Lightweight PUF-based Continuous Authentication Protocol

Konstantinos Goutsos
μSystems Group, School of Engineering
Newcastle University
Newcastle upon Tyne, United Kingdom
k.goutsos1@ncl.ac.uk

Alex Bystrov
μSystems Group, School of Engineering
Newcastle University
Newcastle upon Tyne, United Kingdom
a.bystrov@ncl.ac.uk

***Abstract*—Given the recent rise of the Internet-of-Things (IoT), networked devices are becoming deeply embedded into everyday objects, leading to a need for novel security methods. Physical Unclonable Functions (PUFs) enable the differentiation between instances of the same device and have the potential to replace costly cryptographic operations while providing higher security guarantees, due to their inherent unclonability.**

We present a pairwise, continuous authentication protocol based on Physical Unclonable Functions (PUFs) and supporting mutual authentication on resource constrained nodes. The unclonability provided by the PUFs is an integral part of the authentication process to continuously prove the existence of the PUF secrets and the proposed protocol is executed periodically to enable the establishment of trust between the participants. This is achieved by refreshing the authentication information in every protocol round, leading to a 'CRP Ratchet' mechanism of renewing the authenticating PUF challenge response pairs (CRPs).

We also discuss the security and performance of the protocol in IoT applications with a large number of devices. Since the only operations used in the periodic protocol phase are hashing and exclusive OR, low computation, complexity, and energy consumption overhead is achieved.

***Index Terms*—authentication, physical unclonable functions, security protocols, unclonability, internet of things**

I. Introduction

Due to the difficulty in differentiating between instances of the same hardware, traditional paradigms of network security treat physically distinct devices as identical. Yet, this approach is a limiting factor in the development of novel security methods for resource-constrained, networked devices, especially in the context of the Internet-of-Things (IoT). Devices are deeply embedded in a multitude of objects, and thus vulnerable to physical attacks as well as operator compromise. Furthermore, the increasing number of devices makes it challenging to individually manage their digital secrets. As a result, traditional methods of manually embedding static secrets on devices are no longer satisfying the security requirements of the new digital era, as they expose the secrets to human operators, and do not provide adequate updating mechanisms for these secrets.

Recently, methods for differentiating between device instances have been developed, building upon the invention of Physical Unclonable Functions (PUFs). PUFs are exceptionally useful in IoT applications since they are low cost, easy to integrate into existing designs, and have the potential to replace expensive cryptographic operations.

While the majority of work around PUFs is in the context of integrating them into existing security protocols and methods, we see PUFs as a practical enabler for the *unclonability* primitive and its inclusion in networked systems. We envision a vertical integration of the primitive of unclonability that we systematise in the form of an *unclonability stack*, similarly to the OSI model. The stack, illustrated in Fig. 1, defines operational layers from the unclonable core through to system level interactions and applications built on them. Each layer involves a number of methods aiming to integrate unclonability in their respective domain, securing the relationships between participating entities by detecting topology distortions.

In our generic scenario, the nodes comprising the system are deployed, introduced, configured and otherwise initialised with the aim of operating autonomously and possibly unattended. This configuration is done via one or more *authority devices (ADs)*: mobile, unclonable tokens enabling the secure initialisation and management of the system. In [1] we described the operation and features of such devices in the context of an 'authority device scheme' for networked nodes. The scheme provides methods for common network security provisions but, due to its reliance on static, public keys, it is designed for infrequent use and does not support the creation of *unclonable pairwise links*.

Thus, we present a protocol for PUF-based continuous authentication that does not present those drawbacks. The proposed protocol aims to serve as a method for detect nodes that have been removed, replaced, moved, or newly introduced to the system and leads to the detection of distortions to the system topology.

A. Our Contribution

The proposed solution is a lightweight pairwise mutual authentication protocol based on PUFs. Our protocol is designed to take place periodically between pairs of nodes belonging a larger group or *'neighbourhood'* organised by a group management scheme similar to [1].

The novelty of the protocol is summarised in the following:

- The overarching goal is to establish a chain of trust via refreshing the authentication information in every round. We combine the primitives of 'ratcheting'[2], [3] and PUFs, providing break-in recovery and continuous renewal of unclonability. We thus refer to our protocol as a 'Challenge-Response Pair (CRP) Ratchet'.

This work is partially supported by the School of Engineering, Newcastle University, and EPSRC through the EPSRC Doctoral Training Partnerships (DTP) programme.

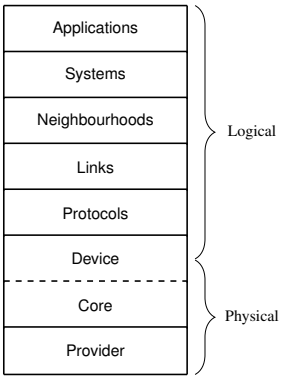

Fig. 1: Unclonability Stack

- We do not assume that either of the nodes has more resources than its counterpart.
- No third party is involved in the authentication process after its initialisation, reducing the attack surface and improving scalability.
- Only one CRP from each device is exchanged and stored in every round. No CRP database or PUF model is stored on the verifier, disallowing impersonation and improving scalability.
- An authority device is required to start the protocol or reset it after the detection of a potential compromise.

Due to the above properties, the proposed protocol creates *unclonable links*, where modifications to the involved nodes directly lead to a change in the state of the link between them. This is in contrast to traditional pairwise communication where the identity of the endpoints has a weaker representation at the link level.

The rest of the paper is organised as follows: Sections I-B and I-C introduce the concept of unclonability and how it is realised in practice with PUFs. Section I-D briefly summarises existing work on PUF-based authentication protocols. Section II describes the CRP Ratchet principles and operation, and Section III discusses the security and performance of the protocol. We conclude with Section IV.

B. Unclonability

Clones of physical objects can be mathematical where the cloned object only replicates the input-output behaviour of the original object, and physical where the clone has an identical physical structure to the original object. Given the difficulty in achieving perfect cloning results in practice, the task of cloning is defined by the capabilities of clone detection. To enable this detection we draw on features which are inherent to the object and beyond any level of control that would allow their exact reproduction, given the available technology over the lifetime of the object, making them *individualising features*[4].

Unclonability refers to the difficulty in controlling all these individualising features with the aim of producing a clone that appears to be an exact copy of the original object. In order to be exploited these features are required to be: measurable, stable over the lifetime of the object, and sufficiently differentiating between instances of the same object.

Unclonability is also closely related to ownership, and thus authority of the owner over an object. At the same time, the value of objects is often determined by their scarcity which means that cloning an object may directly diminish its value. The combination of these issues with the modern practice of online interaction, leads to the need to unquestionably prove one's identity while keeping it safe. Despite the inherent unclonability of humans however, it is difficult and intrusive to measure their individualising features directly and digital devices have to be employed as proxies of authority (authority devices) to bridge that gap. On a conceptual level, we aim to provide methods to enable this unification by injecting unclonability in networking protocols which are already being increasingly used in the place of human interactions.

C. Physical Unclonable Functions

Physical Unclonable Functions (PUFs) are the most prominent method of extracting individualising features in a stable and efficient manner. Initially introduced by Pappu[5], PUFs make use of intrinsic variations present in the fabrication process of hardware components, to generate unique outputs. There has a been growing academic and commercial interest around PUFs in the recent years, leading to a rapid development of the field and electronic implementations have been the target of most of the existing work, since they are easily integrated into existing systems. Notable constructions include SRAM[6], arbiter[7], and ring oscillator[8] PUFs.

In essence, electronic physically unclonable constructions create an additional security layer between the hardware and the state it represents. This allows us to relax the security requirements at a hardware level, since letting an adversary observe the entirety of the hardware gives her no advantage in predicting or replicating the state that this hardware will produce when it is powered on.

PUFs have a number of promising properties, including unclonability, unpredictability, uniqueness, reproducibility, and tamper-evidence[4]. At a high level, PUFs accept an input, commonly referred to as a 'challenge' and produce an output, referred to as a 'response'. The relationship between the challenge and the response is determined by the hardware variations of the PUF, producing responses unique to both the challenge and the specific instantiation of the PUF. PUFs also exhibit a degree of instability and, since no errors can be tolerated in authentication scenarios, a considerable amount of research has targeted error correction methods for PUFs[4], [9], [10].

The above properties allow PUFs to serve as an unclonable core in the unclonability stack discussed above. For the remainder of this paper, we model PUFs as a mapping receiving challenges and producing responses which are both in the form of binary strings:

$$\tau : C \rightarrow R : \tau(c) = r \quad with \quad c \in \mathbb{Z}_n^+,\ r \in \mathbb{Z}_n^+ \qquad (1)$$

D. Related Work

The strong advantages of using PUFs for entity authentication have sparked a great number of related protocols. However, to the best of our knowledge, the proposed solution is the first one to address the scenario discussed above. In this section, we briefly discuss previous work on PUF-based authentication, referencing only representative examples due to the multiplicity of different variations.

Many of the existing protocols rely on the creation of a CRP database or a model of the PUF on the verifier[8], [11], [12].

We think that these requirements negate the biggest advantage of unclonability. Furthermore, storing multiple CRPs on the verifier involves storage costs which are unattainable for IoT nodes with a large number of neighbours. Similarly, protocols involving costly operations similar to public key cryptography[13] are not suitable for our intended use case.

Based on this asymmetry of resources between the prover and the verifier, a different group of protocols was later introduced, sparked by [10], with the aim of reducing the calculation overhead on the prover and relocating the burden to the verifier. While they provide significant performance gains for peers with highly asymmetric resources these solutions fail to address the issues of CRP databases and in many cases introduce additional cost on the verifier side which is not advisable in our scenario[11], [14].

Additionally, the majority of work discussed above includes the generation of one or more secrets based on the PUF which remain static after their generation[8], [10], [11]. In our view, this approach does not exploit the full unclonability potential as the PUFs are not constantly involved in the protocol but rather serve as an unclonable random number generator. On the other hand, proposals that aim to renew the authentication secrets often require several CRPs to be exchanged in each protocol round and do not include information from both participating entities, thus not achieving truly unclonable links and leading to faster exhaustion of the PUF responses[15], [16].

Finally, a different class of solutions aim to employ PUFs in novel ways. For example, SIMPL Systems[17] or commutative PUFs[18] provide a new perspective on unclonability protocols that looks promising. Unfortunately, these constructions have not yet been implemented in practice.

II. PUF CRP Ratchet

The *Challenge Response Pair (CRP) Ratchet* uses an authentication mechanism involving the PUF as a response generator. In each ratchet step, both nodes use information from the previous step to authenticate each other. They subsequently combine parts of their PUF state to prepare their state for the next step. The protocol aims to achieve its security goals while retaining a low computation and energy footprint and thus expensive cryptographic operations are avoided.

A. Scenario and Assumptions

The proposed protocol operates in the context of a 'neighbourhood' of nodes. All neighbourhood members have been enrolled by an AD and have exchanged public keys with each other and with the AD prior to the start of the CRP Ratchet. Nodes are assumed to have access to a wired or wireless communication channel with each other and the AD can occasionally establish its own channels with the nodes. The holder of the AD is also assumed to have physical access to the node with which the AD is communicating.

In the context of this paper, PUFs are considered a component providing the behaviour discussed in Section I-C with responses that are reliably reproduced, error-corrected, and entropy-enhanced at the hardware level, since these issues have been extensively studied in literature[4], [10], [19]. Namely, the PUF component can be modelled as an augmented hash function with outputs that are uniformly random based on the internal PUF state and the corresponding challenge.

TABLE I: Summary of symbols

Symbol	Value or Operation
AD	Authority device
CNT_x	Monotonic counter for x
ACK	Acknowledgement
P_x	Public key of x
S_x	Private key of x
$SIG_x(y)$	Signature of x with private key y
$VER_x(y,z)$	Verification of signature z of y with public key x
$ENC_x(y)$	Public key encryption of y with secret key x
$DEC_x(y)$	Public key decryption of y with public key x
$HM_x(y)$	Calculation of the HMAC of y with secret key x
$HV_x(y,z)$	Verification of the HMAC z of y with secret key x
$PUF_x(y)$	Evaluation of the PUF of x with challenge y
$TRNG_x()$	Evaluation of the True Random Number Generator of x
$\oplus$	Bitwise exclusive OR
$\|\|$	Concatenation

We further assume that the PUF chip is included in a cryptographic core similar to the one we outlined in [1] that executes protocol operations and only exposes their results thus protecting the PUF and any other secrets from direct access. In essence, the only information leaving this core is the data transmitted to peers. Finally, the cryptographic core has access to a true random number generator (TRNG) which can be implemented by means of another PUF chip[20].

B. Symbols and Definitions

In our descriptions we assume that every device keeps track of the protocol progress and will reject unexpected requests. Additionally, we make use of the symbols of Table I.

We also define two security parameters: the step interval t_s represents the time period between two successive, successful steps of the ratchet, and the failure threshold t_f represents the number of authentication failures after which a node is assumed to be compromised. The t_f parameter can be measured in time units or in number of successive retries. These parameters aim to capture the realities of practical applications where transient faults are possible (e.g. networking issues) without being the result of malicious actions.

When the threshold is exceeded, the detecting node deletes the ratchet state concerning the peer which is now in an unverified state, and notifies the higher layers of the stack, making the neighbourhood aware of a security incident. The suspected node is now marked as untrusted and authority action has to be taken to restart the ratchet protocol. With this design we signify that a human operator would have to manually inspect the node in question and, after clearing the threat, reset the system to its normal state using an authority device (AD).

C. Initialisation

This is a 'bootstrapping' phase introducing the nodes to each other in preparation for the periodic ratchet phase. It also allows the corresponding AD to approve the initialisation of the protocol. Thus, this phase is required in two cases: (a) when the nodes are first introduced and (b) after a protocol failure which means that the threshold of acceptable authentication failures has been exceeded. Since the nodes have not yet been introduced, a temporary secure channel is established by

means of public cryptography, with the previously exchanged public keys, and parties use a monotonic initialisation counter to prevent reply attacks. The Initialisation Phase takes place as follows:

Protocol 1 (Ratchet Initialisation)**.** *Nodes A and B have been enrolled into a neighbourhood with AD X. At the end of the protocol, both nodes have established a state that will be used in subsequent Ratchet Step phases (Protocol 2). See Fig. 2 for the detailed interactions.*

1) *X generates a random authorisation token T_{AB} and signs it including the counter.*
2) *X encrypts the token with the public keys of A and B separately.*
3) *X sends the corresponding encrypted token and the signature to A and B.*
4) *A and B decrypt the token, verify the signature and reply with an acknowledgement or abort accordingly.*
5) *X, A and B increment their initialisation counters.*
6) *A initiates the initialisation process by authenticating B with a public key authentication process (e.g. [1]).*
7) *A generates a random PUF challenge C_B^0 and signs it the including the authorisation token.*
8) *A sends the challenge and its signature to B.*
9) *B verifies the signature and aborts on failure.*
10) *B uses the challenge to produce a PUF response $R_B^0 = PUF_B(C_B^0)$.*
11) *B encrypts the response with the public key of A.*
12) *B generates a random PUF challenge C_A^0 and signs $C_A^0 \| R_B^0$ including the authorisation token.*
13) *B sends the encrypted response, the challenge, and the signature to A.*
14) *A verifies the signature and aborts on failure.*
15) *A decrypts the response of B and derives $K_A^0 = R_B^0 \oplus C_A^0$.*
16) *A generates a PUF response with the received challenge C_A^0: $R_A^0 = PUF_A(C_A^0)$.*
17) *A encrypts the PUF response and signs it, including the authorisation token.*
18) *A sends the encrypted response and the signature to B.*
19) *B verifies the signature and aborts on failure.*
20) *B decrypts the response of A and derives $K_B^0 = R_A^0 \oplus C_B^0$.*
21) *B stores the initial state K_B^0 and C_A^0 and replies with an acknowledgement.*
22) *A stores the initial state K_A^0 and C_B^0.*
23) *Both nodes delete the authorisation token. They also discard any intermediate information used in this phase.*

D. Ratchet Step

The Ratchet Step phase serves a dual purpose: authenticating the remote node, and refreshing the secrets used for authentication. This phase is repeated continuously with the step interval t_s defined above. In the context of this phase, the nodes make use of one key each, which we refer to as 'ratchet key' and a common key which we refer to as 'round key'. The Ratchet Step phase for round $j, j \in \mathbb{Z}^+$ takes place as follows:

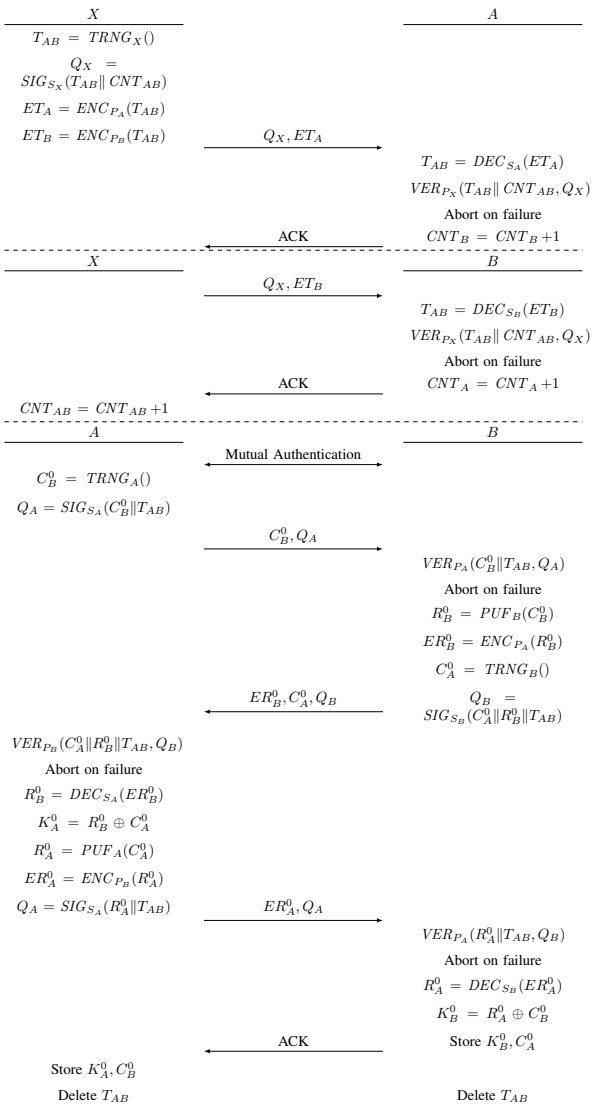

Fig. 2: Ratchet Initialisation

Protocol 2 (Ratchet Step)**.** *Nodes A and B, performing the j-th ratchet step with A as the initiator. At the end of this phase, the nodes have mutually identified each other and the necessary information to enable the next iteration. See Fig. 3 for the detailed interactions.*

1) *Node A has the state information K_A^{j-1} and C_B^{j-1}. Node B has the state information K_B^{j-1} and C_A^{j-1}.*
2) *A generates a random PUF challenge C_B^j and calculates the HMAC tag of C_B^j and C_B^{j-1} with its ratchet key K_A^{j-1}.*
3) *A sends the challenges and the tag to B.*
4) *B derives the ratchet key of A as $K_A^{j-1} = PUF_B(C_B^{j-1}) \oplus C_A^{j-1}$ and validates the received HMAC tag. If there is a mismatch it aborts. If the failure threshold is reached, the failure procedure is followed.*
5) *B derives the round key $K^j = K_A^{j-1} \oplus K_B^{j-1}$.*
6) *B generates the PUF response $R_B^j = PUF_B(C_B^j)$ and a random PUF challenge C_A^j.*
7) *B encrypts the PUF response $ER_B^j = R_B^j \oplus K^j$ and calculates the HMAC tag of $C_A^{j-1} \| C_A^j \| R_B^j$ with the round key.*
8) *B sends the two challenges, the encrypted PUF response*

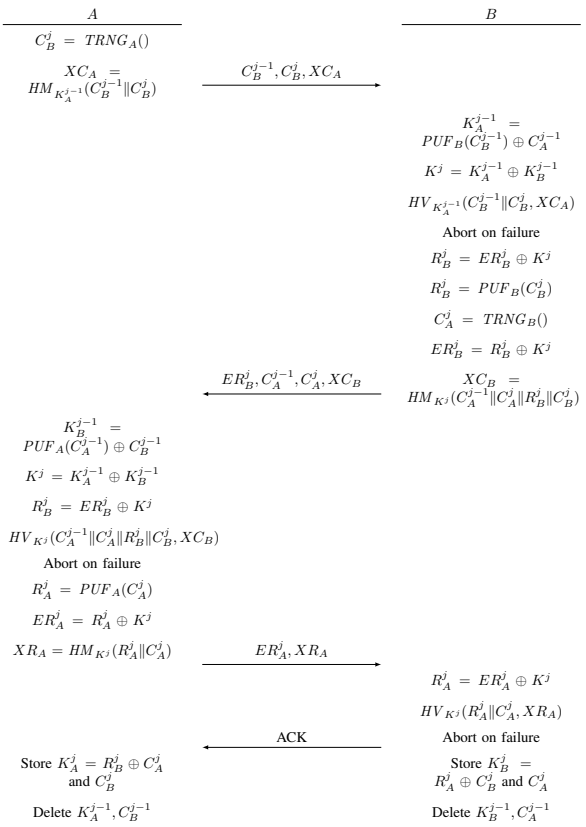

Fig. 3: Ratchet Step

and the HMAC tag to A.

9) *A derives the ratchet key of B as* $K_B^{j-1} = PUF_A(C_A^{j-1}) \oplus C_B^{j-1}$ *and the round key* $K^j = K_A^{j-1} \oplus K_B^{j-1}$ *and decrypts the PUF response of B.*
10) *A validates the HMAC tag. If there is a mismatch it aborts. If the failure threshold is reached, the failure procedure is followed.*
11) *A generates the PUF response* $R_A^j = PUF_A(C_A^j)$*, encrypts it with the round key and calculates the HMAC tag of the PUF response with the round key.*
12) *A sends the encrypted response and the tag to B.*
13) *B decrypts the response and validates the HMAC tag. If there is a mismatch it aborts. If the failure threshold is reached, the failure procedure is followed.*
14) *B replies with an acknowledgement and stores* $K_B^j = R_A^j \oplus C_B^j$ *and* C_A^j*.*
15) *A stores* $K_A^j = R_B^j \oplus C_A^j$ *and* C_B^j*.*
16) *Both nodes delete the previous ratchet keys and challenges. They also discard any intermediate information used in this phase.*

III. Analysis

A. Security

In this section we discuss the security of the proposed protocol, focusing our analysis on the Step phase since it occupies the majority of the protocol runtime. Our analysis is based on a software proof-of-concept that was developed in Python using an emulated SRAM PUF.

We use an adversary model based on the Dolev-Yao model[21], namely allowing the observation and recording of transmitted messages, and the decryption of ciphertexts if the key becomes compromised. The adversary is also able to access the memory of the devices but does not have direct access to the PUF or to protocol state while the latter is executed. Thus it is expected that hardware provisions have been taken for the aforementioned access control.

TABLE II: Step Phase Operations

Operation	Executions on A	Executions on B
TRNG	1	1
PUF	2	2
XOR	5	6
HMAC	3	3

1) XOR encryption security: The encryption of the PUF responses is based on XORing them with the round key. Therefore, to ensure the security of the encryption, the PUF challenges and round keys are required to have the same length. Since the round keys are in turn a combination of PUF challenges and responses, essentially the requirement is that PUF challenges and PUF responses have the same length which is true in our design. In addition both the keys (round keys) and the plain texts (PUF responses) are random and are refreshed in every round, preventing *chosen plaintext* attacks. Thus, we conclude that the security of the XOR encryption is ensured.

2) Message entropy: Three kinds of information are exchanged between the nodes: PUF challenges, encrypted PUF responses, and HMAC tags of both. Given that challenges and responses are generated uniformly at random by the TRNGs and the PUFs respectively, and due to the entropy-preserving nature of HMAC, we conclude the messages of the Ratchet phase appears fully random to an eavesdropping adversary.

3) Replay attacks: The round key of the Step phase is derived from a combination of the state of both participants and is used as the HMAC secret. This directly prevents replay attacks in the ratchet steps since the HMAC verification for older messages will fail. In the communication with the AD during the Initialisation phase, a monotonic counter is used to prevent similar attacks.

4) PUF responses: It is vital for the PUF responses to have a certain length that will prohibit simple brute-force attacks and allow for a large number of responses without repetition. For the proposed protocol, the ratio of the expected lifetime of the system over the step interval t_s determine the required response length. In most practical applications a length that is over 32 bits is deemed sufficient. For example, a response length of 32 bits and a step interval of 100ms would support over *13 years* of continued protocol operation a time period which exceeds the average lifetime of modern electronic systems. Nevertheless, certain PUF constructions are unable to provide responses of the required length, and thus additional methods are needed to improve the number of available responses and extract the required entropy.

5) Node compromise: Every protocol exchange is signed and/or encrypted with a ratchet key or the round key. Furthermore, these keys are derived locally and are protected from invasive attacks. Thus we conclude that the protocol provides *break-in recovery* since an adversary who compromises one of the nodes for a limited time does not gain any advantage over the protocol once her access has been removed.

B. Performance

1) Calculations: Table II summarises the operations involved in a single ratchet step. Regardless of the hardware specifics, a single evaluation of the TRNG or the PUF has the cost of a memory access or an I/O read operation. In addition, XOR and HMAC are efficiently computed in modern processors and can be accelerated further with hardware implementations. The Initialisation phase makes use of symmetric cryptography which is relatively more computationally expensive but can nevertheless also be accelerated and is designed to be used very infrequently.

2) Storage and Message size: Each node is required to store only one challenge and one ratchet key between Step phases, with their size determined directly by the PUF response length which was discussed above. Similarly, the size of the exchanged messages and the corresponding network overhead is only affected by the same response length since the HMAC output size does not depend on its inputs. In our proof-of-concept implementation with 64-bit PUF responses and 256-bit HMACs the maximum message length was 56 bytes.

3) Step interval and Failure threshold: The t_s and t_f parameters also have an important effect on the overhead of the proposed protocol. The failure threshold t_f needs to account for application-dependent issues including network latency, dropped packets etc. Conversely, while the minimum step interval t_s is mainly determined by the maximum throughput of the protocol implementation, the maximum step interval is determined by the human factor. Conceptually, the Step phase is required to be repeated at a rate that is high enough to prevent an adversary from accessing and modifying the participating nodes. Thus, t_s can range from milliseconds to a few seconds or even minutes, depending on the particular deployment, greatly reducing the overhead without necessarily harming the overall security of the system.

IV. Conclusions

We have presented *CRP Ratchet*, a lightweight pairwise protocol providing mutual authentication based on Physical Unclonable Functions. We detailed the operation of the protocol and provided a short analysis of its security and performance in IoT scenarios. Our protocol's strength lies in the renewal of the authentication secrets through combining parts of the secrets of the participating entities. This feature creates *unclonable links* between nodes and supports *break-in recovery*.

Future improvements to the proposed protocol include a formal security proof, performance optimisations, and a hardware proof-of-concept to complement the software implementation. Additionally, we are currently working on variants of the protocol for different adversary models and with different hardware assumptions, to enable the inclusion of the proposed 'unclonability stack' in a variety of scenarios.

Acknowledgment

We would like to thank the anonymous reviewers for their suggestions.

References

[1] K. Goutsos, "PUF-Based Authority Device Scheme," Newcastle University, Tech. Rep., 2019. [Online]. Available: http://async.org.uk/tech-reports/NCL-EEE-MICRO-TR-2019-212.pdf.

[2] M. Abdalla and M. Bellare, "Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques," in *Advances in Cryptology — ASIACRYPT 2000*, 2000, pp. 546–559.

[3] M. Marlinspike and T. Perrin, "The Double Ratchet Algorithm," Tech. Rep. [Online]. Available: https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf.

[4] R. Maes, *Physically unclonable functions: Constructions, properties and applications*. Springer Berlin Heidelberg, 2013, pp. 1–185, ISBN: 9783642413957.

[5] R. Pappu, "Physical One-Way Functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[6] J. Guajardo, B. Škorić, P. Tuyls, S. S. Kumar, T. Bel, A. H. M. Blom, and G. J. Schrijen, "Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions," *Information Systems Frontiers*, vol. 11, no. 1, pp. 19–41, 2009.

[7] J. Lee, D. L. D. Lim, B. Gassend, G. Suh, M. V. Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," *Symposium on VLSI Circuits*, pp. 176–179, 2004.

[8] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *44th ACM/IEEE Design Automation Conference*, 2007, pp. 9–14.

[9] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008.

[10] A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A. R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs," in *Lect. Notes Comput Sc.*, vol. 7397, 2012, pp. 374–389.

[11] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas, "Robust and Reverse-Engineering Resilient PUF Authentication and Key-Exchange by Substring Matching," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 1, pp. 37–49, 2014.

[12] C. Huth, J. Zibuschka, P. Duplys, and T. Güneysu, "Securing systems on the Internet of Things via physical properties of devices and communications," in *SysCon 2015*, 2015, pp. 8–13.

[13] S. Kerr, M. S. Kirkpatrick, and E. Bertino, "PEAR," in *3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, 2010, p. 18.

[14] M. Barbareschi, A. De Benedictis, and N. Mazzocca, "A PUF-based hardware mutual authentication protocol," *Journal of Parallel and Distributed Computing*, vol. 119, pp. 107–120, 2018.

[15] M. N. Aman, K. C. Chua, and B. Sikdar, "Physical Unclonable Functions for IoT Security," in *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security - IoTPTS '16*, 2016, pp. 10–13.

[16] M. H. Mahalat, S. Saha, A. Mondal, and B. Sen, "A PUF based Light Weight Protocol for Secure WiFi Authentication of IoT devices," in *2018 8th International Symposium on Embedded Computing and System Design (ISED)*, 2018, pp. 183–187.

[17] U. Rührmair, "SIMPL Systems as a Keyless Cryptographic and Security Primitive," in *Lect. Notes Comput Sc.* Vol. 6805 LNCS, 2012, pp. 329–354.

[18] Y. Guo, T. Dee, and A. Tyagi, "Barrel Shifter Physical Unclonable Function Based Encryption," *Cryptography*, vol. 2, no. 3, p. 22, 2018.

[19] C. Herder, B. Fuller, M. van Dijk, and S. Devadas, "Public Key Cryptosystems with Noisy Secret Keys," *IACR Cryptology ePrint Archive*, 2017.

[20] E. Leobandung, *SRAM as Random Number Generator*, 2017. [Online]. Available: https://patents.google.com/patent/US20190182054A1/en.

[21] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, 1983.