2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024

# Enhancing Automotive Security and Safety through CSEv2.0 Test Vector Validation with Synopsys MIPI CSI VIP

Subtitle as needed (style: paper subtitle)

Venkata Naga Srideepti Pisipati, SYNOPSYS, Hyderabad, India (*deeptip@synopsys.com*)

Andrew Elias, NVIDIA, Ottawa, Canada (*aelias@nvidia.com*)

***Abstract*— The MIPI CSE v2.0 specification is a complex and multi-configurable specification offering a multitude of configurations to accommodate diverse security requirements for a broad spectrum of camera devices at varying cost points. This presents a challenge for controllers, especially those designed to interface with numerous camera devices in the field, as they need to support an extensive range of configurations while ensuring seamless interoperability with different camera types.**

**In collaboration, NVIDIA and Synopsys have generated a comprehensive set of test vectors. These vectors serve as a valuable resource for both controllers and camera sensors, expediting the development process of CSE v2.0. Leveraging Synopsys' VIP enhances the confidence in the implementation of these devices. The test vectors cover a spectrum of security variant configurations supported by CSE v2.0, contributing to a robust and well-tested implementation.**

## I. Introduction

The automotive industry is undergoing a significant transformation to address rising requirements encompassing bit rates, frame resolution as well as functional safety and security. To safeguard image sensor data from security threats, MIPI Alliance has introduced a security framework designed for camera connectivity known as MIPI Camera Service Extension v2.0 (MIPI CSE 2.0), which is a collaboration between the MIPI Security and MIPI Camera working groups. This new framework successfully aligns with specific automotive standards, including ISO26262 for safety and ISO21434 for security compliance. In addition, this MIPI security framework can be implemented wherever CSI-2 is supported.

The MIPI CSE v2.0 standard is a complex specification that facilitates multi-configurable camera support for diverse security configurations. Addressing the breadth of these configurations is challenging yet essential to guarantee seamless interoperability among different devices.

This paper explains the collaborative effort undertaken by NVIDIA and Synopsys to create and validate a suite of security test vectors. These vectors significantly enhance confidence in the reliability and accuracy of the dataset used for the authentication and encryption algorithms integral to the MIPI CSE v2.0 specification.

## II. MIPI CSE overview and Verification Challenges

### A. *What is MIPI CSE?*

The automotive industry has traditionally relied on camera applications and/or SerDes protocols to implement its own security measures for facilitating secure communication between camera sensors and controllers. To address the growing risks and threats related to data manipulation and privacy concerns, the MIPI Alliance has introduced a novel security framework known as MIPI CSE v2.0, integrated within the MIPI Automotive SerDes Solutions (MASS). This innovative framework adds CSE Security Service Extensions atop the existing Camera Serial Interface 2 (CSI-2) protocol. When security is enabled, the source pixel data is encrypted (optionally) and authenticated based on the Security Variants (SV) configuration negotiated earlier. On the sink (at the receiving end), this security configuration is validated and decrypted (if encrypted) before further processing by the application layer. This flow is captured in Figure 1 below.

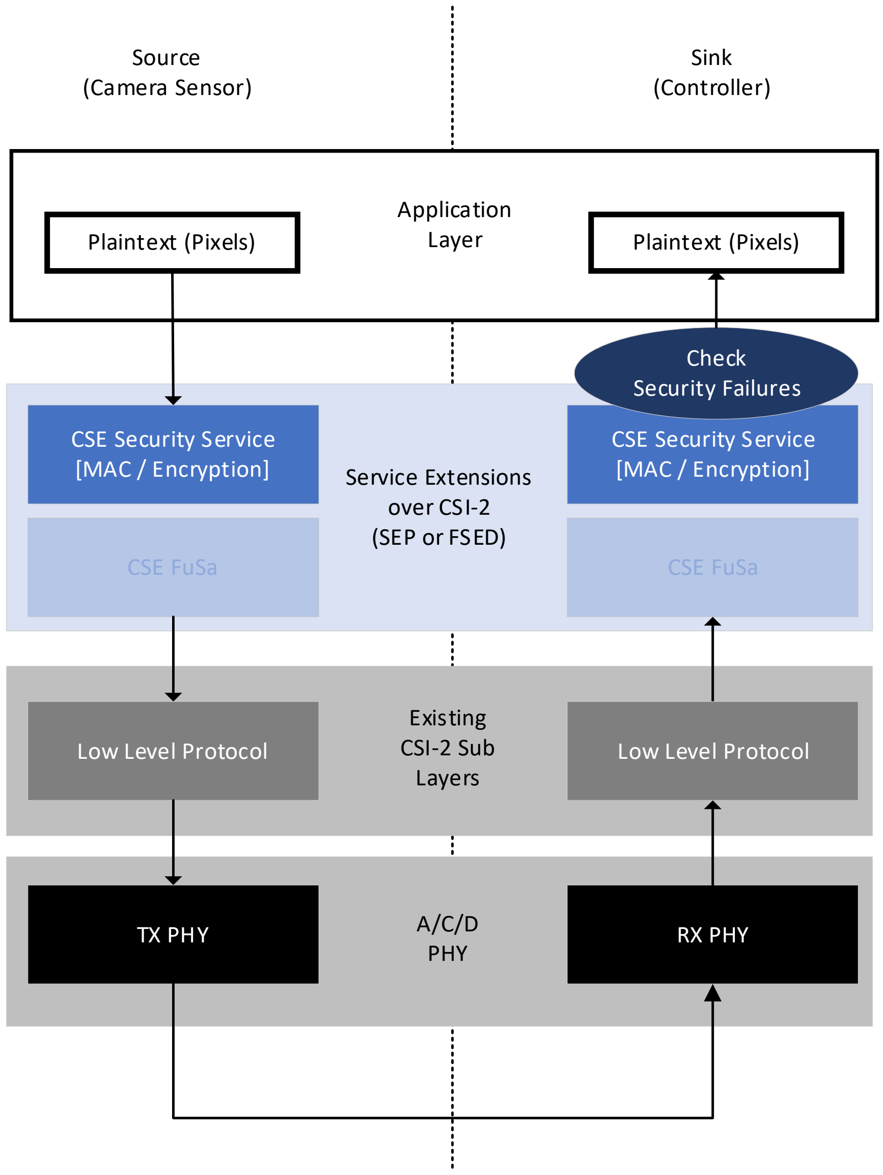

Figure 1: MIPI CSE 2.0 Framework

The CSE v2.0 security framework offers flexibility, allowing vendors to selectively implement security configurations. For instance, vendors that do not plan to support encryption are not required to include the AES-CTR algorithm. However, if a vendor decides to support a specific security profile, they must implement the corresponding security configurations within that profile. A camera and controller vendor can choose to support only a subset of these security profiles.

In this paper, we will focus on the security algorithms supporting different security profiles and present a methodology for validating these algorithms using test vectors.

## B. *MIPI CSE Security Configuration*

To process the Data Service Protocol (DSP) Service Extension Packet (SEP) and Frame-Based Service Extension Data (FSED) packets, the MIPI CSE v2.0 security service extensions offer support for authentication and encryption cryptographic algorithms. The designated algorithms include:

- NIST SP800-38D AES-GMAC Authentication Algorithm
- NIST SP800-38B AES-CMAC Authentication Algorithm
- NIST SP800-38A AES-CTR Encryption Algorithm

These cryptographic algorithms operate in tandem with diverse security configurations outlined in the MIPI CSE v2.0 framework. These configurations encompass:

- Various security levels using distinct SV types as detailed in Table 1
- Placement and parsing of security configurations and protection of the DSP data for SEP and FSED packets
- Defining the cryptographic algorithm type and key size

Each SV type within the MIPI CSE v2.0 framework provides different levels of security protection for the frame. For example, SV-1 ensures comprehensive frame integrity protection with data encryption, representing the highest level of security. However, SV-3 and SV-4 provide partial frame integrity protection, offering a lower level of security. Both the source and sink entities negotiate to determine the SV type used for protection. Additionally, if necessary, negotiations also extend to the parameters associated with the selected SV type. An example includes the negotiation of configuration settings such as Start/Keep/Skip/Stop parameters, used for SV-3 Line Stride Pattern (L-SP) and SV-4 Pixel Block Stride Pattern (PB-SP) configurations. This negotiation process ensures alignment in security measures between the transmitting and receiving ends of the communication.

| Security Variant (SV) | Description |
|---|---|
| SV-1 | Full Integrity, Encryption |
| SV-2 | Full Integrity, No Encryption |
| SV-3 | Line Stride Pattern, No Encryption |
| SV-3a | First Line Stride Pattern, No Encryption |
| SV-4 | Pixel Stride Pattern, No Encryption |
| SV-VD | Vendor Defined |
| SV-5 | No Integrity, No Encryption |

Table 1: CSE v2.0 Security Variants

The support for SV types varies based on factors such as the DSP type (SEP vs. FSED), Tag Mode, and Frame Partition (FP) type. This support can be categorized as mandatory, optional, or not available depending on the specific combination of these parameters. Furthermore, the utilization of AES-GMAC may occur with or without AES-CTR encryption, and AES-CMAC can also be employed. It's noteworthy that the key and MAC size remain independent of this configuration. This diverse range of support configurations is collectively called the CSE Security Landscape. The specific support scenarios for different Tag Modes are thoroughly detailed in the MIPI CSE v2.0 specification, providing a comprehensive reference for understanding the security landscape associated with various DSP types and configurations.

The parsing of security configurations varies for each DSP type. In the case of SEP, the security services are encapsulated within ePH3/ePF1 while for FSED, the security services are encompassed within the FSED Security Field (FSF). Authentication algorithms play a crucial role in safeguarding the integrity of both the header and payload information in each packet. In contrast, encryption is dedicated to preserving the confidentiality of the payload information alone.

With AES-GMAC and AES-CTR, it’s important to note the construction of the initial value (IV) will be carried out using the header information.

Due to factors like cost considerations, different vendors of CSE devices may opt to implement only a subset of configurations and therefore, the specification defines a mandatory minimum configuration for various profiles. These are known as the CSE Security Profiles and describe the minimum DSP type support based on the

SV type for selected Security Tag Modes. In addition, the specification identifies the minimum supported algorithm must use AES-128-GMAC with a 16 Byte MAC size.

### C. *Verification of MIPI CSE Security Configuration*

The CSE Security Landscape for DSP type and Security Tag Mode describes a variety of security configurations that can be supported. The CSE Security Profile provides a subset of configurations for CSE device vendors to use if they want to support a subset of the security configurations. That said, within these CSE Security Profiles, there are still a variety of security parameters that need to be verified.

In either case, verifying these security configurations will be a challenge. To overcome this, NVIDIA and Synopsys generated a set of security test vectors. These vectors are intended to be used by the Controller and Camera sensors to help accelerate the development of these CSE v2.0 devices by providing higher confidence in the implementation, thereby reducing the interoperability risk.

## III. VERIFICATION IP (VIP) EASE OF USE

As protocol complexity continues to grow and evolve, the infrastructure needed for their verification must also become more sophisticated. The use of third-party Verification IP has become widespread, but the ease of adoption varies across the different vendor tools. These IPs come in various forms, differing in methodologies and implementation languages. Therefore, from a user perspective, it is beneficial to invest effort into developing an efficient strategy for adopting third-party Verification IPs. Any Verification IP, which adheres to System Verilog and UVM methodology can be used to verify the Security test vectors. But the effectiveness of this approach will depend on the nature of the security tests and the capabilities of the VIP. Using VIPs that are tailored for these security protocols will ensure that the test vectors align well with the protocol requirements and can cover various security scenarios comprehensively. Security-related IPs, such as those handling cryptographic protocols or secure communication standards (e.g., AES), often have specialized VIPs designed for verifying the security specific algorithms. Security testing often involves custom or proprietary test vectors that are designed to probe for specific vulnerabilities or compliance with security standards. Features like transaction-level modeling, protocol checking, and error injection are important for thorough security testing. UVM provides a flexible framework for creating testbenches and generating test vectors. By leveraging UVM's capabilities, the user can develop sophisticated test scenarios that include security-specific case. UVM's constrained random generation and functional coverage can help in exploring a wide range of potential security issues. In the current scenario, we had chosen Synopsys VIP, which uses UVM methodology with System Verilog concepts to establish a structured approach for verifying the security test vectors. The configurability in the VIP infrastructure is crucial for verification of CSE algorithms. To verify the test vectors, Synopsys VIP provides configuration classes for both Transmitter and Receiver agents with different configuration variables for the user to program values to the test vectors. Synopsys VIP helps generate appropriate stimulus to exercise all verification requirements and monitor the responses and ensure they are correct. Synopsys VIP can be used with any IP for verifying the underlying protocol of MIPI CSE.

## IV. VALIDATION USING TEST VECTORS

Two sets of validation vectors are in the appendix of the CSE v2.0 specification. Each set represents a specific security configuration for SEP and FSED and both are part of the CSE Security Profile described in the specification.

### A. *SEP Tag Mode 1a SV-1*

This configuration targeting SEP Tag Mode 1a using Security Variant 1 (SV-1) includes full AES-128-GMAC integrity and AES-128-CTR encryption. The vector information includes:

a) Frame configuration including the settings for Frame, CSE_TX_CONFIG, CSE_SEP_CONFIG, CSE_SA_CONFIG, CSE_CTRL_CONFIG, CSE_AFC_FC_STATE, etc.
b) Example of the encryption flow for Line 0 of the FP-3 Middle Block

c) A detailed description of the payload input byte and ordering of the CSI-2 packet. In addition, the intermediate values including ePHs, UVs and GMAC data blocks
d) A corresponding detailed description of the payload output byte and ordering of the CSI-2 packet

### *B. FSED Tag Mode 2a FSED_CTRL_SYNC*

This configuration targeting FSED Tag Mode 2a uses AES-128-GMAC for authentication of the FSED_CTRL_SYNC which is transmitted after FP-1 and before FP2. Note, the FSED_CTRL_SYNC message is on every FSED and applies to all supported FSED SV types. From a test vector perspective, the information is provided with the following exceptions:

- Provides example of the authentication flow of the FSED CTRL_SYNC
- A detailed description of the input/output payload based on the FSED message

### *C. Using Synopsys VIP to Generate the NVIDIA/Synopsys Vectors for AES Processing*

Synopsys VIP can be used to generate the AES-GMAC and AES-CTR vectors used in both SEP and FSED test vectors. Each test requires:

- Input configuration data
- Input data
- Expected output data (golden reference)

For AES-GMAC, the expected output data is the MAC tag. Whereas for AES-CTR, it's encrypted data. In both cases, it provides the correctness of the output for each test.

The last vector included in this section is AES-CMAC. It is not part of the vectors provided in the CSE v2.0 specification but it is supported as one of the authentication algorithms camera vendors can choose to use. Note, AES-CMAC is not included as one of the primary profile supports in CSE Security Profile.

**Configuration Data**

Base configuration for the AES-GMAC, AES-CTR and AES-CMAC tests

```
// To set user defined value of frame number set below configuration to 0
cfg.xmtr_cfg.csi2_cfg.frame_number_override_en = 0;
cfg.rcvr_cfg.csi2_cfg.frame_number_override_en = cfg.xmtr_cfg.csi2_cfg.frame_number_override_en;
// To set user defined value of additional_frame_number set below configuration to 0
cfg.xmtr_cfg.csi2_cfg.additional_frame_number_override_en = 0;
          cfg.rcvr_cfg.csi2_cfg.additional_frame_number_override_en =
cfg.xmtr_cfg.csi2_cfg.additional_frame_number_override_en;
// Configuration to set Additional frame number
cfg.xmtr_cfg.csi2_cfg.additional_frame_number_using_cfg = 'h425241;
cfg.rcvr_cfg.csi2_cfg.additional_frame_number_using_cfg =
          cfg.xmtr_cfg.csi2_cfg.additional_frame_number_using_cfg;
// Configuration to set source_id value
cfg.xmtr_cfg.csi2_cfg.source_id = 'h39;
cfg.rcvr_cfg.csi2_cfg.source_id = cfg.xmtr_cfg.csi2_cfg.source_id;
// Key was configured from the testcase using below configuration
cfg.xmtr_cfg.csi2_cfg.traffic_key_block.push_back(temp[127:0]);
cfg.xmtr_cfg.csi2_cfg.traffic_key_block.push_back(temp[255:128]);
cfg.rcvr_cfg.csi2_cfg.traffic_key_block = cfg.xmtr_cfg.csi2_cfg.traffic_key_block;
```

## V. AES-GMAC AUTHENTICATION

AES-GMAC authentication algorithm is used to authenticate the frame data. AES-GMAC algorithm authenticates this data by generating an authentication tag using the Initial Value (IV), 128 bit or 256 bit Key and

Additional Authenticated Data (AAD). Note, the AAD includes SEP and FSED header data and payload. The VIP configuration below shows configuring the key, IV and input data (AAD) to use with the AES-GMAC algorithm.

```
// Configuration to select GMAC Mac algorithm 0 select AES GMAC 128bit and 2 select AES GMAC 256 bit

cfg.xmtr_cfg.csi2_cfg.cse_i_sec_sa_config_register[1][7:4] = 0; // AES GMAC 128 bit
cfg.rcvr_cfg.csi2_cfg.cse_i_sec_sa_config_register[1][7:4] =
    cfg.xmtr_cfg.csi2_cfg.cse_i_sec_sa_config_register[1][7:4];
```

## A. *Example Test Vectors Shared by NVIDIA*

GMAC_KEY = 6cb150ab3150513fe5fd79097e3cb628
GMAC_IV = 390640000004252415640f0000
GMAC_AAD = 023000060200000000000039504f560
GMAC_TAG = 7e7a7cc2f1cd5b7bb9291cdd2efcc934

## B. *Sequence Code*

```
`uvm_do_with(req,{
                req.transmission_mode == svt_mipi_csi2_packet::HS_MODE;
                req.data_type == svt_mipi_csi2_packet::SEP;
                req.ephen == 'h2;
                req.epfen == 2'b10;
                req.extended_data_type == FS;
                req.payload_descriptor == 'h0;
                req.evc == 'h6;
        get_response(rsp);  // wait for transaction to complete
});
```

## C. *Expected/Obtained Results*

| NVIDIA test vectors | Synopsys VIP Output | Description |
|---|---|---|
| GMAC_KEY = 6cb150ab3150513fe5fd79097e3cb628<br><br>GMAC_IV = 390640000004252415640f0000<br><br>GMAC_AAD = 023000060200000000000039504f560<br><br>GMAC_TAG = 7e7a7cc2f1cd5b7bb9291cdd2efcc934 | K = 6cb150ab3150513fe5fd79097e3cb628<br><br>A [1]= 023000060200000000000039504f560000<br><br>IV = 390640000004252415640f0000<br><br>H = 15fc72851089785380511673Bbab3ea1<br><br>Y [0] = 390640000004252415640f000000000001<br><br>E(K,Y[0] = fc2c2160a756fa19f00d8d97629596b1<br><br>X[1] = 752131e95de30362ff7282abb7dac71d<br><br>GHASH(H,A,C) = 82565da2569ba1624924914a4c695f85<br><br>T = 7e7a7cc2f1cd5b7bb9291cdd2efcc934 | • In this example the vectors are generated and validated for SEP FS Packet<br>• AES GMAC algorithm with 128-bit Key and user defined FC and AFC values are being transmitted<br>• SNPS Output Mac Tag has been compared with NVIDIA expected results and thus verified the vectors |

Table 2: Simulation results with NVIDIA test vectors for AES GMAC

## VI. AES-CTR ENCRYPTION

AES-CTR encryption requires configuring the IV, 128 bit or 256 bit key along with providing the input data. The input data is plain-text on the camera sensor and cipher-text on the controller. The Synopsys VIP configuration example below demonstrates how to configure the key, IV and input data for the AES-CTR.

```
// Configuration to select Mac algorithm 1 select AES CTR 128bit and 3 select AES GMAC 256 bit
cfg.xmtr_cfg.csi2_cfg.cse_i_sec_sa_config_register[1][7:4] = 1; // AES CTR 128 bit Encryption
cfg.rcvr_cfg.csi2_cfg.cse_i_sec_sa_config_register[1][7:4] =
cfg.xmtr_cfg.csi2_cfg.cse_i_sec_sa_config_register[1][7:4];
```

### A. *Example Test vectors shared by NVIDIA*

CTR_KEY = 735d549a41277e18bc922c59200fbab8
CTR_IV = 390640000042524156 4f000100000000
CTR PLAINTEXT = 000102030405060708090a0b0c0d0e0f 1011121314151617
CTR CYPHERTEXT = 8ee92bd463d9ce2f075f352d7b7b7ca3 ad00c3a7749171df

### B. *Sequence code*

```
`uvm_do_with(req,{
            req.transmission_mode == svt_mipi_csi2_packet::HS_MODE;
            req.data_type == svt_mipi_csi2_packet::SEP;
            req.ephen == 'h2;
            req.epfen == 2'b10;
            req.extended_data_type == 'h2a;
            req.payload_descriptor == 'h0;
            req.evc == 'h6;
            req.payload_length == 'h18;
            req.extended_payload[0] == 'h00;
            req.extended_payload[1] == 'h01;
            req.extended_payload[2] == 'h02;
            req.extended_payload[3] == 'h03;
            req.extended_payload[4] == 'h04;
            req.extended_payload[5] == 'h05;
            req.extended_payload[6] == 'h06;
            req.extended_payload[7] == 'h07;
            req.extended_payload[8] == 'h08;
            req.extended_payload[9] == 'h09;
            req.extended_payload[10] == 'h0a;
            req.extended_payload[11] == 'h0b;
            req.extended_payload[12] == 'h0c;
            req.extended_payload[13] == 'h0d;
            req.extended_payload[14] == 'h0e;
            req.extended_payload[15] == 'h0f;
            req.extended_payload[16] == 'h10;
            req.extended_payload[17] == 'h11;
            req.extended_payload[18] == 'h12;
            req.extended_payload[19] == 'h13;
            req.extended_payload[20] == 'h14;
            req.extended_payload[21] == 'h15;
            req.extended_payload[22] == 'h16;
            req.extended_payload[23] == 'h17;
/** wait for transaction to complete */
     get_response(rsp);
});
```

### C. *Expected/Obtained Results*

| NVIDIA test vectors | Synopsys VIP Output | Description |
|---|---|---|
| CTR_KEY = 735d549a41277e18bc922c59200fbab8<br><br>CTR_IV = 39064000004252415 64f000100000000<br><br>CTR PLAINTEXT = 000102030405060708090a0b0c0d0e0f 1011121314151617<br><br>CTR CYPHERTEXT = 8ee92bd463d9ce2f075f352d7b7b7ca3 ad00c3a7749171df | K = 735d549a41277e18bc922c59200fbab8<br><br>P[1] = 000102030405060708090a0b0c0d0e0f<br>P[2] = 10111213141516170000000000000000<br><br>IV = 39064000004252415 64f0001<br><br>Y[0] = 39064000004252415 64f000100000000<br><br>E(K,Y[0] = 8ee829d767dcc8280f563f26777672ac<br>E(K,Y[1] = bd11d1b4608467c8b59ca02d93c721bc<br>E(K,Y[2] = b7183b6e8eb6b4798a06c0ba2bc228b6<br><br>C[1] = 8ee92bd463d9ce2f075f352d7b7b7ca3<br>C[2] = ad00c3a7749171df0000000000000000 | • In this example the vectors are generated and validated for SEP FS Packet |

Table 3: Simulation results with NVIDIA Test vectors for AES-CTR

## VII. AES-CMAC AUTHENTICATION

AES-CMAC authentication algorithm is used to verify the integrity of the frame data by generating an authentication tag using 128 bit or 256 bit Key and passing in the input data (message). Unlike the AES-GMAC configuration, which requires an IV input, the AES-CMAC VIP configuration only includes the key and input data.

```
// Configuration to select Mac algorithm 8 select AES CMAC 128bit and 9 select AES CMAC 256 bit
cfg.xmtr_cfg.csi2_cfg.cse_i_sec_sa_config_register[1][7:4] = 8; // AES CMAC 128 bit
cfg.rcvr_cfg.csi2_cfg.cse_i_sec_sa_config_register[1][7:4] =
cfg.xmtr_cfg.csi2_cfg.cse_i_sec_sa_config_register[1][7:4];
```

### A. *Example Test Vectors Shared by NVIDIA*

CMAC_KEY = 6cb150ab3150513fe5fd79097e3cb628
INPUT DATA = 0230000602000000000039504f56415242
MSG LENGTH = ‘h11
CMAC_TAG = 1f1fa7ad70118b2adc56e8f05a613de0

### B. *Sequence code*

```
`uvm_do_with(req,{
             req.transmission_mode == svt_mipi_csi2_packet::HS_MODE;
             req.data_type == svt_mipi_csi2_packet::SEP;
             req.ephen == 'h2;
             req.epfen == 2'b10;
             req.extended_data_type == FS;
             req.payload_descriptor == 'h0;
             req.evc == 'h6;
/** wait for transaction to complete */
      get_response(rsp);
});
```

### C. *Expected/Obtained Results*

| **NVIDIA test vectors** | **Synopsys VIP Output** | **Description** |
|---|---|---|
| CMAC_KEY = 6cb150ab3150513fe5fd79097e3cb628<br><br>INPUT DATA = 023000060200000000039504f56415242<br><br>MSG LENGTH = ‘h11<br><br>CMAC_TAG = 1f1fa7ad70118b2adc56e8f05a613de0 | *K = 6cb150ab3150513fe5fd79097e3cb628*<br><br>*L = 15fc7285108978538051 16738bab3ea1*<br><br>*K1 = 2bf8e50a2112f0a700a22ce717567d42*<br>*K2 = 57f1ca144225e14e014459ce2eacfa84*<br><br>*Msg Data[0] = 023000060200000000039504f564152*<br><br>*Padded Msg = 42800000000000000000000000000000*<br><br>*Msg Length = 00000011*<br><br>*T = 1f1fa7ad70118b2adc56e8f05a613de0* | • In this example the vectors are generated and validated for SEP FS Packet |

Table 4: Simulation results with NVIDIA Test vectors for AES-CMAC

## VIII. Conclusion

Advancements in the automotive industry are transforming the future of transportation by incorporating both functional safety and security features. MIPI CSE v2.0 plays a pivotal role in enabling automotive stakeholders to integrate cutting-edge security measures such as tamper detection and camera packet confidentiality. This integration contributes to the creation of a more comprehensive and secure automotive system. Validating these features with test vectors becomes imperative for confirming the correctness of the transmitted packets.

The use of Synopsys MIPI CSI VIP proves instrumental in streamlining the validation process of test vectors, with results showcased at the MIPI Working Group meeting. MIPI CSE Synopsys VIP testbench architecture helped in verifying these test vectors, allowing to present the validation results before the MIPI Org meeting. Leveraging Synopsys MIPI CSI2 v4.0 VIP emerged as an asset in the validation process. This success facilitated the proposal to include the test vectors as part of the annexure in the specification during the Working Group meeting.

## IX. References

[1] MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2®), version 4.0.1,
[2] https://www.synopsys.com/verification/verification-ip/mipi.html