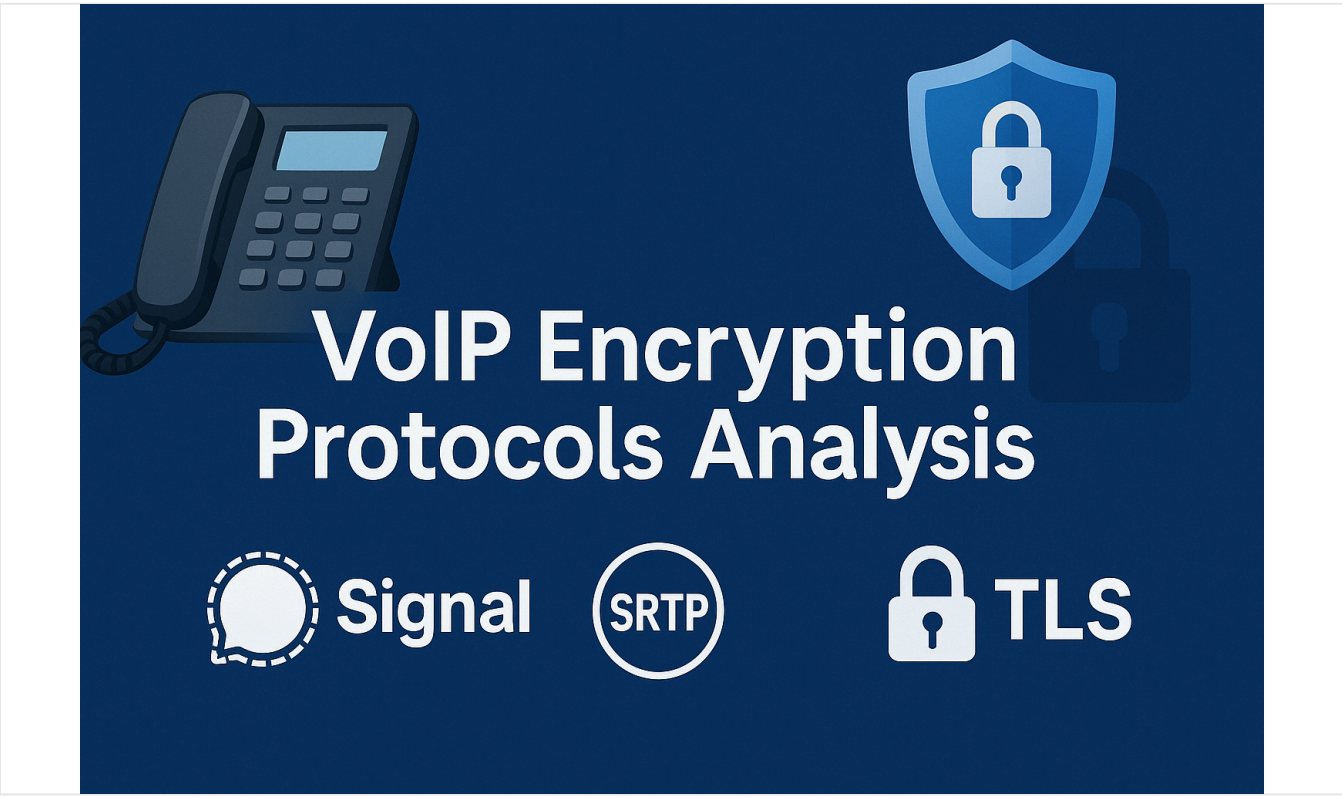
VoIP Encryption Protocols: SRTP, ZRTP, DTLS-SRTP, SIP-TLS Explained

By ClearlyIP Published February 3, 2025 60 min read



Encryption Protocols for VoIP: SRTP, ZRTP, DTLS-SRTP, SIP-TLS, and S/MIME – A Comprehensive Overview

Introduction

Voice over IP (VoIP) has become ubiquitous in modern communications, carrying sensitive voice and video conversations over IP networks. Ensuring the privacy and integrity of these calls is paramount – without encryption, attackers could intercept calls, eavesdrop on confidential conversations, or

manipulate signaling to reroute or hijack sessions. VoIP security generally operates on two levels: **signaling protection** (securing the call setup messages such as SIP invites) and **media protection** (securing the audio/video streams themselves). Over the years, several encryption protocols have been developed to address these needs. This report provides an in-depth examination of all major VoIP encryption protocols, including the Secure Real-Time Transport Protocol (SRTP) for media encryption, key exchange protocols like ZRTP and DTLS-SRTP that establish SRTP keys, and signaling-layer protections like SIP over TLS and S/MIME for SIP. We delve into how each protocol works, their architecture and cryptographic building blocks, how they integrate into VoIP systems, real-world implementation challenges, and performance and interoperability considerations. A comparison of their strengths, weaknesses, and typical use cases is also presented, along with notes on standards and implementations by bodies such as the IETF and ITU-T. The goal is to equip network engineers, cybersecurity analysts, and VoIP developers with a comprehensive understanding of VoIP encryption mechanisms and best practices for securing internet telephony.

Secure Real-Time Transport Protocol (SRTP)

Secure Real-Time Transport Protocol (SRTP) is the fundamental protocol for encrypting VoIP media streams (audio and video). Defined in RFC 3711 by the IETF in 2004 (Source: datatracker.ietf.org) (Source: datatracker.ietf.org), SRTP is a profile of the standard RTP (Real-Time Transport Protocol) that adds confidentiality, message authentication, and replay protection to RTP packets (Source: datatracker.ietf.org)(Source: learn.microsoft.com). In essence, SRTP protects the actual voice/video packets traveling between callers so that only authorized endpoints can decode and understand them, and it ensures that packets cannot be maliciously altered or replayed.

How SRTP Works: SRTP uses symmetric key cryptography to efficiently secure real-time streams. Each call (RTP session) is associated with secret session keys that are known only to the two endpoints (caller and callee) and used to encrypt and decrypt the media. SRTP employs strong encryption (by default, the **Advanced Encryption Standard** in counter mode) to encrypt the RTP payload, and applies a message authentication code (HMAC-SHA1 by default) to each packet for integrity checking (Source: digitalsamba.com). A typical SRTP packet thus consists of an encrypted RTP payload and an authentication tag (appended to the packet) that covers both the header and payload. The default cryptographic transforms in SRTP are AES-128 in Counter Mode (AES-CTR) for encryption and HMAC-SHA1 for authentication (using a 160-bit key to produce an 80-bit tag) (Source: datatracker.ietf.org). These choices were made to balance strong security with high performance; AES-CTR can encrypt data quickly with low CPU overhead, and the 80-bit HMAC tag offers strong integrity protection while adding only 10 bytes per packet. SRTP also derives a

sequence of **derived keys** from a single master key (using a key derivation function based on AES-CTR) so that each direction of media (and RTCP control traffic) uses separate keys and to enable periodic key refreshes (Source: datatracker.ietf.org). For replay protection, SRTP receivers maintain a **replay list** of recently seen packet sequence numbers; if an attacker tries to resend old packets, SRTP can detect the duplicate sequence numbers and reject those packets (Source: learn.microsoft.com). In summary, SRTP provides confidentiality by encrypting the media, authenticity by HMAC-signing it, and anti-replay by sequence checking – all with minimal overhead and preserving RTP's real-time characteristics. It "safeguards against eavesdropping, tampering, and replay attacks" while adding very little bandwidth or latency penalty (Source: digitalsamba.com).

SRTP Integration into VoIP Systems: An important aspect of SRTP is that it functions as a drop-in security layer for RTP without altering the basic RTP packet structure. This means SRTP can use the same UDP ports as RTP and is largely transparent to network elements that don't need to inspect the payload (Source: digitalsamba.com). In a typical VoIP call, once SRTP is in use, anyone capturing the packets will see only encrypted payload bytes (which appear as random data) instead of audio, and any attempt to modify packets will be caught by the authentication check. SRTP is **lightweight** enough for real-time use – its designers specifically tailored it for low overhead on network and CPU, making it viable even on bandwidth-constrained links or devices like IP phones (Source: digitalsamba.com). In fact, SRTP's security is often compared to IPsec's, except SRTP operates at the application layer for media and is optimized for the unique needs of real-time streams (e.g., it doesn't require reliable delivery or lengthy handshakes) (Source: digitalsamba.com). Because of this efficiency, SRTP has become "the standard for securing real-time multimedia" traffic such as VoIP and video conferencing (Source: digitalsamba.com). It is widely used in enterprise telephony, SIP trunking, and virtually all WebRTC-based communications for protecting media.

Key Management Considerations: Notably, SRTP itself focuses on encrypting and authenticating packets *given* a set of cryptographic keys – but it does not specify how the two endpoints agree on those keys. Securely exchanging or negotiating the SRTP keys is a separate challenge, handled by ancillary protocols. When SRTP was first introduced, several methods were proposed for key management (Source: datatracker.ietf.org). The most common approaches include in-band key exchange protocols like ZRTP (which runs over the media path) and out-of-band methods that use the call signaling channel (such as SDP Security Descriptions and MIKEY), as well as leveraging existing security handshakes (like TLS) to derive SRTP keys in a mechanism called DTLS-SRTP. Modern VoIP implementations have largely converged on a few standard key exchange protocols, which we explore in detail in the next sections. It's important to understand that SRTP is only as secure as the method used to obtain its keys – a fact that shaped the development of robust key management protocols for VoIP (Source: datatracker.ietf.org).

Key Exchange Protocols for SRTP

Before diving into ZRTP and DTLS-SRTP, which are major SRTP keying protocols, we briefly outline two other key management methods: **SDP Security Descriptions (SDES)** and **Multimedia Internet KEYing (MIKEY)**. These protocols were historically significant and are still used in certain scenarios to negotiate SRTP keys.

- **SDP Security Descriptions (SDES):** SDES is an approach where the SRTP keys are conveyed in the Session Description Protocol (SDP) of a SIP call, i.e. in the signaling plane. Despite the name similarity, SDES has nothing to do with the old Data Encryption Standard cipher; it refers to "Security Descriptions for Media Streams" (RFC 4568). In SDES, the calling party's SDP (for example, in a SIP `INVITE` message) includes one or more `a=crypto` attributes which contain the cryptographic parameters for SRTP – including the cipher and an encrypted form of the SRTP master key. The answering party's SDP then includes its own `a=crypto` attribute in the 200 OK response. In essence, the two endpoints exchange SRTP keys by piggybacking them on the call setup signaling (Source: commscouncil.uk). SDES was one of the earliest and simplest SRTP keying methods and became **widely implemented** in SIP phones and gateways (Source: commscouncil.uk). However, because the keys are transmitted in SDP, it is absolutely required that the SIP signaling is protected (e.g. via TLS) – otherwise an eavesdropper could read the keys and decrypt the call (Source: commscouncil.uk). SDES is straightforward and works with existing SIP infrastructure, but it provides **no end-to-end security** (the keys are visible to any SIP proxies that handle the SDP) and is vulnerable if any signaling hop is unencrypted or compromised. For this reason, the IETF now discourages SDES unless used within a fully secured and controlled environment (Source: rfc-editor.org)(Source: commscouncil.uk). Nonetheless, SDES support remains common in many SIP implementations due to its simplicity and legacy: for example, systems like Asterisk and many IP phones used SDES for years (often under the label "TLS/SRTP support"), relying on SIP over TLS to protect the key exchange.

- **Multimedia Internet KEYing (MIKEY):** MIKEY is a standalone key management protocol defined in RFC 3830 (2004) that was designed specifically to establish SRTP keys for real-time sessions (Source: rfc-editor.org). Unlike SDES, which is basically an SDP convention, MIKEY is an active protocol that exchanges messages (which can be carried in SDP or other signaling) to set up keys. MIKEY supports multiple modes of operation (it defines **four key distribution methods**) – including a pre-shared secret mode, a public-key encryption mode (where one side encrypts the SRTP key with the other's RSA public key), a Diffie-Hellman exchange mode, and even a mode based on sending tickets (Source: rfc-editor.org)(Source: rfc-editor.org). This flexibility allows MIKEY to be used in different scenarios: for example, a closed network might use a pre-shared

key for all devices, whereas a secure calling system could use Diffie-Hellman for per-call ephemeral keys. MIKEY is compact and designed for real-time use (its messages are kept small enough for inclusion in SIP/SDP). It also can support group keying for conferences and has extensions for various additional methods (Source: rfc-editor.org). In practice, MIKEY found adoption in specific domains – notably in **3GPP IP Multimedia Subsystem (IMS)** networks (the basis of VoLTE in mobile carriers), where it has been used to negotiate SRTP keys for voice calls within and between carriers. The ITU-T's H.323 standard even added an annex (H.235 Annex G) to use MIKEY for SRTP key management in H.323 systems (Source: itu.int). However, outside of telecom carrier infrastructure, MIKEY has seen limited use. Its complexity and the need for a pre-established trust (either pre-shared keys or a PKI for certificates) made it less popular in the general SIP world compared to lighter solutions. As we'll see, by the late 2000s the industry gravitated more toward DTLS-SRTP for general use, while MIKEY remains a niche solution for certain controlled environments.

With those additional methods noted, we now focus on the two prominent SRTP key exchange protocols specifically highlighted in our scope: **ZRTP** and **DTLS-SRTP**. These are often considered *end-to-end* keying methods, in that they allow two endpoints to agree on keys without any intermediary needing to see those keys – a critical property for true privacy.

ZRTP (Zimmermann Real-Time Transport Protocol)

ZRTP is a media-path key agreement protocol designed to provide secure, end-to-end key exchange for SRTP sessions. Developed by Phil Zimmermann (creator of PGP) and others, ZRTP was standardized as RFC 6189 (published for information in 2011) (Source: datatracker.ietf.org). The name "ZRTP" reflects its creators (Zimmermann) and its role in securing RTP. Uniquely, ZRTP operates *in-band on the RTP media stream itself*, rather than through the signaling channel. In other words, after a call is set up (e.g. the SIP INVITE/answer has taken place and the RTP sockets are established), the two endpoints use the RTP packets to perform a Diffie–Hellman key exchange and agree on SRTP encryption keys (Source: datatracker.ietf.org). This approach has several notable implications:

- **No Dependence on SIP or PKI:** ZRTP is completely independent of the signaling protocol – it doesn't require any SIP support or server involvement. The key negotiation packets are multiplexed on the same port as RTP, appearing as a series of RTP messages between the endpoints (Source: datatracker.ietf.org)(Source: icterra.com). As a result, ZRTP can work **peer-to-peer** and does not rely on any infrastructure like a certificate authority or registration server. It "does not require support in the signaling protocol" and "does not assume a Public Key

Infrastructure (PKI) or the complexity of certificates in end devices" (Source: datatracker.ietf.org). This makes ZRTP attractive in scenarios where users want maximal privacy without trusting service providers – even if someone intercepts the SIP signaling or if a malicious/compromised server is in the loop, it cannot interfere with or learn the ZRTP-negotiated keys.

- **Diffie-Hellman Key Exchange with Human Verification:** ZRTP uses an ephemeral Diffie–Hellman (DH) key exchange to establish shared secrets between the two VoIP endpoints. The DH exchange itself, if done alone, is susceptible to man-in-the-middle (MitM) attacks (an attacker could intercept the key handshake and insert themselves). To mitigate this, ZRTP introduces a clever user-level verification step: the **Short Authentication String (SAS)**. During the ZRTP handshake, both endpoints jointly compute a short checksum of the DH results, which is rendered as a short (typically 4-character or 4-word) string on each user's device. The users are then prompted to verbally read and compare this SAS with each other. If the strings match, it is virtually impossible that a MitM is present (an attacker would have to successfully guess the correct SAS code) (Source: icterra.com). If the SAS does not match, it indicates the possibility of an interception and users are warned that the call may not be secure (Source: icterra.com). This design leverages the human in the loop as a form of authentication – a distinctive approach that sets ZRTP apart from purely automated protocols. The SAS mechanism makes MitM attacks extremely impractical: even if an attacker managed to intercept the very first call between two parties, they would then have to guess a 16-bit SAS correctly to avoid detection (a *1 in 65,536* chance) (Source: icterra.com). Thus, ZRTP reduces MitM risk to a negligible level, assuming users perform the SAS check.

- **Key Continuity and Forward Secrecy:** ZRTP also implements a form of key continuity to further thwart intercept attempts. After a successful ZRTP session, the endpoints can store a shared secret (hashed) that is then mixed into the DH exchange of subsequent calls between the same parties (Source: icterra.com). This retained secret (sometimes called a cache or "ZRTP trusted secret") means that even if users *don't* verify the SAS on every call, an undetected MitM would have to have been present from the very first call ever, and stay present in all calls, to avoid detection. If a MitM appears in a later call, the retained secret will cause a mismatch in computed SAS, alerting the users. ZRTP also provides **Perfect Forward Secrecy** – since it uses ephemeral Diffie-Hellman, the SRTP keys are not derived from any long-term identity keys and are destroyed at the end of each session (Source: icterra.com). This means that even if one session's key were somehow compromised, it would not compromise past or future calls.

In practice, a ZRTP handshake occurs at the start of a call's media flow. The RTP channel is first used to exchange ZRTP messages (in a defined format that ordinary RTP receivers will ignore). The endpoints negotiate cryptographic parameters (ZRTP supports multiple DH groups, and cipher options for SRTP like AES types, etc.), perform the Diffie-Hellman exchange, compute the SAS, and confirm it. Once done, they derive the SRTP session keys and switch to secure SRTP for the actual conversation. If one of the endpoints doesn't support ZRTP, the protocol has a discovery mechanism (including an SDP attribute to signal ZRTP capability if desired (Source: datatracker.ietf.org)) and will simply fall back to regular RTP – meaning ZRTP can be deployed opportunistically without breaking calls (Source: datatracker.ietf.org).

Strengths and Limitations: ZRTP's strengths lie in its **independence and end-to-end security**. Not even the SIP servers or any intermediate devices know the keys or can decrypt the media – only the two end users who compared SAS know the call is secure (Source: icterra.com)(Source: icterra.com). It's an open standard and has been implemented in various softphones and tools (e.g., Phil Zimmermann's own Zfone project, the Jitsi client, certain versions of Linphone, etc.). However, ZRTP adoption in mainstream VoIP has been limited. One reason is that it requires changes on the client side (SIP proxies or carriers cannot force it if the endpoint doesn't support it) and not all vendors chose to integrate it. Also, it relies on user interaction (reading the SAS) for full security, which in some commercial scenarios is seen as an inconvenience. Additionally, enterprise or carrier environments often want to be able to monitor or record calls (lawful intercept or QA recording), which end-to-end ZRTP encryption can prevent – thus, such environments prefer hop-by-hop encryption where they can still decrypt at a trusted point. Nevertheless, ZRTP remains a compelling choice for peer-to-peer secure calling, especially in applications focusing on privacy (for example, it inspired aspects of secure calling apps and was used in early versions of Signal/RedPhone). Security analyses of ZRTP have been positive, showing it effectively achieves its goals as long as SAS verification is done (Source: icterra.com)(Source: icterra.com). In summary, ZRTP provides a **"zero-trust" VoIP encryption** – no third parties needed – making it uniquely suited for truly private conversations between two parties on the internet.

DTLS-SRTP (DTLS Key Agreement for SRTP)

DTLS-SRTP is a key exchange mechanism that uses **Datagram Transport Layer Security (DTLS)** to negotiate keys for SRTP. Unlike ZRTP's media-path approach, DTLS-SRTP leverages the well-known TLS handshake (adapted for UDP as DTLS) to perform a cryptographic negotiation between endpoints, then uses the result of that handshake to derive SRTP keys. This method was

standardized by the IETF in RFC 5764 (and the SIP framework for it in RFC 5763) in 2010 (Source: datatracker.ietf.org)(Source: datatracker.ietf.org), and it has since become the **de facto standard for securing media in WebRTC and many modern SIP deployments** (Source: rfc-editor.org).

Protocol Overview: DTLS is essentially TLS (the protocol behind HTTPS) modified to run over an unreliable transport (UDP) – it provides equivalent security (certificates, cipher suites, etc.) but with mechanisms to handle packet loss. In DTLS-SRTP, the two VoIP endpoints perform a DTLS handshake *with each other* on the RTP ports, before sending any actual media. This DTLS handshake is a standard TLS exchange: one side acts as the DTLS client and the other as the server. They exchange hello messages, exchange certificates (which can be self-signed or from a CA), perform a key exchange (typically an Elliptic Curve Diffie-Hellman these days), and verify the handshake messages. Once the DTLS handshake completes, the peers have established a shared secret and cryptographic context – normally, this would be used to encrypt application data in TLS. **However, in DTLS-SRTP, no actual media is sent through DTLS records.** Instead, the shared secret from the handshake is used to derive SRTP master keys and salts. The endpoints then switch to sending real-time media as SRTP packets (not inside DTLS) using those keys (Source: datatracker.ietf.org). In other words, DTLS is used only as a transient, out-of-band key negotiation channel; after that, the media flows as SRTP directly (which avoids the per-packet overhead of wrapping RTP in DTLS). The RFC illustrates this clearly: once the `use_srtp` DTLS handshake extension is negotiated by both sides, "the RTP or RTCP application data is protected solely using SRTP" and is no longer sent in DTLS `application_data` packets (Source: datatracker.ietf.org).

The flow is: peer A initiates a DTLS handshake to peer B; they exchange and verify certificates and agree on cipher suite (e.g. using ECDHE for forward secrecy); both compute the same key material; then they both extract SRTP keys from that material (using a defined mapping of the TLS "master secret" to SRTP master keys) (Source: soufianebouchaara.com). Finally, they send each other a confirmation (in the form of a special DTLS "Finished" message) and the DTLS session is established. Immediately after, they begin secure media transmission with SRTP using the keys derived. This sequence is typically very fast – often on the order of a few RTTs (round-trip times) which, in a LAN or good internet connection, might be tens of milliseconds. In practice, this handshake latency is negligible compared to human call setup times and often can be done in parallel with other call setup steps. As soon as the handshake is done, media flows securely. The use of DTLS also means the key exchange benefits from the mature security of TLS: strong authentication, support for modern ciphers, and well-tested implementations.

Certificate Handling and Authentication: One might wonder, if DTLS-SRTP exchanges certificates, who is the authority and how do endpoints verify each other? There are a few modes: in closed environments, endpoints could have CA-signed certificates (or device certificates issued by an

enterprise PKI) and thus truly validate each other's identity. But in most VoIP cases (especially WebRTC), endpoints do not have certificates from a common CA. Instead, DTLS-SRTP is usually deployed with **self-signed certificates** on each side. The authentication then relies on a technique called *certificate fingerprint matching*. Essentially, each endpoint includes in the call's signaling (SDP) a fingerprint (hash) of its certificate. For example, in SIP or WebRTC's SDP offer/answer, an attribute like `a=fingerprint:sha-256 12:34:...` is included (Source: datatracker.ietf.org)(Source: soufianebouchaara.com). The remote side, upon receiving the offer, knows in advance the expected hash of the certificate it will see in the DTLS handshake. During the DTLS handshake, the certificate presented is hashed and checked against the signaled fingerprint. If it matches, the certificate is accepted (even if self-signed); if not, the call is aborted. This way, the signaling channel (which is assumed to be integrity-protected, e.g. via SIP over TLS) carries the verification data to prevent man-in-the-middle attacks on DTLS. An attacker trying to intercept the media would also have to intercept and modify the signaling fingerprints – which is hard if the signaling path is secure, and would typically be detected. Thus, DTLS-SRTP's security against MitM is as strong as the signaling channel's integrity (or the trust in certificates if using a CA). In WebRTC, this mechanism is mandatory and automated – browsers handle it under the hood – and in SIP it's defined in RFC 5763 how to use it (Source: datatracker.ietf.org). If both sides trust each other's certificates (e.g. via a PKI or pre-known fingerprints), DTLS-SRTP can also provide true mutual authentication of endpoints, which is a bonus beyond just exchanging keys.

Usage and Performance: The adoption of DTLS-SRTP has been widespread. Notably, the WebRTC standard (driven by IETF and W3C) *mandates* the use of DTLS-SRTP for all peer-to-peer media – every WebRTC call in browsers uses DTLS-SRTP (with self-signed certs and fingerprint verification), providing encryption and authentication of media by default. This decision around 2013 effectively made "encrypted VoIP" the norm for browser communications. In the SIP world, DTLS-SRTP has also become the preferred approach for new deployments, especially with the rise of WebRTC-SIP gateways and modern IP phone equipment. The IETF's "Hitchhiker's Guide to SIP" in 2009 noted that of the three SRTP keying techniques considered (SDES, MIKEY, and DTLS), *DTLS-SRTP was chosen as the final solution going forward* (Source: rfc-editor.org). One reason is that it leverages TLS's proven security rather than inventing a new mechanism from scratch (Source: icir.org). It also operates end-to-end between endpoints (like ZRTP does) but without requiring user interaction – it fits seamlessly into the protocol. Another advantage is that DTLS-SRTP can be implemented relatively easily using existing TLS libraries (OpenSSL, NSS, etc.), which already support certificate handling and key negotiation. This eased adoption by developers. In terms of performance, after the handshake, using SRTP for the actual media means the ongoing encryption overhead is just the symmetric crypto (AES, etc.) which is very low. The handshake itself involves public-key operations (ECDHE, RSA or ECDSA for certs), but those occur only once at call setup. Modern cipher suites

using elliptic curve algorithms have made this quite efficient – often the CPU cost is on the order of a few milliseconds on a typical device, and hardware acceleration (if present for TLS) can offload much of it. Thus, DTLS-SRTP does not introduce any significant call latency or quality impact in practice. It *does* add a slight complexity in that media ports must accept DTLS handshaking packets (which are not RTP), but implementations handle this by demultiplexing DTLS vs RTP on the same port (using the first byte of the packet to distinguish protocols, as defined in RFC 5764 (Source: datatracker.ietf.org)(Source: datatracker.ietf.org)).

One limitation to mention is that DTLS-SRTP, by itself, is a hop-to-hop protocol if there are media relays or SBCs in the path. For instance, in a conference call scenario with an SFU (Selective Forwarding Unit) server, each DTLS-SRTP handshake is between an endpoint and the SFU (not directly between the two users), meaning the SFU terminates and re-encrypts the media. In such cases, the SFU (or any VoIP middlebox) can still access the media plaintext, which is acceptable in many use cases (for media processing or recording) but is not end-to-end encryption. Efforts like PERC (Privacy Enhanced Conferencing) are addressing that by layering end-to-end encryption on top of DTLS-SRTP, but those are beyond our scope. In pure peer-to-peer calls, DTLS-SRTP *is* end-to-end.

In summary, DTLS-SRTP combines the strengths of TLS's security with the real-time efficiency of SRTP. It ensures that VoIP media enjoys **data integrity, confidentiality, and even perfect forward secrecy** (when using ephemeral Diffie-Hellman in the handshake) (Source: soufianebouchaara.com). It has been widely standardized and adopted, making it a cornerstone of secure VoIP and video (especially in WebRTC-based applications). As one source puts it, DTLS-SRTP acts as a "guardian" against eavesdropping, tampering, and replay threats on the public internet, by establishing secure keys for media in a robust way (Source: soufianebouchaara.com).

SIP over TLS (Secure SIP Signaling)

While SRTP and its keying protocols protect the media content, it is equally important to protect the *signaling* of a VoIP call. The Session Initiation Protocol (SIP), being a text-based protocol akin to HTTP, can carry sensitive information: phone numbers, user identities (SIP addresses), and even credentials (for registration or proxy authentication). If SIP messages are sent in the clear, attackers could intercept calls (e.g., by reading and manipulating the SIP `INVITE` requests), harvest phone numbers or account data, or conduct session hijacking. **SIP over TLS (Transport Layer Security)** is the primary method to secure SIP signaling on the network. It establishes an encrypted TLS

connection for SIP traffic, typically on port 5061 (as opposed to 5060 for unencrypted SIP) (Source: support.telnyx.com)(Source: support.telnyx.com), ensuring that all SIP requests and responses between two nodes are encrypted and authenticated just like HTTPS web traffic.

Hop-by-Hop Encryption and SIPS: In SIP, the mechanism for using TLS is often tied to the use of the **SIPS URI scheme**. A SIP address like `sip:alice@example.com` indicates no particular transport security, whereas `sips:alice@example.com` mandates that the request be sent securely. According to the standards, using `sips:` means that each hop in the SIP routing (from the originator to proxies to the destination) **must use TLS**, and if a secure hop is not possible, the call should fail rather than fall back to insecure transport (Source: ietf.org). In effect, `SIPS` is a directive that the entire signaling path be TLS-protected. For example, if a caller's phone contacts its proxy over TLS, and that proxy needs to forward the request to another carrier, it must also use TLS to the next hop, and so on, all the way to the callee's device. This hop-by-hop model is important: TLS in SIP is not inherently end-to-end (unless there are literally only two hops, i.e., directly from caller to callee with mutual TLS). Each proxy decrypts the SIP message to process it (read headers, make routing decisions) and then re-encrypts when forwarding. Thus, one must trust the intermediate servers. The advantage, however, is that **on each network segment, the SIP traffic is protected** – eavesdroppers on the wire cannot read or alter the messages in transit. A user agent server (UAS) receiving a request over `sips:` knows that, according to spec, it should have been sent over TLS on every hop, though it ultimately has to trust the proxies to have followed the rules (Source: ietf.org) (Source: ietf.org). (The SIP standards note that a UAS cannot *guarantee* the entire path was secure, but SIPS is designed to enforce it among compliant agents (Source: ietf.org)(Source: ietf.org).)

Benefits of SIP-TLS: By running SIP over TLS, we gain **confidentiality and integrity** for the signaling. This means an attacker on the same network (e.g., someone with a packet sniffer on a public Wi-Fi or an ISP) cannot read the SIP messages or tamper with them. For instance, the SIP `INVITE` contains fields like the caller and callee addresses, call subject or metadata, and can include things like session keys (in SDP for SRTP via SDES) or authentication tokens. TLS ensures all of that is encrypted in transit (Source: blog.voip.ms). It also protects SIP authentication credentials: SIP often uses HTTP Digest authentication (which, while not sending plaintext passwords, can still be susceptible to replay or offline crack if intercepted). With TLS, the entire authentication exchange is inside the encrypted tunnel, thwarting man-in-the-middle attempts to steal or manipulate credentials (Source: blog.voip.ms). Furthermore, TLS provides server authentication via certificates, so that a UA can validate it's connecting to the genuine SIP server (preventing certain impersonation attacks, assuming a proper PKI or known certificate). Many service providers deploy SIP-TLS using standard X.509 certificates (the same kind as HTTPS). For example, a SIP provider like Telnyx or Twilio will have a certificate for their domain (sip.telnyx.com, etc.) and configure their proxies to listen on TLS

port 5061 with that certificate (Source: support.telnyx.com)(Source: support.telnyx.com). When a customer's device connects, it performs a TLS handshake just like a web browser, verifying the server's cert (often these certs are signed by public CAs or by private CAs that the device trusts). The encryption then "ensures that such information is indecipherable" to anyone except the client and server (Source: blog.voip.ms)(Source: blog.voip.ms). In short, SIP over TLS provides a secure pipe for signaling, preventing a host of potential attacks on VoIP such as call interception, credential sniffing, and session manipulation.

Deployment and Challenges: Using TLS for SIP is a well-established best practice and is supported by most modern SIP software (IP PBXs, softswitches, VoIP ATAs, softphones, etc.). It's often as simple as changing the transport to TLS and providing certificates. However, it is not universally enforced. In the early days of VoIP, many systems ran SIP over UDP or TCP with no encryption (for simplicity and performance reasons). Even today, some carriers and PBX vendors don't enable TLS by default. One reason is that TLS requires managing certificates and keys, which can be non-trivial for some IT staff. Additionally, SIP TLS is hop-by-hop: it only works if *every* proxy/server on the path supports and is configured for TLS. If any segment is unencrypted (e.g., two peering carriers that haven't set up TLS between them), then the chain is broken and using `sips:` would fail or be downgraded (which is not allowed by spec, but misconfiguration can occur). The SIP community has addressed some of these issues through clear guidelines (e.g., RFC 5630 focuses on SIPS URI usage and clarifications (Source: ietf.org)(Source: ietf.org)). In practice, within a single domain or between tightly federated domains, TLS is often used. For example, enterprises using SIP trunking will often have their IP-PBX connect to the provider's SBC via TLS, ensuring privacy of signaling over the public Internet. Major unified communication systems (Cisco, Avaya, Microsoft Skype for Business, etc.) use SIP-TLS as the default for client-server signaling in their deployments (Source: synchronet.net)(Source: synchronet.net).

The performance of SIP over TLS is generally not an issue; signaling traffic is relatively low bandwidth (compared to media) and the TLS overhead (handshake + encryption of a few messages) is negligible for typical call setups. There is some memory/CPU cost on servers to maintain many TLS connections (versus UDP which is stateless), but modern hardware handles thousands of TLS sessions easily, and re-use of TLS sessions or long-lived connections mitigates repeated handshakes. One must also consider SIP over TLS in contexts like emergency calling (e.g., 911) where intermediaries and PSAPs must support it to keep the call secured end-to-end; this is still evolving in standards and implementations.

In summary, **SIP over TLS** is the primary means of achieving signaling security in VoIP. It is a **hop-by-hop TLS encryption** of SIP messages providing confidentiality and integrity on each link (Source: ietf.org). It greatly raises the bar for attackers (they can no longer simply sniff SIP traffic or perform

on-path modifications easily). However, it assumes trust in the SIP service providers and proxies, since they see messages in plaintext. For truly confidential information exchange within SIP messages that even proxies shouldn't see, S/MIME (discussed next) is an additional tool – though rarely used. Nonetheless, the combination of SIP over TLS for signaling and SRTP for media has become the **industry-standard security baseline** for VoIP: "SIP uses TLS and SRTP for strong media and signaling protection," as a straightforward contrast with older H.323 systems that used their own encryption scheme (Source: synchronet.net).

S/MIME for SIP (Secure MIME Messaging in SIP)

Secure/Multipurpose Internet Mail Extensions (S/MIME) is a framework originally developed for secure email (providing end-to-end message encryption and signing using X.509 certificates). SIP adopted S/MIME as a method to secure certain parts of SIP messages end-to-end between user agents, independently of the hop-by-hop security like TLS. The idea is that an INVITE or other SIP request can carry MIME bodies (or even some headers) that are encrypted or signed such that only the originating and terminating user agents can decrypt/verify them, while intermediate proxies just treat them as opaque data. S/MIME in SIP was specified in the base SIP standard RFC 3261 and later updated (RFC 3853 mandated AES support, etc.) (Source: rfc-editor.org). In concept, S/MIME for SIP could, for example, encrypt the session description (SDP) or a text message within a SIP MESSAGE request so that only the far-end can read it (Source: mobius-software.com). It can also be used to digitally sign SIP messages or fragments thereof, providing end-to-end authentication of the sender's identity and message integrity even through proxies (Source: mobius-software.com).

How it Works: S/MIME uses public key cryptography via X.509 certificates. Each SIP user would have a certificate (containing their public key, typically signed by some certificate authority). When sending a SIP message, the user agent can include a MIME body that is of type `application/pkcs7-mime` – which could be, for instance, an encrypted block containing another MIME body (like an SDP). The recipient UA would use the sender's public key (or a shared secret) to decrypt, or if it's a signed message, use the sender's public key to verify the signature. SIP messages can thus be layered: a proxy sees the outer SIP headers (needed for routing) but the inner content can be protected. RFC 3261 originally mandated support for S/MIME with DES/3DES, but RFC 3853 updated this to require AES as the mandatory cipher for S/MIME in SIP (Source: rfc-editor.org), aligning with modern crypto standards. The mechanism is similar to secure email – indeed, it uses the same CMS (Cryptographic Message Syntax) structures. For instance, an INVITE could carry an S/MIME encrypted body that includes the SDP (with SRTP keys) and perhaps some Identity info. Only the callee's UA (with its private key) could decrypt that body and retrieve the SDP (ensuring that no

intermediary – not even the proxies or SBCs – could see the media keys). In theory, this provides **true end-to-end encryption of call setup** information. Similarly, S/MIME signatures could ensure that a header like the caller's identity (in a hypothetical signed Identity header or body) is verifiable by the recipient, guarding against impersonation in a way that even malicious proxies can't fake (this idea was an early approach to secure caller ID, although a different solution, STIR/SHAKEN, was later adopted for that purpose).

Deployment Issues: Despite being part of SIP's spec for decades, S/MIME in SIP has seen *very little real-world deployment* (Source: rfc-editor.org). Several factors contribute to this. First, **certificate management** for end users is non-trivial. Unlike HTTPS where servers have certificates and clients just trust them, in SIP every user agent would ideally have its own cert (or at least each domain has certs for users). There was no global PKI readily in place for SIP addresses (some proposals and services tried to fill this gap, but none became universal). The IETF even defined a "certificate service for SIP" (RFC 6072, the CERTS SIP service) to let clients fetch each other's certs to facilitate S/MIME (Source: rfc-editor.org), but this was complex and again not widely implemented. Secondly, many SIP proxies *need* to inspect or modify certain message parts (like SDP or specific headers for NAT handling, billing, etc.). If those parts are encrypted end-to-end, it can break functionality. For example, if SDP was S/MIME encrypted, a proxy that needs to do RTP proxying or codec rewriting would be blind. Because of this, S/MIME in SIP is most feasible in environments where proxies are mostly transparent and not offering value-added services on the SDP. In practice, most SIP networks found it sufficient to use hop-by-hop TLS to secure signaling, and relied on that, rather than adding the complexity of S/MIME.

There are, however, niche uses of S/MIME in SIP. Certain military or government communication systems that require end-to-end encryption and have a managed PKI have used S/MIME for securing mission-critical signaling (and sometimes media keys within that signaling). Also, S/MIME can secure the SIP MESSAGE method (instant messages in SIP) or presence information. In fact, RFC 8591 (2019) provides guidance on using S/MIME specifically for SIP messaging, improving interoperability for cases like MSRP or MESSAGE requests (Source: mobius-software.com). These use cases may see more uptake, since messaging could benefit from end-to-end encryption even if it goes through servers (similar to email). But for voice call setup, it remains rare.

Contemporary developments have somewhat eclipsed S/MIME for SIP. For instance, to verify caller identity, the IETF introduced a new mechanism (STIR, using PASSporTs in headers) rather than relying on S/MIME signatures of the whole INVITE. And for securing media keys, the rise of DTLS-SRTP (with fingerprint in SDP) provided a simpler path that doesn't require a full PKI per user. The

consensus in the industry is that SIP S/MIME is overly complex and impractical for broad deployment, a view echoed as early as 2009: "SIP S/MIME has seen very little deployment" (Source: rfc-editor.org).

Summary: S/MIME for SIP is an intriguing but underutilized tool. It **enables end-to-end encrypted and signed SIP message content**, using a methodology analogous to secure email (public-key encryption of MIME bodies) (Source: mobius-software.com)(Source: mobius-software.com). In theory, it addresses a threat model where even the service providers or proxies are not trusted with certain data. In practice, its deployment is confined to specialized scenarios. The challenges of certificate distribution and the complications with proxy operations have prevented it from gaining mainstream adoption. As a result, most VoIP security relies on TLS for signaling and SRTP for media, accepting that proxies are part of the trusted base. S/MIME remains part of the SIP standard suite and could see use in closed ecosystems that need that extra layer of security, but the average SIP user will likely never encounter it.

(It's worth noting that the lack of S/MIME usage in SIP was so pronounced that later analyses and guides explicitly call it out: for example, an IETF guide notes that despite the defined capability, "SIP S/MIME has seen very little deployment," and efforts were made to facilitate it via a certificate service (Source: rfc-editor.org) – indicating the gap between the theoretical standard and real-world adoption.)

Comparison of VoIP Encryption Protocols

VoIP encryption protocols span different layers and serve different purposes. Here we compare the major protocols discussed – SRTP (for media) and its key exchange methods (SDES, MIKEY, ZRTP, DTLS-SRTP), along with SIP-TLS and S/MIME for signaling – highlighting their relative strengths, weaknesses, and typical deployment scenarios:

- **SRTP (Media Encryption):** *Strengths:* Efficient, low overhead encryption tailored to real-time traffic; provides strong confidentiality, integrity, and replay protection for RTP/RTCP (Source: digitalsamba.com)(Source: digitalsamba.com). Widely supported by VoIP equipment and mandated in WebRTC, making encrypted media ubiquitous. Uses well-vetted ciphers (AES) and allows algorithm agility (e.g., newer suites like AES-GCM for authenticated encryption have been added to SRTP in RFC 7714). *Weaknesses:* SRTP on its own does not handle key exchange – it relies on external protocols to function. Thus, its security is only as good as the key management in use. Also, SRTP is only point-to-point; if media flows through a relay (SFU or mixer), that device must either be trusted with keys or else special end-to-end schemes must be applied

(e.g., double encryption). *Use Cases:* Virtually all VoIP/media applications can and do use SRTP for encryption: enterprise IP phones, softphones, mobile VoIP apps, and conferencing systems. SRTP is the default in any scenario where media confidentiality is needed, ranging from SIP trunk calls to Zoom/Teams meetings (often combined with DTLS or SDES for keys). Performance-wise, SRTP's overhead is minimal (e.g., ~32 bits of RTP header may remain unencrypted for functionality, but payload and extension encryption is complete), so quality of service is maintained.

- **SDES (Key exchange via signaling):** *Strengths:* Simplicity – easy to implement since it's just an SDP attribute; widely implemented in legacy equipment making it a **most common denominator** for SRTP keying in many SIP deployments (Source: commscouncil.uk). Doesn't require user interaction or complex handshakes, and fits naturally into call signaling. *Weaknesses:* Not secure unless signaling is secure (leaks keys if someone intercepts SIP); even with TLS signaling, it's not truly end-to-end (any intermediate proxy that sees SDP can see the keys). Offers no identity verification of keys (it assumes the SIP path is trusted). Essentially provides encryption against outsiders but not against insiders who handle signaling. *Use Cases:* Still used in many enterprise systems and SIP trunks where both ends are within a known environment (and where TLS is used to protect SIP). For example, an IP phone registering to a corporate PBX might use SDES because both phone and PBX support it and the SIP link is TLS. SDES is often the fallback method if newer ones aren't available. However, due to security concerns, new systems (especially involving untrusted networks) avoid SDES.

- **MIKEY (Key management protocol):** *Strengths:* Flexible – supports multiple modes (PSK, public-key, DH) (Source: rfc-editor.org), including group keying and advanced features. It can operate in constrained environments (designed to be efficient, with small messages). When used with the right mode (e.g., Diffie-Hellman), it can provide PFS and other security properties. *Weaknesses:* Complexity – implementing all modes is non-trivial; requires a pre-existing trust infrastructure (either pre-shared secrets distributed, or a PKI for certificate mode). Not widely implemented outside certain sectors. Harder to get different vendors' MIKEY implementations to interoperate unless they support the same modes and parameters. *Use Cases:* Primarily in 3GPP IMS and cellular networks (VoLTE) where network operators can manage the keys or certificates. For instance, in VoLTE, when a call is set up, the phones and the network may use MIKEY (carried in SIP/SDP or in IPsec) to exchange SRTP keys under the hood – the users don't see this, but it's happening per standards. Some military or mission-critical comms systems (which have their own PKI) might use MIKEY to ensure only intended parties get the keys. Outside these, MIKEY is seldom seen; most SIP phones don't support it or if they do, it's a single mode like pre-shared which isn't interoperable universally.

- **ZRTP (In-band DH exchange with SAS):** *Strengths:* True end-to-end key exchange independent of servers – **no reliance on infrastructure or even on a PKI** (Source: datatracker.ietf.org)(Source: icterra.com). The SAS verification provides a very high assurance against MitM (with human confirmation) (Source: icterra.com). Provides perfect forward secrecy and key continuity for enhanced security (Source: icterra.com). ZRTP can work across any network as long as the two endpoints support it, even if the call is routed through untrusted or hostile networks. Doesn't require prior arrangements or accounts; works opportunistically. *Weaknesses:* Requires both endpoints to implement ZRTP (not universally supported by all VoIP products, especially many commercial ones omitted it). The need for user SAS verification, while a security strength, can be seen as a usability hurdle – users may neglect to do it, or it might be impractical in some call scenarios. Without SAS checking, ZRTP is vulnerable to MitM (though with key continuity even that is mitigated after the first call). Also, ZRTP only handles key exchange; if a call involves more than two endpoints (like a conference with a bridge), ZRTP isn't end-to-end (each leg could do ZRTP, but the conference bridge would be an intermediary that terminates and reoriginates media). *Use Cases:* ZRTP is favored in privacy-focused applications. It's been used in some encrypted VoIP apps and devices (e.g., the original Silent Circle phone service, some open-source softphones like Jitsi, CSipSimple, etc.). It's ideal for peer-to-peer calls where users want to ensure no eavesdropping even by their provider. For instance, two lawyers calling directly might use a ZRTP-capable app to ensure their conversation remains confidential to them only. Adoption in enterprise or carrier scenarios is rare; those environments typically prefer managed security (DTLS-SRTP or SDES where the servers can still assist or at least know the keys if needed). Essentially, ZRTP shines in **decentralized or user-driven security contexts**.

- **DTLS-SRTP (TLS handshake for keys):** *Strengths:* Leverages the robustness of TLS – uses standard cipher suites, can piggyback on existing TLS implementations, benefits from decades of TLS security research (Source: icir.org). Provides mutual authentication if needed (via certificates), and always provides *cryptographic binding* of the handshake to the signaling (through fingerprint exchange) to prevent MitM (Source: datatracker.ietf.org)(Source: soufianebouchaara.com). Has perfect forward secrecy by default (since typically an ephemeral Diffie-Hellman exchange is part of the TLS handshake). Once set up, uses SRTP so has minimal ongoing overhead. Widely adopted and interoperable – all major WebRTC endpoints use it, many SIP endpoints now support it, making it a modern standard. *Weaknesses:* A bit more complex to set up than SDES (requires implementing DTLS and handling certificates). In pure SIP land, not all older devices support DTLS-SRTP, which can cause compatibility issues (e.g., a newer WebRTC endpoint might only do DTLS-SRTP, but an older SIP phone only does SDES – making direct secure media negotiation impossible without an intervening gateway). Hop-by-hop nature

in multi-hop calls (as discussed, a media relay will terminate DTLS-SRTP, meaning not *end*-to-end if you trust only the endpoints). Also, while typically automated, any mismanagement of the fingerprint (like not verifying it) could open a hole (though in standards-based usage this is well handled). *Use Cases:* WebRTC (browser-based communications) **mandatorily** uses DTLS-SRTP, so any web-based meeting, call, or peer connection is secured by it. For SIP trunking or enterprise, many have begun using DTLS-SRTP especially when integrating with WebRTC or for new secure softphones. For example, a cloud PBX might offer WebRTC webphone clients – those use DTLS-SRTP – and still talk to legacy phones via a gateway that might do SDES on the other side. Over time, DTLS-SRTP is expected to replace SDES in most scenarios as equipment is updated, because of its better security properties. It is effectively the **preferred key exchange for any open network or internet-facing call**, as it protects against active attackers on the network, which SDES could not do.

- **SIP over TLS (Signaling Encryption):** *Strengths:* Widely supported and deployed; fairly easy to enable with server certificates. Provides a good layer of defense for call control messages, preventing eavesdropping or tampering by unauthorized parties on the network (Source: blog.voip.ms). Essential for protecting SIP credentials (which, if stolen, could be used for fraud like making expensive calls on someone else's account). Also critical for privacy: without SIP-TLS, an eavesdropper could even see phone numbers or user IDs being called. TLS is a proven standard and, when used with proper certificate validation, can thwart man-in-the-middle attacks on the signaling path. *Weaknesses:* It is **hop-by-hop**, so the user must inherently trust the SIP service providers and any transit proxies – each of those sees the full signaling in plaintext. If any proxy is compromised or malicious, that hop could be attacked (though such an attacker would still be constrained to that hop – they couldn't eavesdrop beyond it). Deployment sometimes falters if any segment of the call path doesn't support TLS, causing interoperability issues (though the use of SIPS is supposed to enforce it, in practice some carriers would simply not connect calls if they're forced to TLS and can't). Additionally, managing certificates for many endpoints (in scenarios where mutual TLS is used) can be complex, though typically only servers use certs and clients authenticate by credentials. *Use Cases:* Almost all secure VoIP deployments: Enterprises use TLS for SIP trunks (so that registration and call setup to the SIP provider is encrypted), VoIP cloud providers provide a TLS port for clients (e.g., many ITSPs document how to configure your phone for TLS+SRTP). Within corporate networks, if VoIP travels over secure LANs, some might skip TLS internally but use it externally. Many modern IP phones (Cisco, Polycom, etc.) are configured by default to use TLS to the call server, both for SIP and for configuration downloads. Regulatory and privacy requirements in certain industries (finance, healthcare) often mandate encrypted communications, so TLS is turned on in those environments. Essentially, whenever SIP signaling might traverse networks where others could

observe it (the public Internet, Wi-Fi, etc.), TLS is or should be used. In closed telecom interconnects (like between two telecom operators over a leased line), they may or may not use TLS, but the trend is increasingly toward encryption everywhere.

- **S/MIME for SIP:** *Strengths:* Only mechanism to provide **end-to-end signaling security** – even the proxies can't read the protected parts if used. This is important if, for instance, the content of a SIP MESSAGE or a SIP SUBSCRIBE (with sensitive data) needs to be kept confidential from the service provider carrying it. It can also prove the identity of the sender to the recipient with a cryptographic signature, which is stronger than hop-by-hop authentication (which only proves to each proxy). S/MIME is based on well-understood PKI (same as email), so in theory it can leverage existing certificate infrastructures. *Weaknesses:* Rarely implemented, so even if one UA wants to use it, the other might not understand it or might not have a certificate. Tied to the cumbersome world of PKI for users – distributing and verifying user certs is non-trivial. Proxies that can't see encrypted bodies might break certain services (for example, if the SDP is encrypted, a proxy can't do call forking decisions based on SDP, or can't insert a media relay if needed – unless those functions are moved to endpoints). In general, the industry saw it as overkill for most VoIP needs given that TLS already secures things to a reasonable degree in most models. *Use Cases:* Almost extinct in the wild. Potentially applicable in extremely high-security environments where not even the network operator is fully trusted – for example, a government might mandate that even if they outsource some SIP infrastructure, the actual message contents must be unreadable to the provider. Or for secure messaging using SIP (some systems might send secure instant messages or files via SIP MESSAGE or INFO, and use S/MIME to encrypt those payloads end-to-end). With the renewed attention on end-to-end encryption in messaging apps (like Signal, etc.), one could imagine a resurgence in interest for S/MIME-like functionality in SIP messaging, but it would likely use newer techniques rather than the literal S/MIME spec. For voice call setup, practically no standard SIP services use S/MIME as of 2025, and it remains a theoretical tool.

The table below summarizes some key comparative points:

PROTOCOL	PURPOSE	SECURITY SCOPE	KEY EXCHANGE / AUTH MODEL	TYPICAL USE TODAY
SRTP	Encrypt media (RTP/RTCP)	Packet confidentiality, integrity, replay protection for voice/video (Source: digitalsamba.com).	Symmetric keys (per session) – requires external key agreement.	Virtually all secure VoIP (SIP, WebRTC, etc.) uses SRTP for media encryption. Standard in WebRTC, VoLTE, enterprise VoIP (Source: digitalsamba.com).
SDES (SDP Security Descriptions)	SRTP key exchange via signaling (SDP)	Hop-by-hop (depends on SIP TLS for security) (Source: commscouncil.uk); not end-to-end.	Keys exchanged in SIP offer/answer (in SDP) (Source: commscouncil.uk); trust relies on signaling path.	Legacy and interoperability. Still common in older SIP phones, trunking, where both sides support it and TLS is used. Being phased out for Internet-facing calls (Source: rfc-editor.org).
MIKEY	SRTP key exchange protocol	End-to-end between endpoints (carried in signaling or elsewhere).	Flexible: PSK, RSA, DH modes (Source: rfc-editor.org); can authenticate via shared secrets or certs.	Used in specialized systems (e.g., telecom IMS, military). Rare in general SIP endpoints.
ZRTP	SRTP key exchange in RTP media path	End-to-end (true peer-to-peer) (Source: datatracker.ietf.org). Does not trust or involve servers.	Diffie-Hellman key agreement with SAS verification (user-authenticated) (Source: icterra.com). No PKI	Niche use for high-confidentiality calls (certain secure softphones/apps).

PROTOCOL	PURPOSE	SECURITY SCOPE	KEY EXCHANGE / AUTH MODEL	TYPICAL USE TODAY
			needed (Source: icterra.com).	Not widely supported by mainstream IP phones or mobile dialers.
DTLS-SRTP	SRTP key exchange via DTLS handshake	End-to-end between endpoints (unless a middlebox terminates it) (Source: datatracker.ietf.org).	DTLS (TLS) handshake with certificates; typically verified via fingerprint in signaling (Source: datatracker.ietf.org). Provides PFS (ECDHE) and strong auth.	**Current standard** for WebRTC and many new SIP devices (Source: rfc-editor.org). Used whenever possible for internet calls, bridging WebRTC to SIP, etc.
SIP over TLS	Encrypt SIP signaling (hop-by-hop)	Secure transport on each network hop (Source: ietf.org); proxies decrypt and re-encrypt.	TLS handshake with X.509 certificates (server auth, optional mutual auth) (Source: support.telnyx.com) (Source: support.telnyx.com).	Widely deployed for securing SIP trunks, register, and call signaling over untrusted networks. Often mandated by policy for VoIP providers and enterprises.
SIP S/MIME	End-to-end encrypt/sign parts of SIP message	End-to-end between UAs for the protected parts (Source: mobius-software.com) (proxies can't read those).	Uses X.509 public-key encryption and signatures (CMS) (Source: mobius-software.com). Requires shared CA or exchanged certs.	Rarely used. Potentially in closed environments with a PKI. Largely theoretical in public inter-domain SIP.

In essence, the choice of protocols often comes down to balancing **security needs vs. practical deployment constraints**. For most applications today, the combination of SIP over TLS for signaling and DTLS-SRTP (or SDES in older systems) for media forms a sufficient security architecture: it protects against outside eavesdroppers and opportunistic attackers, which is the threat model for most VoIP deployments. ZRTP and S/MIME cater to a stronger threat model – protecting against even insiders or infrastructure – but at the cost of needing all parties to explicitly support those methods and possibly involve users (for ZRTP SAS). Thus, those see use mainly where maximal privacy is required and users are willing to participate in the security process.

Standards, Implementations, and Recent Developments

VoIP encryption protocols have been shaped by various standards organizations and have seen numerous implementations, both open-source and commercial:

Standards Bodies: The **Internet Engineering Task Force (IETF)** has been the primary body defining VoIP security protocols in the SIP era. IETF RFCs cover SRTP (RFC 3711) (Source: datatracker.ietf.org), key management like MIKEY (RFC 3830) and DTLS-SRTP (RFC 5764) (Source: datatracker.ietf.org), as well as SIP-S/TLS and S/MIME for SIP (RFC 3261, 3853) (Source: rfc-editor.org). The IETF's focus has been on open interoperable standards to enable any SIP endpoints to communicate securely. The **ITU-T**, which was behind H.323, addressed VoIP security in its H.235 series. H.235 defines security for H.323 signaling and media, including the use of AES encryption and even the adoption of SRTP in later versions. For example, ITU-T H.235 Annex G mapped the IETF's MIKEY and SRTP into H.323 usage (Source: itu.int), and H.235.6 defined mechanisms analogous to SRTP for H.323 streams. However, as the industry gravitated to SIP, the IETF solutions (TLS, SRTP, etc.) became the dominant ones. In fact, a succinct comparison is that *H.323 uses H.235 encryption, whereas SIP uses TLS and SRTP for strong protection* – both can meet strong security requirements, but via different standard suites (Source: synchronet.net)(Source: synchronet.net). Another relevant body is **3GPP** (3rd Generation Partnership Project), which in its IMS specifications for mobile networks references these protocols (e.g., 3GPP uses IPSec at the access layer for SIP signaling from phones, and mandates SRTP for media in IMS with keying via either MIKEY or SDES depending on scenario). **IEEE** doesn't directly define VoIP encryption, but things like IEEE 802.11 (Wi-Fi) and 802.1X add network-level encryption that can complement VoIP security (though not VoIP-specific). Overall, collaboration between these bodies has ensured that, for instance, a voice call from an LTE phone to a VoIP softphone can be fully encrypted using standardized methods end-to-end (with carriers typically using DTLS-SRTP or IMS security on their side).

Open-Source Implementations: There is a rich ecosystem of implementations for these protocols, which has greatly aided adoption. For SRTP, a widely used library is **Cisco's libSRTP**, an open-source C library implementing SRTP and its cryptographic transforms (Source: github.com). This library is used in many applications (from Asterisk PBX to Firefox's WebRTC stack) to handle SRTP encryption/decryption. For ZRTP, Phil Zimmermann's team provided libraries like **GNU ZRTP (part of GNU ccRTP)** and others, and projects like Jitsi and PJSIP integrated ZRTP support. The Jitsi client, for instance, allowed ZRTP secure calls with SAS verification in its softphone (Source: icterra.com). On the DTLS-SRTP side, popular SSL/TLS libraries such as **OpenSSL**, **GnuTLS**, and Mozilla's NSS all support DTLS and have the extensions needed for SRTP key export – meaning any application using those libraries can relatively easily implement DTLS-SRTP. WebRTC's reference implementation (in WebRTC.org library used by Chrome, etc.) uses a combination of libSRTP and OpenSSL to implement DTLS-SRTP. **PJSIP**, a popular open-source SIP stack, supports SIP over TLS, SRTP (via SDES and DTLS-SRTP), and even has options for ZRTP and MIKEY, making it a comprehensive toolkit for secure communications. On the server side, open-source SIP servers like **Asterisk** and **FreeSWITCH** have long supported TLS and SRTP (initially via SDES, and now also DTLS for WebRTC endpoints). These projects also utilize open crypto libraries, and in some cases hardware cryptography (e.g., via OpenSSL engine) to efficiently handle many calls.

Commercial Implementations: Virtually every major VoIP vendor has incorporated these protocols. Enterprise IP phone systems (Cisco Unified Communications Manager, Avaya Aura, Microsoft's Lync/Skype for Business/Teams, etc.) all provide options for TLS signaling and SRTP media for their phones and soft-clients (Source: synchronet.net). Often these systems come with their own certificate management tools to ease deployment (for example, Cisco phones can automatically download certificates from the call manager). VoIP service providers (from large ones like Vonage or Twilio to smaller ITSPs) typically offer encrypted trunks – customers are given a SIP TLS endpoint to register to. Many VoIP phones (Polycom, Yealink, Cisco, etc.) now support SIP-TLS/SRTP out of the box, sometimes even mandating it for certain services (e.g., some providers require encryption for authentication reasons). In the realm of conferencing and video, products like Zoom, Webex, Microsoft Teams, Google Meet all rely on DTLS-SRTP (since they are built on WebRTC or similar), meaning media is encrypted for all those calls. Even when those calls go through centralized servers (for mixing or recording), the links from clients to server are SRTP encrypted, and the servers don't divulge media in plaintext except as needed internally. Some of these services have also introduced true end-to-end encryption modes (Zoom, for example, introduced an E2E mode in 2020 where the meeting key is agreed via a participant-driven DH and not even their servers get it – akin to ZRTP's philosophy). This shows a general trend of pushing the boundaries of VoIP encryption further.

For mobile applications, many messaging apps that offer voice/video calls (WhatsApp, Signal, FaceTime, etc.) use custom encryption protocols rather than the standard SIP/SRTP, but under the hood they share similar building blocks. For instance, Signal's voice calling once used ZRTP, but later moved to a custom key exchange (the Signal Protocol's Double Ratchet) – still, the media encryption in those calls is effectively SRTP (with a different key management). Apple's FaceTime uses its own signaling but the media encryption is a variant of SRTP with AES-256 and peer verification via Apple's identity services (Source: csrc.nist.gov). These are proprietary, but they show that SRTP and strong key exchanges are universally recognized as necessary.

Recent Advancements: Recent years have seen enhancements rather than entirely new protocols in VoIP encryption. Some advancements include:

- **Stronger Cryptographic Algorithms:** SRTP has been extended to support AES-GCM, an authenticated encryption mode that combines encryption and integrity in one step for higher security and performance (Source: potaroo.net). AES-GCM (and ChaCha20-Poly1305 in some cases) are now available in WebRTC and other implementations, offering a boost in efficiency on hardware that supports them (AES-GCM can be hardware accelerated on modern CPUs). There's also support for 256-bit keys (AES-256) if needed, though 128-bit is deemed sufficient for now (Source: potaroo.net). The introduction of new ciphers is usually in response to future-proofing security (e.g., against quantum computing, larger keys might be considered).

- **Double Encryption for Conferences (PERC):** The IETF has worked on the PERC project (Privacy Enhanced RTP Conferencing) which involves "double encryption" – basically layering an end-to-end SRTP encryption on top of a hop-by-hop one, so that a conferencing server can still mix or route media without having the actual media plaintext (Source: rfc-editor.org). In PERC, each participant shares an end-to-end key with others (through a key management service) and the media is encrypted first with that, then with a hop key known to the server. The server removes the outer layer (so it can do its job with packet headers) but still never sees plaintext. This is cutting-edge and not widely deployed yet, but shows the direction for enhancing privacy even in multi-party calls.

- **Integration with Identity and Verification:** Protocols like STIR/SHAKEN (for caller identity in SIP) don't encrypt media or signaling, but they use certificates to sign calling numbers to combat robocalls. This indicates an increasing use of cryptography in VoIP for various purposes. We might see, for instance, closer integration between identity verification and media encryption (so that you not only have a secure call, but you know *who* you have a secure call with). ZRTP's

SAS indirectly did this (as you typically verify the voice of the person when reading the SAS). The future might combine protocols – e.g., use an identity token along with a DTLS-SRTP handshake to assure the key is bound to a known identity.

- **Improved Performance and Scaling:** Implementations have optimized over time – e.g., WebRTC implementations now use ICE and DTLS in parallel to shorten call setup, and can handle thousands of DTLS-SRTP handshakes per second on server hardware, enabling large-scale encrypted conferencing. Hardware support for SRTP (in DSPs or NICs) exists in some enterprise SBCs to offload the encryption so that even large call centers can run all calls encrypted without extra lag.

- **Post-Quantum Considerations:** While not yet in standards, there is research on how to make VoIP key exchanges resilient to quantum attackers. This could mean using post-quantum algorithms in the DTLS handshake or in a future ZRTP variant. The crypto community will likely push updates to DTLS (as part of TLS 1.3+ evolution) to support post-quantum key exchange, which would trickle down to DTLS-SRTP usage in VoIP when needed.

In terms of **interoperability**, today one can generally mix and match as follows: A call between a WebRTC endpoint and a SIP phone can be secured by having a gateway translate between DTLS-SRTP and SDES or between DTLS-SRTP and plain RTP (if the phone doesn't support encryption, though that loses security). Many SBCs now can do **DTLS-SRTP termination** (act as a DTLS endpoint to the WebRTC side) and then re-encrypt to SRTP (SDES) for the SIP phone side – not ideal in purity (since the SBC has the media in plaintext), but at least the call is encrypted on both network segments. Over time, as more SIP devices support DTLS-SRTP, we expect end-to-end DTLS-SRTP without such termination. ZRTP, being end-to-end, only works if both ends support it; it will simply not engage otherwise. So interoperability with ZRTP is basically an on/off thing – it either secures the call if both sides have it, or the call just goes unsecured (or secured by other means) if not. There's no network device that can "translate" ZRTP to something else, because that would involve knowing the keys (which defeats the purpose). S/MIME likewise requires both ends. In practice, the mainstream path is clearly toward universal support of SRTP with DTLS keying and TLS signaling, as it balances security with compatibility.

To illustrate the state of deployments: The **Comms Council UK** noted in a 2017 briefing that "the majority of SIP deployments use UDP... in clear text" but also that best practice is moving to TLS, and that among key exchange methods, SDES was the most widely implemented but that newer approaches like DTLS-SRTP were the way forward (Source: commscouncil.uk)(Source: rfc-editor.org). Since then, especially with WebRTC's influence and increasing security awareness, adoption of TLS/SRTP has accelerated. In some regions and sectors, unencrypted VoIP is now the exception rather than the norm.

Open-Source and Commercial Support Summary: Today you can pick virtually any major open-source VoIP software and get support for encryption: e.g., Asterisk (yes), FreeSWITCH (yes), Kamailio/OpenSIPS (they proxy SIP messages and can pass TLS, and while they don't handle media, they can coordinate keys or act as certificate authorities for SIP Identity headers etc.), Jitsi (yes, as a client and they also have Jitsi Meet which uses WebRTC encryption), Linphone (yes, supports SRTP with multiple key exchange options including ZRTP and DTLS), and so on. On the commercial side, providers like **VoIP.ms** or **Twilio** publish guides strongly encouraging customers to use SIP-TLS and SRTP, calling it "the best encryption channel for your business VoIP traffic" (Source: blog.voip.ms). Enterprise vendors often have whitepapers on how they secure VoIP: for example, Cisco's SRTP is FIPS 140-2 validated in some of their products, and Microsoft's Lync/Skype enforced TLS and SRTP (with their own flavor called MS-SRTP which was based on SRTP). Even the telecom carriers are getting on board for inter-carrier links (with initiatives to use TLS/IPsec between SBCs of different providers to thwart call interception by third parties).

In conclusion, the standards are mature and the implementations are plentiful. The remaining challenges are largely about **ensuring consistent use** (turning on the features, distributing certificates properly, etc.) and **keeping up with new threats** (which might necessitate protocol tweaks down the line, as mentioned).

Conclusion and Future Outlook

VoIP encryption protocols have evolved into a robust suite that, when applied correctly, can secure voice and video communications to a very high degree. Today's technical professionals have at their disposal a layered defense: **SIP over TLS** protects call setup and tear-down signaling from prying eyes and tampering, while **SRTP (with a strong key exchange like DTLS-SRTP or ZRTP)** protects the media content, ensuring conversations remain confidential and authentic. Together, these measures address the vast majority of VoIP threat vectors, from casual network sniffing to sophisticated man-in-the-middle attacks.

One of the key themes in this evolution is the move toward **end-to-end security**. Traditional telephony assumed the network (telephone exchanges, etc.) was trusted, but IP telephony operates over the public internet and zero-trust environments. Protocols like ZRTP and S/MIME attempted to provide true end-to-end encryption where even service providers need not be trusted with keys. While those haven't become universal, the philosophy is seen in modern systems (for instance, WhatsApp and Signal delivering end-to-end encrypted calls to millions of users, albeit using different protocols). There is an increasing demand by users that their communication apps be **secure by default**. In the standards world, this was exemplified by WebRTC requiring encryption in all sessions

(no option for unencrypted media). We can expect this trend to continue: unencrypted VoIP will become increasingly rare. Regulatory pressures (e.g., GDPR's focus on protecting communications, or industry-specific regs for healthcare/finance) also incentivize encryption everywhere.

Looking ahead, we might see further convergence of identity and encryption. One challenge that remains is how to **ensure the person on the other end is who they claim to be, in a cryptographic sense**. S/MIME tried to leverage certificates for this, and ZRTP uses SAS with human verification. The STIR/SHAKEN framework addresses the calling number authenticity (preventing Caller ID spoofing) by having carriers sign calling numbers with certificates, which is complementary to encryption (it doesn't encrypt but prevents impersonation). A fully secure VoIP system of the future may combine these so that you get both a secure channel and a verified identity (e.g., your device might show a "green lock" for encryption and a checkmark that the caller's identity was cryptographically verified). This would mirror what we have on the web today (HTTPS padlock with an SSL certificate proving the site's identity).

Another area is the **post-quantum era**. While current VoIP encryption (AES, SHA, ECDH) is considered very secure against today's adversaries, the emergence of quantum computing in the future could threaten the public-key components (like Diffie-Hellman or RSA). Standards bodies are already preparing post-quantum algorithms. It's likely that protocols like TLS (and thus DTLS-SRTP) will get post-quantum cipher suite options. When that happens, VoIP products will need to update so that negotiations use quantum-resistant key exchanges, ensuring long-term confidentiality of voice data. The symmetric encryption of SRTP (AES) is expected to remain strong for the foreseeable future (and can be upgraded to larger key lengths easily). So, much of the groundwork in VoIP encryption will extend into the future with iterative improvements.

From an interoperability and adoption standpoint, one remaining challenge is to bring all devices up to parity. As of 2025, there are still some legacy systems that either don't support encryption or use outdated methods (like SDES without TLS, or old algorithms like triple-DES for S/MIME). Professionals upgrading systems should ensure that encryption capabilities are considered a must-have. Fortunately, most new equipment has these features by default. A push from management and regulatory compliance often drives the final step – enabling and requiring encryption in configuration. For example, a company might mandate that all SIP endpoints register via TLS and negotiate SRTP or else calls are rejected. Such policies are becoming more common.

On the flip side, encryption does complicate some things: **Lawful interception (LI)** is one. When a law enforcement agency needs to tap a VoIP call, end-to-end encryption can be an obstacle. Most enterprise and carrier solutions address this by doing encryption in a way that a trusted server still can access media (e.g., a call recording system is integrated at the PBX which has the keys). In truly

end-to-end scenarios (like two Signal users), lawful intercept is nearly impossible without access to the endpoints themselves. This raises policy questions beyond the technical realm. It's possible we'll see more debates and possibly technical solutions that allow a form of lawful intercept that doesn't weaken security for everyone (a hard problem). However, the prevailing direction in the tech industry is to prioritize user security and handle lawful access via endpoint-based approaches if at all (rather than backdoors in protocols).

In conclusion, VoIP encryption protocols have matured significantly and are now an integral part of deploying any voice or video service. The combination of protocols like SRTP, ZRTP, DTLS-SRTP, SIP-TLS, and S/MIME (the latter in niche cases) provides a toolkit that can be tailored to specific needs and threat models. When implemented with best practices, they protect voice communications at a level comparable to other secure data services. As a technical professional, it's important to stay updated on these protocols, ensure that implementations are configured correctly (e.g., verifying certificate chains, using strong ciphers, enabling SAS verification in apps that support ZRTP), and keep an eye on emerging enhancements. The landscape of VoIP threats will continue to evolve, but so will the defenses. The trajectory is clear: towards ever more **secure, private, and trustworthy** internet communications, where users can confidently speak their minds knowing their voice is only heard by whom they intend.

Sources:

- Rescorla, E. *et al.*, **"The Secure Real-time Transport Protocol (SRTP)," RFC 3711**, IETF (2004) – Introduces SRTP, providing confidentiality, integrity, and replay protection for RTP (Source: datatracker.ietf.org)(Source: learn.microsoft.com).

- Zimmermann, P. *et al.*, **"ZRTP: Media Path Key Agreement for Unicast Secure RTP," RFC 6189**, IETF (2011) – Defines ZRTP protocol (Diffie-Hellman key exchange over RTP) and its properties (no PKI, SAS verification, etc.) (Source: datatracker.ietf.org)(Source: icterra.com).

- Fischl, J. *et al.*, **"Framework for Establishing an SRTP Security Context using DTLS," RFC 5763**, IETF (2010) – Describes how to use SIP/SDP with DTLS-SRTP keying, including the fingerprint mechanism (Source: datatracker.ietf.org)(Source: soufianebouchaara.com).

- McGrew, D. and Rescorla, E., **"Datagram Transport Layer Security (DTLS) Extension to Establish Keys for SRTP," RFC 5764**, IETF (2010) – Details the DTLS-SRTP handshake and that media is sent as SRTP (not in DTLS records) (Source: datatracker.ietf.org).

- Rosenberg, J. *et al.*, **"A Hitchhiker's Guide to SIP," RFC 5411**, IETF (2009) – Overview of SIP extensions, notes that DTLS-SRTP was selected as the preferred SRTP keying method and that SIP S/MIME saw very little deployment (Source: rfc-editor.org)(Source: rfc-editor.org).
- Comms Council UK, **"VoIP Encryption Briefing Paper"** (2017) – Industry paper noting SDES as widely implemented and explaining differences of SDES vs ZRTP (Source: commscouncil.uk) (Source: commscouncil.uk).
- VoIP.ms Blog, **"Understanding VoIP Encryption: SIP, TLS & SRTP"** (2019) – Emphasizes the importance of SIP-TLS and SRTP for business VoIP, ensuring signaling privacy and media encryption (Source: blog.voip.ms)(Source: blog.voip.ms).
- Coser, M. (Twilio), **"SRTP and You: A Deep Dive into Encrypted VoIP"** (2022) – Explains SRTP's use of AES encryption and HMAC-SHA1, and how it protects against eavesdropping and tampering with low overhead (Source: digitalsamba.com)(Source: digitalsamba.com).
- ICterra, **"ZRTP: The Steel Wall of VoIP Encryption"** (2021) – Describes ZRTP's operation (in-band key exchange, no PKI) and its SAS mechanism for MitM prevention (Source: icterra.com) (Source: icterra.com).
- SynchroNet, **"H.323 vs SIP: VoIP Protocol Differences"** (2023) – Notes that H.323 uses H.235 for encryption while SIP uses TLS/SRTP for strong security in modern systems (Source: synchronet.net)(Source: synchronet.net).

Tags: voip, encryption, srtp, zrtp, dtls-srtp, sip-tls, s/mime, network security, protocols, internet telephony

---

About ClearlyIP

ClearlyIP Inc. — Company Profile (June 2025)

---

1. Who they are

ClearlyIP is a privately-held unified-communications (UC) vendor headquartered in Appleton, Wisconsin, with additional offices in Canada and a globally distributed workforce. Founded in 2019 by veteran FreePBX/Asterisk contributors, the firm follows a "build-and-buy" growth strategy, combining in-house R&D with targeted acquisitions (e.g., the 2023 purchase of Voneto's EPlatform UCaaS). Its mission is to "design and develop the world's most respected VoIP brand" by delivering secure, modern, cloud-first

communications that reduce cost and boost collaboration, while its vision focuses on unlocking the full potential of open-source VoIP for organisations of every size. The leadership team collectively brings more than 300 years of telecom experience.

---

2. Product portfolio

- **Cloud Solutions** – Including *Clearly Cloud* (flagship UCaaS), **SIP Trunking**, **SendFax.to** cloud fax, **ClusterPBX OEM**, **Business Connect** managed cloud PBX, and **EPlatform** multitenant UCaaS. These provide fully hosted voice, video, chat and collaboration with 100+ features, per-seat licensing, geo-redundant PoPs, built-in call-recording and mobile/desktop apps.
- **On-Site Phone Systems** – Including CIP PBX appliances (FreePBX pre-installed), ClusterPBX Enterprise, and Business Connect (on-prem variant). These offer local survivability for compliance-sensitive sites; appliances start at 25 extensions and scale into HA clusters.
- **IP Phones & Softphones** – Including CIP SIP Desk-phone Series (CIP-25x/27x/28x), fully white-label branding kit, and *Clearly Anywhere* softphone (iOS, Android, desktop). Features zero-touch provisioning via Cloud Device Manager or FreePBX "Clearly Devices" module; Opus, HD-voice, BLF-rich colour LCDs.
- **VoIP Gateways** – Including Analog FXS/FXO models, VoIP Fail-Over Gateway, POTS Replacement (for copper sun-set), and 2-port T1/E1 digital gateway. These bridge legacy endpoints or PSTN circuits to SIP; fail-over models keep 911 active during WAN outages.
- **Emergency Alert Systems** – Including **CodeX** room-status dashboard, **Panic Button**, and **Silent Intercom**. This K-12-focused mass-notification suite integrates with CIP PBX or third-party FreePBX for Alyssa's-Law compliance.
- **Hospitality** – Including **ComXchange** PBX plus PMS integrations, hardware & software assurance plans. Replaces aging Mitel/NEC hotel PBXs; supports guest-room phones, 911 localisation, check-in/out APIs.
- **Device & System Management** – Including **Cloud Device Manager** and **Update Control (Mirror)**. Provides multi-vendor auto-provisioning, firmware management, and secure FreePBX mirror updates.
- **XCast Suite** – Including Hosted PBX, SIP trunking, carrier/call-centre solutions, SOHO plans, and XCL mobile app. Delivers value-oriented, high-volume VoIP from ClearlyIP's carrier network.

---

3. Services

- **Telecom Consulting & Custom Development** – FreePBX/Asterisk architecture reviews, mergers & acquisitions diligence, bespoke application builds and Tier-3 support.
- **Regulatory Compliance** – E911 planning plus **Kari's Law**, **Ray Baum's Act** and **Alyssa's Law** solutions; automated dispatchable location tagging.

- **STIR/SHAKEN Certificate Management** – Signing services for Originating Service Providers, helping customers combat robocalling and maintain full attestation.
- **Attestation Lookup Tool** – Free web utility to identify a telephone number's service-provider code and SHAKEN attestation rating.
- **FreePBX® Training** – Three-day administrator boot camps (remote or on-site) covering installation, security hardening and troubleshooting.
- **Partner & OEM Programs** – Wholesale SIP trunk bundles, white-label device programs, and ClusterPBX OEM licensing.

---

4. Executive management (June 2025)

- **CEO & Co-Founder: Tony Lewis** – Former CEO of Schmooze Com (FreePBX sponsor); drives vision, acquisitions and channel network.
- **CFO & Co-Founder: Luke Duquaine** – Ex-Sangoma software engineer; oversees finance, international operations and supply-chain.
- **CTO & Co-Founder: Bryan Walters** – Long-time Asterisk contributor; leads product security and cloud architecture.
- **Chief Revenue Officer: Preston McNair** – 25+ years in channel development at Sangoma & Hargray; owns sales, marketing and partner success.
- **Chief Hospitality Strategist: Doug Schwartz** – Former 360 Networks CEO; guides hotel vertical strategy and PMS integrations.
- **Chief Business Development Officer: Bob Webb** – 30+ years telco experience (Nsight/Cellcom); cultivates ILEC/CLEC alliances for Clearly Cloud.
- **Chief Product Officer: Corey McFadden** – Founder of Voneto; architect of EPlatform UCaaS, now shapes ClearlyIP product roadmap.
- **VP Support Services: Lorne Gaetz** (appointed Jul 2024) – Former Sangoma FreePBX lead; builds 24×7 global support organisation.
- **VP Channel Sales: Tracy Liu** (appointed Jun 2024) – Channel-program veteran; expands MSP/VAR ecosystem worldwide.

---

5. Differentiators

- **Open-Source DNA:** Deep roots in the FreePBX/Asterisk community allow rapid feature releases and robust interoperability.
- **White-Label Flexibility:** Brandable phones and ClusterPBX OEM let carriers and MSPs present a fully bespoke UCaaS stack.

- **End-to-End Stack:** From hardware endpoints to cloud, gateways and compliance services, ClearlyIP owns every layer, simplifying procurement and support.
- **Education & Safety Focus:** Panic Button, CodeX and e911 tool-sets position the firm strongly in K-12 and public-sector markets.

In summary

ClearlyIP delivers a comprehensive, modular UC ecosystem—cloud, on-prem and hybrid—backed by a management team with decades of open-source telephony pedigree. Its blend of carrier-grade infrastructure, white-label flexibility and vertical-specific solutions (hospitality, education, emergency-compliance) makes it a compelling option for ITSPs, MSPs and multi-site enterprises seeking modern, secure and cost-effective communications.
