Verkada

Enterprise Controlled Encryption Technical Whitepaper

About the Authors

Benjamin Bercovitz

Co-Founder, Verkada

Anurag Arora

Software Engineering Manager, Verkada

Aleksandr Avseyev

Software Engineer, Verkada

Jingwen Li

Software Engineer, Verkada

Samuel Yuen

Software Engineer, Verkada

Nick Sullivan

Cryptography and Computer Security Expert
(External Consultant)

Table of Contents

Introduction

Enterprise Controlled Encryption (ECE) is a new data encryption scheme that significantly enhances Verkada's security posture by adding customer-controlled, client-side encryption with a convenient key distribution mechanism.

Client-side encryption, wherein data is encrypted in the application before transmission to cloud servers, is emerging as a valuable tool for building modern security frameworks. By encrypting data before transmission, there is a significantly lower risk of data compromise since only parties with access to the client-side encryption key can access the data. ECE leverages this approach, providing customers with enhanced control over their encryption keys and, by extension, their data.

Enterprise Controlled Encryption further addresses two problems that can make client-side encryption unsuitable for enterprise-scale deployments of IoT sensors: having remotely located headless clients and requiring contactless provisioning of customer encryption keys.

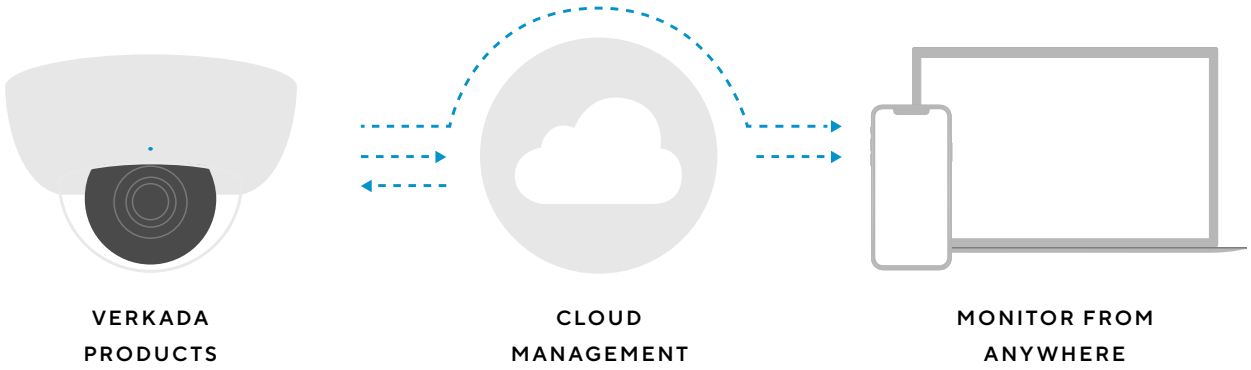
This document explains the technical underpinnings of ECE, including its key generation processes, encryption workflows, and security considerations. The subsequent sections examine the cryptographic principles employed, the system's architecture, and future goals.

Background and Security Goals

Cloud-managed video surveillance systems have brought scalability, accessibility, and cost-effectiveness to the field of physical security. These systems allow organizations to store and manage vast amounts of video data without needing extensive on-premises infrastructure, provide remote access to live and recorded footage from any internet-connected device, and offer advanced analytics capabilities previously impractical or prohibitively expensive for many users.

However, this shift to cloud-based solutions has also introduced new security challenges, particularly regarding data privacy and control. The centralization of sensitive video data reachable via cloud environments raises concerns about unauthorized access, data breaches, and the potential for abuse by service providers or malicious actors. Customers want assurances that their video surveillance data is private and can only be viewed by their authorized personnel.

Cryptography has become one of the go-to tools for software engineers when designing systems that address data privacy controls. Effective design requires enumeration of the requirements and components.



Requirements

The primary goal of Enterprise Controlled Encryption (ECE) is to provide meaningful and practical security enhancements to Verkada's product suite. Key objectives include:

1. Ensuring privacy-sensitive data is encrypted at all times, except when in use.

2. Implementing key splitting across distinct infrastructure to minimize single points of failure.

3. Limiting potential compromise windows to rare, specific events.

4. Establishing key trust on managed devices.

5. Providing user-friendly explanations and documentation for customers.

To ensure frictionless customer adoption, ECE must:

- Eliminate the need for manual per-user provisioning.
- Offer a seamless upgrade path for existing customers.
- Maintain unimpeded access for time-sensitive scenarios (e.g., law enforcement emergencies).
- Preserve the current user experience.
- Minimize major software and operational changes.
- Minimize key management operations performed by the customer.

ECE was designed to maintain the product's core value proposition while enabling future enhancements. To satisfy these twin goals, we adopted a platform-centric approach, ensuring modular software design compatible with multiple integration points. ECE's implementation balances strict security controls with practical functionality, enhancing security without compromising the product's fundamental benefits and flexibility for future innovation.

System Components

Verkada's ECE system involves several key participants and distinct computing environments, each relevant to the overall security architecture.

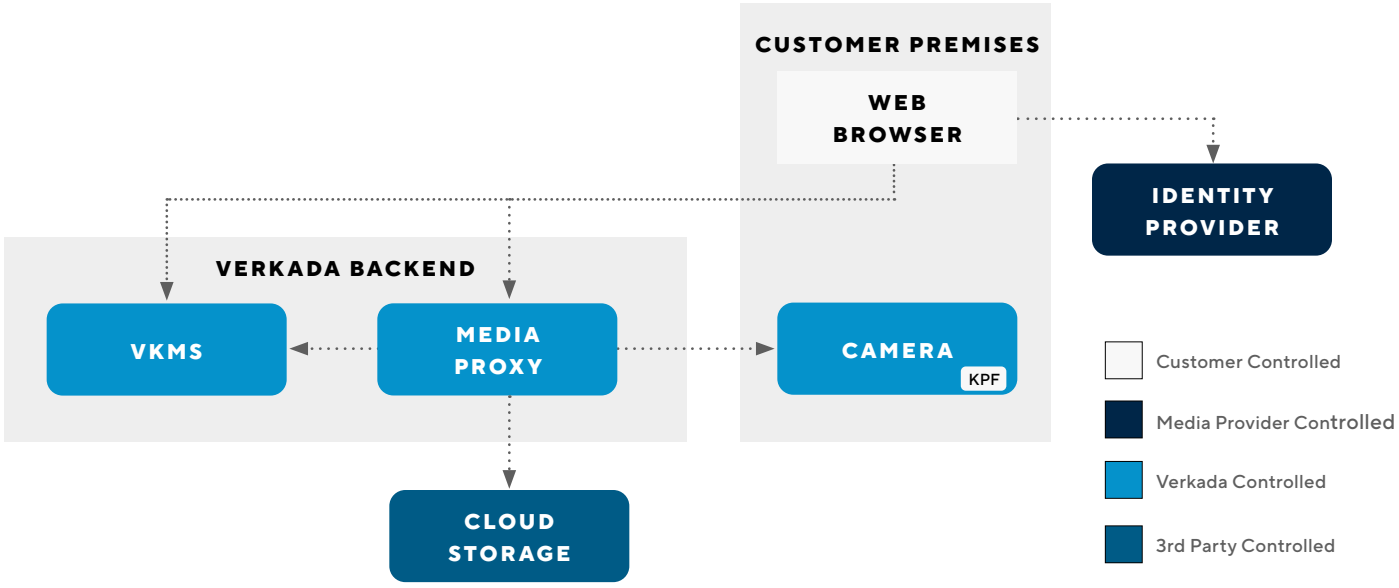


Figure 1: Overview of system components and their locations, control and connections

Participants:

- **Customer:** Organization who owns cameras and software licenses for the Verkada cloud service.
- **Identity Provider:** Software-as-a-service (SaaS) product that manages the Customer's user accounts.
- **Verkada:** Purveyor of cloud-managed physical security products, including sensors, access controls, and cameras.
- **Third Party:** An indirect vendor used by Verkada or the Identity Provider, for example Amazon Web Services.

Computing Environments:

- **Camera:** Operates at the Customer Premises and captures and encrypts video data before transmission or storage.
- **Verkada Key Management System (VKMS):** Operates in the Verkada backend systems (Verkada Backend) and oversees the encryption key hierarchy and delivery.
- **Web Browser:** Operates at the Customer Premises and is the primary interface for customers to access cameras.
- **Identity Provider (IdP):** Operated by the Customer or an Identity Provider vendor, authenticates users and assists with key management.
- **Media Proxy:** Operates in the Verkada Backend and manages secure video streaming and retrieval.
- **Cloud Storage:** Operated by third-party providers and sometimes used to store encrypted media files, augmenting local camera storage.

Cryptographic Basics

The ECE system employs a targeted set of cryptographic techniques to enhance data security. This section provides an overview of the specific encryption methods used in the system.

1. Video Encryption

(AES-128-CBC): For bulk video data encryption, we use the Advanced Encryption Standard (AES) with a 128-bit key length in Cipher Block Chaining (CBC) mode. AES-128-CBC provides a good balance of security and performance, making it suitable for encrypting large volumes of video data. [1]

2. Key Encryption (Wrapping)

a. AES-256-GCM: AES with a 256-bit key length in Galois/Counter Mode (GCM) for symmetric key encryption. The same key is used for encryption and decryption, providing both confidentiality and integrity for key material.

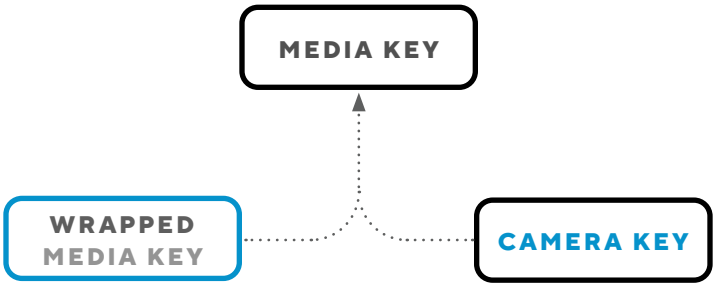
b. RSA-2048-OAEP: RSA with a 2048-bit key size for asymmetric encryption using Optimal Asymmetric Encryption Padding (OAEP). Each RSA key has two components: a public key used for encryption and a private key for decryption. The OAEP encryption mode provides for confidentiality and integrity.

c. RSA-2048-OAEP-AES-128-GCM: A hybrid scheme to asymmetrically encrypt larger key data while providing confidentiality and integrity. We use a mezzanine AES-128-GCM key for symmetric encryption of the key data and then encrypt the mezzanine key using RSA-2048-OAEP. The encrypted mezzanine key, GCM nonce, encrypted key data and authentication tag are saved together. [2]

Key Splitting Via Wrapping

Key splitting is the practice of breaking an encryption key into multiple parts, each of which is required to perform cryptographic operations. Key splitting through key wrapping is a method where a single cryptographic key is encrypted, or "wrapped," with one or more different keys to divide its control among different entities. In this process, the original key is first encrypted using one key, then the resulting ciphertext is further encrypted with another key, and so on, creating a chain of wrapped keys. This ensures that no single entity has complete access to the original key; all involved parties must provide their respective wrapping keys to decrypt and access it. This technique enhances security by requiring collaboration for key access, significantly reducing the risk of compromise by a single party.

Figure 2: An encrypted media key wrapped by a camera key is decrypted to produce the media key



We use a visual representation to represent data and keys in the context of the key wrapping hierarchy. Data fields are represented by boxes with names, colors, and borders. Boxes with a black border are not encrypted; boxes with a colored border are encrypted with the corresponding key. Decryption is represented by an upward arrow that takes the encrypted data field on the left and pairs it with the encryption key on the right. The output is presented at the top.

1. Although this cipher does not provide integrity over the ciphertext, it is nonetheless required by some common video streaming protocols. Future work could include layering in a HMAC and wherever possible modifying the player components.
2. We evaluated using HPKE (RFC 9180) but library support was still early as of the project inception. Using vetted cryptographic libraries is foundational to properly implemented encryption schemes and the support for RSA-OAEP and AES-GCM is much broader and better reviewed as of this writing.

Conventions

Throughout this document, we use the figures and symbols below to visually represent the following objects:

 **= Symmetric Key or Asymmetric Private Key**

 **= Asymmetric Public Key**

 **= Contents Within Encrypted to Key Match Color**

 **= Plaintext Media**

Security Goals

To illustrate the resilience of ECE against different compromise scenarios, we first define two security concepts.

Forward Secrecy means previous communication sessions are still secure after a breach. Each session uses a unique session key that is independent of others, preventing attackers from decrypting past data even if they obtain long-lived private keys.

Post-Compromise Recovery focuses on restoring security after a breach. If an attacker gains access to critical keys, the system can regenerate new keys and re-establish secure communication channels. This capability minimizes the impact of a compromise, allowing the system to quickly return to a secure state. This covers the protection of future communications.

We also define the following agents:

- **Users:** The end-users of the video surveillance system, including administrators and viewers.
- **Outsiders:** Any external entities, including potential attackers or unauthorized users.
- **Insiders:** Employees or contractors with privileged access to Verkada's systems.

The security goal of ECE is to limit access to sensitive data to authorized users only. Verkada's existing encryption architecture helps protect against outsiders, and ECE improves this by also helping protect data from insiders. [3]

3. Note that network data transmission between participants is protected by Transport Layer Security (TLS) regardless of the techniques described here. This measure provides a layer of security for data in transit, but it is not specifically related to the protections introduced with ECE and will not be covered in this document.

Current Video Encryption Scheme

Before discussing the specifics of Enterprise Controlled Encryption (ECE), it is necessary to understand the foundational security measures Verkada already has in place. ECE builds upon these foundational security controls.

Verkada's current architecture implements industry-standard security practices to protect data at rest and in transit. This includes robust application-layer encryption methods, secure communication protocols, and stringent access controls. While these measures provide a high level of security for most use cases, ECE is designed to address more advanced threat models while offering customers greater control over their data.

This section outlines the key components of Verkada's existing security architecture, including:

1. **Key hierarchy**
2. **Authentication and authorization**
3. **Data flows**
    a. Enrollment and key generation
    b. Video encryption
    c. Video decryption and viewing

Understanding these baseline security features provides helpful context for the additional protections and capabilities introduced by ECE, and demonstrates how they complement and enhance the overall security framework.

Key Hierarchy and Usage

In addition to the encryption at rest and in transit protections provided by Verkada, an application-layer encryption system is also implemented to safeguard video files using a hierarchical key structure. We describe below how the current system works.

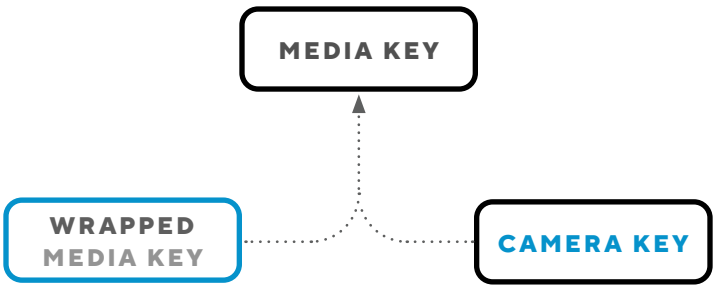


Figure 3: An encrypted media key wrapped by a camera key is decrypted to produce the media key

Media Key	Camera Key
• **Purpose:** Encrypts video data for each recording session or specific period, providing a base layer of security. It is wrapped by the Camera Key, meaning it is encrypted and can only be accessed using the Camera Key. • **Format:** AES-128 • **Usage:** HTTP Live Streaming (HLS) encryption • **Lifecycle:** Generated directly on the camera and wrapped before saving. They are rotated frequently (currently every day) to help ensure forward secrecy and post-compromise security, minimizing impact in case a Media Key is compromised. • **Security Role:** Stored media cannot be viewed without the Media Key required to decrypt it, and if one recorded video stream session or period is compromised, the others are still protected. Each period of the video stream uses a unique Media Key, which is a best practice to limit the impact of potential breaches to cryptographic systems.	• **Purpose:** Device-specific key used to encrypt Media Keys. • **Format:** RSA-2048 • **Usage:** Used to securely wrap the Media Keys, helping ensure that only authorized users can access the video data. • **Lifecycle:** The camera key is generated on the Verkada Backend during the camera's provisioning and is linked to that specific device. The public key is transmitted across the network and stored on the camera. The private key is kept in the VKMS system, protected by an AWS Key Management Service (KMS) key with restricted access. The Camera Key is designed to persist throughout the device's lifecycle. • **Security Role:** Stored encrypted Media Keys must be unwrapped by the Camera Key using VKMS before stored media can be viewed.

Authentication

Verkada customers can manage users through a Single Sign-on (SSO) Identity Provider (IdP). When a user logs in to Verkada using SSO, the process starts with the user entering their login details, such as a username and password, on a login page provided by the Identity Provider (which is often run by a trusted SaaS IdP vendor like Microsoft Entra or Okta). Customers who do not want to use SSO can use a Verkada-provided authentication service. For IdPs that support the OpenID Connect (OIDC) protocol, the following steps occur:

1. **Authentication Request:** The user's login request is sent to the OpenID Connect provider, which checks the user's credentials.
2. **Token Creation:** If the credentials are correct, the provider generates an ID token, which is returned to the browser client in response. The ID token is a special kind of JSON Web Token (JWT) that contains information about the user, such as their username or email address, and confirms that they have successfully logged in. This token is digitally signed by the OpenID Connect provider to confirm its authenticity.
3. **Token Validation:** The browser client forwards the ID token to the Verkada Backend, which checks the token's signature, issuer and expiration to confirm its validity. Some IdP implementations further use PKCE (RFC 7636) which provides additional protections.
4. **Login Confirmation:** Once the ID token is validated, the Verkada Backend logs the user in and grants them a Verkada session token to access resources. The session token is a bearer token that can be presented to any Verkada Backend API to identify the user within the Verkada system. The ID token from the IdP is not used again after the Verkada session token has been issued.
5. **Metadata Request:** The browser can also make additional requests to the IdP provider to retrieve metadata as a logged in user. This allows the browser to retrieve additional fields related to the customer or user which are not present in the ID token.

In simple terms, the ID token acts like a digital passport, proving the user's identity to the Verkada Backend. The Verkada Backend validates the token and thereafter issues a secondary, Verkada session token which acts like a residency permit, establishing privileges assigned to the bearer.

Note that in this process, the Verkada Backend does not gain access to the IdP directly–it only receives proof of identity. Only the browser can make the Metadata Request described in step (5).

Data flows

Now that we have defined the basics, we can provide an overview of the system.

Camera Provisioning

1. The account administrator authenticates via the Identity Provider (IdP).
2. The camera is added via the web interface, and provisioning begins.
3. A unique Camera Key is generated on the Verkada Backend and stored in VKMS.
4. The Camera Key's public key is deployed to the camera over a secure connection.

Video Recording, Encryption and Storage

1. Camera generates an ephemeral, random Media Key every 24 hours.
2. Camera captures video footage.
3. Video data is encrypted with the Media Key using the HLS standard's (RFC 8216) encryption (AES-128-CBC) mode.
4. The Media Key is encrypted using the camera's public key (RSA-2048-OAEP).
5. Encrypted video data and encrypted Media Keys are stored on the device and/or sent to Verkada's cloud infrastructure.

Video Decryption and Viewing

1. User authenticates via the Identity Provider (IdP).
2. User with proper authorization requests video from the Media Proxy through the browser.
3. Encrypted video and encrypted Media Keys are retrieved from cloud storage or directly from the camera (authenticated).
4. Media Proxy returns encrypted media data to the client.
5. Media Proxy passes the wrapped Media Key to VKMS for decryption by the Camera Key and returns the Media Key to the client.
6. The browser receives the encrypted media data and the decrypted Media Key.
7. The decrypted Media Key is used to decrypt the video data in the browser and the media is displayed to the user.

Shortcomings

The Camera Key is the lynchpin for decrypting all video data associated with a particular camera. If this key is compromised through a security breach or an insider attack, all encrypted video data associated with that camera no longer benefits from our application layer encryption. The compromise of this single (per-camera) key could expose a camera's sensitive video footage if combined with the ability to download encrypted media data.

Enterprise Controlled Encryption (ECE)

Enterprise Controlled Encryption (ECE) significantly enhances security by introducing several advanced cryptographic features. These features address the limitations of traditional encryption methods by decentralizing key management and providing robust protections against various attack vectors.

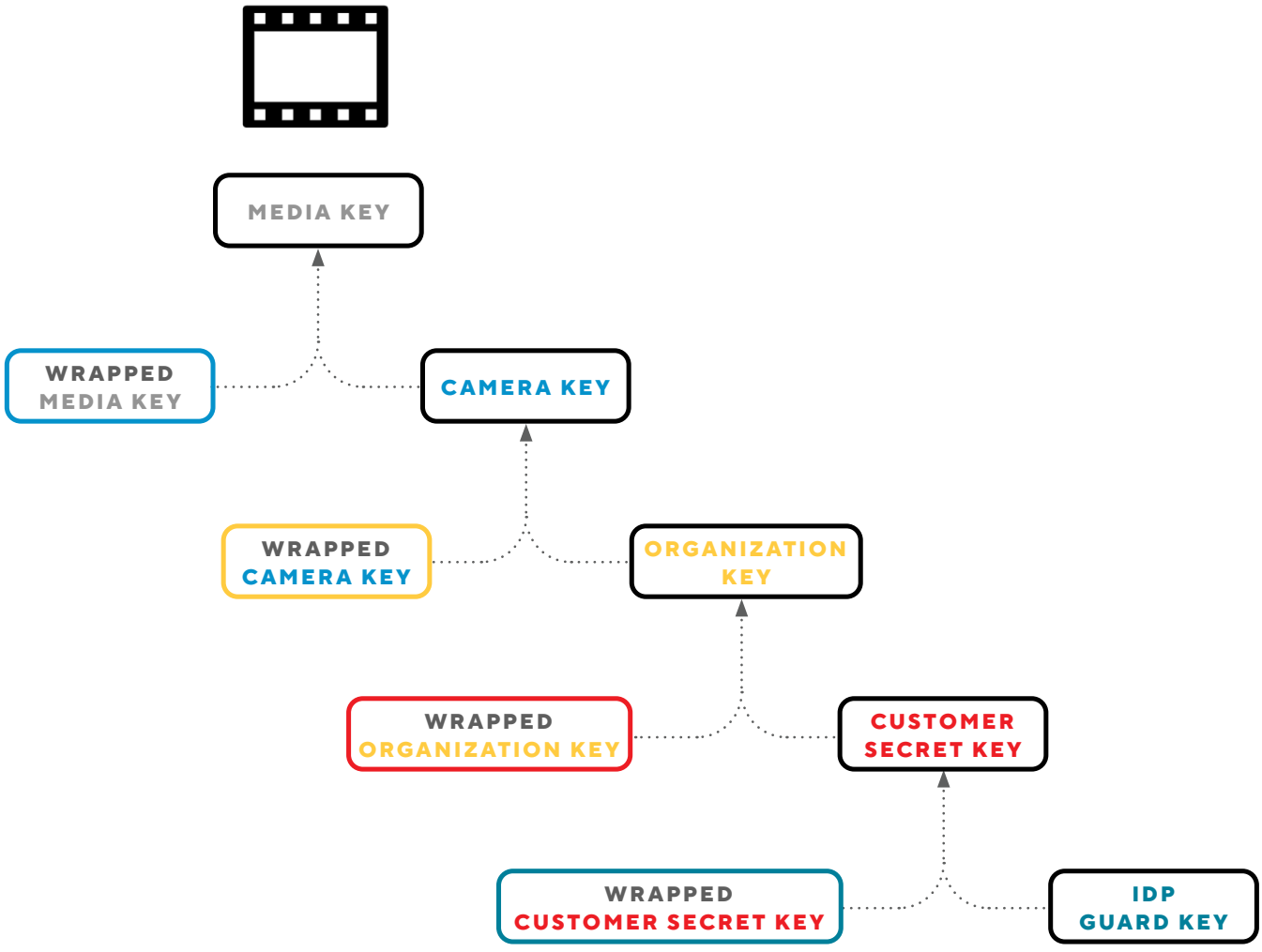


Figure 4: ECE Key Hierarchy

Camera Key now under Customer Control

The fundamental change from the current scheme to ECE is that the Camera Key is now unknown to Verkada–it is exclusively a secret of the customer. To avoid confusion, we will refer to a Camera Key under customer control as a Customer Camera Key. These keys are then wrapped with another customer controlled key: the Organization Key. This allows a single small customer secret value to protect an arbitrary number of Customer Camera Keys and by extension all the Media Keys down to the media itself.

Customer Camera Key	Organization Key
• **Purpose:** Camera-specific key is used to encrypt Media Keys and is unknown to Verkada. • **Format:** RSA-2048 • Usage: Securely wraps the Media Keys, helping ensure that only the customer can access the video data. • **Lifecycle:** Generated during the camera's provisioning by either the Browser or ephemerally on the Verkada Backend and associated with a specific device. The public key is retrieved by the camera across the network in a Key Policy File (KPF). The camera uses the KPF signing mechanism (described below) to validate updates to this key. The private key is encrypted using the Organization Key. The encrypted Customer Camera Key is stored in VKMS and further protected by AWS KMS. • **Security Role:** Media Keys are encrypted by the Customer Camera Key. Being camera-specific limits the impact of disclosure. When combined with key rotation, forward secrecy is also promoted since the encrypted Customer Camera Key in VKMS is regarded as a secret value that must be protected. Rotating the Customer Camera Key periodically and protecting its encrypted form means that a leaked encrypted Media Key might not be decrypted by a previously leaked Organization Key.	• **Purpose:** Encrypts Customer Camera Keys, thereby, simplifying key management. • **Format:** RSA-2048 • **Usage:** Browser uses the Organization Key to unwrap the Customer Camera Keys, which are then used to decrypt Media Keys after the vendor layer has been removed by VKMS. • **Lifecycle:** Generated in the administrator's browser when the organization is provisioned. It is encrypted by the Customer Secret Key (described below), and the wrapped key is stored securely in VKMS. The unwrapped Organization Key is discarded by the Browser after the session is complete to minimize the client device compromise scenario. • **Security Role:** Enables the use of a short string as the Customer Secret. Also allows decoupling the Customer Secret Key, which can only be rotated by an IdP administrator, from the Customer Camera Key. This enables customer-initiated rotation of a Customer Camera Key without IdP administrator access.

Organization Key Split Between IdP and VKMS

Under ECE, the Organization Key is split, via key wrapping, between two entities: the Identity Provider (IdP) and the Verkada Key Management Service (VKMS). The Organization Key is first encrypted by a Customer Secret Key, and then the Customer Secret Key is encrypted by an IdP Guard Key as described below. The full hierarchy is depicted in Figure 4.

This split ensures that no single third-party entity (either Verkada or the IdP) has full control over the customer's Organization Key, reducing the risk of compromise through insider attacks or breaches. In other words, the Organization Key is a secret of the customer client devices. Customers who want to view recorded video in a specific camera must collaborate with and be verified by both Verkada and the Identity Provider to generate or access the Customer Camera Key.

Customer Secret Key

- **Purpose:** Protects the Organization Key.
- **Format:** AES-256
- **Usage:** Wraps the Organization Key.
- **Lifecycle:** Generated in the administrator's browser when the organization is provisioned. It is encrypted by the IdP Guard Key to form the Customer Secret, which is stored securely in the customer's IdP application profile metadata.
- **Security Role:** The root of secrecy that provides customer control over their encryption. Value is kept secret from both Verkada and IdP.

IdP Guard Key

- **Purpose:** Protects the Customer Secret Key.
- **Format:** AES-256
- **Usage:** Wraps the Customer Secret Key to produce the Customer Secret.
- **Lifecycle:** Generated by VKMS when the organization is provisioned. The key is available upon request to users once they log in.
- **Security Role:** Guards against disclosure of the Customer Secret Key by the IdP. Splits control over the Organization Key with the IdP, providing an additional layer of security by requiring a valid authenticated session with VKMS to retrieve the IdP Guard Key. Value is kept secret from the IdP.

Customer Secret

- **Purpose:** Wrapped Customer Secret Key.
- **Format:** Base64 encoded AES-256-GCM ciphertext (with 12 byte IV, 16 byte tag)
- **Usage:** Value stored in the IdP and not in the Verkada Backend.
- **Lifecycle:** Generated in the administrator's browser when the organization is provisioned. Encrypted by the IdP Guard Key.
- **Security Role:** An encrypted copy of the root of secrecy and intended for storage in the customer's IdP system.

Camera Key Split between Customer and Verkada

ECE introduces the concept of key splitting via wrapping. The Camera Key is replaced with two RSA keypairs that are applied sequentially: the Customer Camera Key and the Vendor Camera Key (Verkada being the vendor). By having one layer of encryption that is only accessible by the customer on their client device and another layer of encryption that is only accessible by the Verkada Backend, there is a stronger assurance that both Verkada and another party need to be breached to expose the media.

Crucially, it also provides an opportunity for Verkada to detect and mitigate a potential attack. Under the ECE scheme, Verkada's backend remains the primary line of defense against decrypting stolen encrypted media and can bring all of its infosec tools and expertise to bear against an attack–both preventing and mitigating, but also logging so that the extent of any compromise can be accurately assessed and shared with the customer. Yet, under ECE the customer can always retain a check on Verkada's decryption capabilities, helping to nullify attacks that involve a compromised VKMS system.

Vendor Camera Key

- **Purpose:** Camera-specific key is used to add another layer of encryption to Media Keys.
- **Format:** RSA-2048 or higher
- **Usage:** Adds a second (vendor) layer of encryption to stored Media Keys in addition to using the Customer Camera Key.
- **Lifecycle:** The Vendor Camera Key is generated in VKMS and retrieved by the device. It is protected by AWS KMS at rest.
- **Security Role:** Encrypted Media Keys on persistent storage, either on device or in the cloud, require online VKMS requests to decrypt the vendor layer. VKMS only permits authorized parties to remove this second layer of encryption, keeping the Customer Camera Key from being a single point of failure. Making this key camera-specific also limits the impact of disclosure.

Workflows

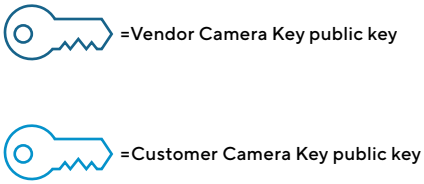
Now that we have defined the extended key hierarchy, we can provide an overview of the system usage.

Organization Enrollment	Camera Provisioning
1. The account administrator is authenticated via the Identity Provider (IdP). 2. A unique Organization Key, Customer Secret Key, and IdP Guard Key are generated. 3. Organization Key is encrypted with the Customer Secret Key (AES-256-GCM). 4. Customer Secret Key is encrypted with the IdP Guard Key (AES-256-GCM) to form the Customer Secret. 5. The IdP Guard Key and the wrapped Organization Key are uploaded to VKMS. 6. Customer Secret is uploaded to the IdP.	1. The camera is added via the web interface, and provisioning begins. 2. A unique Customer Camera Key is generated either in the administrator's browser or ephemerally in VKMS. 3. The new Customer Camera Key is encrypted with the current Organization Key (RSA-2048-OAEP-AES-128). 4. A unique Vendor Camera Key is generated and stored on VKMS. 5. The first Customer Camera Key's public key is deployed to the camera in a KPF (described later). 6. The first Vendor Camera Key's public key is retrieved from VKMS.

Video Recording, Encryption, and Storage

On Camera:

1. Media Key encrypts video (AES-128)



2. Customer Camera Key encrypts Media Key (RSA-OAEP)

3. Vendor Camera Key encrypts encrypted Media Key (RSA-OAEP + AES-128)

4. Camera and/or backend stores persistent data

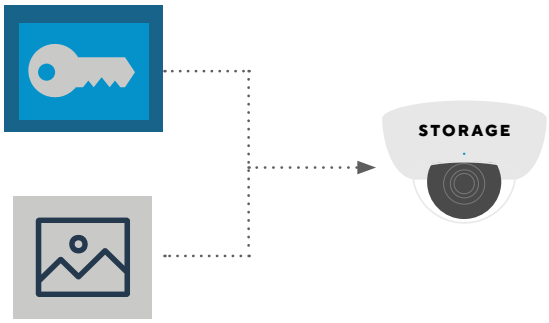


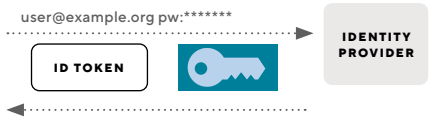
Figure 5: ECE video encryption sequence

1. Camera generates an ephemeral random Media Key every 20 minutes, then the camera captures video footage and video data is encrypted using the Media Key (AES-128-CBC).
2. The Media Key is encrypted to public keys according to the definitions in the Key Policy File (typically RSA-2048-OAEP).
3. Encrypted Media Keys intended for long term persistent storage are further encrypted using the Vendor Camera Key (RSA-2048-OAEP-AES-128-GCM).
4. Encrypted video data and encrypted Media Keys are stored on the device and/or sent to Verkada's cloud infrastructure.

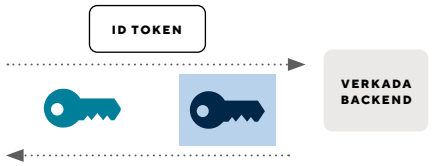
Video Decryption and Viewing

On Customer Client Device:

1. Log in to receive ID token and Customer Secret



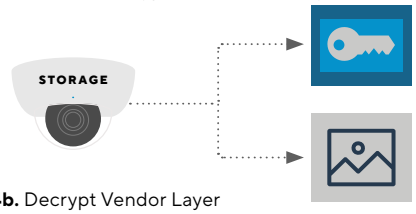
2. SSO to Verkada to receive keys



3. Request Media

On Verkada Backend:

4a. Retrieve encrypted Media and keys

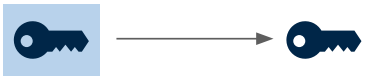


4b. Decrypt Vendor Layer

4c. Send encrypted Media and keys

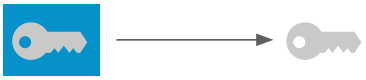
On Customer Client Device:

5a. Decrypt Organizations Key



5b. Decrypt Customer Camera Key

5c. Decrypt Media Key



6. Decrypt and play Media

Figure 6: ECE video decryption sequence

=Customer Secret Key
=IdP Guard Key
=Organization Key
=Vendor Camera Key
=Customer Camera Key public key
=Media Key
=Organization Key public key
=Customer Camera Key
=Media Key
=Customer Secret Key
=Organization Key

1. The user is authenticated via the Identity Provider (IdP) and receives a payload containing a wrapped Customer Secret Key (usually stored as a custom claim within a JWT access token).
2. The user is logged in to Verkada Backend with an ID token from the IdP. The user then receives the IdP Guard Key and encrypted Organization Key.
3. The user requests media.
4. The Verkada Backend retrieves encrypted video, the vendor-encrypted and customer-encrypted media key from cloud/device storage. It then decrypts one layer of the encrypted media key with the Vendor Camera Key (authenticated).
5. The IdP Guard Key decrypts the Customer Secret Key, which decrypts the Organization Key, which decrypts the Customer Camera Key, which decrypts the Media Key in the browser.
6. The decrypted Media Key is used to decrypt the video data and display it to the user.

Key Rotation

Generating and using new key material (i.e. "rotating" a key) is a critical step for post-compromise recovery. Depending on the attack scenario, different secrets playing different roles in the system may be at risk; however, as a general rule it is best practice to assume each secret is at risk of undetected exposure with some probability that increases over time. To help remedy this, keys should be rotated periodically at a frequency dictated by the risk of their anticipated compromise.

We plan to announce general availability of Customer Secret Key rotation, described below, in a future release.

Customer Secret Key Rotation

The Customer Secret Key possibly has the greatest risk of exposure because it is usually available in the browser whenever the user has logged in using SSO to Verkada. It is also the only key in the decryption chain which VKMS does not have access to, requiring a recursive key rotation (down to the Organization Key and the Customer Camera Keys) to recover post-compromise since all of these intermediate encrypted keys can be decrypted. Media Keys are already frequently rotated.

Fortunately, the ECE scheme is designed to withstand a Customer Secret Key compromise. Two additional elements–the underlying encrypted data (including encrypted Customer Camera Keys) and the Vendor Camera Keys held in VKMS–also need to be accessed. Since Verkada controls VKMS and can retrieve the encrypted data, when illicit insider access is a paramount concern, we recommend somewhat frequent rotation of the Customer Secret Key. On the other hand, if ECE is used as a defense in depth against a variety of threat actors, our design requires the involvement of Verkada Backend in providing data and Vendor Camera Key decryption and thus extends the recommended time between Customer Secret Key rotations.

To rotate the Customer Secret Key, the steps for Organization Enrollment are essentially performed again:

1. Generate a new Customer Secret Key and IdP Guard Key to form a new Customer Secret.
2. Save the corresponding new Customer Secret with the IdP and the new IdP Guard Key gets saved with VKMS.
3. The old Customer Secret Key is encrypted with the new Customer Secret Key (AES-256-GCM) and saved in VKMS, allowing access to previously recorded data to be accessible using just the new Customer Secret.
4. A new Organization Key generated.
5. New Customer Camera Keys are also generated and encrypted to the new Organization Key. The new public key is retrieved by the camera in a KPF (described later).

Customer Camera Key rotation in ECE is implemented by changing the recipient keys in the Key Policy File as described in the next section.

Customer Camera Keys may be numerous so generation and provisioning of new ones may happen gradually and take some time to complete. In the interim, encrypting data to the old Customer Secret Key will continue to work but at a reduced level of secrecy.

Vendor Camera Key Rotation

Although the process is not automated yet, Vendor Camera Keys can be rotated by VKMS without customer involvement. This provides post-compromise recovery for a VKMS compromise.

Key Policy File (KPF) as Customer Root of Control

Previous sections have established that cameras need to receive a public key used for encrypting video (Customer Camera Key). This key may rotate over time as needed by the customer and additional recipient keys may need to be added or removed to accommodate features. In order to prevent the camera from encrypting to an arbitrary key, the file must be signed by a customer controlled key.

The ECE scheme uses a Key Policy File (KPF) to define strict policies on key usage, instructing the firmware to use specified keys for specific time ranges and data categories. The Key Policy File acts as a governance mechanism, providing transparency and control over how the camera performs media encryption.

Cameras regularly interact with VKMS to receive updates to their KPFs. Each update is authenticated through verification of a signature generated in the browser by a private KPF Signing Key. The signing key is protected by the Organization Key and thus controlled by the customer so that only customer-authorized changes can be applied.

Within the ECE framework, the KPF can specify multiple Customer Camera Keys and define different usage for each. For instance, a key may be associated with live streaming but not history streaming or with video but not images. Data classification, provided through encryption and enforced by the KPF definitions, allows for flexibility in separating classes of recipients. For example, a live link viewer or an API consumer or a support technician might have a key that can decrypt live video but not history and this can be rotated at a different pace. This technology foundation enables a sophisticated key management scheme and will power additional capabilities for customer control into the future.

Note that while key rotation is not fully implemented in the initial ECE release, the Key Policy File foundation allows for its secure addition in the future.

KPF Signing Key

- **Purpose:** Establishes customer control over a Key Policy File.
- **Format:** RSA-2048
- **Usage:** Signs a version of the Key Policy File, including the public key of the next KPF Signing Key to use. The signature utilizes RSA-PSS with a SHA256 hash.
- **Lifecycle:** The key is available to certain logged in users upon request. It is rotated for each new version of the KPF file.
- **Security Role:** Establishes the integrity of the Key Policy File, which contains definitions of which keys to use for each category of data.

Workflows

With KPF, the camera provisioning process described earlier is augmented to bootstrap the customer-signed update mechanism.

Camera KPF Updates

1. The client generates a new KPF Signing Key.
2. The new KPF Signing Key is encrypted with the Organization Key.
3. The client generates a KPF specifying the desired public key as an encryption key and the new KPF Signing Key's public key as the update key.
4. The client retrieves the encrypted existing KPF Signing Key for the camera from VKMS.
5. The client decrypts the KPF Signing Key and uses it to sign the new KPF.
6. The new KPF, its signature, and the encrypted new KPF Signing Key are uploaded to VKMS.
7. The camera pulls the KPF, verifies the new KPF signature using the old KPF's update key definition.
8. The camera begins to encrypt according to the rules on the new KPF.

Sequence Number and Key Rotation

One critical feature of the KPF system is that it doesn't permit a rollback to older, potentially compromised keys. Each KPF contains a sequence number that the camera checks to ensure it is higher than the previous sequence number before applying any updates. If the new KPF has a higher sequence number, the camera applies the changes, which may involve rotating to new keys, updating key usage groups, and implementing temporary keys with specified expiration times. This key rotation process is important for maintaining forward secrecy, as it minimizes the impact of any potential key compromise by regularly replacing old keys with new ones.

Trust Model

The KPF addresses the problem of making trusted updates to recipient public keys. This means that only the customer would be able to modify the list of public keys that are able to decrypt media data.

However, the problem of setting the initial KPF remains. The camera and the administrator are physically remote, necessitating a middleman to transfer. The administrator must be able to asynchronously initialize a camera while it is offline, which eliminates the possibility of a handshake phase before transferring. A pre-initialized device keypair could act as a trust anchor which a customer client device could verify–but the verification needs to happen in the other direction: the camera needs to verify a signature created by the customer, not the other way. Without a handshake or a separate secure channel, we are unaware of a way to fully prevent the installation of a modified first KPF.

For ECE, we therefore rely on a Trust-on-First-Use (TOFU) model. The first KPF for a camera is always trusted, but updates are chain-verified from that point on. This makes the first KPF essentially the root of control. Since the camera initialization happens only once, the window of compromise for a particular device is small and this risk should be acceptable. [4]

4. There are possibilities to detect if the TOFU process was compromised even if it cannot be prevented. The previously mentioned possible pre-initialized keypair could be used for attestation that the proper KPF is being used. We could alternatively embed a visual watermark in the pre-compression video to essentially create a separate channel. Nevertheless, we currently regard surreptitiously modifying the initial KPF along with the entire chain of future updates as being a difficult attack.

Attack Scenarios

When assessing the security of the ECE system, it is important to understand the potential risks associated with various compromises. These include IdP compromise, browser compromise, session compromise, mass download attacks, and VKMS compromise. Each scenario presents distinct challenges, and in combination can compound the overall risk to the system.

In this analysis, it is assumed that the overall goal of the attacker is to obtain plaintext media either for any arbitrary camera or for a particular chosen target camera ID. In order to obtain plaintext media, the attacker needs to obtain (1) the encrypted media data and (2) all the encryption keys required to decrypt it.

Verkada Session Compromise

In a session compromise, the attacker obtains a string called the session token which is used to authenticate requests from a client to a server.

The Verkada session token is how a logged-in client gains access to keys in VKMS and encrypted media data. Without separate access to the Customer Secret, the attack is ineffective and plaintext media cannot be produced. However, the intermediate keys and encrypted media data could be harvested and saved offline for a future attack when the Customer Secret might become available through other means. If encrypted media data and intermediate keys have been harvested from a session compromise, it may be necessary to rotate the Customer Secret Key in order to restore the original level of security.

Verkada session tokens are afforded many best-practices protections: limited validity, early revocation by either the user or the organization administrator, use of the HttpOnly attribute in cookies and the use of TLS.

Targeting a particular camera ID with this attack further requires that the Verkada session belongs to a user who has sufficient permissions to view the camera in question because the encrypted Customer Camera Key is only returned to users with permission to view the associated camera.

Another form of session compromise is an authorization bypass, where the attacker doesn't have stolen session tokens but is able to skip authorization checks. This could take the form of a legitimate user account from a different customer organization, which could be missing firewall or filter configuration, or an insider with overprivileged access to production resources. If this compromise affects only Verkada and not the IdP, it should be ineffective in the same way because the Customer Secret is kept in the actual customer's IdP.

Identity Provider (IdP) Session Compromise

The IdP session token is how a logged-in client gains access to the IdP system. The power and validity of these depend on the IdP vendor's implementation. In general, however, the IdP session token might be enough to perform the SSO login to obtain a Verkada session token and also request the encrypted Customer Secret Key. In this case, the attacker's goal can be achieved.

Targeting a particular camera ID with this attack further requires that the IdP session token belongs to a user with sufficient permissions to view the camera in question.

IdP Compromise

An IdP compromise could provide an attacker with two possible resources: first, the Customer Secret; second, the ability to login without having user credentials.

Obtaining the encrypted Customer Secret Key accomplishes the attacker's goal if the intermediate keys from VKMS and the encrypted media data have also been obtained.

Having the capability to perform arbitrary valid logins results in a compromise equivalent to the IdP Session Compromise except that it can be performed on any chosen user (within that IdP) and, by extension, chosen camera ID.

User Credentials Compromise

When user credentials, such as a username and password, matching those in the Identity Provider are compromised, the attacker can perform an IdP login and then gain a valid session token with Verkada through SSO. Because the attacker is authenticated with the IdP, they can also request the Customer Secret from the IdP. Together, these achieve the attacker's goal while the session is valid because the encrypted media and intermediate keys are available from Verkada using the session token.

Targeting a particular camera ID with this attack further requires that the credentials belong to a user who has sufficient permissions to view the camera in question.

Revoking the Verkada session token, which can be done by an organization administrator, will remove access to further encrypted media data and new intermediate keys, thus stopping the attacker. The attacker could still possess the Customer Secret Key and other customer keys (potentially Organization Keys, Customer Camera Keys and Media Keys) that were available to the user. Post-compromise recovery is accomplished by following the Customer Secret Key rotation procedure described earlier in the Key Rotation section.

Mass Data Download

A mass download attack occurs when an attacker gains direct access to Cloud Storage and downloads large volumes of encrypted or unencrypted video data. Data at rest is typically encrypted at the application level using a Media Key and the encrypted data is stored together with an encrypted copy of the Media Key. Since the encrypted Media Key is designed to be guarded by an additional Vendor Key layer as described earlier, offline decryption further requires compromise of both VKMS and the Customer Secret Key. As long as VKMS's Vendor Camera Keys are not disclosed, the attacker should be forced to perform an online attack using a valid session, requesting VKMS to decrypt each encrypted Media Key.

While this attack can be devastating because of its scope, it now requires three components to succeed: 1) the encrypted media data from Cloud Storage, 2) the Vendor Camera Keys or authenticated access to VKMS, and 3) the encrypted Customer Secret Key from the IdP or logged-in client device.

VKMS Compromise

If Verkada's VKMS is breached, attackers could access several critical resources for a customer: 1) the IdP Guard Key, which protects the Customer Secret Key, 2) the encrypted Organization Key, 3) all encrypted Customer Camera Keys, and 4) all Vendor Camera Keys.

Yet, even with all these keys, an attacker still requires both the Customer Secret and the encrypted media data itself to be able to decrypt and access any media data.

Post-compromise, the security of newly encrypted data continues to be degraded until the keys are rotated. The customer must perform the Customer Secret Key rotation procedure, described earlier, to restore full ECE protection. VKMS can rotate the Vendor Camera Keys without interaction of the customer, facilitating a rapid partial recovery.

Browser Compromise

In a browser compromise, an attacker exploits vulnerabilities in the user's web browser to intercept session tokens, execute malicious scripts, or access decrypted data variables. Some possible vectors include malicious browser plugins, host machine malware and supply chain attacks in web application source code. A logged-in client browser stands at the most privileged position since it is already capable of obtaining plaintext media.

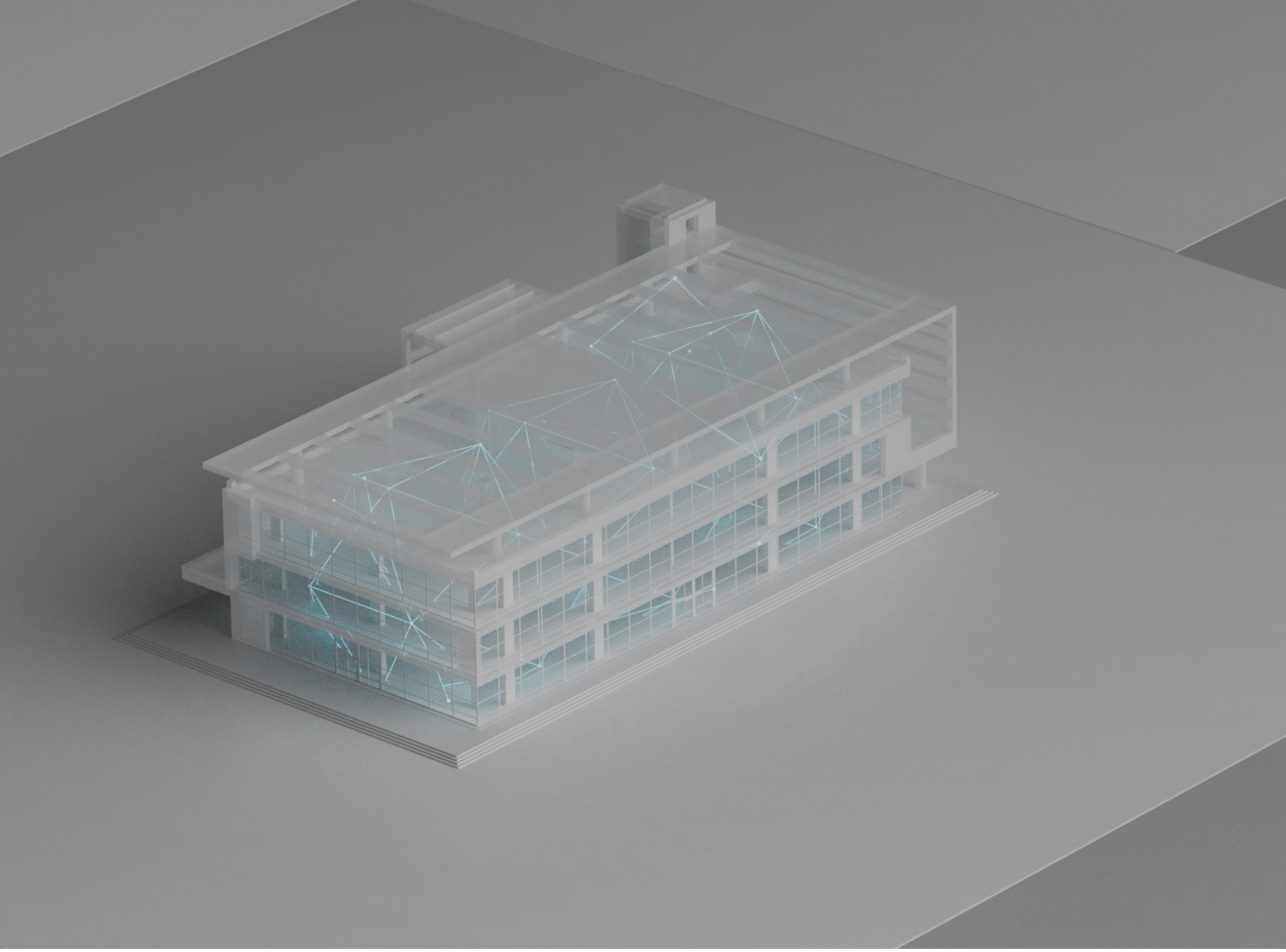
Since the web application is used to perform many of the decryption operations in the ECE framework, this type of compromise can lead to the disclosure of many customer keys in addition to credentials or session tokens, potentially providing a persistent advantage to the attacker even after the user account or session has been deactivated. Customer Secret Key rotation is necessary to restore the full protection of ECE. Since it is not always possible to detect a browser compromise, it is recommended to perform the Customer Secret Key rotation periodically to narrow the time window of attacker advantage.

Cascading attacks

These attack scenarios are interconnected. ECE's security is built on splitting secrets across independent targets. For instance, a VKMS compromise alone is not enough to compromise data privacy. Similarly, accessing the Customer Secret or a Mass Data Download alone does not compromise data privacy. A VKMS compromise combined with the encrypted media data does not achieve the attacker's goal without an authentication bypass for the IdP or the Customer Secret.

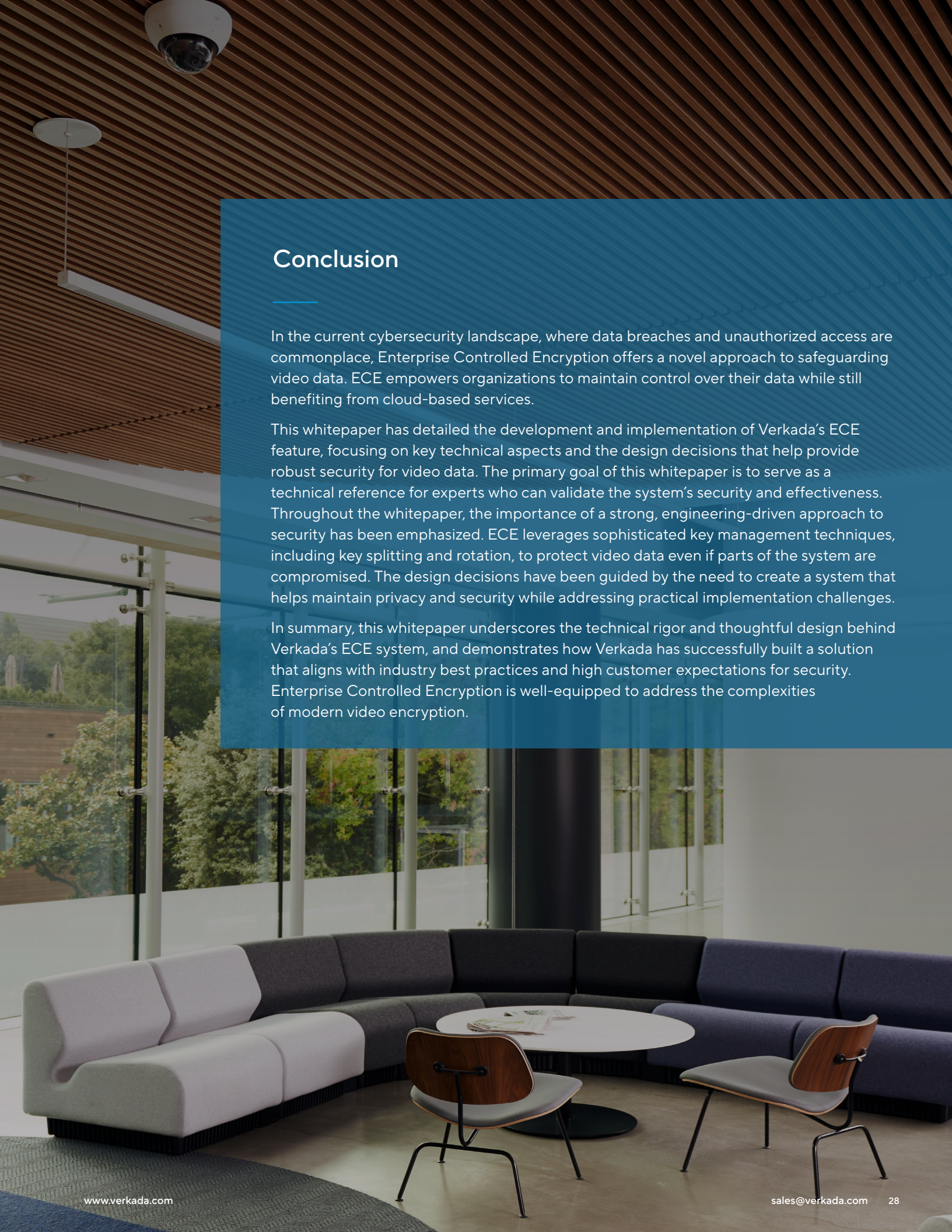
Each risk must be addressed holistically to ensure that the entire system's security is maintained and the cascading effects of any compromise are minimized. The critical part of this analysis is that a VKMS compromise alone is not enough to compromise data privacy.

Further research and analysis on the effects of key rotation on forward secrecy and post-compromise security is needed to formalize the security bounds. There are also clear design options to improve the security properties of this system concerning offline attacks using advanced constructions like OPRF; however, the gains would be limited compared to the engineering cost of deploying a new cryptographic primitive that isn't supported on all platforms at this time.



Additional Work

This whitepaper presents the fundamental elements of the ECE scheme that Verkada has implemented to expand the security protections in its products. It's important to note that we did not discuss other numerous additions and extensions that have built on these foundations to provide additional functionality beyond viewing pre-recorded videos: more secure live link sharing, support case handling, browser compatibility and computer vision search. In many of these cases we have been able to leverage the limited scope or lifetime of each key and the interlocking nature of separate controls to build functionality that combines camera, browser, and backend processing while retaining many ECE protections. We see this as a hopeful portent for continuing to develop advanced cloud-managed physical security products in a privacy aware way.



Conclusion

In the current cybersecurity landscape, where data breaches and unauthorized access are commonplace, Enterprise Controlled Encryption offers a novel approach to safeguarding video data. ECE empowers organizations to maintain control over their data while still benefiting from cloud-based services.

This whitepaper has detailed the development and implementation of Verkada's ECE feature, focusing on key technical aspects and the design decisions that help provide robust security for video data. The primary goal of this whitepaper is to serve as a technical reference for experts who can validate the system's security and effectiveness. Throughout the whitepaper, the importance of a strong, engineering-driven approach to security has been emphasized. ECE leverages sophisticated key management techniques, including key splitting and rotation, to protect video data even if parts of the system are compromised. The design decisions have been guided by the need to create a system that helps maintain privacy and security while addressing practical implementation challenges.

In summary, this whitepaper underscores the technical rigor and thoughtful design behind Verkada's ECE system, and demonstrates how Verkada has successfully built a solution that aligns with industry best practices and high customer expectations for security. Enterprise Controlled Encryption is well-equipped to address the complexities of modern video encryption.