2016 IEEE Computer Society Annual Symposium on VLSI

FPGA Based Cyber Security Protocol for Automated Traffic Monitoring Systems: Proposal and Implementation

(Invited Paper)

Anupam Chattopadhyay, Vikramkumar Pudi, Anubhab Baksi and Thambipillai Srikanthan
School of Computer Science & Engineering, Nanyang Technological University, Singapore

***Abstract*—There is a rapidly growing interest in the field of unmanned road vehicles across the world. To aid the traffic management of such systems, there is an urgent need to develop appropriate security protocols facilitating car-to-car and car-to-traffic controller systems. Ensuring security requires both confidentiality (will be understandable only to intended recipients) as well as authenticity (message is not tampered during communication), both of which are taken care of in an Authenticated Encryption with Associated Data (AEAD) scheme. In this paper, we propose a new AEAD-based protocol for secure and authenticated transmission of videos to the base station captured by traffic monitoring systems in real time. Our protocol utilizes ACORN v2, a lightweight AEAD primitive. For the secret key to be used in encryption-authentication, we use the concept of Physically Unclonable Functions (PUFs). The entire protocol is implemented and evaluated with an FPGA-based prototype, using a 640×480 pixel camera with 30 frames per second. The area required for the proposed protocol is 5% of the total FPGA device (Xilinx Zynq-XC7Z020-1clg484).**

I. Introduction

Automated cars and automated traffic monitoring systems are of great interest nowadays. In this direction, one may notice that, Tesla's Model 3 car has 325k pre-orders, totaling an amount of $14.5B in just one week after the announcement for the car is made [1]. Another example would be, the US government has already fixed vehicles based on the level of human participation requirement in to five categories; ranging from No-Automation (Level 0) to Full Self-Driving Automation (Level 4) [2]. Although such vehicles are expected to be on road within a few years, there is a lack of standardized cyber security protocols for communications among the cars, the traffic monitoring systems and the base stations (which analyze the data sent by the traffic monitoring systems). In this work, we focus on the issue of wireless communication of the Automated Traffic Monitoring Systems (ATMS) with the base station, and propose a cyber security protocol.

The novelty of our protocol lies in the use of Physically Unclonable Functions (PUFs) [3], [4]; as a mean to solve the issue of secret key establishment. To the best of knowledge, this is the first of this kind. The usual wireless communication can be carried out with the help of standard protocols like IEEE 802.11 [5]. Thus, we do not intent to develop a full-fledged protocol like IEEE 802.11. Rather our protocol offers flexibility in the sense that, it could be used on top of any wireless communication protocol, like IEEE 802.11 or some military protocol. Hence, we consider attacks like spoofing, denial of service etc as out of scope for this work, which can be taken care of by the actual communication protocol itself.

In the subsequent sections, we discuss the our proposed protocol in details. Further, we demonstrate the practicality of our protocol by implementing it on Xilinx FPGA board [6], which is summarized in Section VI.

II. Background

Self driving cars like Google driver-less car [7] communicate with ATMS for directions and guide lines; like speed limit, school zones, road construction, traffic density etc. An ATMS uses cameras for road traffic monitoring to observe particulars and facilitates necessary traffic information to be available in places like traffic signals, bus information display etc; as shown in Fig. 1.

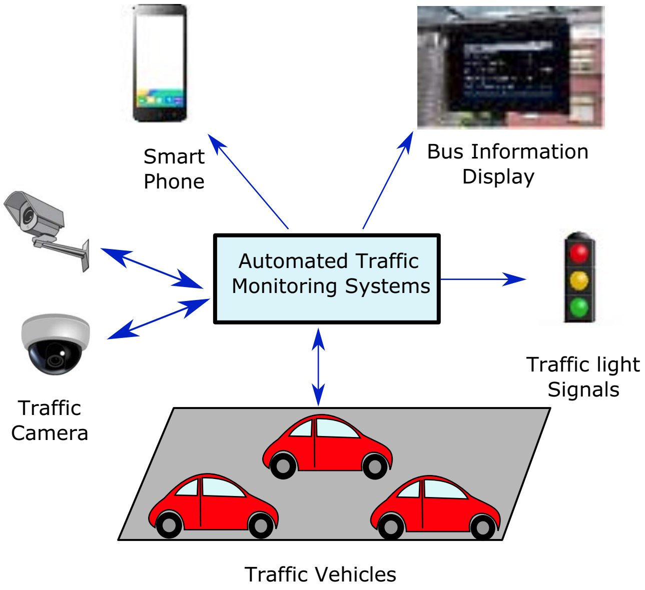


Fig. 1. Functionality of Automated Traffic Monitoring Systems

Thus, the data from the ATMS' need to be sent securely to the base station such that any malicious person[1] can not send any wrong/misleading information about the real traffic.

[1] We call such a person an attacker from now on.

978-1-4673-9039-2/16 $31.00 © 2016 IEEE
DOI 10.1109/ISVLSI.2016.97

Here we argue that such communication is to be encrypted to provide confidentiality as well as authentication to prevent tampering. Encryption is needed to guarantee secrecy: Only the intended recipient can understand it. This scheme works as follows. There is a secret key, which is known to the sender (ATMS) and the receiver (base station). The sender sends the data after encrypting with this key, the receiver understands the intended data by decrypting it with the same key. For anyone, who has no knowledge of the key, the encrypted message will look like a random stream. Further, suppose that the attacker is able to alter the data being communicated. To resist such a case, we would use authentication; i.e., we would generate a tag (which is unique for each data) and append the tag with the data. The base station would compute tag for the data, then check if both the tags (generated and received) match.

ATMS' typically operate in constrained resources and transmit data over wireless networks. This means that, the modules intended to provide security should be light-weight in hardware. Further, it is recommended that the same secret key should not more than once (for example, as discussed in [8]).

In [9], [10], [11], authors proposed methods for preserving security in road traffic monitoring system, but they did not address hardware/software complexity of implementing these methods. Moreover, those methods store secret keys in the device memory. This suggests that, an attacker is able to extract secret keys from senor nodes, thereby causing a serious security threat.

III. Scope of Our Work

Our protocol deals with the situation when several ATMS' are sending information to one base station. Each ATMS captures real time video, (with a secret key) encrypts it, and sends to the base station along with an authentication tag. The base station, having known the secret key, decrypts the video and checks if the tag is valid. Here we focus on the problem of communicating the secret keys with the ATMS and the base station. The base station contains necessary tools (human or automated) to take appropriate decisions (e.g., license plate recognition, crowd monitoring, vehicle collision detection, traffic light monitoring, traffic queue estimation etc.), and is assumed to be out of reach for the attacker.

Now, the first challenge here is to pre-distribution of the keys. In this regard, we propose to use one inherent hardware property, Physically Unclonable Function (PUF) [3], [4], of the ATMS to be used as the key. In principle, PUF will receive challenge as an input and generates corresponding output used as a secret key. PUFs are easy to get for a given hardware, and are difficult to replicate since it is generated by some kind of noise at the time of manufacturing. However, for a given chip, one can record all PUFs and store it. In our case, we pre-compute and store these PUF information to the base station before deployment of the ATMS'. Our proposed PUF implementation requires less than 1 GB of memory for storing pre-computing the outputs from PUF at base station. Using todays current technology we can easily offer the memory required for all ATMS' sensors nodes for storing PUF outputs at base station. Now, the ATMS' use different PUFs as the secret key for different communications; which is known to the base station, as it already holds the record for all PUFs. Thus, the problem of key establishment is resolved. Since PUFs are inherent to the hardware, it can be used when constrained hardwares are in use. In another direction, one may note that, while we implement our protocol in Field Programmable Gate Arrays (FPGA) the methods presented in [12] are not suitable for FPGA implementation. Moreover, the use of reconfigurable PUFs [13] makes it possible to get the secret key length of desired size.

As we mentioned earlier, the communication from the ATMS' to the base station should be encrypted as well as tampering proof. Now, Authenticated Encryption with Associated Data (AEAD) schemes aim at providing both confidentiality and authenticity together. Hence, we choose one light-weight AEAD scheme, named ACORN v2[2] [14]. The secret key for this AEAD is obtained from the PUF, thus the PUF is fed to the AEAD circuit (which prevents the PUF being exposed to prevent modeling attack [15]). One may argue that, instead of a light-weight AEAD, some more popular schemes like AES [16] or some other Public Key cyrptosystem with authentication module. While such arguments are valid indeed, we would like to emphasize on the point that, ACORN, being a light-weight AEAD, consumes less area compared to AES/Public Key Cryptosystem with authentication module. For example, public key cryptosystems like Diffie-Hellman, RSA etc in ATMS sensor nodes require exponential computation complexity [12]. This reduction of hardware is particularly beneficial for tightly constrained devices like ATMS.

To summarize, our protocol uses PUFs for the key predistribution between ATMS and the base station, which is cheaper compared to conventional methods. This key is used in a light-weight AEAD scheme for encryption and authentication. Finally, these whole process works on top of a full-fledged wireless communication like IEEE 802.11.

Now, we discuss some of the associated topics which are out of scope of this work. We intentionally do this for the time being, and hope to propose an extension of this work that could address all these issues in near future.

Here are such a few exemplary cases. There is need for a synchronization mechanism between the ATMS' the base station in case some data packet is lost. Also, the ATMS' should receive a handshaking from the base station. The time-stamp, message counter should be associated with each data packet the ATMS' are sending. Moreover, as our implementation is only for practical demonstration purpose, we use random bits for IV and AD here, which needs to be standardized. Last, but not the least, one should consider the cases for snooping (data packets can be captured completely by the attacker, and the base station receives no response from ATMS') and the denial of service.

[2] Henceforth, we use 'ACORN' and 'ACORN v2' interchangeably.

IV. Proposed Cyber Security Protocol

This section presents the proposed protocol for automated traffic monitoring systems and its architecture. The architecture of proposed protocol contains mainly key generation and authenticated encryption modules. Special care is taken in order to ensure a lightweight architecture realization.

A. Cyber Security Requirements

We briefly revisit the major security features required for such a protocol.

- **Security:** The main requirement of any security protocol is confidentiality of transmitted data and received data.
- **Message Integrity:** The data packets transmitted from sensor nodes cannot be modified by a malicious attacker. For this purpose, we use authentication tag of size 128-bits and its generation depends on input data and secret key.
- **Secret Key Generation and Exchange:** The generation of encrypted data and authenticated tag depends mainly on secret key. In our proposed security protocol we do not transmit the secret key via any public-key protocol. Rather, only the inputs (called, *Challenge*) to the PUF are transmitted. At the sensor node we generate secret key to avoid memory based attacks, whereas at the server side we will store the secret key tables for corresponding challenge.
- **Efficiency:** The available resources at ATMS traffic sensor side are limited and most of the traffic monitoring cameras are used for transmission of data. Because of this we have to use lightweight and low cost modules.

B. Components in Our Protocol

Fig. 2 shows the architecture of our proposed protocol used in ATMS. It consists of four components: Keygen, TagEncrypt, TagDecrypt and Authenticate, which we explain shortly. PUFs are used in the key generation (Keygen) module, authenticated encryption and decryption (using ACORN) used as TagEncrypt and TagDecrypt modules and a simple comparison circuit used as Authenticate module. A pictorial view is given in Fig. 2.

- *Keygen:* Takes challenge Ch as input and generates the secret key K. We denote this operation as $K =$ Keygen(Ch). This module is implemented using PUFs.
- *TagEncrypt:* Takes the secret key K, associated data AD, initialization vector IV and plain-text PT as inputs and generates cipher-text CT and authentication tag T (based on the ACORN encryption module) as outputs.
- *TagDecrypt:* Receives the secret key K, associated data AD, initialization vector IV and cipher-text CT as inputs and generates plain-text PT and a new authenticated tag T' (based on the ACORN decryption module) as outputs.
- *Authenticate:* Takes authenticated tags' T and T' from the TagEncrypt and TagDecrypt modules and returns a validation signal V as output. If $V = 1$, then it means that both the tags are equal, so received data is authenticated; otherwise the received data is not authenticated, possibly because an attacker was able to send a tampered/fake data to the base station.

C. Authenticated Encryption Using ACORN v2

An AEAD scheme provides both authenticity and confidentiality. Thus, it is a natural choice for our case. Here we take ACORN, which is lightweight. It takes one secret key (denoted by K henceforth; to be known only to sender and recipient), Initialization Vector (IV), Plain-text (PT), Associated Data (AD) as inputs and outputs one Cipher-text (CT) & a tag (T). For the plain-text, we want the encryption, while we want the authentication to be done on both associated data and plain-text. Thus, the cipher-text (to be sent to the intended recipient) depends on the plain-text: It is obtained by XORing one binary stream (called, key-stream) with the plain-text; and the tag depends on both the plain-text and AD.

The core of ACORN is a stream cipher having a state register of 293 bits, which consists of six Linear Feedback Shift Registers (LFSRs) of different size, and one four bit Non-linear Feedback Shift Registers (NFSRs) which is updated from a feedback (f) of the whole state. At each update (call it *StateUpdate*), one input bit, m (through which K, IV, AD and PT are inserted), is fed to the NFSR. Now, K (128-bit), IV (128-bit), AD (up to 2^{64}-bits) and PT or CT (based on encryption or decryption; up to 2^{64}-bits) are loaded with this bit. The tag is of 128 bits, which is the recommended length by the designer.

We exclude detailed description, interested readers may find it in [14]. For the sake of completeness, we present a compact description here. Each StateUpdate function, at i-th round, performs three operations: Update each LFSRs, compute feedback (f_i) and key-stream bit (ks_i), accept one external bit (m_i). Before processing AD, the cipher is updated 1536 rounds with key K and IV. Then, it updates the state for $512 +$ length(AD) rounds while processing AD. While processing PT or CT, the state is also updated in a similar fashion.

The key-stream bits (to be XORed with PT to yield CT) and the tag bits are both generated by ks_i, but for separate rounds, i.e., for separate values for i. From $i = 2048 +$ length(AD) to $i = 2048 +$ length(AD) $+$ length(PT), we get the key-stream; and from $i = 2943+$length$(AD)+$length(PT) to $i = 3072 +$ length(AD) $+$ length(PT) we get the tag. The processing of CT along with T is very similar.

Denoting the j-th state bit at i-th round by $S_{i,j}$ for $j = 0, 1, \ldots, 292$; then, each StateUpdate is described by the following operations:

$$\begin{aligned}
S_{i,289} &\leftarrow S_{i,289} \oplus S_{i,235} \oplus S_{i,230}\\
S_{i,230} &\leftarrow S_{i,230} \oplus S_{i,196} \oplus S_{i,193}\\
S_{i,193} &\leftarrow S_{i,193} \oplus S_{i,160} \oplus S_{i,154}\\
S_{i,154} &\leftarrow S_{i,154} \oplus S_{i,111} \oplus S_{i,107}\\
S_{i,107} &\leftarrow S_{i,107} \oplus S_{i,66} \oplus S_{i,61}\\
S_{i,61} &\leftarrow S_{i,61} \oplus S_{i,23} \oplus S_{i,0}\\
S_{i+1,292} &\leftarrow f_i \oplus m_i,
\end{aligned}$$

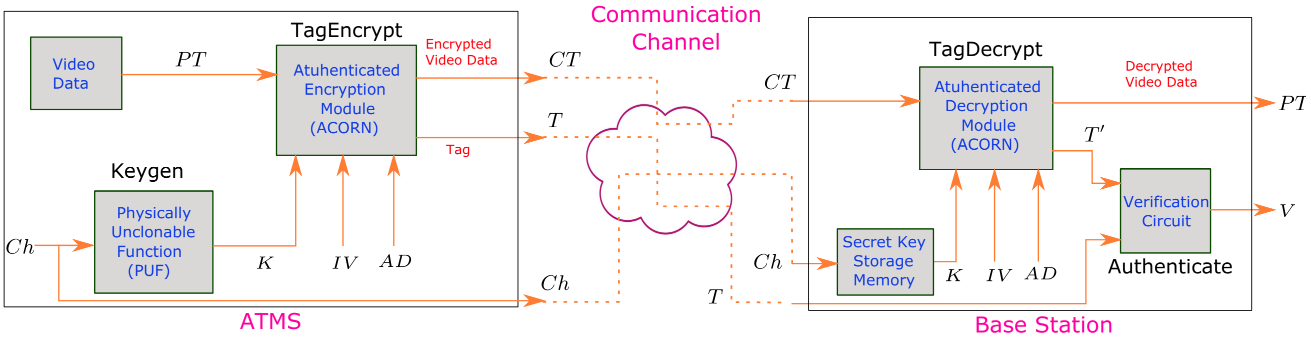


Fig. 2. Overview of the Proposed Protocol

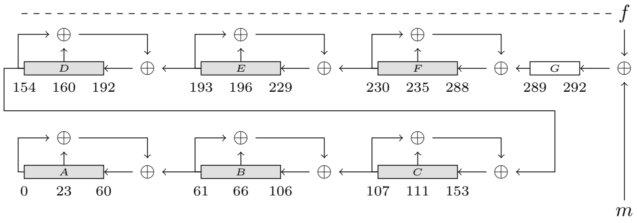


Fig. 3. Structure of ACORN: six LFSRs (A-F) and one NFSR (G)

$$ks_i \leftarrow S_{i,12} \oplus S_{i,154} \oplus (S_{i,235} \wedge S_{i,61}) \oplus (S_{i,235} \wedge S_{i,193}) \oplus (S_{i,61} \wedge S_{i,193}),$$

$$f_i \leftarrow S_{i,0} \oplus (\neg S_{i,107}) \oplus (S_{i,244} \wedge S_{i,23}) \oplus (S_{i,244} \wedge S_{i,160}) \oplus (S_{i,23} \wedge S_{i,160}) \oplus (S_{i,230} \wedge S_{i,111}) \oplus ((\neg S_{i,230}) \wedge S_{i,66}) \oplus (ca_i \wedge S_{i,196}) \oplus (cb_i \wedge ks_i).$$

Here the variables m_i, ca_i, cb_i are determined by some simple rules, which can be found in [14, Chapter 1].

D. Physically Unclonable Functions for Key Generation

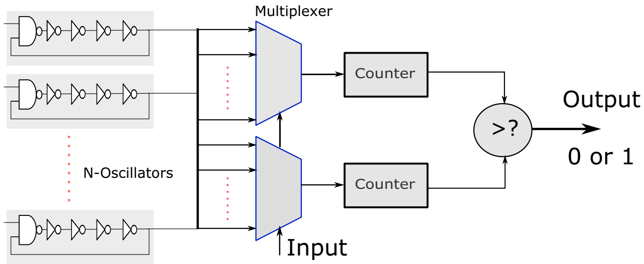


Fig. 4. Block Diagram of Ring Oscillator based PUF

In this work, we implemented Ring Oscillator Physically Unclonable Function (RO-PUF), which has simpler architecture compared to the other PUFs described in [3]. Each RO-PUF is made of two N-bit multiplexers, two counters, one comparator and N-ring oscillators (ROs), as shown in Fig. 4. Every ring oscillator in RO-PUFs contains odd number of inverters connected in a loop. The output of each ring oscillator depends upon the physical characteristics of inverter. The characteristics of each inverter vary within and across a chip due of manufacturing variations, based on which the ring oscillator achieves a unique frequency.

In the RO-PUF, depending upon the multiplexer selection logic, two ROs provides inputs to the counter. The counters are compared in fixed intervals (called, the comparison time) to generate the PUF output bit (called, *Response*). RO-PUF produces single output bit at each comparison interval. We can generate more PUF bits by comparing the counters at multiple times with fixed comparison intervals. The output of the PUF can be used as low cost authentication method for devices and secret key generation [3], [4].

In our protocol we use PUF architecture presented in [17], which uses 256 ring oscillators and it is implemented using FPGA. The experimental results in [17] shows that, this PUF will give the correct output response almost 95% of the time and more than 40% of output bits are different for different PUFs with the same inputs. By applying challenge Ch as input to the PUF, we generate secret key K, bit by bit at each comparison intervals.

V. Security Claims

Our protocol is secure against some common attack scenarios, as described here.

- There could be some attacks in the wireless communication; like confidentiality, integrity and denial of service (DoS) attacks [18]. In the confidentiality attack, the attacker secretly attempts to get the information being communicated, which we prevent by encrypting it. In integrity attack, the attacker is able to alter the data from the ATMS, including data deletion/addition, flipping bits, or sending the same data multiple times (*Replay attacks*). In such a case, the Tag would be a mismatch, and the base station could easily identify such an incident. Also, as we mentioned earlier, this protocol is not suitable to resist DoS attack by itself, rather it is dependent on the actual protocol (for wireless communication) for resolving such an issue.
- Capturing one ATMS would only reveal the secret keys associated with it, all the other ATMS' will remain secure.

- It is known that, if PUF is exposed to the public directly, then it would be subject to modeling attack [15]. However, our protocol does not expose PUF, rather it uses PUF internally. Thus, this attack is not applicable for our protocol.

VI. FPGA Implementation & Results

In this section, we present the prototype of proposed protocol for ATMS and its FPGA implementation results. Here, we are using OV7670 camera as the traffic camera under surveillance; one FPGA based board, Zedboard [6] for implementation shown in Fig. 2 and LCD monitor for displaying the outputs from the board. Fig. 5 shows the experimental setup of proposed protocol. For Keygen module we use PUF construction explained in section IV-D. This PUF takes a 16-bit challenge Ch to generate key K. For the PUF implementation, we use 256 ring oscillators with seven inverters, two $256:1$ MUXes, two 32-bit counters and one 32-bit comparator. We use 0.01 ms as comparison interval for taking each bit of secret key from the PUF.

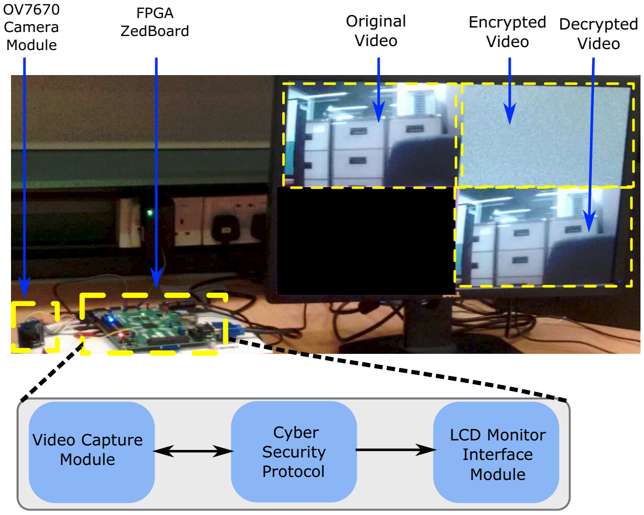
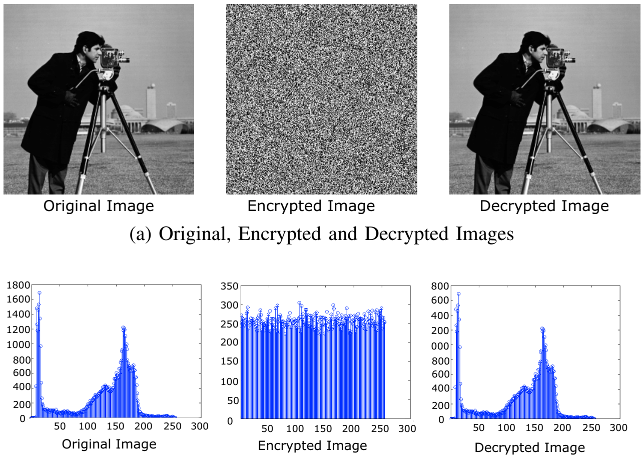


Fig. 5. Demonstration for the Proposed Protocol using FPGA

The camera module captures real time video with a resolution of 640×480 at 30 frames per second. This video is fed as the input to the TagEncrypt module. For TagEncrypt and TagDecrypt modules we use ACORN (described in section IV-C). In our protocol demonstration, we choose random bits of size 128 and 256 for IV and AD respectively, as inputs for ACORN. In our current setup we are using single LCD monitor for displaying the original, encrypted and decrypted video. The Fig. 5 shows the original, encrypted and decrypted real time video displayed on LCD monitor.

A. Experimental Results

Fig. 6 shows the original, encrypted and decrypted images, and its corresponding histograms. The plain image (middle subfig.) is encrypted using ACORN with the secret key generated from PUF from the orginal image (left subfig.), the decrypted image (right subfig.) is obtained from the encrypted image with the same secret key. The corresponding histograms are shown at the bottom of Fig. 6; one may notice that the histogram for the encrypted image is uniformly distributed.



(a) Original, Encrypted and Decrypted Images

(b) Histograms of Original, Encrypted and Decrypted Images

Fig. 6. Encrypted and Decrypted Images with its corresponding Histograms

Table I shows the resource consumption and speed of the PUF, ACORN and prototype of proposed security protocol design results using targeted Zedboard with Xilinx Zynq-XC7Z020-1clg484 device. The area consumption of proposed protocol is 5% of overall FPGA device. The area consumption of PUF is more than ACORN, because we designed RO-PUF using the method presented in [17], this design uses 256 ring oscillators. For low area RO-PUF we can reduce the ring oscillators to 128 or 64, this saves area required for RO-PUFs. Table II shows the performance comparisons of our authenticated encryption(ACORN) FPGA results with existing AES-GCM and AES-SHA256 results presented in [19]. From Table II, we observe that ACORN module require less area compared to the AES-GCM and AES-SHA256.

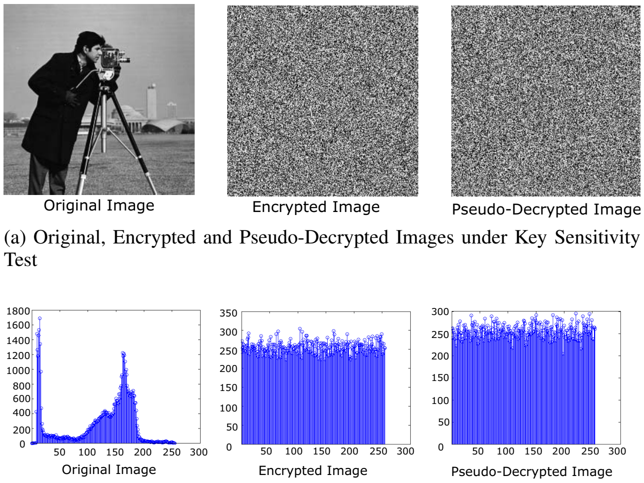
TABLE I
Synthesis Results of Proposed Designs on Zedboard

Module	Slice LUTs	Registers	Frequency(MHz)
PUF	641	269	333.3
ACORN	146	109	271
Complete Protocol	2726	1517	222.2

TABLE II
FPGA Performance Comparisons

Module	Device	Slice LUTs	Frequency (MHz)
ACORN	XC5VLX50T	377	238
AES-GCM [19]	XC5VLX50T	2687	91.6
AES-SHA256 [19]	XC5VLX50T	2730	100

Ideally, a good AEAD scheme should be such that, even if a single bit of the secret key is flipped after the encryption, then it would be neither possible to get the plain-text from the cipher-text nor the tags are matched. We try to explore the vulnerability of our implementation under such a scenario, which is known as the *Key Sensitivity Test*. Since there is not much scope to show outputs in details, we only show one example with encryption and decryption only. Here, Fig. 7 shows the original, encrypted and pseudo-decrypted images, and its corresponding histograms under key sensitivity test. For the set-up, we generate the encrypted image from one particular key, flip the MSB of it, and then try to decrypt with this altered key. The output (pseudo-decrypted) image in Fig. 7 completely different from original image, the histogram of the pseudo-decrypted image is uniformly distributed and it is similar to the encrypted image. Similar we have conducted experiments by flipping single bit of IV and AD, we have obtained similar results as key sensitivity test, which we do not include here.



(a) Original, Encrypted and Pseudo-Decrypted Images under Key Sensitivity Test

(b) Histograms of Original, Encrypted and Pseudo-Decrypted Images under Key Sensitivity Test

Fig. 7. Encrypted and Decrypted Images with its corresponding Histograms under Key Sensitivity Test Results

VII. Conclusion

This work aims at developing a protocol for key establishment, encryption and authentication between multiple ATMS' and one base station. For this purpose, we use Physically Unclonable Functions (PUFs); along with an Authenticated Encryption with Associated Data (AEAD) scheme (in this case, we choose ACORN v2 as the AEAD). To the best of our knowledge, this is the first work in this field, which offers key establishment to be done with low hardware overhead. Also, as we discussed, it can resist a few common attack scenario. Moreover, it allows flexibility to be used in association with some standard wireless communication protocol (like IEEE 802.11).

We would like to mention that, we implement our proposed protocol on a FPGA based Zedboard from Xilinx. Our implementation occupies 5% of total area of FPGA device, thereby keeping remaining area for other processing elements (compression, detection etc.). We successfully processed (encrypted/decrypted and authenticated) real time videos captured from camera with resolution of 640×480 pixels at 30 frames per second.

References

[1] Tesla's model 3 now has 325k pre-orders – and $14.5b in potential sales. http://www.computerworld.com/article/3053553/car-tech/teslas-model-3-now-has-325k-pre-orders-and-145b-in-potential-sales.html. Online; accessed 29-April-2016.

[2] U.s. department of transportation releases policy on automated vehicle development. http://www.nhtsa.gov/About+NHTSA/Press+Releases/U.S.+Department+of+Transportation+Releases+Policy+on+Automated+Vehicle+Development. Online; accessed 29-April-2016.

[3] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, 2007, pp. 9–14.

[4] A. Maiti, I. Kim, and P. Schaumont, "A robust physical unclonable function with enhanced challenge-response set," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 333–345, 2012.

[5] "IEEE Standard for Wireless LAN and mdash, Medium Access Control and Physical Layer Specification, P802.11," Nov. 1997.

[6] (2012) Avnet-zedboard: A development board for xilinx zynq-7020. [Online]. Available: http://zedboard.org/

[7] S. L. Poczter and L. M. Jankovic, "The google car: Driving toward a better future?" *Journal of Business Case Studies (Online)*, vol. 10, no. 1, p. 7, 2014.

[8] Taking advantage of one-time pad key reuse? http://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse. Online; accessed 29-April-2016.

[9] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, "Enhancing security and privacy in traffic-monitoring systems," *IEEE Pervasive Computing*, vol. 5, no. 4, pp. 38–46, 2006.

[10] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[11] A. J. Blumberg, L. S. Keeler, and A. Shelat, "Automated traffic enforcement which respects" driver privacy"," in *Intelligent Transportation Systems: Proceedings of IEEE*, 2005, pp. 941–946.

[12] C. K. Li, G. Yang, D. S. Wong, X. Deng, and S. S. Chow, *An efficient signcryption scheme with key privacy*. Springer, 2007, pp. 78–93.

[13] S. Katzenbeisser, Ü. Koçabas, V. van der Leest, A. Sadeghi, G. J. Schrijen, and C. Wachsmann, "Recyclable pufs: logically reconfigurable pufs," *J. Cryptographic Engineering*, vol. 1, no. 3, pp. 177–186, 2011. [Online]. Available: http://dx.doi.org/10.1007/s13389-011-0016-9

[14] H. Wu. (2015) Acorn: A Lightweight Authenticated Cipher (v2). CAESAR Competition. [Online]. Available: https://competitions.cr.yp.to/round2/acornv2.pdf

[15] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, 2010, pp. 237–249. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866335

[16] "Announcing the Advanced Encryption Standard (AES)," National Institute of Standards and Technology, FIPS PUB 197, Nov. 2001.

[17] M. Patterson, J. Zambreno, C. Sabotta, S. Vyas, and A. Mills, "Ring oscillator puf design and results."

[18] Wireless hacking tools. http://www.cse.wustl.edu/~jain/cse571-07/ftp/wireless_hacking/. Online; accessed 29-April-2016.

[19] Y. Hori, A. Satoh, H. Sakane, and K. Toda, "Bitstream encryption and authentication with aes-gcm in dynamically reconfigurable systems," in *International Conference on Field Programmable Logic and Applications*, Sept. 2008, pp. 23–28.