A Fine-Grained and Real-Time Functional Video Encryption and Sharing Scheme

Haikuo Yu, Jiahui Hou, Suyuan Liu, Lan Zhang, and Xiang-Yang Li[⋆]

School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
`{yhk7786,jhhou,lsysue}@mail.ustc.edu.cn`, `zhanglan03@gmail.com`, `xiangyangli@ustc.edu.cn`

Abstract. In video-centric applications, video objects and backgrounds often contain sensitive information, which raises serious privacy concerns. It is necessary to restrict access to certain objects or backgrounds in the video stream while allowing permitted users to view a specific subset of video content. However, masking the prohibited objects for each user, then encoding and delivering each individually processed video to the target user will generate multiple copies of the same video. This can lead to significant storage, processing, and communication overhead when the number of users is large. In this work, inspired by the idea of functional encryption, we propose a fine-grained and real-time video encryption and sharing scheme, named FVES$^+$. In FVES$^+$, we encode and encrypt the videos only once, and different users can decode and decrypt the encrypted videos to acquire different contents depending on their access authorities. Theoretically, FVES$^+$ achieve IND-CPA security.
We implement and evaluate FVES$^+$ using PC and mobile devices as end-devices. FVES$^+$ achieves real-time performance, averaging 35.1 FPS across three public datasets, including the stages of object detection, encoding, and encryption. When there are n different access requirements by end-users, FVES$^+$ improves video sharing speed by $\Theta(n)\times$ compared to the baseline methods. Our experiments validate the effectiveness and efficiency of FVES$^+$.

Keywords: Video Encryption · CP-ABE · Fine-Grained Video Access Control.

1 Introduction

In the last decade, privacy concerns surrounding video footage have become increasingly significant in a variety of fields, such as elderly care, traffic surveillance, online meetings, and school safeguarding. Video streams often contain sensitive information, such as license plates, credit cards, people, or computer screens displaying confidential information, which should be safeguarded against prying eyes and malicious use [41]. Existing systems typically assign access rights

[⋆] Corresponding author

at the file level. For instance, in nursing homes, suppose Alice's family has access to video files from both Alice's room and the public areas. This coarse-grained approach presents drawbacks in both security and usability. Specifically, Alice's family can monitor Bob in the public areas, compromising Bob's privacy, while they are unable to observe Alice when she is in Bob's room, which undermines Alice's usability. In a factory environment, surveillance cameras often record both confidential industrial processes and areas requiring security oversight. In such cases, the same camera feed should appear differently to technical staff and security personnel. In privacy-sensitive, video-centric scenarios, it is essential to account for the varying access priorities of different end-users for different objects within the same video. This becomes a critical consideration for improving both security and usability.

A traditional method for protecting video privacy is to mask certain types of objects, such as faces, without considering the user's identity. Numerous deep learning methods have shown promising results, including object detection [16], blurring, inpainting [38,21], and face de-identification [37,17]. However, due to varying privacy requirements among users, it is not always feasible to summarize a common Region of Interest (ROI) list as the privacy protection target for all users [36]. Another widely used method is to protect entire video frames. Wu *et al.* [36] proposed a method to hide and recover detailed information of entire frames. Additionally, many video encryption methods [20,24,36] apply an all-or-nothing protection policy: for shared encrypted videos, either users have access to all the frames, or nothing is accessible.

Given a video stream and n users, each user has an allow list (or block list) of permitted (or prohibited) objects or backgrounds. Encrypting/masking the prohibited objects for each user, then encoding and delivering each individually processed video to the target user, will result in n unique versions of the same video. When n is large, this will result in a large storage, processing, and communication overhead. Furthermore, while frame-level desensitization/encryption generates only one version of the processed video, it cannot guarantee object-level protection granularity. To fulfill diverse and fine-grained user privileges at a small cost, we aim to design a real-time *functional encryption* [3] mechanism for video streams, in which the video-capturing-device "encrypts" the video stream only once, and multiple users acquire customized and object-grained decryption results from the *same encrypted* video stream by their own access authorities.

Our intuitive idea of functional video encryption is to divide each frame of the video into small regions as finely as possible, and each region is encoded and encrypted separately and independently. We first validate the feasibility of this idea using a basic Functional Video Encryption and Sharing (FVES) framework. FVES separates video frames into several independent-encoded and encrypted tiles. To meet the video coding standards and transmission protocols, video Selective Encryption (SE) [28,27,29], a commonly used video encryption method, is adopted. Encrypted bitstreams by SE still satisfy video coding standards, but the video content is severely distorted; FVES uses attribute-based encryption to manage the encryption keys of tiles according to the semantic information of each frame. Despite its intuitive appeal, this approach faces significant challenges in terms of security and real-time performance.

- Functional video encryption requires fine granularity of video partition to handle complex semantic information in videos, particularly in terms of the spatial relationships between objects. However, finer partitioning leads to a higher cost of encoding and encryption. Moreover, when the positions of objects overlap, the overlapping parts require new control strategies generated from the control strategies of the individual objects, resulting in high access control costs in one frame.
- Moving objects often change position rapidly within a few frames, necessitating rapid and frequent updating of control strategies. Updating real-time semantic control policy must be accomplished while maintaining high levels of forward and backward privacy.

To address the aforementioned challenges, we propose FVES$^+$, an extension of the FVES framework. First, we introduce an adaptive frame partition technique that takes into account the spatial distribution and inter-frame correlation of objects. This technique reduces and bounds the number of partitioned regions without sacrificing control granularity compared to the finest uniform partition. Second, we design an object-key generation method to bound the time complexity of access control, in which the number of keys does not fluctuate in the complexity of the spatial relationship of the objects in the video. Finally, we prove that both FVES and FVES$^+$ satisfy IND-CPA security. We also discuss how to enhance security in real-world applications, particularly in the face of hardware limitations.

We implemented and evaluated FVES$^+$ on a personal computer and three mobile platforms. We selected datasets from various scenes, including urban and indoor surveillance and automatic-driving cameras, with up to 50 objects per frame. Our experimental results demonstrate that FVES$^+$ delivers outstanding performance on videos with multiple objects and diverse privacy needs. The encryption of FVES$^+$ achieves an average of 35.1 FPS on four open datasets. FVES$^+$ speeds up the video sharing process by encoding the video only once, achieving up to $(0.58+0.31n)\times$ faster processing on average when there are n different privacy needs, compared to computer vision-based detecting-and-masking methods. Overall, FVES$^+$ enables fine-grained video access control while sustaining high efficiency in real-time streaming by addressing the complexity introduced by object motion and variability. In summary, the main contributions of this work are as follows.

- To the best of our knowledge, this is the first real-time video encryption scheme that enables different users to acquire different decryption results at the object level using a single common encrypted video.
- To achieve real-time video sharing while simultaneously considering control granularity and encryption performance, we carefully craft several innovative techniques, including a position-resilient access control mechanism in the temporal domain and an adaptive tile partitioning strategy in the spatial domain.

– We implement a prototype system and conduct comprehensive experiments to demonstrate the control granularity and efficiency of our approach. We also show that our approach achieves better encryption performance compared to existing methods in real-world applications.

2 Preliminaries and Background

2.1 Video Coding Framework and Selective Encryption (SE)

Fig. 1 shows the workflow of video encoding with SE [29,24,4]. Video coding is the process of converting a sequence of frames into a bitstream that reduces storage and bandwidth usage while preserving visual quality. Our proposed FVES and its extension, FVES$^+$, are implemented in the High-Efficiency Video Coding (HEVC) framework [10], which has become widely adopted in recent years.

Video coding starts by predicting pixel values based on intra-frame or inter-frame similarities and calculating residuals between predicted and actual values. Each frame is divided into Coding Tree Units (CTUs) of size 64×64 pixels. These CTUs are further subdivided into Coding Units (CUs) of varying sizes, from 8×8 to 64×64 pixels. For each CU, HEVC determines the prediction mode (either intra-frame or inter-frame), calculates the predicted values and residuals, and applies the Discrete Cosine Transform (DCT) to the residuals before quantizing them into coefficients. The resulting syntax elements are binarized and encoded into the video stream using entropy coding. Before encoding, these elements can be encrypted, such as via XOR operations.

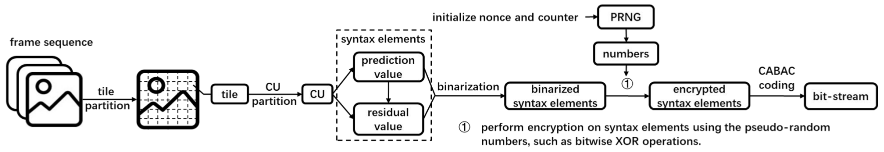
initialize nonce and counter
PRNG
numbers
frame sequence
tile partition
tile
CU partition
CU
syntax elements
prediction value
residual value
binarization
binarized syntax elements
①
encrypted syntax elements
CABAC coding
bit-stream
① perform encryption on syntax elements using the pseudo-random numbers, such as bitwise XOR operations.

Fig. 1. Selective encryption diagram in HEVC coding.

Tile partition: HEVC uses tiles to enable parallel encoding. A tile is a rectangular group of CTUs, and each frame can be divided into multiple tiles, with each tile encoded independently. Tile sizes can vary, but adjacent tiles must align in width and height according to the HEVC standard. Since tiles are the smallest independent coding unit, we use them as the minimum control granularity in our scheme.

Video Selective Encryption: Before video coding, the SE scheme initializes a pseudo-random number generator (PRNG) using a secret nonce and a counter, typically in AES-CTR mode [14]. During coding, before the video encoder writes the binarized syntax elements to the bitstream, the SE scheme uses the PRNG to generate a pseudo-random number for each element. This pseudo-random number is then used to encrypt the syntax elements, typically through bitwise XOR, while ensuring compliance with the coding standard. The counter is incremented after each encryption.

Note that the complete video bitstream is not composed solely of tile bitstreams; it also includes necessary information for decoding, such as tile partitioning. This non-encrypted format ensures that the SE scheme does not encrypt every bit, maintaining real-time performance and the structure required for video transmission protocols. The encrypted video stream can still be decoded by an HEVC decoder, but the content appears distorted. Traditional SE schemes encrypt frames using a single nonce, but in our approach, each tile is encrypted with a unique nonce, which serves as the key for that tile. The formal definition of SE is outlined as follows:

– $\texttt{Setup}(\lambda) \to K$: The `Setup` takes a security parameter λ as input. It outputs a symmetric key $K \in 0, 1^{\lambda}$.
– $\texttt{Encrypt}(T, K, C) \to C_T$: The `Encrypt` takes as input the tile T, a key K, and a counter C. It outputs the encrypted bitstream C_T.
– $\texttt{Decrypt}(C_T, K, C) \to T'$: The `Decrypt` takes as input an encrypted bitstream of a tile C_T, the key K, and a counter C. It outputs a tile T'.

SE is a symmetric encryption scheme. The security of SE relies on the security of the pseudo-random number stream. In our implementation, we utilize AES-CTR mode to generate pseudo-random numbers, which have been proven secure when using 192-bit or 256-bit keys [14]. SE encrypts the syntax elements. Therefore, if all the CUs in the video are encoded using syntax elements of the same type and length, and the PRNG used in SE is secure, then SE achieves ciphertext indistinguishability.

2.2 Ciphertext-Policy Attribute-Based Encryption (CP-ABE)

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [2,35] is a cryptographic scheme that provides fine-grained access control by associating a user's identity with a set of attributes. In CP-ABE, users' identities are characterized as a set of attributes drawn from the system's attribute universe. Access control policies are formulated by combining attribute values using Boolean logic operations and threshold conditions. A plaintext encrypted under a policy can be decrypted only if users' attribute sets satisfy the policy. A CP-ABE scheme, CPABE=(`Setup`, `KeyGen`, `Encrypt`, `Decrypt`), has four algorithms:

– $\texttt{Setup}(\lambda) \to (MPK, MSK)$: The `Setup` takes a security parameter λ as input. It outputs the master public-secret key pair (MPK, MSK).
– $\texttt{Encrypt}(MPK, \mathbb{A}, M) \to C_{\mathbb{A}}$: The `Encrypt` takes the master public key MPK, an access policy $\mathbb{A}$, and a plaintext M as input. It outputs the ciphertext $C_{\mathbb{A}}$.
– $\texttt{Decrypt}(SK_S, C_{\mathbb{A}}) \to M'$: The `Decrypt` takes as input a secret key SK_S associated with an attribute set S and the ciphertext $C_{\mathbb{A}}$ encrypted under an access policy $\mathbb{A}$. It succeeds and outputs a plaintext M' if S satisfies $\mathbb{A}$. Otherwise, it aborts.

If the decryption of a ciphertext yields the original plaintext, we call the CP-ABE scheme correct. In our construction of FVES and FVES$^+$, we employ CP-ABE as a foundational mechanism.

3 Threat Model and System Overview

3.1 System Model

We design a real-time system that allows numerous video consumers to subscribe to video streams with object-level granularity in a privacy-preserving way. Fig. 2 shows the architecture used for building a functional video encryption and sharing scheme. In our system, there are three entities: the video owner (VO), video consumers (VC), and a trusted authority (TA).

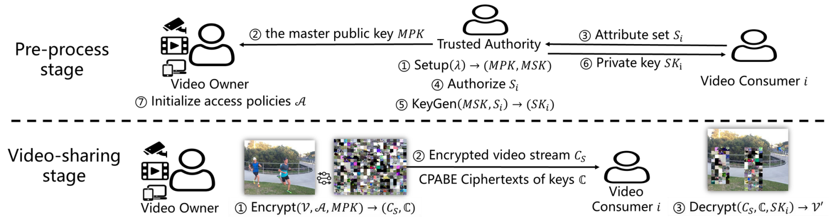
Pre-process stage
② the master public key MPK
Trusted Authority
③ Attribute set S_i
⑥ Private key SK_i
Video Owner
① Setup(λ) → (MPK, MSK)
④ Authorize S_i
⑤ KeyGen(MSK, S_i) → (SK_i)
Video Consumer i
⑦ Initialize access policies $\mathcal{A}$
Video-sharing stage
Video Owner
① Encrypt($\mathcal{V}, \mathcal{A}, MPK$) → ($C_S, \mathbb{C}$)
② Encrypted video stream C_S
CPABE Ciphertexts of keys $\mathbb{C}$
Video Consumer i
③ Decrypt($C_S, \mathbb{C}, SK_i$) → $\mathcal{V}'$

Fig. 2. System model of FVES. In the pre-processing stage, the TA sets up the system and sends the master public key to the VO. VCs send their attribute sets to the TA, which then provides them with their corresponding private keys. In the video-sharing stage, the VO encrypts the video frames and sends the ciphertext to the VCs.

- **VO**. The VO is the owner of the videos to be shared. In many cases, the VO is the owner of video-capturing devices, such as cameras and personal computers. Videos may contain private elements, and video owners do not trust the networks and video consumers. They know the access policies of video elements and encode and encrypt the raw video data into an encrypted video stream, then push the encrypted video stream to networks. VO's devices are capable of video information extraction, video coding, and encryption.
- **VC**. VCs receive encrypted video streams and perform object-level decryption based on their access authorities in each frame. A VC is capable of video decoding and decryption using an end device, e.g., a mobile device.
- **TA**. A TA is only introduced to authorize the attributes of VCs. According to the granted attributes, it generates the private keys for the VCs.

3.2 Formal Definition

Before providing the formal definition of our FVES scheme, we first define the symbols used within the scheme. Given N sequential frames $\mathbb{V} = \{V^1, V^2, ..., V^N\}$, each frame $V^k, k \in [1, N]$ is represented as a $w \times h$ matrix. Each frame can be partitioned into t_k rectangular tiles, denoted by $\mathbb{T}_k = \{T_1, T_2, ..., T_{t_k}\}$, where each tile corresponds to a distinct rectangular region of the frame. Fig. 3 illustrates an example where a frame is evenly divided into tiles of equal size. We assume that there are $c - 1$ objects in the video that require access control, while the remaining parts of the video are considered the background. The objects and the background are collectively denoted as $\mathbb{E} = \{e_1, e_2, \cdots, e_c\}$. For instance, Fig. 3 depicts a frame with four elements: two people, a car, and the background. As illustrated, each object is covered by a specific set of tiles, while tiles that do not cover any object correspond to the background.

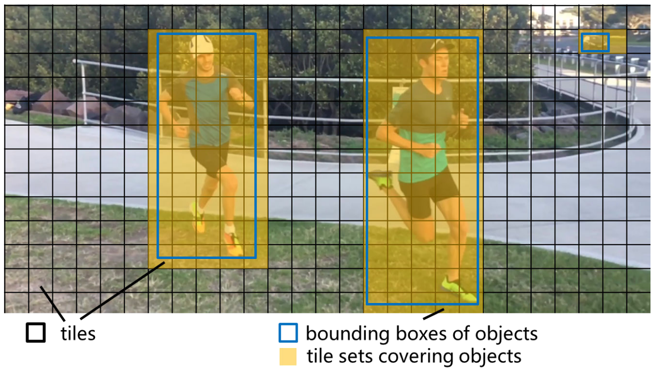

Fig. 3. A frame is partitioned into rectangular tiles. Each object is covered by a set of tiles, and we denote tiles that do not cover any object as the cover of the background.

There are n end VCs $\mathbb{U} = \{u_1, u_2, \cdots, u_n\}$ who require the video stream. The access permissions for each VC are as granular as each element within a frame. Each VC i is characterized by a set of attributes S_i. For instance, $S_i = \{\text{school of computer science, Professor, leader of a surveillance camera project}\}$ might define VC i's role, which determines the VC's authority to access specific elements of the video stream. We assume that the VO knows the access policy of each element in $\mathbb{E}$. Based on whether S_i satisfies the access policies, we define $\mathbb{E}_i \subseteq \mathbb{E}$ as the allow-list that VC i is permitted to view. $\mathbb{E} \backslash \mathbb{E}_i$ forms the block-list, which contains elements that VC i is not permitted to access.

Each frame V^k is partitioned into tiles $\mathbb{T}_k$. According to $\mathbb{E}_i$, the content visible to VC i from frame V^k is a subset of these tiles, denoted as $\mathbb{T}_{k,i} \subseteq \mathbb{T}_k$. $\mathbb{T}_{k,i}$ is a cover on $\mathbb{E}_i$. The video content of the remaining tiles, $\mathbb{T}_k \backslash \mathbb{T}_{k,i}$, including the object category, outline, color, and behavior, cannot be discerned by i.

Based on the system model and the defined symbols, our FVES scheme consists of four algorithms, which can be described as follows:

- $\texttt{Setup}(\lambda) \rightarrow (MPK, MSK)$: The `Setup` takes a security parameter λ as input. It outputs the master public-secret key pair (MPK, MSK).
- $\texttt{KeyGen}(MSK, S_i) \rightarrow SK_i$: The `KeyGen` takes as input the master secret key MSK and S_i. It outputs a secret key SK_i. The authority sends SK_i to the video consumer i.
- $\texttt{Encrypt}(\mathcal{V}, \mathcal{A}, MPK) \rightarrow (C_S, \{C_{k,j}\})$: The `Encrypt` takes as input the video frame sequence $\mathcal{V}$, the access policies $\mathcal{A}$ of elements, and the master public key MPK. It outputs the encrypted bitstream C_S and CP-ABE ciphertexts of keys $\{C_{k,j}\}$ for each frame V^k and element e_j.
- $\texttt{Decrypt}(C_S, \{C_{k,j}\}, SK_i) \rightarrow \{V'^1, V'^2, \cdots, V'^N\}$: The `Decrypt` takes as input the encrypted bitstream C_S, the CP-ABE ciphertexts of keys $\{C_{k,j}\}$, and secret key SK_i. It outputs a frame sequence $\{V'^1, V'^2, \cdots, V'^N\}$.

As shown in Fig. 2, during the pre-processing stage, the TA runs $\texttt{Setup}(\lambda) \rightarrow (MPK, MSK)$, generating the master public key MPK and the master secret key MSK, and sends MPK to the VO. Each VC i sends its attribute set S_i to the TA, which then authorizes S_i and runs $\texttt{KeyGen}(MSK, S_i) \rightarrow SK_i$, generating the private key SK_i and sending it to VC i. The VO then initializes the access policies $\mathcal{A}$ for the elements.

In the video-sharing stage, the VO has a sequence of video frames. The VO runs $\texttt{Encrypt}(\mathcal{V}, \mathcal{A}, MPK) \rightarrow (C_S, \{C_{k,j}\})$ to generate the encrypted video stream C_S and the CP-ABE ciphertexts $\{C_{k,j}\}$. The VO then sends C_S and $\{C_{k,j}\}$ to each VC i. Each VC i runs `Decrypt`. If S_i satisfies the access policies of the elements, VC i can retrieve the original pixel data of the corresponding tiles in $\{V'^1, V'^2, \ldots, V'^N\}$.

3.3 Security Model

We assume that the TA is fully trusted. The encoding and encryption of the original video frames are performed by a trusted device, as all computations are carried out locally by the VO. We assume that VCs have the motivation to infer the protected video content. Legitimate consumers can potentially be attackers with the goal of accessing video content beyond their access permissions.

For each video, we allow VCs to know the total number of objects and their positions in each frame, as well as whether any objects overlap. This assumption is reasonable in real-world scenarios. As frames are decrypted according to the permissions granted to the VC, the VC may infer the presence of objects, their positions, and the potential overlaps by observing the areas that they cannot decrypt. For example, as illustrated in Fig. 2, the decrypted frame indicates that at least three objects remain encrypted, while the VC does not have access to their original content. We also assume that when an object moves from one position to another, the access policy of the same object remains unchanged. The critical information we aim to protect includes objects and backgrounds within the video, encompassing pixel values, textures, contours, and behavior.

Access Policies of Overlapped Objects. Before detailing the formal security definition of FVES and FVES$^+$, it is crucial to clarify the access policies of the overlapping regions of multiple objects. In our system model, the access policies for objects and their overlapping regions are determined by the VO. We assume that the VO is aware of the access policies for objects and backgrounds. For regions where the pixels of multiple objects overlap, a VC can access overlapping regions if at least one of the object policies is satisfied. This ensures VCs can access the entire object, which is necessary for the usability of tasks like human inspection or machine learning.

Security of FVES and FVES$^+$. Formally, the security of the schemes is defined using an IND-CPA game between a challenger $\mathcal{C}$ and an attacker as follows:

- **Setup:** The challenger runs $\texttt{Setup}(\lambda)$ to obtain the master key-pair MPK and MSK. The MPK is given to the attacker.

- **Key Generation Queries:** The attacker adaptively issues private key queries by submitting attribute sets $\mathcal{S}_i$ to $\mathcal{C}$. $\mathcal{C}$ gives the corresponding private key SK for the attribute set $\mathcal{S}_i$ to the attacker.
- **Challenge:** The attacker submits two plaintext videos, denoted as $\mathcal{V}_0$ and $\mathcal{V}_1$. Both $\mathcal{V}_0$ and $\mathcal{V}_1$ are of the same size and contain the same number of objects, represented as $e_0^1, e_0^2, \ldots, e_0^c$ in $\mathcal{V}_0$ and $e_1^1, e_1^2, \ldots, e_1^c$ in $\mathcal{V}_1$, where for each object i, e_0^i and e_1^i cover the tiles with the same position in their respective videos. The objects accessible to the attacker are identical across $\mathcal{V}_0$ and $\mathcal{V}_1$. There exists at least one object $e_b^i, b \in \{0, 1\}$ for which the attacker's attribute set does not satisfy the corresponding access policy. These tiles covered by $e_b^i, b \in \{0, 1\}$ contain arbitrary pixels. $\mathcal{C}$ then selects a random bit $b \in \{0, 1\}$ and encrypts the video $\mathcal{V}_b$. The resulting ciphertext C is returned to the attacker.
- **Phase 2:** The attacker continues to issue key generation queries for different attribute sets. However, the restriction is that the attacker must not query private keys for any attribute set that satisfies any of the access policies associated with the challenge ciphertexts.
- **Guess:** The attacker outputs a guess $b' \in \{0, 1\}$. If $b' = b$, the attacker wins the game.

The advantage of the attacker in breaking the IND-CPA security of the FVES scheme is defined as:

$$\mathcal{A}dv_{\text{FVES}}^{\text{IND-CPA}} = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

The FVES scheme is said to be IND-CPA secure if, for any probabilistic polynomial-time adversary, the advantage $\mathcal{A}dv_{\text{FVES}}^{\text{IND-CPA}}$ is negligible in the security parameter λ.

4 Construciton of FVES

In this section, we present the construction of the encryption and decryption algorithms in FVES, followed by a detailed analysis of their efficiency.

4.1 Encryption Algorithm of FVES

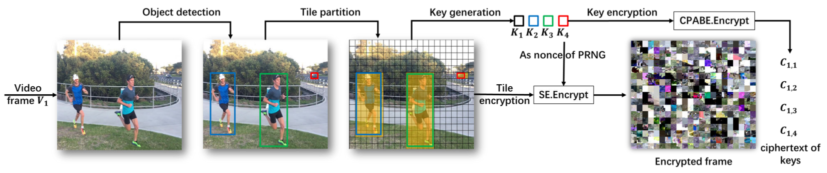

Fig. 4. A schematic diagram of the encryption algorithm in FVES.

The FVES.`Encrypt` algorithm is executed by the VO. As shown in Fig. 4, the high-level workflow of the algorithm consists of four steps:

- **Object detection.** This step detects objects in each frame and marks their bounding boxes.

- **Tile partition.** The video frames are divided into tiles (with boundary adjustments if necessary), and each object, based on the bounding boxes, has a tile cover. Each overlapped region of objects is treated as a new tile cover.

- **Video encoding with SE.** Each tile cover is assigned a random key, which is used to encrypt its tiles using SE.`Encrypt`. SE.`Encrypt` is primarily based on bitwise XOR, resulting in minimal additional time overhead.

- **Key encryption.** Each key generated for encrypting an element and the four coordinates of the corresponding tile cover are concatenated as a message. The message is encrypted using CPABE, based on the corresponding access control policies of the elements.

Algorithm 1: FVES.`Encrypt`$(\mathcal{V}, \mathcal{A}, MPK) \rightarrow (C_S, \{C_{k,j}\})$

```
Input: Frame sequence $\mathcal{V} = \{V^1, V^2, \ldots, V^N\}$, Access strategies of
       video element universe $\mathcal{A} = \{\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_c\}$, The master public
       key $MPK$.
Output: Encrypted video stream $C_S$, CPABE ciphertexts of keys
        $\{C_{k,j}\}$ for each frame $V^k$ and element $e_j$.
1  Step 1: Object Detection;
2  foreach $V^k \in \{V^1, V^2, \cdots, V^N\}$ do
3      Detect each object $o$ in $V^k$, bounding box of $o$ is $[w_1^o, w_2^o, h_1^o, h_2^o]$;
4      Label object-free regions as background;
5  Step 2: Tile Partition;
6  Initialize the video stream $C_S$ and the key-counter dict $D$;
7  foreach $V^k \in \{V^1, V^2, \cdots, V^N\}$ do
8      Divide frame $V^k$ of size $w \times h$ into tiles of size $w_t \times h_t$ For each
        object $o$, relax $[w_1^o, w_2^o, h_1^o, h_2^o]$ into tile-aligned bounds;
9      Classify each tile $T$ into different tile sets $\mathbb{T}_{k,j}$ based on the objects
          or background it covers. // Overlapping regions are
          treated as separate sets.
10 Step 3: Video Encoding with SE;
11 foreach $V^k \in \{V^1, V^2, \cdots, V^N\}$ do
12     Encode the tile partitioning of $V^k$ with the HEVC encoder.;
13     foreach $\mathbb{T}_{k,j}$ do
14         Generate a random key $K_j$;
15         Set the value $D\{K_j\}$ in key-counter dict as 0;
16     foreach $T \in \mathbb{T}_{k,j}$ do
17         Encode $C_T = $ SE.Encrypt$(T, K_j, D\{K_j\})$ in $C_S$;
18         Update the counter $D\{K_j\}$;
19     Finish encoding $V^k$;
20 Step 4: Key Encryption;
21 foreach $\mathbb{T}_{k,j}$ do
22     Concatenate the coordinates of the bounding box of $\mathbb{T}_{k,j}$ into integer
        $L_j$, concatenate $L_j$ and $K_j$: $L_j || K_j$;
23     $C_{k,j} = $ CPABE.Encrypt$(MPK, \mathbb{A}_j, L_j || K_j)$;
24 return $C_S$, $\{C_{k,j}\}$;
```

One tile T may be covered by several tile cover sets, as the bounding boxes of different objects may overlap, or their borders may fall within the same tile. We treat the overlapping region as a new tile cover, combining the access policies of the overlapping objects to generate a new access policy for this tile cover. The tiles within this tile cover are then encrypted in the same manner as above. Algorithm 1 shows the description for FVES.`Encrypt` algorithm.

Object overlapping leads to an increase in the number of keys and the execution times of CPABE.`Encrypt`, which in turn constrains the encryption efficiency of FVES. For each frame, the upper bound of the execution times of

CPABE.`Encrypt` is equivalent to the number of tiles, denoted as c:

$$c \leq \left\lceil \frac{w}{w_t} \right\rceil \times \left\lceil \frac{h}{h_t} \right\rceil \tag{1}$$

In FVES$^+$, we reduce this bound, thereby improving the encryption efficiency.

4.2 Decryption algorithm of FVES

Each VC with an attribute set S receives the PRNG $\mathbb{G}$ and the private key SK_S from the TA during the pre-processing stage. In the video-sharing phase, each VC receives the encrypted video stream C_S and the ciphertexts of element keys $C_{k,j}$ for each frame V^k and element e_j from the VO. For each VC i, the FVES.`Decrypt` algorithm is executed following the steps outlined in Algorithm 2 to obtain the functionally decrypted video frames.

Algorithm 2: FVES.`Decrypt`$(C_S, \{C_{k,j}\}, SK_S) \rightarrow \mathcal{V}'$

```
Input: Encrypted video stream $C_S$, ciphertexts $C_{k,j}$ of the element keys for
       each frame $V^k$ and element $e_j$, and the secret key $SK_S$
Output: A sequence of frames $\mathcal{V}' = \{V'^1, V'^2, \cdots, V'^N\}$
1  Decode the tile partition information and bitstream of each frame $V^k$ from $C_S$
     using the HEVC decoder;
2  Initialize the key-counter dictionary $D$;
3  foreach bitstream of $V^k \in \{V^1, V^2, \cdots, V^N\}$ do
4      foreach ciphertext $C^k_{\mathbb{A}_j}$ do
5          Run CPABE.Decrypt$(SK_S, C^k_{\mathbb{A}_j})$;
6          if the attribute set $S$ satisfies the access structure $\mathbb{A}_j$ then
7              CPABE.Decrypt outputs $L_j||K_j$;
8              Set the value $D\{K_j\}$ in the key-counter dictionary as 0;
9      Determine the encrypted tile sets $\mathbb{C}_{\mathbb{T}k,j}$ and corresponding $K_j$ from
         decrypted $K_j$ and $L_j$;
10     Decode each encrypted tile $C_T$ from the bitstream of $V^k$;
11     foreach $C_T \in \mathbb{C}_{\mathbb{T}k,j}$ do
12         SE.Decrypt$(C_T, K_j, D\{K_j\}) \rightarrow T'$;
13         Update the $D\{K_j\}$;
14     Reconstruct $V'^k$ using all tiles;
15 return $\mathcal{V}'$;
```

4.3 Efficiency Analysis of FVES

In this section, we analyze the time overhead of FVES encryption in both the spatial and temporal domains.

First, we calculate the average time overhead per frame for encrypting 1920×1080 resolution videos when running FVES on a PC using the latest CP-ABE implementation proposed by [30], the average running time of CPABE.`Encrypt` per frame is approximately two seconds. This constitutes the majority of the encryption time, which is insufficient to meet real-time processing requirements. Other operations take only 0.017 seconds on average.

In the spatial domain, Eq 1 shows that the upper bound of ABE encryption executions per frame depends on the number of tiles rather than objects. This is because the overlap between objects leads to the generation of new keys and corresponding ABE encryptions. However, there is a substantial difference between the number of tiles and the number of objects. For videos with a resolution of 1920×1080, the number of 64×64 tiles can be calculated as $\lceil \frac{1920}{64} \rceil \times \lceil \frac{1080}{64} \rceil = 510$. In contrast, our evaluation of the average number of objects in two open datasets and a real surveillance video captured in an indoor office reveals that the upper bound of the number of objects per frame does not exceed 60, which is significantly lower than the number of tiles, i.e., 510. While increasing the tile size reduces the number of tiles and thus reduces the encryption time, it also leads to a more coarse-grained partitioning of tiles, reducing control granularity. Therefore, it is essential to identify a method that reduces the number of tiles without sacrificing control granularity.

In the temporal domain, object movement results in changes to the pixel positions of object bounding boxes in each frame. According to Algorithm 1, both the key of the object and its position must be re-encrypted using CPABE for the same object in every frame. To enhance efficiency, it is essential to develop a method that encrypts an object's key with CPABE only once, from its appearance to disappearance in the video.

5 FVES$^+$: Refinement and Enhancement

In this section, we further propose FVES$^+$ to address the efficiency issues of FVES. FVES$^+$ adheres to the FVES scheme described above, with significant improvements focusing on the efficiency of the `Encrypt` and `Decrypt` algorithms.

5.1 Adaptive Tile Partitioning

An increase in the number of tiles harms video encoding and encryption speed, which motivates us to find an adaptive tile partitioning method to decrease the number of tiles without affecting the control granularity.

The number of tiles in a uniform tile partition is related to the resolution of frames and the size of CTUs, while the number of tile cover sets is related to the number of elements. As shown in Fig. 5, we set the boundaries of the tile cover set as the boundaries of the new tile grid. For k elements, the tile grid has at most $2k$ horizontal lines and $2k$ vertical lines, which leads to $(2k+1)^2$ tiles.

In the worst-case scenario of adaptive tile partitioning, each tile could potentially contain a separate object. Thus, the upper bound of the number of tiles with adaptive tile partitioning is $O\left(\left\lceil \frac{w}{w_t} \right\rceil \times \left\lceil \frac{h}{h_t} \right\rceil\right)$. However, based on our analysis of four real-world datasets, for a 1920×1080 frame, the average number of tiles generated by adaptive tile partitioning is approximately 109. In comparison, uniform tile partitioning produces 510 tiles per frame. This demonstrates that adaptive tile partitioning significantly reduces the number of tiles while maintaining the granularity of access control.

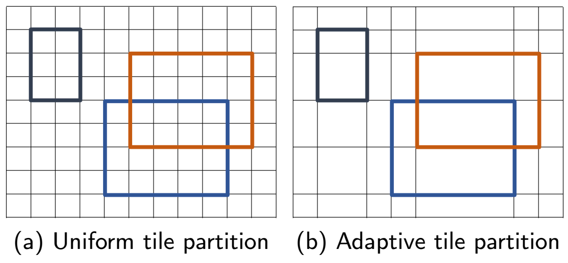

Fig. 5. Adaptive tile partition and uniform tile partition with the same control granularity. The grids of thin lines are the tiles of the smallest size, and thick boxes denote the bounding boxes of objects.

Additionally, adaptive tile partitioning improves the performance of video encoding compared to uniform tile partitioning. Since the encoding of tiles is independent of each other, tile boundaries disrupt coding dependencies. As the number of tile boundaries is reduced, better coding performance is achieved.

5.2 Position-Resilient Access Control

As analyzed in Section 4.3, FVES.`Encrypt` encrypts the position and the SE key of objects and overlapped regions by CPABE in each frame. FVES$^+$.`Encrypt` algorithm seeks to optimize efficiency by calling CPABE.`Encrypt` only once per object, avoiding redundant re-encryption caused by object movement or overlapping throughout its appearance in the video.

We now propose a new key generation method for FVES$^+$.`Encrypt`. A PC with any single object's key can derive the key for the overlapping region. Our core idea is to use the shared common substrings of the object keys as the key for the overlapping pixel region. To achieve this and ensure the security of the keys, we require the object keys to satisfy the following properties:

Property 1. 1) The object keys are all unique. 2) Any group of keys contains a common substring, and different groups of keys contain different substrings. 3) It is computationally hard to infer any keys and common substrings of any group of keys if the keys in the group are unknown.

Given a set of objects $\{o_1, o_2, \ldots, o_c\}$, each object has a key. Each key K_j of object o_j is constructed by concatenating d base substrings, where each substring is an l-length bit-string. If a tile T covers multiple objects, we denote them as $\{o_1, o_2, \ldots, o_{c_1}\}$. The shared common base substrings from all object keys $K_j \mid j \in [1, c_1]$ are used as the SE key K_T for T. To identify the positions of these common base substrings, we use a d-length vector p_T, defined as follows:

$$p_T[h] = \begin{cases} 1, & \text{if the } h\text{-th base substrings are the same in all keys} \\ 0, & \text{otherwise,} \end{cases}$$

p_T is part of the output from the FVES$^+$.`Encrypt` algorithm, indicating the positions of the common base substrings in the keys K_j. The key generation process is outlined in Algorithm 3.

Algorithm 3: Key Generation method in FVES$^+$.

```
Input: elements $\{e_1, e_2, \ldots, e_c\}$, the length of each base substring $l$, the
       number of base substrings $d > c$, parameter $n > c$
Output: keys of objects $\{K_1, K_2, \ldots, K_c\}$
1  Randomly generate an $l$-length bit-string $s_0$;
2  Initialize $K_1 = K_2 = \cdots = K_c = s_0$;
3  for $i \in [1, c]$ do
4      Randomly generate an $l$-length bit-string $s_i$ and $r_i$;
5      $K_i = K_i || s_i$. // $||$ denotes concatenation of two bit-strings
6      foreach $K_j$ where $j \neq i$ do
7          $K_j = K_j || r_i$;
8  for $i \in [c+1, d]$ do
9      Randomly generate $n$ $l$-length bit-strings $\{s_1, \ldots, s_n\}$;
10     for $h \in [1, c]$ do
11         Randomly select $b \in \{1, 2, \ldots, n\}$;
12         $K_h = K_h || s_b$;
13 Generate a random permutation $\mathbb{O}$ of integers from 1 to $d$, rearrange the base
     substrings in each key using $\mathbb{O}$;
14 return $\{K_1, K_2, \ldots, K_c\}$;
```

First, after executing loop 3- 7, each key K_i consists of $c+1$ base substrings. All keys share the same first base substring s_0, ensuring that any group of keys will have at least one shared common substring. From the second to the $(c+1)$-th base substrings, the keys are constructed in a manner analogous to an identity matrix. Fig. 6 illustrates an example when $c = 4$. Next, the loop from lines 8 to 12 generates n random substrings for each base substring. Each key randomly selects one of these substrings as its i-th base substring. This ensures that different groups of keys have distinct common base substrings. Thus, the keys generated by Algorithm 3 satisfy Property 1.

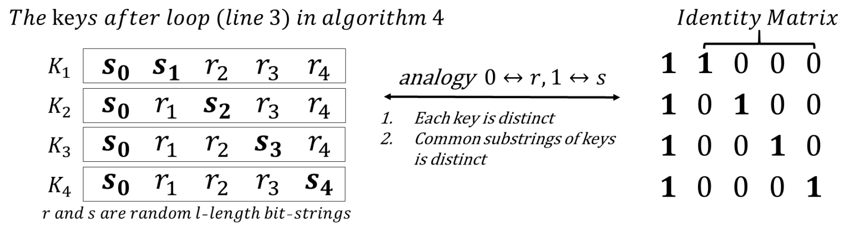
The keys after loop (line 3) in algorithm 4
K_1 s_0 s_1 r_2 r_3 r_4
K_2 s_0 r_1 s_2 r_3 r_4
K_3 s_0 r_1 r_2 s_3 r_4
K_4 s_0 r_1 r_2 r_3 s_4
r and s are random l-length bit-strings
analogy $0 \leftrightarrow r, 1 \leftrightarrow s$
1. Each key is distinct
2. Common substrings of keys is distinct
Identity Matrix
1 1 0 0 0
1 0 1 0 0
1 0 0 1 0
1 0 0 0 1

Fig. 6. An example of keys after loop (line 3- 7) in Algorithm 3 when $c = 4$.

We now prove that, when n and d are sufficiently large, it is computationally hard to infer the common base substrings of any two unknown keys. Consider the worst case where an attacker knows the keys of objects $\{o_3, \ldots, o_c\}$, with $c > 3$, and all the common substrings between these keys and the keys of objects

o_1 and o_2. We aim to show that it is computationally hard to determine K_1, K_2, and the common base substrings of K_1 and K_2.

At each position h in the base substrings from position $c+2$ to d, the base substrings $\{K_1[h], K_2[h], \ldots, K_c[h]\}$ are randomly selected from a set of n random l-length bit-strings. Treating c as a constant, the probability Pr that $K_1[h]$ is equal to any of $\{K_3[h], \ldots, K_c[h]\}$ is

$$\Pr(K_1[h] \in \{K_3[h], \ldots, K_c[h]\},\ c > 3) = \left(1 - \frac{(n-1)^{c-2}}{n^{c-2}}\right) = O\left(\frac{1}{n}\right).$$

Similarly, the probability that $K_2[h]$ is equal to any of $\{K_3[h], \ldots, K_c[h]\}$ is $O\left(\frac{1}{n}\right)$. As a result, the probability that an attacker can deduce $K_1[h]$ and $K_2[h]$ at all positions $h \in [c+2, d]$ is $O\left(\frac{1}{n^{2(d-c-1)}}\right)$. When n and d are sufficiently large, this probability becomes negligibly small, making it computationally infeasible to obtain K_1 and K_2 in the worst case. Consequently, it is also difficult to recover the common substrings of K_1 and K_2.

The key generation method ensures that the number of CPABE.`Encrypt` executions remains unaffected by object movement or overlap, being influenced solely by the number of objects in FVES$^+$.`Encrypt` algorithm.

6 Security Analysis

In this section, we present a formal security proof of the FVES and FVES$^+$ schemes. We also provide visual results of encrypted and decrypted frames.

Theorem 1. *If the CPABE and SE schemes utilized in FVES and FVES$^+$ are IND-CPA secure, then the FVES and FVES$^+$ schemes are also IND-CPA secure.*

Proof. In the high-level workflow, both FVES and FVES$^+$ employ SE to encrypt individual video tiles. SE is a symmetric encryption scheme, where the keys for each tile are assigned based on the objects covered by the tiles. These keys are managed by the CP-ABE scheme according to access policies corresponding to the objects. The encryption process is performed locally on the VO's device.

For tiles covering multiple objects, FVES generates a new random SE key, whereas FVES$^+$ uses the bitwise XOR of the SE keys of the covered objects as the SE key for the tile. In the key generation method of FVES$^+$, bitwise XOR ensures that the new key is indistinguishable from random as long as the key of any individual object remains unknown.

Assume that there exists an adversary $\mathcal{A}dv_1$ capable of breaking the IND-CPA security of the FVES or FVES$^+$ schemes with a non-negligible advantage. In such a case, an adversary $\mathcal{A}dv_2$ could leverage $\mathcal{A}dv_1$'s ability to compromise the FVES or FVES$^+$ security to break the IND-CPA security of either the CP-ABE or SE scheme. However, since both the CP-ABE and SE schemes are assumed to be secure, it follows that the FVES and FVES$^+$ schemes must also maintain IND-CPA security.

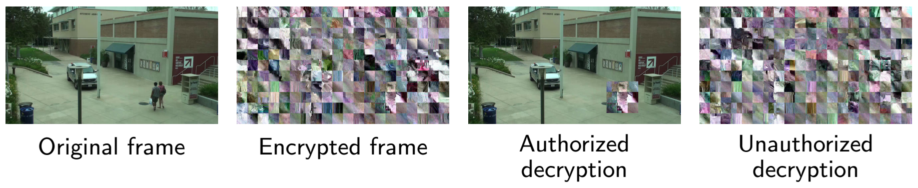

Fig. 7. Examples of the decryption results using the right keys and the wrong keys.

Besides providing formal proof, we also visually demonstrate the security of FVES and FVES$^+$. Fig. 7 presents an example of decryption using incorrect keys, resulting in persistent distortion of the decrypted video content.

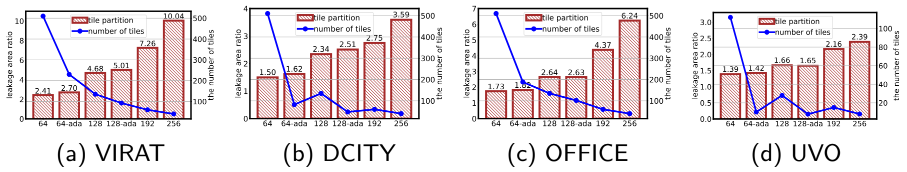

Fig. 8. The number of tiles and the ratio of the tile-covered area to the tight bounding box area. $64, 128, 256$ denote uniform tile partition with tile sizes $64, 128, 256$, 64-ada and 128-ada denote adaptive tile partition using a granularity of 64 or 128.

In our security model, the granularity of access control is at the tile level. At this granularity, we prove that FVES and FVES$^+$ are secure. As shown in Fig. 3, a small pixel difference exists between an object's tight bounding box and its tile-aligned bounding box. To quantify this, we calculate the ratio of the tile-covered area to the tight bounding box area across four video datasets. Our experimental results show that this pixel difference is minimal. When the tile size is set to 64, the ratio is only 1.75% (Fig. 8). Moreover, adaptive tile partitioning achieves a similar ratio to the uniform tile partition while significantly reducing the number of tiles.

7 Evaluations

In this section, we conduct an extensive evaluation to measure the efficiency and privacy protection performance of FVES$^+$ on different devices.

7.1 Evaluation Setup

Device. We use two kinds of devices to perform video owners. A Personal Computer (PC) with 12 Intel Cores i5-11500 (2.70GHz) and one NVIDIA GTX 3060 GPU; and a Jetson Nano (Dual-Core NVIDIA Denver 2 64-Bit + Quad-Core ARM Cortex-A57 MPCore CPU). We use the same PC and Jetson Nano, and two mobile devices 1). Xiaomi 10, 2). Huawei P10 as VCs' devices.

Datasets. We use the same four datasets as introduced in Sec. 6. We use three metrics to assess the complexity of video content in Tab. 1: 1) the average number of objects (ANO), ANO indicates the average quantity of tiles in FVES$^+$; 2) the object change frequency (OCF): the average number of frames in which the number of objects in the frames remains unchanged; 3) the object moving rate (OMR): the average pixel euclidean distance of the object bounding box center between consecutive frames. OCF and OMR indicate the updating frequency of object access strategies.

Table 1. Video datasets and complexity of privacy requirements.

dataset	scene	camera	ANO	OCF	OMR
VIRAT	urban	stationary	18.98	15.02	19.82
DCITY	road	moving	8.47	9.71	46.24
OFFICE	office	stationary	50.28	43.65	159.89
UVO	indoor	moving	11.36	38.66	14.04

Baseline method. A typical privacy protection approach runs object detection once, masks prohibited objects and encodes the masked frames n times for n VCs with different block lists. Blurring is chosen as the masking method for its best efficiency.

Other Settings. For encoding, we use the HEVC standard with the Kvazaar encoder [18] and FFmpeg [8] for decoding. YOLOv8 [16] is employed for object detection, and FCEABE [30] is used as the CP-ABE implementation.

7.2 Evaluation of FVES$^+$ Real-time Performance

In the baseline method, the VO must generate and send a separate version of the video for each VC. Regardless of the method used—whether encryption, redaction, or other techniques—the VO needs to detect the objects in the video, determine access permissions for each object, and encode the video as many times as there are VCs. As a result, the time complexity of the baseline method is up to $O(n)$. Fig. 9 shows the Frames Per Second (FPS) of FVES$^+$ and the baseline method running on a PC using the VIRAT dataset. The average number of objects per frame in VIRAT is 8.47. When the number of privacy requirements is below 8.47, the FPS for 64-uni and 64-ada decreases slightly. Beyond this point, the FPS stabilizes around 33 for 64-uni and 34 for 64-ada.

Compared to the baseline method, FVES$^+$ introduces additional computational overhead on the VC side, as it requires decrypting the encrypted video. Fig. 10 presents the average time overhead for ABE decryption and video decoding across the four datasets. The decoding time overhead is minimally affected by different tile partitioning methods, with a difference of no more than 0.49 ms per frame. In 64-uni, the average ABE decryption overhead is 4.79 ms per frame, while in 64-ada, it is 2.79 ms per frame. Thus, these additional time overheads have minimal impact on real-time performance, allowing the VC side to smoothly decode the encrypted video, similar to the baseline method.

To sum up, FVES$^+$ speeds up the encoding efficiency by $\Theta(n)\times$ compared to the baseline method on the VO side. Specifically, our results show that 64-ada

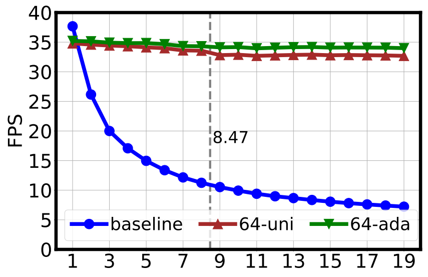

Fig. 9. The FPS of FVES$^+$ encryption algorithm and the baseline method vs. the number of VCs with different block lists.

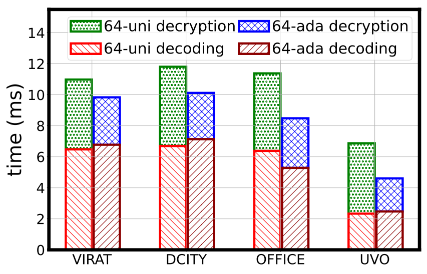

Fig. 10. The average time overhead for decoding and decryption per frame in the FVES$^+$ on the VC side.

achieves $(0.58 + 0.31n)\times$ on average for n different privacy needs. Additionally, FVES$^+$ introduces an additional $O(1)$ decryption time overhead on the VC side, specifically, an average of 2.79 ms per frame. This does not hinder real-time decryption and decoding on the VC side.

Table 2. The average running time (ms) per frame of object detection (O.D.), CPABE, and SE on a PC with 64-uni and 64-ada on different datasets.

average running time (ms) /frame						
dataset	O.D.	FCEABE	64-uni		64-ada	
			SE	all	SE	all
Office	14.76	0.95	12.75	28.46	10.56	26.26
VIRAT	13.91	1.24	12.96	28.11	13.55	28.69
Dcity	13.87	2.78	13.39	30.04	14.27	30.93
Uvo	10.69	1.56	3.18	15.44	3.31	15.57

We evaluate three key components of FVES$^+$: object detection, key generation & encryption, and encoding with SE. We use the ground truth of the objects' labels. Tab. 2 shows that FVES$^+$ achieves an execution efficiency of 28.50 ms per frame, or 35.1 FPS, on the OFFICE, VIRAT, and D2 datasets with a resolution of 1920×1080, and 64.5 FPS on the UVO dataset. In FVES$^+$, the bottleneck in runtime is the efficiency of video encoding, not the encryption.

Efficiency of decoding with decryption. Tab. 3 shows the average decryption and decoding FPS on PC is higher than 53.65, and the FPS on mobile devices is higher than 13. Tab. 2 and Tab. 3 show that the efficiency of the 64-ada is higher than the 64-uni, indicating the effectiveness of adaptive tile partitioning on efficiency.

7.3 Visually Overwhelming

To improve the visual experience, enhancements can be applied to the portions that cannot be decrypted by VCs, depending on whether decryption is successful. Techniques such as blurring or blacking out can be used to make the encrypted frames more user-friendly, as illustrated in Fig. 11.

Table 3. The average runnting time (ms) per frame of decryption and decoding on mobile devices and PC on four datasets

plateform		PC	Jetson	Device 1	Device 2
OFFICE	64-uni	11.4	22.6	44.7	59.7
	64-ada	8.5	18.0	42.8	56.8
VIRAT	64-uni	11.0	20.5	33.8	42.9
	64-ada	9.8	15.9	31.3	39.4
DCITY	64-uni	11.8	23.0	58.5	75.2
	64-ada	10.1	18.1	56.1	71.7
UVO	64-uni	6.9	11.8	19.4	30.2
	64-ada	4.6	7.9	18.9	29.7

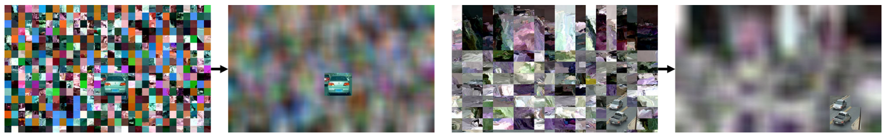
Fig. 11. Functionally decrypted frames and their visually enhanced processing results.

8 Related Work

Video privacy protection methods keep private video content from users without permission. They can be categorized into three types of methods as follows.

CV-based protection methods. Traditional computer vision-based methods use mosaic, blur, and pixel-covering to cover privacy information. As deep learning methods develop rapidly, some deep learning-based methods show advantages in visual effects and usability after desensitization. Wu et al. [36] propose a video privacy protection framework based on cycle-GAN to protect and recover the details of the whole frame. They claim that it is almost impossible to summarize a common Region of Interest (ROI) list as the privacy protection target for all users, so they protect the details of the whole frame. However, their framework has frame-level granularity and cannot completely preserve visual privacy, as only removing details is not enough when privacy is clearly defined. Some works focus on image desensitization technology. Face de-identification methods [9,37,17] focus on protecting face privacy by replacing original faces in the images or videos with fake faces. Inpainting-based methods [21,38,40] are used to remove the ROI region of images by filling the target region with estimated values. Those methods have relatively high requirements on the GPU and cannot meet the real-time performance on weak-computing devices.

Video encryption methods. NEA [20] directly encrypts video streams as binary sequences without considering video coding. The work [25] proposes a method to encrypt every bit of MPEG video by a pseudo-random number generated by the international data encryption algorithm. The video encrypted by NEA is incompatible with video coding standards, and the high complexity of encryption algorithms makes it impossible to meet the requirements of real-time video transmission. SE [33,32] algorithms explore more efficient video encryption and ensure that encrypted videos satisfy the decoding standard but the content is distorted. The work [7] proposes a tile-based ROI encryption scheme for HEVC.

Most CV-based and encryption-based methods have to encode videos several times to process varying privacy requirements, while FVES$^+$ only encodes videos once. Some methods encode videos once but only provide frame-level or file-level protection, FVES$^+$ has object-level granularity.

Access control methods. Database management and querying [11,1,12,13] are used as the conventional user access control methods. The control granularity in most of those works is video-level or frame-level. Jin et al. [15] provided a privacy-protection architecture to pre-process and minimize outgoing data before sending videos to external cloud servers, which supports object-level database-like queries. The work [6] develops a tile-based storage management system that splits video frames into independently queryable tiles for different video queries. However, the access control methods cannot support real-time video streaming for multiple privacy needs.

Many cryptography techniques have been developed in the literature to achieve personalized decryption objectives. Functional Encryption (FE) [3,23,19] allows authorized parties to compute specific functions over encrypted data without revealing the plaintext. As a specific case of FE with an "all or nothing" function, ABE is adopted to share scalable media based on attributes rather than consumers' names. The work [39] proposes a provably secure time-domain attribute-based access control scheme to securely share video content to a certain group of people during a particular period in cloud-based multimedia systems. The work [1] uses ABE to secure the information of streamers and viewers and uses smart contracts and blockchain to ensure reliability and prevent the manipulation or forging of multimedia content. To the best of our knowledge, there is currently no existing FE work targeting the object-level ROI function of videos.

9 Conclusion

We design and implement a scheme FVES and its enhancement, for real-time object-level-granularity personalized video encryption and access control among a group of users. The main contribution of FVES is the functional encryption for unstructured video data, i.e., "one encryption and multiple object-level decryptions". We demonstrate that FVES$^+$ is promising for fine-grained real-time video access control and sharing even for end-users with mobile devices. FVES$^+$ achieves 35.1 FPS using the PC on four open-source datasets on average. When there are n different access requirements by end-users, FVES speeds up the video sharing speed by $\Theta(n)\times$ compared to the SOTA CV-based methods.

References

1. Ahmed, F., et al.: Toward fine-grained access control and privacy protection for video sharing in media convergence environment. Int. J. Intell. Syst. **37**(5), 3025–3049 (2022)

2. Bethencourt, J., et al.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA. pp. 321–334. IEEE Computer Society (2007)
3. Boneh, D., et al.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6597, pp. 253–273. Springer (2011)
4. Boyadjis, B., et al.: Extended selective encryption of H.264/AVC (CABAC)- and hevc-encoded video streams. IEEE Trans. Circuits Syst. Video Technol. **27**(4), 892–906 (2017)
5. Che, Z., Li, M.G., Li, T., Jiang, B., Shi, X., Zhang, X., Lu, Y., Wu, G., Liu, Y., Ye, J.: D^2-city: A large-scale dashcam video dataset of diverse traffic scenarios. CoRR **abs/1904.01975** (2019)
6. Daum, M., et al.: TASM: A tile-based storage manager for video analytics. In: 37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021. pp. 1775–1786. IEEE (2021)
7. Farajallah, M., et al.: ROI encryption for the HEVC coded video contents. In: 2015 IEEE International Conference on Image Processing, ICIP 2015, Quebec City, QC, Canada, September 27-30, 2015. pp. 3096–3100. IEEE (2015)
8. FFmpeg Developers: ffmpeg tool. `http://ffmpeg.org/` (2022), version N-108064-gdb85e01fd7
9. Gafni, O., et al.: Live face de-identification in video. In: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. pp. 9377–9386. IEEE (2019)
10. Hannuksela, M.M., et al.: Overview of the multiview high efficiency video coding (MV-HEVC) standard. In: 2015 IEEE International Conference on Image Processing, ICIP 2015, Quebec City, QC, Canada, September 27-30, 2015. pp. 2154–2158. IEEE (2015)
11. He, K., et al.: Secure independent-update concise-expression access control for video on demand in cloud. Inf. Sci. **387**, 75–89 (2017). `https://doi.org/10.1016/J.INS.2016.08.018`
12. He, Q., et al.: BPS-VSS: A blockchain-based publish/subscribe video surveillance system with fine grained access control. In: Blockchain and Trustworthy Systems - Second International Conference, BlockSys 2020, Dali, China, August 6-7, 2020, Revised Selected Papers. Communications in Computer and Information Science, vol. 1267, pp. 255–268. Springer (2020)
13. Hu, L., et al.: The efficiency improved scheme for secure access control of digital video distribution. Multim. Tools Appl. **75**(20), 12645–12662 (2016)
14. Isa, H., et al.: AES: current security and efficiency analysis of its alternatives. In: 7th International Conference on Information Assurance and Security, IAS 2011, Melacca, Malaysia, December 5-8, 2011. pp. 267–274. IEEE (2011)
15. Jin, H., et al.: Peekaboo: A hub-based approach to enable transparency in data processing within smart homes. In: 43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022. pp. 303–320. IEEE (2022)
16. Jocher, G., et al.: YOLO by Ultralytics (2023)
17. Kuang, Z., et al.: Effective de-identification generative adversarial network for face anonymization. In: MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021. pp. 3182–3191. ACM (2021)
18. Lemmetti, A., et al.: Kvazaar 2.0: fast and efficient open-source HEVC inter encoder. In: Proceedings of the 11th ACM Multimedia Systems Conference, MMSys 2020, Istanbul, Turkey, June 8-11, 2020. pp. 237–242. ACM (2020)

19. Lewko, A.B., et al.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques. vol. 6110, pp. 62–91. Springer (2010)
20. Liu, F., König, H.: A survey of video encryption algorithms. Comput. Secur. **29**(1), 3–15 (2010)
21. Liu, R., et al.: Fuseformer: Fusing fine-grained information in transformers for video inpainting. In: 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021. pp. 14020–14029. IEEE (2021)
22. Oh, S., Hoogs, A., et al.: AVSS 2011 demo session: A large-scale benchmark dataset for event recognition in surveillance video. In: 8th IEEE International Conference on Advanced Video and Signal-Based Surveillance, AVSS 2011, Klagenfurt, Austria, August 30 - September 2, 2011. pp. 527–528. IEEE Computer Society (2011)
23. O'Neill, A.: Definitional issues in functional encryption. IACR Cryptol. ePrint Arch. p. 556 (2010)
24. Peng, F., et al.: A tunable selective encryption scheme for H.265/HEVC based on chroma IPM and coefficient scrambling. IEEE Trans. Circuits Syst. Video Technol. **30**(8), 2765–2780 (2020)
25. Qiao, L., Nahrstedt, K.: A new algorithm for mpeg video encryption (08 2001)
26. Sallam, A.I., Faragallah, O.S., El-Rabaie, E.M.: HEVC selective encryption using RC6 block cipher technique. IEEE Trans. Multim. **20**(7), 1636–1644 (2018)
27. Sallam, A.I., et al.: Efficient HEVC selective stream encryption using chaotic logistic map. Multim. Syst. **24**(4), 419–437 (2018)
28. Sheng, Q., et al.: A fast selective encryption scheme for H.264/AVC video with syntax-preserving and zero bit rate expansion. Signal Image Video Process. **18**(1), 975–989 (2024)
29. Sidaty, N.O., et al.: A new perceptual assessment methodology for selective HEVC video encryption. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017. pp. 1542–1546. IEEE (2017)
30. Tomida, J., et al.: Fast, compact, and expressive attribute-based encryption. Designs, Codes and Cryptography **89**(11), 2577–2626 (2021)
31. Vincent, O.R., Folorunso, O.: A descriptive algorithm for sobel image edge detection. In: Proceedings of informing science & IT education conference (InSITE). pp. 97–107 (2009)
32. Wallendael, G.V., et al.: Encryption for high efficiency video coding with video adaptation capabilities. IEEE Trans. Consumer Electron. **59**(3), 634–642 (2013)
33. Wallendael, G.V., et al.: Format-compliant encryption techniques for high efficiency video coding. In: IEEE International Conference on Image Processing, ICIP 2013, Melbourne, Australia, September 15-18, 2013. pp. 4583–4587. IEEE (2013)
34. Wang, W., et al.: Unidentified video objects: A benchmark for dense, open-world segmentation. In: 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021. pp. 10756–10765. IEEE (2021)
35. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6571, pp. 53–70. Springer (2011)

36. Wu, H., et al.: PECAM: privacy-enhanced video streaming and analytics via securely-reversible transformation. In: ACM MobiCom '21: The 27th Annual International Conference on Mobile Computing and Networking, New Orleans, Louisiana, USA, October 25-29, 2021. pp. 229–241. ACM (2021)
37. Wu, Y., et al.: Privacy-protective-gan for privacy preserving face de-identification. J. Comput. Sci. Technol. **34**(1), 47–60 (2019)
38. Xu, R., et al.: Deep flow-guided video inpainting. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 3723–3732. Computer Vision Foundation / IEEE (2019)
39. Yang, K., et al.: Time-domain attribute-based access control for cloud-based video content sharing: A cryptographic approach. IEEE Trans. Multim. **18**(5), 940–950 (2016)
40. Yu, H., et al.: Efficient object-grained video inpainting with personalized recovery and permission control. In: 2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS). pp. 1–10 (2024)
41. Zhou, W., et al.: Discovering and understanding the security hazards in the interactions between iot devices, mobile apps, and clouds on smart home platforms. In: 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019. pp. 1133–1150. USENIX Association (2019)

A Appendix

A.1 Details of FVES$^+$.`Encrypt`

We present the details of FVES$^+$.`Encrypt` in Algorithm 4. The algorithm ensures that the number of CPABE.`Encrypt` executions are unaffected by the movement or overlapping of objects and are influenced only by the number of objects, thereby improving efficiency.

Algorithm 4: FVES$^+$.`Encrypt` algorithm

```
Input: Frame sequence $\mathcal{V} = \{V^1, ..., V^N\}$, set of access strategies
       $\mathbb{A} = \{\mathbb{A}_1, \mathbb{A}_2, ..., \mathbb{A}_c\}$ of the video element universe, $c$ is the size of the
       element universe, master public key $MPK$
Output: Encrypted video stream $C_{VS}$, tile cover information $\{\mathbb{I}^k\}$ for each
        frame $V^k$, encrypted keys $\{C_j\}$ for each element shown in $\mathcal{V}$
1  Arrange video elements (objects and background) in a random order $\mathcal{O}$;
2  Initialize video stream $C_{VS}$, empty element list $L$, and the key-counter
     dictionary $D$;
3  for each frame $V^k$ in $\mathcal{V}$ do
4      Detect elements in $V^k$;
5      for each detected element $e_j$, where $j \in \{1, 2, \ldots, c\}$ do
6          if $e_j \notin L$ then
7              Add $e_j$ to $L$;
8              Generate the key $K_j$ for $e_j$ using ;
9              Compute $C_j = $ CPABE.Encrypt$(MPK, \mathbb{A}_j, K_j)$;
10     Perform adaptive tile partitioning;
11     Label what elements each tile covers using the element index in $\mathcal{O}$, and
         yield information $\mathbb{I}^k$;
12     for each tile $T$ do
13         if $T$ covers a single element $e_j$ then
14             Use the key of $e_j$ as $K_T$;
15         else
16             Use the common substrings of the keys of the covered objects as
                 the tile key $K_T$;
17         if $K_T \notin$ keys of $D$ then
18             Set the value $D\{K_T\}$ in the key-counter dictionary as 0;
19         Encode $C_T = $ SE.Encrypt$(T, K_T, D\{K_T\})$ into the video stream $C_{VS}$;
20         Update $D\{K_T\}$;
21 return $C_{VS}$, $\{C_j\}$, $\{\mathbb{I}^k\}$
```

A.2 Discussion on Real-World Applications

In this section, we discuss potential challenges when applying these video encryption schemes in real-world scenarios: **Hardware Constraints.** A standard video encoder generates syntax elements of varying types and lengths across different CUs. For instance, a CU representing a uniform-colored region may use syntax elements different from those with complex textures. Consequently, modifications to the standard encoding algorithm are necessary to secure SE.

To address this, the FVES scheme's video encoder applies a uniform encoding method for all CUs, producing syntax elements of consistent type and length. This ensures that encoding and encrypting video frames of identical size result in encrypted bitstreams with uniform length and format. However, in many practical scenarios, the video encoding algorithm is hardcoded into hardware encoders and cannot be altered. To apply FVES or FVES$^+$ in such cases, the VO must rely on standard video encoders. We analyze the potential security risks in this context and propose a shuffle-based technique to mitigate them.

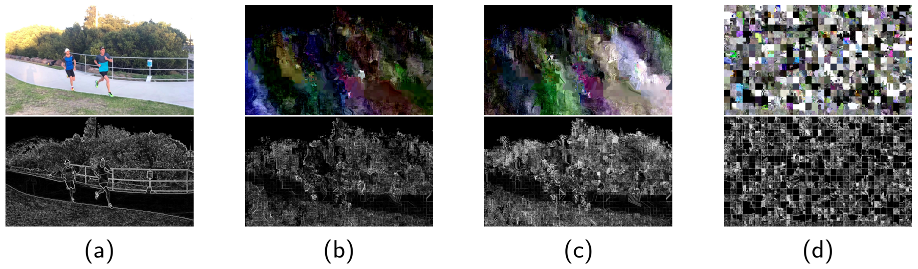

Fig. 12. (a) Original frame, (b) encrypted frame, (c) encrypted frame enhanced by coding coefficient scrambling [24], (d) encrypted frames enhanced by our method and their edge information leakage.

In the above case, an attacker still cannot decrypt unauthorized tiles to reveal their original pixel values. However, while encrypted syntax elements are fully randomized, using a standard video encoder causes different CUs to be encoded with varying syntax elements, leading to inconsistent levels of perturbation. This pattern can allow attackers to infer texture complexity for each tile. By applying edge detection techniques from computer vision, they could partially reconstruct the edges of the original frame (Fig. 12a, 12b). Peng et al. [24] attempted to protect edge information by scrambling residual syntax elements (Fig. 12d), but their method failed to obscure the overall contours of objects.

Our goal, under the constraint of using a standard video encoder, is to ensure that the security of each tile is at least as strong as that of existing SE methods, while simultaneously disrupting the overall contours of objects at the tile level and obscuring the continuity of edge information between tiles.

In the encryption algorithms of FVES and FVES$^+$, each tile is assigned an encryption key. Since each tile is encoded independently, the arrangement of tiles does not impact encoding efficiency or video quality. After tile encryption, we group tiles into different sets based on their shape. For each set, we apply random shuffling of their positions. For each tile, a pseudo-random number is generated using the SE.`Encrypt` algorithm's PRNG, which is then bitwise XORed with the original positions of the tiles. The ciphertext, along with the ciphertext from either FVES.`Encrypt` or FVES$^+$.`Encrypt`, is transmitted to the VCs.

In comparison to existing SE encryption algorithms, the shuffle-based method disrupts the edge relations between tiles, rendering the edge information invisible (Fig. 12d). We evaluate the shuffle-based SE-enhanced method based on four datasets, which encompass different scenes and camera states, as follows: (1)

VIRAT [22] showcases a variety of camera viewpoints in street scenes captured by stationary surveillance cameras. (2) DCITY [5] displays urban road scenes captured by moving cameras. (3) UVO [34] is a video dataset for open-world object segmentation collected from YouTube. (4) Additionally, we collected 24-hour videos from a stationary surveillance camera in indoor-office scenes, named OFFICE. VIRAT, DCITY, and OFFICE have resolutions of 1920x1080, while the resolutions of UVO videos are variable.

We measure the frame distortion of FVES and FVES$^+$ quantitatively using SSIM. The smaller the SSIM values, the higher the distortion of the frame, which means better encryption performance. The comparison between the four datasets in Fig. 13 demonstrates the privacy enhancement of FVES$^+$ with its SE-enhanced method compared to FVES.

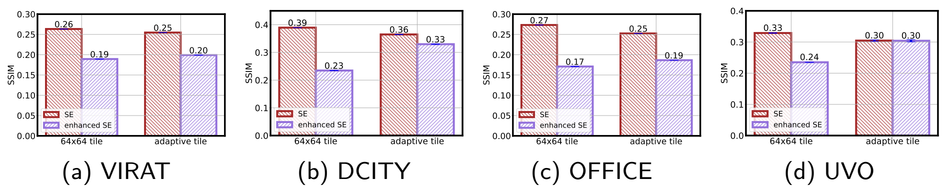

Fig. 13. The SSIM values of 64-uni and 64-ada with/without SE-enhanced method in four datasets.

Except the SSIM, We also measure frame distortion of FVES and FVES$^+$ quantitatively using the Edge Distortion Ratio (EDR) [26].

$$EDR = \frac{\sum_{m=1}^{M} |PI_{ori}(m) - PI_{enc}(m)|}{\sum_{m=1}^{M} |PI_{ori}(m) + PI_{enc}(m)|},$$

where M represents the total number of edge pixels extracted by the Sobel operator [31], and $PI_{ori}(m)$, $PI_{enc}(m)$ represent the m-th pixel of the original and the encrypted frames in the edge region, respectively. The higher the EDR values, the higher the distortion of the frame, which means better encryption performance.

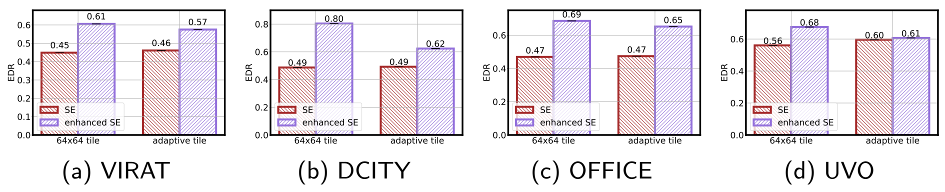

Fig. 14. The EDR values of 64-uni and 64-ada with/without SE-enhanced method in four datasets.

Fig. 14 demonstrates the effectiveness of SE-enhanced methods. Additionally, the performance improvement of enhanced SE on 64-ada is not as significant as that on 64-uni because the shuffling space is smaller.