INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 4
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC 1/SC 29/WG 4 m 69555

Kemer – November 2024

Title: [MIV] V3C Bitstream-level DRM Encryption for Heterogeneous Volumetric Video Objects
Author: Isaac Yang, Soo-Been Jo, Jong-Beom Jeong, Eun-Seok Ryu (SKKU), Woo-Chool Park (KETI), Sungheun Oh (DigiCAP)

Abstract

This document proposes a concept of volumetric visual video-based coding (V3C) bitstream-level digital rights management (DRM) encryption for heterogeneous volumetric video objects. Extending the application of a common encryption (CENC) protection system-specific header, which typically applied to segments or fragments in MPEG dynamic adaptive streaming over HTTP (DASH)-packaged media files for traditional 2D video DRM streaming, this document proposes a DRM encryption option designed for use on the atlas tile or patch of V3C bitstream. The proposed DRM syntax is signaled within the atlas data (AD). This approach is expected to support encryption of volumetric video objects represented by tracks in a multi-track V3C bitstream, including heterogeneous objects.

1 Introduction

The volumetric visual video-based coding (V3C) standard has established itself as a crucial technology for the efficient transmission and storage of immersive media content. As immersive experiences such as virtual reality (VR), augmented reality (AR), and mixed reality (MR) continue to evolve, the demand for content capable of accurately representing diverse heterogeneous volumetric objects in a 3D space is also increasing. These objects include point clouds, immersive videos, and mesh objects, which are compressed and transmitted using technologies such as video-based point cloud compression (V-PCC), MPEG immersive video (MIV), and video-based dynamic mesh coding (V-DMC).

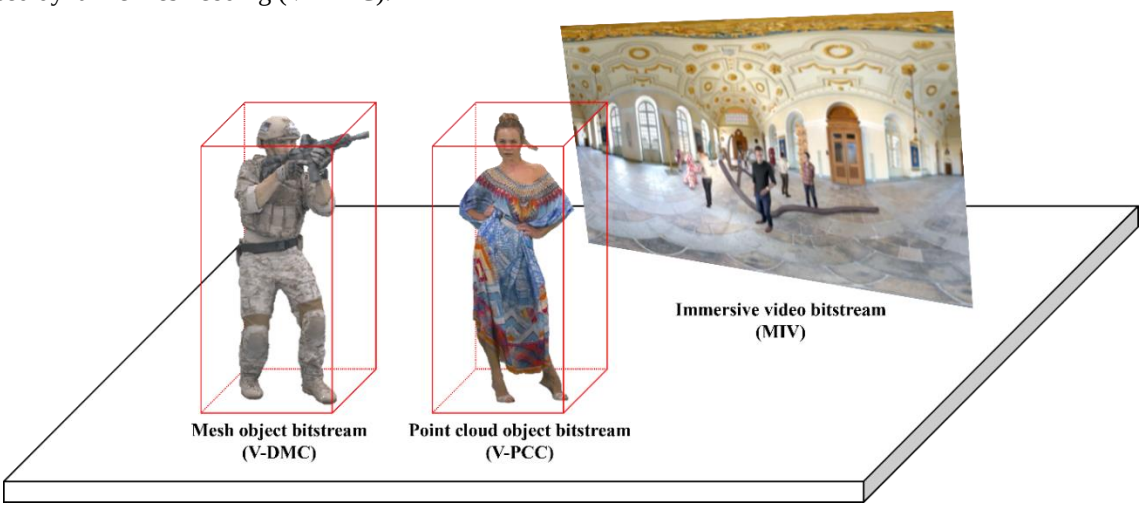

Figure 1. Conceptual diagram of heterogeneous objects in a V3C bitstream of a volumetric space

Figure 1 illustrates a conceptual diagram of these heterogeneous volumetric objects being processed within a single volumetric space. As the need to simultaneously handle heterogeneous volumetric objects grows, there is a corresponding demand for a multi-track structure within the V3C standard. The introduction of a multi-track structure would enable the independent management and processing of different volumetric objects, ensuring that each object is encoded, transmitted, and decoded according to its unique characteristics. Additionally, this approach would reduce the workload if decoding instance on the client side. This extension allows V3C standard to more effectively meet the diverse requirements of heterogeneous volumetric content [1]. Furthermore, new parameters have been proposed to extend the existing V3C bitstream syntax, the system to determine the appropriate codec for each object [2].

While well-established digital rights management (DRM) standards exist for traditional 2D video, no such standards have yet been defined for volumetric video content. As the use of volumetric objects expands across various industries, the development of robust DRM technologies tailored to volumetric content is essential. Particularly, applying DRM to the multi-track V3C structure for heterogeneous volumetric objects, as discussed earlier, would allow differentiated encryption options to be applied to each object. This approach would ensure that the encryption applied to each track is aligned with the specific encryption requirements of the volumetric object it represents.

Traditional DRM encryption methods typically operate at the transport layer, where media data is encrypted during transmission. For example, in an MPEG-DASH environment, a single encryption option is applied across the entire content, ensuring the content remains encrypted during transmission [3]. However, transport layer-based encryption has limitations when handling a V3C bitstream that contains multiple heterogeneous objects.

At the transport layer, encryption is applied uniformly to the entire content, which means there is no ability to assign customized security settings based on the characteristics of individual objects. In scenarios where multiple heterogeneous objects are included, it is often necessary to apply differentiated encryption options to each object. However, transport-layer encryption lacks the flexibility to meet these detailed requirements [4].

In contrast, applying DRM encryption options within the V3C bitstream allows for more tailored security settings at the track or tile level. This enables the system to address the unique security needs of each object, which is especially important for immersive content like VR and AR. Therefore, in the V3C bitstream, applying DRM encryption at the media layer, rather than at the transport layer, is more appropriate.

By applying different levels of encryption to volumetric objects, sensitive content such as high-resolution immersive videos can be secured with stronger encryption, while less critical content can be protected with lighter encryption methods. This document proposes a flexible DRM encryption framework that supports such versatility, ensuring the secure transmission and storage of volumetric video objects. This solution provides a scalable encryption model that allows for more efficient and reliable management of volumetric video bitstream.

2 Motivation

In the V3C bitstream, the multi-track structure provides an efficient means to protect heterogeneous volumetric objects by defining DRM encryption options within the AD rather than modifying the higher-level syntax, such as VPS. Following the use-case of [2] for heterogeneous objects, this approach adopts an atlas and tile-based method, which allows DRM encryption to be applied at the object level without changing the VPS syntax. This structure offers flexibility for consistent protection across different tracks while enabling encryption levels to be customized according to the security requirements of each object.

First, the ability to individually protect heterogeneous objects through DRM encryption options in a multi-track structure is essential for managing diverse volumetric content. When objects are processed using different formats or codecs, applying encryption tailored to the characteristics of each object is crucial. For example, point clouds, immersive videos, and mesh objects are processed using distinct data formats and compression technologies, making it difficult to meet all encryption requirements with a uniform encryption method. However, by using the DRM encryption options available in the multi-track structure, unique encryption settings can be assigned to each track, effectively addressing this issue.

This approach provides the flexibility to apply different encryption levels to each track, allowing sensitive and less critical content to be distinguished and protected accordingly. For instance, high-resolution immersive videos may require stronger encryption, while less important objects can be handled with lighter encryption. This flexible

encryption method allows encryption method allows encryption strength to be adjusted based on the encryption needs of each object, providing a more suitable protection solution. As a result, the DRM encryption options enable the implementation of a consistent encryption strategy that is not dependent on format or codec, ensuring that heterogeneous objects are comprehensively protected even in complex environments. Moreover, the multi-track structure of the V3C bitstream enables the collective assignment of encryption options through the AD while managing multiple objects independently. These multiple encryption methods ensure a consistent protection mechanism throughout the entire bitstream.

Traditional DRM encryption methods for 2D video formats typically apply encryption options at the transport layer during streaming, rather than within the bitstream itself. However, applying DRM encryption options at the transport layer presents practical limitations when considering use cases involving heterogeneous volumetric objects. In content delivery scenarios, it is often necessary to apply different encryption options to individual objects, but the transport layer generally only supports blanket encryption or no encryption for the entire content. This lack of flexibility in providing object-specific protection can hinder the implementation of a coherent security strategy.

Therefore, applying differentiated encryption options at the media layer is more efficient. By assigning encryption options tailored to each object at the media layer and conveying this information during transport, the system can address the security needs of the content with greater precision. This approach enables object-specific encryption settings and provides a more appropriate security solution that aligns with the service provider's requirements.

Another significant advantage of the multi-track structure is the ability to reduce the number of atlases by managing multiple objects within a single, unified bitstream instead of separating them into individual bitstreams [5], [6]. Reducing the number of atlases consequently decreases the number of atlas video decoder instances required, improving decoding performance and optimizing system resource usage. This structure is particularly beneficial for enabling the smooth processing of high-quality 3D content on lower-specification devices.

The reduction in the number of decoder instances not only conserves system resources but also improves the overall efficiency of the decoding process. When multiple objects are managed within a unified bitstream, the need for synchronization between decoders is minimized as the number of atlases decreases. This reduces potential decoding bottlenecks and allows concurrent decoding processes to be handled more stably, ultimately contributing to enhanced overall decoding performance [7].

Furthermore, reducing the number of decoder instances enables the efficient handling of 3D volumetric content on both high-end and low-end devices [8]. This approach supports the seamless playback of high-quality content using fewer hardware resources, ensuring a consistent high-quality content experience across a wide range of devices. Additionally, reducing the number of atlases simplifies system resource management, enhancing processing performance and improving the stability and quality of the overall service.

3 Common Encryption (CENC)

Common encryption (CENC) is a standardized encryption method designed to allow multiple DRM systems to protect the same media content file [9]. CENC is primarily used in file structures based on the ISO/IEEC 14496-12 ISO base media file format and protects media sample data using AES-128 symmetric block encryption. This allows content providers to protect a single media file across different DRM systems, ensuring interoperability across various user devices.

CENC conveys the information necessary for DRM systems through a metadata block known as the protection system specific header (PSSH). The PSSH includes specific information required by each DRM system for content protection, providing details such as the key identifier (KID). DRM systems use this information to control access to the encrypted content and decrypt it if necessary. Through this structure, CENC offers interoperability by allowing multiple DRM solutions to decrypt the same encrypted content. As a result, content providers can distribute the same file across multiple platforms without needing to perform additional encryption processes.

CENC uses the advanced encryption standard counter mode (AES-CTR) to protect media data. AES-CTR mode is highly performant and efficient, making it suitable for real-time streaming applications while meeting a wide range of encryption requirements. This encryption method provides flexibility, allowing content providers to choose the encryption strategy that best fits their environment, ensuring that diverse security needs are met.

Furthermore, CENC is closely integrated with MPEG dynamic adaptive streaming over HTTP (MPEG-DASH). DASH is an HTTP-based adaptive streaming technology that adjusts the quality of content based on network conditions while maintaining continuous streaming. When used in conjunction with DASH, CENC allows each media segment to be streamed in an encrypted state, with the necessary keys and metadata provided through the media presentation description (MPD) file. The MPD file contains information about each media segment as well as metadata related to the key management system required to decrypt the encrypted content.
By combining CEENC with MPEG-DASH, encrypted content can be securely transmitted and played in real-time within an adaptive streaming environment. Thanks to the adaptive nature of DASH and the interoperability of CENC, content providers can deliver encrypted content securely across varying network conditions, ensuring a seamless streaming experience for users without compromising on security.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated with https://github.com/shaka-project/shaka-packager version f07fd0d997-debug-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd" xmlns:cenc="urn:mpeg:cenc:2013"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011" minBufferTime="PT2S" type="static"
mediaPresentationDuration="PT31.498133S">
  <Period id="0">
    <AdaptationSet id="0" contentType="video" width="3840" height="1920" frameRate="90000/3003"
    subsegmentAlignment="true" par="2:1">
      <ContentProtection value="cenc" schemeIdUri="urn:mpeg:dash:mp4protection:2011"
      cenc:default_KID="267cc205-7144-45cf-9058-6948cf54b205"/>
      <ContentProtection schemeIdUri="urn:uuid:edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">
        <cenc:pssh>AAAAOHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAABgSECZ8wgVxREXPkFhpSM9UsgVI49yVmwY=</cenc:pssh>
      </ContentProtection>
      <Representation id="0" bandwidth="60072467" codecs="avc1.428033" mimeType="video/mp4" sar="1:1">
        <BaseURL>Encrypted%2Ftest1_video_encrypted.mp4</BaseURL>
        <SegmentBase indexRange="1069-1172" timescale="90000">
          <Initialization range="0-1068"/>
        </SegmentBase>
      </Representation>
    </AdaptationSet>
    <AdaptationSet id="1" contentType="audio" lang="en" subsegmentStartsWithSAP="1" subsegmentAlignment="true">
      <ContentProtection value="cenc" schemeIdUri="urn:mpeg:dash:mp4protection:2011"
      cenc:default_KID="267cc205-7144-45cf-9058-6948cf54b205"/>
      <ContentProtection schemeIdUri="urn:uuid:edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">
        <cenc:pssh>AAAAOHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAABgSECZ8wgVxREXPkFhpSM9UsgVI49yVmwY=</cenc:pssh>
      </ContentProtection>
      <Representation id="1" bandwidth="195343" codecs="mp4a.40.2" mimeType="audio/mp4"
      audioSamplingRate="48000">
        <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011"
        value="2"/>
        <BaseURL>Encrypted%2Ftest1_audio_encrypted.m4a</BaseURL>
        <SegmentBase indexRange="999-1102" timescale="48000">
          <Initialization range="0-998"/>
        </SegmentBase>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Figure 2. Example of DRM encryption using MPEG-DASH MPD with CENC

Figure 2 illustrates an MPD file for a 360-degree MP4 video in equirectangular projection (ERP) format, packaged using DASH with DRM encryption option applied via CENC. This experiment, conducted using Google's **shaka-packager**, demonstrates how DRM encryption through CENC is integrated into the DASH streaming environment [10]. As shown in the MPD file, content protection is defined using the **cenc** scheme, and a **PSSH** is included to provide the necessary encryption information for DRM systems.
In this experiment, the **PSSH** box contains metadata required to decrypt the encrypted media segments. This metadata includes the **default_KID**, which identifies the encryption key associated with the media file. The **PSSH** ensures that different DRM systems can retrieve the appropriate keys and decrypt the content, supporting interoperability across multiple platforms. The DASH MPD file applies CENC encryption to both video and audio adaptation sets, with distinct **PSSH** boxes included for each media type.

4 V3C Syntax in Heterogeneous Objects

Recent extensions to the V3C syntax have been proposed to support heterogeneous volumetric content within a single V3C bitstream. Notably, a recent contribution has proposed methods to enable the coexistence of V-PCC and MIV within the same bitstream [11]. These proposed extensions introduce new syntax elements and modify parameter sets to identify and manage different content types. Such extensions are essential for efficiently encoding and transmitting a wide variety of volumetric content types.

For tile-based approach for heterogeneous objects, the tile-based approach presents an efficient method for managing heterogeneous content such as V-PCC and MIV within a single atlas. By processing multiple volumetric objects within a single bitstream, this approach resolves the issues of managing each object separately and optimizes bandwidth and resource usage [12].

The main advantage of the tile-based approach is that it allows various object types to be packed and processed within the same syntax. This enables simultaneous handling of multiple volumetric contents within a single sequence, offering far greater efficiency than managing multiple bitstreams separately. Furthermore, this approach provides the flexibility needed to ensure smooth playback even on resource-constrained devices, making it a key solution for future volumetric content applications.

5 Bitstream Encryption for Heterogeneous Objects with CENC

The AD can provide customized DRM encryption options to each track. In a multi-track structure handling heterogeneous volumetric objects like V-PCC and MIV, the AD enables independent assignment of encryption options per track. Note that the proposed encryption method is also available in homogeneous use-cases, i.e., V-PCC and MIV profiles.

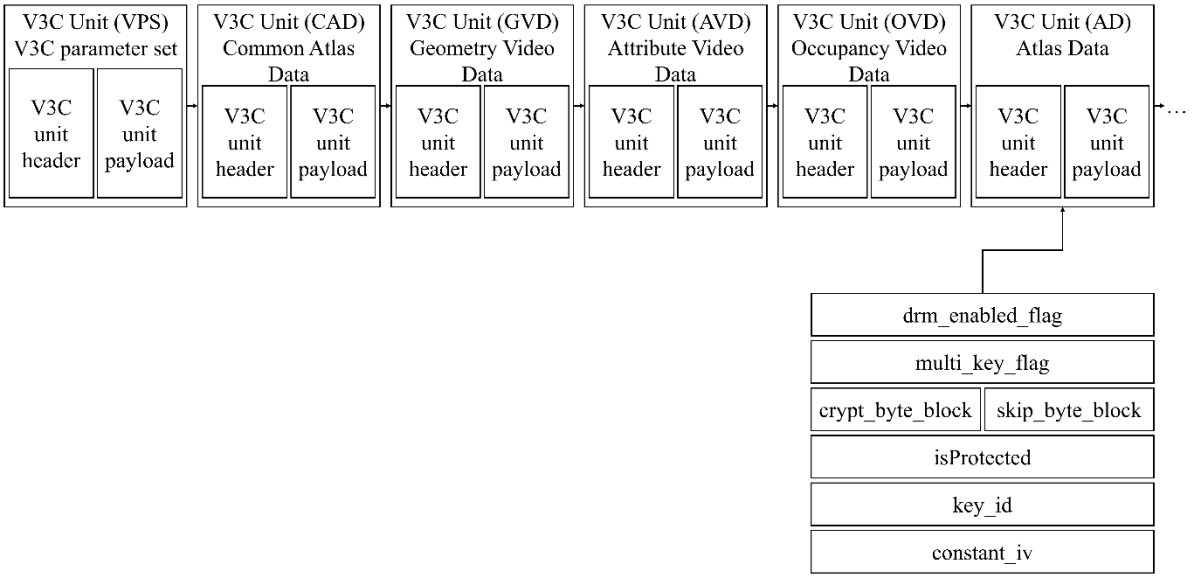

Figure 3. Proposed extended V3C bitstream structure with DRM encryption for heterogeneous objects

Figure 3 shows an example of such a V3C bitstream syntax. ***drm_enabled_flag*** is signaled within the AD and plays a role in conveying the required DRM options to the ***multi_key_flag***, ***byte_block***, ***isProtected***, ***key_id***, and ***constant_iv***. The ***multi_key_flag*** indicates the multiple key versions of the atlas or tile. The ***multi_key_flag*** enables multiple keys for atlases or a tile, enhancing flexibility content protection. The ***crypt_type_block*** and ***skip_byte_block*** parameters specify the pattern of encrypted and unencrypted blocks, supporting efficient encryption management. The ***isProtected*** value defines the number of keys applicable to a sample, while ***key_id*** uniquely identifies each key associated with an atlas or tile. Additionally, ***per_sample_iv_size*** and ***constant_iv*** parameters provide initialization vectors, establishing consistent encryption processes across volumetric content. This fields together allow tailored security settings, meeting varied encryption needs within a unified V3C bitstream.

5.1. Atlas-level DRM Encryption in Atlas Sequence Parameter Set

This syntax proposes parameters for DRM encryption within the V3C bitstream, specifically applied to the AD's atlas sequence parameter set (ASPS) within V3C. This configuration allows customized encryption settings to be assigned individually to each atlas. The ***asps_drm_enabled_flag*** signals if DRM encryption is active, and when enabled, the ***asps_drm_information*** syntax provides the DRM settings for each atlas. Based on ***asps_map_count_minus1***, ***atlas_drm_enabled_flag*** is declared for each atlas, indicating whether DRM is applied; if active, detailed encryption configurations are then specified.

This syntax allows for flexible DRM encryption across different atlases within a single bitstream, tailored to meet the unique security needs of complex volumetric content. The ***atlas_multi_key_flag*** denotes multiple key usage, while ***atlas_per_sample_iv_size*** and ***atlas_constant_iv*** define initialization vectors (IV) for each sample, maintaining encryption randomness. Additionally, ***atlas_crypt_byte_block*** and ***atlas_skip_byte_block*** enable specific byte blocks to be encrypted or skipped, aligning DRM encryption to the unique security demands of each atlas. This approach enhances the adaptability of DRM options in V3C, providing stronger protection for heterogeneous volumetric content.

Table 1. Atlas sequence parameter set DRM enabled flag of extended atlas sequence parameter set RBSP syntax

atlas_sequence_parameter_set_rbsp(numBytesInV3CPayload) {	**Descriptor**
asps_atlas_sequence_parameter_set_id	ue(v)
asps_frame_width	ue(v)
asps_frame_height	ue(v)
asps_geometry_3d_bit_depth_minus1	u(5)
asps_geometry_2d_bit_depth_minus1	u(5)
asps_log2_max_atlas_frame_order_cnt_lsb_minus4	ue(v)
asps_max_dec_atlas_frame_buffering_minus1	ue(v)
asps_lng_term_ref_atlas_frames_flag	u(1)
asps_num_ref_atlas_frame_lists_in_asps	ue(v)
for(i=0; i<asps_num_ref_atlas_frame_lists_in_asps; i++)	
ref_list_struct(i)	
asps_use_eight_orientations_flag	u(1)
asps_extended_projection_enabled_flag	u(1)
if(asps_extended_projection_enabled_flag)	
asps_max_number_projections_minus1	ue(v)
asps_normal_axis_limits_quantization_enabled_flag	u(1)
asps_normal_axis_max_delta_value_enabled_flag	u(1)
asps_patch_precendence_order_flag	u(1)
asps_log2_patch_packing_block_size	u(3)
asps_patch_size_quantizer_present_flag	u(1)
asps_map_count_minus1	u(4)
asps_pixel_deinterleaving_enabled_flag	u(1)
if(asps_pixel_deinterleaving_enabled_flag)	
for(j=0; j<=asps_map_count_minus1; j++)	
asps_map_pixel_deinterleaving_flag[j]	u(1)
asps_raw_patch_enabled_flag	u(1)
asps_eom_patch_enabled_flag	u(1)
if(asps_eom_patch_enabled_flag && asps_map_count_minus1 == 0)	
asps_eom_fix_bit_count_minus1	u(4)
if(asps_raw_patch_enabled_flag || asps_eom_patch_enabled_flag)	
asps_auziliary_video_enabled_flag	u(1)

asps_plr_enabled_flag	u(1)
if(asps_plr_enabled_flag)	
asps_plr_information(asps_map_count_minus1)	
asps_vui_parameters_present_flag	u(1)
if(asps_vui_parameters_present_flag)	
vui_parameters( )	
asps_extension_present_flag	u(1)
if(asps_extension_present_flag) {	
asps_vpcc_extension_present_flag	u(1)
asps_miv_extension_presesnt_flag	u(1)
asps_miv_2_extension_present_flag	u(1)
asps_extension_5bits	u(5)
}	
if(asps_vpcc_extension_present_flag)	
asps_vpcc_extension( )	
if(asps_miv_extension_present_flag)	
asps_miv_extension( )	
if(asps_miv_2_extension_present_flag)	
asps_miv_2_extension( )	
if(asps_extension_5bits)	
while(more_rbsp_data())	
asps_extension_data_flag	u(1)
asps_drm_enabled_flag	u(1)
if(asps_drm_enabled_flag) {	
asps_drm_information( )	
rbsp_trailing_bits()	
}	

asps_drm_enabled_flag indicates whether DRM encryption is enabled within the ASPS.

Table 2. DRM information syntax

asps_drm_information( ) {	**Descriptor**
asps_drm_multi_key_flag	u(1)
asps_drm_crypt_byte_block	u(4)
asps_drm_skip_byte_block	u(4)
if(asps_drm_multi_key_flag)	
asps_drm_key_count_minus1	u(16)
else	
asps_drm_key_count_minus1 = 0	
for(i=0; i<=asps_drm_key_count_minus1; i++) {	
asps_drm_per_sample_iv_size	u(8)
for(j=0; j<16; j++) {	
asps_drm_key_id[j]	u(8)
if(asps_drm_per_sample_iv_size == 0) {	
asps_drm_constant_iv_size_minus1	u(8)
for(j=0; j<asps_drm_constant_iv_size_minus1; j++)	
asps_drm_constant_iv[j]	u(8)
}	
}	

}	

asps_drm_enabled_flag defines whether DRM encryption is applied for each atlas.
asps_drm_multi_key_flag indicates that multiple key versions of the atlas are used. If this flag is set, multiple keys will be described for current atlas: otherwise, a single key is described for current atlas.
asps_drm_crypt_byte_block specifies the count of the encrypted atlas in the protection pattern, where each block is of size 16-bytes.
asps_drm_skip_byte_block specifies the count of the unencrypted atlas in the protection pattern.
asps_drm_key_count_minus1 plus 1 indicates the number of keys that may apply to a sample associated to the current atlas. It is not required that a sample associated with current atlas uses all keys described.
asps_drm_key_id is the key identifier used for the current atlas. The size of asps_drm_key_id shall be in the range of 0 to 15, inclusive.
asps_drm_per_sample_iv_size is the initialization vector size in bytes for samples in the current atlas.
asps_drm_constant_iv_size_minus1 plus 1 is the size of a possible initialization vector used for current atlas.
asps_drm_constant_iv, if present, is the initialization vector used for current atlas. The size of asps_drm_constant_iv shall be in the range of 0 to asps_drm_contant_iv_size, inclusive.

5.2. Tile-level DRM Encryption in Atlas Tile Header

Applying DRM encryption options at the tile level enables more granular security management within the V3C bitstream. Tiles represent individual objects or data blocks, and each tile can contain different types of data. For example, V-PCC tiles and MIV tiles may coexist within the same bitstream, necessitating security settings tailored to each tile.
Tile-based encryption offers the flexibility to distinguish between sensitive and less critical data. If certain atlas tiles contain high-resolution video or important information, those tiles can be assigned stronger encryption settings. Conversely, less important tiles can be encrypted with lighter methods to improve processing efficiency. This approach is particularly useful when dealing with complex volumetric content that contains multiple objects and data types.
Furthermore, the tile-based approach reduces the complexity of handling the content while providing appropriate security settings for each tile. This method contributes to maintaining content security while optimizing resource usage.

Table 3. Atlas tile header DRM enabled flag of extended atlas tile header syntax

atlas_tile_header( ) {	**Descriptor**
if(nal_unit_type >= NAL_BLA_W_LP && nal_unit_type <= NAL_RSV_IRAP_CAL_29)	
ath_no_output_of_prior_atlas_frames_flag	u(1)
ath_atlas_frame_parameter_set_id	ue(v)
ath_atlas_adaptation_parameter_set_id	ue(v)
ath_id	u(v)
ath_type	ue(v)
if(afps_output_flag_resent_flag)	
ath_atlas_output_flag	u(1)
ath_atlas_frm_order_cnt_lsb	u(v)
if(asps_num_ref_atlas_frame_lists_in_asps > 0)	
ath_ref_atlas_frame_list_asps_flag	u(1)
if(ath_ref_atlas_Frame_list_asps_flag == 0)	
ref_list_struct(asps_num_ref_atlas_frame_lists_in_asps)	
else if(asps_num_ref_atlas_frame_lists_in_asps > 1)	
ath_ref_atlas_frame_list_idx	u(v)
for(j=0; j<NumLtrAtlasFrmEntries[RlsIdx]; j++) {	
ath_additional_afoc_lsb_present_flag[j]	u(1)
if(ath_additional_afoc_lsb_present_flag[j])	
ath_additional_afoc_lsb_val[j]	u(v)
}	

if(ath_type != SKIP_TILE) {	
if(asps_normal_axis_limits_quantization_enabled_flag) {	
ath_pos_min_d_quantizer	u(5)
if(asps_normal_axis_max_delta_value_enabled_flag)	
ath_pos_delta_max_d_quantizer	u(5)
}	
if(asps_patch_size_quantizer_present_flag) {	
ath_patch_size_x_info_quantizer	u(3)
ath_patch_size_y_info_quantizer	u(3)
}	
if(asps_raw_3d_offset_bit_count_explicit_mode_flag)	
ath_raw_3d_offset_axis_bit_count_minus1	u(v)
if(ath_type == P_TILE && num_ref_entries[RlsIdx] > 1) {	
ath_num_ref_idx_active_override_flag	
if(ath_num_ref_idx_active_override_flag)	
ath_num_ref_idx_active_minus1	ue(v)
}	
}	
ath_drm_enabled_flag	u(1)
if(ath_drm_enabled_flag) {	
ath_multi_key_flag	u(1)
ath_crypt_byte_block	u(4)
ath_skip_byte_block	u(4)
if(ath_multi_key_flag)	
ath_key_count	u(16)
else	
ath_key_count_minus1 = 0	
for(i=0; i<=ath_key_count_minus1; i++) {	
ath_per_sample_iv_size	u(8)
for(j=0; j<16; j++)	
ath_key_id[j]	u(8)
if(ath_per_sample_iv_size == 0) {	
ath_constant_iv_size_minus1	u(8)
for(j=0; j<ath_constant_iv_size_minus1; j++)	
ath_constant_iv[j]	u(8)
}	
}	
}	
byte_alignment( )	
}	

ath_drm_enabled_flag defines whether DRM encryption is applied for each tile.
ath_multi_key_flag indicates that the multiple key versions of the tile are used. If this flag is set, multiple keys will be described for current tile: otherwise, a single key is described for current tile.
ath_crypt_byte_block specifies the count of the encrypted tile in the protection pattern, where each block is of size 16-bytes.
ath_skip_byte_block specifies the count of the unencrypted tile in the protection pattern.
ath_key_count_minus1 plus 1 indicates the number of keys that may apply to a sample associated to the current tile. It is not required that a sample associated with current tile uses all keys described.
ath_key_id is the key identifier used for the current tile. The size of ath_key_id shall be in the range of 0 to 15, inclusive.
ath_per_sample_iv_size is the initialization vector size in bytes for samples in the current tile.
ath_constant_iv_size_minus1 plus 1 is the size of a possible initialization vector used for current tile.

atlas_constant_iv, if present, is the initialization vector used for current tile. The size of ath_constant_iv shall be in the range of 0 to ath_constant_iv_size, inclusive.

6 Conclusion

This document proposes a framework for DRM encryption to heterogeneous volumetric video objects within the V3C bitstream. By enabling DRM options at the AD level, this approach allows customized encryption settings per atlas or tile, facilitating secure management of diverse content types such as V-PCC, MIV, and other volumetric content types. Leveraging atlas and tile-based structures, the proposed method applies distinct encryption levels according to the security needs of each object, supporting robust protection for various media applications and flexible integration across different volumetric content. Furthermore, this structure is expected to support the DRM encryption to both heterogeneous and homogeneous volumetric video objects within the V3C bitstream. This document recommends further study for application of DRM in MIV, V-PCC, and V-DMC, for successful commercialization of volumetric video streaming.

Patent rights declaration(s)

Sungkyunkwan University may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).

Reference

[1] Z. Zhu, X. Jin, L. Yu, "[V3C JEE2] Proposal of enabling heterogeneous sources in a same video bitstream of V3C bitstream", ISO/IEC JTC 1/SC 29/WG 4 m59560, Online, April 2022.

[2] Z. Zhu, X. Jin, L. Yu, Y. Yu, V. Zakharchenko, D. Wang, "Support of heterogeneous contents in the same video stream of V3C bitstream, ISO/IEC JTC 1/SC 29/WG 7 m61867, Online, January 2023.

[3] "Text of ISO/IEC FDIS 23090-2 3rd edition Reference software and conformance", ISO/IEC JTC 1/SC 29/WG 11 w18642, Gothenburg, July 2019.

[4] "ATSC Standard: Signaling, Delivery, Synchronization, and Error Protection", Advanced Television Systems Committee, ATSC A/331:2024-04, April 2024.

[5] J. Boyce, B. Salahieh, "MIV frame packing and extended profile", ISO/IEC JTC1/SC29/WG11 M54491, Virtual, July 2020.

[6] B. Salahieh, G. Nadaf, J. Boyce, "Frame Packing Implementation in TMIV", ISO/IEC JTC 1/SC 29/WG 4 m56827, Online, April 2021.

[7] J. -B. Jeong, S. Lee, E. -S. Ryu, "[MIV] Geometry Packing Implementation in TMIV for Frame Packed Video", ISO/IEC JTC 1/SC 29/WG 4 m59442, Online, April 2022.

[8] J. Fleureau, F. Thudor, G. Briand, T. Tapie, R. Gendrot, R. Doré, "MIV/MPI on Android", ISO/IEC JTC 1/SC 29/WG 4 m56727, Online, April 2021.

[9] "Text of ISO/IEC FDIS 23001-7 4th edition Common encryption in ISO base media file format files", ISO/IEC JTC 1/SC 29/WG 3 n00412, Online, October 2021.

[10] Google, "shaka-packager," GitHub repository. Available: https://github.com/shaka-project/shaka-packager

[11] "Text of ISO/IEC 23090-5 DIS Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC) 3rd edition", ISO/IEC JTC1/SC29/WG7 output document n00849, Online, January 2024.

[12] "Text of ISO/IEC DIS 23090-12 MPEG immersive video (2nd edition)", ISO/IEC JTC1/SC29/WG4 output document n00533, Sapporo, July 2024.