.

Lemma 2 (Integrity). *Let $\mathcal{A}$ be an adversary against the integrity of media payloads in SafeCast, and let $\text{Adv}_{\text{SafeCast}}^{\text{int}}(\mathcal{A})$ be its probability of forging a packet that an honest receiver accepts. Under the safety predicate, with $g_{\max}$ and $\epsilon_{\text{MLS}}^{\text{PCS}}, \epsilon_{\text{HKDF}}^{\text{prf}}, \epsilon_{\text{SRTP-KDF}}^{\text{prf}}, \epsilon_{\text{AES-GCM}}^{\text{aead-int}}$ as in Lemma 1:*

$$\text{Adv}_{\text{SafeCast}}^{\text{int}}(\mathcal{A}) \leq \epsilon_{\text{MLS}}^{\text{PCS}} + (1 + g_{\max}) \epsilon_{\text{HKDF}}^{\text{prf}} + g_{\max} \epsilon_{\text{SRTP-KDF}}^{\text{prf}} + g_{\max} \epsilon_{\text{AES-GCM}}^{\text{aead-int}}.$$

Similarly, all terms are negligible, so $\mathcal{A}$ forges such a packet only with negligible probability.

6 Evaluation

We implement SafeCast in Rust and evaluate it to answer the following questions that capture the requirements in §3.4.

- **Is media protection fast enough?** Encryption and decryption must sustain the bitrates of uncompressed video (P1). We measure the throughput of SafeCast across payload sizes and compare it with PEP in §6.1.
- **How fast is rekeying?** Per-epoch rekeying must complete within milliseconds even for large groups (P4). Within an epoch, SafeCast must additionally provide forward secrecy. The faster the rekeying, the finer the keying granularity and the less media is exposed in case of compromise, at the cost of more key derivations. We measure the effects of rekeying granularity in §6.2.
- **Does performance hold on a realistic network?** A facility network delivers packets with jitter, loss, and reordering. As the media key advances during the stream, a reordered packet may need a key the receiver has not derived yet (and thus needs to be derived on the fly) or one it has already deleted, effectively losing the packet. We investigate the effect of realistic conditions in §6.3.
- **How expensive is per-sender authentication?** The shared group key proves only that *some* member sent a packet. Source authentication closes this gap, but at a cost in communication (P3) and computation, which we study in §6.4.

Our artifact (<https://github.com/ArmanKolozyan/SafeCast>) includes additional experiments covering the impact of interleaved RTCP protection on throughput, the cost of rejecting replayed packets, microbenchmarks that isolate

the decryption overhead, and a breakdown of SRTP’s cost over raw AES-GCM. Since these are not critical for real-world adoption, we omit them from the paper. We also do not focus on audio, since it is far less demanding than video and hence covered by our experiments.

Setup. We benchmark with Rust’s Criterion framework. After a 3-second warmup, each benchmark collects 100 samples, each timing a batch of many consecutive operations. Criterion sizes the batches so that the samples fill a fixed measurement window (e.g., 10 seconds for the throughput benchmarks), yielding millions of calls per reported number. For SRTP, we use Cisco’s libsrtp 2.8.0, written in C. libsrtp delegates all cryptographic operations to OpenSSL (v3.6.0), which in turn uses hardware AES instructions. We access libsrtp from Rust via srtp2-sys. We maintain our own version of this binding because the upstream version bundles an older libsrtp (v2.3.0), whose bug adds roughly 120 ns of spurious overhead per encryption call [67]. For MLS, we use OpenMLS v0.8.1 with the `MLS_128_DHKEMX25519_AES128GCM_SHA256_Ed25519` ciphersuite. All benchmarks run on an Apple M2 (8-core, 16 GB RAM) under macOS Tahoe 26.3.1. The Apple M2 chip provides hardware acceleration for AES-GCM.

Reference video bitrates. We benchmark against uncompressed video bitrates provided to us by the European Broadcasting Union (EBU), listed in [App. F](#).

6.1 Encryption and Decryption Throughput

Cost model. During encryption and decryption, SafeCast pays a fixed cost of about 107 ns per packet (cipher setup, header authentication, and per-packet bookkeeping), plus about 0.10 ns per byte of payload. Using larger packets spreads the fixed cost over more media bytes, so throughput rises with the payload size.

Payload sizes. In ST 2110, payload size is bounded by two Ethernet Maximum Transmission Unit (MTU) classes, *standard* (1500 B) and *jumbo* (9000 B) [13]. Subtracting the UDP [107], RTP [115], and SRTP [87] overheads, we are left with 1424 B and 8924 B of media payload, respectively.

Throughput. [Fig. 4](#) shows single-core SafeCast encryption and decryption throughput across payload sizes, with the bitrates of representative broadcast formats as reference lines. At the standard MTU, encryption reaches 46.1 Gbps and decryption 39.5 Gbps. At the jumbo MTU, they rise to 69.0 and 60.7 Gbps. Decryption is slower than encryption by roughly 40 ns per packet, due to two extra OpenSSL calls to supply and verify the received GCM tag. The receiver’s replay-window check [52] adds a constant 7.3 ns per packet, which is negligible next to the cryptographic work.

Meeting production rates. Through our interactions with the EBU, we learned that the two most common production formats are 1080p and 4K (2160p), run at 50 fps in Europe and 60 fps in the US. At the standard MTU, SafeCast can

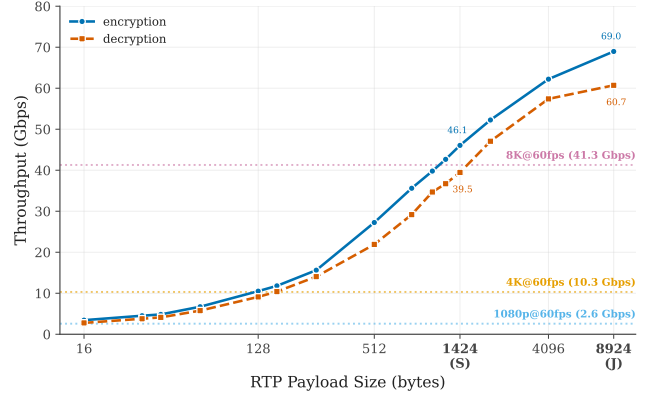


Figure 4: SafeCast encryption and decryption throughput vs. RTP payload size. Dotted horizontal lines mark uncompressed video bitrates. Labels (*S*) and (*J*) on the x-axis denote the standard and jumbo sizes. We omit error bars, as the 95% confidence intervals are below 0.5% of the mean for most payload sizes.

comfortably support both even on a single core. Streams share no cryptographic state, so a device with multiple streams can split its throughput across them. At 1080p60, this yields 17 concurrent streams per core for encryption and 15 for decryption. Hence, even a multiviewer consuming tens of feeds fits on a few cores. Moreover, broadcast equipment commonly uses FPGAs and ASICs, which run AES-GCM far more efficiently than our CPU.

Comparison with PEP. PEP, the only standardized media-encryption protocol for ST 2110, has been implemented and deployed in hardware such as FPGAs [28, 59]. Its throughput therefore sets a bar that has proven sufficient for real-world deployment. [Fig. 5](#) compares SafeCast against PEP [26] in its two modes: the mandatory CTR-only mode (no authentication) and the optional CTR+CMAC-64 mode. PEP CTR-only achieves the highest throughput: 77.4 Gbps at the standard MTU and 102.8 Gbps at jumbo. PEP’s authenticated mode reaches only 4.5 Gbps at the standard MTU and 9.8 Gbps at jumbo, roughly 10× slower than SafeCast. This is because GCM was designed to provide authenticated encryption that keeps pace with high-speed counter-mode encryption [89, 90]. Its performance advantage over CTR+CMAC-64 stems from two factors [90]. First, GCM replaces CMAC’s second AES pass with GHASH, which is a hardware-accelerated finite-field multiply. It costs a fraction of an AES block encryption and can run concurrently with the CTR encryption pass. Second, GHASH is parallelizable, since each block’s multiply is independent, whereas CMAC is inherently sequential.

6.2 Keying Granularity

To address cases in which media confidentiality is crucial, SafeCast can advance the media key within an epoch, reducing exposure in case of compromise. This comes at the cost of

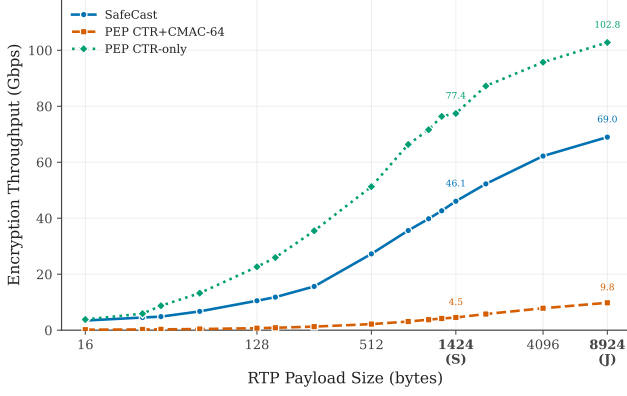


Figure 5: Encryption throughput of SafeCast with AES-128-GCM, PEP CTR-only, and PEP CTR+CMAC-64.

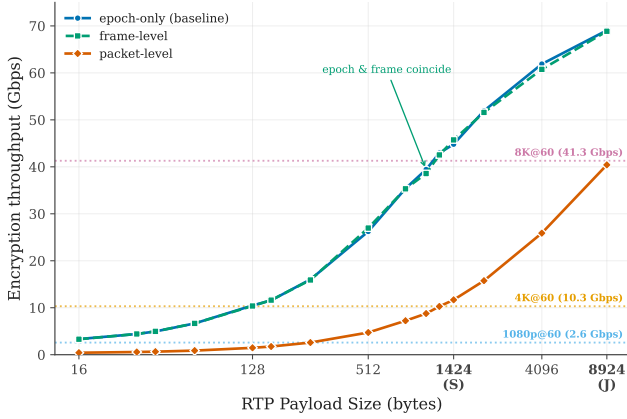


Figure 6: Encryption throughput across RTP payload sizes for the three keying granularities. Frame-level coincides with epoch-only. Packet-level is slower, but its penalty shrinks as payload sizes grow.

key derivations, which entail two HKDF-Expand steps, plus the time to set up the key into the AES-128-GCM cipher. We measure its effect on throughput for three granularities: a per-epoch baseline that holds one key for the whole epoch, and two finer ones that rekey once per frame or once per packet. Fig. 6 plots encryption throughput across payload sizes for all granularities, under ideal in-order delivery.

Per-epoch. Per-epoch rekeying (i.e., removing devices and refreshing keys for post-compromise security) happens during live operation and must hence complete within milliseconds even at scale. Our measurements (see App. G for details) confirm the roughly linear scaling with group size reported by prior work [117], from a few milliseconds for groups of hundreds of devices up to 56.7 ms (commit creation) and 47.4 ms (commit processing) in a 5000-member stress case.

Frame-level. A single frame spans thousands of packets (around 3600 for a 1080p 60 fps frame at the standard MTU), so the per-frame cost is amortized to a fraction of a nanosec-

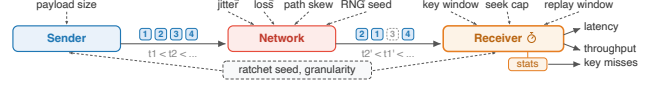


Figure 7: Overview of our realistic-transport benchmark simulator.

ond per packet. Frame-level throughput is therefore indistinguishable from epoch-only. Decryption follows the same shape at slightly lower throughput, as discussed in §6.1.

Packet-level. Packet-level keying incurs a fixed cost on every packet. At the standard MTU, encrypting a packet takes about 260 ns, while deriving and installing its key adds roughly 700 ns on top. In this scenario, keying costs more than the encryption, and throughput drops to 11.7 Gbps, about $3.9\times$ below epoch-only. This still covers the most common production formats, 1080p and 4K using the standard MTU (with more room using the jumbo MTU), so it is possible to obtain the most strict forward secrecy in production if necessary.

6.3 Performance under Realistic Transport

To understand the cost of cryptographic operations, the previous sections assumed ideal in-order delivery. In a real facility network, however, it is common for packets to be reordered (due to jitter or difference between redundant paths) and sometimes dropped. To investigate the impact of such conditions on performance and security, we simulate a realistic broadcast network in software (see Fig. 7). In our simulation, a sender emits a paced, encrypted stream, using one key per generation. A network component then delays, drops, and reorders the packets. The receiver decrypts them with its generation key, reusing a cached key when it can and deriving it otherwise. The simulator outputs decryption latency, throughput, and statistics such as the fraction of packets that could not be decrypted because their key had already been deleted.

Parameters. Every stage of our simulator is configurable. The sender component receives the payload size. The network receives the jitter, the loss, and the path skew (the fixed delay between the two paths). The receiver has three parameters: the key window (how many recent keys it keeps for late packets), the seek cap (maximum number of keys it derives to catch up to a jumped-ahead packet), and the replay window (how far back it tracks indices to reject replays). For each network parameter, we use the worst value our broadcast setting permits. For instance, we set a per-path loss of at most 10^{-5} [73]. The EBU confirmed that although a single path drops packets at this rate, the dual-path redundancy drives end-to-end loss to zero. When one path does drop a packet, its copy still arrives on the other path, delayed by the path skew, which we set to a realistic 150 μ s [10]. Finally, we set the per-path jitter to at most 50 μ s [15].

Tail latency. Reordering does not lower decryption throughput, since generation keys are still derived once. Reordering

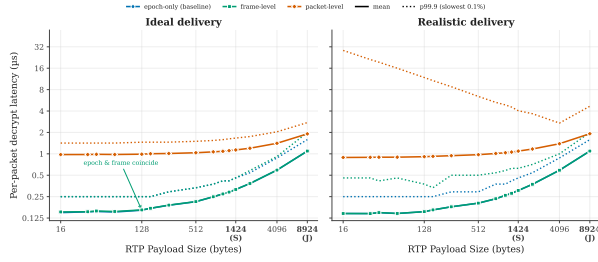


Figure 8: Per-packet decryption latency vs. payload size, under ideal and realistic delivery. Reordering inflates only the packet-level tail.

instead changes which packets require key derivations, increasing the tail of per-packet latency. A packet whose generation is ahead of the receiver’s current key forces it to derive all keys in between, with larger decryption costs the further ahead a packet is. Fig. 8 shows this effect. Reordering has little impact on epoch-only and frame-level rekeying, because most packets reuse the current key. However, for packet-level, the slowest 0.1% of decrypts at the standard MTU climb from 1.7 μ s under ideal delivery to 4.0 μ s. For production, the delay matters only when a packet misses its playout deadline. At the MTU sizes that carry video, our tail’s delay is only a few microseconds. Even the tightest playout buffers, sized for the precise pacing of FPGA-based senders, run for tens of microseconds [15], while typical receivers buffer milliseconds. We thus conclude that SafeCast’s decryption is sufficiently fast even under tough realistic conditions.

Effect on forward secrecy. Forward secrecy relies on deleting old keys, but the receiver must retain enough keys to decrypt late packets, trading the exposure of a compromise against tolerance of reordering. App. H quantifies this tradeoff for a standard-MTU 1080p stream and identifies a sweet spot at a rekeying interval of 44 packets: the receiver keeps only two keys, a compromise exposes just 0.4 ms of media, and decryption still reaches 86% of the epoch-only throughput.

6.4 Cost of Source Authentication

SafeCast provides optional per-sender authentication, which entails extra communication (i.e., extra bytes on the wire) and extra computation on both ends.

Communication. Each packet carries a 34-byte extension. Adding this information reduces the media payload by 2.4% of a standard-MTU packet, on top of SRTP’s own 16-byte GCM tag (1.1%). We consider this byte cost small (and smaller when using the jumbo MTU, 0.4%). For an uncompressed 1080p stream (2.6 Gbps), this amounts to roughly 60 Mbps of additional bandwidth. Facilities reserve bandwidth for each stream in advance (§3.1), so the extension means reserving 2.4% more.

Computation. We benchmark source authentication with two algorithms, HMAC-SHA256 and GMAC, on both sender

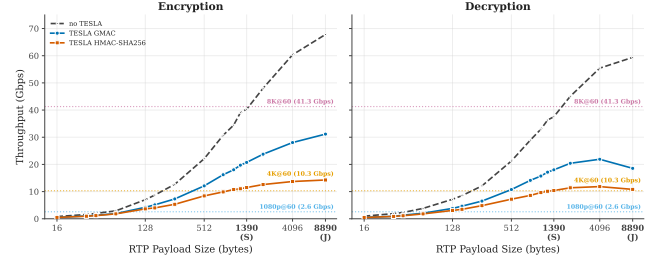


Figure 9: Sender and receiver throughput with and without source authentication. Dashed lines mark uncompressed bitrates.

and receiver. The results are presented in Fig. 9. GMAC is the cheaper one, because it is built on GHASH, so it runs on the same hardware acceleration that AES-GCM already uses for its tag. At the standard MTU, it roughly halves throughput, to 20.7 Gbps for the sender and 18.0 Gbps for the receiver, both still well above 4K. HMAC-SHA256 has no such acceleration and costs about twice as much as GMAC, giving 11.5 Gbps for the sender and 10.4 Gbps for the receiver, right at the 4K line. Receiver throughput first rises with the payload, as each packet’s fixed cost is spread over more bytes, then falls again beyond about 4 KB. This is because the receiver must keep each packet’s ciphertext until its key is disclosed, so it copies the packet aside before decrypting it in place, and this extra per-byte copy eventually outweighs the amortization.

7 Takeaways and Next Steps

Broadcast facilities have been carrying the world’s most-watched content over IP without cryptographic protection for nearly a decade. Yet this infrastructure has received little attention from the security community. In this work, we introduced SafeCast, which encrypts and integrity-protects broadcast media, provides tight forward secrecy, and optionally authenticates each packet’s sender. SafeCast fulfills the functional and security requirements we elicited from broadcast professionals, while supporting common production formats with ample headroom on a single core.

Our work is a firm first step towards protecting broadcast production environments. Still, open challenges remain in fully securing modern productions that our community can help with. One direction is improving SafeCast’s efficiency (e.g., decreasing the overhead of joining groups by adapting the Partial MLS extension [80] to our scenario; determining which epoch’s key protects a packet arriving around a commit without the per-packet cost of a Master Key Identifier (MKI)). An analysis of the monetary cost of deploying SafeCast is also left for future work. Finally, emerging broadcast architectures raise new security needs. For example, software-based broadcast processing exchanges media via shared memory (e.g., DMA) instead of RTP [23]; securing such internal shared-memory exchanges is an open problem.

Acknowledgments

We thank Willem Vermost and Lucille Verbaere of the European Broadcasting Union (EBU) for generously sharing their expertise on real-world broadcast deployments, which helped us align our threat model and evaluation with how live productions operate in practice. This work is partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA – 390781972.

References

- [1] ST 259:2008 – SDTV Digital Signal/Data: Serial Digital Interface. Technical report, SMPTE, 2008. [doi:10.5594/SMPTE.ST259.2008](https://doi.org/10.5594/SMPTE.ST259.2008).
- [2] Ethernet jumbo frames. Technical report, Ethernet Alliance, 2009.
- [3] Beyond the digital conversion: The integration of information technology and professional media. Technical report, SMPTE, 2014.
- [4] IP live production and the business of broadcasting. Technical report, Sony Corporation, 2015.
- [5] TR-03: Transport of Uncompressed Elementary Stream Media over IP. Technical report, Video Services Forum (VSF), 2015.
- [6] ER-3:2017 – Engineering Report: Study Group on Flow Management in Professional Media Networks. Technical report, SMPTE, 2017.
- [7] ST 2110 standards suite, 2017. Accessed: 2026-04-05. URL: <https://www.smpte.org/standards/st2110>.
- [8] High-bandwidth Digital Content Protection System: Mapping HDCP to HDMI. Technical report, Digital Content Protection LLC, 2018.
- [9] IEEE standard for local and metropolitan area networks: Media access control (MAC) security. *IEEE Std 802.1AE-2018*, 2018. [doi:10.1109/IEEESTD.2018.8585421](https://doi.org/10.1109/IEEESTD.2018.8585421).
- [10] ST 2022-7:2019 – Seamless Protection Switching of RTP Datagrams. Technical report, SMPTE, 2019. [doi:10.5594/SMPTE.ST2022-7.2019](https://doi.org/10.5594/SMPTE.ST2022-7.2019).
- [11] F1 steers future sustainable broadcasting with innovative Remote Operations, 2020. Accessed: 2026-06-17. URL: <https://corp.formula1.com/f1-steers-future-sustainable-broadcasting-with-innovative-remote-operations>.
- [12] IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. *IEEE Std 1588-2019*, 2020. [doi:10.1109/IEEESTD.2020.9120376](https://doi.org/10.1109/IEEESTD.2020.9120376).
- [13] ST 2110-10:2022 – Professional Media Over Managed IP Networks: System Timing and Definitions. Technical report, SMPTE, 2022. [doi:10.5594/SMPTE.ST2110-10.2022](https://doi.org/10.5594/SMPTE.ST2110-10.2022).
- [14] ST 2110-20:2022 – Professional Media Over Managed IP Networks: Uncompressed Active Video. Technical report, SMPTE, 2022. [doi:10.5594/SMPTE.ST2110-20.2022](https://doi.org/10.5594/SMPTE.ST2110-20.2022).
- [15] ST 2110-21:2022 – Professional Media Over Managed IP Networks: Traffic Shaping and Delivery Timing for Video. Technical report, SMPTE, 2022. [doi:10.5594/SMPTE.ST2110-21.2022](https://doi.org/10.5594/SMPTE.ST2110-21.2022).
- [16] Uncompressed media over IP on Amazon Web Services (AWS). Technical report, Amazon Web Services, 2022.
- [17] ST 2110-40:2023 – Professional Media Over Managed IP Networks: Ancillary Data. Technical report, SMPTE, 2023. [doi:10.5594/SMPTE.ST2110-40.2023](https://doi.org/10.5594/SMPTE.ST2110-40.2023).
- [18] AMWA IS-04 NMOS Discovery and Registration Specification. NMOS Specification, Advanced Media Workflow Association (AMWA), 2024. Accessed: 2026-05-15. URL: <https://specs.amwa.tv/is-04/>.
- [19] TR-06-2: Reliable Internet Stream Transport (RIST). Technical report, Video Services Forum (VSF), 2024.
- [20] ST 2110-30:2025 – Professional Media Over Managed IP Networks: PCM Digital Audio. Technical report, SMPTE, 2025. [doi:10.5594/SMPTE.ST2110-30.2025](https://doi.org/10.5594/SMPTE.ST2110-30.2025).
- [21] AMWA IS-05 NMOS Device Connection Management Specification. NMOS Specification, Advanced Media Workflow Association (AMWA), 2026. Accessed: 2026-05-15. URL: <https://specs.amwa.tv/is-05/>.
- [22] DICOM PS3.22 2026c - Real-Time Communication. Technical report, DICOM Standards Committee, 2026. Accessed: 2026-08-04. URL: <https://dicom.nema.org/medical/dicom/current/output/html/part22.html>.
- [23] The dynamic media facility: Reference architecture. Technical report, European Broadcasting Union, 2026.

- [24] Internet Protocol Media Experience (IPMX): General Organization. Technical report, Video Services Forum (VSF), 2026.
- [25] Internet Protocol Media Experience (IPMX): HDCP Key Exchange Protocol. Technical report, Video Services Forum (VSF), 2026.
- [26] Internet Protocol Media Experience (IPMX): Privacy Encryption Protocol (PEP). Technical report, Video Services Forum (VSF), 2026.
- [27] NMOS Glossary. NMOS Specification, Advanced Media Workflow Association (AMWA), 2026. Accessed: 2026-05-15. URL: <https://specs.amwa.tv/nmos/>.
- [28] Adeas. AIP-PEP (Privacy Encryption Protocol), 2025. Accessed: 2026-08-11. URL: <https://www.adeas.nl/wp-content/uploads/2025/09/PEP-leaflet.pdf>.
- [29] Advanced Media Workflow Association. AMWA BCP-003-01: Secure communication in NMOS systems, 2023. Accessed: 2025-06-04. URL: <https://specs.amwa.tv/bcp-003-01/>.
- [30] Advanced Media Workflow Association. AMWA BCP-003-03: Certificate provisioning in NMOS systems, 2024. Accessed: 2025-06-04. URL: <https://specs.amwa.tv/bcp-003-03/>.
- [31] Advanced Media Workflow Association. AMWA BCP-003: Security recommendations for NMOS APIs, 2024. Accessed: 2026-08-05. URL: <https://specs.amwa.tv/bcp-003/>.
- [32] Advanced Media Workflow Association. AMWA BCP-003-02: Authorization in NMOS systems, 2025. Accessed: 2025-06-04. URL: <https://specs.amwa.tv/bcp-003-02/>.
- [33] Advanced Media Workflow Association. AMWA BCP-005-03: NMOS support for IPMX/PEP, 2026. Accessed: 2026-08-07. URL: <https://specs.amwa.tv/bcp-005-03/>.
- [34] Advanced Media Workflow Association. AMWA IS-05: Controllers, 2026. Accessed: 2026-05-15. URL: <https://specs.amwa.tv/is-05/releases/v1.1.2/docs/Controllers.html>.
- [35] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *2013 IEEE Symposium on Security and Privacy*, pages 526–540, Berkeley, CA, USA, May 19–22, 2013. IEEE Computer Society Press. doi:10.1109/SP.2013.42.
- [36] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yianis Tselekounis. Security analysis and improvements for the IETF MLS standard for group messaging. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 248–277, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-56784-2_9.
- [37] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yianis Tselekounis. Modular design of secure group messaging protocols and the security of MLS. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021: 28th Conference on Computer and Communications Security*, pages 1463–1483, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press. doi:10.1145/3460120.3484820.
- [38] Flemming Andreassen, Dan Wing, and Mark Baugher. Session Description Protocol (SDP) Security Descriptions for Media Streams. RFC 4568, 2006. doi:10.17487/RFC4568.
- [39] Apple Inc. FairPlay Streaming Overview, 2016. Accessed: 2026-04-07. URL: <https://developer.apple.com/streaming/fps/FairPlayStreamingOverview.pdf>.
- [40] ARET Video and Audio Engineering. 14 HD Cameras OB Van with Side Expansion, 2019. Accessed: 2026-06-17. URL: <https://aret-engineering.com/portfolio/14-hd-cameras-ob-van-side-expansion-evp-hd-10-unit/>.
- [41] Richard Barnes, Benjamin Beurdouche, Raphael Robert, Jon Millican, Emad Omara, and Katriel Cohn-Gordon. The Messaging Layer Security (MLS) Protocol. RFC 9420, 2023. doi:10.17487/RFC9420.
- [42] Ali C. Begen, Paul Kyzivat, Colin Perkins, and Mark J. Handley. SDP: Session Description Protocol. RFC 8866, 2021. doi:10.17487/RFC8866.
- [43] Ali C. Begen and Colin Perkins. Duplicating RTP Streams. RFC 7198, 2014. doi:10.17487/RFC7198.
- [44] Ali C. Begen, Colin Perkins, Dan Wing, and Eric Rescorla. Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs). RFC 7022, 2013. doi:10.17487/RFC7022.
- [45] Benjamin Beurdouche, Eric Rescorla, Emad Omara, Srinivas Inguva, and Alan Duric. The Messaging Layer Security (MLS) Architecture. RFC 9750, 2025. doi:10.17487/RFC9750.

- [46] Karthikeyan Bhargavan, Richard Barnes, and Eric Rescorla. TreeKEM: Asynchronous decentralized key management for large dynamic groups: A protocol proposal for messaging layer security (MLS). Technical report, 2018.
- [47] Gavin Blackburn. Hackers target Iran state TV’s satellite transmission to broadcast exiled crown prince, 2026. Accessed: 2026-08-05. URL: <https://www.euronews.com/2026/01/19/hackers-target-iran-state-tv-satellite-transmission-to-broadcast-exiled-crown-prince>.
- [48] Tania Branigan. Chinese anger after TV station leaks Olympics opening ceremony rehearsal, 2008. Accessed: 2026-08-04. URL: <https://www.theguardian.com/world/2008/jul/31/china.olympicgames2008>.
- [49] Bradley Cain, Steve E. Deering, Bill Fenner, Isidor Kouvelas, and Ajit Thyagarajan. Internet Group Management Protocol, Version 3. RFC 3376, 2002. doi:10.17487/RFC3376.
- [50] Jon Callas, Alan Johnston, and Philip Zimmermann. ZRTP: Media Path Key Agreement for Unicast Secure RTP. RFC 6189, 2011. doi:10.17487/RFC6189.
- [51] Elisabetta Carrara and Mark Baugher. The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP). RFC 4383, 2006. doi:10.17487/RFC4383.
- [52] Elisabetta Carrara, Karl Norrman, David McGrew, Mats Naslund, and Mark Baugher. The Secure Real-time Transport Protocol (SRTP). RFC 3711, 2004. doi:10.17487/RFC3711.
- [53] Check Point Research. EvilPloy: Attack against Iran’s state broadcaster, 2022. Accessed: 2026-08-05. URL: <https://research.checkpoint.com/2022/evilploy-attack-against-irans-state-broadcaster/>.
- [54] CHESA. Modernizing broadcast at the U.S. Department of Health and Human Services, 2026. Accessed: 2026-08-04. URL: <https://www.chesa.com/from-legacy-sdi-to-full-st-2110/>.
- [55] Cisco. IP Fabric for Media Design Guide, 2024. Accessed: 2026-06-19. URL: <https://www.cisco.com/c/en/us/td/docs/dcn/whitepapers/cisco-ipfm-design-guide.html>.
- [56] Cisco. Cisco Validated Design with Panasonic - A blueprint for IP Media Success, 2026. Accessed: 2026-06-19. URL: <https://www.cisco.com/c/en/us/td/docs/dcn/whitepapers/cisco-cvd-blueprint-for-ip-media-success.html>.
- [57] Matt Crawford. Transmission of IPv6 Packets over Ethernet Networks. RFC 2464, 1998. doi:10.17487/RFC2464.
- [58] DASH Industry Forum. DASH-IF Interoperability Points v5.0, Part 6: Content Protection, 2022. Accessed: 2026-04-07. URL: <https://dash-industry-forum.github.io/docs/IOP-Guidelines/DASH-IF-IOP-Part6-v5.0.0.pdf>.
- [59] Spencer Deame and Robert Green. IPMX Demystified: What We’ve Learned and How to Deploy, 2026. Accessed: 2026-08-11. URL: <https://www.amd.com/en/blogs/2026/ipmx-demystified--what-we-ve-learned-and-how-to-deploy-.html>.
- [60] Lars Eggert, Gorrry Fairhurst, and Greg Shepherd. UDP Usage Guidelines. RFC 8085, 2017. doi:10.17487/RFC8085.
- [61] Saad El Jaouhari, Bernard Gibaud, Philippe Lemonnier, Guillaume Pasquier, Eric Poiseau, Eric Guiffard, Patrick Hardy, and Emmanuel Cordonnier. Introduction to DICOM-RTV: a new standard for real-time video communication in hospitals. In *2019 IEEE International Conference on E-health Networking, Application & Services (HealthCom)*, 2019. doi:10.1109/HealthCom46333.2019.9009435.
- [62] Mathis Engelbart, Joerg Ott, and Spencer Dawkins. RTP over QUIC (RoQ). Technical report, Internet Engineering Task Force, 2025. Work in Progress.
- [63] European Broadcasting Union. EBU report highlights cybersecurity shortcomings of networked media products, 2020. Accessed: 2026-08-03. URL: <https://tech.ebu.ch/news/2020/02/ebu-report-highlights-cybersecurity-shortcomings-of-networked-media-products>.
- [64] EVS. Cerebrum: Broadcast Control and Monitoring System, 2022. Accessed: 2026-05-15. URL: <https://evs.com/sites/default/files/2022-03/Cerebrum%20-%20product%20Sheet%20+%20protocol%20list%20V2.2.pdf>.
- [65] Brian Flood. ABC News’ Amy Robach caught on hot mic saying network spiked Jeffrey Epstein bombshell. Fox News, 2019. Accessed: 2026-04-30. URL: <https://www.foxnews.com/media/abc-amy-robach-jeffrey-epstein-hot-mic>.
- [66] FOR-A Company Limited. MV-3240IP IP multi viewer. Accessed: 2026-08-20. URL: <https://www.for-a.com/products/mv3240ip/>.
- [67] Marcus Froeschl. libsrtp fix commit, 2020. Accessed: 2026-08-10. URL: <https://github.com/cisco/libsrtp/commit/837ba9d9>.

- [68] Noa Gal. Behind the Scenes of Broadcasting the Paris 2024 Olympics, 2024. Accessed: 2026-06-17. URL: <https://www.viaccess-orca.com/blog/behind-the-scenes-of-broadcasting-the-paris-2024-olympics>.
- [69] Noa Gal. Behind the Scenes of Broadcasting the 2026 Winter Olympics, 2026. Accessed: 2026-06-17. URL: <https://www.viaccess-orca.com/blog/behind-the-scenes-of-broadcasting-the-2026-winter-olympics>.
- [70] Joe Gallop. Jack Morton on the challenges involved in staging NYE fireworks show, 2022. Accessed: 2026-08-04. URL: <https://accessaa.co.uk/jack-morton-explain-challenges-of-producing-behind-closed-doors-nye-fireworks-show>.
- [71] Grass Valley. Inside Walmart’s IP-based broadcast transformation, 2026. Accessed: 2026-08-04. URL: <https://www.grassvalley.com/inside-walmarts-ip-based-broadcast-transformation/>.
- [72] Jake Holland, Lucas Pardue, Max Franke, and Kyle Rose. Multicast Extension for QUIC. Technical report, Internet Engineering Task Force, 2026. Work in Progress.
- [73] International Telecommunication Union. Network Performance Objectives for IP-Based Services. Technical Report Y.1541, 2011.
- [74] ISO/IEC. Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats, 2022.
- [75] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, 2021. doi:10.17487/RFC9000.
- [76] Goce Jakimoski. Some notes on the security of the timed efficient stream loss-tolerant authentication scheme. In Eli Biham and Amr M. Youssef, editors, *SAC 2006: 13th Annual International Workshop on Selected Areas in Cryptography*, volume 4356 of *Lecture Notes in Computer Science*, pages 342–357, Montreal, Canada, August 17–18, 2007. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-74462-7_24.
- [77] Sadiq Khan. MD3009 New Year’s Eve 2022, 2023, 2024 and 2025, 2022. Accessed: 2026-08-04. URL: <https://www.london.gov.uk/who-we-are/governance-and-spending/promoting-good-governance/decision-making/decisions/md3009-new-years-eve-2022-2023-2024-and-2025>.
- [78] Franziskus Kiefer. OpenMLS performance, 2021. Accessed: 2025-06-03. URL: <https://blog.openmls.tech/posts/2021-05-18-openmls-first-benchmarks/>.
- [79] Franziskus Kiefer. Performance of large group operations, 2025. Accessed: 2025-06-03. URL: <https://github.com/openmls/openmls/blob/4af317c5a/openmls/examples/large-groups.rs#L290-L295>.
- [80] Franziskus Kiefer, Karthikeyan Bhargavan, Richard Barnes, Joël Alwen, and Marta Mularczyk. Partial MLS. Technical report, Internet Engineering Task Force, 2025. Work in Progress.
- [81] Adi Kouadio, Gerben Dierick, and Alvaro Martin Santos. Cybersecurity vulnerability assessment: General report. Technical report, JT-NM, 2020.
- [82] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331, Santa Barbara, CA, USA, August 19–23, 2001. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-44647-8_19.
- [83] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648, Santa Barbara, CA, USA, August 15–19, 2010. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-14623-7_34.
- [84] Renaud Lavoie, Sithideth Viengkhou, and Francois Gagnon. Three tips to accelerate your IP (ST 2110) deployments, 2021. Accessed: 2026-06-19. URL: <https://www.thebroadcastbridge.com/content/entry/16803/three-tips-to-accelerate-your-ip-st-2110-deployments>.
- [85] Steve Layman. Inside the CBS TV compound at the Super Bowl, 2019. Accessed: 2026-06-17. URL: <https://www.newschannel5.com/sports/inside-the-cbs-tv-compound-at-the-super-bowl>.
- [86] Jonathan Lennox. Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP). RFC 6904, 2013. doi:10.17487/RFC6904.
- [87] David McGrew and Kevin Igoe. AES-GCM Authenticated Encryption in the Secure Real-time Transport Protocol (SRTP). RFC 7714, 2015. doi:10.17487/RFC7714.

- [88] David McGrew and Eric Rescorla. Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP). RFC 5764, 2010. doi:10.17487/RFC5764.
- [89] David A. McGrew and John Viega. The security and performance of the Galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004: 5th International Conference in Cryptology in India*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355, Chennai, India, December 20–22, 2004. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-30556-9_27.
- [90] David A. McGrew and John Viega. The Galois/Counter Mode of Operation (GCM). Technical report, NIST, 2004.
- [91] Charlotte McLaughlin. Eurovision 2025: Lawyers called in after ‘hack of rehearsal footage’, 2025. Accessed: 2026-08-04. URL: <https://www.independent.co.uk/arts-entertainment/eurovision/eurovision-2025-hack-rehearsal-footage-semi-final-b2750216.html>.
- [92] Dan Meier. 3, 2, 1, ssh!: How London’s New Year’s Eve producers orchestrated the UK’s largest drone display in secret, 2022. Accessed: 2026-08-04. URL: <https://www.tvbeurope.com/live-production/3-2-1-ssh-how-londons-new-years-eve-producers-orchestrated-the-uks-largest-drone-display-in-secret>.
- [93] NEP Group. OB Truck Specifications, 2024. Accessed: 2026-06-17. URL: <https://www.nepgroup.com/trucklocations/hd-6>.
- [94] NEP Group. NEP Bow Tie Supports The Welsh Parliament In Delivering Its 2110 Broadcast Infrastructure, 2026. Accessed: 2026-08-04. URL: <https://www.nepgroup.com/resources/nep-bow-tie-supports-the-welsh-parliament-in-delivering-its-2110-broadcast-infrastructure>.
- [95] Net Insight. Remote Production 2.0, 2018. Accessed: 2026-06-19. URL: https://netinsight.net/wp-content/uploads/2015/12/Remote_production_Solutionpaper.pdf.
- [96] NFL. Arizona man pleads guilty to hacking Super Bowl telecast, 2011. Accessed: 2026-08-05. URL: <https://www.nfl.com/news/arizona-man-pleads-guilty-to-hacking-super-bowl-telecast-09000d5d8234e3e0>.
- [97] Karl Norrman, Fredrik Lindholm, Elisabetta Carrara, Jari Arkko, and Mats Naslund. MIKEY: Multimedia Internet KEYing. RFC 3830, 2004. doi:10.17487/RFC3830.
- [98] NVIDIA. NMOS Controller, 2026. Accessed: 2026-05-15. URL: <https://catalog.ngc.nvidia.com/orgs/nvidia/teams/holoscan-for-media/containers/controller-ui>.
- [99] Emad Omara, Justin Uberti, Sergio Garcia Murillo, Richard Barnes, and Youenn Fablet. Secure Frame (SFrame): Lightweight Authenticated Encryption for Real-Time Media. RFC 9605, August 2024. doi:10.17487/RFC9605.
- [100] Tony Orme. Advances in 12G-SDI. Technical report, AJA Video Systems, 2020.
- [101] Panasonic Corporation. AW-UE160 4K PTZ camera. Accessed: 2026-08-20. URL: <https://eu.connect.panasonic.com/gb/en/broadcast-proav/remote-ptz-camera-systems/aw-ue160>.
- [102] Roger Pantos and William May. HTTP Live Streaming. RFC 8216, 2017. doi:10.17487/RFC8216.
- [103] Adrian Pennington. Olympian effort: Tokyo 2020 host production upgrades to UHD, IP and cloud, 2021. Accessed: 2026-06-17. URL: <https://www.thebroadcastbridge.com/content/entry/17189/olympian-effort-tokyo-2020-host-production-upgrades-to-uhd-ip-and-cloud>.
- [104] Adrian Perrig, Ran Canetti, Dawn Song, J. Doug Tygar, and Bob Briscoe. Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction. RFC 4082, 2005. doi:10.17487/RFC4082.
- [105] Adrian Perrig, Ran Canetti, Dawn Xiaodong Song, and J. Doug Tygar. Efficient and secure source authentication for multicast. In *ISOC Network and Distributed System Security Symposium – NDSS 2001*, pages 35–46, San Diego, CA, USA, February 7–9, 2001. The Internet Society.
- [106] Adrian Perrig, Ran Canetti, J. Doug Tygar, and Dawn Xiaodong Song. Efficient authentication and signing of multicast streams over lossy channels. In *2000 IEEE Symposium on Security and Privacy*, pages 56–73, Oakland, CA, USA, May 2000. IEEE Computer Society Press. doi:10.1109/SECPRI.2000.848446.
- [107] Jon Postel. User Datagram Protocol. RFC 768, 1980. doi:10.17487/RFC0768.

- [108] Max Pritikin, Peter E. Yee, and Dan Harkins. Enrollment over Secure Transport. RFC 7030, 2013. doi:10.17487/RFC7030.
- [109] Radar Online. HGTV star Nicole Curtis caught saying racial slur in unaired footage, 2026. Accessed: 2026-04-30. URL: <https://www.youtube.com/watch?v=rOUl7kSrxFQ>.
- [110] Range Commanders Council. Standards-Compliant Architecture for Range Optics. Technical report, Range Commanders Council, 2024. Accessed: 2026-08-04. URL: <https://www.trmc.osd.mil/wiki/download/attachments/271690727/WP-22-001.pdf>.
- [111] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 9846, 2026. doi:10.17487/RFC9846.
- [112] Eric Rescorla, Hannes Tschofenig, and Nagendra Modadugu. The Datagram Transport Layer Security (DTLS) Protocol Version 1.3. RFC 9147, 2022. doi:10.17487/RFC9147.
- [113] Reuters. Hackers hit Poland Euro 2024 match broadcast in second attack, 2024. Accessed: 2026-08-05. URL: <https://www.reuters.com/sports/soccer/hackers-hit-poland-euro-2024-match-broadcast-second-attack-2024-06-21/>.
- [114] Reuters. Ukraine’s 1+1 media group satellite TV channels hit by cyberattack, 2024. Accessed: 2026-08-05. URL: <https://www.reuters.com/world/europe/ukraines-11-media-group-reports-cyberattack-satellite-tv-channels-2024-04-17/>.
- [115] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, 2003. doi:10.17487/RFC3550.
- [116] Maria Sharabayko, Maxim Sharabayko, Jean Dube, Joonwoong Kim, and Jeongseok Kim. The SRT Protocol. Technical report, Internet Engineering Task Force, 2021.
- [117] David Soler, Carlos Dafonte, Manuel Fernández-Veiga, Ana Fernández Vilas, and Francisco J. Nóvoa. Analysis of efficiency of the messaging layer security protocol in experimental settings, 2026.
- [118] Sony. nmos-js: An NMOS Client in JavaScript, 2026. Accessed: 2026-05-15. URL: <https://github.com/sony/nmos-js>.
- [119] Andy Stout. FIFA World Cup Kicks Off with 45 Cameras for Every Match, 2026. Accessed: 2026-06-17. URL: <https://www.redsharknews.com/fifa-world-cup-2026-hbs-broadcast-45-cameras>.
- [120] Justin Uberti, Cullen Fluffy Jennings, and Sergio Garcia Murillo. Completely Encrypting RTP Header Extensions and Contributing Sources. RFC 9335, 2023. doi:10.17487/RFC9335.
- [121] Serge Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS... In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–546, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-46035-7_35.
- [122] Endrit Vorfaj. Hardening security of SMPTE ST 2110 networks. EBU webinar presentation, 2025. Accessed: 2026-04-05. URL: https://tech.ebu.ch/files/live/sites/tech/files/shared/events/webinars/presentations/20251009_webinar_Hardening_ST2110_Webinar_Vorfaj.pdf.
- [123] Endrit Vorfaj and Carmela Troncoso. Designing secure ST 2110 systems. *SMPTE Motion Imaging Journal*, 135(1), 2026. doi:10.5594/JMI.2026/YOXS1352.
- [124] Klaus Weber. Remote Productions with Lower Bandwidth Requirements, 2024. Accessed: 2026-06-19. URL: https://www.grassvalley.com/wp-content/uploads/2025/01/remote_production_wp-ub-3-1069a-en_v1-cr.pdf.
- [125] YouTube Help. Youtube recommended upload encoding settings. Accessed: 2026-04-30. URL: <https://support.google.com/youtube/answer/1722171>.
- [126] Laura Zuckerman. That “zombie apocalypse” warning in montana? it was fake. Reuters, 2013. Accessed: 2026-08-05. URL: <https://www.reuters.com/article/lifestyle/that-zombie-apocalypse-warning-in-montana-it-was-fake-idUSBRE91B1K2/>.

Ethical Considerations

Securing broadcast infrastructure, whose content reaches millions of people, raises ethical questions about the research process, the publication of our threat analyses, and the effects of deploying SafeCast. We address each in turn.

The research process posed minimal risk. All experiments ran on our own hardware with synthetically generated media streams. We accessed no production network and processed no real broadcast content or personal data. In our interactions with the European Broadcasting Union (EBU) and with engineers in the Internet Protocol Media Experience (IPMX) working group, we mainly asked about broadcast practices to inform our design and evaluation. These include how facilities operate ST 2110 networks, which media formats are common, and how much latency a security mechanism may add. The people we consulted knew the purpose of our questions, and we attribute no statements to named individuals.

Like any threat model, ours may also inform attackers. The threats we describe, however, build on information that is already public: the attacker capabilities we assume are standard for networked systems, and the incidents we cite come from sources such as news reports. Our threat model organizes this known exposure into requirements that SafeCast addresses. Similarly, our analysis of PEP identifies weaknesses in the public specification rather than vulnerabilities in a deployed implementation, and discusses potential mitigations for each. We communicated our findings to one of the specification authors, who explained the design constraints behind these choices. Through this conversation, we also learned that PEP targets professional audio-visual settings in general, such as conference rooms and lecture halls. An attack in these settings has far less impact than in broadcast production, which limits the harm our analysis could enable.

SafeCast can reduce harm to broadcasters from eavesdropping, media manipulation, and disruption. It also protects viewers at home, who might otherwise receive manipulated content, and bystanders, whose unblurred faces and off-camera moments could otherwise become public. We acknowledge that not every stakeholder gains from this protection. In some cases, leaked raw material has served the public interest, for example by revealing that a broadcaster had declined to air an interview with one of Jeffrey Epstein’s accusers [65]. We believe the benefits to broadcasters, viewers, and bystanders outweigh this drawback.

Open Science

Our artifact, available at <https://github.com/ArmanKolyan/SafeCast>, contains all components needed to reproduce the results of §6: the Rust implementation of SafeCast, the benchmark suite, the raw measurements behind every reported number, and the Jupyter notebook that turns these measurements into figures. All code is extensively documented

and covered by integration tests.

Obtaining correct measurements required patching our dependencies. For instance, as discussed in the software setup of §6, the SRTP implementation we build on ships with a bug that adds spurious overhead to every encryption call. The artifact therefore bundles our patched copies of these dependencies, each with a README explaining what we changed and why.

To ease reproduction, we provide a `REPRODUCE.sh` script that runs all benchmarks and then executes the Jupyter notebook to regenerate every figure of this paper, taking about 2.5 hours. Individual benchmarks can be run in isolation with the `cargo bench` command, as explained in the README, which further documents the required toolchains, the repository structure, and the figures each benchmark produces. As the artifact includes our raw measurements, the `GENERATE_FIGURES.ipynb` notebook can also be executed to regenerate all figures without re-running the benchmarks.

A Production Network Topology

Fig. 10 shows the network inside a production facility, which is typically organized in a *spine-leaf* topology. While monolithic designs built around a single large switch are common in small studios for operational simplicity, they are limited by the capacity of that switch [84]. Larger studio deployments adopt spine-leaf networks to overcome this bottleneck [55, 56], often organized into two tiers: a bottom *leaf* tier, to which media devices attach, and a top *spine* tier that interconnects the leaves. Media streams are distributed via IP multicast and replicated across the network to the subscribed receivers. For resilience, the fabric is dual-redundant, with each device connecting to two independent networks [10].

When a sender transmits a stream, its leaf switch forwards a single copy upward to a spine. Receivers express interest in the stream by subscribing to its multicast address using the Internet Group Management Protocol (IGMP) [49], and the switches track these subscriptions. The spine replicates the copy only to the leaf switches that have interested receivers. Each such leaf in turn replicates it locally to its own subscribed devices [55]. The sender hence does not open a separate point-to-point connection to each receiver, as it would with unicast delivery. Instead, it transmits the stream once, and the network replicates it on the way to the receivers.

As the transport is IP-based, sites in different locations can also be linked together: a leaf switch at a remote site (e.g., an event venue or an outside broadcast (OB) truck) can be connected back to the central facility over a long-distance network link leased from a telecom provider [55]. This is what enables the REMI model described in §2.

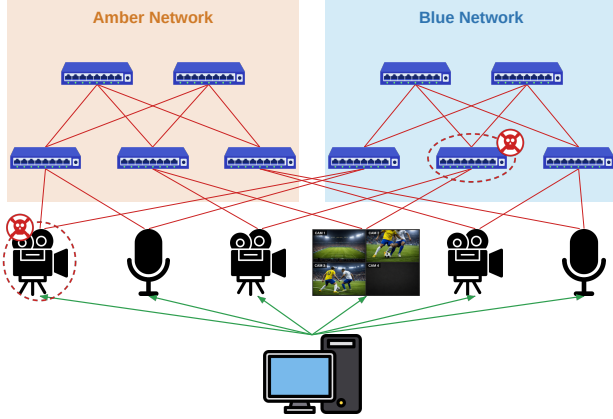


Figure 10: Spine-leaf hierarchy in an IP broadcast deployment. Media devices connect to two redundant network fabrics via leaf switches, possibly through an IP gateway (e.g., an SDI-to-IP converter). A controller orchestrates connections between senders and receivers. Red lines denote media transport exposed to on-path attackers. Green lines denote the controller’s connections, for which protection mechanisms exist [29, 31, 32]. Dashed circles indicate compromised devices.

B OTT Delivery vs. Production-Side Transport

Table 3 summarizes the key differences between over-the-top (OTT) delivery, in which finished content is streamed to end viewers via services such as Netflix or YouTube, and production-side media transport within an IP broadcast facility. These differences explain why security mechanisms designed for consumer streaming cannot be directly applied to the broadcast production environment. The rows marked with † indicate differences that directly affect the design of the security mechanism.

Content. In OTT delivery, content is a finished product: produced, edited, and compressed before distribution to viewers. Audio and video are bundled into a single adaptive bitrate stream, which means that the server offers multiple quality levels and the player selects which one to request based on its available bandwidth. Production-side transport carries the raw, high-quality (up to 8K), uncompressed output of cameras and microphones [14], with each media component (video, audio, and ancillary data) as a separate RTP stream [7]. This includes content that may never reach the final broadcast.

Operational. OTT streams typically share bandwidth with other internet traffic on a best-effort basis, meaning the network provides no guaranteed bandwidth and packets may be delayed or dropped under congestion (i.e., when the network is overloaded). Throughput per viewer is measured in hundreds of megabits per second at most [125]. Production facilities use dedicated network infrastructure capable of sustaining a total throughput in the hundreds of gigabits per

second, according to data provided to us by the European Broadcasting Union (EBU). OTT viewers tolerate seconds of buffering because they are passive consumers with no real-time interaction with the stream: whether the content is live or pre-recorded, a few seconds of delay has no functional consequence. Live production, by contrast, tolerates almost no added delay, for two reasons: it is interactive (e.g., directors and camera operators react to live video in real time), and uncompressed streams are too large to buffer away delays.

Delivery. OTT delivery uses HTTP over TCP, which provides reliable in-order delivery through retransmission. Production-side transport uses RTP over UDP, which offers no retransmission because waiting for a retransmitted packet would introduce too much delay for live production. OTT delivers content via unicast to individual viewers, while production facilities use multicast to distribute each stream to multiple receivers simultaneously. Redundancy strategies also differ: OTT distributes content through Content Delivery Networks (CDNs), which are geographically distributed servers that cache and serve content close to viewers. When one server fails, the player switches to another server, which may cause a brief interruption (rebuffering) while the new server takes over. Production facilities, by contrast, transmit two identical copies of each stream over physically separate network paths and switch seamlessly between them when one path fails [10]. Finally, OTT players synchronize audio and video using their local clock, while IP broadcast devices rely on a shared PTP clock to keep related streams in sync across the facility [13].

C Candidate Protocol Descriptions

This appendix describes the remaining candidate protocols compared in §4.

C.1 Key Establishment Mechanisms

MIKEY. Multimedia Internet KEYing (MIKEY) is designed for both peer-to-peer and group communication [97]. It supports several methods for distributing keys, including pre-shared keys, public-key encryption, and Diffie-Hellman key exchange. MIKEY can be adapted to various use cases, but it does not provide forward secrecy or post-compromise security for group communication.

DTLS. Datagram Transport Layer Security (DTLS) is a variant of TLS designed for unreliable transports such as UDP [112]. It can be used solely for key establishment, where only the DTLS handshake is run to derive session keys from certificates or pre-shared keys [88]. As a point-to-point protocol, however, DTLS is not suitable for group communication.

SDS. Session Description Protocol Security Descriptions (SDS) is a key management mechanism in which the cryptographic parameters, including keying material, are conveyed in the Session Description Protocol (SDP) signaling

Table 3: Key differences between OTT streaming to end viewers and transporting media within a live production facility. Rows marked with † indicate properties that directly affect the design of the security mechanism.

Property		OTT delivery	Production-side
<i>Content</i>	Stage	Finished	Raw
	Resolution	Adaptive	Fixed (up to 8K)
	Compression†	Compressed (lossy)	Uncompressed
	Media packaging†	Bundled	Separate stream per component
<i>Operational</i>	Bandwidth	Shared (best-effort)	Dedicated
	Throughput†	Up to 300 Mbps	Up to 500 Gbps
	Latency tolerance†	Seconds of buffering	Near-zero
<i>Delivery</i>	Protocol†	HTTP over TCP	RTP over UDP
	Topology†	One-to-many	Many-to-many
	Distribution†	Unicast	Multicast
	Redundancy†	CDN failover (rebuffering)	Seamless path switching
	Synchronization†	Local clock	Shared clock (PTP)

exchanged when a session is set up [38]. Unlike DTLS, which establishes keys through a dedicated handshake between the two endpoints, SDP places the keys directly in the signaling path. This simplicity comes with an important limitation: the security of the media depends heavily on the security of the signaling channel that carries the SDP.

ZRTP. ZRTP runs a Diffie-Hellman key agreement in-band over the media path, requiring no public-key infrastructure or certificates [50]. To detect a man-in-the-middle, each endpoint derives a Short Authentication String (SAS) from the exchanged keys, and the two users verbally confirm that these match. This manual step, together with its restriction to point-to-point sessions, makes ZRTP unsuitable for our setting.

PEP (Key Management). PEP [26] combines key management with media-plane encryption inside the control-plane architecture. We cover its key management here and its data transport below. PEP’s key management is based on pre-shared keys (PSKs). Each receiver derives a stream’s encryption key from a PSK it shares with the sender, together with a set of non-secret parameters. The controller distributes these parameters over the control plane, carried in SDP transport files [21, 42].

C.2 Secure Media Transport

RTP over QUIC. QUIC is a general-purpose transport protocol, originally developed for the web, that runs over UDP and provides authenticated encryption [75]. RTP over QUIC (RoQ) carries RTP and RTCP inside QUIC and thus inherits this protection [62]. While an experimental extension to QUIC adds multicast delivery [72], it still requires a persistent unicast connection to every receiver for per-packet integrity data and acknowledgments. Neither RoQ nor the multicast extension has advanced beyond draft status.

SFrame. SFrame provides end-to-end authenticated encryption of media payloads for group conferencing, built as an extra layer on top of a transport-security protocol such as SRTP. Its purpose is to hide the media from an untrusted central server that forwards it between participants [99].

PEP (Media Transport). PEP also defines a media-plane encryption mechanism for RTP streams [26]. Each stream is encrypted using keys derived from the PSK and the non-secret parameters. It provides confidentiality but leaves integrity optional and provides no replay protection. Some modes support in-band key updates, carrying a plaintext key version alongside the media that is incremented on each rekey, so receivers detect key transitions without out-of-band signaling.

D Full Security Analysis of PEP

While §4 explains the design decisions behind PEP [26], this appendix details the security weaknesses that follow from them. For each weakness, we describe how an adversary can exploit it and discuss potential mitigations.

D.1 Lack of Mandatory Authentication

PEP mandates support for AES-128 in counter mode (AES-CTR) for RTP media encryption, while support for authentication is optional. As a result, a deployment can be fully PEP-compliant while providing confidentiality only.

Exploitation. An active attacker can exploit this malleability for targeted content manipulation, modifying selected bits in the ciphertext to cause controlled bit flips in the decrypted audio or video. Even non-targeted modifications can be weaponized to mount denial of service (DoS) attacks at scale.

Potential Mitigations. Authentication for RTP media (ideally via an AEAD such as AES-GCM) should be mandatory.

D.2 Unprotected RTCP Traffic

As explained in §2, RTCP periodically reports metrics such as packet loss and jitter. Applications use this information to monitor stream quality, keep related streams such as audio and video synchronized, and trigger repair actions. In PEP, RTCP traffic is left unauthenticated and unencrypted.

Exploitation. An active attacker can inject forged RTCP packets to mount DoS attacks. For example, a spoofed RTCP BYE tricks receivers into believing the sender has left the session, causing them to stop playback. Likewise, forged feedback triggers unnecessary retransmissions that waste bandwidth. More subtly, falsified reports can mislead the adaptation and synchronization mechanisms, for example by corrupting the timestamp mapping that receivers use to synchronize audio and video [115]. Finally, RTCP messages leak metadata such as participant identifiers and quality metrics to passive attackers.

Potential Mitigations. RTCP protection should be a requirement of PEP, with integrity protection at a minimum to prevent forgery.

D.3 No Revocation Mechanism

Once a pre-shared key (PSK) is distributed to a receiver, that receiver can access all streams protected under it indefinitely. The specification explicitly states that to prevent further access, the sender must stop using that key. It provides neither receiver-specific revocation nor any notion of post-compromise security. Moreover, PEP leaves key provisioning to vendor-specific mechanisms.

Exploitation. A stolen or hacked device, or an insider who extracts the PSK, can continue decrypting any stream protected with it without further interaction with the system. This risk is amplified by the fact that PSK reuse is explicitly permitted in PEP, as the specification even notes the simplest configuration where one PSK is used for all streams.

Potential Mitigations. Mitigating this weakness requires moving beyond static pre-shared keys toward group key management with support for membership changes and rekeying.

D.4 Dependence on the Security of External Channels

PEP’s security depends on external channels: it distributes cryptographic metadata, including the cipher mode and IV, via Session Description Protocol (SDP) files that controllers deliver to devices. The specification only suggests protecting this exchange with HTTPS as an “extra layer of security”.

While these parameters are not secret, their integrity directly determines how keys are derived and how media is protected.

Exploitation. An active attacker who can modify SDP files in transit can mount downgrade attacks. Since AES-128-CTR is guaranteed to be supported, rewriting the advertised crypto mode forces unauthenticated encryption even when both sender and receiver support stronger modes. Modifying values such as the IV corrupts decryption, which is a cheap DoS attack.

Potential Mitigations. The security mechanism should ideally not depend on a protected control plane for its own guarantees. MLS, for example, treats its delivery service as untrusted and authenticates its messages end to end.

D.5 MAC-then-Encrypt

In the AES-CTR plus CMAC modes specified in PEP, integrity is provided by computing a CMAC over the plaintext packet, appending it, and encrypting the payload together with the MAC. In this MAC-then-encrypt design, the authentication tag only becomes available after decryption, which is more error-prone than modern authenticated encryption.

Exploitation. If any plaintext is acted upon before a successful CMAC check, an attacker can use malformed ciphertexts to trigger unnecessary work (DoS). The design also increases exposure to side-channel and oracle-style attacks, as the receiver must decrypt attacker-controlled ciphertexts before authenticating them. If the implementation exhibits any distinguishable behavior on invalid packets, such as timing differences, an attacker can probe repeatedly to build a feedback channel that recovers plaintexts [35, 121].

Potential Mitigations. PEP should recommend a standard AEAD construction such as AES-GCM, which provides confidentiality and integrity in a single, well-analyzed design. If the ecosystem must keep AES-CTR plus a separate MAC, the design should move toward encrypt-then-MAC, so receivers can verify integrity before decryption and reject modified packets early [82]. Protocols such as TLS adopted encrypt-then-MAC specifically for this reason. Moreover, key separation should be mandated. To simplify key management, PEP uses a single key K for both the AES-CTR encryption and the CMAC computation. However, two independent subkeys can be derived cheaply from K using a KDF:

$$\begin{aligned} K_{\text{enc}} &= \text{KDF}(K, \text{“enc”}) \\ K_{\text{mac}} &= \text{KDF}(K, \text{“mac”}) \end{aligned}$$

Further Weaknesses. Beyond these issues, PEP offers forward secrecy only in peer-to-peer configurations, provides no replay protection, and even its optional authenticated modes cover only parts of the packet, leaving the RTP headers unauthenticated.

E Proof of SafeCast Security

We sketch game-based security proofs for media payloads protected by SafeCast, which encrypts them with SRTP under keys derived from MLS via the exporter and the within-epoch ratchet. We focus exclusively on AEAD SRTP with AES-GCM [87]. We prove both confidentiality and integrity. Both proofs follow the hybrid (game-hopping) structure used by Alwen et al. [37] by (i) decomposing the protocol into its cryptographic building blocks, (ii) defining a challenge game with a safety predicate (to rule out trivial corruption wins), and (iii) gradually replacing each building block with uniform randomness, with each hop justified by a standard primitive security assumption. The two proofs share a hybrid chain that replaces MLS and SRTP key material with random values. They differ only in the final reduction step, which targets a different AEAD security property.

E.1 Protocol Model and Notation

We first fix the protocol model and state the security assumptions on the underlying cryptographic primitives.

MLS epochs and healing. We identify each MLS epoch by an index e and write e^* for the *challenge epoch*: the epoch in which the adversary issues its challenge query. A central concept in the proof is the *healing epoch* e_h ($e_h \leq e^*$): if an adversary has corrupted group members at some point, e_h is the first subsequent epoch by which every compromised member has either been removed from the group or refreshed its own leaf keys through a commit after it was no longer compromised.

SRTP keys from MLS. Recall from §5.1 that in each epoch, the MLS exporter derives a per-stream ratchet seed S_0 , using the stream’s SSRC as context so that different streams produce independent seeds. From S_0 , the HKDF ratchet derives each generation’s key material $key_g \parallel salt_g$ and the next chain secret S_{g+1} under distinct labels, and members delete S_g once the ratchet advances. We write $g_{\max}$ for the number of generations the challenged stream uses in one epoch. For each generation, $(key_g, salt_g)$ serve as the SRTP master key and master salt.

SRTP with AES-GCM. SRTP mandates a KDF that derives session encryption keys and session salts from the master key and master salt [52, 87]. We model this KDF as a PRF. For each packet, AES-GCM authenticates the RTP header as associated data and encrypts the payload [87]. Its 12-octet IV is the session salt XORed with $0^{16} \parallel SSRC \parallel ROC \parallel SEQ$, where the rollover counter ROC and the 16-bit sequence number SEQ together form a 48-bit packet index that is unique per key.

Cryptographic assumptions. The proof relies on the following standard assumptions about the primitives used. For each, we write $e_Y^\times$ for the maximum advantage of any efficient adversary against property $\times$ of scheme Y :

1. **MLS post-compromise security (PCS).** After the healing epoch, the MLS update secret is computationally indistinguishable from a uniformly random value, even if the adversary previously corrupted group members [37].
2. **PRF security of HKDF.** When keyed by a pseudorandom secret, HKDF outputs are computationally indistinguishable from uniform random, given distinct (label, context) inputs [83]. We rely on this both for the exporter and for the within-epoch ratchet.
3. **PRF security of the SRTP KDF.** The SRTP key derivation function, when keyed by a uniform random master key, produces session keys and salts indistinguishable from uniform random [52, 87].
4. **AEAD security of AES-GCM.** AES-GCM provides IND-CPA confidentiality and INT-CTXT integrity under unique IVs [87].

Safety predicate. Both games require a safety predicate $\text{Safe}(S, e)$ on the target sender S and epoch e (the adversary loses if it is violated). For SafeCast, $\text{Safe}(S, e)$ requires:

1. No exposure of the target stream’s SRTP keys: At no point during the game has the adversary learned the stream’s ratchet state (the seed S_0 or a chain secret S_g) or the derived SRTP keys.
2. Post-compromise recovery: If the adversary ever corrupted any member that could reveal group secrets, then there exists a healing epoch $e_h \leq e$.
3. IV uniqueness: The IV never repeats under the same session key for that stream (i.e., the packet index $ROC \parallel SEQ$ does not wrap without rekeying).

E.2 Confidentiality

We define an experiment $\text{Exp}^{\text{conf}}(\mathcal{A})$ with hidden bit $b \in \{0, 1\}$. The adversary controls delivery of network messages, can request sends, and can corrupt parties. It may issue a single challenge query. Following Alwen et al. [37], the adversary commits at initialization to the sender, stream, and epoch it will target. It additionally commits to the challenge generation g^* .

Oracles.

- $\text{Send}(S, e, m, \text{hdr})$: returns a packet produced by honest sender S in epoch e , encrypting and authenticating payload m , with the RTP header hdr as AEAD associated data, under the MLS-derived SRTP keys for that stream.
- $\text{Corrupt}(P)$: returns party P ’s current local state (including any currently stored key material).
- $\text{Chall}(S, e, m_0, m_1, \text{hdr})$: requires $|m_0| = |m_1|$ and returns a packet encrypting m_b with header hdr as associated data.

Advantage. The experiment outputs 1 when $\mathcal{A}$ correctly guesses b , and we define:

$$\text{Adv}_{\text{SafeCast}}^{\text{conf}}(\mathcal{A}) := \left| \Pr[\text{Exp}^{\text{conf}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right|.$$

Hybrid chain. Let H_0 denote the real-world confidentiality game Exp^{conf} , where all keys are derived as in the real protocol (see Fig. 2). For each hybrid H_i , let $p_i := \Pr[H_i(\mathcal{A}) \Rightarrow 1]$, where $:=$ means “defined as” and $H_i(\mathcal{A}) \Rightarrow 1$ denotes the event that the game, run with adversary $\mathcal{A}$, outputs 1 (i.e., the adversary wins).

Hybrid H_1 (randomize MLS update secret). H_1 is identical to H_0 except that the MLS update secret at the healing epoch e_h is replaced with a uniformly random string of the same length. All subsequent MLS key schedule values (including those for epoch e^*) are re-derived from this randomized secret. By Assumption 1, this replacement is computationally indistinguishable to the adversary, hence $|p_0 - p_1| \leq \epsilon_{\text{MLS}}^{\text{PCS}}$.

Hybrid H_2 (randomize ratchet seed). H_2 is identical to H_1 except that the ratchet seed S_0 for the challenged stream is replaced with a uniform random value (rather than being derived via the MLS exporter). After the replacement in H_1 , the exporter secret at e^* (derived from the epoch secret [41]) is pseudorandom, and the exporter’s (label, context) input is unique per stream, as the context is the stream’s SSRC. Assumption 2 therefore gives $|p_1 - p_2| \leq \epsilon_{\text{HKDF}}^{\text{prf}}$.

Hybrid H_3 (randomize ratchet outputs). H_3 is identical to H_2 except that, for the challenged stream, the ratchet outputs of the challenge epoch (i.e., $(key_g, salt_g)$ and S_{g+1} for every generation $g < g_{\max}$) are replaced with independent uniform random values. We argue via a sequence of $g_{\max}$ sub-hybrids, one per generation, in increasing order of g . In sub-hybrid g , the chain secret S_g is uniform random (for $g = 0$ by H_2 , otherwise by the previous sub-hybrid) and is used only as the key of the two HKDF-Expand calls, which carry distinct labels. Both outputs are therefore computationally indistinguishable from uniform by the PRF security of HKDF. Summing over the generations: $|p_2 - p_3| \leq g_{\max} \cdot \epsilon_{\text{HKDF}}^{\text{prf}}$.

Hybrid H_4 (randomize SRTP session keys). H_4 is identical to H_3 except that the session encryption keys and session salts (derived by the SRTP KDF from $(key_g, salt_g)$) are replaced with independent uniform random values of the correct lengths. After H_3 , each key_g is uniform random, which is what Assumption 3 requires. Applying it once per generation: $|p_3 - p_4| \leq g_{\max} \cdot \epsilon_{\text{SRTP-KDF}}^{\text{prf}}$.

Final reduction. After H_4 , every session key is a fresh, independent random value. An adversary that still wins H_4 can therefore be turned directly into an adversary against AES-GCM itself by using the challenger’s hidden key for generation g^* . Under unique IVs (guaranteed by the safety predicate), this gives $|p_4 - \frac{1}{2}| \leq \epsilon_{\text{AES-GCM}}^{\text{aead-conf}}$.

Summing the per-hop bounds $|p_{i-1} - p_i|$ via the triangle

inequality yields the bound of Lemma 1:

$$\begin{aligned} \text{Adv}_{\text{SafeCast}}^{\text{conf}}(\mathcal{A}) &\leq \epsilon_{\text{MLS}}^{\text{PCS}} + (1 + g_{\max}) \epsilon_{\text{HKDF}}^{\text{prf}} \\ &\quad + g_{\max} \epsilon_{\text{SRTP-KDF}}^{\text{prf}} + \epsilon_{\text{AES-GCM}}^{\text{aead-conf}}. \end{aligned}$$

Since $g_{\max}$ is polynomially bounded and each advantage term is negligible under its respective assumption, the overall confidentiality advantage is negligible.

E.3 Integrity

We define an experiment $\text{Exp}^{\text{int}}(\mathcal{A})$. The adversary controls delivery of network messages, can request sends, and can corrupt parties. As in the confidentiality game, it commits at initialization to the sender, stream, and epoch it will target. The adversary wins if it outputs an SRTP packet that an honest receiver accepts even though the packet was not previously output by the honest sender for that stream.

Oracles.

- $\text{Send}(S, e, m, \text{hdr})$ and $\text{Corrupt}(P)$: as in the confidentiality game, except that the game records all honestly generated ciphertexts.
- $\text{Verify}(R, e, c, \text{hdr})$: the adversary submits a candidate SRTP packet c with header hdr to honest receiver R in epoch e . The game returns accept or reject. The adversary wins if accept is returned and c was not previously output by Send for the same stream and epoch.

Advantage. We define the adversary’s advantage as the probability that it wins:

$$\text{Adv}_{\text{SafeCast}}^{\text{int}}(\mathcal{A}) := \Pr[\text{Exp}^{\text{int}}(\mathcal{A}) \Rightarrow 1].$$

Hybrid chain. The same hybrid chain as in the confidentiality proof applies with $H_0 = \text{Exp}^{\text{int}}$, so the per-hop bounds carry over unchanged.

Final reduction. Any accepted forgery would break the ciphertext integrity of AES-GCM under one of the $g_{\max}$ generation keys. A union bound over the generations gives, under unique IVs (guaranteed by the safety predicate), $p_4 \leq g_{\max} \cdot \epsilon_{\text{AES-GCM}}^{\text{aead-int}}$.

Summing the per-hop bounds $|p_{i-1} - p_i|$ via the triangle inequality yields the bound of Lemma 2:

$$\begin{aligned} \text{Adv}_{\text{SafeCast}}^{\text{int}}(\mathcal{A}) &\leq \epsilon_{\text{MLS}}^{\text{PCS}} + (1 + g_{\max}) \epsilon_{\text{HKDF}}^{\text{prf}} \\ &\quad + g_{\max} \epsilon_{\text{SRTP-KDF}}^{\text{prf}} + g_{\max} \epsilon_{\text{AES-GCM}}^{\text{aead-int}}. \end{aligned}$$

Since $g_{\max}$ is polynomially bounded and each advantage term is negligible under its respective assumption, the overall integrity advantage is negligible.

F Uncompressed Video Bitrates

Table 4 reports the uncompressed video bitrates for broadcast formats provided to us by the European Broadcasting Union (EBU), which serve as the reference target rates in our evaluation.

Table 4: Uncompressed video bitrates for broadcast formats. The Color column indicates how much color information is stored per frame. Since 4:4:4 denotes more color information than 4:2:2, it requires a higher bitrate.

Width	Height	FPS	Color	Bitrate (Gbps)
1280	720	50	4:2:2	0.96
1280	720	60	4:2:2	1.15
1920	1080	50	4:2:2	2.15
1920	1080	60	4:2:2	2.58
3840	2160	50	4:2:2	8.60
3840	2160	60	4:2:2	10.32
3840	2160	120	4:2:2	20.65
3840	2160	300	4:2:2	51.62
7680	4320	50	4:2:2	34.41
7680	4320	60	4:2:2	41.29
7680	4320	120	4:2:2	82.59
7680	4320	300	4:2:2	206.47
7680	4320	300	4:4:4	496.52

G Per-Epoch Rekey Latency

This appendix details the per-epoch rekeying measurements summarized in §6.2.

Group tree configuration. The cost of an MLS commit depends not just on the group size but also on the tree topology: a freshly joined member leaves parts of the ratchet tree *blank* (without key pairs), and a commit must spend extra encryptions to cover them. To obtain reproducible benchmarks, we fully populate the tree by having each member issue a self-update commit after joining. This corresponds to the “base” tree variant described by OpenMLS [78] and matches its own benchmarking strategy [79].

Latency. To rekey, the controller creates a commit that advances the group to a new epoch. Media devices then process it to derive fresh keys from the new epoch secret. Exclusion therefore completes once every media device has processed the commit and switched to the new keys. Fig. 11 plots the two costs: commit creation at the controller and commit processing at a media device, across group sizes from 2 to 5000. Both grow roughly linearly, reaching 56.7 ms for creation and 47.4 ms for processing at 5000 devices. Commit creation is slower because it involves $O(\log n)$ public-key encryptions, one at each level up to the root. A media device, in contrast, performs a single public-key decryption and derives

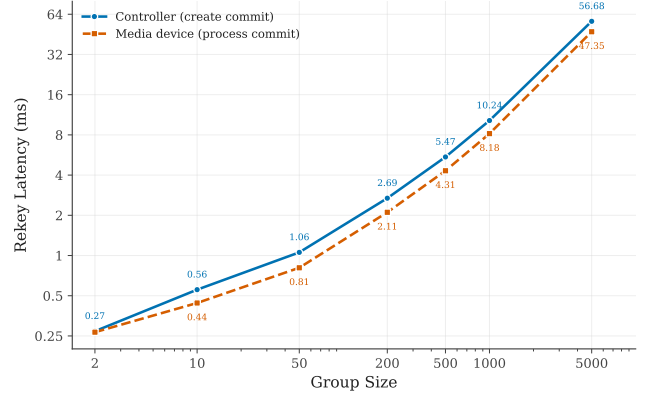


Figure 11: SafeCast per-epoch rekey latency vs. group size.

the remaining secrets up to the root with cheap symmetric hashing [41, 46].

Validation. Our measurements match prior empirical studies of MLS. Soler et al. [117] report the same roughly linear scaling, despite the $O(\log n)$ complexity of the underlying cryptographic operations. They attribute it to non-cryptographic work over the full tree (e.g., parsing it, updating tree hashes, and persisting state), much of which cannot be circumvented for security and practical reasons.

Representativeness. A controller, the only creator of commits, is general-purpose software running on a PC [34], deployed as a native [64], web [118], or containerized [98] application. Our benchmark hardware therefore represents the controller well. Media devices, however, process commits on their own hardware, a CPU or the embedded core of an FPGA SoC, which may be slower than our machine (i.e., their processing times should be read as optimistic).

H Forward Secrecy under Realistic Transport

This appendix expands on the forward-secrecy tradeoff of §6.3, using the same realistic-transport simulator and parameters. Forward secrecy relies on deleting old keys, but a late packet can no longer be decrypted once its key is gone. The receiver therefore keeps its most recent K generation keys. How many keys that takes depends on how late packets can be (which the transport parameters fix), and on the rekeying interval N (the number of packets a single key covers).

Worst lateness. We first bound how far a packet can slip behind. The standard-MTU 1080p stream sends about 3640 packets per frame, one every 4.6 μ s. Two effects reorder them. Per-path jitter delays a packet by up to 50 μ s. The 150 μ s skew between the two redundant paths delays the rescued copy of a packet dropped on the other path. A packet that suffers both arrives at most 200 μ s late. One packet leaves every 4.6 μ s, so that delay spans at most $L = 44$ packets, the worst lateness the receiver’s key window must cover.

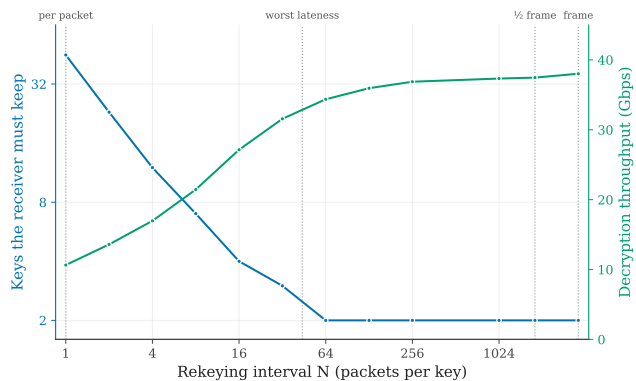


Figure 12: Keys the receiver must retain and decryption throughput vs. rekeying interval N . Both flatten once N reaches the worst packet lateness (44), where the window collapses to two keys and throughput nears its plateau.

The tradeoff. A packet L positions late spans $\lceil L/N \rceil$ rekeyings, so the receiver must retain $\lceil L/N \rceil + 1$ keys to still decrypt it (the extra one for the current key). Fig. 12 plots this key count against N , alongside the decryption throughput each interval sustains. Finer keying (small N) forces the window to span many generations: at packet-level ($N = 1$) the receiver keeps 45 keys and throughput falls to 10.6 Gbps. Coarser keying shrinks the window and lifts throughput. Both curves flatten at $N = L = 44$, the worst lateness: it is the smallest interval at which the window collapses to 2 keys, while throughput has already climbed to within a few percent of its plateau.

Sweet spot. Coarser keying beyond this point still costs forward secrecy, because each retained key now covers more media: the exposure of a receiver compromise is $K \cdot N$ packets, or $K \cdot N \cdot 4.6 \mu\text{s}$ of stream. Frame-level keying ($N = 3640$), for instance, keeps the same 2 keys but spans two whole frames (about 33 ms). Matching N to the worst lateness minimizes that exposure. Rekeying every 44 packets keeps only 2 keys, exposes just 0.4 ms of media, and still sustains about 86% of the epoch-only throughput.