Proceedings on Privacy Enhancing Technologies ; 2022 (2):25–46

Yohan Beugin*, Quinn Burke, Blaine Hoak, Ryan Sheatsley, Eric Pauley, Gang Tan, Syed Rafiul Hussain, and Patrick McDaniel

Building a Privacy-Preserving Smart Camera System

Abstract: Millions of consumers depend on smart camera systems to remotely monitor their homes and businesses. However, the architecture and design of popular commercial systems require users to relinquish control of their data to untrusted third parties, such as service providers (e.g., the cloud). Third parties therefore can (and in some instances have) access the video footage without the users' knowledge or consent—violating the core tenet of user privacy. In this paper, we present `CaCTUs`, a privacy-preserving smart Camera system Controlled Totally by Users. `CaCTUs` *returns control to the user*; the root of trust begins with the user and is maintained through a series of cryptographic protocols, designed to support popular features, such as sharing, deleting, and viewing videos live. We show that the system can support live streaming with a latency of 2 s at a frame rate of 10 fps and a resolution of 480 p. In so doing, we demonstrate that it is feasible to implement a performant smart-camera system that leverages the convenience of a cloud-based model while retaining the ability to control access to (private) data.

Keywords: Smart Camera System, Privacy-Preserving, Complete Mediation, End-to-end Video Encryption, Fine-grained and Peer-to-Peer Delegation

DOI 10.2478/popets-2022-0034
Received 2021-08-31; revised 2021-12-15; accepted 2021-12-16.

***Corresponding Author: Yohan Beugin:** The Pennsylvania State University, E-mail: yohan@beugin.org
Quinn Burke: The Pennsylvania State University, E-mail: qkb5007@psu.edu
Blaine Hoak: The Pennsylvania State University, E-mail: bhoak@psu.edu
Ryan Sheatsley: The Pennsylvania State University, E-mail: sheatsley@psu.edu
Eric Pauley: The Pennsylvania State University, E-mail: epauley@psu.edu
Gang Tan: The Pennsylvania State University, E-mail: gtan@psu.edu
Syed Rafiul Hussain: The Pennsylvania State University, E-mail: hussain1@psu.edu
Patrick McDaniel: The Pennsylvania State University, E-mail: mcdaniel@cse.psu.edu

1 Introduction

Smart camera systems are changing the way consumers secure their homes and businesses. Commercial camera systems have been remarkably successful; they have become the *de facto* monitoring system, as they offer the following essential services with plug-and-play support: (1) watch live and recorded video feeds, (2) share videos with others, (3) delete recorded videos, (4) recover access to the system, and (5) perform a full factory reset. Yet, while the market demand for smart camera systems continues to grow rapidly as reported by *Ring* [10, 22, 23, 52], *Wyze* [53], and *Arlo* [4], consumers have come to realize that the costs of owning a smart camera system are not exclusively monetary.

Commercially available smart camera systems follow a threat model that mandates undue trust *by design*; the service provider is granted unfettered access to the video content of any consumer who uses their system. Ring has recently come under legal scrutiny [17, 20] for allowing more than 2,000 government agencies to directly request videos from users without formal due process [7, 30, 41, 44]. Perhaps even more troubling, employees are viewing and annotating live user streams for research [5, 13] while others are abusing their access to view and share users' videos online [25, 46]. Moreover, research has posited that these systems can be transformed into mass surveillance systems, given their widespread adoption [8–10, 28, 30, 34, 35, 43]. The message behind the underlying design of modern smart camera systems is clear: users do not have control over their own videos and system, compromising user privacy.

To this end, we answer the following question: *can users afford all of the features present in commercial smart camera systems, without compromising their privacy?* A cryptographic approach is a plausible way to protect users' privacy in that it enables users to solely assume control over videos stored in the cloud. However, practical realizations of such systems face several key challenges: cryptographic protections (e.g., encryption) incur computational overheads, affecting system performance, since stored videos are encrypted; a fine-grained sharing scheme (i.e., sharing specific video fragments)

requires a user-controlled key management system; and generating, storing, rotating, and re-negotiating cryptographic keys poses further challenges on performance and usability. Troublingly, even if such challenges could be addressed, recent events have demonstrated that encryption alone is not sufficient to protect users from abuses by governments through coercion [19, 29]. Thus, meeting performance, security, privacy, and usability goals demands a novel approach that is sensitive to the unique requirements of this domain.

In this paper, we present `CaCTUs`, a privacy-preserving smart **Ca**mera system **C**ontrolled **T**otally by **Us**ers. Inspired by information privacy laws, `CaCTUs` is designed to enforce three privacy goals through known security properties; (1) *the right to not be seen*: the user is assured *confidentiality* of stored videos and live video streams, (2) *the right of sole ownership*: the user (and only the user) is trusted, and has *complete mediation* over access to their data by others, (3) *the right to be forgotten*: deleted videos are not recoverable, even in cases of coercion.

To meet the required feature set of commercial systems and address the stringent technical challenges and privacy goals of smart camera systems, we design `CaCTUs` as follows: it allows the user to solely assume control of the smart camera system through a direct and physical pairing process (that is, without relying on or trusting third parties); isolates and protects access to video footage through encryption, key rotation, and key management; enables viewing live and stored videos through performance-aware cryptographic algorithms; supports video deletion and factory reset via key rotation and management; and provides fine-grained (i.e., on the scale of seconds) peer-to-peer delegation of video footage through a binary key tree. We make the following contributions:

1. We present `CaCTUs`, a privacy-preserving smart **Ca**mera system **C**ontrolled **T**otally by **Us**ers, that returns controls of the system to users without compromising features found in commercial smart camera systems.
2. We perform a functional user evaluation of our system and find that `CaCTUs` is natural and easy to use, all while meeting our privacy goals.
3. We perform a performance evaluation of `CaCTUs` on a Raspberry Pi and find that we can serve a live video stream at a resolution of 480 p, at a frame rate of 10 fps, and with a latency of 2 s.

To encourage future privacy-preserving smart camera systems, we release `CaCTUs` as open-source software, available at https://github.com/siis/CaCTUs.

2 Background

2.1 Smart Camera Systems

A smart camera system is a collection of cameras that are connected to the Internet, allowing *owners* (i.e., those who purchase and configure the system) to view live and recorded videos of their homes from anywhere. Most companies sell their systems as an integrated ecosystem: cameras work with a purpose-built smartphone application that allows the owner to view footage, delegate access, and administer their smart camera system. At the core of these systems are five functions: (1) recording and streaming, (2) sharing (delegation), (3) deleting, (4) access recovery, and (5) factory reset. Each function places requirements and motivates the architecture of the ecosystems available to consumers.

Recording and Streaming. Camera systems allow owners to view live and recorded footage from all cameras they own using an application on their smartphone, allowing them to monitor the current status of their property. As the most fundamental function provided by camera systems, this is expected to work reliably and globally: users want to be able to view footage anywhere, and recover footage even in the case of physical failures of the camera or home Internet connection. To facilitate this, consumer smart camera systems currently entrust the data to a cloud provider, streaming camera data to cloud storage as it is captured and making it available to the owner's device. As a result, access to the footage is managed by the cloud provider, who must be trusted to prevent unauthorized access.

Delegation. Owners want to share access to their camera systems with others. We refer to this capability as *delegation*. Whether used to provide a house-sitter with access to live footage during a vacation, or sharing video of an incident after the fact, this delegation is expected to be *fine-grained*, meaning it applies to specific users (*delegatees*) for only the portion of time that they need access. Consumer smart camera systems allow policy enforcement as a means of delegation: each user has an attached policy for the time range of live or recorded footage they may access, and cloud storage mediates this to prevent delegatees from exceeding their policies.

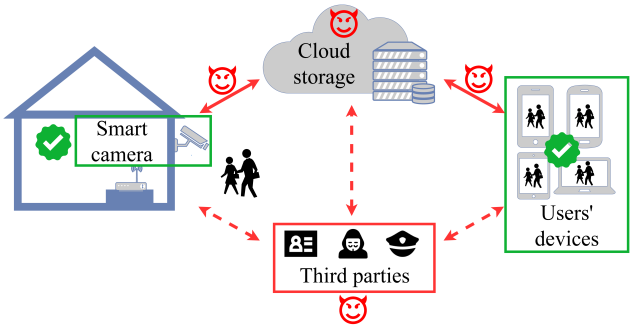

Fig. 1. Overview of the components and actors in the privacy-preserving smart camera system CaCTUs: the camera devices and the users' devices are trusted while the cloud storage provider, the networks, and any other third party are untrusted.

Deletion. Owners expect the ability to fully delete their data to prevent further access by any party. The right of consumers to delete their personal data has been codified in legal frameworks, such as the European General Data Protection Regulation (GDPR) [40] and California Consumer Privacy Act (CCPA) [31]. As recorded data from camera systems are saved to the cloud, owners must trust cloud providers to delete their data when requested, including copies stored elsewhere in the cloud.
Access Recovery. Since access to smart camera systems is mediated by a set of credentials (e.g., a username and password), these systems must account for the possibility of a user losing these credentials. When authentication is performed by a cloud service, this is relatively straightforward: the user's identity is verified via other means, such as a password reset through email. As we will discuss, however, such recovery is only trivial because of the trust assumptions of these systems, and we will see that this critical function requires careful thought under other trust models.
Reset. Finally, owners may wish to stop use of the smart camera system. In this case, they will expect all of their stored footage to be deleted, access to live footage revoked, and the device returned to a condition where it may be set up by another user. This is generally equivalent to a delete operation for all stored data, followed by resetting the physical camera itself.

2.2 Privacy in Smart Camera Systems

Smart camera systems have been shown to have both privacy and security risks [5, 7, 12, 13, 25, 30, 41, 44, 46]. As a result, information privacy laws have been passed, which aim to address expectations of privacy in online and physical environments (e.g., the California Consumer Privacy Act (CCPA), California Privacy Rights Act (CPRA), and the European General Data Protection Regulation (GDPR), among others [6, 31, 32, 38–40, 42]). However, these legal frameworks often suggest vague, rather than concrete requirements for enforcing privacy in specific end-user devices such as smart cameras. Motivated by these recommendations and requirements necessary to prevent previously-discussed privacy incidents, we can achieve a *privacy-preserving system* by affording the system owner the following rights :

1. **Right to not be seen**: the owner is assured that no unauthorized user can view stored videos or live video streams.
2. **Right of sole ownership**: the owner retains full control of their data and who they trust.
3. **Right to be forgotten**: deleted videos are not recoverable, even in cases of coercion.

In practice, these rights imply that device owners must have exclusive control over the collection of data, its uses, and the access delegations to it.

2.3 Threat Model

Our goal in this work is to demonstrate a smart camera system that provides feature parity with commercial systems while placing no trust in a cloud provider or other third party. As such, we work under a threat model wherein edge devices (i.e., the smart camera and end-user devices) are trusted, but the cloud storage provider, network, and any other third-party service are untrusted (see Figure 1 for an overview of CaCTUs).

We only trust the devices owned by the users (cameras, smartphones, laptops, or tablets) to securely handle the encryption and decryption keys used in the system, and we trust the device manufacturer to provide us with a camera device that correctly executes its functionality. This additionally implies that supply-chain exploits against the camera manufacturer are out of scope. We trust the other applications running on the users' devices (or that the operating system sufficiently isolates these applications) and we assume that the cryptographic algorithms used provide the advertised guarantees (e.g., Diffie-Hellman assumption [14] and RSA public-key cryptosystem [45]).

We also acknowledge that access under our system may be universally delegated: once granted access to a video, a party is not prevented from sharing with others or downloading and storing the videos somewhere else. Partial mitigations to this may be considered, but

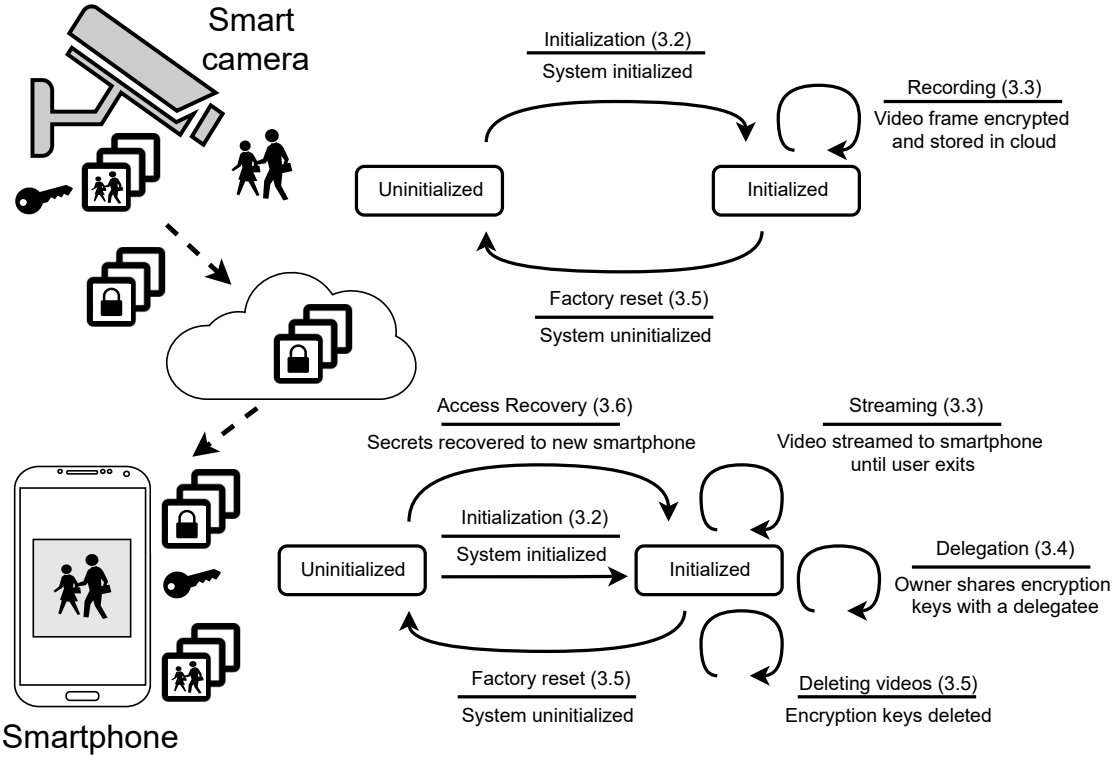

Fig. 2. Overview of the recording and streaming operations; the camera records video frames, encrypts them locally, sends the encrypted frames to the cloud, from where the smartphone application downloads them, decrypts them locally, and plays the video. The right part is the overview of the states and actions in the CaCTUs system for the camera at the top right and the owner's smartphone at the bottom right (references indicated between parentheses are the subsections to refer to for more details).

as such sharing can occur outside the purview of our system complete prevention is not possible. Encrypted frames are assumed to be publicly accessible (as we do not trust the cloud to do any access control mediation), thus we acknowledge that access pattern to the cloud storage may be leaked in CaCTUs. Ongoing research in Private Information Retrieval (PIR) or Oblivious RAM (ORAM) could provide potential mitigations through the use for example of random accesses and dummy writes to the cloud storage. Finally, physically tampering with the devices (modifying the hardware, chip-level changes to edit the software execution, etc.) and denial of service attacks are outside the scope of our work.

3 CaCTUs

3.1 Overview

In the following sections, we will describe how CaCTUs meets the privacy goals described in Section 2 by providing the following three security properties: (1) *confidentiality*: stored videos and live video streams cannot be viewed by unauthorized parties, (2) *complete mediation*: the user fully controls access to their data by others, and (3) *deletion*: deleted videos are not recoverable, even in cases of coercion. Each subsection motivates the privacy goals before providing technical details of the feature. We refer to Appendix A for the notation used in our cryptographic constructions and algorithms, to Appendix D for the details of the protocols, and to Appendix F for the storyboard of the CaCTUs's smartphone application. For clarity, we consider only a single camera, though our approach readily generalizes to multi-camera systems by applying the described protocols to each camera individually.

Figure 2 shows an overview and state diagram of CaCTUs. The camera locally encrypts recorded video frames before uploading them to cloud storage. The smartphone application performs the reverse operations: it downloads the frames, decrypts them locally, and plays the video. Regular key rotation and secure key management allow the system to support secure streaming, delegation, deletion, recovery, and reset.

	Camera	👁	ᛒ	Smartphone
1		$\overset{h_{PK_f}}{\longrightarrow}$		
2			$\overset{PK_f}{\longrightarrow}$	$h'_f = hash(PK_f)$ $h'_f \overset{?}{=} h_{PK_f}$
3	$h'_o = hash(PK_o)$		$\overset{PK_o}{\longleftarrow}$	$gen(SK_o, PK_o)$
4	$h'_o \overset{?}{=} h_{PK_o}$	$\overset{h_{PK_o}}{\longleftarrow}$		
5	$DH(SK_f, PK_o)$		$\overset{verif}{\longleftrightarrow}$	$DH(SK_o, PK_f)$
6	$gen(SK_c, PK_c)$		$\overset{PK_c}{\underset{(RSA)}{\longrightarrow}}$	
7	$init(secrets)$		$\overset{secrets}{\underset{(RSA)}{\longleftarrow}}$	$gen(escrow)$ $passphrase$

Table 1. Protocol followed by the camera and the owner's smartphone during the initialization, 👁 and ᛒ respectively correspond to what is obtained through the visual and Bluetooth channels.

Fig. 3. Establishment of a secure and authenticated Bluetooth pairing to setup CaCTUs by scanning QR codes (visual channel).

3.2 Initialization

During initialization, the user's smartphone and camera establish a trust association used for all other steps, so the security of this step is critical to that of the system as a whole. In CaCTUs, we adapt the *Seeing-Is-Believing* (SiB) technique introduced by McCune, Perrig, and Reiter [33] to establish an authenticated communication channel between the camera and smartphone when the devices share no prior context (and without having to trust any third party). Specifically, we use the visual channel as an out-of-band means to verify the authenticity of each end of a Bluetooth channel (see Figure 3).

This secure pairing bootstraps the system to allow the owner of the device to communicate directly with the camera (for system initialization) while ensuring confidentiality and integrity. Thus, negotiation of encryption keys can be done without any other party involved. Moreover, the required proximity and physical interaction between the devices would render attack attempts easily detectable by the system owners. The initialization protocol between the camera and the owner's smartphone application is as follows (see Table 1):

1. A pair of factory-generated asymmetric keys (SK_f, PK_f) is present on the camera device, and the hash of its public key h_{PK_f} is embedded into a QR code on the back of the device (recall from Section 2.3 that the supply chain is trusted). The owner's smartphone scans this QR code and stores its content.
2. The camera and the owner's smartphone connect through Bluetooth and the camera sends its public key PK_f to the owner, who computes the hash of the camera's public key h'_f and checks that it matches the hash retrieved from the QR code.
3. If they match, the owner's smartphone generates its own asymmetric pair of keys (SK_o, PK_o) and sends its public key PK_o through Bluetooth, the camera computes the hash h'_o of the owner's public key.
4. The owner points their smartphone's screen at the camera. On the screen is displayed the QR code with the hash of the owner's public key. The camera retrieves the content from the QR code and checks that it matches h'_o.
5. If the key hashes match, both devices now verify that the other device knows the secret key corresponding to the public key that they advertised earlier. This is can be done for instance by applying the Diffie-Hellman key exchange to compute their shared secret and then by exchanging a series of encrypted messages where both parties prove their knowledge.
6. The camera then generates a new asymmetric key pair (SK_c, PK_c) that it will use for future communication (to avoid further reliance on the factory-generated key). The camera then shares its public key PK_c with the owner through Bluetooth in an authenticated way using RSA and the factory-generated asymmetric key pair (SK_f, PK_f). Note that we could have used the shared secret computed at the previous step through Diffie-Hellman, however, we chose to use RSA to align this step of our initialization protocol with the future protocols pre-

sented in CaCTUs as well as because we want authentication of the messages.

7. Next, the owner sends their *secrets* to the camera device to complete system setup, this is done in a secure and authenticated way using RSA, but this time the new asymmetric key pair of the camera is used (SK_c, PK_c). First, they send *wifi credentials* of the wifi network the camera should connect to. Then, they generate and send a *seed key* that will be used to derive the keys to encrypt video frames. Lastly, they send *escrow material* (protected by a non-recoverable *passphrase*) that may be used by the owner to recover access to the system (see Section 3.6 for details about the escrow material). If necessary, during this step the owner could also configure the cloud storage option they want to use if different than the default one.

At this point, the system is initialized, the camera begins recording, and the owner can begin executing the other camera functions.

3.3 Recording and Streaming Videos

Here we describe how to ensure the *confidentiality*, *integrity*, *authenticity*, and *freshness* of the recorded video footage. The video frames and metadata are encrypted locally at the camera and asymmetrically signed in blocks of N frames, before being uploaded to cloud storage. At each key rotation, the camera device securely erases the encryption keys previously used as it does not need them anymore. To view a video, users of the system download the encrypted data from the cloud storage, derive the decryption keys locally (if they have access to them), and decrypt the frames to rebuild the video. Thus, only users with access to the appropriate decryption keys can view the footage. We discuss in Appendix C how to leverage one-time signatures (i.e., hashed signatures [18]), to build a scheme that allows us to sign every frame[1]. Nonetheless, as the camera is recording, it performs the following:

1. Consider a block of N frames. Each frame F_i is recorded at timestamp t_i.
2. A key rotation scheme is used to derive encryption keys for a given frame (this will prove useful for delegation, discussed in Section 3.4). The rotation scheme $\mathcal{K}$ provides a key k_i, the key used to encrypt frame F_i. An initialization vector IV_i is randomly generated.

$$\{k_i = Extract(\mathcal{K}, i) | i \in [\![1, N]\!]\}$$

$$\{IV_i = RandBytes(16) | i \in [\![1, N]\!]\}$$

3. Next, each frame is symmetrically encrypted (**confidentiality)** into the corresponding ciphertext C_i using the AES algorithm in Galois/Counter Mode (GCM, chosen for its performance) with a 256-bit key. C_i is then concatenated with IV_i and t_i to be hashed into h_i (**integrity and freshness**).

$$\{C_i = AES256Enc(IV_i, k_i, F_i) | i \in [\![1, N]\!]\}$$

$$\{h_i = HMAC(k_i, C_i || IV_i || t_i) | i \in [\![1, N]\!]\}$$

4. A signature σ of the block is computed using the private key SK_c of the camera and the N hashes of the frames in this block (**integrity and authenticity**).

$$\sigma = Sign(SK_c, h_1 || h_2 || ... || h_N)$$

5. The encrypted and authenticated frames $\langle \{C_i, IV_i, t_i | i \in [\![1, N]\!]\}, \sigma \rangle$ are uploaded to the cloud along with their corresponding metadata (initialization vector used for encryption and timestamp).

Each user who has access to the correct decryption keys can download these encrypted and authenticated frames $\langle \{C_i, IV_i, t_i | i \in [\![1, N]\!]\}, \sigma \rangle$. To view the video, the user performs the following:

1. We consider a block of N frames of signature σ downloaded on demand. Each ciphertext C_i, encrypted using the initialization vector IV_i, corresponds to a frame recorded at timestamp t_i .
2. The corresponding symmetric key k_i is extracted from key rotation scheme $\mathcal{K}$. The hash h_i of each ciphertext C_i is computed.

$$\{k_i = Extract(\mathcal{K}, i) | i \in [\![1, N]\!]\}$$

$$\{h_i = HMAC(k_i, C_i || IV_i || t_i) | i \in [\![1, N]\!]\}$$

3. The signature σ of the block is verified with the public key PK_c of the camera and the N hashes of the frames in this block (**authenticity, integrity, and freshness**).

$$1 \stackrel{?}{=} Verify(PK_c, \sigma, h_1 || h_2 || ... || h_N)$$

1 The overhead of asymmetrically signing every frame is sufficiently large that it decreases the performance of the system by several orders of magnitude; thus the need for hashed signatures.

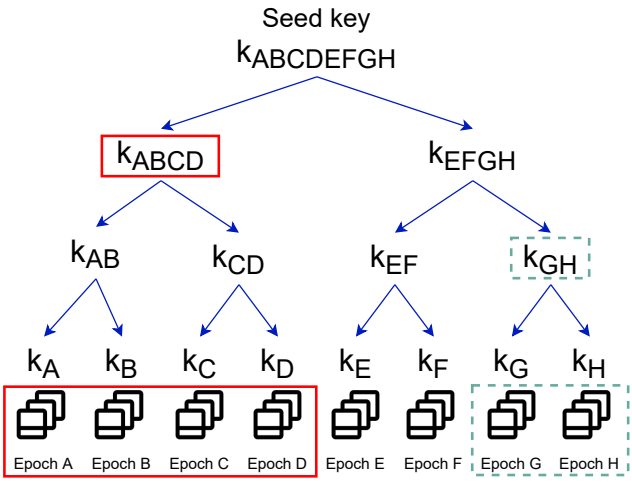

Fig. 4. Key Tree Construction used by CaCTUs, the keys to be shared to give access to the corresponding footage are framed.

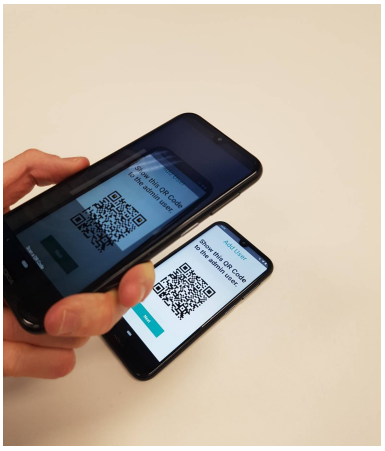

Fig. 5. The establishment of a secure and authenticated Bluetooth pairing with a shared user's device to delegate access.

4. If the signature is correct, each ciphertext C_i is then symmetrically decrypted into the corresponding frame F_i (**confidentiality**).

$$\{F_i = AES256Dec(IV_i, k_i, C_i) | i \in [\![1, N]\!]\}$$

Encryption is performed *end-to-end*: data is encrypted locally at the camera before being stored in the cloud and decrypted locally at the smartphone after being retrieved. Furthermore, integrity is ensured using an authenticated encryption scheme, so that the video footage cannot be tampered with during transmission or storage. Lastly, the identity of the camera is embedded into the video frames (and signed) so that users can attest the authenticity of the video footage. This scheme allows the user to verify the integrity, authenticity, and freshness of an arbitrary set of video frames, so that they then decrypt the frames and rebuild the video for playback.

3.4 Delegation

To protect privacy, users must have complete mediation over access to their videos. CaCTUs achieves this by ensuring that the owner has control of the keys used to encrypt the video footage. However, they may also want to delegate access to their videos (e.g., to friends or family) for different periods of time. Achieving fine-grained sharing capabilities for delegatees is nontrivial: we want to support delegation without knowing beforehand to whom the owners will delegate access or for how long. To enable this, we rotate the key used to encrypt the video frames at the end of every *epoch* (a fixed-size time interval). We use a binary key tree construction to facilitate the management of all the keys for the camera device, owners, and delegatees. A peer-to-peer pairing is adopted for sharing keys so that there is no reliance on a third party.

Recall that frames are encrypted using a symmetric key derived from a key rotation scheme $\mathcal{K}$. In practice, to support delegation, this rotation scheme is a binary key tree, inspired by the key tree introduced by Kocher [27]. The tree is of a fixed depth $d_{\mathcal{K}}$. In the tree, each leaf node holds some cryptographic key k and covers a specific epoch: a time interval $[t_j, t_{j+1})$ of fixed-size $\delta_{\mathcal{K}}$. The root node of the tree is initialized with a seed key that is negotiated during the initialization of CaCTUs (see Section 3.2). The timestamp of this negotiation is used for t_0, with $t_{j+1} = t_j + \delta_{\mathcal{K}}$. The leaf nodes in the tree hold the encryption keys for every epoch. Each node in the tree can be derived from the root node knowing the derivation equations and relations between the parent node and its two children.

Within each epoch, the symmetric key used for each frame is identical. The derivation of keys is based on a Hash-Based Key Derivation function (HKDF), which is a one-way process [24]. If we have k_{parent}, then:

$$\begin{aligned} k_{left} &= HKDF(k_{parent}) \\ k_{right} &= HKDF(k_{parent} \oplus 1) \end{aligned} \tag{1}$$

Therefore, for a given key in the tree, a user can only derive the keys below it but not the ones above (see Figure 4 for an illustration). The binary key tree is useful for several reasons. First, it decreases the amount of keys that need to be shared with the delegatees, as the derivation algorithm is publicly known by the Kerckhoffs's principle and a specific part of the key tree

can be reconstructed on-demand by a delegatee that is given access to a node of the tree. Moreover, it is storage space-efficient, as the key rotation mechanism generates a large number of encryption keys. When key rotation happens and the camera device securely erases the previous encryption keys, the camera device only needs to save at most $d_{\mathcal{K}}$ nodes to keep functioning. Further, it is simple to find which node is responsible for the encryption or decryption of a specific timestamped frame (through a binary search). Lastly, it facilitates the recovery process for the owners as only a subset of keys needs to be recovered to rebuild the tree from the escrow material.

Delegation is supported by giving some keys of the key tree to each delegatee (an example is given in Figure 4 where keys to share and their corresponding epochs are framed). From there, each delegatee can derive the associated keys to correctly decrypt and view only video footage captured within the time window they were authorized for. The depth $d_{\mathcal{K}}$ and the epoch size $\delta_{\mathcal{K}}$ are configurable parameters of the system. For our implementation, we selected $d_{\mathcal{K}} = 32$ and $\delta_{\mathcal{K}} = 10\,\text{s}$. Importantly, this choice of parameters demonstrates the worst-case performance users can expect from CaCTUs. Specifically, these parameters result in a key tree that covers a lifespan of 1362 years at a 10 s level of granularity. We provide more reasonable parameter choices (and, thus, expected performance at deployment) in Table 2. The size of an epoch $\delta_{\mathcal{K}}$ corresponds to the lowest delegation granularity achievable in CaCTUs. Delegation uses a peer-to-peer approach similar to the secure and authenticated pairing presented in Section 3.2. The only difference being that there is no need to use a factory-generated key and QR code as both devices have a screen and can therefore generate their initial asymmetric key pairs dynamically. Exact details of this protocol can be found in Section D.1.

	Lifespan for		Storage space
$d_{\mathcal{K}}$	$\delta_{\mathcal{K}} = 10\,\text{s}$	$\delta_{\mathcal{K}} = 60\,\text{s}$	(worst-case scenario)
24	5 years	32 years	256 MB
26	21 years	128 years	1 GB
28	85 years	511 years	4 GB
30	340 years	2043 years	16 GB
32	1362 years	8172 years	64 GB

Table 2. Lifespan of the key tree for an epoch time $\delta_{\mathcal{K}}$ of 10 s and 60 s and storage space needed to save to disk the encryption keys in the worst-case scenario for different depth sizes $d_{\mathcal{K}}$.

3.5 Deleting Videos and Factory Reset

CaCTUs users must be able to delete their videos and factory reset their system. Owners can achieve this by deleting select decryption keys (i.e., a subset of nodes in the key tree) so that the keys below them in the tree cannot be recomputed. Recall that the camera device is securely deleting the encryption keys as soon as it does not need them anymore due to key rotation. As the keys used to encrypt the video frames are at the leaves of the key tree, to prevent being able to recompute such leaves, each node (i.e., key) along the path to the leaf must also be deleted. Thus, with a tree depth of $d_{\mathcal{K}}$, for each portion of video content composed of n_e epochs to delete, the upper bound of the number of nodes that must be deleted from the tree is $\mathcal{O}(d_{\mathcal{K}} n_e)$.

To provide an example: in Figure 4, to delete k_A (the key for epoch A), keys $\{k_{AB},\ k_{ABCD},\ k_{ABCDEFGH}\}$ must also be deleted. Thus, $\{k_B, k_{CD}, k_{EFGH}\}$ must be saved in this sparser key tree so that the corresponding videos can still be decrypted. The key material in the escrow material also needs to be updated accordingly. Table 2 presents the storage space required in the worst-case scenario where every other epoch has been deleted. Note that in practice, such a scenario is very unlikely to happen as it will render the system unusable. Furthermore, the owner is very unlikely to delete beforehand keys that would have been used to encrypt future videos. Thus, in practice far less amount of storage space is required, specifically thanks to our binary key tree structure that enables dynamic derivation of lower keys. In this way, the binary key tree allows the owner to delete video footage at arbitrary time scales. Note that this operation is indeed equivalent to deleting the video frames as they can not be correctly decrypted without the keys, even in cases of coercion.

To factory reset the camera, the owner sends the request to the camera, timestamped and authenticated with the secret key SK_o of the owner to verify the legitimacy of the request. Then, both the camera and owner's smartphone delete the key tree $\mathcal{K}$ they have access to, returning both devices to an uninitialized state. See Section D.2 for more details.

Note that with this mechanism, delegatees may still know some decryption keys that were deleted from the owner's device. As discussed in our threat model (see Section 2.3), once granted access to a video, a party is not prevented anyway from having already shared or downloaded the video.

3.6 Access Recovery

In case owners lose access to their smartphone, they must be able to recover access to the system. However, in a privacy-preserving system where no third party is trusted, achieving this is nontrivial. To solve this, we create *escrow material* during the initialization step that contains all of the information needed by the owners to recover access to their system. The escrow material is encrypted and stored on the camera. We note that the escrow material is protected by an unrecoverable passphrase only known by the owner and is therefore not restricted to being stored on the camera. The escrow material gives access to the following secrets :

- The *owner's asymmetric key pair* (SK_o, PK_o) encrypted with a randomly generated key of size 128 bits which representation in hexadecimal corresponds to the passphrase displayed to the owner during initialization.
- The *key material* necessary to build the key tree $\mathcal{K}$ (for details about this key tree see Section 3.4) asymmetrically encrypted with the owner's key.
- The *asymmetric public key* PK_c of the camera (does not need to be encrypted).

To recover access, owners use their new smartphone to open a Bluetooth connection to the camera to retrieve the escrow material. Note that no assumption regarding the status of the owner has been made so far; anyone who is in physical range from the camera can request the escrow material. However, only the owners have knowledge of the recovery passphrase and can use it to decrypt the escrow material. Once decrypted, the owner assumes control of the system from the new smartphone and can reconstruct the corresponding key tree $\mathcal{K}$ to retrieve their video footage. More details can be found in Section D.3.

4 Evaluation

The goal of this section is to evaluate the effectiveness and efficiency of `CaCTUs` with respect to four metrics: security, privacy, usability, and performance. To this end, we evaluate three research questions:

- **RQ1:** Does `CaCTUs` enforce our privacy requirements, while offering the same feature set found in commercial systems?
- **RQ2:** Is `CaCTUs` easy to use for end users?
- **RQ3:** Can `CaCTUs` operate at sufficient resolution, frame rate, and latency to meet the needs of smart camera system owners?

Experimental Setup. Experiments were performed using a *Raspberry Pi 4 Model B* with 2GB of RAM, equipped with a camera module. Our implementation of the `CaCTUs` camera system is written in C, while the paired mobile companion application was written in Java and installed on a *Nokia 4.2* smartphone, running Android 10. Further evaluation setup details are described in Appendix B.

4.1 Privacy and Security Analysis (RQ1)

We begin by analyzing how `CaCTUs` achieves the security and privacy properties discussed in Section 2.2, as well as related practical concerns.

Confidentiality. `CaCTUs` aims to protect user data from being read by unauthorized third parties. As such, owners manage access to the decryption keys, and any administrative actions must be authorized by them. Further, this encryption is performed *end-to-end*: data is encrypted locally at the camera before being stored in the cloud and decrypted locally at the smartphone after being retrieved. As a result, parties who have not been delegated access through key sharing are unable to view the encrypted frames, protecting confidentiality of the footage, which ensures the right to not be seen.

Complete Mediation. The key-tree construction of `CaCTUs` ensures that access delegation is performed at a fine-grained level. Because this delegation must be performed by the owner (who holds the seed key of the key tree), no other party is able to grant access to footage, thus providing complete mediation of access. The key tree ensures this mediation cryptographically: a user without the seed or delegated key is unable to decrypt footage regardless of access control on the encrypted data, assuming the cryptographic primitives are secure. This enforces the right to sole ownership.

Data Deletion. An emerging property of the previous two properties is that the owner has sole ownership and control over the encryption/decryption keys. Thus, if the owner decides they want to delete all stored videos in the cloud, they can simply delete their keys, which makes it impossible for them to view the corresponding encrypted footage stored in the cloud, even in case of coercion. This ensures owners' right to be forgotten.

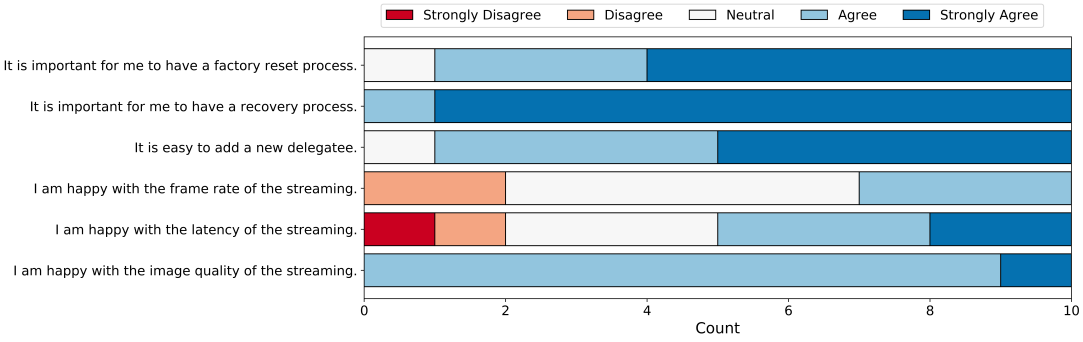

Fig. 6. Likert scale evaluation of different statements made by the 10 participants during the functional user evaluation of CaCTUs.

4.2 Functional User Evaluation (RQ2)

We performed a functional user evaluation of our implementation of CaCTUs with ten participants. Studies have shown that this group size is sufficient (twice the minimum number) to identify most of the issues within a system design [36, 37]. The goals were to assess the functional ease of use of CaCTUs and identify what aspects could be improved further. To this end, participants were asked to initialize the system and use the different functionalities as if they had just bought it: view live and recorded streams, share videos with a delegatee, recover access after simulating the loss of their smartphone, and reset the system. The detailed protocol that was followed is described in Appendix E. All institutional requirements were met for this functional user evaluation of CaCTUs. We obtained approval from the Institutional Review Board (IRB) of our university and a consent form was signed by the participants at the beginning of their session.

Out of the ten participants, four were female and six male, seven of them were between 20-25 years old and the three between 45-55 years old.

Goals and Limitations. Our modest goals through this functional user evaluation of CaCTUs were to assess whether guided users can perform the functions of the system that we laid out and identify where our proof-of-concept of the implementation fell short (with respect to performance or feature design) and how it could be improved. As a consequence, our participants knew they were evaluating CaCTUs. Thus, our functional evaluation differs from traditional usability studies that may compare different system designs to identify which is the most usable, has the best interface, or what UX options are optimal for ensuring that users understand a specific privacy concept of the system. We defer a more comprehensive usability study to future work.

User Interface. Overall, the participants thought that the interface of the implemented smart camera system was simple to understand and navigate through. They liked that for every process there were step-by-step instructions displayed to them. They noted that these directions were clear, straightforward, and self-explanatory.

Secure and Authenticated Pairing. Despite that pairing requires both setting up a Bluetooth connection and scanning QR codes, the participants found the process simple and easy. They expressed that it was more straightforward than steps they had to perform with other systems. They also felt that using both Bluetooth and a visual channel was more secure even if they did not explicitly always know why.

Quality of the Footage. As shown in Figure 6, the participants agreed that the image quality of the video stream (480 p) and the frame rate (10 fps) were sufficient for security and surveillance purposes. However, some believed that the latency of the system was a bit of a downside as it tended to defeat the initial purpose of being able to monitor what was recorded by the smart camera system in real-time. See Section 5 for potential optimizations to CaCTUs.

Granularity Options. The participants were impressed by the granularity with which they could view specific segments of video footage (up to the second), but believed that the same granularity option for delegation was not necessary; they stated that the primary use case would be to share access for several hours or days. This concern can be addressed by displaying only up to the minute and adding an option for more precision in the settings of the application for instance. The majority of the participants also expressed desire to have an option to quickly delegate unlimited access, but then figured out that they would not be able to revoke such access without having to reset the system.

Access Recovery and Factory Reset. Regarding the recovery process, the participants were divided on whether the recovery passphrase should be randomly generated by the system or if it should give the owner the opportunity to choose it. They felt that they could lose the recovery passphrase or forget about it if they did not choose it. However, they agreed that as this passphrase allows to recover full access to the system, it might be less secure to let the owners pick their own. One participant remarked that the ability to recover access without needing to trust a third party (i.e., without using a recovery email address for instance) was interesting.

As shown on Figure 6, most participants found it important to be able to recover access to and factory reset their system, as oftentimes owners may want to recover access (to retrieve videos) before factory resetting it. The participants found the recovery and factory reset processes easy to perform.

Missing Features. Participants expressed that they would like to see the following features implemented: motion detection-triggered recording, remote pairing and delegation, two-way audio support to listen in on and remotely speak through devices, password-locked smartphone application for additional security, and application availability across different platforms (e.g., Android, iOS, and web interface). See Section 5 for details about such extensions to CaCTUs.

4.3 Performance Evaluation (RQ3)

We now evaluate the streaming performance of CaCTUs, focusing our efforts on three key metrics: latency (delay from time of recording), stream image resolution, and frame rate. As a baseline, commercial systems achieve frame rates of 30 fps in 1080 p (1920 x 1080 pixels), with a latency of several milliseconds. However, we note that these systems have been largely optimized to be sold to consumers, and they do not guarantee similar privacy-preserving features as CaCTUs. We discuss potential optimizations to CaCTUs in Section 5.

System Latency. For a video resolution of 480 p we obtained a latency of 2 s at a frame rate of 10 fps.

System Bottlenecks. Next, we measure the effect that each phase of streaming has on the stream latency. The values have been averaged over 1,000 frames. For this evaluation, we picked the same parameters as in Section 3.4 for a worst-case scenario (i.e., $d_{\mathcal{K}} = 32$ and $\delta_{\mathcal{K}} = 10$ s), to show the performance baseline that can be expected from CaCTUs. Table 3 shows the results when the camera device is recording for a video quality of 480 p. As shown, the largest contributor to the latency is the upload/download of the encrypted frames to/from the cloud storage server during live stream. As previously stated, we have not optimized this specific point in our implementation of CaCTUs -which would have the same contribution to latency in a system not using encryption techniques- but we discuss means to do so in Section 5. Note that within the same epoch the same key is used, but between epochs, we need to derive the new key, that is why the standard deviation is larger than the average for the key extraction.

Device	Operation	Delay (ms)	σ (ms)
Camera	Key Extraction	0.05 ms	0.2 ms
	Frame Encryption	2.8 ms	1.0 ms
	Hash	1.9 ms	0.7 ms
	Signature	8.8 ms	2.9 ms
	Upload	510 ms	420 ms
Smartphone	Download	420 ms	160 ms
	Key Extraction	0.02 ms	0.1 ms
	Frame Decryption	0.4 ms	0.5 ms
	Hash Verification	1.9 ms	0.4 ms
	Signature Verification	0.5 ms	0.5 ms

Table 3. Time delay averaged over 1,000 frames and standard deviation σ for each live stream operation, while recording with a resolution of 480 p.

5 Discussion

In the following, we discuss potential optimizations for improving the performance of CaCTUs, practical considerations for deploying it as a commercial system, and extensions of our approach to other devices.

5.1 Improving Latency

We identify several components that could be improved to reduce the overall latency of the system, namely; (1) cryptographic accelerators, (2) network relays, (3) streaming libraries, and (4) video compression techniques. Such improvements are described below.

As discussed in Section 4, our implementation used a Raspberry Pi, wherein the (relatively weak) CPU was responsible for handling all processing, including encryption. Since cryptographic operations dominate many of the features in CaCTUs, we expect substantial

gains in latency by leveraging dedicated cryptographic accelerators as seen in many other crypto-dominated applications, such as in IoT [26].

Streaming live video can be demanding on the network. Specifically, popular video streaming platforms (such as Netflix, Hulu, and Disney Plus) employ a variety of techniques to bring video data as close to the user as possible. These techniques often take the form of caches, content delivery networks, or dedicated network infrastructure designed to serve high-bandwidth content quickly [1, 16, 47]. Naturally, these techniques could substantially improve the performance of CaCTUs to be even closer to commercial-grade systems.

The smart camera systems available today use optimized streaming protocols to deliver video content quickly [1]. Given that it was necessary for us to implement our video streaming protocol from scratch (to support our encryption and key rotation schemes), we could see further improvements by augmenting current protocols to support our design. In a similar vein, our current implementation operates at the frame-level, while commercial systems operate at the block-level (and thus exploit compression algorithms commonly used in video streaming applications [3]). Moreover, popular techniques such as adaptive video playback or frame dropping could also be used to improve the throughput of CaCTUs. We defer such improvements to future work.

5.2 Deploying CaCTUS as a Commercial System

Here, we highlight some challenges (motivated by notable features in commercial systems and suggestions from our functional user evaluation of CaCTUs) that should be addressed to realize commercial implementations of CaCTUs without compromising any of our privacy goals.

Relaxing Proximity. To uphold our privacy goals, delegatees need to be within local proximity of the owners (e.g., to perform delegation and pairing through the QR codes). While this was not an area of concern in the functional user evaluation of CaCTUs since participants were always close to the camera system, this can be challenging if users wish to delegate access remotely. We did conceptually derive a scheme to support this capability, while upholding our privacy goals. Specifically, CaCTUs could be extended to asymmetrically[2] encrypt the keys needed by delegatees and upload them to the cloud storage or use another third party service like email, from where the remote delegatee could retrieve, decrypt, and then use these keys to access the video stream for some predetermined period of time.

Revoking Access to Unrecorded Videos. Note that a similar scheme to the one described in the previous paragraph could also be used to allow owners to revoke access to delegatees to unrecorded videos without needing to factory reset the system. The sharing of the keys only needs to be done periodically at a certain interval that could be configured by the owner for each delegatee.

Supporting Audio. In our current implementation, we focused on supporting video data only, and not audio. However, the system design does not prohibit extensions to support audio-video recording and streaming. The extension is straightforward: we can treat some audio sample as a "frame", as it is done for video, and apply similar encryption operations. For sampling, a Linear Predictive Coding (LPC), which is widely deployed by telephone companies for speech encoding and processing, could be used [48]. Moreover, LPC can even be leveraged to transmit audio in reverse; that is, be applied so the users could speak into their smartphones and have the audio replayed through the camera, as commonly seen in commercial systems.

Supporting Motion Detection. Many commercial systems support forms of motion detection to notify users of events they may be interested in viewing. We find this feature valuable for both users as well as the camera system in that video recording need only be saved if motion was detected[3]. CaCTUs could support this capability through addition of a physical hardware sensor or a software solution. This addition would save significant space in cloud storage systems and reduce the cost related to storage. In a similar vein older data could be overwritten at a fixed interval that could be configured by the owner.

Physical Security. Finally, as explained in our threat model, we did not take into consideration physical tampering attacks against the camera device. Techniques

2 Clearly, how the public and private keys are computed and transferred between the owner and the remote delegatee needs to be done in a security-preserving manner.

3 To mitigate side-channel attacks and to not leak behavior patterns, meaningless data could be randomly uploaded to the cloud.

	Right to not be seen		*Right of sole ownership*	*Right to be forgotten*
Camera System	**Video Encryption**	**Owner Controls Access**	**Only the Owner is Trusted**	**Video Deletion**
PrivacyCam [11]	✗ **(ROI only)**	✗	✗	✗
TrustCAM [50]	✗ **(ROI only)**	✗	✗ **(trusts a central station)**	✗
TrustEYE.M4 [51]	✓	✗	✓	✗
SoC-based [21]	✓	✗	✗ **(relies on a Trusted Authority)**	✗
Signcryption [49]	✓	✗	✗ **(uses a Key Distribution Center)**	✗
Pinto [54]	✗ **(ROI only)**	✗	✗	✗
`CaCTUs`	✓	✓	✓	✓

Table 4. Comparison of privacy guarantees of smart camera systems proposed in the literature (ROI stands for Region Of Interest).

such as package design so that the device is tamper-proof should be explored. Specifically, local storage, battery, reset buttons, serial ports, firmware, etc., should be protected to prevent physical access by untrusted parties.

5.3 Extending to Other Devices

The protocols used in `CaCTUs` could be easily extended to other devices recording time-stamped data streams such as; temperature, humidity level, heartbeat monitoring, audio, power usage, etc. As video stream is the most bandwidth-intensive of these applications, performance for such devices is expected to be even better. Note that in the initialization step of `CaCTUs`, we leverage the fact that both devices (the camera and the smartphone) have a camera sensors, however, this may not be the case for other IoT devices, we defer to the *Seeing-Is-Believing* (SiB) technique introduced by McCune, Perrig, and Reiter [33] for a discussion of the guarantees provided in such a case.

6 Related Work

While this work is the first to examine a smart camera system that is privacy-preserving, there is already a large body of research studying the security of smart camera systems. Here, we detail the gaps in prior work and how `CaCTUs` addresses them. Refer to Table 4 for a comparison of the privacy guarantees of these systems.

Alharbi and Aspinall introduced a security analysis framework for IoT smart cameras that weighs the threat of significant risks (e.g., unencrypted video streaming) across various platforms [2]. However, it focuses on the security of the camera device and only partially addresses some vulnerabilities of other components of the system. For instance, the authors do not discuss the use of cloud storage to remotely access recordings, nor procedures for securely pairing a smartphone and camera. `CaCTUs` provides an end-to-end secure solution starting from camera initialization to protect the secrecy and integrity of both communications and data.

Haider and Rinner proposed a SoC-based smart camera that uses physically unclonable functions (PUFs) to generate encryption keys [21]. The keys are used to encrypt video frames at the camera before storing them in remote cloud storage, thus removing the requirement that the owner trust the cloud service provider. However, this approach has several limitations. First, a trusted authority is required to create camera device fingerprints during key generation; in a commercial system, the trusted authority will likely be the manufacturer, whom in general is not trusted by the camera owner. Moreover, the key extraction procedure using PUFs only produces a fixed number of encryption keys, which does not align with the feature-set typically desired by smart-camera owners—e.g., being able to delegate camera access to other people with fine granularity. Finally, as the fingerprint is physically embedded into the hardware, camera owners will likely need to get a brand new device if the encryption keys are leaked. `CaCTUs` addresses these shortcomings by making the owner (i.e., their smartphone) the root of trust: they store the secrets and share expendable keys with both the camera and delegatees. This simultaneously gives control back to the owner while enabling delegation and system reset without specialized (or new) hardware.

Winkler and Rinner introduced TrustEYE.MP4 [51], a monitoring framework that provides similar secrecy and integrity guarantees envisioned by `CaCTUs`. However, as the same key is used to encrypt all the videos, they do not address the unique challenges in delegating access (e.g., how to share and revoke access to video data at particular timescales) or deleting videos. Similarly, Ullah, Rinner, and Marcenaro proposed using signencryption [49] to encrypt and sign

video frames at once, but they also did not consider the challenges related to video deletion and delegation. CaCTUs addresses these issues by storing the system secrets at the owner's smartphone and using a secure key rotation scheme to enable fine-grained delegation.

Finally, PrivacyCam [11], TrustCAM [50], and Pinto [54] do not fully address the privacy challenges in a smart camera system as they attempt to solve a problem of a different nature and setting. These systems try to protect the anonymity of people or of vehicle license plates recorded in public spaces by detecting privacy sensitive regions and selectively encrypting or blurring them while leaving the rest of the video frame unperturbed. However, even this approach is limited in protecting anonymity: people can still be identified through their clothes or actions, and blurred videos still disclose behavior patterns such as when users are at home. Moreover, the system does not address data privacy of the video recordings.

7 Conclusion

This paper presented CaCTUs, a smart camera system with popular commercial features that *returns control of videos to the users*. In CaCTUs, we provide the owners with full control over their system, isolation and protection of the access to the video footage, deletion and factory reset, as well as peer-to-peer and fine-grained delegation. We leverage physical and direct pairing for system initialization (that is, without relying on trusting third parties), performance-aware cryptographic algorithms to support video streaming, key rotation and management through a binary key tree to provide video deletion, factory reset, and fine-grained (i.e., on the order of seconds) peer-to-peer delegation of video footage. Additionally, CaCTUs satisfies performance requirements necessary to execute the most popular commercial functionalities of smart camera systems, such as live streaming with 2 s of latency at a frame rate of 10 fps and resolution of 480 p, while protecting the users' privacy. CaCTUs serves as an existence proof that smart camera systems need not compromise the privacy of users to be afforded the modern capabilities that commercial systems offer today.

8 Acknowledgment

We would like to thank Christie Warren for her help in the design of the user interface of the smartphone application for this project as well as Dr. Hanrahan for his valuable feedback on the protocol of the functional user evaluation of CaCTUs. We also sincerely thank all the users that have participated in the test and evaluation sessions of our implementation of CaCTUs.

Funding acknowledgment: This material is based upon work supported by, or in part by, the National Science Foundation under Grant No. CNS-1805310 and Grant No. CNS-1564105, and the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant No. W911NF-19-1-0374. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation hereon.

References

[1] V. K. Adhikari et al. Measurement study of Netflix, Hulu, and a tale of three CDNs. IEEE/ACM Transactions on Networking, 23 (2014)(6):1984–1997.

[2] R. Alharbi and D. Aspinall. An IoT analysis framework: An investigation of IoT smart cameras' vulnerabilities. In: Living in the Internet of Things: Cybersecurity of the IoT - 2018, pages 1–10 (2018). 10.1049/cp.2018.0047.

[3] J. G. Apostolopoulos, W.-t. Tan, and S. J. Wee. Video streaming: Concepts, algorithms, and systems. HP Laboratories, report HPL-2002-260, (2002).

[4] Arlo. Arlo - Investor Relations (2021). URL: https://investor.arlo.com/ir-home/default.aspx. Last Accessed: 2021-04-06.

[5] S. Biddle. For Owners of Amazon's Ring Security Cameras, Strangers May Have Been Watching Too. The Intercept, (2019). URL: https://theintercept.com/2019/01/10/amazon-ring-security-camera/. Last Accessed: 2020-09-08.

[6] E. D. P. Board. Guidelines 3/2019 on processing of personal data through video devices (2019). URL: https://edpb.europa.eu/sites/edpb/files/consultation/edpb_guidelines_201903_videosurveillance.pdf. Last Accessed: 2021-04-19.

[7] T. Brewster. Smart Home Surveillance: Governments Tell Google's Nest To Hand Over Data 300 Times. Forbes, (2018). URL: https://www.forbes.com/sites/thomasbrewster/2018/10/13/smart-home-surveillance-governments-tell-googles-nest-to-hand-over-data-300-times/. Last Accessed: 2020-09-08.

[8] L. Bridges. Amazon's Ring is the largest civilian surveillance network the US has ever seen | Lauren Bridges. The Guardian, (2021). URL: http://www.theguardian.com/commentisfree/2021/may/18/amazon-ring-largest-civilian-surveillance-network-us. Last Accessed: 2021-05-18.

[9] D. Cameron. Amazon Is Marketing Face Recognition to Police Departments Partnered With Ring: Report. Gizmodo, (2019). URL: https://gizmodo.com/amazon-is-marketing-face-recognition-to-police-departme-1839073749. Last Accessed: 2020-09-08.

[10] D. Cameron and D. Mehrotra. Ring's Hidden Data Let Us Map Amazon's Sprawling Home Surveillance Network. Gizmodo, (2019). URL: https://gizmodo.com/ring-s-hidden-data-let-us-map-amazons-sprawling-home-su-1840312279. Last Accessed: 2020-01-13.

[11] A. Chattopadhyay and T. E. Boult. PrivacyCam: a Privacy Preserving Camera Using uCLinux on the Blackfin DSP. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8 (2007). 10.1109/CVPR.2007.383413.

[12] C. Cimpanu. Hackers keep dumping Ring credentials online 'for the giggles'. ZDNet, (2019). URL: https://www.zdnet.com/article/hackers-keep-dumping-ring-credentials-online-for-the-giggles/. Last Accessed: 2020-01-13.

[13] D. Deahl. Ring let employees watch customer videos, claim reports. The Verge, (2019). URL: https://www.theverge.com/2019/1/10/18177305/ring-employees-unencrypted-customer-video-amazon. Last Accessed: 2020-09-08.

[14] W. Diffie and M. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, 22 (1976)(6):644–654. 10.1109/TIT.1976.1055638.

[15] B. Dirks et al. Video for Linux Two API Specification (2009). URL: https://www.linuxtv.org/downloads/legacy/video4linux/API/V4L2_API/spec-single/v4l2.html. Last Accessed: 2021-03-21.

[16] K. Florance. About Netflix - How Netflix Works With ISPs Around the Globe to Deliver a Great Viewing Experience. About Netflix, (2016). URL: https://about.netflix.com/en/news/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experience. Last Accessed: 2021-05-28.

[17] Y. Flores. Bad Neighbors? How Amazon's Ring Video Surveillance Could be Undermining Fourth Amendment Protections (2020). URL: https://www.californialawreview.org/amazon-ring-undermining-fourth-amendment/. Last Accessed: 2021-05-18.

[18] R. Gennaro and P. Rohatgi. How to Sign Digital Streams. Information and Computation, 165 (2001)(1):100–116. 10.1006/inco.2000.2916.

[19] A. Greenberg. Two Cases' Lessons: If Cops Don't Know What You Encrypted, They Can't Make You Decrypt It (2021). URL: https://www.forbes.com/sites/andygreenberg/2012/02/24/two-cases-lessons-if-cops-dont-know-what-you-encrypted-they-cant-make-you-decrypt-it/. Last Accessed: 2021-08-02.

[20] M. Guariglia and M. Maas. LAPD Requested Ring Footage of Black Lives Matter Protests. Electronic Frontier Foundation, (2021). URL: https://www.eff.org/deeplinks/2021/02/lapd-requested-ring-footage-black-lives-matter-protests. Last Accessed: 2021-05-11.

[21] I. Haider and B. Rinner. Private Space Monitoring with SoC-Based Smart Cameras. In: 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pages 19–27 (2017). 10.1109/MASS.2017.15.

[22] J. Herrman. Who's Watching Your Porch? The New York Times, (2020). URL: https://www.nytimes.com/2020/01/19/style/ring-video-doorbell-home-security.html. Last Accessed: 2021-04-06.

[23] B. Huseman. Huseman reply to Wyden, Markey, Van Hollen, Coons, Peters letter about Ring's Data Security Practices (2020). URL: https://regmedia.co.uk/2020/01/08/ringsenateresponse.pdf. Last Accessed: 2021-04-06.

[24] J. Katz and Y. Lindell. Introduction to Modern Cryptography, Second Edition (2014). Chapman & Hall/CRC.

[25] C. Keck. Amazon's Ring Security Cameras May Have Let Employees Spy on Customers: Report. Gizmodo, (2019). URL: https://gizmodo.com/amazons-ring-security-cameras-may-have-let-employees-sp-1831658669. Last Accessed: 2020-09-08.

[26] P. Kietzmann, L. Boeckmann, L. Lanzieri, T. C. Schmidt, and M. Wählisch. A Performance Study of Crypto-Hardware in the Low-end IoT. In: Proceedings of the 2021 International Conference on Embedded Wireless Systems and Networks, pages 79–90 (2021). 10.5555/3451271.3451279.

[27] P. Kocher. Complexity and the challenges of securing SoCs. In: 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC), pages 328–331 (2011).

[28] R. Kraus. Ring watched your kids trick or treat and then bragged about it. Mashable, (2019). URL: https://mashable.com/article/ring-halloween-surveillance/. Last Accessed: 2021-05-11.

[29] D. Kravets. Indefinite prison for suspect who won't decrypt hard drives, feds say. Ars Technica, (2016). URL: https://arstechnica.com/tech-policy/2016/05/feds-say-suspect-should-rot-in-prison-for-refusing-to-decrypt-drives/. Last Accessed: 2021-05-31.

[30] C. Lecher. Ring reportedly outed camera owners to police with a heat map. The Verge, (2019). URL: https://www.theverge.com/2019/12/3/20993814/ring-user-location-heat-map-police-privacy-tool-camera-owners. Last Accessed: 2020-09-08.

[31] C. S. Legislature. TITLE 1.81.5. California Consumer Privacy Act of 2018 [1798.100 - 1798.199.100] (2018).

[32] C. S. Legislature. The California Privacy Rights Act of 2020 (2020).

[33] J. McCune, A. Perrig, and M. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In: 2005 IEEE Symposium on Security and Privacy (S P'05), pages 110–124 (2005). 10.1109/SP.2005.19.

[34] A. Ng. Amazon's Ring wanted to use 911 calls to activate its video doorbells. CNET, (2019). URL: https://www.cnet.com/home/smart-home/amazons-ring-wanted-to-use-911-calls-to-activate-its-video-doorbells/. Last Accessed: 2021-05-18.

[35] A. Ng. Ring let police view map of video doorbell installations for over a year. CNET, (2019). URL: https://www.cnet.com/news/ring-gave-police-a-street-level-view-of-where-video-doorbells-were-for-over-a-year/. Last Accessed: 2020-01-13.

[36] J. Nielsen. Why You Only Need to Test with 5 Users. URL: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. Last Accessed: 2021-08-30.
[37] J. Nielsen and T. K. Landauer. A mathematical model of the finding of usability problems (1993). 10.1145/169059.169166.
[38] H. of Commons of Canada. Bill C-11 (First Reading) (2020).
[39] E. Parliament and C. of the European Union. ePrivacy Directive - Directive 2009/136/EC (2009).
[40] E. Parliament and C. of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance) (2016).
[41] K. Paul. Amazon's doorbell camera Ring is working with police – and controlling what they say. The Guardian, (2019). URL: https://www.theguardian.com/technology/2019/aug/29/ring-amazon-police-partnership-social-media-neighbor. Last Accessed: 2020-01-13.
[42] T. C. Project. Guidelines for public video surveillance - A guide to protecting communities and preserving civil liberties (2007). URL: https://archive.constitutionproject.org/pdf/Video_Surveillance_Guidelines_Report_w_Model_Legislation4.pdf. Last Accessed: 2021-04-19.
[43] Ring. Ring Video Doorbells Get 15+ Million Dings This Halloween and Capture Cute Costumes and Fun Pranks. The Ring Blog, (2019). URL: https://blog.ring.com/neighborhood-stories/ring-video-doorbells-get-15-million-dings-this-halloween-and-capture-cute-costumes-and-fun-pranks/. Last Accessed: 2021-05-18.
[44] Ring. Active Agency Map (2021). URL: https://www.google.com/maps/d/viewer?mid=1eYVDPh5itXq5acDT9b0BVeQwmESBa4cB. Last Accessed: 2021-05-18.
[45] R. L. Rivest, A. Shamir, and L. M. Adleman. Cryptographic communications system and method (1983). URL: https://patents.google.com/patent/US4405829/en. Last Accessed: 2021-08-18.
[46] L. Ropek. A Home Security Worker Hacked Into Surveillance Systems to Watch People Have Sex. Gizmodo, (2021). URL: https://gizmodo.com/a-home-security-worker-hacked-into-surveillance-systems-1846111569. Last Accessed: 2021-01-23.
[47] H. Salah, S. Zimmermann, and J. A. Cabrera G. Chapter 5 - Content distribution (2020). https://doi.org/10.1016/B978-0-12-820488-7.00016-5.
[48] G. Scorletti. Traitement du Signal (2016). URL: https://cel.archives-ouvertes.fr/cel-00673929. Last Accessed: 2021-05-11 (Lecture material in French).
[49] S. Ullah, B. Rinner, and L. Marcenaro. Smart cameras with onboard signcryption for securing IoT applications. In: 2017 Global Internet of Things Summit (GIoTS), pages 1–6 (2017). 10.1109/GIOTS.2017.8016279.
[50] T. Winkler and B. Rinner. TrustCAM: Security and Privacy-Protection for an Embedded Smart Camera Based on Trusted Computing. In: 2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance, pages 593–600 (2010). 10.1109/AVSS.2010.38.
[51] T. Winkler and B. Rinner. Secure embedded visual sensing in end-user applications with TrustEYE.M4. In: 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pages 1–6 (2015). 10.1109/ISSNIP.2015.7106934.
[52] R. Wyden, C. Van Hollen, E. Markey, C. Coons, and G. Peters. Wyden, Markey, Van Hollen, Coons, Peters Question Ring's Data Security Practices (2019). URL: https://www.wyden.senate.gov/news/press-releases/wyden-markey-van-hollen-coons-peters-question-rings-data-security-practices. Last Accessed: 2021-04-06.
[53] Wyze. Wyze Cam - Our Story (2018). URL: https://wyze.com/our-story. Last Accessed: 2021-04-06.
[54] H. Yu, J. Lim, K. Kim, and S.-B. Lee. Pinto: Enabling Video Privacy for Commodity IoT Cameras. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 1089–1101 (2018). 10.1145/3243734.3243830.

A Notation

B Experimental Setup

Camera Device. On a *Raspberry Pi 4 Model B Rev 1.1* (Broadcom BCM2711, 1.5 GHz quad-core Cortex-A72 ARM v7 64-bit, 2GB RAM), we used the Video4Linux2 driver [15] to interface with the camera sensor and capture frames that are then encrypted using OpenSSL3.0[4]. The Raspberry Pi Camera Module v2 that we used has a still resolution of 8 Megapixels, a sensor resolution of 3280×2464 pixels, and supports the three following video modes 1080 p/30 fps, 720 p/60 fps, and 480 p/90 fps (respectively video quality and maximum frame rate).
Android Smartphone. We used a *Nokia 4.2* smartphone with Android 10 on which we have installed the implemented application. In this application, we use C native libraries that we have cross-compiled, and C code to download and decrypt the frames. We leveraged the MediaCodec class[5] to perform the encoding and decoding of video files, as well as the Quirc[6] and Bluetooth libraries to perform the pairing.
Cloud Storage. An *AWS EC2 t3.small* instance was used to deploy a Nginx web server. Upon request, we

4 https://www.openssl.org/
5 https://developer.android.com/reference/android/media/MediaCodec
6 https://github.com/dlbeer/quirc

Symbol	Name
$\|\|$	Concatenate function
$\oplus$	XOR function
$AES256Dec$	Symmetrical decryption function
$AES256Enc$	Symmetrical encryption function
C_i	ith cipher frame
$check$	Function to check that corresponding party knows the secret key linked to the public key.
$d_{\mathcal{K}}$	Depth key tree
$\delta_{\mathcal{K}}$	Epoch size
escrow material	Escrow material encrypted with a passphrase
$Extract$	Key extraction function from $\mathcal{K}$
F_i	ith frame
gen	Generate function
h_i	ith hash
h_{PK_d}	Hash of the public key of a delegatee
h_{PK_f}	Hash of the factory-generated public key of the camera
h_{PK_o}	Hash of the public key of the owner
$hash$	Hash function
$HKDF$	Hash-based key derivation function
$HMAC$	Hash-based message authentication code function
$init$	Initialization function
IV_i	ith initialization vector
$\mathcal{K}$	Binary key tree
k_i	Symmetric encryption key for F_i
passphrase	Passphrase encrypting the escrow material
$RandBytes$	Random bytes generation function
RSA	Rivest–Shamir–Adleman encryption and signature scheme
secrets	Secrets sent by the owner to the camera during the initialization
seed key	Encryption material owned by the root node of $\mathcal{K}$
σ	Asymmetrical signature of a block of frames
$Sign$	Asymmetrical signature function
(SK_c, PK_c)	Asymmetric key pair of the camera
(SK_d, PK_d)	Asymmetric key pair of a delegatee
(SK_f, PK_f)	Factory-generated asymmetric key pair of the camera
(SK_o, PK_o)	Asymmetric key pair of the owner
t_i	Timestamp of ith frame
$[t_j, t_{j+1})$	Time interval of the jth epoch
$Verify$	Signature verification function
wifi credentials	Wifi credentials of the owner's network

Table 5. Notation used by CaCTUs cryptographic constructions.

serve the list of encrypted frames that were recorded during the time frame specified in the request.

C Signing Every Frame

We can individually sign each transmitted frame with an adaptation of the one-time signatures (also known as hashed signatures) process described by Gennaro and Rohatgi [18]. This approach is more suitable for live streaming, as the receiving device does not need to verify the entire block of N frames before playing a frame. The idea is to incorporate a one-time public key into each frame payload, which is then used to sign the next frame. During initialization, only the first frame is signed asymmetrically, and subsequent frames are faster to sign and verify. Note, however, that there is a trade-off to this approach, as generation of the one-time keys still incurs a computational cost (though this could be reduced by pre-generating keys). We concretely describe this technique, which is implemented using a hash-based key tree:

1. Each frame F_i is recorded at timestamp t_i.
2. The corresponding symmetric key k_i is extracted from the key rotation scheme $\mathcal{K}$, an initialization vector IV_i is randomly generated, and a pair of one-time keys (PK_{i+1}, SK_{i+1}) is derived.

$$k_i = Extract(\mathcal{K}, i)$$

$$IV_i = RandBytes(16)$$

$$(PK_{i+1}, SK_{i+1}) = OneTimePair()$$

3. Each frame is then symmetrically encrypted (**confidentiality**) into the corresponding cipher C_i using the AES algorithm with a 256-bit key in Galois/Counter Mode (GCM) mode. C_i is then concatenated with IV_i, t_i, and PK_{i+1} to be hashed into h_i (**integrity and freshness**).

$$C_i = AES256Enc(IV_i, k_i, F_i)$$

$$h_i = HMAC(k_i, C_i||IV_i||t_i||PK_{i+1})$$

4. A signature σ_i is then computed (**authenticity and integrity**). For the first signature σ_1, we use the private key SK_c of the camera, while for the other frames we use the one-time keys.

$$\sigma_i = \begin{cases} Sign(SK_c, h_1) & \text{if } i = 1 \\ Sign(SK_i, h_i) & \text{otherwise} \end{cases}$$

5. The encrypted and authenticated frames $\langle C_i, IV_i, t_i, \sigma_i, PK_{i+1} \rangle$ are uploaded to the cloud.

Each user who has access to the correct decryption keys can download these encrypted and authenticated frames

$\langle C_i, IV_i, t_i, \sigma_i, PK_{i+1} \rangle$. The user can verify the integrity, authenticity, and freshness of the data, then decrypt the frames, and rebuild the video.

1. Each cipher C_i, encrypted using the initialization vector IV_i, with timestamp t_i, one-time public key PK_{i+1}, and signature σ_i is downloaded on demand.
2. The corresponding symmetric key material k_{t_i} is extracted from the key rotation scheme $\mathcal{K}$. The hash h_i of each cipher C_i is computed.

$$k_i = Extract(\mathcal{K}, i)$$

$$h_i = HMAC(k_i, C_i || IV_i || t_i || PK_{i+1})$$

3. The signature σ_i is verified with the public key PK_c of the camera if this is the first frame or with the one-time public key PK_i (**authenticity, integrity, and freshness**).

$$1 \stackrel{?}{=} \begin{cases} Verify(PK_c, \sigma_1, h_1) & \text{if } i = 1 \\ Verify(PK_i, \sigma_i, h_i) & \text{otherwise} \end{cases}$$

4. If the signature is correct, each cipher C_i is then symmetrically decrypted into the corresponding frame F_i to rebuild the video (**confidentiality**).

$$F_i = AES256Dec(IV_i, k_i, C_i)$$

D Protocol Details

D.1 Delegation

Table 6 shows the details of the operations performed during the delegation protocol between the owner and a delegatee. The process is very similar to the one done during initialization with the camera:

1. The pair of asymmetric keys (SK_o, PK_o) is present on the smartphone of the owner, when the delegation process starts the smartphone application of the owner displays a QR code in which the hash of the owner's public key h_{PK_o} is embedded. The delegatee's smartphone scans this QR code and stores its content.
2. The delegatee and the owner's smartphone connect through Bluetooth and the owner sends its public key PK_o to the delegatee, who computes the hash of the owner's public key h'_o and checks that it matches the hash retrieved from the QR code.
3. If they match, the delegatee's smartphone generates its own asymmetric pair of keys (SK_d, PK_d) and sends its public key PK_d through Bluetooth, the owner computes the hash h'_d of the delegatee's public key.

	Owner	👁	ᛒ	Delegatee
1		$\stackrel{h_{PK_o}}{\longrightarrow}$		
2			$\stackrel{PK_o}{\longrightarrow}$	$h'_o = hash(PK_o)$ $h'_o \stackrel{?}{=} h_{PK_o}$
3	$h'_d = hash(PK_d)$		$\stackrel{PK_d}{\longleftarrow}$	$gen(SK_d, PK_d)$
4	$h'_d \stackrel{?}{=} h_{PK_d}$	$\stackrel{h_{PK_d}}{\longleftarrow}$		
5	$DH(SK_o, PK_d)$		$\stackrel{verif}{\longleftrightarrow}$	$DH(SK_d, PK_o)$
6	keys $\{k_i\}$		$\stackrel{\{k_i\}}{\underset{(RSA)}{\longrightarrow}}$	$\mathcal{K} = init(\{k_i\})$

Table 6. Protocol followed by the smartphone application of the owner and the delegatee during delegation, 👁 and ᛒ respectively correspond to what is obtained through the visual and Bluetooth channels.

4. The delegatee points their smartphone's screen at the owner. On the screen is displayed the QR code with the hash of the delegatee's public key. The owner retrieves the content from the QR code and checks that it matches h'_d.
5. If the key hashes match, both devices now verify that the other device knows the secret key corresponding to the public key that they advertised earlier. This is can be done for instance by applying the Diffie-Hellman key exchange to compute their shared secret and then by exchanging a series of encrypted messages where both parties prove their knowledge.
6. Then, the owner extracts the keys $\{k_i\}$ and share them in an encrypted and authenticated way using RSA with the delegatee to give them access to the corresponding videos. Note that this last step does not necessary need to be done through Bluetooth and could be done over the Internet too without undermining the security of the delegation protocol.

D.2 Deletion and Factory Reset

When the owner decides to delete some videos, they delete the corresponding decryption keys in the key tree $\mathcal{K}$ they have access to. They also need to update accordingly the *key material* inside the *escrow material* saved on the camera. Similarly to the last step of the delegation protocol, updating the *key material* on the camera

Owner	ᛒ or 🌐	Camera
		Verify request?
1	$\overset{request\ key\ material\ (RSA)}{\longrightarrow}$	
		Perform operation

Table 7. Protocol followed by the owner and the camera during deletion. ᛒ and 🌐 correspond to what is obtained through Bluetooth or over another channel such as the Internet.

can be done through Bluetooth or remotely over the Internet. Table 7 shows the details of this part of the protocol:

1. The owner just sends the timestamped and authenticated request as well as the updated *key material* encrypted and authenticated with RSA (recall that the owner and the camera have shared their asymmetric public keys during initialization). At reception, the camera verifies the authenticity of the update order and performs the operation if it is valid.

Likewise, to factory reset the camera, the owner sends the request to the camera (through Bluetooth or remotely over the Internet), timestamped and authenticated with the secret key SK_o of the owner to verify the legitimacy of the request. Then, both the camera and owner's smartphone delete the key tree $\mathcal{K}$ they have access to, returning both devices to an uninitialized state.

D.3 Access Recovery

New smartphone	ᛒ	Camera
1	$\overset{request}{\longrightarrow}$	
2 **Passphrase known?**	$\overset{escrow\ material}{\longleftarrow}$	
3 (SK_o, PK_o), PK_c, and $\mathcal{K}$ retrieved		

Table 8. Protocol followed by the camera and the smartphone application of the user trying to recover access. ᛒ correspond to what is obtained through the Bluetooth channel.

Table 8 shows the steps executed when someone tries to recover access to the system:

1. The owner uses their new smartphone to open a Bluetooth connection to the camera and request the escrow material.
2. The camera sends back the escrow material. Recall that this escrow material is encrypted with the recovery passphrase that was displayed to the owner during initialization.
3. If the owner knows the recovery passphrase, they are able to recover access to the asymmetric key pair (SK_o, PK_o) of the owner, to the public key PK_c of the camera, and to the *key material* necessary to build the key tree $\mathcal{K}$.

E Functional User Evaluation Protocol

All institutional requirements were met for this functional user evaluation of `CaCTUs`. We obtained approval from the Institutional Review Board (IRB) of our university and a consent form was signed by the participants at the beginning of their session. We also tested the protocol with coworkers and collaborators beforehand to identify possible limitations.

All the material was provided to the participants that were guided by a researcher through the different tasks to perform. We introduced each task with a real-life scenario to help the participants behave as if they were using the system in their real life. To collect feedback, we observed a talk aloud process asking the participants to express aloud what they are doing or looking for while performing the task, allowing us to better identify potential issues in the system. Between each task, we also asked specific questions about the process that had just been completed.

E.1 Before Each Session

Before each session, we verified that all the material needed for the session was provided, was working as expected, and was in its initial state:

1. **Camera Device:** 1 *Raspberry Pi 4 Model B 2GB*, 1 camera sensor, 1 case, 1 power supply, 1 micro SD card with OS image flashed on it and our software installed.
2. **Android Smartphone Devices:** 3 *Nokia 4.2* with our application pre-installed, wifi connection configured, location enabled (to enable discovery of nearby Bluetooth devices), Bluetooth disabled with no prior device paired, and with no saved media on the smartphone.

3. **Other:** Piece of paper and pen provided (to write down the recovery passphrase).

E.2 During Each Session

As participants are being recorded using the system and are recording their own video, we first asked participants for explicit consent to be recorded. We then proceeded by asking some preliminary questions about their background and their experience with smart camera systems to get the discussion started.

Next, we explained that the objective of the session was to evaluate the functional usability of the smart camera system that we designed. Details about privacy violations in current available systems were briefly described to help the participants understand the motivation of the project. Then, we presented how the system we implemented was giving back full control to the users and enforcing their privacy. These explanations were very high level, since our goal was to make sure that the participants understood that the video frames were encrypted before leaving the camera device and being uploaded to cloud storage, that they were then decrypted on the smartphone side, and that the encryption and decryption keys were only known by the devices that the user would allow to share access with.

After describing how the rest of the session was going to take place, each task was introduced by a scenario and feedback obtained through the talk aloud process as well as follow up questions. We verified that the participants were either familiar with Android or we showed them how to navigate between applications on Android.

Finally, we asked each participant to perform the different tasks in the following order.

E.2.1 Initialization

Scenario: You want to secure your home, so you just bought this new smart camera system online. You received the package with the camera and just downloaded the application on your smartphone. Go ahead with the rest of the configuration.

Questions:

- What did you like about the initialization?
- What did you dislike about the initialization?
- Any further comments?

E.2.2 System Usage

Scenario: You have your new smart camera all set up, so now you want to be able to see what is happening inside/outside of your home. For that you open the application to view the live streaming and access the different functionalities of the application.

Questions:

1. How do you feel about the quality of the video streaming?
2. What is your opinion about the following statements (Likert scale: strongly disagree, disagree, neutral, agree, strongly agree)?
   (a) I am happy with the image quality of the streaming.
   (b) I am happy with the latency of the streaming.
   (c) I am happy with the frame rate of the streaming.
3. What did you like about the usage of the system/application?
4. What did you dislike about the usage of the system/application?
5. Any further comments?

E.2.3 Delegation

Scenario: You want to give access to someone else to the live streaming of your camera, as you are going on vacation abroad. They have downloaded the application on their phone, you need to add them as a new delegatee on your application.

Questions:

1. What did you like about the delegation process?
2. What did you dislike about the delegation process?
3. What is your opinion about the following statement: It is easy to add a new delegatee (Likert scale: strongly disagree, disagree, neutral, agree, strongly agree)?
4. Would you like to see any change in the delegation process or the options for the access control?
5. Who would you typically add as a delegatee and for how long?
6. What do you think of the granularity of the delegation control?
7. Any further comments?

E.2.4 Access Recovery

Scenario: Unfortunately, on your way back home during the layover, you lost your smartphone, which was the device you used to access your camera system, and when you come back home, you figured out that someone has broken in during your holidays and robbed you. You buy a new smartphone, install back the application on it, and want to recover access to your system to see what happened. Luckily, you wrote down your recovery passphrase.

Questions:

1. What did you like about the recovery process?
2. What did you dislike about the recovery process?
3. What is your opinion about the following statement: It is important for me to have a recovery process. (Likert scale: strongly disagree, disagree, neutral, agree, strongly agree)?
4. Any further comments?

E.2.5 Factory Reset

Scenario: Finally, you want to reconfigure your system as you lost your smartphone, but before you want to make sure to factory reset the system.

Questions:

1. What did you like about the factory reset process?
2. What did you dislike about the factory reset process?
3. What is your opinion about the following statement: It is important for me to have a factory reset process. (Likert scale: strongly disagree, disagree, neutral, agree, strongly agree)?
4. Any further comments?

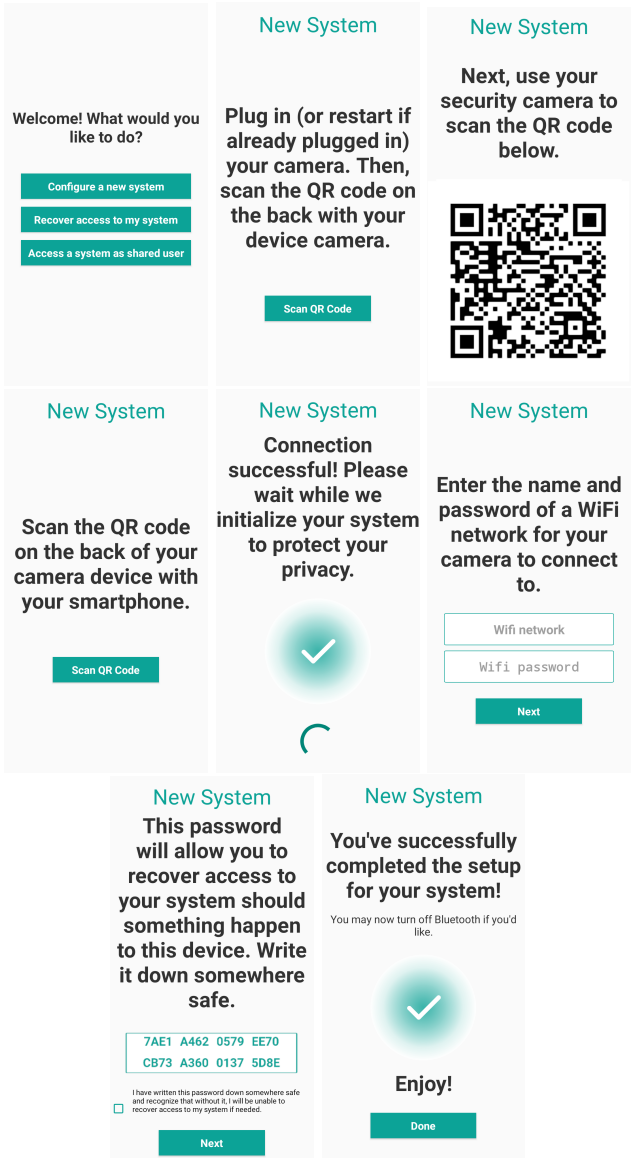

Fig. 7. Screenshots of the CaCTUs smartphone application of the owner during initialization of the system.

E.3 After Each Session

After each session, we made sure that every device was reset and back into its initial state, as if the session had not occurred.

F Storyboard

Following is the storyboard of the CaCTUs's smartphone application.

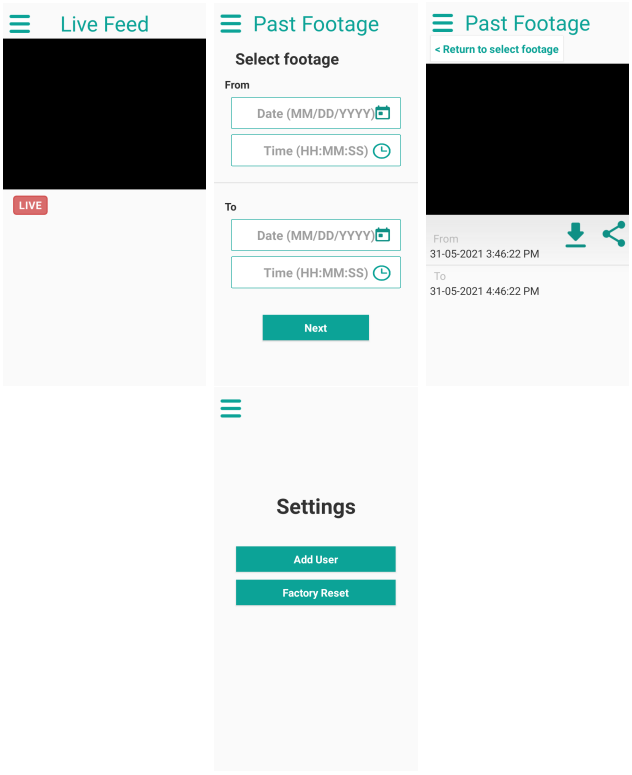

Fig. 8. Screenshots of the CaCTUs smartphone application: home page (live feed), access to past footage, and settings.

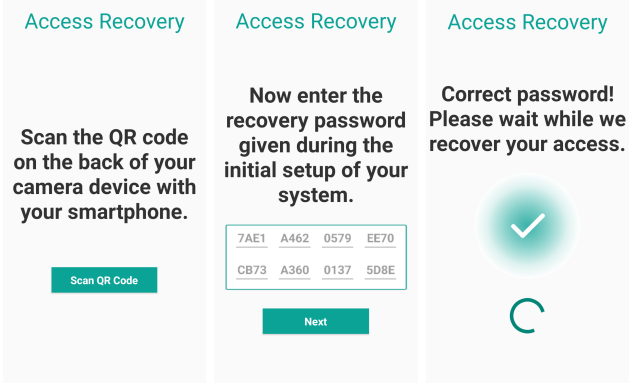

Fig. 9. Screenshots of the CaCTUs smartphone application of the owner during access recovery.

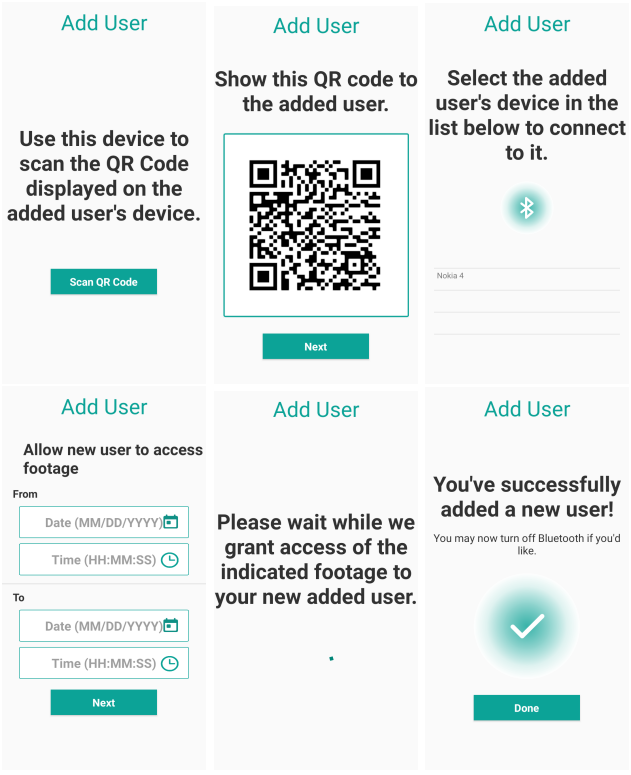

Fig. 10. Screenshots of the CaCTUs smartphone application of the owner during delegation.

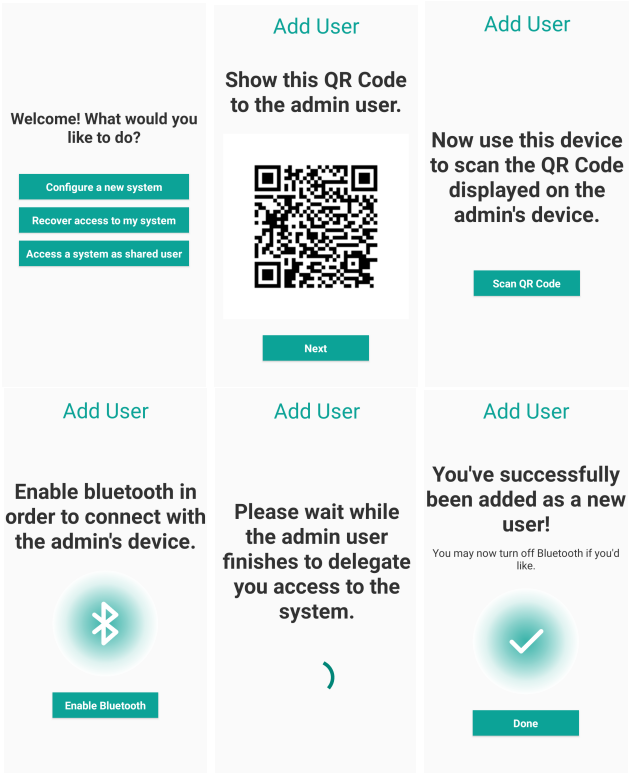

Fig. 11. Screenshots of the CaCTUs smartphone application of the delegatee during delegation.