Blind Brother: Attribute-Based Selective Video Encryption

Eugene Frimpong
eugene.frimpong@tuni.fi
Tampere University
Tampere, Finland

Bin Liu*
bin.liu@tuni.fi
Tampere University
Tampere, Finland

Camille Nuoskala
camille.nuoskala@tuni.fi
Tampere University
Tampere, Finland

Antonis Michalas†
antonios.michalas@tuni.fi
Tampere University
Tampere, Finland

ABSTRACT

The emergence of video streams as a primary medium for communication and the demand for high-quality video sharing over the internet have given rise to several security and privacy issues, such as unauthorized access and data breaches. To address these limitations, various Selective Video Encryption (SVE) schemes have been proposed, which encrypt specific portions of a video while leaving others unencrypted. The SVE approach balances security and usability, granting unauthorized users access to certain parts while encrypting sensitive content. However, existing SVE schemes adopt an all-or-nothing coarse-grain encryption approach, where a user with a decryption key can access all the contents of a given video stream. This paper proposes and designs a fine-grained access control-based selective video encryption scheme, ABSVE, and a use-case protocol called `Blind Brother`. Our scheme encrypts different identified Regions of Interest (ROI) with a unique symmetric key and applies a Ciphertext Policy Attribute Based Encryption (CP-ABE) scheme to tie these keys to specific access policies. This method provides multiple access levels for a single encrypted video stream. Crucially, we provide a formal syntax and security definitions for ABSVE, allowing for rigorous security analysis of this and similar schemes – which is absent in prior works. Finally, we provide an implementation and evaluation of our protocol in the Kvazaar HEVC encoder. Overall, our constructions enhance security and privacy while allowing controlled access to video content and achieve comparable efficiency to compression without encryption.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Security and privacy** → **Key management**; *Access control*; *Privacy-preserving protocols.*

KEYWORDS

Access Control, Attribute-Based Encryption, HEVC, Selective Video Encryption

1 INTRODUCTION

In the modern digital era, visual communication has become a cornerstone of human interaction, with video streams emerging as the go-to medium for sharing information, expressing creativity, and fostering connectivity [18]. With this increased usage and consumption of video streams, the demand for high-quality video sharing over the internet has skyrocketed. This high demand led to the introduction of the High-Efficiency Video Coding (HEVC) [17, 26] standard to support higher video quality and more efficient video compression techniques. Alongside the myriad benefits of video sharing and using the HEVC standard, security and privacy concerns exist that may undermine individual rights, compromise sensitive data, and facilitate malicious activities. One of the most significant security concerns associated with videos is the risk of unauthorized access and data breaches [10, 22]. In a world where video streams traverse numerous networks and devices, malicious actors can intercept video transmissions, gain unauthorized entry to video databases, and exploit sensitive information for nefarious purposes. Alternatively, unauthorized people in an organization may gain access to the contents of a video stream that they otherwise should not have access to. These concerns severely threaten individuals, organizations, and even governments, jeopardizing personal privacy and sometimes national security.

A straightforward approach to address the privacy concerns related to video sharing is to encrypt the video bitstream, popularly referred to as Naive Encryption Algorithm (NEA) [1, 22]. However, this has been proven to be impractical, as it proceeds to encrypt and decrypt the entire bitstream as textual data. More specifically, NEA is incompatible with various video codecs, formats, or streaming protocols, as video streams must be decrypted before being viewed, which makes the encrypted video useless to a user who does not possess a decryption key. Additionally, due to the large size of video streams, encryption and decryption of the entire bitstream requires significant computational resources, which makes NEA an inefficient technique. To this end, several Selective Video Encryption (SVE) [6, 11, 12, 14, 27, 31] schemes have been proposed in recent years. In an SVE scheme, specific portions of a video stream are encrypted while leaving other portions of the bitstream unencrypted. This approach provides fine-grained control over which parts of the video are protected and which parts remain accessible without decryption (i.e., codec compliance [22] – the SVE encrypted video can be played without being first decrypted). The primary goal of SVE is to strike a balance between security and usability, allowing unauthorized users to access certain parts of the video while keeping sensitive or private content encrypted and protected. Standard cryptographic techniques, such as basic symmetric or asymmetric encryption methods, can be used to implement SVE. However, these require careful consideration of the encryption process, key

*Also with grchain.io.
†Also with RISE Research Institutes of Sweden.

management, and the design of mechanisms to ensure that access to different parts of the video is controlled appropriately. Furthermore, SVE schemes often utilize techniques provided by the encoder (e.g. HEVC tiles [24]) that segment a given video frame into different units, such as blocks, tiles or slices[1], and then apply encryption selectively to these units.

The technique of splitting a video frame into different units introduced the possibility of encrypting specific regions of interest (ROI) in a video stream [12, 27]. For example, given a video of traffic surveillance, SVE can be used to encrypt all identified faces to ensure unauthorized parties with access to the encrypted video can still play the video without seeing citizens' faces. However, to the best of our knowledge, existing SVE schemes adopt an *all-or-nothing* course-grain approach when encrypting the identified ROIs. That is, one key stream is used to encrypt all ROIs in a given video stream, and once a user has access to this key, they can decrypt all parts of the video.

However, existing SVE schemes are directly constructed using specific cryptographic primitives without establishing formal syntax and security definitions for SVE itself. As a result, the security guarantees provided by those underlying primitives do not automatically extend to the SVE scheme and the protocols that employ them. The lack of formal definitions means that current SVE constructions lack rigorous and formal security proofs, undermining their reliability as a security-critical component. Without such formal definitions, it is impossible to rigorously reason about the security of these schemes. To address this gap and meet the need for fine-grained access control, we introduce the Attribute-based Selective Video Encryption (ABSVE) scheme by formally defining its syntax and security properties. ABSVE extends SVE by enabling policy-based access control over video contents. This enhancement provides greater flexibility in controlling access to controlling access to encrypted video content based on predefined attributes and policies.

ABSVE enables fine-grained access control by encrypting each identified ROI in a video stream with a unique symmetric key. Subsequently, the encryption keys are encrypted with a Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [2, 4, 8] scheme, which links the keys to a specific policy. Using this approach, we consider a use case where every encryption key corresponding to a different ROI is encrypted under a different policy, thereby ensuring multiple access levels to the contents of a single video stream. Based on ABSVE, we present `Blind Brother`, a fine-grained access control-based video encryption protocol that enforces access control over video streams among system participants. Finally, we demonstrate that our protocol is provably secure with respect to the formal security definitions we proposed for fine-grained access control-based video encryption protocols.

Contributions: The core contribution of this work lies in demonstrating the implementation and feasibility of combining CP-ABE with ROI encryption on video files, highlighting the practicality and achievability of this integration. To support this, our sub-contributions are summarized as follows:

- **C1.** We establish the first formal syntax and security definitions for ABSVE, an extension of SVE designed to enable policy-based access control over video content.
- **C2.** We design an ABSVE scheme, enabling fine-grained access control by leveraging a CP-ABE scheme to associate symmetric keys for encrypting distinct regions of a given video stream to an access policy.
- **C3.** Subsequently, we construct `Blind Brother` – a use-case protocol based on the proposed scheme that demonstrates the applicability of our work in a real-world application. The security of the scheme and protocol is extensively analyzed and proved.
- **C4.** Finally, we implement and evaluate the performance of our scheme in the real-time Kvazaar HEVC encoder [19, 28].

2 MOTIVATION AND APPLICATION DOMAIN

Video stream encryption with object-specific keys has a broad application spectrum ranging from the protection of sensitive information in public surveillance systems to data-safeguarding in both commercial and private environments. The following fields could become possible implementation examples: *(a)* smart cities with the aim of anonymizing individuals in real-time, *(b)* retail settings in order to preserve customer privacy while monitoring store activities, *(c)* in healthcare facilities to ensure patient interactions captured by security cameras remain confidential. In this section we will focus on an example demonstrating how the proposed system can be protective of individual privacy on a daily basis while enabling necessary interventions in exceptional cases.

- **Privacy Protection for Everyday Life:** Helen, a 52-year-old history professor at the University of Athens, follows a predictable routine: she leaves her home every morning at 8:30 AM and cycles to her university office. Her movements could be captured by cameras along her route. This would reveal sensitive personal information, such as her home address, workplace, and daily habits. In combination with facial recognition technologies, her identity could be exposed and result to a breach of her privacy, though Helen is an ordinary citizen with no connection to criminal activities.
- **Crime Investigation Scenario:** One evening, while cycling back home, Helen is assaulted and her bag is stolen. The police consults footage from cameras along her route and identifies a suspect.

Societal Paradox

These examples underline a critical societal paradox with societal repercussions: while surveillance systems are invaluable for crime prevention and resolution, continuous recording of daily life causes intrusions into personal privacy. This calls for innovative approaches that balance these competing priorities.

This challenge can be addressed by our proposed system via selective encryption of sensitive video content. If, for example, we take the second scenario, faces and other sensitive elements in the

[1] A slice is a data structure (an entire picture or region of a picture) that can be decoded independently from other slices of the same picture, in terms of entropy coding, signal prediction, and residual signal reconstruction [26].

footage would remain encrypted unless the authorities detected a possible suspect. In this case, the system would only decrypt the necessary portions of the data, such as the suspect's face. As a result, the system protects individual privacy during routine operations, while allowing room for effective law enforcement when necessary.

Through advanced encryption technologies, object detection algorithms, and selective decryption protocols, the system initiates privacy-preserving surveillance solutions that safeguard individual rights and reinforce safety in public spaces, thus respecting and protecting democratic values in modern society.

3 BACKGROUND

This section is divided into two parts. First, we present the fundamental video concepts and frameworks utilized in this paper. Next, we provide definitions for the core cryptographic components of our design.

3.1 HEVC

HEVC is a video project standard developed by the ITU-T Video Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) [26]. The standard's primary aim is to provide a generic syntax that developers can follow when building video compression applications. It is important to mention that the standardized processes of the HEVC are limited to bitstream structure, syntax, constraints on the bitstream, and bitstream mapping for decoded pictures. The rest of this section provides an overview of the various components of the HEVC encoding process.

HEVC Motion Prediction: HEVC utilizes advanced motion prediction to reduce a given video sequence's spatial and temporal redundancy during video encoding and decoding. HEVC utilizes a hybrid prediction approach for the video coding layer – *inter/intra picture prediction*. Intra-prediction is used for encoding frames that do not rely on any previously encoded frames. It exploits the spatial redundancy within a frame by predicting pixel values from neighbouring pixels. HEVC supports various intra-prediction modes, such as vertical, horizontal, DC, and angular modes, thus allowing flexible prediction based on the characteristics of the video content. On the other hand, inter-prediction is used for encoding frames by exploiting temporal redundancy between the current frame and previously encoded frames. It predicts the current frame by motion-compensating (MC) from reference frames. HEVC uses block-based motion estimation to search for the best matching block in reference frames and then applies motion compensation to generate the prediction.

Motion Prediction in Action: In a typical HEVC encoder, the encoding algorithm follows a specific procedure. Firstly, each input video frame is split into blocks (the specification of the block partitioning is shared with the decoder). Subsequently, when coding the first frame of the input video or any frame at a clean random access point, only intra-picture prediction is used. For all other frames in the video sequence, inter-picture prediction is used for most of the blocks. To encode with inter-picture prediction, the algorithm selects motion data and a motion vector (MV) from a chosen reference frame to predict the samples of each block. MVs are used to estimate the motion between frames by describing the spatial shift of blocks. HEVC divides the frames into smaller blocks, typically ranging from 4x4 to 64x64 pixels, and performs motion estimation to find the best matching block in the reference frames. The MV value represents the displacement in horizontal and vertical directions needed to align the current block with its best match in the reference frames. MC is performed once the MVs are obtained to generate the predicted frame.

MV Differences and Signs: Components of the MV coding process that help efficiently represent and transmit motion information between frames. They are part of the process of encoding and decoding motion vectors in inter-frame prediction. MV differences, also known as motion vector differentials, refer to the differences between the motion vectors of neighbouring blocks or partitions. Instead of directly encoding the absolute motion vectors for each block, HEVC encodes the differences between the motion vectors of the current block and its predicted motion vectors. This allows for a more efficient representation of motion vectors, as the differences are often smaller values compared to absolute motion vectors. MV signs, on the other hand, represent the direction of motion for a given MV. In inter-frame prediction, a motion vector can be positive, indicating motion in a specific direction, or negative, indicating motion in the opposite direction. HEVC employs differential coding to represent the signs of motion vectors. The sign of a motion vector is predicted based on the signs of neighbouring motion vectors or partitions, and the difference in signs is encoded using fewer bits compared to encoding the absolute sign for each motion vector. By utilizing MV differences and MV signs, HEVC achieves to efficiently code motion vectors and thus reduce bit rates for motion information transmission.

HEVC Transform Coding: HEVC utilizes a block-based coding technique to convert blocks of pixels from the spatial domain to the frequency domain. The most commonly used technique in HEVC is the High-Efficiency Transform (HEVC-Transform), based on the discrete cosine transform (DCT). The transform process decorrelates the pixel values within a block and concentrates the energy in fewer coefficients, enabling efficient representation and compression of the video data. Transform coefficients (TCs) are numerical values that represent the frequency components of a block after it has undergone a transformation. After successful transformation, the TCs are quantized to reduce their precision and eliminate perceptually insignificant information. This quantization process introduces a trade-off between coding efficiency and video quality. TCs that contribute less to visual quality are assigned higher quantization values and may be discarded or represented with fewer bits during encoding, resulting in higher compression.

HEVC Tiles: The HEVC standard introduced the **tile** feature to support parallel processing and packetization. In a nutshell, tiles are independently decodable frame regions encoded with some shared header information [24]. Coding dependencies do not cross the tile boundary when tiles are enabled. These dependencies include MV prediction, intra-picture prediction, and context selection. Controlling coding dependencies within a particular tile ensures that a decoder can process multiple tiles in parallel, greatly improving efficiency. Additionally, the partitioning afforded by tiles can be used to facilitate the identification and independent processing of specific ROIs within a video. Figure 1 shows a video frame partitioned into 15 tiles (red rectangles).

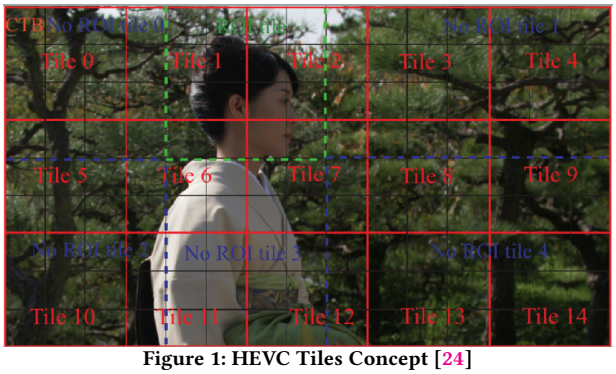

Figure 1: HEVC Tiles Concept [24]

Kvazaar: This is a cross-platform HEVC encoder developed and maintained by a large community of researchers [19, 28]. The Kvazaar project is written in C and supports HEVC Main, Main Still Picture, and Main 10 profiles for an 8-bit 4:2:0 progressive video. The core advantages of the project include coding efficiency close to the reference software implementation [17], real-time coding speed, optimized computation and memory resources, easy portability to various platforms, and a well-documented source code.

3.2 Core Cryptographic Building Blocks

In this section we provide the main cryptographic building blocks utilized in this work. More precisely, we give definitions for a symmetric encryption and for a CP-ABE scheme followed by a brief discussion of the concept of hybrid cryptosystems.

Definition 3.1 (Private Key Encryption (SKE)). A private-key encryption scheme E is a tuple of three algorithms E = (KeyGen, Enc, Dec) such that:

- **KeyGen** : The Key Generation is a probabilistic algorithm that takes as input a security parameter λ, and outputs a private $\mathsf{K} \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- **Enc** : Encryption is a possibly probabilistic algorithm that takes as input a private key K and a message $\mathsf{m} \in \mathcal{M}$ and outputs a ciphertext $c \leftarrow \mathsf{Enc}(\mathsf{K}, \mathsf{m})$.
- **Dec** : Decryption is a deterministic algorithm that takes as input a secret key K and a cipertext c and outputs a message $m \leftarrow \mathsf{Dec}(\mathsf{K}, \mathsf{c})$.

CP-ABE is an encryption scheme that offers fine-grained access control over encrypted data [2, 4]. In CP-ABE, the data and the access policies are associated with a concrete set of attributes $\mathcal{Y}$. A user's access to the encrypted data depends on whether their attributes satisfy the access policy P associated with the ciphertext. This attribute-based approach allows for flexible access control, where access can be granted based on various combinations of attributes. It is particularly useful when data is shared among multiple users with different access privileges. In CP-ABE, every user has a secret key generated based on a set of attributes. Each ciphertext is associated with an access policy defined in terms of attributes. The ciphertext can only be decrypted by a user if their attribute set satisfies the policy associated with the ciphertext (i.e., $P(\mathcal{Y}) = \mathsf{True}$). For the rest of this paper, we refer to the space of attributes as $\Omega = \{a_1, \ldots, a_n\}$, while the space of policies will be denoted as $\mathcal{P} = \{P_1, \ldots, P_m\}$.

Definition 3.2 (Ciphertext-Policy ABE). A CP-ABE scheme is a tuple of the following four algorithms:

- CPABE.Setup: This is a probabilistic algorithm that takes as input a security parameter λ and outputs a master public key MPK and a master secret key MSK. We denote this by $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$.
- CPABE.Gen: This is a probabilistic algorithm that takes as input a master secret key MSK, a set of attributes $\mathcal{Y} \subseteq \Omega$ and the unique identifier of a user u_i, and outputs a decryption key bound to the user and their list of attributes. We denote this by $\mathsf{dk_i} \leftarrow \mathsf{Gen}(\mathsf{MSK}, \mathcal{Y}, u_i)$.
- CPABE.Enc: A probabilistic algorithm that takes as input a master public key MPK, a message m, and a policy $P \in \mathcal{P}$. The algorithm outputs a ciphertext c_m associated with the policy P on a successful run. We denote this by $c_m \leftarrow \mathsf{Enc}(\mathsf{MPK}, m, P)$.
- CPABE.Dec: A deterministic algorithm that takes as input a user's secret key and ciphertext and outputs the original message m *iff* the set of attributes $\mathcal{Y}$ associated with the underlying secret key satisfies the policy P associated with c_m. We denote this by $m \leftarrow \mathsf{Dec}(\mathsf{dk_i}, c_m)$.

Hybrid Cryptosystems: In cryptography, a hybrid cryptosystem combines the convenience of a public-key cryptosystem with the efficiency of a symmetric-key cryptosystem. Public-key cryptosystems are advantageous because they do not require the sender and receiver to share a common secret for secure communication. Examples include the TLS protocol [9], the SSH protocol [30], Kerberos [25], and many others.

Our work adopts a similar approach, integrating SKE with CP-ABE, as demonstrated in prior research (e.g., [23]). The concept is straightforward: symmetric encryption is employed to encrypt specific ROIs in a video, while CP-ABE is used to distribute the symmetric keys to users who meet a defined policy. These users can then use the corresponding symmetric keys to decrypt certain objects within the video.

4 RELATED WORKS

Several works have explored the concept of SVE in the context of video streams [6, 7, 11, 12, 22, 27, 29, 31]. For example, in [12], authors proposed a selective encryption solution to encrypt ROIs in specific tile based on the HEVC standard using AES symmetric encryption. Their work implemented the encryption at the Context-Adaptve Binary Arithmetic Coding (CABAC) binstring level and focused on encrypting a set of HEVC parameters such as the MV difference, MV signs, TC, and TC signs. Subsequently, authors in [27] built upon the work proposed in [12] by extending the encryption to the luma and chroma Intra Prediction Modes (IPMs). In both papers, the authors prevented the propagation of encryption outside the identified tiles by restricting the MVs of non-ROI regions inside the background region. Another approach to SVE was proposed in [29], combining encryption with data hiding. In this work, authors encrypted parameters such as MV signs and IPMs. Meanwhile, they used TC levels for data embedding. They also expanded

the encryption space within CABAC encoded streams, including IPMs, to enhance video reconstruction, and employed a coefficient modification technique for data hiding, ensuring format compatibility and visual quality. More recently, authors in [6] proposed a randomly selective encryption scheme which utilizes the RC4 algorithm and focuses on expanding the encryption positions in the video stream as well as randomizing the positions. Authors in [7] proposed a robust SVE scheme that focuses on synchronizing slices to ensure decryption is possible in case of packet loss. In this work, each slice of the I/P/B frame is independently encrypted based on a pseudorandom binary sequence (PRBS) generated by an RC4 stream cipher. The PRBS is designed to be related to the coding parameters of each slice. Not all the schemes target the HEVC standard, although it is considered the most popular and mature standard. In [11], authors explore SVE in the Versatile Video Coding (VVC) Standard [5]. The VVC standard is the new generation of video coding standards developed by the MPEG and ITU/VCEG group. Authors in [11] extend SVE to VVC by proposing a scheme which determines encryptable bins with the TCs and encrypts a set of VVC elements much like [12] and [27].

In the works mentioned above, the central objective has been to devise an SVE approach that seamlessly integrates with the encoding process. The aim is to strike a balance between achieving efficiency without overwhelming computational resources while ensuring that the resulting visual quality remains within acceptable bounds. However, amid these pursuits, the critical aspects of key distribution and access control have regrettably been relegated to the background. In our work, we address the aforementioned gap and extend the capabilities of SVE to encompass a finer level of access control. Our scheme takes a more comprehensive approach, leveraging multiple encryption keys corresponding to the identified ROIs. `Blind Brother` provides a secure and controlled method of granting access to the encryption keys, intricately tied to a user's specific access rights, thus ensuring a robust and tailored security framework. By merging encryption efficiency, visual quality preservation, and fine-grain access control, we endeavour to establish a more holistic foundation for video data protection.

5 SYSTEM MODEL

In this section, we provide a brief description of the system model considered for `Blind Brother`. Our setup consists of four entities: *(i)* Administrator (AD), *(ii)* Edge Devices ($\mathcal{D}$), *(iii)* Cloud Service Provider (CSP), and *(iv)* Users ($\mathcal{U}$).

- **Administrator**: We assume the existence of a trusted administrator **AD**, who manages and is responsible for a set of edge devices in a specific environment. **AD** is responsible for generating the CP-ABE master secret and public key as well as the CP-ABE decryption key for a user.
- **Edge Devices**: Let $\mathcal{D} = \{d_1, \ldots, d_y\}$ be a set of edge devices deployed in an environment to capture video streams V. Each device encrypts the captured video streams and uploads them to cloud storage.
- **Cloud Service Provider**: We assume the existence of a cloud service provider **CSP**, an abstract external storage platform whose primary responsibility is the storage of encrypted videos received from each device d_i.
- **Users**: Let $\mathcal{U} = \{u_1, \ldots, u_z\}$ be the set of registered users in our environment who wish to view parts of an encrypted video uploaded by a device d_i. A user u_j is considered to be registered if they possess a CPABE decryption key dk_j generated based on their set of attributes $\mathcal{Y}_j = (a_1, \ldots, a_t)$.

TEE – To Use or Not to Use?

Despite criticism from the cryptographic community—primarily due to the numerous attacks that have been discovered—TEE remains an active area of research, ensuring continuous improvements. Furthermore, although in crypto community it is preferred to place trust in mathematics than in the hardware (a perspective with which the authors of this paper agree), we believe that using a TEE in a scenario like the one presented in this paper is a well-suited approach. Specifically, the authority responsible for generating ABE keys could operate within an isolated environment, enhancing both security and functionality. Due to space constraints and to keep our description simpler, we have omitted the inclusion of a TEE in our discussion. However, we encourage the reader to refer to the architectures and proofs in [3].

6 SELECTIVE VIDEO ENCRYPTION

This section presents the fundamental concepts underlying our fine-grained SVE scheme for the HEVC standard. The scheme is using HEVC standard and extends it by incorporating an encryption mechanism with access control capabilities. The encryption process focuses on encrypting the MV differences, MV signs, TC, TC signs, and Luma and Chroma Intra Prediction Modes (IPM) HEVC parameters. These parameters have been proven to be the ideal encryption parameters to enable successful decryption and decoding without increasing the bitrate [7, 11, 31]. Additionally, to identify the various ROIs, we utilize `Dlib`[2] – an open-source Object Identifier (OI) through which we identify and track objects in a frame.

In the following subsections, we first provide a detailed description of how we combine the HEVC tile concept with the OI to achieve our goals. Subsequently, we present the formal definition of the ABSVE and its security definitions.

6.1 Tile Concept and Object Identification

Using the HEVC tile concept briefly described in section 3, we assume that each frame of an input video stream is partitioned into a number of tiles. For this work, let's consider a 6×3 matrix of tiles (i.e. 18 tiles per frame). Hence, let $\mathcal{TL} = (TL_1, \ldots, TL_{18})$ be the set of all tiles in a given video frame. Furthermore, we use an OI to provide the necessary capabilities for object identification in the given video stream. More precisely, when the video stream is fed to the OI, it initializes a tracker with a bounding box around the object of interest in the first frame. Subsequently, the OI employs a correlation filter-based tracking algorithm to estimate the object's position and size in subsequent frames. Although the Dlib library

[2] https://github.com/davisking/dlib

focuses on object tracking rather than identification, once the object is consistently tracked, additional techniques, such as image classification models, are utilized to identify the object within the bounding box. Integrating the OI with the Kvazaar HEVC encoder enables the OI to output a tile TL_s representing a tile containing an object of interest. More formally, when a given video stream V is fed into the OI, it outputs a set of tiles $(TL_s)_{s\in\mathcal{S}}$, where $\mathcal{S} \subseteq [18]$, which contain objects of interest. Given the specific tiles containing the objects of interest, we proceed to encrypt the necessary parameters in each tile.

6.2 Encryption of Parameters

To encrypt IPM parameters and syntax elements of variable length in a tile of interest TL_s, we utilize a stream cipher. The encryption process occurs at the CABAC bin string level to distort the visual quality of the specific tile of interest. We note that the tiles not identified by the OI to contain any object of interest are untouched and remain clear. To this end, the encryption process is divided into encryption of the IPM and encryption of the syntax elements. **IPM Encryption:** To ensure compliance with the HEVC standard, we encrypt the IPMs while preserving their original scanning order before encryption. This encryption method allows the encryption of IPMs to be compatible with HEVC. Overall, there are 35 IPMs, and the scanning order details the order in which these modes are tested during the encoding process [26]. The IPM scanning order follows the raster scan pattern, allowing it to efficiently explore and select the best inter-prediction mode for each coding unit. The scanning order starts from the top-left corner of a coding unit and proceeds row by row, from left to right, until it reaches the bottom-right corner. To this end, the encryption process involves classifying the IPM elements into three sets:

$$SetVert = \{6, 7, 8, 9, 10, 11, 12, 13, 14\},$$
$$SetHor = \{22, 23, 24, 25, 26, 27, 28, 29, 30\},$$
$$SetDiag = \{0, 1, 2, 3, 4, 5, 15, 16, 17, 18, 19, 20, 21, 31, 32, 33, 34\}$$

Each set represents IPMs in the same scanning direction (vertical, horizontal, and diagonal). To encrypt, each set undergoes a circular shift operation based on the input key, resulting in a new position for the IPM in the same set. Furthermore, to encrypt the luma and chroma IPMs, we assume N as the number of elements in a set $M = [1, \ldots, N]$, where $M \in \mathbb{R}^N$, nb is the number of bits generated by the keystream, and j is the index position of the IPM. On encryption, the new IPM produced at the $s-th$ position of IPM $M[s]$ is given by:

$$M'[j] = M[(s + nb) \pmod N] \tag{1}$$

Syntax Elements Encryption: A simple XOR operation is used when encrypting the syntax elements (i.e. MV differences, MV signs, TC and TC signs). The equation used is given below:

$$C_s = Q_s \oplus \mathsf{X}(\mathsf{k_s}), \tag{2}$$

where C_s is the resulting ciphertext, Q_s syntax element, and $\mathsf{X}(\mathsf{k_s})$ is the key stream generated from the secret key $\mathsf{k_s}$ for tile TL_s.

Eventually, the algorithm outputs C_V, an encryption of the video containing both encryptions of tiles of interest TL_s and plain tiles TL_t, for $t \notin \mathcal{S}$. Note that, while the encryption of a tile TL_s consists in computing both C_s and $M'[s]$, the latter is not part of the output ciphertext as it never leaves the encoding of decoding process hence never comes out. The complete video encryption is provided in algorithm 1.

Algorithm 1: ABSVE.EncVid

Input (V):
Output $(C_V, (\mathsf{k_s})_{s\in\mathcal{S}})$:
Partition V into 18 tiles $(TL_1, \ldots, TL_{18})$;
Pass V through the OI to output $(TL_s)_{s\in\mathcal{S}}$, $\mathcal{S} \subseteq [18]$;
for $s \in \mathcal{S}$ **do**
  Generate a key $\mathsf{k_s}$;
  IPM
  SetVert = {6, 7, 8, 9, 10, 11, 12, 13, 14};
  SetHor = {22, 23, 24, 25, 26, 27, 28, 29, 30};
  SetDiag = {0, 1, 2, 3, 4, 5, 15, 16, 17, 18, 19, 20, 21, 31, 32, 33, 34};
  if *IPM > 5 and IPM < 15* **then**
    Enc_IPM = Circular shift(SetVert, IPM, X($\mathsf{k_s}$));
  else if *IPM > 21 and IPM < 31* **then**
    Enc_IPM = Circular shift(SetHor, IPM, X($\mathsf{k_s}$));
  else
    Enc_IPM = Circular shift(SetDiag, IPM, X($\mathsf{k_s}$))
  Luma and Chroma IPM
  $M'[s] = M[(s + nb) \pmod N]$;
  Syntax Elements
  $C_s = Q_s \oplus \mathsf{X}(\mathsf{k_s})$;
end

6.3 Attribute-Based Selective Video Encryption Scheme

We extend the SVE scheme in [27] to support enforcing fine-grained access control over video streams based on predefined attributes and policies. An ABSVE scheme is given by a tuple of five algorithms: ABSVE = (KeyGen, DKGen, EncVid, EncKeys, Decrypt), with the universe of attributes $\Omega = \{a_1, ..., a_n\}$ and the space of policies $\mathcal{P} = \{P_1, ..., P_m\}$. An encrypted video stream corresponding to a policy P can be decrypted by a secret key associated with a set of attributes $\mathcal{Y} \subseteq \Omega$ only when $P(\mathcal{Y}) = \text{True}$. The formal syntax of ABSVE is as follows.

Definition 6.1. (Attribute-based SVE) An ABSVE scheme is defined by the following five algorithms:

- ABSVE.KeyGen: This is a probabilistic algorithm that takes as input a security parameter λ and outputs a master public key MPK and a master secret key MSK. We denote this by $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$.
- ABSVE.DKGen: This is a probabilistic algorithm that takes as input the master secret key MSK, a registration token $\mathtt{regA}$ which consists of an attribute set $\mathcal{Y} \subseteq \Omega$ and a user identifier u, outputs a decryption key dk for the user and its associate attribute sets. We denote this by $\mathsf{dk} \leftarrow\!\!\$\ \mathsf{DKGen}(\mathsf{MSK}, \mathtt{regA})$.
- ABSVE.EncVid: This is a probabilistic algorithm that takes as input a video stream V and a set $\mathcal{S}$ indicating objects of

interest in V, outputs the encrypted video C_V along with a set of corresponding keys K for subsequent video access. We denote this by $(K, C_V) \leftarrow\!\$\ \mathsf{EncVid}(V, \mathcal{S})$.
- ABSVE.EncKeys: This is a probabilistic algorithm that takes as input a set of keys K, the master key MPK and a policy P, outputs the encryption of keys C_K. We denote this by $C_K \leftarrow\!\$\ \mathsf{EncKeys}(K, \mathsf{MPK}, P)$.
- ABSVE.Decrypt: This is a deterministic algorithm that takes as input a decryption key dk, an encrypted key set C_K and an encrypted video C_V, outputs a decrypted video V'. We denote this by $V' \leftarrow \mathsf{Decrypt}(\mathsf{dk}, C_K, C_V)$.

Correctness: An ABSVE scheme is *correct* if for all λ, V, $\mathcal{Y}$, P, $\mathcal{S}$, satisfying $P(\mathcal{Y}) = \text{True}$, all (MPK, MSK) outputted by KeyGen, all dk outputted by DKGen(MSK, regA) with $\mathsf{regA} = (\mathcal{Y}, u)$, all (K, C_V) outputted by EncVid$(V, \mathcal{S})$, all C_K outputted by EncKeys(K, MPK), it holds that:

$$\mathsf{Decrypt}(\mathsf{dk}, C_K, C_V) = V.$$

Semantic Security: We define semantic security for ABSVE schemes to ensure that ciphertexts (encrypted video streams) are indistinguishable. This is formalized through a game where the adversary attempts to distinguish ciphertexts encrypted under fixed policy sets. The adversary has access to two oracles: DKGen, which generates decryption keys for specified attribute sets, and Challenge, which allows requests to encrypt a video along with associated keys and a policy from the supported policy set, based on a random bit b chosen at the beginning of the game. To prevent trivial wins, the adversary cannot use any decryption key from DKGen to recover any encrypted content returned from Challenge. Specifically, for all policies P in list Ch and attribute sets $\mathcal{Y}$ in list I, it must hold that $P(\mathcal{Y}) = \text{False}$ throughout the game.

Definition 6.2 (Semantic Security). We say an ABSVE scheme for tile set $\mathcal{S}$, policy set $\mathcal{P}$ and attribute universe Ω is semantically secure, if for all p.p.t. adversary $\mathcal{A}$, it holds that the advantage of $\mathcal{A}$:

$$\mathbf{Adv}^{\text{ind-}\mathcal{S}\text{-}\mathcal{P}\text{-}\Omega}_{\text{ABSVE},\mathcal{A}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\text{ind-}\mathcal{S}\text{-}\mathcal{P}\text{-}\Omega}_{\text{ABSVE},\mathcal{A}}(\lambda) \rightarrow \text{true}] - \frac{1}{2}\right|$$

is negligible in λ, where the experiment is defined as follows.

$\mathbf{Exp}^{\text{ind-}\mathcal{S}\text{-}\mathcal{P}\text{-}\Omega}_{\text{ABSVE},\mathcal{A}}(\lambda)$
$b \leftarrow\!\$\ \{0,1\}$; $\mathcal{I}, Ch \leftarrow \emptyset$
$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda)$
$b' \leftarrow\!\$\ \mathcal{A}(1^\lambda, \mathsf{MPK} : \text{DKGen}, \text{Challenge})$
Return $(b = b')$

Oracles

DKGen(regA)
Parse regA as $(\cdot, \mathcal{Y})$
If $\mathcal{Y} \not\subseteq \Omega$ then Return $\perp$
For all $P \in Ch$:
  If $P(\mathcal{Y}) = \text{True}$ then
    Return $\perp$
$\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{Y}$
Return $dk \leftarrow\!\$\ \mathsf{DKGen}(\mathsf{MSK}, \mathsf{regA})$

Challenge(V_0, V_1, P)
If $|V_0| \neq |V_1| \vee P \notin \mathcal{P}$ then
  Return $\perp$
For all $\mathcal{Y} \in \mathcal{I}$:
  If $P(\mathcal{Y}) = \text{True}$ then
    Return $\perp$
$K, C_V \leftarrow\!\$\ \mathsf{EncVid}(V_b, \mathcal{S})$
$C_K \leftarrow\!\$\ \mathsf{EncKeys}(K, \mathsf{MPK}, P)$
$Ch \leftarrow Ch \cup \{P\}$
Return C_K, C_V

7 BLIND BROTHER

In this section, we present our ABSVE protocol `Blind Brother`, a core contribution of this work based on the ABSVE scheme introduced in subsection 6.3. Overall, `Blind Brother` utilizes the tuple of five algorithms introduced by ABSVE, namely, ABSVE.KeyGen, ABSVE.EncVid, ABSVE.EncKeys, ABSVE.DKGen, and ABSVE.Decrypt Furthermore, we consider that encryption of syntax elements is performed using a symmetric encryption denoted as SKE = (Gen, Enc, Dec). `Blind Brother` employs the following building blocks to provide secure communication among all protocol entities:

- An IND-CPA secure public key encryption scheme PKE = (Gen, Enc, Dec);
- An EUF-CMA secure signature scheme S = (Gen, Sign, Verify);
- A first and second pre-image resistant cryptographic hash function H(·);
- A synchronized clock between all entities.

To provide a comprehensive overview of `Blind Brother`, we consider a use case scenario, where an administrator **AD** manages a number of camera-equipped devices $(d_1, \ldots, d_n)$ deployed in an environment to capture video feeds. Each device d_i encodes and encrypts video streams V_i and securely uploads to the **CSP**. To access an uploaded video, a registered user u_j contacts the **CSP** to retrieve a specific video and decrypts parts of the encrypted video based on their access rights and decryption key. We divide the entire process into three phases: *Setup and Registration*, *Video Upload* and *Video Access*. A complete overview of `Blind Brother` is presented in Figure 2.

Setup and Registration Phase: Each entity first generates its signing and verification key pair via the signature scheme S and asymmetric encryption and decryption key pair via the public key encryption scheme PKE. Subsequently, **AD** runs the ABSVE.KeyGen algorithm with a security parameter λ as input and outputs the master public and private keys MPK, MSK. For simplicity, we denote the signature on message m generated by party x as $\sigma_x(m)$. On a successful run, **AD** securely transfers MPK to each device d_i via m_1:

$$m_1 = \langle t_1, \mathsf{MPK}, \sigma_{AD}(\mathsf{H}(t_1 || MPK)) \rangle.$$

Finally, each user u_j contacts **AD** to initiate user registration. During user registration, a user u_j who wishes to access video streams stored with **CSP** requests a CP-ABE decryption key bound to their specific attributes. This is done by generating a registration token regA, which contains the user's identity and attributes such that $\mathsf{regA} = (u_j, \mathcal{Y}_j)$. The generated token is then securely sent to **AD** through m_2:

$$m_2 = \langle t_2, \mathsf{PKE.Enc}(\mathsf{pk_{AD}}, \mathsf{regA}), \sigma_{u_j}(\mathsf{H}(t_2 || c_{regA})) \rangle,$$

where $c_{regA} = \mathsf{Enc}(\mathsf{pk_{AD}}, \mathsf{regA})$. On receiving m_2, **AD** verifies the message's authenticity. Subsequently, **AD** executes the ABSVE.DKGen algorithm with inputs $\mathcal{Y}_j, u_j$, and MSK, and outputs the CP-ABE decryption key $\mathsf{dk_j}$. To complete user registration, the generated key is securely sent to u_j through m_3:

$$m_3 = \langle t_3, \mathsf{PKE.Enc}(\mathsf{pk_{u_j}}, \mathsf{dk_j}), \sigma_{AD}(\mathsf{H}(t_3 || c_{dk_j})) \rangle,$$

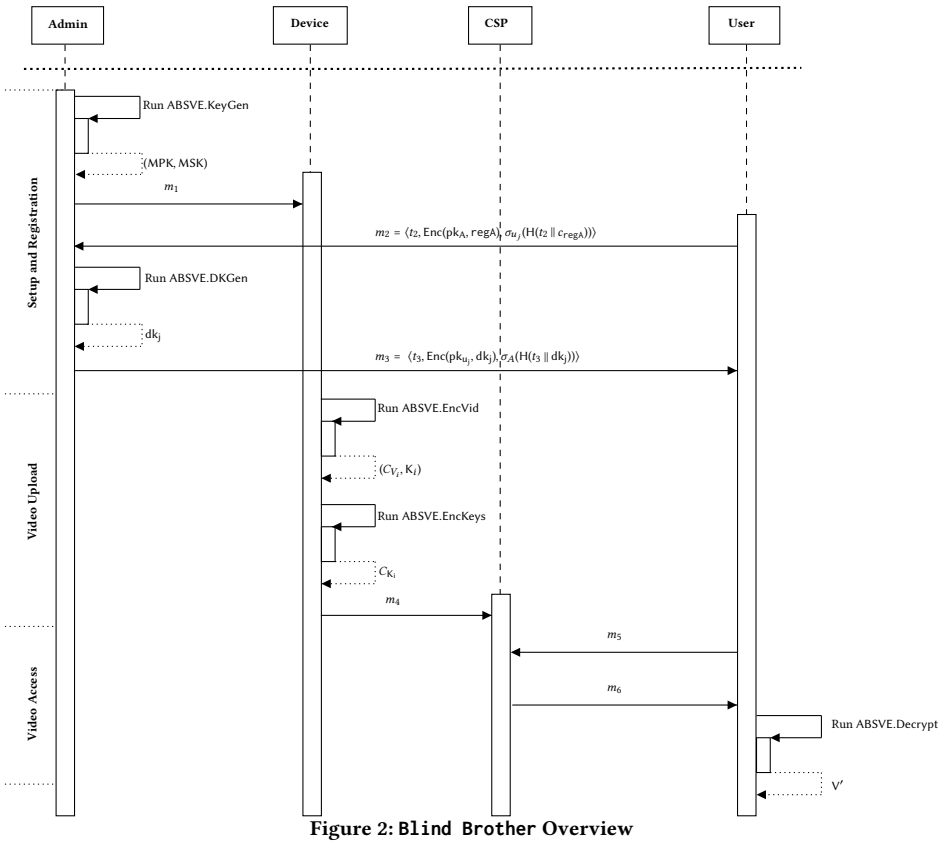

Figure 2: Blind Brother Overview

where $c_{dk_j} = \text{PKE.Enc}(\text{pk}_{\text{u}_j}, \text{dk}_\text{j})$. When u_j receives m_3, they verify the message's authenticity, integrity, and freshness and retrieve dk_j on successful verification.

Video Upload Phase: To initiate this phase, each device d_i captures a video stream V_i. The video stream is then passed through an object detector to identify the set of tiles $(TL_s)_{s\in\mathcal{S}}$ containing regions of interest. Now, using the set of index $\mathcal{S} \subseteq [n]$ and the raw video file V_i as inputs, d_i executes the ABSVE.EncVid algorithm to selectively encrypt the video during compression (algorithm 1). On a successful run, ABSVE.EncVid outputs the encrypted compressed video C_{V_i} and the set of symmetric keys used for the video encryption process $\text{K}_\text{i} = (\text{k}_\text{s})_{s\in\mathcal{S}}$. Thereafter, d_i runs the ABSVE.EncKeys algorithm to encrypt the set of symmetric keys. This algorithm takes as input a specific policy P_i, MPK and the set of symmetric keys K_i, and outputs $C_{\text{K}_\text{i}} = (c_{\text{k}_\text{s}})_{s\in\mathcal{S}}$ bound to P_i. Once completed, d_i uploads the encrypted video stream to **CSP** through m_4:

$$m_4 = \langle t_4, C_{K_i}, C_{V_i}, \sigma_{d_i}(\text{H}(t_4||C_{V_i}||C_{K_i})) \rangle.$$

Video Access Phase: When a user u_j wishes to access video streams stored with **CSP**, they contact the **CSP** to request the uploaded video along with its corresponding key set. This is accomplished with a video access token reqV, which contains an identifier for a specific video or a description of a possible event. This is sent to the **CSP** through m_5:

$$m_5 = \langle t_5, \text{PKE.Enc}(\text{pk}_{\text{CSP}}, \text{reqV}), \sigma_{u_j}(\text{H}(t_5||c_{reqV})) \rangle,$$

where $c_{reqV} = \text{PKE.Enc}(\text{pk}_{\text{CSP}}, \text{reqV})$. On receiving a request for an uploaded video, **CSP** verifies the message and, on successful verification, returns the video and its corresponding set of keys to the requesting party through m_6:

$$m_6 = \langle t_6, C_{K_i}, C_{V_i}, \sigma_{CSP}(\text{H}(t_6||C_{V_i}||C_{K_i})) \rangle.$$

When u_j receives the response from the **CSP**, they verify the message's authenticity, integrity, and freshness and proceed to video decryption. To do so, u_j runs the ABSVE.Decrypt algorithm with inputs C_{K_i}, C_{V_i}, and the CP-ABE decryption key dk_j. When successful, ABSVE.Decrypt first outputs K'_i, the set of keys the user u_j can access according to their attributes and the policy set by **A**. Using K'_i, ABSVE.Decrypt finally outputs V'_i, where V'_i is a selectively decrypted version of V_i.

8 SECURITY DEFINITIONS

In this section, we introduce two formal security definitions, namely *robustness* and *secure access*, for the fine-grained access control-based video encryption protocol described in section 7.

The first security definition we present is *robustness*. In standard fine-grained access control protocols, correctness guarantees that any user can successfully access the files they are authorized to retrieve. Similarly, in fine-grained access control-based video encryption protocols, correctness should guarantee that authorized users can successfully access the video tiles they are entitled to. In the typical security definition of correctness in cryptographic access

control systems (e.g. [13, 20, 21]), the adversary is restricted from taking over users and altering the messages exchanged between parties in the system. However, robustness is a strictly stronger security notion than correctness. In the game defining robustness, the adversary has the additional ability to generate and tamper with messages exchanged between the parties. Despite the presence of such an active adversary, the protocol is expected to guarantee correctness of the system.

This security property is defined via the following experiment involving a challenger, acting as **AD**, and an adversary $\mathcal{A}$ interacting with the fine-grained access control-based video encryption protocol Π. The challenger first runs Setup, which generates initial local states for all parties and provides each device d with an initialization message msg_d. Upon receiving msg_d, device d executes Init_D, verifying msg_d and updating its local state. All device initialization messages are revealed to $\mathcal{A}$, who may then invoke the oracle DeviceInit(d, msg_d) to further initialize devices, receiving a boolean response indicating success. User registration proceeds analogously: the adversary may call UserReg($u, \mathcal{Y}$, regA) to register a user u with attributes $\mathcal{Y}$ and upon success, an update message is returned. The user then updates its local state by calling UserInit, which runs Init_U after verifying this update message.

The adversary can request a device d to upload a video stream V with policy P by querying VideoUpload(d, V, P). The oracle runs VideoEncrypt to produce (C, vid), and then executes VideoUpload with d's local state and (vid, C), yielding the upload message. An entry vid, V, P, C is then recorded in the list $\mathcal{V}$. The upload message is revealed to $\mathcal{A}$. We do not consider availability attacks (e.g., blocking messages), and the cloud service provider (**CSP**) is not required to validate inputs. The oracle directly records each entry as generated by the device.

The adversary can also invoke VideoUpload1(d, msg) to provide arbitrary upload information. If msg is valid and no entry for vid exists in $\mathcal{V}$, the oracle returns vid and sets a flag forged = True, indicating successful forgery. To access an encrypted video, the adversary can query VideoRAccess oracle with a video access token reqV. The token is structured as a specific video identifier vid along with a potential event name $event$. After that, the adversary receives the encrypted video C from the records in $\mathcal{V}$.

There are two situations in which the adversary can be considered to win:

(1) A device uploads a video V according to a policy P, but no authorized user (with attributes satisfying P) can correctly recover V.
(2) A user gains access to video data identified by some vid that was never uploaded by any device (i.e., the data is forged).

In either case, if there exists a user u^* and a video identifier vid^* such that u^* fails to decrypt an authorized video or the forged flag is set, the adversary $\mathcal{A}$ wins the game.

Definition 8.1 (Robustness). A fine-grained access control-based video encryption protocol Π for edge devices $\mathcal{D}$, registered users $\mathcal{U}$ and policies $\mathcal{P}$ is said to be **robust**, if for all p.p.t adversary $\mathcal{A}$, it holds that

$$\mathbf{Adv}^{\text{robust-}\mathcal{D}\text{-}\mathcal{U}\text{-}\mathcal{P}}_{\Pi,\mathcal{A}}(\lambda) = \Pr[Exp^{\text{robust-}\mathcal{D}\text{-}\mathcal{U}\text{-}\mathcal{P}}_{\Pi,\mathcal{A}}(\lambda) \rightarrow \text{True}]$$

is negligible in λ, where the experiment $Exp^{\text{robust-}\mathcal{D}\text{-}\mathcal{U}\text{-}\mathcal{P}}_{\Pi,\mathcal{A}}$ is defined as follows.

$Exp^{\text{robust-}\mathcal{D}\text{-}\mathcal{U}\text{-}\mathcal{P}}_{\Pi,\mathcal{A}}(\lambda)$

$\mathcal{V} \leftarrow \emptyset$; forged $\leftarrow$ False
$(\text{pp}, st_{\text{AD}}, \{st_d\}_{d\in\mathcal{D}}, \{msg_d\}_{d\in\mathcal{D}}, \{st_u\}_{u\in\mathcal{U}}) \leftarrow\$ \text{Setup}(1^\lambda)$
$(u^*, \text{reqV}^*) \leftarrow\$ \mathcal{A}(1^\lambda, \text{pp}, \{msg_d\}_{d\in\mathcal{D}} : O_{\text{robust}})$
Parse reqV* as $(vid^*, event)$
If (forged = True) $\vee$
  $(\exists\ vid^*$ such that $(vid^*, V^*, P^*, C^*) \in \mathcal{V} \wedge$
  $P^*(\mathcal{Y}_{u^*}) = \text{True} \wedge V^* \neq \text{VideoAccess}(st_{u^*}, C^*))$ then
Return True

Oracles O_{robust}

DeviceInit(d, msg_d)
If $d \notin \mathcal{D}$ then Return $\perp$
result = Init_D(pp, st_d, msg_d)
If result = $\perp$ then Return $\perp$
  Else st_d = result
Return True

UserReg(regA)
Parse regA as $(u, \mathcal{Y})$
If $u \notin \mathcal{U}$ then Return $\perp$
$msg_u \leftarrow\$$ UserReg$(u, \mathcal{Y})$
Return msg_u

UserInit(u, msg_u)
If $u \notin \mathcal{U}$ then Return $\perp$
result = Init_U(pp, st_u, msg_u)
If result = $\perp$ then Return $\perp$
  Else st_u = result
Return True

VideoUpload(d, V, P)
If $d \notin \mathcal{D} \vee P \notin \mathcal{P}$ then
  Return $\perp$
$(vid, C) \leftarrow\$$ VideoEncrypt(st_d, V, P)
$msg \leftarrow\$$ VideoUpload(st_d, vid, C)
$\mathcal{V} \leftarrow \mathcal{V} \cup \{vid, V, P, C\}$
Return (vid, msg)

VideoUpload1(d, msg)
If $d \notin \mathcal{D}$ then Return $\perp$
Extract (vid, C) from msg
If $\exists\ (vid, *, *, *) \in \mathcal{V}$ then
  Return $\perp$
If $\text{Verify}_{\text{CSP}}(d, msg) = \text{True}$ then
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{id, *, *, C\}$
  forged = True
  Return vid
Return $\perp$

VideoRAccess(reqV)
Parse reqV as $(vid, event)$
If $\exists\ (vid, *, *, C) \in \mathcal{V}$ then
  Return C
Else Return $\perp$

The second security definition we introduce is *secure access*. Essentially, it ensures that unauthorized users cannot obtain any partial content of encrypted video tiles. This property is defined by a security game similar to the robustness game, but now the adversary $\mathcal{A}$ attempts to guess a random bit b chosen at the beginning of the game. The game maintains three lists: $\mathcal{V}$ for encrypted video content, Ch for challenge policies, and Cr for corrupted users. Unlike the robustness game, the adversary may call a corruption oracle Corrupt to compromise users. Thus, UserReg is modified to permit corruption before and after user registration. However, $\mathcal{A}$ cannot run UserInit, since only the challenger can initialize users. Honest users do not write data, so they cannot aid the adversary in leaking video content through write operations. Similarly, the adversary has no capability to impersonate devices.

The adversary issues a challenge via Challenge by specifying a device d, two equal-length video streams V_0 and V_1, and a policy P. The challenger encrypts and stores V_b (for the secret bit b) in $\mathcal{V}$. To prevent trivial wins, every corrupted user's attribute set must fail to satisfy the challenge policies. The game ends when the adversary outputs a guess b'. The adversary wins if $b' = b$.

We require that a fine-grained access control-based video encryption protocol guarantees secure access if, for all, efficient adversaries cannot win the above game with a probability significantly higher than one-half.

Definition 8.2 (Secure Access). A fine-grained access control-based video encryption protocol Π for edge devices $\mathcal{D}$, registered users $\mathcal{U}$ and policies $\mathcal{P}$ is said to be secure with respect to video access, if for all p.p.t adversary $\mathcal{A}$, it holds that

$$\mathbf{Adv}^{\text{ind-}\mathcal{D}\text{-}\mathcal{U}\text{-}\mathcal{P}}_{\Pi,\mathcal{A}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\text{ind-}\mathcal{D}\text{-}\mathcal{U}\text{-}\mathcal{P}}_{\Pi,\mathcal{A}}(\lambda) \rightarrow \text{true}] - \frac{1}{2}\right|$$

is negligible in λ, where the experiment $\mathbf{Exp}^{\text{ind-}\mathcal{D}\text{-}\mathcal{U}\text{-}\mathcal{P}}_{\Pi,\mathcal{A}}$ is defined as follows.

$\mathbf{Exp}^{\text{ind-}\mathcal{D}\text{-}\mathcal{U}\text{-}\mathcal{P}}_{\Pi,\mathcal{A}}(\lambda)$

$\mathcal{V}, Ch, Cr \leftarrow \emptyset$; $b \leftarrow\$ \{0,1\}$
$(\text{pp}, st_{\text{AD}}, \{st_d\}_{d\in\mathcal{D}}, \{msg_d\}_{d\in\mathcal{D}}, \{st_u\}_{u\in\mathcal{U}}) \leftarrow\$ \text{Setup}(1^\lambda)$
$b' \leftarrow\$ \mathcal{A}(1^\lambda, \text{pp}, \{msg_d\}_{d\in\mathcal{D}} : \mathcal{O}_{\text{ind}})$
Return $(b = b')$

Oracles $\mathcal{O}_{ind}$

DeviceInit(d, msg_d)
If $d \notin \mathcal{D}$ then Return $\perp$
result = $\text{Init}_D(\text{pp}, st_d, msg_d)$
If result = $\perp$ then Return $\perp$
Else st_d = result
Return True

UserReg(regA)
Parse regA as $(u, \mathcal{Y})$
If $u \notin \mathcal{U}$ then Return $\perp$
If $u \in Cr$ then
For all $P \in Ch$:
If $P(\mathcal{Y}_u)$ = True then
Return $\perp$
$st_u \leftarrow\$$ UserReg$(u, \mathcal{Y})$
If $u \in Cr$ then Return st_u
Else Return u

VideoUpload(d, V, P)
If $d \notin \mathcal{D} \vee P \notin \mathcal{P}$ then
Return $\perp$
$(vid, C) \leftarrow\$$ VideoEncrypt(st_d, V_b, P)
$\mathcal{V} \leftarrow \mathcal{V} \cup \{vid, V, P, C\}$
Return vid

VideoRAccess(reqV)
Parse reqV as $(vid, event)$
If $\exists\ (vid, V, P, C) \in \mathcal{V}$ then
Return C

Corrupt(u)
If $u \notin \mathcal{U}$ then Return $\perp$
For all $P \in Ch$:
If $P(\mathcal{Y}_u)$ = True then
Return $\perp$
$Cr \leftarrow Cr \cup \{u\}$
Return st_u

Challenge(d, V_0, V_1, P)
If $d \notin \mathcal{D} \vee P \notin \mathcal{P}$ then
Return $\perp$
If $|V_0| \neq |V_1|$ then
Return $\perp$
For all $u \in Cr$:
If $P(\mathcal{Y}_u)$ = True then
Return $\perp$
$(vid, C) \leftarrow\$$ VideoEncrypt(st_d, V_b, P)
$\mathcal{V} \leftarrow \mathcal{V} \cup \{(vid, V_b, P, C)\}$
$Ch \leftarrow Ch \cup \{P\}$
Return (id, C)

9 SECURITY ANALYSIS

Having introduced the security definitions, we now analyze the security of our construction `Blind Brother`. All participants are assumed to have synchronized clocks, so we treat each timestamp as a nonce for simplicity. The proofs for all security theorems presented in this section can be found in Appendix A.

The secure access property of our construction is proved in two steps. First, we prove the semantic security of `Blind Brother`, assuming the underlying schemes CPABE and SKE meet their respective IND-CPA security notion.

Theorem 9.1. *If the CP-ABE scheme* CPABE *is selective IND-CPA secure and the symmetric encryption* SKE *is IND-CPA secure, the ABSVE scheme* ABSVE *is semantically secure.*

Up to now, we have not discussed how user attribute sets are assigned. If no user can access any video content under the policies in $\mathcal{P}$, we end up with a trivial scenario: devices could upload arbitrary (even random) content, preserving secure access without robustness. Such a scenario is clearly meaningless in practice. Therefore, we assume that each video file is accessible to at least one user according to the policy. Under this non-trivial assumption, we establish the secure access property of our construction through the following theorem.

Theorem 9.2. *If the public key encryption scheme* PKE *is IND-CPA secure, the signature scheme* S *is EUF-CMA secure, and the ABSVE scheme* ABSVE *is semantically secure, the construction* `Blind Brother` *is secure with respect to video access in the random oracle model.*

Next, we establish the robustness of the construction, relying on the security of the signature scheme.

Theorem 9.3. *If the signature scheme* S *is EUF-CMA secure, the construction* `Blind Brother` *is robust in the random oracle model.*

10 EXPERIMENTS

This section presents a detailed account of the experimental evaluations performed to assess `Blind Brother`'s computational overhead and objective video quality. Our primary objective was to evaluate the trade-off between security, efficiency and computational complexity, alongside the influence of `Blind Brother` on the video quality and transmission bandwidth. To this end, we integrated `Blind Brother` with the real-time Kvazaar [19] encoder and evaluated it on five video sequences of different resolutions (Table 1). Our experiments were conducted on a 64-bit 20-core Intel Core 12th Gen Core i7 machine with 32GB RAM and running Ubuntu 20.04 operating system. Additionally, we used a modified version of the OpenABE[3] library to implement the CP-ABE component for key distribution. Finally, to ensure statistical significance, each experiment was repeated 20 times, with the average results considered to provide a comprehensive overview of `Blind Brother`'s performance.

Video Sequence	Resolution	Size
FlowerKids	3840 x 2160	7.5 GB
Beauty	1920 x 1080	1.9 GB
YachtRide	1920 x 1080	1.9 GB
Demonstrator	1280 x 720	0.8 GB
FourPeople	1280 x 720	0.8 GB

Table 1: Test Video Sequences

Computational Overhead: In this phase, we focused on the computational expenses incurred when integrating `Blind Brother` into Kvazaar. More specifically, we compared the object identification, encoding and encryption processes against the encoding process when using Kvazaar without any modifications. Additionally, we measured the computational cost of the ABSVE.EncKeys

[3] https://github.com/zeutro/openabe

algorithm to demonstrate the feasibility of the algorithm for a varying number of symmetric keys and attributes. For the comparison between `Blind Brother` and encoding with Kvazaar without any modifications, we considered five video sequences (Table 1) with a 6x3 tile configuration and a different number of ROIs. Using video sequences with a different number of ROIs allowed us to observe `Blind Brother` when run with a different number of symmetric encryption keys (number of ROIs = number of used symmetric keys). Overall, we made several observations from the computational experiments (Table 2). Notably, in the case of relatively low-resolution video sequences, i.e. the 1280x720p *FourPeople* and *Demonstrator* sequences, we observed a notably more significant disparity in computational times, roughly measuring around 58.4%. Conversely, when examining the higher-resolution video sequences, more specifically the 1920x1080p *Beauty* and *YachtRide* sequences, the disparity in computational times was comparatively modest, at approximately 6.5%. This divergence in computational impacts can be attributed to the inherent efficiency of the **OI** component utilized in our experiments. Additionally, we observed that the number of ROIs in the video sequence did *not* impact the computational complexity in any visible manner. These results provide evidence that the additional computational complexity introduced by `Blind Brother` does not adversely affect the performance of Kvazaar and can be easily applied to a real-world scenario.

Video Sequence	Basic Encoding	`Blind Brother`
FlowerKids	700.6s	842.1s
Beauty	335.7s	355.8s
YachtRide	245.7s	261.3s
Demonstrator	51.3s	81.6s
FourPeople	52.6s	84.2s

Table 2: Computational Comparisons

Evaluation of ABSVE.EncKeys: A core component of `Blind Brother` is the execution of the ABSVE.EncKeys algorithm at the end device. In this part of our experiments, we focused on assessing the performance of the ABSVE.EncKeys algorithm. This evaluation entailed executing the algorithm with a variable number of symmetric encryption keys, generated by the ABSVE.EncVid algorithm, alongside an array of access policies set by the end device. The combination of an arbitrary number of keys and policy size lends realism to our simulations. It's noteworthy, however, that the number of symmetric keys remains bounded by the total number of tiles in a given video frame in a practical setting. To begin, we first evaluated the ABSVE.EncKeys algorithm by employing a fixed policy of five (5) attributes. Within this context, we encrypted a list of keys from 10 to 1000 (Figure 3). Our evaluations revealed that encrypting a 10-key list with a 5-attribute policy took approximately 0.12 seconds, while encrypting a 1000-key list with the same policy size took approximately 12.13 seconds. Furthermore, decryption took 26.64 seconds for a 1000-key list and 0.30 seconds for a 10-key list with the same policy size.

Subsequently, we evaluated ABSVE.EncKeys, this time bearing a fixed list of 18 keys and a variable policy size ranging from 5 to 25 attributes (Figure 4). Encrypting an 18-key list alongside

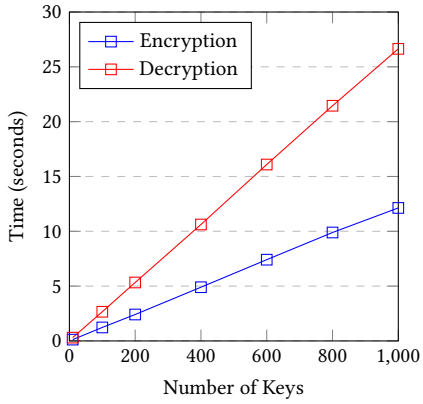

Figure 3: Varying Key Size

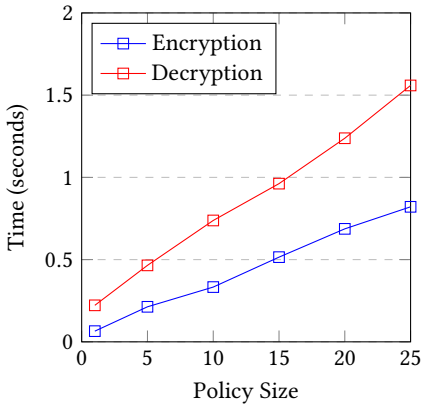

Figure 4: Varying Policy Size

a 25-attribute policy took approximately 0.82 seconds, while for a policy of 5 attributes, it took 0.21 seconds. To decrypt a list of 18 encrypted keys with a policy size of 25, it took 1.56 seconds; for a policy size of 5 attributes, it took 0.46 seconds. From these results, we can stipulate that the computational overhead incurred by the encryption of the symmetric keys with CP-ABE, when juxtaposed with the selective encryption of a given video, is negligible. This assertion, in turn, lends credence to the practical viability of `Blind Brother`.

Objective Video Quality Assessment: To measure the effect of `Blind Brother` on video quality, we employ the widely recognized Peak Signal to Noise Ratio (PSNR) metric [16]. PSNR quantifies the differences between the original signal (i.e. the original video) and the compressed version, factoring in distortions and noise introduced during compression. More precisely, PSNR quantitatively assesses the extent to which a compressed video strays from its original counterpart. This assessment evaluates the mean squared error (MSE) calculated between corresponding pixels in the original and the compressed video. This method enables us to quantify how the two versions differ from one another, providing valuable insights into the perceptual quality of our proposed scheme's results. Overall, a higher PSNR value indicates less distortion and better similarity between the original and reconstructed video. In

other words, a higher PSNR generally implies higher perceived quality. However, it is worth acknowledging that PSNR does not always match human perception perfectly. It may not capture all visual quality aspects, especially complex compression artefacts not well represented by MSE [15]. For this reason, PSNR is often complemented with other metrics like Structural Similarity Index (SSIM) and Video Quality Metric (VQM), which better accommodate perceptual variations. However, for the purposes of this work, our evaluations utilize PSNR as the sole chosen metric.

Video Sequence	Basic Encoding	`Blind Brother`
FlowerKids	42.99	23.65
Beauty	40.39	11.12
YachtRide	43.52	27.56
Demonstrator	43.51	11.85
FourPeople	42.51	12.56

Table 3: PSNR Evaluations

To provide comprehensive results, we calculated the PSNR value for each video sequence when encoded without any encryption and when encoded with `Blind Brother`. In general, `Blind Brother` significantly decreases the quality of the video sequences as compared to basic encoding (Table 3). These results are not surprising as we encrypt specific ROIs which leads to greater visual distortion; however, these results buttress our claims that `Blind Brother` provides privacy in video streams based on identified ROIs.

11 POSSIBLE SOCIETAL IMPACT

Edward Snowden's disclosures shed light on the fact that the current level of surveillance in modern societies is incompatible with human rights. Many people, especially city dwellers, have a hard time going about their daily lives without falling under the gaze of public surveillance cameras in streets, public transport, buildings and public spaces. Cameras are everywhere, including drones or other devices. With surveillance technology becoming more sophisticated, many of us must come to terms with the unsettling truth that maintaining our private lives will become harder and harder. Even if the intention is to keep us safe, public surveillance can be harmful to society: there is indeed proof that the need for national security often trespasses individual privacy. Maintaining a balance between the two has been a very challenging task. Advancements in technology (cryptography, hardware, computational power, etc.) can help us build privacy-respecting surveillance systems.

Keeping in mind that encryption is the best option for standing up against surveillance of any kind, this work aspires to pave the way for the implementation of camera programs that respect the democratic nature of societies. In addition to that, we hope to help governments fulfill their responsibility to citizens while answering the call for national security. After all, privacy is not just a regulation that authorities must abide by, but a fundamental human right that requires safeguarding.

12 CONCLUSION

In this work, we demonstrated the implementation and feasibility of combining ciphertext-policy attribute-based encryption with region-of-interest encryption on video files, highlighting the practicality of this integration. To achieve this, we designed a fine-grained access control selective video encryption scheme that leverages CP-ABE to tie symmetric keys for video stream encryption to specific access policies, ensuring robust access control. Building on this foundation, we developed `Blind Brother`, a use-case protocol that illustrates the real-world applicability of our scheme and conducted a comprehensive security analysis to validate its resilience. Finally, we implemented our scheme within the real-time Kvazaar HEVC encoder, demonstrating its performance and feasibility in practical encoding scenarios. Together, these contributions provide a foundation for privacy-preserving video encryption that balances security, functionality, and efficiency.

REFERENCES

[1] Agi, I., Gong, L.: An empirical study of secure mpeg video transmissions. Proceedings of Internet Society Symposium on Network and Distributed Systems Security p. 137–144 (1996)

[2] Agrawal, S., Chase, M.: Fame: Fast attribute-based message encryption. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (2017)

[3] Bakas, A., Michalas, A.: Modern family: A revocable hybrid encryption scheme based on attribute-based encryption, symmetric searchable encryption and sgx. In: Chen, S., Choo, K.K.R., Fu, X., Lou, W., Mohaisen, A. (eds.) Security and Privacy in Communication Networks. pp. 472–486. Springer International Publishing, Cham (2019)

[4] Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. 2007 IEEE Symposium on Security and Privacy (SP '07) (2007)

[5] Bross, B., Wang, Y.K., Ye, Y., Liu, S., Chen, J., Sullivan, G.J., Ohm, J.R.: Overview of the versatile video coding (vvc) standard and its applications. IEEE Transactions on Circuits and Systems for Video Technology **31**(10), 3736–3764 (2021)

[6] Chen, C., Wang, X., Huang, G., Liu, G.: An efficient randomly-selective video encryption algorithm. 2022 IEEE 8th International Conference on Computer and Communications (ICCC) (2022)

[7] Chen, C., Wang, X., Liu, G., Huang, G.: A robust selective encryption scheme for h.265/hevc video. IEEE Access **11** (2023)

[8] Cheung, L., Newport, C.: Provably secure ciphertext policy abe. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. CCS '07, Association for Computing Machinery, New York, NY, USA (2007)

[9] Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard) (Aug 2008), http://www.ietf.org/rfc/rfc5246.txt, updated by RFCs 5746, 5878, 6176

[10] Du, H., Chen, L., Qian, J., Hou, J., Jung, T., Li, X.Y.: Patronus: A system for privacy-preserving cloud video surveillance. IEEE Journal on Selected Areas in Communications **38**(6), 1252–1261 (2020)

[11] Farajallah, M., Gautier, G., Hamidouche, W., Deforges, O., Assad, S.E.: Selective encryption of the versatile video coding standard. IEEE Access **10** (2022)

[12] Farajallah, M., Hamidouche, W., Deforges, O., Assad, S.E.: Roi encryption for the hevc coded video contents. 2015 IEEE International Conference on Image Processing (ICIP) (2015)

[13] Ferrara, A.L., Fuchsbauer, G., Warinschi, B.: Cryptographically enforced rbac. In: 2013 IEEE 26th Computer Security Foundations Symposium. pp. 115–129 (2013)

[14] Hamidouche, W., Farajallah, M., Sidaty, N., Assad, S.E., Deforges, O.: Real-time selective video encryption based on the chaos system in scalable hevc extension. Signal Processing: Image Communication **58**, 73–86 (2017)

[15] Hore, A., Ziou, D.: Image quality metrics: Psnr vs. ssim. 2010 20th International Conference on Pattern Recognition (2010)

[16] Huynh-Thu, Q., Ghanbari, M.: Scope of validity of psnr in image/video quality assessment. Electronics Letters **44**(13), 800 (2008)

[17] Institute, F.H.H.: High efficiency video coding (hevc) reference software (Feb 2015), https://hevc.hhi.fraunhofer.de/

[18] Kemp, S.: Digital 2023: Global overview report – global digital insights (Feb 2023), https://datareportal.com/reports/digital-2023-global-overview-report

[19] Lemmetti, A., Viitanen, M., Mercat, A., Vanne, J.: Kvazaar 2.0: Open-source hevc/h.265 encoder. Proceedings of the 11th ACM Multimedia Systems Conference (2020)

[20] Liu, B., Michalas, A., Warinschi, B.: Cryptographic role-based access control, reconsidered. In: Provable and Practical Security - 16th International Conference (ProvSec) (2022)

[21] Liu, B., Warinschi, B.: Universally composable cryptographic role-based access control. In: Chen, L., Han, J. (eds.) Provable Security - 10th International Conference, ProvSec 2016, Nanjing, China, November 10-11, 2016, Proceedings. Lecture

Notes in Computer Science, vol. 10005, pp. 61–80 (2016)
[22] Liu, F., Koenig, H.: A survey of video encryption algorithms. Journal of Computers and Security **29**(1), 3–15 (2010)
[23] Michalas, A.: The lord of the shares: Combining attribute-based encryption and searchable encryption for flexible data sharing. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 146–155. SAC '19, ACM, New York, NY, USA (2019), http://doi.acm.org/10.1145/3297280.3297297
[24] Misra, K., Segall, A., Horowitz, M., Xu, S., Fuldseth, A., Zhou, M.: An overview of tiles in hevc. IEEE Journal of Selected Topics in Signal Processing **7**(6) (2013)
[25] Neuman, B., Ts'o, T.: Kerberos: an authentication service for computer networks. IEEE Communications Magazine **32**(9), 33–38 (1994)
[26] Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (hevc) standard. IEEE Transactions on Circuits and Systems for Video Technology **22**(12), 1649–1668 (2012)
[27] Taha, M.A., Sidaty, N., Hamidouche, W., Dforges, O., Vanne, J., Viitanen, M.: End-to-end real-time roi-based encryption in hevc videos. 2018 26th European Signal Processing Conference (EUSIPCO) (2018)
[28] Viitanen, M., Koivula, A., Lemmetti, A., Ylä-Outinen, A., Vanne, J., Hämäläinen, T.D.: Kvazaar: Open-source hevc/h.265 encoder. Proceedings of the 24th ACM international conference on Multimedia (2016)
[29] Xu, D.: Commutative encryption and data hiding in hevc video compression. IEEE Access **7**, 66028–66041 (2019)
[30] Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Connection Protocol. RFC 4254 (Proposed Standard) (January 2006), http://www.ietf.org/rfc/rfc4254.txt
[31] Zhang, Q.J., Ye, Q., Yuan, Z.M., Li, L.: Fast hevc selective encryption scheme based on improved cabac coding algorithm. 2020 IEEE 6th International Conference on Computer and Communications (ICCC) (2020)

A SECURITY OF THE CONSTRUCTION

In this section, we provide the proofs for the security theorems presented in section 9.

A.1 Proof of Theorem 9.1

Proof. We prove the theorem with a security game for an adversary $\mathcal{A}$. We prove that `Blind Brother` achieves s-IND-CPA security *iff* CPABE is s-IND-CPA secure. The game is initialized by flipping a coin $\beta \xleftarrow{\$} \{0,1\}$. The adversary $\mathcal{A}$ picks an access structure $(\mathcal{Y}, \{P_1, \ldots, P_\ell\}) \subset (\Omega, \mathcal{P})$ and receives the keys MPK, for $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{CPABE.Setup}(1^\lambda)$. $\mathcal{A}$ picks two valid videos V_0, V_1, identifies the tiles of interest through the object identifier (subsection 6.1) and extracts the corresponding sets $\mathcal{S}_0, \mathcal{S}_1$. Then, $\mathcal{A}$ sends the pairs $\Pi_0 = (V_0, \mathcal{S}_0), \Pi_1 = (V_1, \mathcal{S}_1)$ to $\mathcal{C}$. $\mathcal{A}$ can send up to n adaptative requests for decryption keys related to sets of attributes $\mathcal{Y}_1, \ldots, \mathcal{Y}_n \notin \Omega \backslash \mathcal{Y}$. She obtains a set of decryption keys $\mathsf{dk}_{\mathcal{Y}_1}, \ldots, \mathsf{dk}_{\mathcal{Y}_n}$ for $\mathsf{dk}_{\mathcal{Y}_i} \leftarrow \mathsf{CPABE.DKGen}(\mathsf{regA}_i)$. The key point here is that $\mathcal{A}$ sends request for attributes $\mathcal{Y}_i$ such that $\forall P \in \{P_1, \ldots, P_\ell\}$, $P(\mathcal{A}_i) = \mathit{False}$. $\mathcal{A}$ receives an encryption (C_{V_β}, C_K) of Π_β, where $C_{V_\beta} \leftarrow \mathsf{CV.EncVid}(V_\beta, \mathcal{S}_\beta)$ and $C_K \leftarrow \mathsf{CV.EncKey}$ (K, MPK). It then has the possibility to repeat the key generation step for additional adaptative requests. Then, $\mathcal{A}$ sends a guess β' for β. Denote λ_1, λ_2 the respective security parameters of CPABE and SKE. By assumption, $\mathcal{A}$ has an advantage $\epsilon_1 = \mathit{negl}(\lambda_1)$ against CPABE and an advantage $\epsilon_2 = \mathit{negl}(\lambda_2)$ against SKE. Remark that, by definition, the distribution of C_{V_β} and C_K are indistinguishable from the uniform distribution on their respective space. Hence, ϵ_1 and ϵ_2 are independent. This results in an overall advantage $\epsilon = \epsilon_1 + \epsilon_2 = \mathit{negl}(\lambda_1, \lambda_2)$. □

A.2 Proof of Theorem 9.2

Proof. We now provide a proof of this theorem using a sequence of games, which we describe below. Similar to the proof of the previous theorem, we will not present the reductions in full details, but will discuss the general ideas and the key steps involved.

Game 0: The first game is the experiment which defines secure access in Theorem 8.2.

Game 1: This game proceeds similarly to the previous one, with the exception that when the adversary requests to initialize a device d with a valid initial message msg_d, the challenger aborts the game. Analogous to the proof of the previous security theorem, we can readily establish the following claim.

Claim 1. $|\Pr[\mathbf{Game}_{0,\mathcal{A}}] - \Pr[\mathbf{Game}_{1,\mathcal{A}}]| \leq \epsilon_0$, where ϵ_0 is the advantage of an efficient adversary that breaks EUF-CMA security of the signature scheme S.

Game 2: We now transform **Game** 1 into **Game** 2 with the following modifications. During this game, the challenger maintains a list $\mathcal{L}$ indexed by users to record the decryption keys. More specifically, when the challenger runs the ABSVE.DKGen algorithm to generate the decryption key dk_u for a user u in response to a user registration request (by querying the oracle UserReg with a valid registration token `regA`), it records dk_u in $\mathcal{L}[u]$. Meanwhile, the challenger runs the encryption algorithm of the public key encryption scheme, PKE.Enc, to encrypt a random string of the same length as dk_u using u's public key pk_u. In the case that u is corrupted or the adversary requests to corrupt u after registration, the challenger retrieves the decryption key from $\mathcal{L}[u]$ and provides it to the adversary.

Claim 2. $|\Pr[\mathbf{Game}_{1,\mathcal{A}}] - \Pr[\mathbf{Game}_{2,\mathcal{A}}]| \leq \epsilon_1$, where ϵ_1 is the advantage of an efficient adversary that breaks IND-CPA security of the public key encryption scheme PKE.

We prove this claim by showing the construction of a distinguisher $\mathcal{D}$ between **Game** 1 and **Game** 2 given the public key and the oracle access in the IND-CPA security game for the public key encryption scheme PKE. The main idea is, $\mathcal{D}$ plants its challenge for the IND-CPA game in its simulation of secure access game for `Blind Brother` and therefore obtains a hybrid between the above two games. More specifically, $\mathcal{D}$ randomly selects a user $u \in \mathcal{U}$ and chooses a random bit b at the beginning of the simulated game. When generating the decryption key for u, $\mathcal{D}$ calls the challenge oracle in its game with a query of $(\mathsf{dk}_u, \mathsf{rs})$ to obtain a ciphertext c, where rs is a random string of the same length of dk_u. Then, $\mathcal{D}$ includes c as the ciphertext in the message as a response to $\mathcal{A}$'s query to UserReg with respect to u. Whenever $\mathcal{A}$ requests to corrupt u to learn its local state, $\mathcal{D}$ terminates the simulation and outputs 0. When $\mathcal{A}$ outputs a guess b' for the random bit b, $\mathcal{D}$ outputs 1 if $b' = b$; otherwise, it outputs 0.

We get a closer look at the distinguisher $\mathcal{D}$. If the challenge oracle in the IND-CPA game always returns an encryption of dk_u, the simulated game is identical to **Game** 1; if it always returns an encryption of rs, then the game corresponds to **Game** 2. Thus, $\mathcal{D}$ effectively simulates a hybrid game of **Game** 1 and **Game** 2. Based on the mild assumption we made regarding the non-trivial system in section 9, regardless of the file that $\mathcal{A}$ has specified as its challenge, there must be at least a user in the system has access to it. In other words, the assumption ensures that there exists at least one user who remains uncorrupted during the game and $\mathcal{D}$ has the probability at least $\frac{1}{\mathcal{U}}$ of selecting that user, which provides a lower bound on its success probability $(\frac{1}{\mathcal{U}} \cdot \epsilon_1)$. Additionally, when

$\mathcal{A}$ manages to win the game without taking over any user, $\mathcal{D}$'s probability is bounded by ϵ_1. Thus we have

$$\frac{1}{\mathcal{U}} \cdot \epsilon_1 \leq \Pr[\mathcal{D} \rightarrow 1] \leq \epsilon_1,$$

and the claim is proved.

Finally, we analyze advantage of $\mathcal{A}$ in **Game** 2. It is clear that the adversary cannot learn any part of a user's local state (specifically, the decryption key sent from **AD**) without corrupting it, as the local state is replaced with a random string which is independent of the decryption key. Furthermore, $\mathcal{A}$ cannot impersonate **AD** to have any device encrypt and upload video contents using an inappropriate key crafted by the adversary. A successful adversary in this final game would need to gain some advantage in breaking the semantic security of ABSVE.

Claim 3. $\Pr[\mathbf{Game}_{2,\mathcal{A}}] = \epsilon_2$, where ϵ_2 is the advantage of an efficient adversary that breaks semantic security of the attribute-based video encryption scheme ABSVE.

The claim can be proven by showing a construction of an adversary $\mathcal{B}$ for semantic security of ABSVE, from an adversary $\mathcal{A}$ that breaks the secure access property of `Blind Brother`. The main idea is that $\mathcal{B}$ can simulate to $\mathcal{A}$ the **Game** 2 in such a way that $\mathcal{B}$'s advantage in winning its game is exactly the same as the advantage of $\mathcal{A}$ can gain in **Game** 2. Upon receiving the master public key MPK from its challenger, $\mathcal{B}$ initialises `Blind Brother` without generating the corresponding MSK by running ABSVE.KeyGen. After that $\mathcal{B}$ starts to simulate the oracles that $\mathcal{A}$ has access to according to their specifications, exception for Challenge1. When $\mathcal{A}$ queries Challenge1 by specifying a device d, two video contents V_0, V_1 of the same length, and a policy $P \in \mathcal{P}$ as its challenge, $\mathcal{B}$ checks that if there is no corrupt user can get access the content of that entry according to P, it calls Challenge with (V_0, V_1, P) to obtain the ciphertext C and sends it with a unique video id *vid* to $\mathcal{A}$.

It is clear that the two games share the identical policy set $\mathcal{P}$, while the attribute sets of the users in `Blind Brother` are recorded in the list Ch during the semantic security game. Therefore, as long as $\mathcal{B}$ maintains the invariant that no corrupt user can get access to any of the challenged files, the game for semantic security of ABSVE will never return an error, as $P(\mathcal{Y})$ will never hold for any associated attribute set $\mathcal{Y} \in Ch$. Thus we can conclude that the simulation that $\mathcal{B}$ provides is perfect. Moreover, the simulated game is fully determined by the random bit selected in $\mathcal{B}$'s game. Therefore, the probability that $\mathcal{B}$ correctly guesses the random bit is the same as the probability that $\mathcal{A}$ wins the simulated game, and we have

$$\Pr[\mathbf{Game}_{2,\mathcal{A}}] = \epsilon_2.$$

From the Claims 1,2 and 3, we have

$$\Pr[\mathbf{Game}\ 0] \leq \epsilon_0 + \epsilon_1 + \epsilon_2.$$

In conclusion, assuming the public key encryption scheme is IND-CPA secure, the signature is EUF-CMA secure and attribute-based video encryption scheme is semantically secure, `Blind Brother` is secure with respect to video access. □

A.3 Proof of Theorem 9.3

Proof. We prove the theorem through a sequence of games, which are described in detail below. Since the reductions involved in this proof are straightforward, we will not provide a detailed description for each of them. Instead, we will discuss the general idea and the key steps involved in the reductions. Additionally, we treat the hash function as a random oracle.

Game 0: The initial game is the original experiment which defines robustness. Here and below, we denote the probability that $\mathcal{A}$ wins in game i as $\Pr[\mathbf{Game}_{i,\mathcal{A}}]$. It is important to notice that the protocol participants are assumed to have synchronized clocks. Therefore, we treat the timestamps as random strings for simplicity and do not consider their cryptographic properties. In addition, to clearly demonstrate that each of the winning conditions in the final game cannot be satisfied, we will transform the initial game in two steps.

Game 1: This game proceeds the same as the initial game, with the following modifications. Whenever the adversary queries the oracle UserInit with an initial message msg_u for a user $u \in \mathcal{D}$ that includes a valid signature not previously generated by the challenger in response for some query to the oracle UserReg, the game aborts. The validity of the signature (contained in m_3) can be verified by running the initialisation algorithm Init_U with the public parameter pp and the user's initial local state st_u.

Claim 4. $|\Pr[\mathbf{Game}_{0,\mathcal{A}}] - \Pr[\mathbf{Game}_{1,\mathcal{A}}]| \leq \epsilon_0$, where ϵ_0 is the advantage of an efficient adversary that breaks EUF-CMA security of the signature scheme S.

The claim can be proved by simply constructed an adversary $\mathcal{B}$ that breaks the game that defines EUF-CMA security, given an adversary $\mathcal{A}$ who can successfully forge a valid initial message with a signature not generated by the challenger for some user in the system. $\mathcal{B}$ can easily simulate the robustness game for $\mathcal{A}$, except that it does not run the KeyGen algorithm of the signature scheme S. Therefore, it needs to call the signing oracle to obtain signatures for the initial messages generated in response to queries to the simulated oracle UserReg. Instead, upon receiving the registration token from $\mathcal{A}$, $\mathcal{B}$ queries the signing oracle in its EUF-CMA game and attaches the obtained signature to the initial message. When $\mathcal{A}$ queries UserInit with a valid initial message msg_u for some user u, $\mathcal{B}$ takes the timestamp t and the encrypted key c_{dk_u}, along with the corresponding signature and forwards this as its output in the EUF-CMA game. Since the message was not signed by the challenger in $\mathcal{B}$'s game, $\mathcal{B}$ wins the game whenever $\mathcal{A}$ wins the simulated game, with the same probability.

Game 2: This game proceeds as the previous one, with the exception that the challenger aborts the game when the adversary queries the DeviceInit oracle with a valid initial message msg_d (containing a valid signature) for a device d, which is not generated during the setup process. The validity of the signature can be verified by running the initialisation algorithm Init_D with the public parameter pp and the device's initial local state st_d.

Claim 5. $|\Pr[\mathbf{Game}_{1,\mathcal{A}}] - \Pr[\mathbf{Game}_{2,\mathcal{A}}]| \leq \epsilon_1$, where ϵ_1 is the advantage of an efficient adversary that breaks EUF-CMA security of the signature scheme S.

The proof of this claim can be carried out similarly as in the previous one. The adversary $\mathcal{B}$ for the EUF-CMA game simulates the robustness game for the adversary $\mathcal{A}$. Since it does not possess the signing key of the challenger, $\mathcal{B}$ must query its signing oracle to

obtain the signatures for the initial messages for each device in the system during the setup process. When $\mathcal{A}$ queries DeviceInit with a valid initial message for some device d, $\mathcal{B}$ takes the timestamp and ciphertext (t, C_{K_d}, C_{V_d}), and the corresponding signature as its output. It is clear that $\mathcal{B}$'s simulation is perfect. To win the game, $\mathcal{A}$ has to come up with a valid signature for a message which is not previously signed by $\mathcal{B}$. Therefore, $\mathcal{B}$ wins the game when $\mathcal{A}$ wins the simulated game. Thus we have $|\Pr[\mathbf{Game}_{1,\mathcal{A}}] - \Pr[\mathbf{Game}_{2,\mathcal{A}}]| \leq \epsilon_1$.

Now we analyze the adversary's advantage in the final game. The adversary can win the game if either of the two winning conditions is satisfied: (1) it manages to generate an invalid video entry which was not previously generated by any device and successfully uploads it to **CSP** by calling the VideoUpload; (2) The user it specifies cannot correctly retrieve some video content uploaded by a device genuinely.

We first elaborate the second condition. Recall that the correct decryption of the video data depends on the local states of both the device uploading the video and the user accessing the video. However, after the consecutive modifications in the previous game, the adversary will not be able to manipulate the local states of either the devices or the users, since the challenger will abort the game. The correctness of the underlying cryptographic primitives guarantees that the specified authorised user must be able to correctly retrieve the video content. Therefore, the only scenario where the adversary can win the game is by calling VideoUpload to upload invalid contents, which could result in decryption failure (thereby setting the flag forged to True). Notice that if such a content is previously generated by any device in the system, it will be recorded in the list $\mathcal{V}$ and the adversary cannot win in this case. Therefore, the adversary must be able to forge a valid signature for the valid content so that it can be successfully accepted by the **CSP**.

Claim 6. $\Pr[\mathbf{Game}_{2,\mathcal{A}}] \leq \frac{1}{\mathcal{D}} \cdot \epsilon_2$, where ϵ_2 is the advantage of an efficient adversary that breaks EUF-CMA security of the signature scheme S.

Now we show that, given an adversary $\mathcal{A}$ that breaks robustness of `Blind Brother`, an adversary $\mathcal{B}$ for the EUF-CMA security game can be constructed as follows. $\mathcal{B}$ first selects a random index i from the range $\{1, ..., \mathcal{D}\}$ before simulating the robustness game for $\mathcal{A}$. Then it runs Setup initialises the cryptographic primitives employed in the protocol, with the exception that it does not generate the signing key for the device i (i.e. it is not included in the initial local state st_i) but includes the verification key it obtains from the EUF-CMA game in the public parameters. After that, $\mathcal{B}$ runs a local copy of $\mathcal{A}$ and simulates to it the oracles $\mathcal{O}_{\text{robust}}$. During the simulation, whenever $\mathcal{B}$ needs to generate a signature on behalf of device i, it queries the signing oracle in its game to obtain the signature. When $\mathcal{A}$ queries VideoUpload with a valid content which is not recorded in the list $\mathcal{V}$ for a device d, $\mathcal{B}$ aborts the game if $d \neq i$; otherwise it records the signature and stores the content in $\mathcal{V}$. When $\mathcal{A}$ specifies a user with an access request `reqV` as its output, $\mathcal{B}$ retrieves the encrypted video stream with the timestamp (t, C_{K_i}, C_{V_i}) from $\mathcal{V}$ and outputs it with the corresponding signature. It is clear that the above simulation is perfect. With the use of the signing oracle in the EUF-CMA game, $\mathcal{B}$ is able to answer all queries from $\mathcal{A}$ appropriately according to the protocol specification. Additionally, the valid forgery submitted by $\mathcal{A}$ is guaranteed not to have been generated by the device i previously. Therefore, when $\mathcal{A}$ wins the simulated game, $\mathcal{B}$ wins the EUF-CMA game.

From *Claims* 4, 5 and 6, we can conclude that

$$\mathbf{Adv}^{\text{robust-}\mathcal{D}\text{-}\mathcal{U}\text{-}\mathcal{P}}_{\text{CV},\mathcal{A}} = \Pr[\mathbf{Game}_{0,\mathcal{A}}] \leq \epsilon_0 + \epsilon_1 + \frac{1}{\mathcal{D}} \cdot \epsilon_2,$$

and the theorem is proved. □