Addressable PUF Generators for Database-free Password Management System

Bertrand Cambou
bertrand.cambou@nau.edu
Northern Arizona University

ABSTRACT

The objective of our research effort is to mitigate a prevalent attack on Cyber Physical Systems: the hacking of databases of UserID-Password pairs. We developed a database-free password generator architecture based on memory arrays, and Addressable Physical unclonable function (PUF) Generator (APG). APGs can generate passwords, and can authenticate a client on the network without keeping in memory UserID-Password pairs.

Keywords

Hardware Authentication, Password management, Memory arrays, Hash functions, Public Key Infrastructure, Physically Unclonable Functions, Ternary states.

1. INTRODUCTION

One of the most commonly reported cyber-attacks is the compromising of extremely large databases of User ID and password pairs. Recent events reported in the news include hackers stealing information by getting un-authorized access to databases within internet providers, US government agencies, the IRS, health institutions, political parties, banks, and many more. These types of cyber-attacks could create huge financial damages or psychological traumas. The essence of access control is to match a password, secret key, biometric print, or any other reference pattern that is associated with a particular user or peripheral against the same reference pattern that is stored in the secure host server. The vulnerability of the databases of reference patterns is a major threat. We are presenting the development of access control architectures that do not require the existence of a centralized database of reference patterns and passwords, thereby removing the risks related to stolen databases. The proposed architectures reuse known technology modules, such as memory based PUFs, true random number generators, hash functions, and public key infrastructure. We designed several prototypes to test the proposed schemes.

2. BACKGROUND INFORMATION

Public key infrastructure (PKI) is a method introduced in the 80's by Merkle, Hellman, and Diffie [1-3], see Fig. 1. Each user has a pair of keys, the public key that is not secret, and the highly secret private key. These two keys are interchangeably used to encrypt a message, while the second key is the only one capable of decrypting the message. RSA and elliptic curves (EC) are two widely used examples of PKI-based algorithms. After distribution of the secret key, as shown in Fig. 1, the protocol of communication between the secure server and the secure memories embedded in each IoT node is trustworthy. A loss to a hacker of the database of public keys is irrelevant because this data base is indeed "public". This protocol has several weaknesses for a network that integrates IoT devices; the first issue is that the secure memories in IoTs are not always immune to side channel attacks and therefore there is a chance for private key leakage; secondly, the overall key distribution is a challenging task, and can be risky. The loss of the private key is considered a total loss of the trust of the security protocol. If a node of the network is compromised, the distribution of a new key to the corresponding node is difficult.

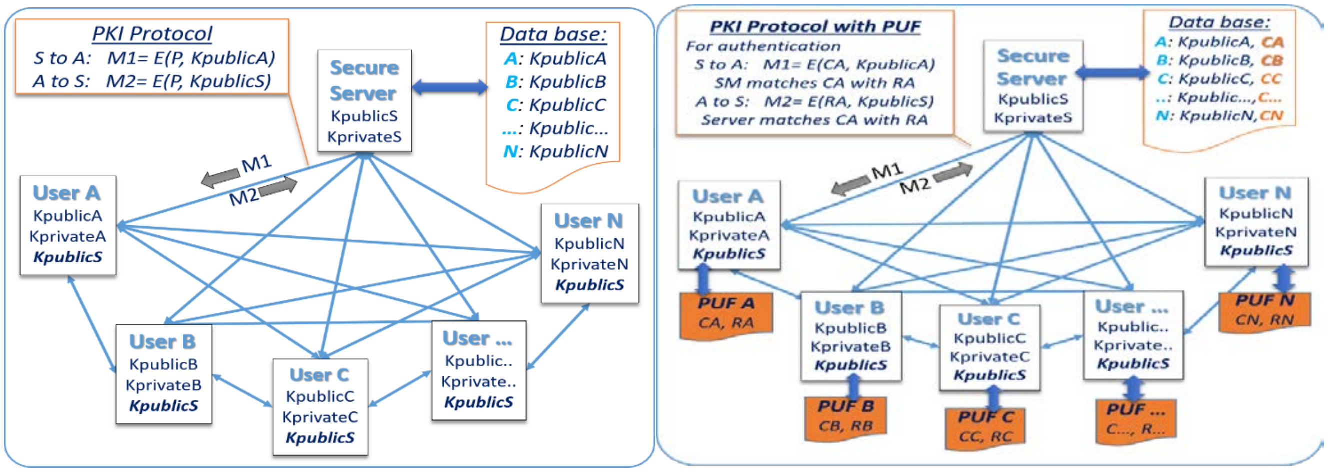


Fig. 1: Block diagram of a PKI protocol, left, protected by distributed PUF, right.

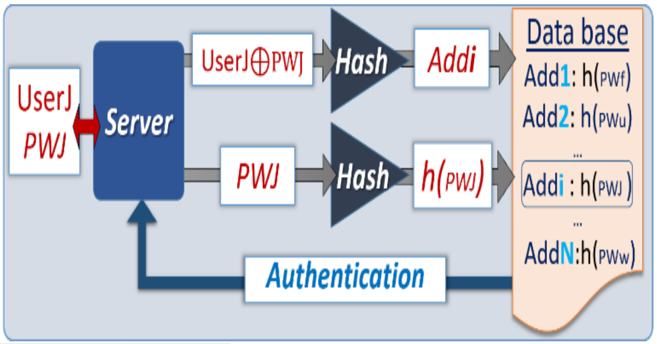
Physically Unclonable Functions (PUFs) can generate from hardware components the equivalent of human fingerprints; they are unclonable, and random. PUFs are low cost, and can strengthen the level of security of authentication protocols as part of a set of cryptographic primitives. PUFs exploit the intrinsic natural manufacturing variations introduced during fabrication of the devices such as local variations in critical dimensions, doping levels of semiconducting layers, and threshold voltages [4-10]. These variations make each device authenticable from each other. The underlying mechanism of PUF is the creation of a large number of Challenge (i.e. Input) Response (i.e. output) Pairs (called CRPs). Once deployed during the authentication cycles, the PUFs are queried with challenges. The authentication is granted when the rate of matching CRPs is statistically high enough (above a predefined threshold). A PKI protocol, which uses a set of distributed PUFs for access control, is shown in Fig. 2 [11]. Each user (client) is associated with its own PUF, which has to generate upfront a challenge that is stored under the control of the secure server, in the central database as reference pattern. Such a protocol enhance the security of PKI's, however the centralized database which keeps track of the challenges of the distributed PUFs is vulnerable, and must be protected [12-13], and need error correction methods [14].

Memory based PUF: Memory structures [15-17], SRAM [18-19], DRAM [20], Flash [21], ReRAM [22], and MRAM [23-24], are excellent elements to generate PUFs. Usually PUFs need only 128 to 256 bits to ensure an acceptable level of security, while commercial secure memory arrays (SM), which are integrated within secure micro-controllers, ordinarily have memory densities in the mega-byte range. One of the generic methods to generate CRPs is to characterize a particular parameter $\mathcal{P}$ of the cells of the array with a built-in-self-test (BIST) module [25]. The values of parameter $\mathcal{P}$ vary cell to cell, and follow a distribution with a median value $\boldsymbol{T}$. In order to generate challenge and response pairs, all cells with $\mathcal{P} < \boldsymbol{T}$ can be quantified as "0", and all others as "1". Assuming that these measurements are reproducible, the resulting streams of data can be used as cryptographic primitives to authenticate the memory array.

3. ADDRESSABLE DATA BASES

3.1 Hashing the passwords

Hash functions can be used to replace the tables containing User IDs with their corresponding passwords by look up tables, as shown in the block diagram of Fig.2. The hashing of the password "PWJ" results in the first message digest h(PWJ); the user ID "UserJ" and "PWJ" are XORed; the hashing of "UserJ⊕PWJ" results in the second message digest h(UserJ⊕PWJ). The second message digest generates the coordinate XY of the look up table storing the first message digest. In the example presented in Fig.2, the user#1 has a user ID "a6c26", a password "12ae5", and a first message digest "a639" generated with SHA-1; only the first five characters of the message digest are kept. The XORing the first three hexadecimal characters of the user ID and the password has a value of "b86". The hashing of "b86" with SHA-1 is generating a message digest that has "3e" as first two hexadecimal characters. We are then placing "a639" in the address "3e" of the table. Five more UserID/password pairs are stored in a similar way in the bottom right look up table of Fig.2. Such a method is applicable to store very large quantities of UserID/Password pairs. During authentication cycles, the users provide their UserID/Password pair; the information extracted from the table at the corresponding address is then compared with the message digest provided by the password. When SHA-1 is replaced by more powerful hashing functions (ex: SHA-3), such look up tables are much more secure than the tables directly storing UserID/Password pairs. However, when look up tables storing message digests are lost to the enemy, the information can be lost overtime with big data analysis, and hashing of commonly used passwords.



User	ID	password	h(pass)	ID⊕pass	h(ID⊕pass)
1	a6c26	12ae5	a639d	101001101100 000100101010 101101000110 b 8 6	3e
2	a311f	221ab	44d46	101000110001 001000100001 100000011111 8 1 f	90
3	ed011	22131	a3171	111011010000 001000100001 110011111110 d f e	8c
4	15cab	320f1	7d5ff	000101011100 001100100000 001001111100 2 7 c	af
5	87a1b	775ed	0454b	100001111010 011101110101 111100001111 f 0 f	93
6	1acf6	523c0	9b4fc	000110101100 010100100011 010010001111 4 8 f	ba
...	...	...	...	...	...

X Y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	---	---	---	---	---	---	---	---	---	44d46	---	---	---	---	---	---
1	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
2	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
3	---	---	---	---	---	---	---	---	---	0454b	---	---	---	---	---	---
4	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
5	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
6	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
7	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
8	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
9	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
a	---	---	---	---	---	---	---	---	---	---	---	9b4fc	---	---	---	---
b	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
c	---	---	---	---	---	---	---	---	a3171	---	---	---	---	---	---	---
d	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
e	---	---	---	a639d	---	---	---	---	---	---	---	---	---	---	---	---
f	---	---	---	---	---	---	---	---	---	---	7d5ff	---	---	---	---	---

Fig. 2: Above, a block diagram describe the data flow for authentication. On the left, Passwords and UserIDs are converted into message digests & addresses (h(ID⊕pass)). On the right, a look up table store the message digests for authentication.

3.2 Addressable memory arrays

We are proposing an architecture based on memory arrays to generate one-time-use UserID/Password pairs that are only stored by the client devices, and allowing one-time authentication, see Fig 3.

Password Generation: The server generate a random number RN that generates an address XY to the memory array through the hash function. The hash generates fixed size data streams that have fixed output length regardless of the size of input. They are "image resistant", i.e. any small change in the input creates a new hash digest that is totally different from the original digest, and "collision resistant", i.e. the probability that two different inputs to create the same output is extremely low. The hash message digest points to an address XY in the memory array. The password consists of the encrypted stream extracted at that address, and the random number generated by the server, see fig 3 on the left.

Authentication: A scheme similar to the one used for password generation is shown Fig.3, on the right. For positive authentication, the stream extracted from the memory should be the same as the stream of the password. With hash functions, the probability of having two users at the same location, the "collision", is minuscule but not necessarily zero, which is acceptable because a few User IDs can occasionally share the same password. A new random number, and a therefore a new password can be generated at each cycle.

If we assume that the size of the memory array is 2^{16}x2^{16} , a capacity of 4Gb, a message digest of SHA-3 containing 512 bits can point to 16 different addresses in the array, each defined by 32 bits. If streams of 512 bits are needed for the password, it is possible to extract 32 bits at each of the 16 addresses pointed by the message digest. This protocol is more secure than the protocol described above in section 3.1.

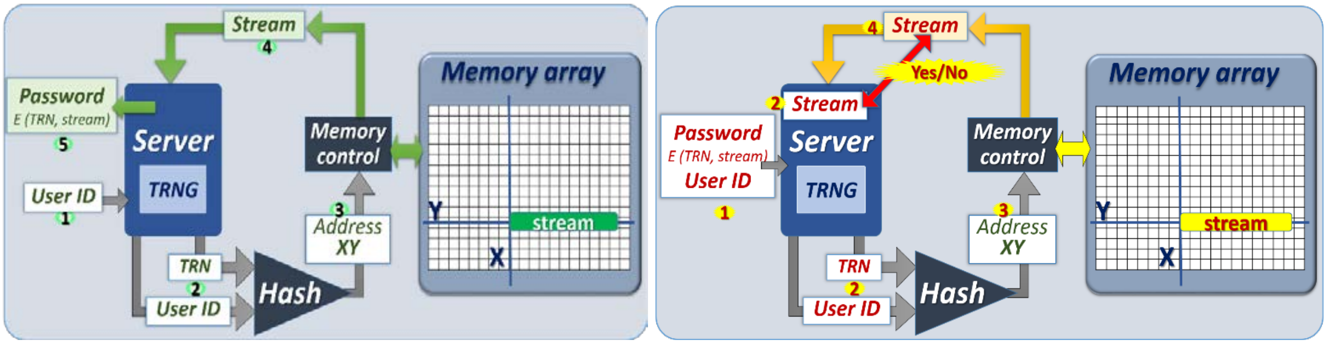


Fig. 3: Block diagram of addressable memory. Left: password generation; right: authentication.

3.3 Example of implementation of addressable memory with ternary states.

We developed a prototype demonstrating the addressable memory scheme of section 3.2 with a window 10 PC driving javacard based secure elements [26]. To enhance entropy we stored three states (-, 0, +) in the memory, with the three states respectively represented by (01), (00), and (10). In Fig.4, we are showing a graphic representation of the extraction of the data streams from the memory. The UserID is XORed with a random number. The resulting stream is feeding a SHA-2 hashing function. The resulting message digest is pointing to 32 different addresses. 16 consecutive trits are extracted at each address to form a stream of 512 trits.The stream of trits is then converted to an hexadecimal character to generate the password. We developed variations of this scheme to enhance entropy. For example [26], we implemented methods to extracts non-consecutive trits at each address that is adjustable when the same address is solicited more than once.

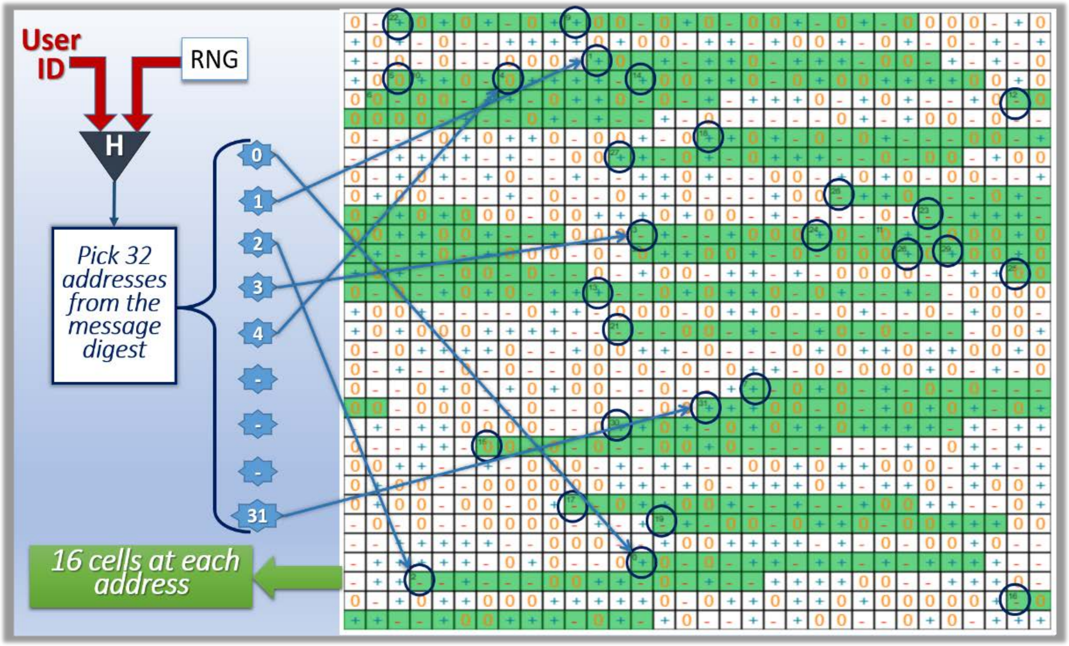


Fig. 4: Example of addressing with a memory containing ternary states.

3.4 Addressable PUF generators (APG) with ternary states.

The risks associated with the loss of the content of the memory array to the enemy are not negligible. A third party knowing the content of the memory could be able to communicate freely with the network of client devices. To mitigate this type of attack, we developed an addressable PUF generator; see the block diagram in Fig. 5. Two distinct operating modes of the APG are important: password generation based on PUF challenge generation, and authentication based on PUF response generation. Unlike a random access memory (RAM) where bits are stored at each cells, a PUFs is based on the analog reading of the cells. This reading varies with the conditions of the read (the instructions), and the relative value of the parameters within the multiple cells that are selected at a particular address. Therefore, as particular cell could be a "0" when part of one group of cell, and a "1" when part of a different group, or when read with different instructions. An hostile party cannot simply read the entire PUF array, and use the reading to communicate with a network of client devices. The cloning of the entire array would be a security threat, however, the level of randomness of such a component is so high that this type of attack is highly unlikely. The use of ternary PUFs to reduce the challenge-response-pair (CRP) error rates [27-29]. During challenge generation three types of cells are identified: the cells that are clearly "0" or "1" in a consistent manner, and the cells that are unstable, switching randomly between "0" and"1". During response generation, the unstable cells are masked; the CRP error rate of the remaining cells is very low. We this method, we previously reported CRP error rates in the part per million range (ppm), with TaO based Resistive RAM cells [27].

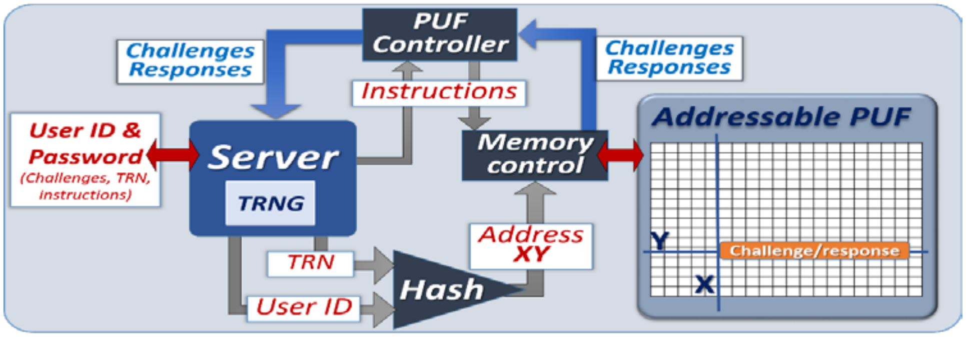


Fig. 5: Block diagram of APG.

3.5 Bandwidth enhancement and cryptographic protocol

The suggested architecture to increase the bandwidth of the architecture is shown in Fig. 6. In this architecture, a router, with an X-Y demux circuitry, drives an array of Host/APG subsystems, as described in Fig 3 to 5. The router directs the User ID – PW pair to a particular Host/APG system based on the first character of the User ID. For example, if the User ID is starting with a "C", the User ID/PW pair is routed to the third column, and first row of the networked array. This architecture is scalable to large numbers of systems, with bandwidth increasing proportionally with the number of Host/APG sub-systems. The router used in this design can be similar to routers that connecting millions of users in wired telecommunication applications handling IP addresses. Fig. 7 depicts how a PKI protocol similar to the one presented Fig 1 can incorporate APG for authentication. The transfer of passwords between the clients and the host is encrypted with public keys, thereby reducing the risks of being hacked while exposed during the communication.

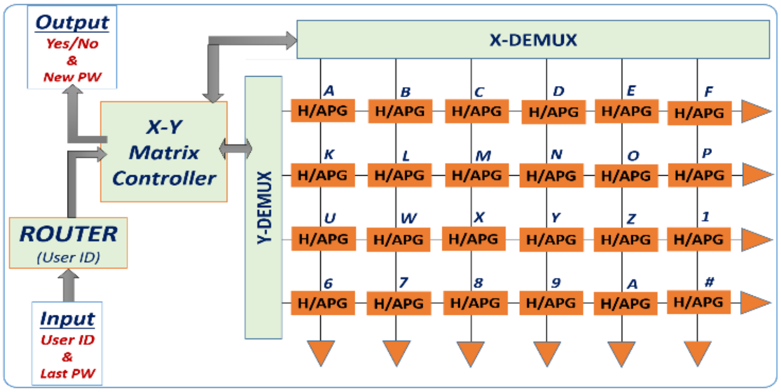


Fig.6: High bandwidth architecture.

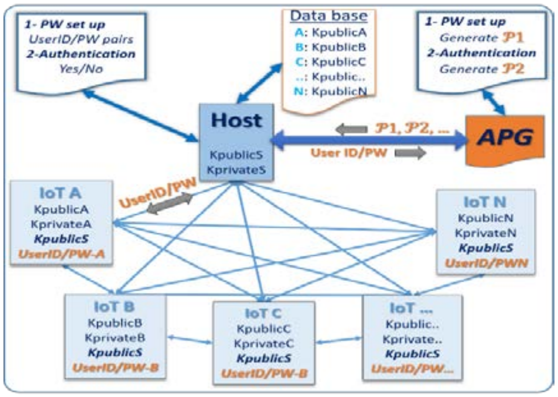


Fig.7: Cryptographic protocol with APG.

4. CONCLUSION

We presented how arrays of addressable PUFs are used to design database-free Password Management Systems. Hash functions convert userID's and TRNG's into addresses generating CRPs for password generation and authentication. Since hash functions are one-way functions, it is impossible to deduce the input of the hash function by looking at the address of a PUF array. Unlike traditional data storage units, the memory arrays used in the APG do not store information; a crypto-analyst cannot extract CRPs from these PUFs without knowing the necessary instructions. The masking of the unstable cells with ternary states increases the quality of the PUF, and reduces the CRP error rates: only the cells with solid "0" and "1" tested during challenge generation are kept to generate PUF responses. To counter the replay attack, each password is used only once, and this leveraging the ease to generate new passwords from the APG. We are integrating this architecture with PKI to protect the communication between the host server, and the client devices.

REFERENCES

[1] C. Paar, J. Pezl; Understanding Cryptography- A text book for students and practitioners; *Spinger editions, 2011*;

[2] H.X. Mel, D. Baker; Cryptography Decrypted; *Addison-Wesley editions, 2001;*

[3] C. P. Pfleeger, and al; Security in Computing; *Fifth edition; Prentice Hall editions, 2015;*

[4] Y. Jin; Introduction to hardware security*; Electronics 2015, 4, 763-784; doi:10.3390/electronics4040763*;

[5] Z. Gong, M. X. Makkes; Hardware Trojan Side-Channels Based on PUF; *Information Security, Volume 6633, Notes in Computer Science pp 294-303*; 2011;

[6] D. Naccache and P. Frémanteau; Unforgeable identification device, identification device reader and method of identification; *Patent US5434917; Aug. 1992;*

[7] M. Delavor, and all; PUF based solution for secure communication in advanced metering infrastructure; *ACR publication, 2014;*

[8] C. Herder and all; Physical Unclonable Functions and Applications; A Tutorial; *Proceedings of the IEEE 102, no. 8 (2014);*

[9] R. Pappu, B. Recht, J. Taylor, and N. Gershenfield; Physical one-way functions; *Science. Vol 297 No5589 pp2026-2030;* 20 Sept 2002;

[10] B. Gassend, and all; Silicon Physical Randomness; *Proceedings of the 9th ACM conference on Computer and communications security, Pages 148-160, CCS'2002;*

[11] B.Cambou, P. Flikkema, and C. Ciocanel; Securing UAVs; *IEEE workshop on IoT security, February 2016;*

[12] S. Katzenbeisser, and all; PUFs: myths, fact or busted? *A* security evaluation of PUFs cast in silicon; *CHES 2012*;

[13] M. Hiller, M. Weiner, L. Rodrigues, M. Birkner and G. and Sigl; Breaking through Fixed PUF Block Limitations with DSC and Convolutional Codes; *in TrustED'13, 2013;*

[14] D. Merli, F. Stumpf, G. Sigl; Protecting PUF Error Correction by Codeword Masking; *IACR Cryptography, e-print archive 2013: 334; 2013;*

[15] Y. Gao, and all; Emerging Physical Unclonable Functions with nanotechnologies; *IEEE, DOI:10.1109/ACCESS.2015.2503432;*

[16] C. Krutzik; Jan 2015; Solid state drive Physical Unclonable Function erase verification device and method; *US Patent Application publication US 2015/0007337 A1;*

[17] J. Plusquellic, and all; Systems and methods for generating PUF's from non-volatile cells; *WO20151056887A1; 2015;*

[18] D. E. Holcomb, W. P. Burleson, K. Fu; Power-up SRAM state as an Identifying Fingerprint and Source of TRN; *IEEE Trans. on Comp., vol 57, No 11; Nov 2008*;

[19] R. Maes, P. Tuyls and I. Verbauwhede,; A Soft Decision Helper Data Algorithm for SRAM PUFs; *IEEE International Symposium on Information Theory*, 2009;

[20] T. A. Christensen, J. E Sheets II; Implementing PUF utilizing EDRAM memory cell capacitance variation*; Patent No.: US 8,300,450 B2; Oct. 30, 2012*;

[21] P. Prabhu, A. Akel, L. M. Grupp, W-K S. Yu, G. E. Suh, E. Kan, and S. Swanson; Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations; *4th international conference on Trust and trustworthy computing; June 2011;*

[22] A. Chen; Comprehensive Assessment of RRAM-based PUF for Hardware Security Applications; *978-1-4673-9894-7/15/IEDM IEEE; 2015;*

[23] X. Zhu, S. Millendorf, X. Guo, D. M. Jacobson, K. Lee, S. H. Kang, M. M. Nowak, D. Fazla; PUFs based on resistivity of MRAM magnetic tunnel junctions; *Patents. US 2015/0071432 A1*; *March 2015;*

[24] E. I. Vatajelu, G. Di Natale, M. Barbareschi, L. Torres, M. Indaco, and P. Prinetto; STT-MRAM-Based PUF Architecture exploiting Magnetic Tunnel Junction Fabrication-Induced Variability; *ACM trans.*; *July 2015;*

[25] A. Gupta; Implementing Generic BIST for testing Kilo-Bit Memories; *Master Thesis No-6030402 Deemed University Patiala India; May 2005;*

[26] B.Cambou, P. Flikkema, J. Palmer, D. Telesca, C. Philabaum; Can Ternary Computing Improve Information Assurance?, Cryptography, MDPI, Feb 2018;

[27] B.Cambou, and M. Orlowski; Design of PUFs with ReRAM and ternary states; *CISR 2016, April 2016;*

[28] B. Cambou, F. Afghah; Physically Unclonable Functions with Multi-states and Machine Learning", *14th International Workshop on Cryptographic Architectures Embedded in Logic Devices (CryptArchi), France; 2016;*

[29] B. Cambou; Physically Unlonable Function generating systems and related methods; *US patent disclosure No: 62/204912; Aug 2015;*