SIGMATA: Storage Integrity Guaranteeing Mechanism against Tampering Attempts for Video Event Data Recorders

The 7th International Multi-Conference on Complexity, Informatics and Cybernatics: IMCIC 2016

Authors: Hyuckmin Kwon, Seulbae Kim, and Heejo Lee

Mar 04, 2016
Presenter: Seulbae Kim

LIBERTAS
JUSTITIA
VERITAS

Background

- VEDR (Video Event Data Recorder)
  - Devices that are installed in a vehicle to record the view through the windshield.
  - The recorded video streams are saved to storage as files.
  - Also known as a dashcam or a car black-box.

LIBERTAS
JUSTITIA
VERITAS

Motivation

- The video data taken from VEDRs constitute **the most important evidence** in the investigation of an accident or crime.

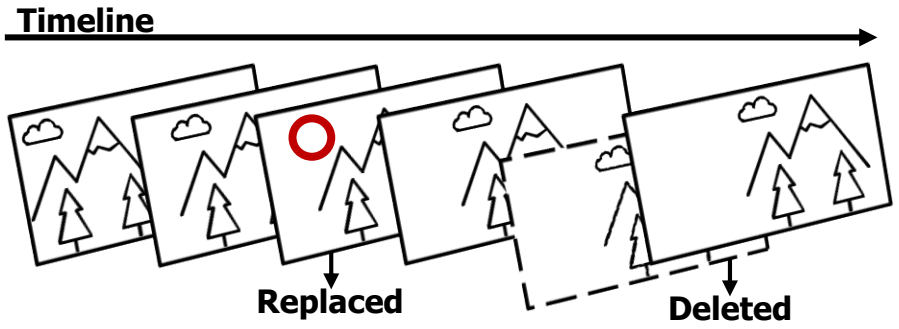
- The owners can **manipulate unfavorable scenes** after accidents or crimes to conceal their recorded behavior.
  - Insert, delete, replace, or reorder the frames.

- Thus, we need to guarantee **"frame-wise integrity"** of VEDR storages, which means the preservation of the
  - Existence
  - Time information
  - Chronological relationship

  of all recorded frames.

Problem Definition

- Detecting **frame-wise forgery** in a VEDR file.
  - Frame-wise forgery: the action of **modifying the byte-sequence** of video frames or **reordering their temporal sequence**.
  - Four types of such forgery:
    - Insertion
    - Deletion
    - Replacement
    - Reordering

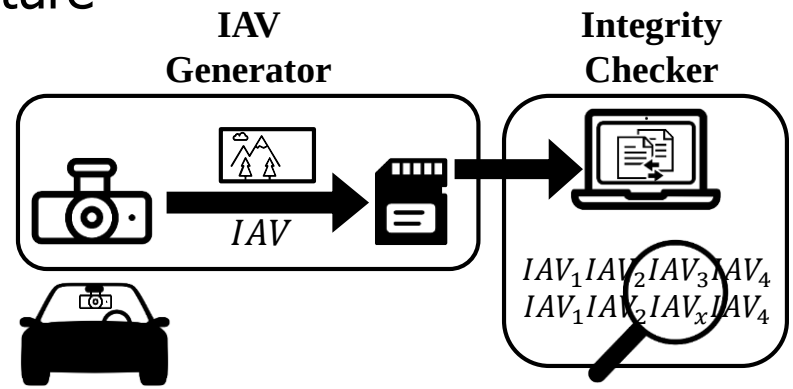


Assumption

- VEDR has a **restricted** operating environment.
  1. Chronological file I/O.
     - The video files of a VEDR are created and stored in chronological sequence.
  2. Isolated device.
     - VEDRs do not support any networking features.
     - Thus, we cannot utilize a remote server to verify integrity.
  3. Open access.
     - The entire body of the VEDR is in the hands of the users, who are simultaneously the adversaries.
     - The adversaries have full access to our underlying technique.

Proposed Mechanism: SIGMATA

- Overall structure



1. IAV Generator
   - In charge of storing the chronological order of frames.
   - Runs during the recording of the video, up to 24 hours a day.
   - Generates integrity assurance values (IAVs) by processing each frame, and saves them in the storage.
2. Integrity Checker
   - Exists independently with the VEDR.
   - Takes advantage of the formerly generated values when it is required, e.g. investigation of a car accident.

LIBERTAS
JUSTITIA
VERITAS

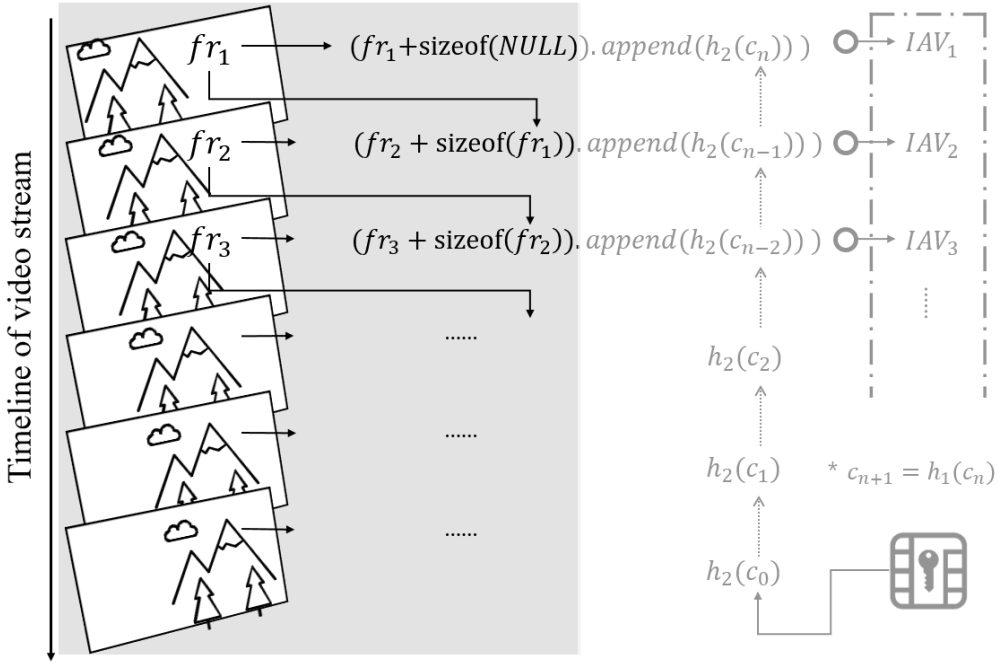
SIGMATA - IAV Generator

- Produces IAVs while the VEDR is recording the video.

- Three steps:
  1. Frame preprocessing
  2. Salted hashing
  3. Storage of the computed IAVs

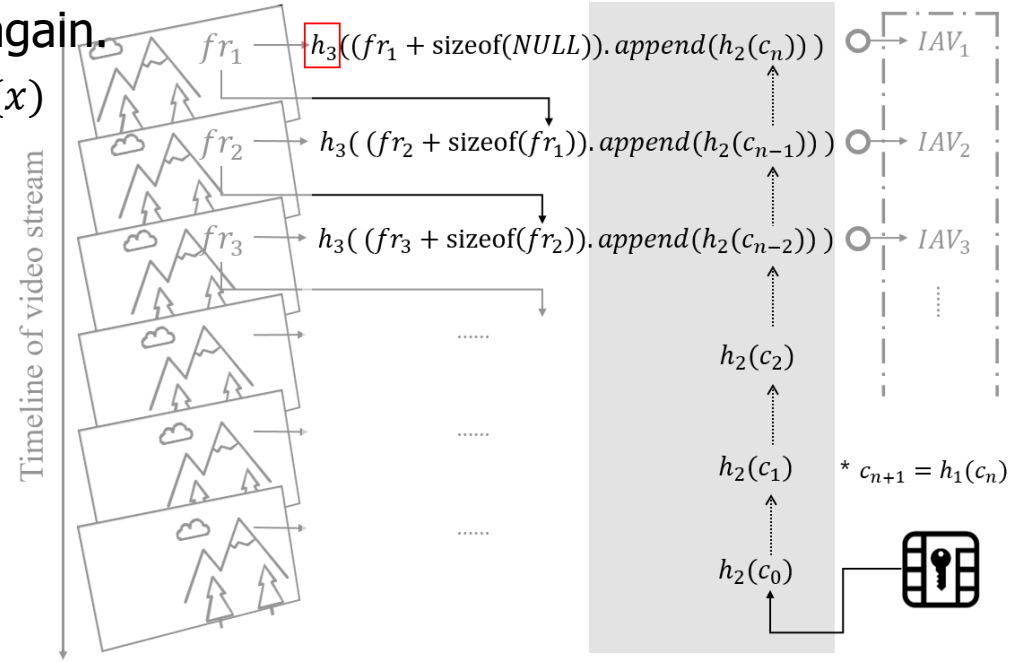
SIGMATA - IAV Generator

- Frame preprocessing
  - Receive a video frame (fr_i) from the VEDR.
  - Then add the size of the previous frame (fr_{i-1}).
  - The resulting value is called "augmented frame."
    - e.g. i^{th} augmented frame is ($fr_i + sizeof(fr_{i-1})$)



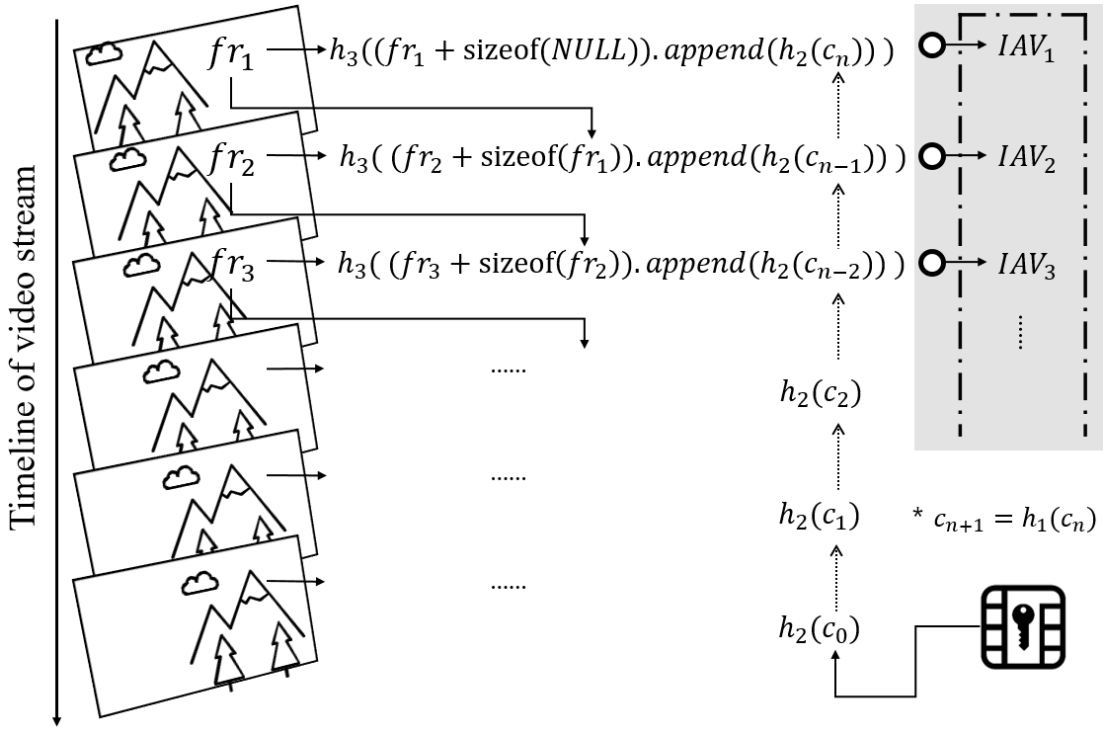
SIGMATA - IAV Generator

- Salted hashing
  - Create a salt.
    - Generate a one-way hash chain of length n, by repeatedly applying a hash function $h_1(x)$ to the elements.
    - Apply another hash function $h_2(x)$ to each element of the chain.
  - Append the salt to each augmented frame.
  - Hash again.
    - $h_3(x)$



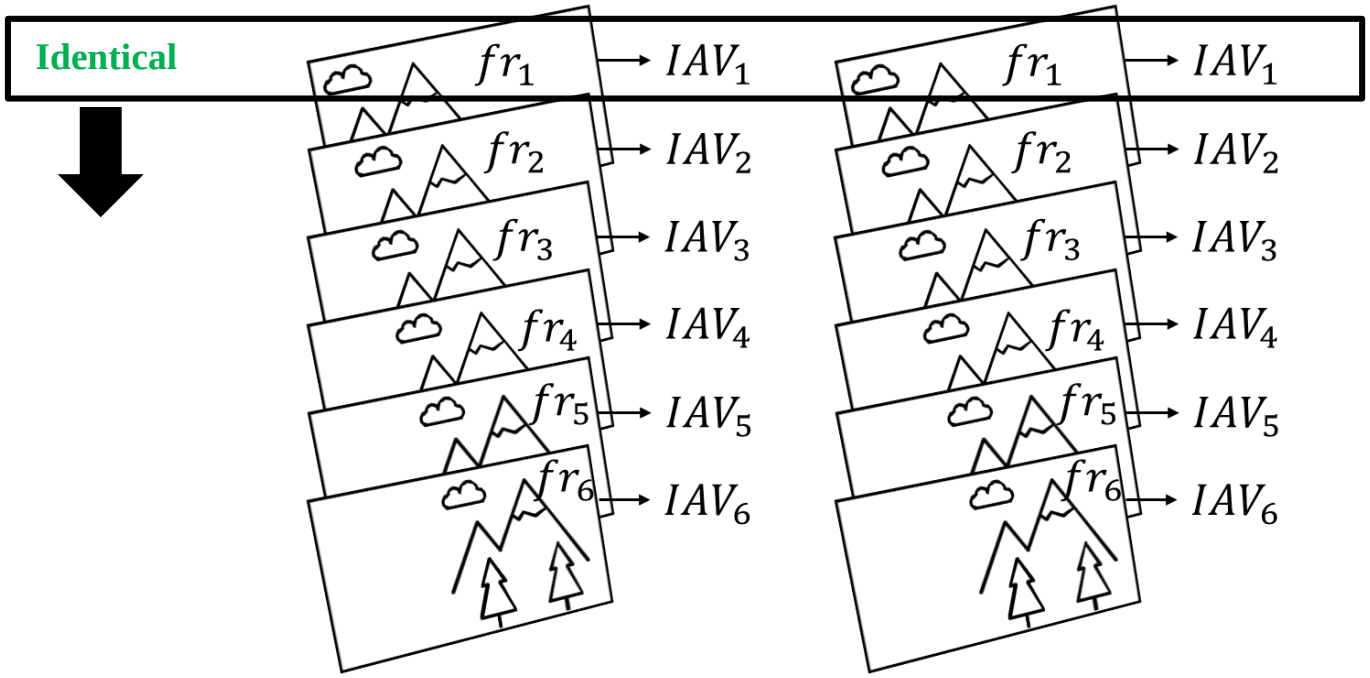
SIGMATA - IAV Generator

- Storage of the computed IAVs
  - Each video frame is transformed into an IAV.
  - Save the consecutive IAVs of the frames in the video storage.



SIGMATA - IAV Checker

- Integrity examination
  - Performs a comparison of two IAV sequences to verify the integrity of frames on the occasion of investigation.

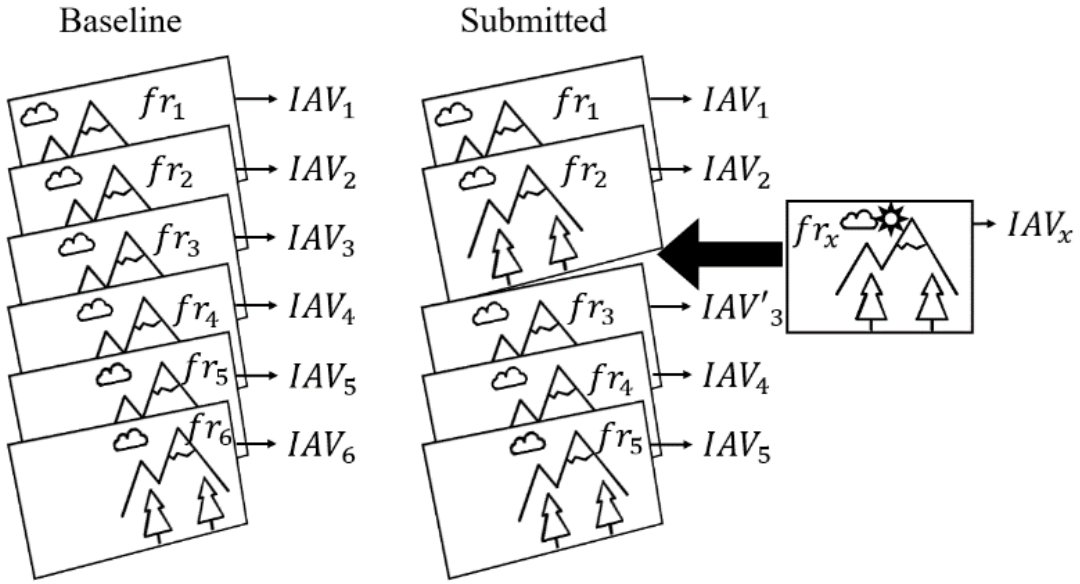


Evaluation

- Attack-suppression scenarios
  - Insertion, deletion, replacement, reordering.
- Security analysis
  - Generation of fake IAVs.
- Feature comparison
  - Comparison with prior works.
- Performance
  - Comparison of encoding time with or without SIGMATA.

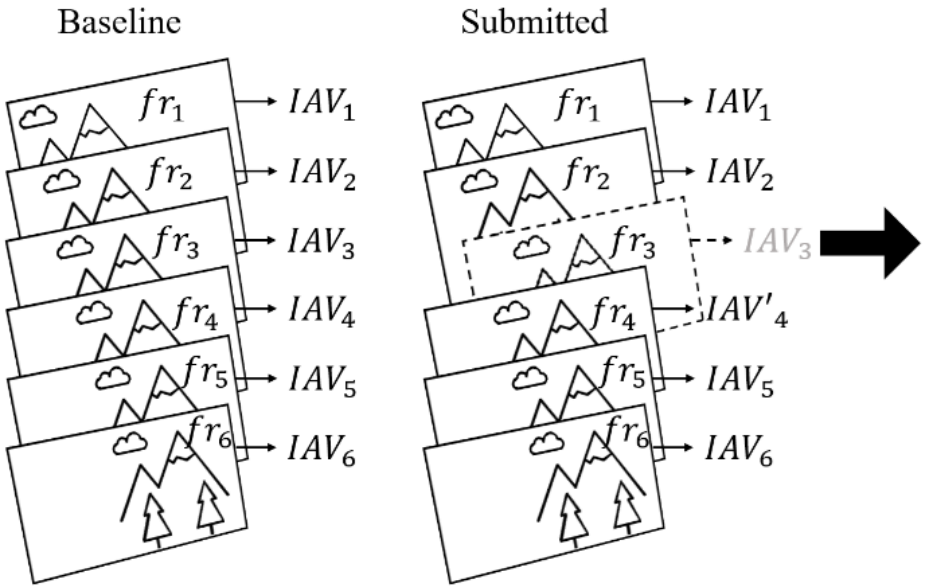
Evaluation - Attack-suppression

- Detection of frame insertion
  - Baseline
    - $Set\ B = \{IAV_1, IAV_2, IAV_3, IAV_4, IAV_5, IAV_6\}$
  - Insertion Attack
    - $Set\ I = \{IAV_1, IAV_2, IAV_x, IAV'_3, IAV_4, IAV_5, IAV_6\}$
    - Previously unseen value is inserted.



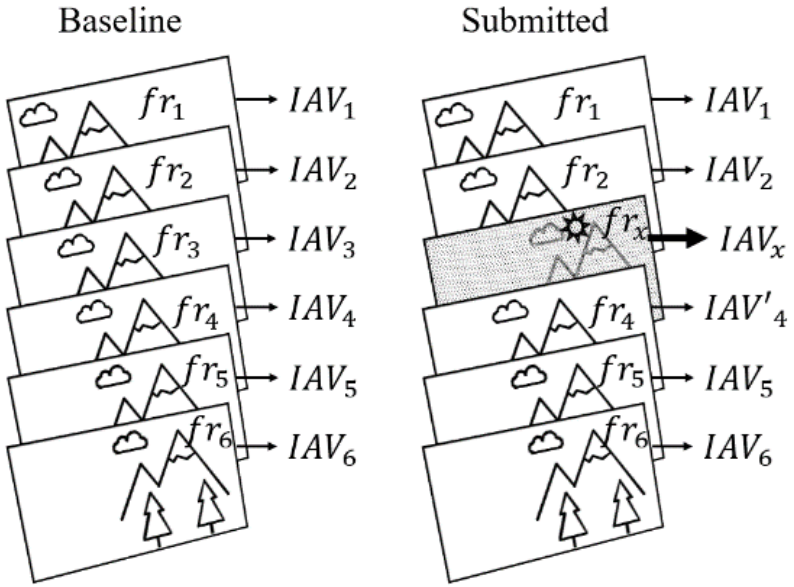
Evaluation - Attack-suppression

- Detection of frame deletion
  - Baseline
    - $Set\ B = \{IAV_1, IAV_2, IAV_3, IAV_4, IAV_5, IAV_6\}$
  - Deletion attack
    - $Set\ D = \{IAV_1, IAV_2, IAV'_4, IAV_5, IAV_6\}$
    - IAV_4 is changed to IAV'_4



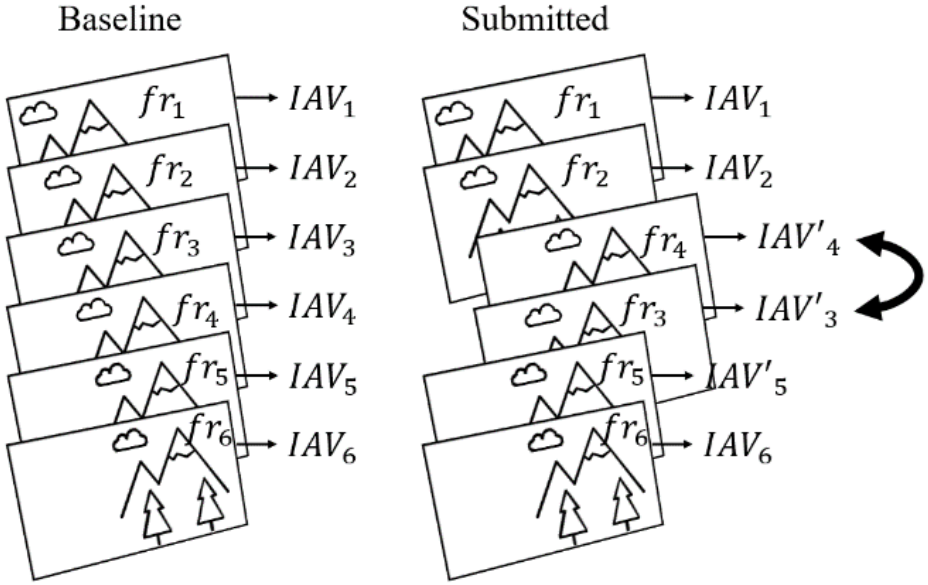
Evaluation - Attack-suppression

- Detection of frame replacement
  - Baseline
    - $Set\ B = \{IAV_1, IAV_2, IAV_3, IAV_4, IAV_5, IAV_6\}$
  - Replacement attack
    - $Set\ RP = \{IAV_1, IAV_2, IAV_x, IAV'_4, IAV_5, IAV_6\}$
    - IAV_3 is missing but the number of IAVs is unchanged.



Evaluation - Attack-suppression

- Detection of frame reordering
  - Baseline
    - $Set\ B = \{IAV_1, IAV_2, IAV_3, IAV_4, IAV_5, IAV_6\}$
  - Reordering attack
    - $Set\ RO = \{IAV_1, IAV_2, IAV'_4, IAV'_3, IAV'_5, IAV_6\}$
    - Supplementary inspection is done to distinguish from replacement attacks.



Evaluation - Security analysis

- **Assumption: The adversary has a thorough knowledge of the mechanism.**

- Generation of fake IAV
  - By deliberately taking advantage of a hash collision to generate the same IAV as the baseline.
  - Three constraints:
    1. Finding the value that causes a hash collision.
    2. Forged frame's size must be of the same size as the original frame.
    3. The forged frame must be visually valid.
  - **Claim: Such an attack is impractical.**

Evaluation - Feature comparison

- Comparison of 8 features

Feature	NCryptFS	Cao et al.	ICAR	SIGMATA
Detection of frame-wise insertion	No	No	No	Yes
Detection of frame-wise deletion	No	No	No	Yes
Detection of frame-wise replacement	No	No	No	Yes
Detection of frame-wise reordering	No	No	No	Yes
Data recovery	No	No	Yes	No
Storage Reusability	Yes	Yes	No	Yes
Network connection required	No	Yes	No	No
Implementation layer	Kernel	Application (server-client)	Kernel	Application (Codec)

Evaluation - Performance

- Experimental setup
  - Raspberry Pi 2
    - 900 MHz quad-core ARM cortex-A7 CPU
    - 1 GB RAM
  - Implementation
    - Modified the FFmpeg encoder. (https://www.ffmpeg.org/)

- Experiment method
  - Used three raw video streams recorded by a VEDR
    - Resolution of 1280 x 720
    - 30 Frames per second
    - 60, 120, 180 seconds long.
  - Compared the encoding time of a raw video stream
    - Without SIGMATA
    - With SIGMATA

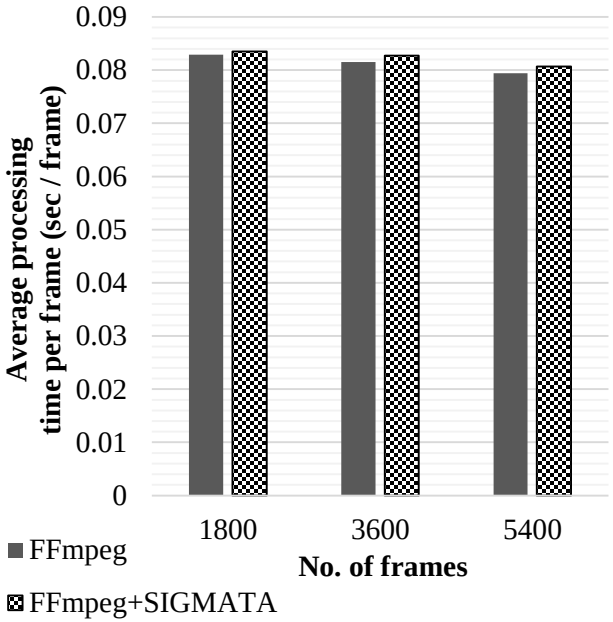
Evaluation - Performance

- Experiment procedure
  1. Decoded the videos to get the raw video stream in YUV format.
  2. Encoded the raw video twice.
     - Once by the unmodified FFmpeg.
     - Once by the FFmpeg in which SIGMATA was implemented.
  3. Preset: 30 FPS, 4:2:0 subsampling, ultrafast mode.

Evaluation - Performance

- Result

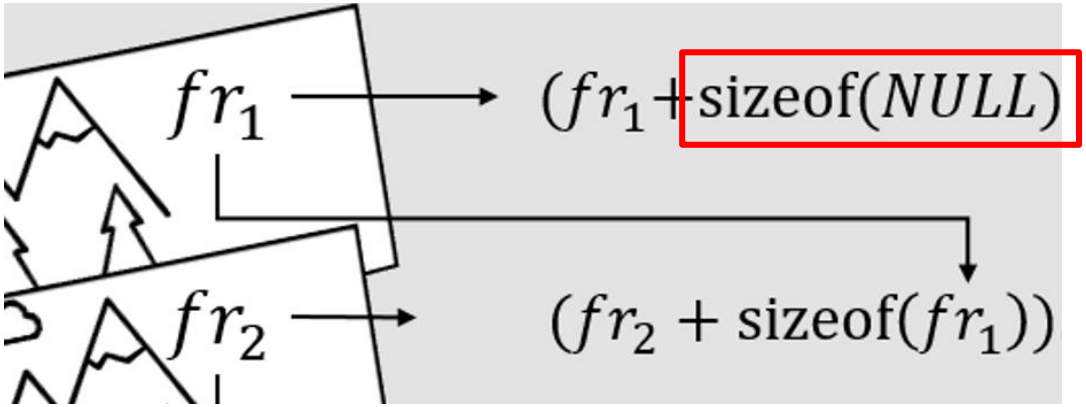
Video	Video 1		Video 2		Video 3	
No. of frames	1,800		3,600		5,400	
Frames per second	30		30		30	
Length (sec)	60		120		180	
SIGMATA applied	No	Yes	No	Yes	No	Yes
Encoding time (sec)	149.30	150.33	293.39	297.84	428.58	436.69
Avg. encoding time/frame	0.0829	0.0835	0.0815	0.0827	0.0794	0.0807



- An average computational overhead of 1.26 % for each frame.

Discussion

- Forgery of the first frame
  - The first frame of the video stream is directly hashed without adding the size of the previous frame, since such a frame does not exist.
    - This may amplify the likelihood of forgery.
  - However, the first frame occupies a small portion, 0.033 sec, of the entire video stream spanning 24 hours.
    - This weakness is negligible.



Discussion

- The use of a user-inaccessible storage
  - We assume the existence of a secure storage
    - Not accessible by users.
    - e.g., Trusted Platform Module (TPM).
  - General VEDRs are ready to utilize such hardware
    - ARMv6 architecture has supported TrustZone since 2001.
    - ARM is the most widespread architectures for embedded processors.
  - For devices that have no such hardware
    - Commercial TPM chips for embedded devices are available.
    - Atmel AT97SC3203S.

LIBERTAS
JUSTITIA
VERITAS

Conclusion

- Proposed a novel concept of frame-wise forgery in VEDR storage and a mechanism named SIGMATA to assure its integrity.
- Solved several problems, including the detection of insertion, deletion, replacement, and reordering of frames.
- Verified the utility of SIGMATA by investigating attack scenarios and conducting a security analysis of the possibility of bypassing SIGMATA.
- Evaluated its performance under Raspberry Pi 2 environment and verified that SIGMATA is applicable to the real-time scenario.

LIBERTAS
JUSTITIA
VERITAS

Thank you

Q & A