IRONCORE LABS

The End-to-End Encryption Guide for Video Conferencing Platforms

An in-depth, technical look at how to make E2EE a reality for your video conferencing platform.

TABLE OF CONTENTS

THE RISE OF VIDEO CONFERENCING IN 2020

Since the start of the pandemic, the number of companies and employees using video conferencing solutions has skyrocketed as large portions of the population work from home.

"Before COVID-19 containment measures went into effect, less than 11% of respondents worked from home. As of June 4, that percentage had expanded to more than 45%." *— IBM Institute for Business Value (IBV)*

Work from home is here to stay. The IBM study found that 81% of respondents would like to continue working remotely at least some of the time, a stat that video conferencing platforms like Zoom are banking on to continue driving demand after the pandemic ends. Google, Facebook and Twitter are among the giants leading the way in this new trend.

Zoom, however, learned early on that popularity brings scrutiny. Researchers found numerous security flaws and questionable features that hurt privacy. And the world watched as security experts called out Zoom for falsely claiming their meetings were end-to-end encrypted when they definitely weren't. Businesses and governments reacted swiftly and dropped Zoom, forbidding employees from using it.

Security conscious customers want zero-trust solutions where their meeting audio and video can only be observed by legitimate, invited participants. This has been doubly important for Zoom who has many U.S. customers, but has a bulk of their Engineering and operations in China. Many of these companies worry about industrial espionage from China and demanded an eavesdrop-proof solution from Zoom.

The issue hasn't stopped with Zoom. In fact, although much of the original scrutiny was aimed at them, the same requirements are being applied to all vendors. More than ever, enterprise companies and government organizations are giving their business to platforms that prioritize security and data privacy.

Those who offer zero-trust end-to-end encryption will win big over video conferencing platforms that don't. The entire industry is now reinventing itself as businesses increasingly rely on virtual meetings through the pandemic and beyond.

1. THE BASICS: WHAT, WHY, AND HOW

What is End-to-end Encryption?

Simply put, end-to-end encryption means the service provider can't see the data or communications that they store or that flows through them. Data is encrypted at the source and decrypted at the destination. And importantly, the parties between the source and the destination never have access to the key(s) used to encrypt the data.

This technology is used to preserve privacy and to reduce the trust required in the service providers. We use the plural here because the video conferencing provider may well use servers from someone like Amazon and third-party transcription services.

End-to-end encryption, if implemented correctly, is a zero-trust security model. In the context of video conferencing, the meeting host knows exactly who has access to the meeting and can selectively add or remove people or service providers. No one else can eavesdrop.

The video conferencing provider still has some degree of trust: they will likely know who is participating in the meeting, how long it lasted, and other metadata like what devices were used by participants. They will also be trusted to ensure the availability of the meeting to invited participants. But the contents of the meeting are completely confidential including the audio, video, screen sharing, chat, and any other assets like shared files.

Why End-to-end Encryption Matters

Video conferencing platforms turn to end-to-end encryption to keep conferences private, secure, and resistant to tampering by malicious parties who may be:

- *Insiders such as server administrators.*
- *Hackers who may gain access to your systems.*
- *Meeting participants who are legitimate but malicious, as well as imposters.*
- *Outsiders of any kind including those in a position to manipulate network traffic.*

Meeting participants invited by or approved by the host should be the only users able to see or hear meeting content but they should be unable to subvert the cryptographic system. Also, participants who leave or are kicked out should not be able (cryptographically) to see or hear the rest of the meeting even if they get access to the encrypted streams of data.

Concerns about End-to-end Encryption

Some politicians worry that strong security will make the job of law enforcement and intelligence agencies nearly impossible. The concern is that child predators, terrorists, and criminals will benefit from this technology. The trade-off is between the risks to the security and privacy of the 99% of meetings that are legitimate vs. the minority of meetings that may not be.

These concerns are often over-dramatized, though. End-to-end encryption is not perfect. Law enforcement will have a harder time with broad surveillance but can still conduct pointed surveillance with narrow and specific warrants by targeting specific users or organizations.

At the end of the day, most of this technology still relies on an identity provider, which is likely either the video conferencing provider or one of their Enterprise customers using a Single Sign-on server. Law enforcement retains legitimate mechanisms for gaining access to specific calls or to the data of specific users even if those mechanisms are narrower and are sometimes less convenient.

For video conferencing providers who are still worried about offending their Government, we suggest offering this feature just to Enterprise businesses and also providing private reporting mechanisms for flagging inappropriate content in meetings, which would be securely shared with an abuse team who could then bring in the authorities, if warranted.

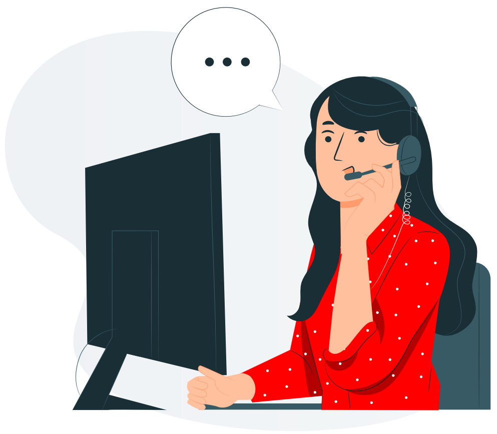
Why IronCore Labs Makes Sense

End-to-end encryption is notoriously difficult to get right. Management and distribution of the encryption keys is challenging. Companies need to be prepared to hire teams of cryptography experts and to invest huge amounts of money and time into those teams and their efforts, or they need to find a partner who has already done this.

IronCore offers a zero-trust end-to-end encryption system that manages all of the hard parts of E2EE for video conferencing platforms, including management of data, device, user, and group keys. And we never have access to private keys or plain-text data.

We bring advanced encryption techniques, such as proxy re-encryption, to simplify otherwise difficult tasks like device management and to enable conferences at large scale with dynamic groups of users.

IronCore Labs is a data control and privacy platform that makes it easy for software developers to add end-to-end encryption capabilities into their applications and we'd love to help you.

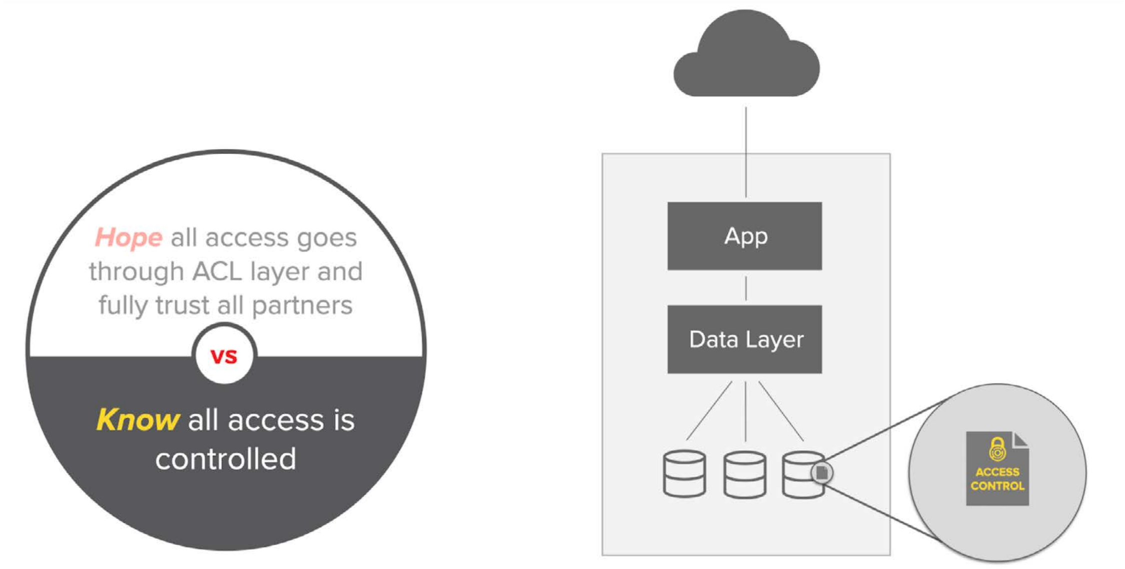


2. HOW IT WORKS

The IronCore Data Control Platform (DCP)

IronCore Labs' Data Control Platform (DCP) enables zero-trust and minimal-trust architectures with cryptography-based access control. This means that data can only be decrypted by permissioned users, and decryption generally happens at the point-of-use (in client device or authorized server) or as near to that point as is feasible. The access controls are provable and travel with the data. The platform fully supports end-to-end use cases.

The DCP platform is unique in that it uses Cryptographic Orthogonal Access Control [1], which allows for public key cryptography to work with large and dynamic groups.


Hope all access goes through ACL layer and fully trust all partners
VS
Know all access is controlled
App
Data Layer
ACCESS CONTROL

The platform works out-of-the-box for stored data, but using some adoption patterns, is well suited for video conferences and other ephemeral communications use cases, as we detail below.

SDK Platform and Language Support

The DCP cryptography layer, recrypt [6], is written in Rust and can be compiled for various architectures including Intel 32/64, ARM 32/64, and WebAssembly. The cryptography is all constant-time, meaning it doesn't branch or loop on any secrets and should be very resistant to side-channel and timing attacks.

At IronCore Labs, we wrap the cryptography layer with a key management layer called IronOxide [7]. For most runtime environments, this is written in Rust and wrapped or exposed from there to different languages. The browser has its own implementation called IronWeb [8], and NodeJS has its own implementation called IronNode [9].

We have native support available for Android and iOS, and all common desktop and server operating systems, plus all major and current browsers.

We also have language bindings for Rust, Java, Scala, C++, JavaScript, TypeScript, and Swift. We also have a C interface layer if required, but it is only available upon demand as we don't publish it for general use.

Other languages and platforms could be supported if necessary.

E2EE Video Conferencing With IronCore Labs

Our SDKs use AES256-GCM with randomly generated DEKs (Document Encryption Keys) to encrypt files, fields, rows, big data partitions, and arbitrary streams of data. The term "DEK" is common in the field, but does not indicate it can only be used for documents. The DEK can be used directly or keys can be derived for it. In our video conferencing patterns, we use the DEK to protect a master session key which is the root for the derived keys used to protect audio and video streams.

IronCore uses an envelope encryption [2] pattern and only approved users can unwrap the master session key using their own private key. We'll explain this more later.

Additional Advantages of Using IronCore

In addition to video and audio content, video conferencing platforms may wish to end-to-end encrypt other assets and by-products of meetings or events.

In choosing IronCore, this functionality will come along for free and will allow you to encrypt and store things like:

- *Shared files*
- *In-meeting chat*
- *Chat transcripts*
- *Call recordings*
- *Abuse reports (encrypted to your safety team)*

IronCore's approach also scales horizontally and performance is not impacted by the number of users in a call. This means that end-to-end encryption can perform well even in 50,000 user events.

3. UNDER THE HOOD

Before we get into the protocols around meeting management, it may be helpful to understand the building blocks that we use. The key takeaway is that we use transform cryptography to enable public key encryption to groups where you encrypt something a single time to a single public key and then any member of the group can decrypt that ciphertext using their own unique private key. That should be enough information to skip over this entire section if you don't want to go deep. This section is the technical underpinnings of the protocols we'll discuss later on in this guide.

Transform Cryptography

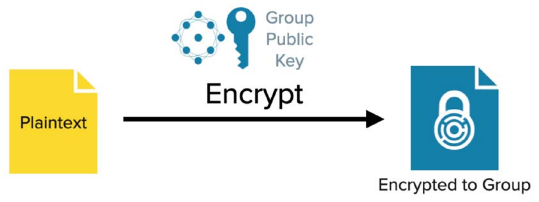
IronCore's DCP leverages elliptic curve cryptography and unidirectional, multi-hop proxy re-encryption, known as transform cryptography [3]. This has been an active area of research for 20 years with thousands of published papers, but IronCore is the only commercial and open-source implementation of proxy re-encryption with these properties.

$$C_a \rightarrow \boxed{\text{proxy}} \rightarrow C_b$$

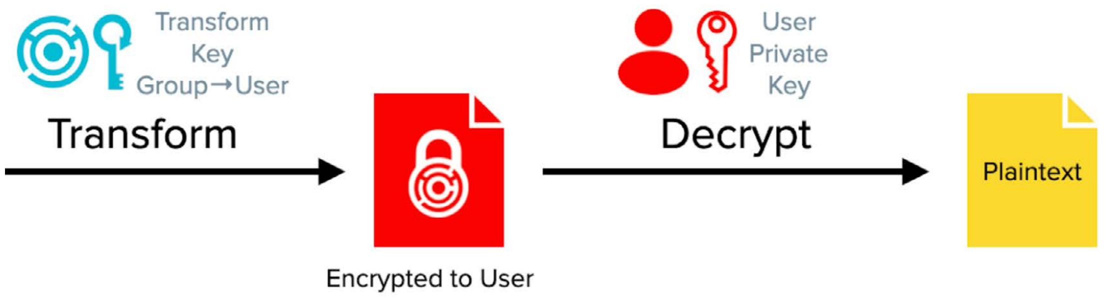
In a nutshell, transform cryptography allows the delegation of decryption capabilities from one key to another with the help of a server that is able to transform the ciphertext so that it goes from being encrypted, for example, to Alice's public key, to being encrypted to Bob's public key without the server decrypting or learning anything about the data or any private keys. The server can only transform the ciphertext to Bob after Alice has explicitly allowed it by sending the server a "transform key."

IronCore's DCP uses the concept of users and groups. The DEK for encrypted data is either encrypted to a group's elliptic curve public key or to a user's public key.

To allow a user to access data that was encrypted to a group, that user must be made a member of the group. Adding a user to a group involves creating a transform key from delegator to delagatee (that is, from group to member). This operation requires the private key of the delegator (group) and the public key of the delegatee (member). Users that are designated as administrators of the group have access to the group's private key, which allows them to create the transform keys required to add a user to a group.

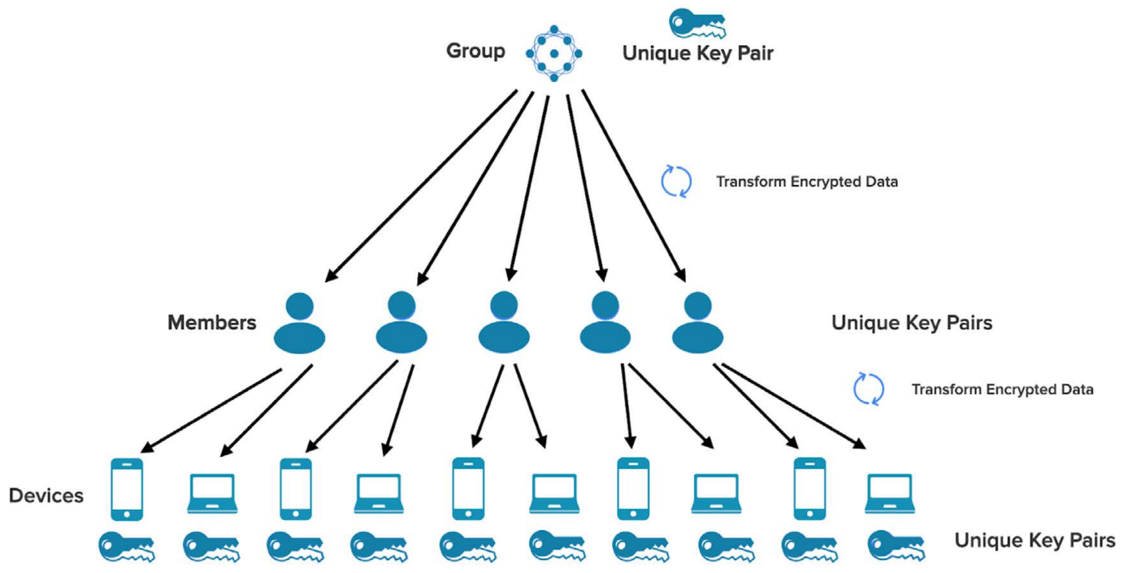


To clarify, the data that is being encrypted here using transform cryptography is a DEK. The actual video, audio, or text data is encrypted using AES-GCM with a randomly generated DEK, then the DEK is encrypted to the group. Any member of the group can then decrypt the data shared with the group using their own private key, but only after the encrypted DEK has been transformed for them by the IronCore service. That service never sees the encrypted video, audio, or text data. Only the encrypted DEK goes to the service.



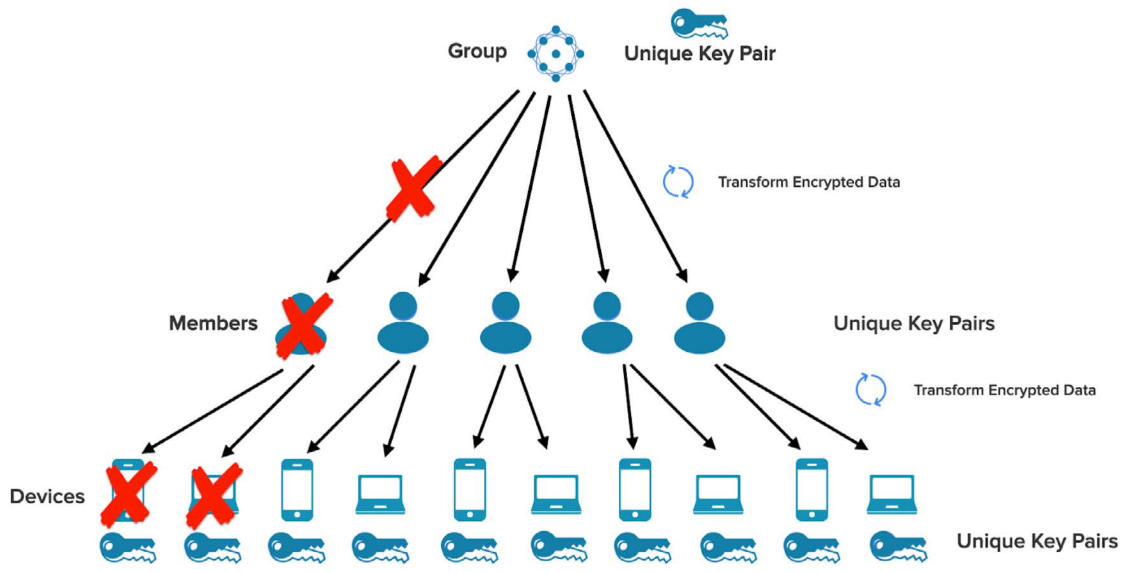
IronCore takes this paradigm of groups and members and extends it by modeling the user as a group of devices. Each device can decrypt anything that is encrypted to the user, and by extension, anything that is encrypted to a group of which the user is a member.

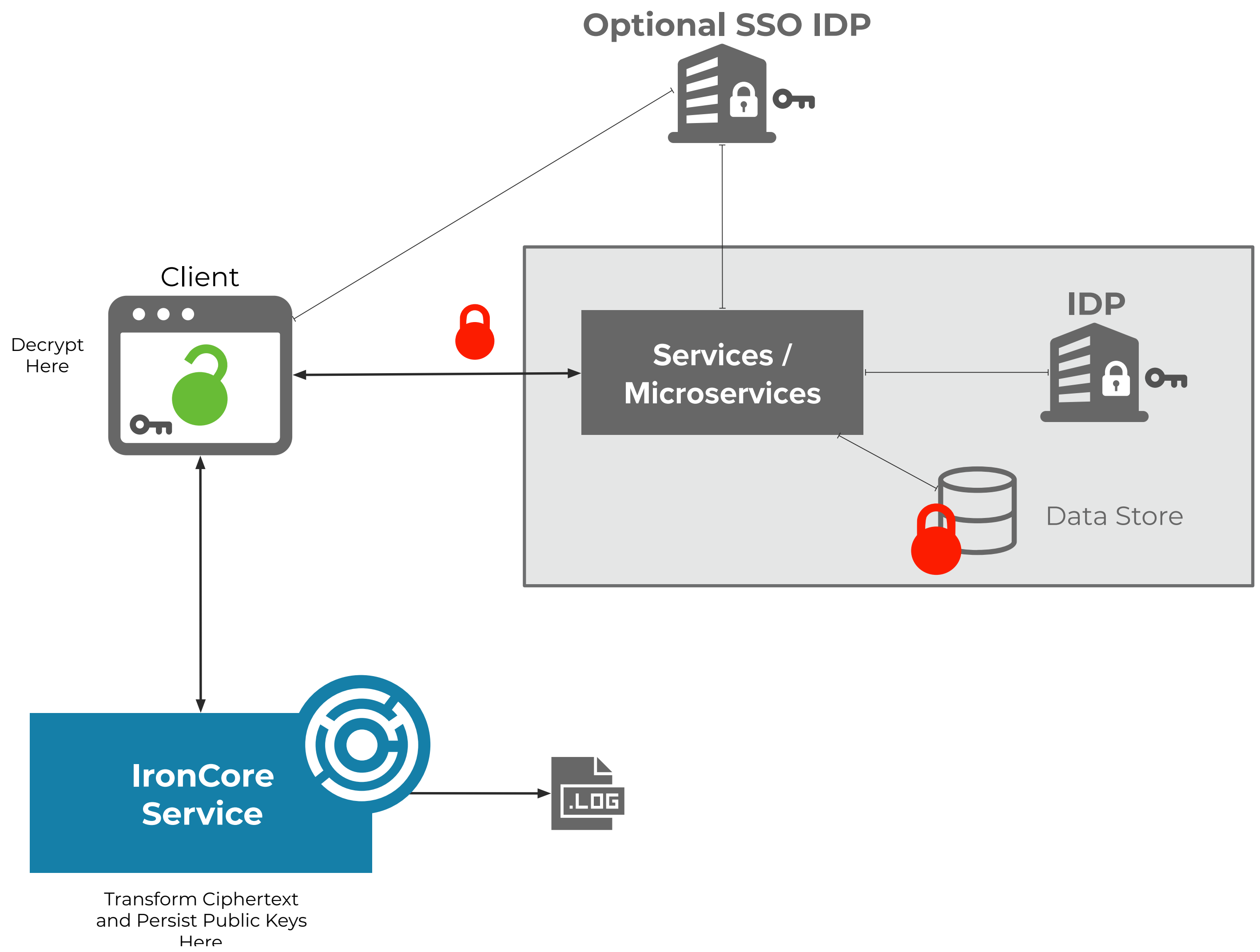
In this system, only user-approved devices can actually decrypt anything (see the paper [1] for the multi-party computation details that enforce this restriction).



The implication of this is that any device can be revoked at any time and for any reason, such as if a user loses it. And if a member is removed from a group, then none of their devices can decrypt data encrypted to that group.

The other implication here is that the IronCore service knows whenever a DEK is unwrapped since the service has to be invoked to transform the encrypted DEK. This lets us generate a non-bypassable audit trail for all data access.





System Components

- *IDP: produces identity assertions. May be operated by you the video conferencing provider or externally by an enterprise customer's SSO server.*
- *IronCore: transforms ciphertext for unwrap (needed for decrypt) and manages public keys, identity assertions, and audit trails.*
- *Video conferencing service: handles routing of streams and a comms channel for encrypted messages.*
- *Video conferencing client: all encrypt/decrypt operations.*

Key Management Protocols

OVERVIEW OF KEYS

There are a number of different keys in use in the system. Ultimately each of these comes down to one of four types.

Public/private encryption key pair:

- *These are straight elliptic curve key pairs on a Barreto-Naehrig curve [4].*
- *The private key is a large random number generated on the client device. In no case does the private key leave the device unencrypted. In no case does the device private key leave the device in any form.*
- *Group and user keys*
    - *The group and user keys can be used to generate transform keys (delegate decryption) but this is all they can do.*
- *Device keys*
- *The device key can be used to unwrap a DEK.*

Public/private signing key pair:

- *Only devices have signing keys.*
- *These use the Ed25519 algorithm [5].*
- *These are used to authenticate all API requests to IronCore and to add tamper evidence to audit logs.*
- *These are also used to sign over encrypted DEKs for non-repudiation.*

Document encryption key (DEK):

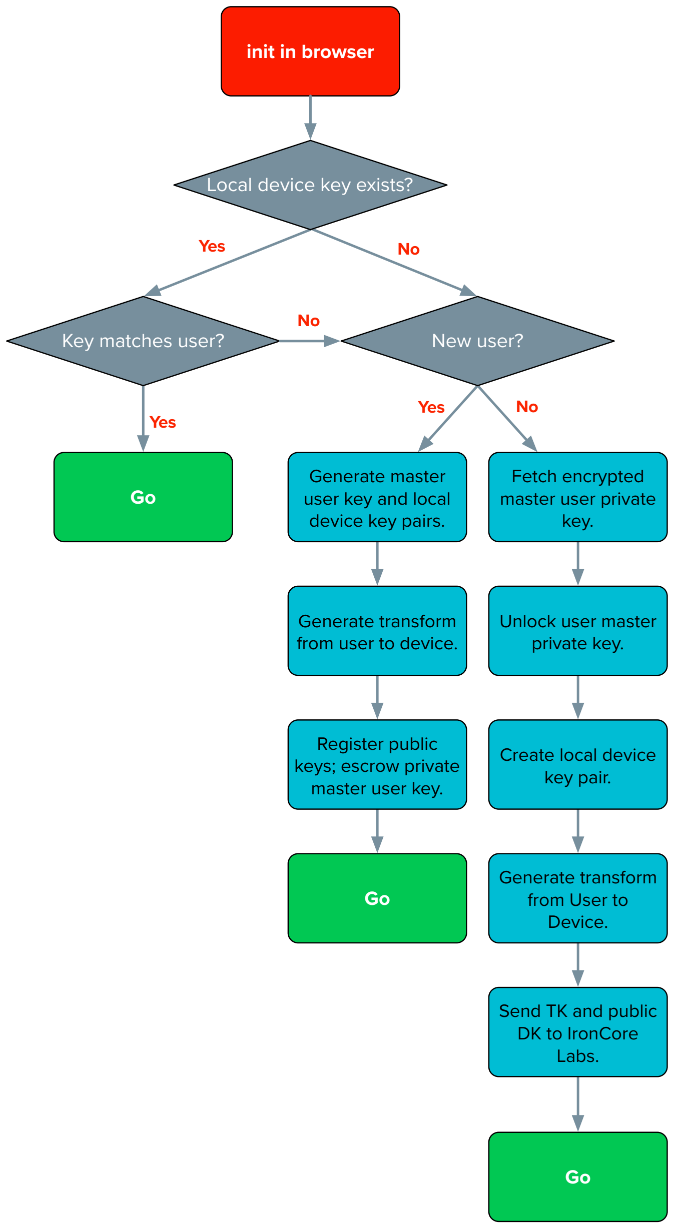
- *Used for fields, files, rows, columns, big data partitions, messages, nested keys, or streams of audio/video.*
- *AES256-GCM*
- *This is used to actually encrypt/decrypt data.*
- *The DEK is encrypted to a public key and becomes an Encrypted DEK (EDEK).*
- *Generated at the point of encryption and never transmitted in DEK form (only EDEK form).*

Transform key:

- *Sometimes called a re-encryption key.*
- *This is held by the IronCore service (the proxy re-encryption proxy) sometimes called the transform service.*
- *There is one transform key for every delegation of access from one key pair to another (generally from a group to a user or a user to a device).*

All keys are generated on the client by the current user. When a user creates a group, they generate a key pair for the group and encrypt the private key to themselves and optionally to other users who they want to have admin privileges for that group.

The user's private key is encrypted using AES256-GCM with a key derived from a secret such as a user password. More details on this later along with alternatives to the user having a second password. This encrypted private key is escrowed with the IronCore service.


init in browser
Local device key exists?
Yes
No
Key matches user?
No
New user?
Yes
Go
Yes
No
Generate master user key and local device key pairs.
Fetch encrypted master user private key.
Generate transform from user to device.
Unlock user master private key.
Register public keys; escrow private master user key.
Create local device key pair.
Go
Generate transform from User to Device.
Send TK and public DK to IronCore Labs.
Go

User public keys are linked to identities by signing over a JWT assertion from an IDP using both the encryption keys (used in this one case for a signature) and the device signing key. This creates a provable link between the public key and a user (assuming we trust the IDP).

Video conferencing platforms and IronCore are zero-trust in regards to confidentiality. The owner of the IDP (which is the video platform or one of its enterprise customers) is trusted to verify and assert the identity of the current user. IronCore is trusted to delete transform keys in revocation scenarios.

INITIALIZATION

The initialization logic works similarly in all of our SDKs, but the example below breaks down the browser initialization.

Calling [IronWeb.initialize()]() is done on load of a page and must be done before any data can be encrypted or decrypted. The initialize call takes two parameters: a JWT callback and a "password" callback. These callbacks are only used if this is a new device.

The callback function is a promise that allows the SDK to asynchronously fetch a JWT, which must be valid for the initialization to be successful.

Once initialized, any user can query their group memberships, create groups, administrate those groups, and encrypt to users and groups they know about, or decrypt anything encrypted to them or to a group of which they are a member.

The JWT is used as proof of identity, but will be signed by a certificate for this purpose only so the JWT cannot be reused for other purposes.

AUTHORIZING NEW DEVICES: BASIC APPROACH

Authorizing a new device means creating a transform key from a master user key to a new device key. If this is a new user, they will generate the master user key and then need a secret to encrypt it before escrowing with IronCore. If this is an existing user, they will need a secret to decrypt the previously created master user key.

In order to give users a seamless and transparent experience, we recommend adopting a split-trust model for storing the secret. In this model, the password callback function actually hits two endpoints: one run by IronCore and one run by the video conferencing platform or one of its customers.

In both cases, a valid, signed JWT with a user ID is sent to each service and a stored random 128-bit number is returned from each service. These two partial secrets meet in the client and are combined with a secure hashing function to produce the “password.” The password is then hashed many more times as part of a PBKDF2 process before AES256-GCM is used to encrypt or decrypt the user master private key.

In this way, every one of your users that can log in will have a cryptographic identity. This can be utilized when sharing session keys (as explained below) or when sharing stored data such as chat transcripts or meeting recordings.

As a future phase enhancement, video conferencing platforms could choose to use IronCore’s SaaS Shield (customer managed keys) solution to protect their portion of the secret. Effectively this would make security conscious customers the root of trust for the creation of new devices and users.

Note that we also have alternative flows for advanced users who wish to more tightly control their keys, as explained below.

AUTHORIZING NEW DEVICES: ADVANCED USERS

The video conferencing platform can choose to give users the opportunity of taking control of their own user master key security. Users who do this will have two options:

1 **Protect the user master key with a password.**

- *If the user forgets the password, they will not be able to authorize new devices and could lose access to data that is encrypted directly to their user key (vs. to a group key).*
- *Even if the key is forgotten, existing devices will have valid device keys and can still work.*
- *Someone who can successfully log in to their account or who can compromise the IDP and impersonate them will still not be able to view sensitive data or participate in E2E meetings without this second password.*

2 **Require an existing device to approve a new device.**

- *Any existing device can decrypt the user master password and can therefore either selectively share that key with a new device, for example via a QR code, or can receive a request to approve a new device via push notification and can then generate the needed transform key to enable that device.*
- *So an advanced user can use their mobile device to approve new devices and in this way, they will know if someone logs into their account or tries to impersonate them.*

MEETING START FLOW

We assume that anyone can start a meeting and the host can join late or never. If only the host can start the meeting, the process below is still valid. We further assume that initialization happens before we get to this point.

- *The first user, U_0, joins the conference.*
- *U_0 generates a random 256-bit number, which will be the first session key, SK_0.*
- *The user creates a group for this specific meeting. The group can use any identifier, but the meeting ID probably makes the most sense.*
- *If U_0 is not the host, then U_0 makes Host an admin right away if the Host already has a cryptographic identity (meaning they've logged in and initialized IronCore at least once in the past, which should always be the case if that happens when creating new meetings). If Host does not have a cryptographic identity held by IronCore, then U_0 will wait and add Host as an admin of the group after Host joins.*
- *U_0 proactively adds anyone it knows is invited to the meeting and who already have existing cryptographic identities as members of the group.*
- *U_0 encrypts the session key, S_0, and the session key sequence number, 0 to the meeting group and posts the signed and encrypted ciphertext to the meeting's communications channel.*

If the host joins late, they follow the meeting join flow and then take over management of the meeting group and start from step 2, but increment the session key sequence, rotating the session key in use.

MEETING JOIN FLOW

When a participant joins, having first initialized the IronCore SDK, they will take the following steps:

- *They will announce themselves on the message channel.*
- *If the first user or host sees them and has not already added them to the meeting group, they do that after the announcement.*
- *The joining user then looks for the most recent encrypted session key and attempts to decrypt it. Because of a possible race condition with being added to the group, the user will retry up to 4 times waiting first 1 second, then 5 seconds, then 10 seconds, then 10 more seconds to attempt to decrypt. If decryption still fails, the participant has not been granted permission to view the meeting content.*

MEETING LEAVE/REKEY FLOW

Whenever a participant leaves, and possibly on some fixed time schedule as well, a new session key is generated. This will prevent the leaving user from being able to decrypt any new content in the meeting even if they still had access to the encrypted stream of data. Here are the steps taken by the host or first user when a participant leaves:

- *Remove that participant from the meeting group*
- *Debounce by waiting 10 seconds for any other participants to leave so as not to thrash session keys when many participants are leaving.*
- *Generate a new session key, increment the sequence*
- *Encrypt the session key to the group*
- *Post to the board the ciphertext holding the new session key*

MEETING END

When the meeting ends, U_0 or the Host deletes the ephemeral group and all users drop their held session keys and any ciphertext.

LOST DEVICE FLOW

A user who can log in on any device and successfully initialize the IronCore SDK (done automatically behind the scenes for them) can revoke old, disused, or lost devices. You will need to add a UI to support listing the devices returned from IronCore and calling the appropriate SDK function to revoke a device.

RATCHETING

Unlike the chat setting or one-to-one meetings, in a meeting where multiple people can talk at the same time, the concept of ratcheting keys needs to be rethought. We believe that for a large group video conference, the best approach is to set a regular schedule for rekeying the session key (and re-deriving per-stream keys). We recommend doing this on a timer that resets every time a rekey event happens (for example, after a participant leaves). So if there hasn't been a rekey in the previous five minutes, then the initial user or host should ratchet the keys in use by issuing a rekey event.

Forward Secrecy/Protecting Against Compromise of Previous Meetings

As long as the meeting group is deleted at the end of the meeting (or within some automatic time after it), there is no way for a compromised user or client to decrypt the data encrypted to that group since the server will not be able to transform the ciphertext.

Interception of transform keys being uploaded by the Host also doesn't compromise the meeting or allow a third party to transform due to a server-side secret that is also needed for a successful transformation.

However, there remains an issue where an attacker that is able to capture all network traffic, including the already transformed ciphertexts that hold the session keys, can then later compromise the device key for a device that was a participant in the meeting and use that key to decrypt the previous meeting's contents.

There are several ways to combat this, but the simplest is to rotate device keys after every meeting. The client deletes (revokes) the previous device key and registers a new one. Any future compromise of the device will not compromise a previous meeting.

4. ADVANTAGES OVER COMMON ALTERNATIVES

The most common alternative to IronCore's approach is to use a **pairwise negotiation protocol at the device-to-device level** where the host negotiates a per-participant secret key with each participant's current device and then sends the meeting session key to each participant encrypted with the negotiated secret. When the session key rotates (because the participant list changes, for example), the host independently encrypts and sends the new session key to each participant again.

This means the host has to know about all of the devices of all of the participants and has to be able to validate users at the device level. It also means there is a potentially heavy-weight operation that the host has to do on a regular basis, which could be crippling when there are a large number of users and/or the host is on a low-power device such as a mobile phone.

IronCore's Proxy Re-encryption-based approach handles all of the hard parts in typical end-to-end systems, but it particularly stands out from the crowd when it comes to large or dynamic groups and things like key rotation, revocation, and secure data sharing.

1. *Non-interactive, durable when needed: the PRE system allows for data to be encrypted even when the other party is not online. This means it can be used to end-to-end encrypt all meeting-related assets like meeting titles, agendas, cloud recordings (encrypted to self once and available on all devices or shared with a team), meeting summaries, action items, chat transcripts, etc. The durable encryption uses the same mechanism but encrypts to different, non-ephemeral groups or users.*
    - *Vs The Alternative: Doing something similar with the alternatives will likely mean encrypting to the current devices of each party. Sharing with 50 users might mean encrypting to 300 devices. A new device won't be able to decrypt the data and there's no way to revoke a previous device from seeing it. Plus you have to store all of that. Basically, this approach doesn't allow secure handling of meeting artifiacts like recordings, agendas, transcripts, etc.*

2. *Device management: the PRE system does not need revocation lists and other awful mechanisms for helping clients stay up-to-date with public keys as is required with common alternatives. Per-user device management (authorizing new devices and revoking lost, outdated, or stolen ones) is simple and is abstracted away from other users.*

3. *Key management: the PRE system allows easy and elegant rotation and revocation of clients and users.*

4. *Key pinning and verification: users need only locally cache ("pin") the public keys of users, not the keys of those users' devices. This makes it more likely the key will be local and that public key attacks will be detected. With the pairwise approach, everything operates at the device level and a mismatch in a cache entry or a new entry cannot be reliably used to warn of an impersonation attack.*

5. *Scalability: The pairwise approach may work fine for 10's of users, but likely degrades the meeting experience with 100's of users and will fail completely with 1000's of users. If the Host's device is weak (like a mobile phone), it will likely burn CPU and battery and possibly even periodically brick the Host's device during rekey events. With PRE, the cryptographic burden can be largely shifted to servers that remain zero-trust with respect to confidentiality and that can horizontally scale infinitely.*

6. *Batch and pre-fetch: with PRE, public keys can be pre-fetched if the Attendee knows who the Host is or if the Host knows who the Attendees will be. This could happen also for the pairwise approach, but you'd have to fetch keys for all devices for each participant and maintain those keys against revocation lists. With PRE, the host could also choose to pre-approve certain attendees before they join by adding them to the ephemeral group without them yet being in the meeting. This will shorten the setup time on initial meeting join.*

7. *Ratchet event duration: the ratcheting approach is essentially the same for both options, but the work done by the Host at each ratchet is much less with PRE and not dependent on the number of Attendees. In the pairwise approach, the Host must separately encrypt and communicate with each Attendee, which will stretch out the time it takes for the ratchet to complete and for all parties to be able to use the new session key. This means the old key will need to be used for longer and potentially some attendees won't have the new key after some attendees start using it.*

8. *Advanced security mode: you can choose to give power users the option of taking control of their cryptographic identity so that no new devices can be added without their approval (meaning they use an existing device to create a transform key for a new device). This avoids the problem of the server just adding devices purporting to be a user, which is a weakness with other approaches.*

9 *Browser support: IronCore supports web browsers out of the box and has native support for all common mobile and desktop operating systems as well. When WebRTC Insertable Streams are widely deployed, end-to-end encryption can extend into the browser.*

5. CRYPTOGRAPHY VALIDATION

Transparent

IronCore's algorithms are based on published and peer-reviewed algorithms [1] and are backed by a full PRE-IND-CCA2 security proof. The implementations are open sourced and available on Github.

Audited

IronCore's underlying cryptography has been audited for correctness and for constant-time processing properties (to avoid side-channel attacks) by the experts at NCC Group.

Compliance

IronCore is SOC2 Type 2 certified, GDPR compliant, and Privacy Shield certified.

6. CONCLUSION

IronCore has invested deeply in building out hardened, tested, scalable, enterprise quality end-to-end encryption, allowing video conferencing platforms to significantly accelerate plans for advanced security and data protection.

What we've outlined above is just one way E2EE could work for your platform. There are many variations and customizations IronCore Labs is equipped to tackle, and we look forward to learning more about your requirements and existing systems so we can help you craft the solution you need.

We hope to become your partner and to work through designs, questions, and implementation with you.

Let's talk!

BIBLIOGRAPHY

[1] Bob Wall, Patrick Walsh. Cryptographically Enforced Orthogonal Access Control at Scale. ACM SCC '18: Proceedings of the 6th International Workshop on Security in Cloud Computing.

[2] Envelope encryption. https://ironcorelabs.com/docs/concepts/envelope-encryption/.

[3] Transform cryptography. Short video or high level description.

[4] P. Barreto, M. Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. Proceedings Lecture notes in computer sciences; 3897 in Selected Areas in Cryptgraphy -- SAC2005, 2006.

[5] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, Bo-Yin Yang. High-speed high-security signatures. Journal of Cryptographic Engineering 2 (2012), 77–89. https://cr.yp.to/papers.html#ed25519.

[6] recrypt. https://github.com/ironcorelabs/recrypt-rs.

[7] IronOxide. https://github.com/ironcorelabs/ironoxide.

[8] IronWeb. https://github.com/ironcorelabs/ironweb.

[9] IronNode. https://github.com/ironcorelabs/ironnode.

Graphics attribution: Thanks to FreePik for work vectors created by Stories.

ABOUT IRONCORE

We are a data privacy platform for application layer encryption and customer managed keys (CMK). We enable software developers and businesses to rapidly build enterprise applications with strong data control. Data owners decide who can access their data, monitor how it's used, when, where, and by whom, and can revoke that access at any time. We are the fastest and easiest way to control data in multi-cloud and SaaS environments.

IronCore Labs
1750 30th Street #500
Boulder, CO 80301, USA

Inquiries
Email: info@ironcorelabs.com
Phone: +1.415.968.9607

CONNECT WITH US

- blog.ironcorelabs.com
- linkedin.com/company/ironcore-labs
- twitter.com/ironcorelabs
- ironcorelabs.com