Secure Embedded Visual Sensing in End-User Applications with TrustEYE.M4

Thomas Winkler
Institute of Networked and Embedded Systems
and Lakeside Labs
Alpen-Adria-Universität Klagenfurt
Lakeside Park B02b, 9020 Klagenfurt
Email: thomas.winkler@aau.at

Bernhard Rinner
Institute of Networked and Embedded Systems
and Lakeside Labs
Alpen-Adria-Universität Klagenfurt
Lakeside Park B02b, 9020 Klagenfurt
Email: bernhard.rinner@aau.at

***Abstract*—Sensor networks are becoming popular for applications in private environments such as home monitoring, baby and child monitoring or ambient assisted living. What is often overlooked is the fact that data collected in this context can be very privacy sensitive. This is especially true if images or videos are captured and streamed into the cloud. This work presents a visual sensor network device along with a secure data delivery and archiving solution using untrusted public cloud storage services. This is achieved by extending the sensing device with a local communication interface for establishing a secure link between data producer and consumer. Our custom-designed TrustEYE.M4 prototype implements secure data delivery and uses Near Field Communication (NFC) for the initialization of the communication link. The presented solution is not limited to visual sensing applications but can be applied to a wide range of pervasive sensing and data delivery scenarios.**

I. Introduction

Traditional applications of smart cameras [1] include, e.g., surveillance [2] or traffic monitoring [3], [4]. With the emerging domain of consumer-centric and pervasive IoT applications, visual sensor networks and intelligent cameras are deployed in smart homes [5], for baby and child monitoring or in assisted living applications [6], [7]. In existing cloud-based video recording solutions such as dropcam [8], data security is considered for the communication channel (data in transit) but not while data is stored (data at rest). This means that the operator of the cloud-based online storage has to be trusted not to manipulate or disclose sensitive data. In this work we present a wireless, embedded visual sensing device called TrustEYE.M4 which is ideally suited for monitoring applications in today's Internet of things. A primary design aspect for TrustEYE.M4 was to make data security and privacy protection integral features of the sensing device itself. The capabilities of the system are demonstrated in an event-triggered, cloud-based video delivery application where a trust relationship between the camera and the client device is established via local, Near Field Communication (NFC).

The contribution of this work is threefold. (1) We present the hardware and software architecture of the custom-designed TrustEYE.M4 which is a versatile state-of-the-art research platform for security-centric monitoring and visual sensor network applications. (2) We discuss a practical solution for securely using cloud storage for delivery and archiving of potentially sensitive video data. This approach does not imply security of the cloud storage, but data protection is applied before any data leaves the embedded camera device. NFC is used to securely establish the link to data recipients. (3) We demonstrate the system's capabilities and evaluated its performance with a real-time event-monitoring and video delivery application, which provides hardware-based data encryption and non-repudiation guarantees based of the on-board TPM security chip.

The remainder of this paper is organized as follows. Section II discusses related work on embedded smart camera systems and gives an overview of camera- and sensor-centric security approaches. Thereafter, Section III discusses our solution for secure cloud-based data delivery for IoT applications. Section IV presents the hardware and software architecture of the TrustEYE.M4 sensing platform. In Section V we discuss the implementation of the secure, cloud-based data delivery case study and present evaluation results. Finally, Section VI summarizes the paper and gives an outlook to future work.

II. Related Work

In this section we present related work in the areas of visual sensor network (VSN) platforms as well as approaches for camera and sensor-level security. A more complete review of this topic is provided in [9].

An early VSN device example with limited resources is Cyclops by Rahimi et al. [10], [11]. It is equipped with an ATmega128 8-bit RISC microcontroller clocked at 7.3 MHz. It provides 4 kB of on-chip SRAM and 60 kB of external RAM. The CMOS sensor delivers RGB images at CIF resolution. For image capturing a CPLD between the sensor and the microcontroller is used. Cyclops does not have on-board networking facilities but it can be attached to a MicaZ mote. CmuCAM 4 [12] is the latest version of a camera designed for robotics applications. It performs on-board image processing and analysis (e.g., color tracking, mean, and median computation, segmentation) at 160×120 pixels. It is powered by a P8X32A (Propeller) CPU and uses an OV9655 sensor. The WiCa wireless camera [13] is based on a single instruction, multiple data (SIMD) processor called Xetal-II [14] clocked at 80 MHz. The processor features 320 RISC processing units which perform line-based, parallel image processing. For general purpose tasks an 8051 microcontroller is used and an 802.15.4 radio serves for inter-node communication. The CITRIC platform by Chen et al. [15] is equipped with a PXA270

CPU clocked at 624 MHz and 64 MB of RAM. Consequently, it facilitates the implementation of more complex computer vision algorithms. Mohanty [16] describes a secure digital camera system that is able to provide integrity, authenticity and ownership guarantees for digital video content. This is achieved using a combination of watermarking and encryption techniques. Due to the high computational effort, a custom hardware prototype based on an FPGA is used to meet the real-time requirements. PrivacyCam [17] is a camera system based on a Blackfin DSP clocked at 400 MHz, 32 MB of SDRAM and an Omnivision OV7660 color CMOS sensor. Regions of interest are identified based on a background subtraction model and resulting regions are encrypted using an AES. Mohanty and Adamo [16], [18] follow this approach and describe a secure digital camera system that provides integrity, authenticity, and ownership guarantees via watermarking and encryption. A binary watermark is encrypted with a user-supplied key before it is embedded into the image. An FPGA-based prototype demonstrates the feasibility of the approach under real-time conditions. De Strycker et al. [19] use a TriMedia digital signal processor to embed an invisible, digital watermark into video frames in real-time. The watermark consists of a pseudo-noise pattern that depends on a secret key. The system is evaluated in the context of a video broadcasting application where it provides authenticity guarantees for delivered video streams. Stifter et al. [20] suggest to integrate a secure storage for a symmetric, cryptographic key into the image sensor. This key is used in an on-chip crypto unit as part of message authentication code (MAC) computations. The system provides integrity and authenticity guarantees for delivered data.

III. Secure Sensing Concept

This section first presents our secure sensing solution which relies on public cloud storage. The second part of the section summarizes the achieved security properties and discusses underlying assumptions and limitations.

To illustrate our secure sensing concept, we have chosen a specific scenario where an embedded camera is used for private home monitoring. However, the presented approach is not limited to visual sensors but can be applied to any sensing application. To keep the cost for the camera device as low as possible we do not assume the availability of permanent storage for videos on the camera itself. As shown in Figure 1, we rely on existing cloud-based storage solutions for long-term data archiving and retrieval via mobile, Internet-enabled user devices. To limit the amount of transmitted data, we perform on-board event detection (e.g., motion detection, behavior analysis, ...) which triggers the upload of video footage to the cloud. What sets our approach apart from existing solutions is that data security is an integral part of the sensing device. For all data that is uploaded to the cloud we ensure non-repudiation (integrity, authenticity, and timestamping) and confidentiality for the entire data lifetime and not only for data in transit. Data can be accessed only via authorized user devices (e.g., tablets, mobile phones). A critical aspect is the establishment of the link between the camera and the user's mobile device. In the initialization phase shown in Figure 1 we assume that the user installs the camera in the desired location. Having physical access to the camera allows the user to use the NFC interface of the camera to securely establish a link between the camera and the mobile device. Since this process is entirely local without involving any Internet communication, it is ensured that cryptographic keys used to provide data confidentiality are not disclosed to third parties.

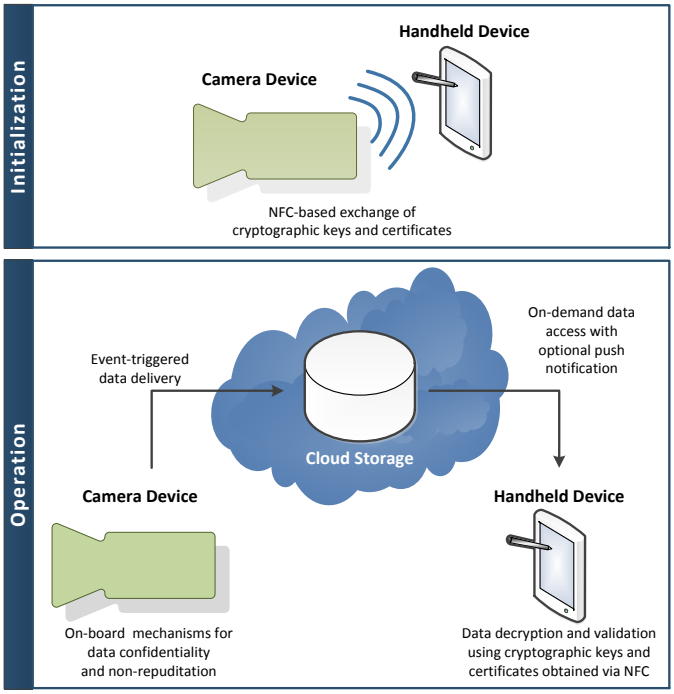

Fig. 1. In the initialization phase, local NFC-based communication is used to deliver required cryptographic keys and certificates to the user's mobile device. During operation, the camera uploads digitally signed and encrypted data to a cloud service from where they can be retrieved by users. Data upload is event-triggered (e.g., motion detection).

Our TurstEYE.M4 prototype (cp. Section IV) is equipped with a Trusted Platform Module (TPM) security chip. It is used for providing non-repudiation guarantees for images, videos and event descriptors based on TPM-protected, non-migratable 2048bit RSA keys. In the initialization phase of the camera, the TPM's owner password has to be set. The TPM's public endorsement key (EK) certificate is made available to the user via NFC. After validation of the EK certificate the user chooses an owner password, encrypts it with the public EK and supplies it to the camera via NFC. The camera executes the TPM's TakeOwnership function and initiates the creation of an attestation identity key (AIK) with the user's mobile device acting as local PrivacyCA. With this AIK the camera certifies the signing key κ_{SIG} and the mobile device is able to validate this certificate. Finally, the WiFi credentials are set via NFC and the camera can now connect to the Internet.

Once the camera is operational it initiates the event-triggered data upload. Before upload, AES256 encryption of video frames is performed using the hardware accelerators of the STM32F417 CPU. A shared AES256 key, created by the camera and exchanged via NFC during initialization, is used. Singing and timestamping are done with the TPM's *TickStampBlob* function. The individual steps are as follows:

1) **CAM:** Event detected (e.g., via motion threshold)
2) **CAM:** Encrypt captured frames:

$$encFrm[i] = ENC_{\kappa_{AES}}(frm[i]).$$

3) **CAM:** Compute hash-chains for frame groups which are signed and timestamped with a non-migratable TPM key:
$$sigFrmGrp = TICKSTAMP_{\kappa_{SIG}}(H(frm[n:m])||H(encFrm[n:m])).$$
4) **CAM $\longrightarrow$ CLOUD:** Upload of encrypted and digitally signed frames
5) **MOBILE $\longleftarrow$ CLOUD:** On-demand delivery of encrypted and signed frames
6) **MOBILE:** Validate κ_{SIG} using κ_{AIK}
7) **MOBILE:** If data verification is successful, decrypt and show the frames

Note that we only provided a high-level description of the used TPM security functions. An in-depth description of such an approach including details on TPM setup, signing and timestamping of frame groups as well as extended functionality such as trusted boot is given in [21]. Subsequently we discuss the achieved security properties together with the involved limitations and assumptions.

A. Security Properties

- **Data non-repudiation.** All image, video and event data delivered by the camera via the cloud comes with non-repudiation guarantees provided via hardware-protected (TPM) security features.
- **Data confidentiality.** All image, video and event information is encrypted on-camera, thereby ensuring confidentiality and privacy protection. Advanced versions of the system could support different protection levels as discussed in [22].
- **Protection lifetime.** Non-repudiation and confidentiality guarantees hold for the entire data lifetime starting from the point the data is created on the camera. Therefore, protection includes data in transit as well as data at rest. Protecting all data before it leaves the camera means that no trust assumption must be made for the cloud service provider w.r.t. data confidentiality and data integrity.
- **Secure link establishment.** A trusted, local connection (NFC) between the camera and the user's mobile device is used to exchange cryptographic keys and establish the link between the communication partners. Other parties such as the cloud service provider do not get access to the locally exchanged information.

B. Assumptions

- **Network connectivity.** We assume that Internet connectivity for the camera is available via, e.g., WiFi.
- **Cloud storage access.** We assume that access to a cloud storage facility exists and sufficient space is available for the upload of images and videos. Ideally, the cloud service supports push notifications if new content is added. Note that we do not impose special security requirements on the cloud service.
- **Limited physical access.** Access to content produced by the camera is granted to any user who is able to establish a link via NFC by having physical access to the camera. For a consumer-class monitoring application (e.g., a camera installed in a private house with limited physical access) this is sufficient. If required, the authorization process can be extended by a second factor (e.g., a PIN or password).

C. Limitations and Threats

- **Denial of service.** The cloud service provider does not get access to plaintext data and potential modifications are detected on the user's mobile device. However, we do not provide countermeasures against denial of service attacks. A cloud service can disrupt the system's functionality by, e.g., deleting uploaded data or blocking downloads.
- **Mobile device security.** The mobile device used for data access is assumed to be trustworthy. Protecting the mobile device is not within the scope of this work. It must be noted that a malicious mobile device could break the system's security by disclosing sensitive information received from the camera via NFC.
- **Camera firmware attacks.** In the current system the camera does not expose any services (i.e., open network ports) via WiFi which reduces the risk for remote exploits. Nevertheless, extended protection measures such as secure boot and Flash memory write protection should be considered to protect the camera's firmware.
- **Side channel attacks.** Since uploads to the cloud service are event triggered the communication pattern of the camera is a potential side channel that leaks, e.g., whether or not somebody is at home. Adding randomized transmission of dummy data is a simple measure to mitigate this problem.
- **NFC communication distance.** The locality of NFC communication is important for the system's security. An attacker using a sophisticated antenna configuration might be a security risk if an attacker is able to get close enough to the camera. Assuming that the camera is installed in a physically protected environment (e.g., a residential house) this risk is relatively low.

IV. Prototype System Architecture

This section presents the hardware and software architecture of the TrustEYE.M4 sensing platform.

The TrustEYE.M4 CPU board (Figure 2) is a custom-designed, 50×50 mm, two layer printed circuit board equipped with an ARM Cortex M4 STM32F417 microcontroller with 192 kB on-chip SRAM and 1 MB on-chip Flash memory. Since the on-chip SRAM is insufficient to store multiple images and intermediate results of CV algorithms, 4 MB of external SRAM are added. Data transfers from the image sensor module to SRAM and from SRAM to other peripherals are implemented via the microcontroller's DMA engines such that the CPU itself is available for image analysis tasks. Figure 3 presents an overview of the core components of the processing board. The system is powered either via Micro-USB or a single-cell lithium polymer battery which is charged automatically if external power is available. The system can be programmed via the Micro-USB port. Figure 2 illustrates the modular design of the TrustEYE.M4 CPU board. The image sensor (Omnivision OV5642) is connected via a dedicated port and can be easily

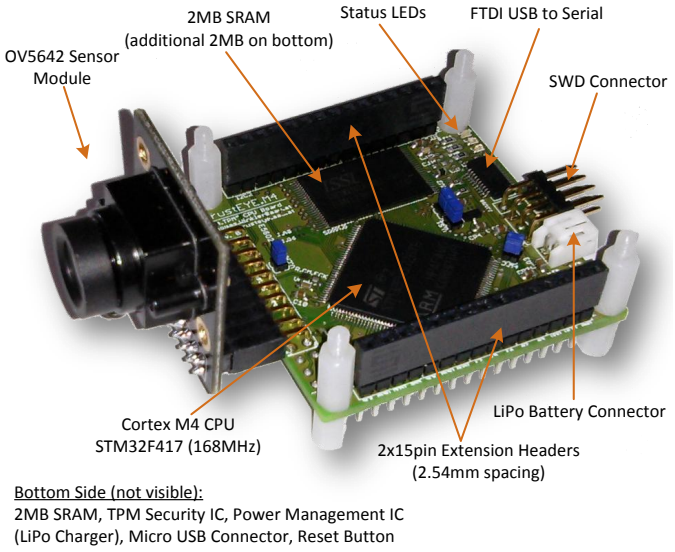

Fig. 2. The 50×50 mm TrustEYE.M4 CPU board with an OmniVision OV5642 image sensor module.

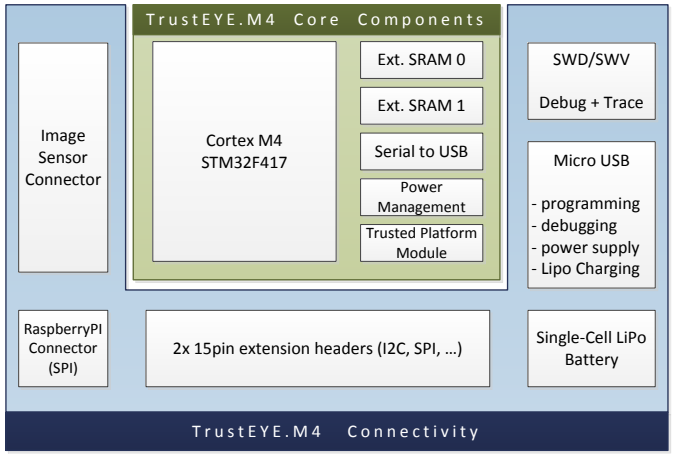

Fig. 3. The core components of the TrustEYE.M4 CPU board consist of a Cortex M4 CPU (STM32F417) clocked at 168 MHz, 4 MB of SRAM, an ST33TPM12SPI security IC, a BQ24074 power management IC and a FT230xs USB to serial converter. Several connectivity options are implemented via breakout headers.

exchanged. Two 15 pin headers provide access to on-board buses such as I2C and SPI as well as to GPIO pins of the microcontroller. Wireless networking is implemented via an extension board with a Redpine Signals RS9110-N-11-24-02 WiFi 802.11b/g/n radio which can be seen in Figure 5. NFC is added via an M24LR dual interface EEPROM chip which can be written and read by the camera's microcontroller over I2C or via an integrated ISO 15693 compliant RF interface.

For enhanced system security, TrustEYE.M4 provides hardware accelerators for cryptographic algorithms including AES256, SHA1, SHA256 and HMAC. Furthermore the SoC provides a true random number generator and a 96-bit unique ID. The on-board ST33TPM12SPI TPM chip provides RSA key generation (2048 bits), RSA signature creation and encryption, secure monotonic counters, remote attestation capabilities and comes with an endorsement key certificate.

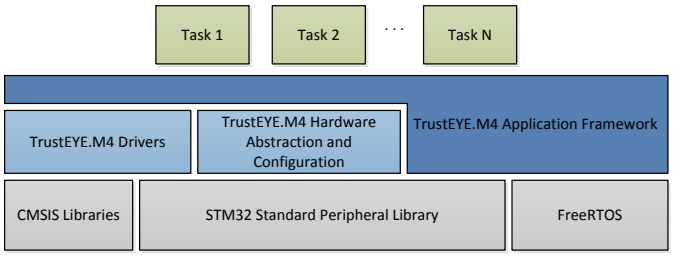

Fig. 4. The TrustEYE.M4 software architecture including drivers, a hardware abstraction and configuration layer and a framework for application tasks.

Figure 4 presents an overview of the TrustEYE.M4 software architecture consisting of (1) the libraries and drivers, the FreeRTOS[1] real-time operating system and the TrustEYE.M4 hardware abstraction and configuration component. The TrustEYE.M4 application framework provides multi-tasking support via synchronized double-buffered queues such that bulk data transfers via DMA do not block other tasks.

V. Case Study and Evaluation

Figure 6 shows the prototype consisting of the TrustEYE.M4 CPU board, the WiFi extensions board on top of it and the M24LR-discovery NFC board. An NFC-enabled Nexus 4 smartphone is used as mobile device. In initialization phase the camera makes the TPM's endorsement key certificate available via the M24LR EEPROM which is read by the smartphone app via NFC. The certificate enables the smartphone app to supply an encrypted TPM owner password and subsequently act as local PrivacyCA for AIK creation. NFC is used also for transmission of the AES256 encryption key which is stored in the smartphone's password-protected keystore.

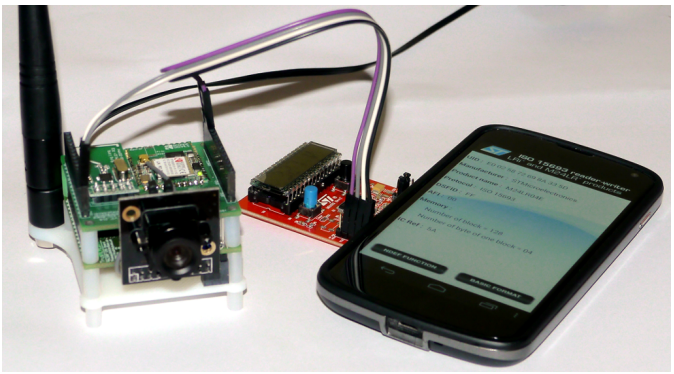

Fig. 5. TrustEYE.M4 prototype with the WiFi extensions board and a M24LR-discovery board (red PCB) with dual interface EEPROM attached via I2C. A Nexus 4 smartphone with integrated NFC reader is used as mobile device.

The secure cloud-based video delivery application from Section III is broken down into four tasks running on the TrustEYE.M4 software framework. The sensor task initially configures the OV5642 image sensor. Thereafter, pairs of JPEG (640×480) and YUV422 (320×240) frames are read from the sensor via DMA. The next task is the application's main processing task which first parses JPEG and YUV422 data which is delivered by the sensor in interleaved format. The

[1]FreeRTOS website: http://www.freertos.org/ (last visited: December 2014)

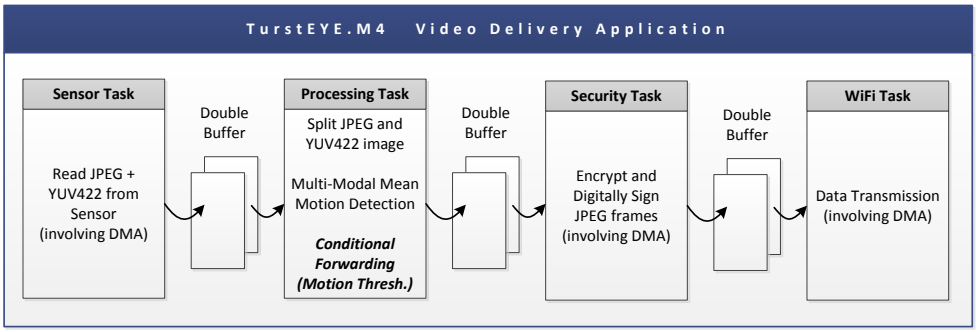

Fig. 6. TrustEYE.M4 video delivery application. JPEG (640×480) and YUV422 (320×240) frame pairs are read from the sensor via DMA. Foreground is determine by multimodal mean and object bounding boxes are computed. Based on the amount of detected motion, the JPEG images are encrypted and authenticity, integrity and freshness are ensured via the TPM's TickStampBlob operation. Finally, the results are sent to the cloud storage.

	Runtime
JPEG/YUV422 parsing	9.2 ms
Multimodal Mean (4 Cells)	41.8 ms
Bounding Box Calculation	4.8 ms
SHA1 Computation	1.2 ms
AES256 Encryption	1.7 ms
Total	58.7 ms

TABLE I. RUNTIMES (AVG. OVER 100 FRAMES) FOR THE INDIVIDUAL STEPS OF THE PROCESSING PIPELINE SHOWN IN FIGURE 6. AES256 AND SHA1 ARE FOR AN AVERAGE DATA SIZE OF 40 KB.

	Memory Usage
Sensor Output (Dbl. Buffer)	2×225 kB
Thumbnail	75 kB
JPEG (Dbl. Buffer)	2× 75 kB
Background	10 kB
Multimodal Mean (4 Cells)	4×225 kB
Heap	30 kB
Total	1615 kB

TABLE II. MEMORY CONSUMPTION FOR THE INDIVIDUAL COMPONENTS OF THE PROCESSING PIPELINE SHOWN IN FIGURE 6.

	Input Current @ 3.3V
STM32F417 CPU	98 mA
SRAM	23 mA
OV5642 Sensor	156 mA
WiFi (TX)	140 mA
TPM	21 mA
Total	438 mA

TABLE III. INPUT CURRENTS TO THE INDIVIDUAL COMPONENTS WHEN EXECUTING THE PROCESSING PIPELINE SHOWN IN FIGURE 6.

YUV422 image is processed by a motion detection module that implements a variant of the multimodal mean background modeling techniques proposed by Apewokin et al. [23]. For each pixel position a set of possible average background pixel values is stored and current pixel values are compared to these potential background pixels. If there is no match the pixel is declared foreground and a new possible background pixel value is added to the background model. The resulting binary foreground image is post-processed and if the amount of motion is above a predefined threshold the JPEG image is handed over to the security task. It uses the hardware crypto engines of the microcontroller for data encryption (AES256). Non-repudiation guarantees for images are implemented using the TPM's signing and timestamping features. Note that we timestamp frame groups to ensure proper frame order. Finally, the data is handed over to the WiFi task. The WiFi module comes with an on-chip TCP/IP stack such that no network stack is required on the main CPU. All bulk data transmission to the WiFi module is implemented via DMA transfers.

With the processing pipeline from Section V, TrustEYE.M4 delivers an encrypted and digitally signed Motion-JPEG stream at a resolution 640×480 pixels at approximately 16 fps. Subsequently we consider additional aspects such as task runtimes, memory usage and power consumption. Table I shows the runtimes (average over 100 frames) for the individual processing steps of the video delivery application. Processing is performed on YUV422 images with a resolution of 320×240 pixels. Reading images from the sensor and transferring the final results to the WiFi module is implemented via DMA transfers. Therefore, they are not included in Table I. Parsing the interleaved JPEG/YUV422 image data takes about 9 ms. Foreground computation takes on average 41.8 ms. For comparison, a simpler foreground detection approach based on a single running average takes 17 ms. Bounding box calculation is performed in less than 5 ms. SHA1 computation and AES256 encryption contribute only little to the overall runtime of 58.7 ms per frame. The TPM TickStampBlob operation takes 172 ms which is too long for signing individual frames. Since we want to ensure correct frame order, we sign and timestamp hash chains of groups of 25 frames. The TPM operates in parallel to the main CPU and its runtime is therefore not included in Table I. Overall runtime leads to a theoretical frame rate of 17 fps. The difference to the previously mentioned 16 fps can be explained by overheads for interrupt handling, DMA setup etc. which not covered in Table I.

Memory requirements for the individual components are presented in Table II. Double-buffered communication between tasks is noted with the respective elements in the table. The 'Heap' memory entry represents memory used for management purposes and includes also data structures used internally by FreeRTOS. Not included in the table is stack memory for which the internal SRAM of the microcontroller is used. With 1615 kB, memory usage is well below the available 4 MB of external SRAM. The size of the application binary that is stored in the microcontroller's Flash memory is 61 kB (compiled with GNU GCC using -Os optimization). The achieved WiFi datarate of 9.8 Mbit/s is sufficient for the framerate and the amount of data that can be handled by TrustEYE.M4 CPU. Finally, Table III presents the average currents drawn by individual hardware components of TrustEYE.M4. The CPU

(STM32F417) draws 98 mA, the SRAM 23 mA, the TPM 21 mA and the WiFi module draws 140 mA. The 156 mA drawn by the OV5642 are higher than the current drawn by any of the other components. The high power consumption of 514.8 mW is responsible for more than a third of the platform's 1.45 W total power consumption. Contrary to the image sensor, the relatively high power consumption of the WiFi module is not unexpected. Based on the detected motion the WiFi module could be turned on only if required and thereby substantial power savings can be achieved.

VI. Summary and Outlook

We presented TrustEYE.M4 - a platform for secure sensing applications. Contrary to previous platforms, security features have been important design requirements. We demonstrated the capabilities of TrustEYE.M4 with a home monitoring application. NFC-based link establishment between the camera device and the user's mobile device enables simple and secure data delivery via existing cloud infrastructure without implicitly trusting the cloud service provider.

Ongoing work is divided into research on sensor-level privacy protection and evolution of the TrustEYE.M4 platform. In this work privacy protection is achieved by encryption all image and video data. Depending on the application, other nuances between full protection (encryption) and full access to videos might be required. In, e.g., assisted living applications a third party such as a first responder might need access to video showing behavioral information to assess a potential emergency situation. We are exploring cartooning effects [24] to provide such an intermediate level of protection. We plan to improve the TrustEYE.M4 platform by integrating a second, low-power radio for establishing a local mesh network between multiple sensing nodes without the need for the high-power WiFi radio. Also related to power-optimization is the integration of duty-cycling strategies and exploitation of low-power and sleep modes of the platform's components.

Acknowledgment

This work is part of *TrustEYE: Trustworthy Sensing and Cooperation in Visual Sensor Networks* [25] and is funded by the European Regional Development Fund and the Carinthian Economic Promotion Fund (grant KWF-3520/23312/35521).

References

[1] M. Reisslein, B. Rinner, and A. Roy-Chowdhury, "Special Issue on Smart Camera Networks," *IEEE Computer*, 2014, (to appear).

[2] A. Cavoukian, "Surveillance, Then and now: Securing Privacy in Public Spaces," Tech. Rep., 2013.

[3] M. Bramberger, J. Brunner, B. Rinner, and H. Schwabach, "Real-Time Video Analysis on an Embedded Smart Camera for Traffic Surveillance," in *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2004, pp. 174–181.

[4] D. Farmer and C. C. Mann, "Surveillance Nation (Part I)," *Technology Review*, vol. 4, pp. 34–43, 2003.

[5] M. Brezovan and C. Badica, "A Review on Vision Surveillance Techniques in Smart Home Environments," in *Proceedings of the International Conference on Control Systems and Computer Science*, 2013, pp. 471–478.

[6] H. Aghajan, J. C. Augusto, C. Wu, P. Mccullagh, and J.-A. Walkden, "Distributed Vision-Based Accident Management for Assisted Living," in *Proceedings of the International Conference on Smart Homes and Health Telematics*, 2007, pp. 196–205.

[7] S. Fleck and W. Straßer, "Smart Camera Based Monitoring System and its Application to Assisted Living," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1698–1714, 2008.

[8] Dropcam, "Dropcam Website," 2014, last visited: June 2014.

[9] T. Winkler and B. Rinner, "Security and Privacy Protection in Visual Sensor Networks: A Survey," *ACM Computing Surveys*, vol. 47, no. 1, p. 42, 2014.

[10] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. B. Srivastava, "Cyclops: In Situ Image Sensing and Interpretation in Wireless Sensor Networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems*, 2005, p. 13.

[11] M. Rahimi, D. Estrin, R. Baer, H. Uyeno, and J. Warrior, "Cyclops, Image Sensing and Interpretation in Wireless Networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems*, 2004, p. 311.

[12] K. W. Agyeman and A. Rowe, "CMUcam4 Feature List," Tech. Rep., 2012.

[13] R. Kleihorst, A. Abbo, B. Schueler, and A. Danilin, "Camera Mote with a High-Performance Parallel Processor for Real-Time Frame-Based Video Processing," in *Proceedings of the International Conference on Distributed Smart Cameras*, 2007, pp. 109–116.

[14] A. Abbo, R. Kleihorst, V. Choudhary, L. Sevat, P. Wielage, W. Mouy, B. Vermeulen, and M. Heijligers, "Xetal-II: A 107 GOPS, 600 mW Massively Parallel Processor for Video Scene Analysis," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 192–201, 2008.

[15] P. W.-C. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. J. Lobaton, M. L. Meingast, S. Oh, S. Wang, P. Yan, A. Yang, C. Yeo, L.-C. Chang, D. Tygar, and S. S. Sastry, "CITRIC: A Low-Bandwidth Wireless Camera Network Platform," in *Proceedings of the International Conference on Distributed Smart Cameras*, 2008, p. 10.

[16] S. P. Mohanty, "A Secure Digital Camera Architecture for Integrated Real-Time Digital Rights Management," *Journal of Systems Architecture*, vol. 55, no. 10-12, pp. 468–480, Oct. 2009.

[17] A. Chattopadhyay and T. E. Boult, "PrivacyCam: A Privacy Preserving Camera Using uClinux on the Blackfin DSP," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[18] O. Adamo, S. P. Mohanty, E. Kougianos, and M. Varanasi, "VLSI Architecture for Encryption and Watermarking Units Towards the Making of a Secure Camera," in *Proceedings of the Int. System-on-Chip Conference*, 2006, pp. 141–144.

[19] L. De Strycker, P. Termont, J. Vandewege, J. Haitsma, A. Kalker, M. Maes, and G. Depovere, "Implementation of a Real-time Digital Watermarking Process for Broadcast Monitoring on a TriMedia VLIW Processor," *IEE Proceedings - Vision, Image, and Signal Processing*, vol. 147, no. 4, p. 371, 2000.

[20] P. Stifter, K. Eberhardt, A. Erni, and K. Hoffmann, "Image Sensor for Security Applications with On-chip Data Authentication," *Proceedings of the Society of Photo-Optical Instrumentation Engineers*, vol. 6241, p. 8, 2006.

[21] T. Winkler and B. Rinner, "Securing Embedded Smart Cameras with Trusted Computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, p. 20, 2011.

[22] ——, "TrustCAM: Security and Privacy-Protection for an Embedded Smart Camera based on Trusted Computing," in *Proceedings of the International Conference on Advanced Video and Signal-Based Surveillance*, 2010, pp. 593–600.

[23] S. Apewokin, B. Valentine, L. Wills, S. Wills, and A. Gentile, "Multimodal Mean Adaptive Backgrounding for Embedded Real-Time Video Surveillance," *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 1–6, Jun. 2007.

[24] T. Winkler, A. Erdélyi, and B. Rinner, "TrustEYE.M4: Protecting the Sensor - not the Camera," in *Proceedings of the International Conference on Advanced Video and Signal Based Surveillance*, 2014, p. 6.

[25] T. Winkler, "TrustEYE Project Website," http://trusteye.aau.at, 2012, last visited: December 2014.