Private Space Monitoring with SoC-based Smart Cameras

Ihtesham Haider
Institute of Networked and Embedded Systems
Alpen-Adria-Universität Klagenfurt, Austria
ihtesham.haider@aau.at

Bernhard Rinner
Institute of Networked and Embedded Systems
Alpen-Adria-Universität Klagenfurt, Austria
bernhard.rinner@aau.at

***Abstract*—Cameras and other sensors are increasingly deployed for private space monitoring applications such as home monitoring, assisted/enhanced living and child monitoring. Since these cameras capture highly sensitive information and transfer it over public communication infrastructures, security and privacy is a major concern. This work presents a secure camera device along with a secure data delivery and archiving solution for private space monitoring applications using untrusted public cloud storage services. Integrity, authenticity, confidentiality and freshness of captured data are protected on-board using physically unclonable functions (PUF). The protection holds true for entire lifetime of the data until it is consumed by an authorized end-user. Experimental results obtained from our Zynq7010 SoC-based prototype shows that the device is able to secure videos with 30 frames per second at 640×480 resolution with marginal overhead. The presented solution is not limited to visual sensing but can be applied to a wide range of pervasive sensing and secure data delivery scenarios.**

***Index Terms*—Private space monitoring, smart camera, security, privacy, physically unclonable function, PUF**

I. Introduction

Since visual data contains information about the identities and behaviors of observed individuals, security and privacy of the captured data have always been a critical issue in visual sensor networks (VSNs). With emergence of IoT, VSNs are becoming increasingly popular in various private space monitoring applications such as smart homes, assisted/enhanced living, child monitoring and home security. A generic architecture for such applications is shown in Fig. 1. One or more smart cameras monitor the space for events of interest. Upon event detection, an alert message is sent to an end-user and the video data capturing the event is uploaded to a cloud storage. Upon upload, a push notification is sent by the cloud server to the end-user who then downloads the data.

The growing software stack on today's smart camera nodes and inevitable use of Internet for communications and storage have tremendously increased the attack surface area of these applications. Data security and user privacy are key requirements that must be met for widespread adaption of these applications. Considering the infrastructure of Fig. 1, an attacker may get access to the sensitive image data compromising the privacy of the monitored individuals, introduce fake data in the network by injecting malware in vulnerable software stack of the nodes or gain either full or partial control of some nodes in the network to launch degradation-/denial-of-service attack.

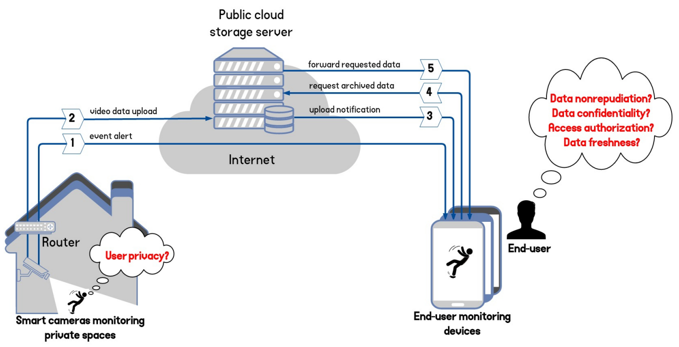

Fig. 1. A high level infrastructure of private space monitoring applications depicting security and privacy requirements

To prevent attackers from mounting these attacks the sensor nodes need to be secured.

Winkler and Rinner [1] have recently surveyed security threats prevalent in today's VSNs and corresponding security requirements to counter these threats. Data protection is typically implemented at application level. Considering that an attacker has modified the underlying OS, middleware or hardware, data security can be completely bypassed. Consequently, node security is a requirement for high-level data protection. According to the survey, data protection encompasses data integrity, authenticity, confidentiality, freshness and access authorization whereas node security subsumes firmware integrity, authenticity, cloning prevention, hardware tamper detection, resistance to side-channel and denial of service attacks.

We present a smart camera node implemented as a system on chip along with a secure data capturing, archiving, and delivery solution using untrusted public cloud storage services. Data and node security is provided on-camera using CMOS fingerprints which are extracted by a physically unclonable function (PUF). The PUF-based solution is lightweight, tightly integrated with the sensor's hardware fabric and can be easily mapped onto existing smart camera nodes.

The contributions of this work are threefold:

- We present a hardware and software architecture of a SoC-based visual sensor node. The node extracts its signing and encryption keys from on-chip PUF. Captured video data and metadata is encrypted-then-signed on camera using the platform-bound security keys to ensure

System	Platform	Software	Computer Vision	Security
Cyclops [2]: 2 fps @64×64	Atmega128 MCU @7.3MHz, XC2C256 CPLD @16MHz, 64kB RAM	Drivers, Libraries, Applications	Hand posture recognition	-
CITRIC [3]: 30 fps @640×480	PXA270 @624MHz, 64MB RAM	Embedded Linux, Drivers, Libraries, Applications	Compression, Background subtraction	Frame encryption
PrivacyCam [4]: 4 fps @320×240	Blackfin DSP @400MHz, 32MB RAM	uCLinux	Compression	ROI encryption,
TrustCAM [5]: 10 fps @320×240	ARM Cortex A8 @480MHz, TMS320C64x+DSP@360MHz, Atmel TPM, 256MB RAM	Embedded Linux, Drivers, Libraries, Applications	ROI extraction, Compression	ROI encryption, Frame signing
TrustEYE [6]: 16 fps @640×480	ARM Cortex M4 @168MHz, 2×2MB SRAM, Atmel TPM	Embedded Linux	Compression	Frame encryption, Signing frame-group (group = 25 frames)
PUF-enabled-SoC based Secure Camera: 30 fps @640×480	Zynq7010 SoC @666MHz, 1GB DDR3 SDRAM	Embedded Linux Drivers, Libraries Application framework	Compression, Motion detection	Secure keys storage, Encrypt-HMAC-Sign footage, Secure boot, Tamper resistance

TABLE I
COMPARISON OF SOME EXISTING SMART CAMERA PLATFORMS WITH THE PROPOSED DEVICE (LAST ENTRY) WITH RESPECT TO HARDWARE PLATFORM, SOFTWARE STACK, SECURITY FEATURES AND ACHIEVED FRAME-RATE

non-repudiation, confidentiality, freshness and access authorization of the data before sending/archiving it over untrusted public infrastructure. Additionally, secure boot of the SoC verifies integrity, authenticity and unclonability of camera firmware. Hardware tampering can be detected due to tamper evidence property of the on-chip PUF. Compared with TPM, the PUF-based solution to secure camera nodes is cost effective, tightly integrated and does not require any additional specialized hardware.

- By leveraging the SoC-based secure visual sensor node, we propose a solution for secure capture, archival and delivery of video data for private space monitoring applications. The solution does not rely on secure cloud storage rather data protection is applied on camera before it leaves the sensing device.
- We present a Zynq7010 SoC-based camera prototype that is capable of securing 30 fps at 640×480 resolution. The proposed approach incurs only 371 bytes of storage, 2488 logic-gates of hardware and $\approx$160 bits/footage of transmission overhead on the node, which is insignificant compared to using trusted platform modules (TPMs) to secure visual sensor nodes.

II. RELATED WORK

The section presents related work in the areas of VSN platforms and different approaches to secure data and nodes. A visual sensor node hardware typically consist of an image sensor array and a host processing platform. The host generally runs an embedded operating system (OS); specific applications are deployed on top of the OS. An early VSN device, Cyclops [2], is based on 8-bit Atmega128 clocked at 7.3MHz and has 64kB RAM. It uses dedicated logic (CPLD) for capturing frames at CIF resolution. However, computer vision algorithms and standard asymmetric cryptography is typically not feasible on such resource constrained platforms. CITRIC [3] offers higher performance capacity by employing PXA270 processor clocked at 624MHz and 64MB RAM. As a result, it enables on-camera distributed pattern recognition, computer vision and security algorithms that reduces network communication overhead.

Mohanty [7] presents a secure digital camera concept with a built-in watermarking and encryption for digital rights management of the video data. Authenticity of images is achieved by watermarking. To meet the real-time computing requirements, FPGA implementation of invisible-robust watermarking and advanced encryption standard (AES) is proposed. PrivacyCam [4] is a privacy preserving camera for surveillance based on uCLinux on Blackfin DSP clocked at 400MHz with 32MB of RAM and Omnivision OV7660 image sensor that secures 4 fps at 320×240 resolution. Privacy is achieved by encrypting the regions of interest in each frame using AES. This allows to monitor behavior of people meanwhile obfuscating their identities. Only authorized people (having access to decryption key) have full access to reveal identities and behavior. Stifler et al. [8] presents monochrome CMOS active pixel image sensor with on-chip cryptographic engine that has digital encryption and authentication circuits to provide data integrity, confidentiality, and non-repudiation.

Approaches to secure VSN nodes mainly fall in two categories: secure camera approach and trustworthy sensing. A secure camera approach aims a holistic security solution addressing all layers of a camera stack including the applications, middle-ware, OS, and the hardware. In our previous work, we followed this approach in TrustCAM [5] where we integrated a trusted platform module (TPM) into camera node. Anonymous attestation and time-stamping features of the TPM were used to protect the integrity, authenticity and confidentiality of the image data. This approach has following limitations: (i) since security has to be meshed with the application logic, it is often left over to the application developers responsibilities and (ii) the large size of software components such as the OS and the network stack makes it difficult to effectively secure them.

TrustEYE [6] followed the trustworthy sensing approach which aims to protect the captured images closer to the sensor. Our TrustEYE node comprised Omnivision OV5642 image sensor and ARM Cortex M4 processor. We integrated TPM into the sensing unit, which has exclusive access to the sensor's data. Integrity, authenticity, confidentiality and freshness of the sensed data are ensured at the sensing unit using 2048-bits RSA keys. The platform is able to process and secure 640×480 frames at 16fps. TPM-based trusted image sensor approach incurs significant hardware overhead on the resource constrained sensing unit. Another major limitation of trustworthy sensing approach is that once data is signed within sensor, any legitimate modification (processing, compression etc.) of data at host processor stage invalidates the security guarantees.

We overcome these limitations by fingerprinting smart camera using lightweight security circuits called physically unclonable function (PUF). The fingerprint serves as entropy source for cryptographic keys. On-chip PUF provides secure storage for keys, binds the keys to camera hardware and makes the hardware tamper evident. Data protection is intrinsic feature of the node. Integrity, authenticity and freshness of data are ensured by digitally signing the timestamped video footage. Privacy and access authorization are ensured by end-to-end encryption of the frames. Since data security is implemented in camera firmware, integrity, authenticity and unclonability of camera firmware is ensured by secure boot. Unlike TPM, the PUF-based solution is lightweight, tightly integrated with the sensor's hardware fabric and can be trivially mapped onto the existing camera hardware. To the best of our knowledge, this is first attempt to secure VSN devices using PUFs. The proposed security mechanism is not limited to smart cameras but can be extended to other sensors and IoT nodes for privacy-protected, secure data delivery. A comparison summary of some existing platforms and the proposed device is given in Table I.

III. Private Space Monitoring

To illustrate our approach, we consider private space monitoring scenario of Fig. 1. Vulnerable OS on the camera nodes and use of public infrastructure for video data transmission and archival necessitates protection of this data closer to the source. Data protection includes data integrity, authenticity, confidentiality, freshness and access control. Since data security is implemented at application level, in order to ensure effective security guarantees, underlying software and hardware stack of the camera node needs to be protected as well. Node security requirements include integrity, authenticity and unclonability of camera firmware, resistance against hardware tampering and side-channel attacks. Below we provide an overview of our scheme for secure data capture, archival and delivery in the given private space monitoring scenario.

A. Overview

In order to keep the cost of the camera device low, we do not assume availability of permanent storage of videos on the camera. During monitoring, we leverage the public cloud storage for long-term video archiving. To limit the amount of data transmitted by the camera device, archiving is triggered upon event detection. The event can be triggered internally (e.g., by on board analytics) or externally (e.g., by auxiliary sensors or a request from the end user). We assume that integrity and authenticity of an external source is verified before triggering an event. Data security is an integral part of the sensing device. Integrity, authenticity, confidentiality and freshness of data is ensured on the sensing device before it is uploaded to the cloud storage. The security guarantees on the sensed data are ensured for entire lifetime of the data. Confidentiality is ensured by encrypting each video frame using AES128 encryption algorithm. Integrity and authenticity are ensured by signing the hash-chain of encrypted frames using our PUF-based Cert-IBS scheme [9]. PUF-based Cert-IBS and AES128 algorithms use platform-bound security keys. On-chip PUF binds the signing and encryption keys to VSN device hardware and serve as secure key storage. On event detection, encrypted-hashed-signed footage is uploaded to a public cloud storage server and an alert message is sent to the end-user who can then download the archived footage from the cloud on demand. Integrity, authenticity and freshness of data is ensured by verifying PUF-based Cert-IBS signatures. Only the authorized end-user (i.e., having access to the decryption key) can decrypt the frames.

In order to bind cryptographic keys with camera platform and limit data access to only legitimate end-user, the scheme requires two steps namely *fingerprint extraction* and *key exchange* to be performed before the camera can be deployed for monitoring. Fingerprint extraction is performed by a trusted authority (TA) in a secure and trusted environment during camera development stage. During this step, TA extracts unique fingerprint of the camera hardware using on-chip PUF. The fingerprint is used to bind the signing and encryption keys with the hardware. When the camera is powered off, these keys exist in form of CMOS manufacturing variations of the hardware which are hard to read. At power-up, the keys are extracted from the hardware using PUF. Key exchange is performed by the end user before deploying the camera for monitoring. During this step, the end-user transfers signature verification and decryption keys from the camera device to her monitoring device such as smartphone or tablet via a local interface e.g., NFC. Since the connection is local (without involving the Internet), it is assumed that the keys are not leaked to a third party. Afterwards, the camera is deployed for monitoring. The fingerprint extraction, key exchange and monitoring phases of our scheme are depicted in Fig. 2.

In remainder of this section, we briefly discuss security preliminaries in section III-B and detail the three phases of our scheme: fingerprint extraction in section III-C, key exchange in section III-D and monitoring in section III-E.

B. Security Preliminaries

Since the proposed scheme uses a PUF framework to bind cryptographic keys with camera platform and PUF-based Cert-IBS scheme to ensure non-repudiation of data, we introduce the framework and PUF-based Cert-IBS scheme first.

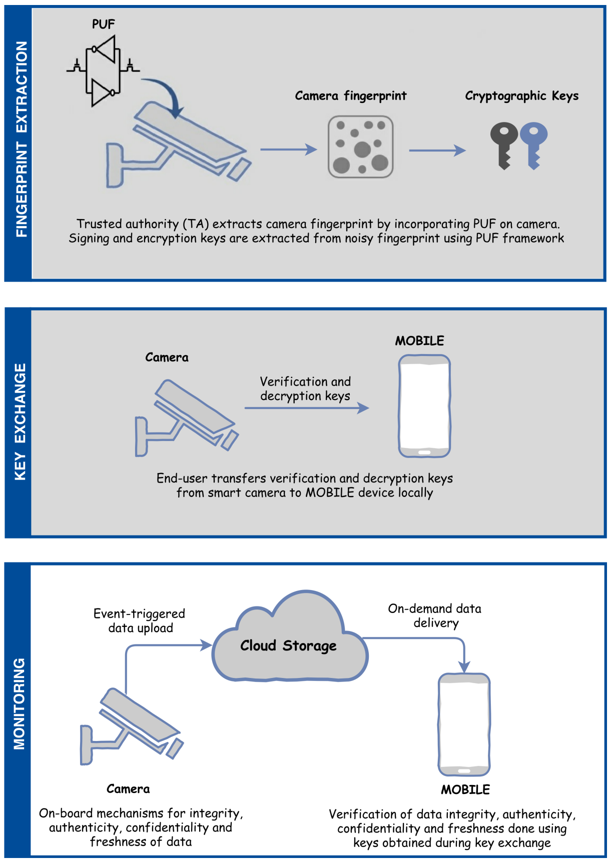
FINGERPRINT EXTRACTION
PUF
Camera fingerprint
Cryptographic Keys
Trusted authority (TA) extracts camera fingerprint by incorporating PUF on camera.
Signing and encryption keys are extracted from noisy fingerprint using PUF framework
KEY EXCHANGE
Camera
MOBILE
Verification and decryption keys
End-user transfers verification and decryption keys from smart camera to MOBILE device locally
MONITORING
Event-triggered data upload
Cloud Storage
On-demand data delivery
Camera
MOBILE
On-board mechanisms for integrity, authenticity, confidentiality and freshness of data
Verification of data integrity, authenticity, confidentiality and freshness done using keys obtained during key exchange

Fig. 2. The fingerprint extraction, key exchange and monitoring phases of proposed scheme for privacy-aware secure private space monitoring using PUF-enabled-SoC based camera node. Fingerprint extraction is performed once in a secure and trusted environment whereby the platform-bound unique cryptographic keys are created. During key exchange, the end-user shares verification keys with her monitoring device using a local interface. Thereafter, the camera is deployed for monitoring.

PUF Framework. Physically unclonable functions (PUF) are special lightweight circuits that use the CMOS manufacturing process variations to generate the fingerprint of the underlying hardware. A PUF circuit provides a challenge-response mapping that is based on the uncontrollable variations in the physical structure of the integrated circuit (IC) introduced during the manufacturing process. These variations are random and unique for each instance. Moreover, the chip manufacturer is not able to control or forge these variations (physical unclonability).

Multiple responses from the same PUF instance obtained under different environmental conditions (e.g., temperature) or operating conditions (e.g., voltage supply) slightly differ from one another. These variations are referred to as PUF noise or error-rate and are measured as intra-Hamming distance (HD^{intra}). Uniqueness of a PUF mapping is measured in terms of inter-Hamming distance (HD^{inter}) which is a measure of how different two responses from two PUF instances are. Randomness of a PUF response is measured in terms of Hamming weight (HW) of the response. Ideally, maximum $HD^{intra} \approx 0\%$, average $HD^{inter} \approx 50\%$ and average $HW \approx 50\%$. In order to extract uniformly distributed random and perfectly reproducible fingerprint from the noisy and biased PUF response, helper data algorithms (HDAs) are used.

Our scheme requires the flexibility of masking an externally generated cryptographic key with the device fingerprint; therefore we use the HDA due to Tuyls [10]. The PUF framework is comprised of two modules: the PUF and the HDA and works in two phases: key binding and key extraction.

1) Key Binding: $W \leftarrow Gen(r, k)$
It is a one-time protocol carried out by a legitimate authority on the PUF in a secure environment to generate helper data W. A challenge c is applied to the PUF and response r is obtained. The authority then chooses a random key $k \in \{0,1\}^k$ and calculates the corresponding helper-data as $W \leftarrow r \oplus C_{\mathrm{k}}$, where C_{k} is the nearest code-word chosen from the error-correcting code $\mathcal{C}$, with $2^k - 1$ code-words. W is integrity protected public information.
2) Key Extraction: $k \leftarrow Rep(r', W)$
It is performed every time the key extraction from the PUF is desired. The PUF is subjected to the same challenge c and a noisy response r' is obtained. The code-word is then calculated as $C_{\mathrm{k}'} \leftarrow r' \oplus W$. If r' corresponds to the same challenge c applied to the same PUF, k is obtained after decoding $C_{\mathrm{k}'}$ using W otherwise an invalid code-word is obtained i.e., $k \leftarrow \texttt{Decoding}(C_{\mathrm{k}'})$, if $\texttt{Hamming distance}(C_{\mathrm{k}}, C_{\mathrm{k}'}) \leq t$, where t is error-correction capacity of $\mathcal{C}$.

This PUF framework offers following key advantages: (i) it binds a unique key with a PUF-enabled hardware (ii) it provides secure storage of the key since the key is derived from device properties during start-up. When device is off, the key exists in form of unreadable CMOS manufacturing variations (iii) it offers more cost-effective secure key storage than a secure memory alternative.

PUF-based Cert-IBS. The PUF-based Cert-IBS [9] is based on the framework [11] to construct certificate-based identity-based signature (IBS) scheme from a standard signature (SS) scheme. A typical SS comprises three algorithms: key generation (K), signing ($Sign$) and verification (Ver). PUF-based Cert-IBS uses a key generation authority. To setup PUF-based Cert-IBS, the authority generates master key-pair (msk, mpk) using K. We denote a camera-platform with a unique identity I and a PUF by CAM(I, PUF). For CAM(I, PUF), the authority generates a unique signing key pair (sk_I, pk_I) using the key generation algorithm K of SS and binds sk_I with the on-chip PUF using the PUF framework i.e., $W_{sk_I} \leftarrow Gen(r_1, sk_I)$. Further, the authority issues a certificate on the corresponding public key given by $cert_I \leftarrow Sign_{msk}(pk_I, I)$. PUF-based Cert-IBS signature of CAM(I, PUF) on video data $frames[1\colon N]$ is given by $(\sigma, cert_I)$, where $\sigma \leftarrow Sign_{sk_I}(frames[1\colon N])$. PUF-based Cert-IBS verification is successful if $Ver_{pk_I}(frames[1\colon N], \sigma) = 1$ and $Ver_{mpk}(I, cert_I) = 1$. Successful Cert-IBS verification ensures that data $frames[1\colon N]$ is signed by CAM(I, PUF) with its platform-bound private key, assigned and bound to

CAM(I, PUF) by the legitimate authority. For detailed construction and security properties of PUF-based Cert-IBS, we refer the reader to [9][11].

C. Fingerprint Extraction

In our scheme, TA serves as the key generation authority. To setup our scheme for private space monitoring, TA generates a master key pair (msk, mpk). Further, it assigns each camera a unique identity I and instantiates a PUF. Serial number of the sensor or a unique bit-string written to on-chip one-time programmable (OTP) memory can be used as I. We denote camera with identity I and on-chip PUF instance PUF as CAM(I, PUF) and the trusted authority with master key pair as TA(msk, mpk). During fingerprint extraction, the authority TA(msk, mpk) binds a signing key pair (sk, pk) and an AES-encryption key (k_E) to the camera node CAM(I, PUF) using the camera fingerprint as follows:

1. CAM(I, PUF): Presents its identity I to the TA
2. TA(msk, mpk): Chooses two challenges (c_1, c_2) and feeds them to the PUF on CAM
3. CAM(I, PUF): Obtains the PUF responses to the challenges as $r_1 \leftarrow PUF(c_1)$ and $r_2 \leftarrow PUF(c_2)$ and returns (r_1, r_2) to the TA

4a. TA(msk, mpk): Generates a signing key-pair (sk, pk) using key generation algorithm of PUF-based Cert IBS. Using the key-binding algorithm of PUF-framework, it binds the private-half of the key-pair to PUF using r_1 i.e., $W_1 \leftarrow Gen(r_1, sk)$. Further, TA issues a certificate consisting of its signature on the CAM's identity and public half of the key-pair i.e., $cert \leftarrow Sign_{msk}(I, pk)$

4b. TA(msk, mpk): Picks a random key k_E and binds it to the PUF on CAM using r_2 i.e., $W_2 \leftarrow Gen(r_2, k_E)$

5. (W_1,W_2, $cert$) are stored in non-volatile memory (NVM) on camera

D. Key Exchange

Before an end user deploys the camera, she transfers identity I, certificate $cert$, verification key pk, and decryption key k_E to her monitoring device (smartphone, tablet etc.) using a local interface such as NFC. Since the transfer is done in a private space using a local connection, it is assumed that k_E is not transferred to a third party. Securing the keys on mobile devices with vulnerable software stack is out of scope of this work however, well established techniques such as virtualization [12] (isolates applications requiring trusted infrastructure) and secure vault can be leveraged for this purpose.

E. Monitoring

On camera power-up, the signing and encryption keys are generated from noisy PUF responses using key extraction phase of PUF-framework i.e., $sk \leftarrow Rep(r'_1, W_1)$ and $k_E \leftarrow Rep(r'_2, W_2)$. In case of an event, each video frame is encrypted using AES128 algorithm to ensure confidentiality. The non-repudiation on the data is ensured by using MAC-then-sign technique. First, each encrypted frame is hashed using HMAC algorithm to ensure integrity. Second, hash-chain from all encrypted frames in the footage is signed using PUF-based Cert-IBS scheme. Signing the entire hash-chain together preserves the frame order. A timestamp (τ) is also included in the signature to ensure freshness of data and thwart replay attacks. We denote the camera with identity I, on chip PUF instance PUF and helper data (W_1, W_2) as CAM(I, PUF, W_1, W_2); and the associated end-user monitoring device with camera's identity I, signature verification key pk, certificate $cert$ and decryption key k_E as MOBILE($I, pk, cert, k_E$). The steps during the monitoring phase of security scheme are listed as follows:

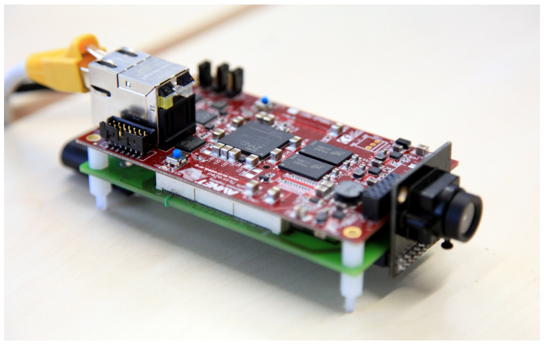

Fig. 3. Zynq7010 SoC and OV5642 image sensor based smart camera: It ensures video data integrity, authenticity, confidentiality and freshness using camera fingerprint extracted from hardware using on-chip PUF

1. CAM(I, PUF, W_1, W_2): Keys extraction from PUF
$$sk \leftarrow Rep(r'_1, W_1);\ k_E \leftarrow Rep(r'_2, W_2)$$
2. CAM(I, PUF, W_1, W_2): Encrypt each frame
$$C_i \leftarrow Enc_{k_E}(frame[i])|_{\ i\ =\ 1 \ldots N}$$
3. CAM(I, PUF, W_1, W_2): Hash the encrypted frames
$$h_i \leftarrow HMAC(C_i)|_{\ i\ =\ 1 \ldots N}$$
4. CAM(I, PUF, W_1, W_2): Timestamp and sign the footage
$$\tau = SHA256(I \parallel event\_count)$$
$$\sigma \leftarrow Sign_{sk}(h_i \,||\, h_{i-1} \,||\cdots||\, h_{i-N} \,||\, \tau)$$
5. CAM(I, PUF, W_1, W_2): Upload the encrypted-then-MACed-then-signed frames $\{C_i, \tau, \sigma\}_{i=i \ldots N}$ to the cloud storage
6. MOBILE($I, pk, cert, k_E$): Download the footage on-demand
7. MOBILE($I, pk, cert, k_E$): Verify the Cert-IBS signature
$$1 \overset{?}{=} Ver_{mpk}(cert, (I, pk))$$
$$1 \overset{?}{=} Ver_{pk}(\sigma, (h_i \,||\, h_{i-1} \,||\cdots||\, h_{i-N} \,||\, \tau))$$
8. MOBILE($I, pk, cert, k_E$): Decrypt the frames to get video data
$$frame[i] \leftarrow Dec_{k_E}(C_i)|_{\ i\ =\ 1 \ldots N}$$

IV. Prototype

The prototype (Fig. 3) is based on 5MP OV5642 image sensor module and MicroZed board, which houses Zynq7010 SoC clocked at 666MHz, 1 GB external RAM for frame buffering and a gigabit Ethernet interface to upload video

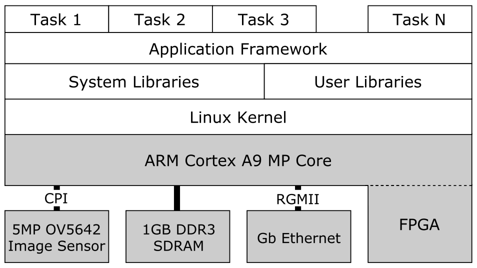

Fig. 4. Hardware (gray) and software (white) stack of Zynq SoC-based camera prototype. Hardware comprises of 5MP OV5642 image sensor, Zynq7010 SoC having FPGA and ARM Cortex A9 core. The software stack comprises embedded Linux kernel, system and user libraries and application framework

footage to cloud server. A custom board (green) was designed to interface the image sensor with the MicroZed board and regulate power to both the modules. Our Zynq SoC-based smart camera leverages on-chip PUF to extract camera hardware's fingerprint which serve as basis for on-camera data protection. Data security is rooted in the system hardware making it an intrinsic element of the device and therefore harder to bypass. Video data processing and protection is done inside the SoC and the data leaves the chip with integrity, authenticity, confidentiality and freshness guarantees. This makes the platform suitable for monitoring applications that use untrusted cloud storage services for short or long term data archival. The prototype is able to protect 30 fps at 640×480 resolution.

Hardware and software stack of the camera node are depicted in Fig. 4. The camera hardware consists of OV5642 5MP CMOS image sensor, Zynq7010 SoC, 1GB SDRAM and a gigabit Ethernet interface. The SoC comprises ARM Cortex A9 processor clocked at 666MHz and FPGA fabric. The processor runs embedded Linux that hosts system libraries (OpenSSL, GMP, libjpeg etc.) and user libraries (pbc, motion-detection etc.) to be used by the applications. The application framework (Fig. 5) is divided into four stages: sensing tasks, processing tasks, security tasks and communication tasks.

Sensing tasks entail reading the image sensor and format adaptation. OV5642 image sensor is configured to provide data in 640×480 (resolution) 8-bit YUV422 (color-space) format. Processing tasks subsumes video compression and event detection by behavioral analysis of video data. Video compression is achieved using JPEG compression engine on the OV5642 sensing unit. Motion is detected using three-frame differencing algorithm by Collins et al. [13], where image difference between frames at time t and $t-1$ and the difference between t and $t-2$, is performed to determine regions of legitimate motion and to erase ghosting. The event is triggered if motion is greater than a predefined threshold. Frames are conditionally forwarded to security tasks in case of event detection.

Security tasks implement data security as well as node security. Data security related tasks perform the following: After device power-up, encryption and signing keys are extracted from the ring oscillator (RO) PUF implemented using reprogrammable fabric and are loaded into cache. Each frame is encrypted using AES128 to ensure data confidentiality. The encrypted frames are MACed-then-signed to ensure integrity and authenticity of data. MAC checksum of each encrypted video frame is computed using HMAC-SHA256. Checksums from all frames in the video are concatenated and signed together using the PUF-based Cert-IBS scheme with BLS [14] as underlying standard signature scheme; this preserves frame order and protects against attacks aimed at manipulating order of the frames. Freshness of data is ensured by including a timestamp τ before signing the checksums. For timestamp generation, the camera uses an event counter that increments whenever an event is detected. Given that an event is detected by motion detection algorithm and the event counter is incremented to $event\_count$, then timestamp is calculated as $\tau =$ SHA256($I \parallel event\_count$). A time-stamp value holds true only for a specific event $event\_count$ detected by camera device I. Following the event $event\_count$, the footage is timstamped with τ. The value of τ should not repeat among footages of different events detected by the same camera or among footages from different cameras. This simple check deters replay attacks. It is important to note that encryption and hashing is performed on frames whereas time-stamping and signing is performed on the complete footage.

Node security is enforced by secure boot of SoC and on-chip PUF. Zynq7010 SoC provides secure boot functionality as part of its boot procedure that verifies authenticity and integrity of the camera firmware based on digital signatures, message authentication code (MAC), and encryption. Boot-chain of Zynq7010 SoC is depicted in Fig. 6. Secure boot foundation is established by the placing boot ROM code in masked ROM (one-time programmable memory). Each successive component of the boot-chain is signed using RSA key and MACed-then-encrypted using HMAC and AES256 algorithms. During every boot up, the chain of trust is established by the successive verification of signature (authentication), MAC (integrity) and decryption (confidentiality) of all software i.e., FSBL, bitstream, u-boot, OS, and apps. This prevents an adversary from tampering with software or the bitstream file. Zynq SoC contains hard IP cores for AES decryption and HMAC computation. The boot time of a secure Linux system is approximately the same as a non-secure system. Incorporating PUF into a chip makes the chip tamper evident [15]. Since PUF behavior depends on the underlying silicon fabric, any tampering with this fabric modifies the PUF behavior, thereby modifying the camera fingerprint. This leads to generation of incorrect signing and encryption keys thereby incorrect signature and cipher text, which is detected by the verifier.

Data with confidentiality, integrity, authenticity and freshness guarantees is then forwarded to Ethernet module for uploading. The prototype merely demonstrates proof of the concept and can be extended with wireless communication capabilities such as WiFi. The block diagram of the node showing core modules of video data path and their mapping on hardware components is depicted in Fig. 7.

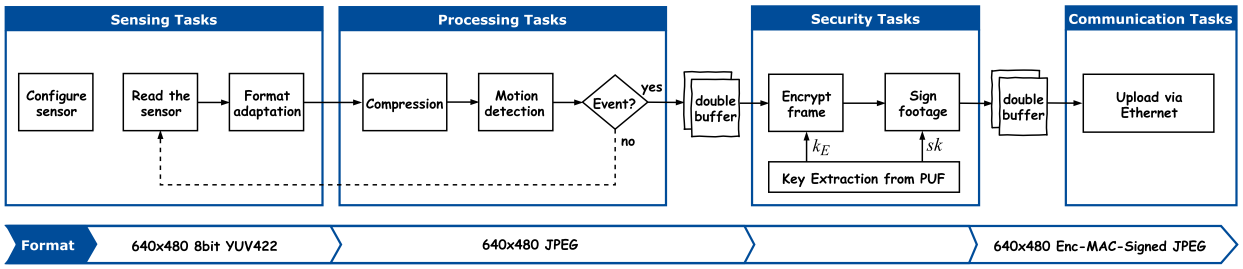

Fig. 5. Application framework of proposed camera comprises sensing tasks, processing tasks, security tasks and communication tasks. The sensing tasks read image data from the sensor gives it desired format. The processing tasks include the application logic (e.g., event trigger based on motion detection). Data is conditionally forwarded to security tasks in case of event detection, where frames are encrypted-MACed-signed. Protected frames are forwarded to communication tasks for upload

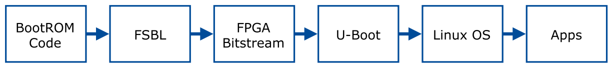

Fig. 6. Chain of trust for secure boot of Zynq7010 SoC

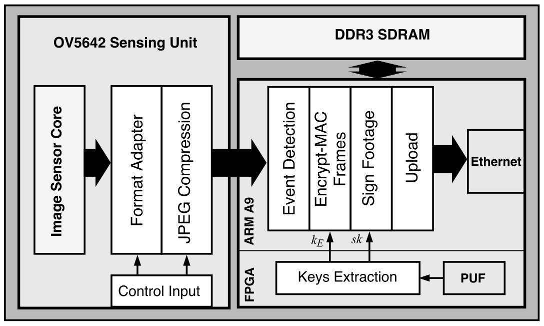

Fig. 7. Block diagram of SoC-based secure camera that depicts core components of camera hardware (gray) and software tasks (white) performed by these components

V. Evaluation

This section discourses on security properties of the proposed scheme and provides overhead incurred on camera node as well as end-user monitoring device due to the proposed scheme in terms latency, storage, hardware and transmission.

A. Security Properties

Data and node security properties of the proposes scheme are as follows:

- **Secure Key Storage.** Secure storage of the keys is ensured by on-chip PUF. On camera power up, the keys are generated from intrinsic variations of the hardware structure and are loaded into cache. When the camera device is off, the keys exist in form on unreadable variations introduced in the hardware by the CMOS manufacturing process. Compared to secure memory alternatives, PUF offers much cheaper secure storage.
- **Platform-bound Keys.** The PUF-framework binds signing and encryption keys to the camera platform. The signing key never leaves the platform thereby minimizing the risk of key compromise.
- **Data Non-repudiation.** All video data leaving the camera carry integrity and authenticity guarantees. Authenticity is ensured by signing the video data using platform bound signing key. PUF-based Cert-IBS with BLS [14] as underlying standard signature scheme is existentially unforgeable under chosen message attack (uf-cma secure), which is the standard security notion for a digital signature algorithm. Any modification or fabrication of data during delivery or archival can be detected at the end user monitoring device. Spoofing using offline images can be addressed by using multiple sources for event detection e.g., on-camera motion detection and sound detection.
- **Data Confidentiality.** Privacy of the monitored individual(s) is ensured by end-to-end encryption of each frame using AES128 algorithm.
- **Access Authorization.** Secure key exchange and key storage on end-user monitoring device ensure that only legitimate end-user can access the decrypted video data.
- **Data Protection Lifetime.** Data is protected close to the source (sensor) and the security guarantees on the data remain valid for entire lifetime of the data (i.e., during transmission, storage on cloud and delivery to monitoring device of end user). On the monitoring device, the data is consumed after successful verification of the security guarantees.
- **Camera Firmware Protection.** During camera development, using secure-boot feature of Zynq SoC, each component of boot-chain (BootROM Code, FSBL, FPGA Bitstream, U-Boot, OS and Apps) is hashed (HMAC), signed (RSA) and encrypted (AES256). During monitoring, at every boot-up, for every component of boot-chain the signatures are verified, decryption is performed and HMAC checksum is verified. This ensures integrity, authenticity and unclonability of camera firmware.
- **Physical Security.** First, On-chip PUF offers resistance against hardware tampering of camera hardware. Since the PUF extracts device fingerprint as a function of intrinsic details of the hardware, any hardware tampering is detected as it results in incorrect device fingerprint and keys. Second, in a TPM based solution, data is transferred from host processor to TPM chip (external to the host), where data protection mechanisms are applied.

This results in exposed interface with unprotected data which can be tapped to bypass security mechanism. With incorporation of PUF in host SoC, these exposed interfaces are eliminated resulting in better physical security.

Next, we enlist some limitations of our scheme and identify some additional countermeasures that may be appended with our scheme to overcome these limitations:

- **Side Channel Attacks.** First, although security keys are generated securely inside the SoC, keys generation on every power-up opens up electromagnetic and power side channels. Analysis of the side channel information be result in partial recovery of keys at the hands of attackers. Approved effective techniques, masking (e.g, reversible process in which intermediate values of variables are randomized by masked with random numbers) and hiding (e.g., use of dual rail logic to flatten the data dependent leakage) can be leveraged to thwart side-channel attacks. Second, data upload is triggered upon every event detection, the transmission pattern of the camera opens up another side-channel that leaks e.g., whether or not someone is at home. Transmitting dummy data at random intervals is a simple countermeasure that can be added to the system to mitigate this threat.
- **Denial of Service.** Denial of service attacks by (i) the cloud, such as deleting the archived data, blocking the downloads or (ii) a third party, such as corrupting the video or control data in transit or storage are not addressed by this scheme.
- **Monitoring Device Security.** The scheme uses symmetric key encryption (i.e., same key for encryption and decryption) since it is orders of magnitude faster and less power hungry than an asymmetric key encryption. However, symmetric key encryption requires secure key exchange between the camera and monitoring device and secure storage of key in the monitoring device. Key exchange is done by the end-user in a private space using a local interface, so the risk of key is relatively low. On monitoring device with untrusted software stack, virtualization, secure key vault or PUF can be used to securely store the key.

B. Camera Node Overhead

This section computes overhead on camera node in two phases: First, we discuss implementation of PUF framework for secure storage and generation of signing and encryption keys and compute overhead incurred by the framework on camera node wrt. latency, storage and hardware. Second, we evaluate the total overhead on the camera node due to sensing, processing, security and communication tasks.

PUF Framework. We implemented 1040 3-stage ring oscillators in FPGA fabric of the Zynq7010 SoC and evaluated the PUF responses for noise, randomness, and uniqueness. A total of 800 responses were obtained over a temperature range of $0-60°$C and $\mathrm{HD}^{\mathrm{intra}}$ (measure of noise/error-rate) and HW (measure of randomness) were calculated. The same PUF was instantiated on 9 different Zynq7010 SoC based Microzed boards and $\mathrm{HD}^{\mathrm{inter}}$ (measure of uniqueness) was computed. The results of PUF error-rate and randomness are depicted in Fig. 8. Average $\mathrm{HD}^{\mathrm{inter}}$ of PUF responses obtained the same PUF replicated on 9 Zynq devices was computed as 49.1%.

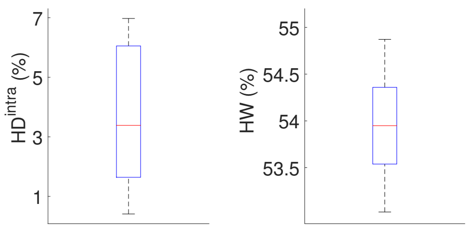

Fig. 8. Quality parameters of RO PUF implemented on Zynq7010 SoC. The PUF is comprised of 1040 3-stage ROs. Parameters are computed from 800 responses obtained over temperature range of $0-60°$C. The mean and max. error-rate ($\mathrm{HD}^{\mathrm{intra}}$) is $\approx 3.6\%$ and 6.97%. The randomness of PUFs ($\mathrm{HW(mean)}$) $\approx 53.95\%$.

The framework uses error correcting codes to correct PUF error-rate/noise which results in perfectly reproducible keys every time PUF-based key extraction is performed. Since the maximum recorded error-rate amounts of 6.97%, we employed two codes capable of correcting 10% error-rate: (i) a simple code i.e., BCH(492,57,171) and (ii) a concatenated code i.e., RM(16,5,8)||Rep(5,1,5). The node uses 160 bit signing key (sk) for BLS signature scheme and 128 bit encryption key (k_E) for AES128 encryption algorithm. Implementation results for PUF framework for sk and k_E generation are summarized in Table II.

PUF Source	**Key**	**Error Correcting Code**	**Hardware** (≈ Logic Gates)	**Latency** (Key Extraction)	**Storage** (Helper Data W)
RO	k_E (128-bit)	BCH	1106	≤ 100 ms	**1105** bits
		RM || Rep	2049		2048 bits
	sk (160-bit)	BCH	1382		**1381** bits
		RM || Rep	2561		2560 bits

TABLE II
OVERHEAD DUE TO PUF FRAMEWORK FOR 128 BIT k_E AND 160 BIT sk

Latency Overhead. Since keys extraction is performed only once at power up whereas timestamping and signing are performed once per footage, they do not incur latency during runtime. At a resolution of 640×480, latency of camera's computer vision pipeline is 27ms (i.e., 37 fps). However at the given resolution, the image sensor can only provide 30 fps. Therefore the prototype is able to secure 30 fps at 640×480 resolution. The running times for individual tasks of the camera application framework are given in Table III.

Storage Overhead. Breakdown of memory overhead due to components of the proposed security mechanism is given by second and third rows of Table IV. First row shows size of a double frame buffer used by event detection and Ethernet tasks. In comparison to memory consumed by frame buffer

Module	Runtime
PUF-based Keys Extraction (once at power-up)	<100 ms
Event Detection	21.1 ms
Frame Encyption (AES128)†	3.4 ms
HMAC-SHA256†	2.5 ms
Footage Signing (BLS)	6.27 ms

† values are averaged over 1000 frames

TABLE III
RUNTIMES FOR INDIVIDUAL MODULES OF APPLICATION FRAMEWORK FOR THE ZYNQ7010 SOC-BASED SECURE CAMERA

(used by application logic), total storage overhead on the camera due to proposed security mechanism is insignificant.

Module	Memory
Double frame buffer	$2{\times}600$kB
PUF-based Keys Extraction	311B (helper data)
Footage Signing	60B $(cert, pk)$

TABLE IV
MEMORY CONSUMPTION OF INDIVIDUAL MODULES OF APPLICATION FRAMEWORK FOR THE ZYNQ7010 SOC-BASED SECURE CAMERA

Hardware Overhead. Hardware overhead is incurred on camera node only if ring oscillators (RO) PUF is implemented in FPGA part of SoC. This can be avoided by exploiting either uninitialized SRAM or fixed pattern noise of image sensor, which are inherently present in a typical camera platform. Since SRAM on Zynq7010 gets initialized during SoC boot up, it cannot be used as PUF. We implemented RO PUF to generate the signing and encryption keys. The total hardware overhead (Table II) for generating 160 bit sk and 128 bit k_E sum up to 2488 logic gates which is negligible as compared to a TPM chip.

Communication Overhead. Given a private space monitoring scenario, a camera with proposed security mechanism uploads encrypted-MACed-timestamped-signed footage i.e., $\{C_i, \tau, \sigma\}_{i=i..N}$ to the cloud storage. The encrypted frame C_i is same size as original image. A 256-bit timestamp τ and a 160-bit signature σ are added to the footage for uploading. For a footage comprised of N frames, total communication overhead incurred on the camera device due to the proposed scheme amounts only to 416 bits, which amounts to 0.008% of a single frame's size.

C. Monitoring Device Overhead

We carried out integrity, authenticity, confidentiality and freshness verification on video data at monitoring device side (see section III-E). We implemented the BLS signature verification, AES decryption and HMAC-SHA256 algorithms on ARM Cortex A9 processor running at 666MHz that is compatible with today's smartphone processor. The results of verification are summarized in Table V.

Module	Runtime
Frame Decryption	3.52 ms
HMAC-SHA256	2.5 ms
BLS Signature Verification	638.2 ms

TABLE V
RUNTIME OF VERIFICATION TASKS ON MOBILE DEVICE

VI. CONCLUSION

We presented SoC-based camera platform for secure monitoring applications where data and node security, rooted in the hardware, are intrinsic features of the device. We demonstrated capabilities of our Zynq7010 SoC-based camera with a private space monitoring application. Platform bound unique keys and local key exchange between cameras and monitoring devices provide a scalable solution for secure data delivery without trusting the cloud service provider. Ongoing work includes exploring other nuances of privacy between end-to-end encryption and full access. Image cartooning [6] is one such intermediate privacy protection approach. Further performance evaluation of the prototype for higher frame resolutions is also under investigation.

ACKNOWLEDGMENT

This research has been supported by the Austrian Research Promotion Agency under grant number 842432 and through the research initiative ”Intelligent Vision Austria”.

REFERENCES

[1] T. Winkler and B. Rinner, “Security and privacy protection in visual sensor networks: A survey,” *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 2, 2014.
[2] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, “Cyclops: in situ image sensing and interpretation in wireless sensor networks,” in *ACM Proc. SenSys 2005.*
[3] P. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. Lobaton, M. Meingast, S. Oh, S. Wang, P. Yan *et al.*, “Citric: A low-bandwidth wireless camera network platform,” in *ACM/IEEE Proc. ICDSC 2008.*
[4] A. Chattopadhyay and T. E. Boult, “Privacycam: a privacy preserving camera using uclinux on the blackfin dsp,” in *IEEE Proc. CVPR 2007.*
[5] T. Winkler and B. Rinner, “TrustCAM: Security and privacy-protection for an embedded smart camera based on trusted computing,” in *IEEE Proc. AVSS 2010.*
[6] T. Winkler, A. Erdélyi, and B. Rinner, “TrustEYE. M4: Protecting the sensorNot the camera,” in *IEEE Proc. AVSS 2014.*
[7] S. P. Mohanty, “A secure digital camera architecture for integrated real-time digital rights management,” *Journal of Systems Architecture*, vol. 55, no. 10, pp. 468–480, 2009.
[8] P. Stifter, K. Eberhardt, A. Erni, and K. Hofmann, “Image sensor for security applications with on-chip data authentication,” in *SPIE Proc. Defense and Security Symposium 2006.*
[9] I. Haider and B. Rinner, “Securing Cloud-based IoT Applications with Trustworthy Sensing,” in *EAI Proc. CN4IoT 2017.*
[10] P. Tuyls and L. Batina, “RFID-tags for anti-counterfeiting,” in *Springer Proc. RSA 2006.*
[11] M. Bellare, C. Namprempre, and G. Neven, “Security proofs for identity-based identification and signature schemes,” *Journal of Cryptology*, vol. 22, no. 1, pp. 1–61, 2009.
[12] X. Chen, T. Garfinkel, E. C. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, J. Dwoskin, and D. R. Ports, “Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems,” in *ACM Proc. SIGARCH 2008.*
[13] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt *et al.*, “A system for video surveillance and monitoring,” 2000.
[14] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” in *Springer Proc. TACIS 2001.*
[15] R. Maes, “Physically unclonable functions: Constructions, properties and applications,” Ph.D. dissertation, University of KU Leuven, 2012.