BARRETT BlockchAin Regulated REmote aTTestation

Michail Bampatsikos
Institute of Informatics and Telecommunications
National Centre for Scientific Research Demokritos, Greece,
m.bampatsikos@iit.demokritos.gr

Christoforos Ntantogian
Department of Digital Systems
University of Piraeus, Piraeus,
Attica, Greece, dadoyan@unipi.gr

Christos Xenakis
Department of Digital Systems
University of Piraeus, Piraeus,
Attica, Greece, xenakis@unipi.gr

Stelios C.A. Thomopoulos
Institute of Informatics and Telecommunications
National Centre for Scientific Research Demokritos, Greece, scat@iit.demokritos.gr

ABSTRACT

Today, an increasing number of Internet of Things (IoT) healthcare devices, crucial to a person's wellbeing and life, connects to the internet and consequently is exposed to a variety of threats. These devices possess low computational resources, and as a result they cannot use security tools such as antivirus or firewalls. Consequently, they become easy targets for cyber-attacks and malware infection, thus putting a person's life at risk. One way to protect these devices from malware infection is Remote Attestation (RA), a process by which a device with low computational power (prover) verifies its internal state to a party with higher computational resources (verifier) upon the latter's request. However, in case the verifier is malicious, it may constantly send numerous requests for RA to a prover to prevent it from performing the functions it was designed for. Thus, keeping it busy and rendering it unusable to its legit users as well as services. In short, the verifier performs a Computational Denial of Service (CDoS) attack against the prover. This paper proposes the BARRETT architecture which uses a Public Ethereum Network (PEN) in conjunction with an RA protocol to protect the prover from CDoS attacks. In particular, the PEN in BARRETT deters CDoS by forcing the verifier to pay a fee in Ether cryptocurrency every time they wish to send an Attestation Request (AR) to a prover. The verifier pays the fee since in BARRETT it can send the AR only via Ethereum transactions. Consequently, any attempt to perform a CDoS becomes prohibitively expensive.

CCS CONCEPTS

• Social and professional topics~Health information exchanges

KEYWORDS

Remote Attestation, Blockchain, Ethereum, IoT, Healthcare, Denial of Service, DoS, Internet of Things, Security

ACM Reference format:

Michail Bampatsikos, Christoforos Ntantogian, Christos Xenakis and Stelios C.A. Thomopoulos. 2019. BARRETT BlockchAin Regulated REmote aTTestation. In *Proceedings of WI '19: IEEE/WIC/ACM International Conference on Web Intelligence (WI '19 Companion), October 14-17, 2019, Thessaloniki, Greece.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3358695.3361752

1 Introduction

Due to their low computational resources, IoT healthcare devices are easy targets of cyber-attacks since they cannot use security measures such as antivirus or firewall to defend themselves. These attacks aim to prevent them from functioning properly or turn them to bots for Distributed Denial of Service (DDoS) attacks. It is crucial to a person's wellbeing and life that such devices operate as expected and without interruptions. There have been examples of attacks against medical IoT devices such as the one [1] presents. In this case, researchers hacked a medical ultrasound IoT device which used outdated software.

A way to protect these devices from malware and cyber-attacks that may corrupt their firmware is RA. RA is a process through which an IoT device with low computational resources (prover) proves the correctness of its firmware and software to a remote party (verifier) with considerably higher computational resources, upon the latter's request. After receiving the request, the prover computes a checksum of its memory and generates a Report (R). Then the prover sends R to the verifier. There R goes through the verification process, and if the verification of R fails, then corrective actions are taken to restore the prover.

In the scope of this paper, our objective is to perform RA on healthcare IoT devices, which in many cases are members of dynamic and heterogeneous networks that dominate daily life. In addition to performing RA on the devices of such networks, we

WI '19 Companion, October 14-17, 2019, Thessaloniki, Greece

ACM ISBN 978-1-4503-6988-6/19/10...$15.00
https://doi.org/10.1145/3358695.3361752

also seek to protect them from problems that may occur from the usage of RA protocols.

A problem that occurs in many RA settings is that a malicious verifier may repeatedly send ARs to a prover in order to prevent it from performing the tasks it was designed for or deplete its battery. This happens when the verifier sends multiple ARs to the prover, thus constantly forcing it to perform attestation. This issue makes a protocol that was designed to protect IoT devices a means to attack them. This type of attack is known as a CDoS attack. Some RA solutions propose the inclusion of timestamps [2] or nonces [3] in the ARs to protect the prover from the above attack. Specifically, the freshness of the timestamp or the uniqueness of the nonce prevents the execution of the RA process for replayed ARs. However, a malicious verifier may bypass these mechanisms in the following way. Let's assume that a verifier creates thousands of ARs, each one of them milliseconds after the other. Each of these ARs has a respective timestamp, and as soon as the verifier generates all of them it sends them to the prover. Since every AR has a different timestamp [2], the RA protocol will not treat this behavior as a replay attack and will accept them all. That will force the prover to measure its internal state for every AR that reaches it. Similarly, in case a nonce is used to prevent AR replay attacks, this does not prevent the prover from invoking and executing the RA process for ARs with different nonces. It is worth mentioning that most of the related work does not address this issue.

Another problem that may manifest in an RA setting is that the verifier may deliberately cause network congestion, by sending multiple ARs, to the part of the network to which the prover is connected. The purpose of this action is delaying the exchange of information between the prover and parties entitled to interact with it. That can have grave consequences in scenarios where the timely exchange of information is crucial such as in the case of healthcare IoT devices. In the context of this paper with the term network congestion we refer to the malicious behavior described above.

These shortcomings make apparent the need to deter such attacks and reduce the frequency on which a prover receives ARs by verifiers. To achieve this, we propose the BARRETT architecture, the purpose of which is to protect provers from CDoS and network congestion by combining an RA protocol with a PEN.

The PEN acts as the component that prevents CDoS attacks and network congestion as follows: A verifier, which is a general-purpose computer and a node in the PEN, submits ARs in the form of Ethereum transactions. The term Ethereum transaction in the context of this work refers to a transaction that contains an AR unless stated otherwise. To submit a transaction to the PEN every node, including the verifiers, must pay a fee. This requirement makes any attempts of conducting a CDoS attack or causing network congestion prohibitively expensive in terms of monetary cost. Therefore, the potential loss of funds, which is proportional to the number of ARs, de-incentivizes the verifier from performing CDoS attacks or causing network congestion. After the verifier submits the transaction containing AR to the PEN, the transaction undergoes validation and then is inserted into a block. The mining process adds the block on the Ethereum blockchain, and then a smart contract notifies the prover about the AR. Besides notifying the prover about the AR, the smart contract also acts as an additional defense mechanism against CDoS and network congestion. It does that by setting a limit to the number of verifiers that can concurrently send ARs to a prover as well as to the number of ARs that can reach a prover in a period of time.

An additional characteristic of PENs that contributes to the prevention of CDoS attacks in an RA setting are the mining delays. Specifically, these delays reduce the frequency on which ARs reach a prover, thus diminishing the potential workload that a multitude of these ARs might impose on a device. These delays also prevent the verifier from causing network congestion to the network segment the prover is connected. In short, we employ Ethereum to protect the prover from CDoS attacks and network congestion by having it regulate the number of ARs that reach the latter.

In summary, the contributions of this paper are the following:

- We propose the BARRETT architecture which protects provers from CDoS attacks and from network congestion. It achieves that through the PEN, which imposes a fee for every AR, thus making CDoS prohibitively expensive. Additionally, a smart contract executing on the PEN enables BARRETT to protect the provers from the above issues by setting a limit to the number of ARs that can reach a prover in a period of time.
- We present a solution which via the Ethereum blockchain provides non-repudiation of actions and an immutable log that enables us to determine if a verifier behaves dishonestly. In particular, if a verifier labels a legit R as dishonest, BARRETT will detect its foul play since the blockchain stores both the prover's R and the verifier's evaluation of R.

The paper is structured as follows. Section2 provides a brief description of the related work with regards to RA as well as the use of blockchain in DDoS mitigation. Section3 presents the BARRETT architecture as well as the threat model and assumptions. Performance and security of the architecture are examined in section 4. Finally, section 5 concludes the paper and discusses future work.

2 Related Work

A typical RA protocol operates as follows: A verifier sends an AR, which contains a challenge c, to a prover. The prover invokes some trusted attestation code which computes a measurement over a memory region that was specified by the verifier in its request. In most cases, this measurement is a Message Authentication Code (MAC) computation over the memory region using a secret key K shared between the prover and the verifier. The prover's security architecture protects K. The prover then sends R, which is the output of the measurement process, to the verifier who determines if it is valid by using K and reaches a conclusion whether the prover is in a healthy or compromised state. There are four types of RA, (1) Hardware-based RA, (2) Software-based RA, (3) Hybrid RA, and (4) Swarm RA.

Hardware-based RA relies on using a secure hardware component such as the Trusted Platform Module (TPM) [4], or secure coprocessors to compute the checksum of the content loaded into memory. The purpose of the hardware components is to protect cryptographic keys and utilize them to encrypt or digitally sign data. Moreover, they can be used to produce a hash of a system's firmware and software configuration. An early example of hardware-based RA is secure boot [5]. It is worth noting that Intel also produced a hardware-based approach called Software Guard Extension (SGX) RA [6]. Another hardware-based RA protocol, which is very relevant to our work, because it combines RA with blockchain, is TM-Coin [7]. TM-Coin is the first and only approach so far, to the authors' knowledge, that combines an RA protocol with blockchain. However, TM-Coin only uses the blockchain and its miners to store the prover's R and perform verification duties respectively. Furthermore, it requires the provers that participate in it to use ARM Trustzone, which results in an increased monetary cost. The problem of high cost exists in every hardware-based RA approach. That is because the implementation of these approaches requires to purchase and integrate the security-oriented hardware with the device.

Software-based RA verifies the internal state of devices that have no security-oriented hardware to support attestation. These approaches allow a device to use only software (e.g., hash functions) to compute a checksum of their memory. Some software-based RA protocols are SWATT [8], VIPER [9], SBAP [10], SAKE [11], SCUBA [12] as well as ReDAS and ICE presented in [13] and [14] respectively. These protocols rely on time-based checksums and are only suitable for resource-constrained devices that communicate directly with the verifier [15]. Moreover, they are vulnerable to a variety of attacks such as the Time of Check Time of Use attack as well as return-oriented programming-based attacks [16].

Hybrid RA is a hardware (Read-Only Memory/ ROM)-software co-designed approach. In most hybrid RA approaches, the prover stores the code responsible for the attestation as well as the shared secret key K inside its ROM. The first hybrid RA protocol was SMART [16], which did not allow a device to interrupt the RA process to perform a time-critical task. Trustlite [17] changed that by allowing interruptions of the RA process. Other hybrid RA approaches are HYDRA, TyTan, and SMARM [2] [18] [19]. HYDRA was the first hybrid RA to build upon formally verified software components that ensure memory isolation and protection but did not allow interruptions of the RA process. SMARM enabled the detection of malware that can relocate itself in different regions in the prover's memory. Additionally, it gave the ability to the prover to interrupt the attestation process to perform time-critical tasks.

Finally, swarm RA is a fourth kind of RA, the purpose of which is to assure a verifier that the internal state of a group of IoT devices is correct. Examples of swarm RA protocols are SEDA, SANA, and LISA [20] [21] [22]. Swarm RA uses one of the previous three approaches to attest a group of devices. A problem of most swarm RA approaches is that they do not support the interruption of the attestation process as well as the detection of relocating malware.

Regarding our proposal to use blockchain as a defensive mechanism against CDoS attacks in RA settings, it is worth mentioning that several works have proposed using blockchain technology to mitigate DDoS attacks. Although DDoS differ from CDoS attacks, both aim to make a network resource unavailable. Many approaches suggest using blockchain to reinforce collaborative defense mechanisms against DDoS attacks. Collaborative defense mechanisms aim to mitigate the impact of DDoS attacks by having various organizations and stakeholders to cooperate through the exchange of DDoS related information. Examples of such approaches are [23], [24], [25]and [26] which propose the use of blockchain to handle the exchange of information regarding DDoS attacks as an alternative to complex information-sharing protocols. This information may include the IP addresses of the attackers that performed DDoS, the IP addresses of the victims of these attacks, as well as whitelisted IP addresses.In addition to those, [27] proposes making the IoT devices and the server with which they communicate members of an Ethereum network in order to prevent DDoS. For a device to send data to the server, it must generate a transaction containing these data and submit it to the blockchain. To submit a transaction, the device must pay a fee, so it must possess Ether balance (gas). A smart contract limits the amount of gas every device can have. This limitation depends on the device's specifications, its bandwidth as well as resources requirements. After the gas is depleted, the IoT device can no longer send messages. This is done to protect the server from DDoS attacks in case the devices become bots. However, [27] does not consider using delays the mining process causes as an additional way to mitigate DDoS attacks, something we do to mitigate CDoS attacks.

It is worth noting that many past works proposed using a Blockchain in conjunction with IoT devices for various purposes. One such example is [28], which proposed using blockchain and smart contracts to manage firmware updates of IoT devices.

3 BARRETT Assumptions and architecture

3.1 Assumptions and Threat Model

In this paper, we assume that the verifier aims to attest healthcare IoT devices which may belong to a heterogeneous and dynamic network. This type of networks comprises devices that may have different software and hardware configuration. Moreover, some of these devices do not possess a TPM and are not single network-hop neighbors with the verifier.

The threat model of this paper focuses on adversaries that can interact only remotely with the prover. Thus, they cannot modify the hardware of the device since they do not have physical access to it. Moreover, we consider only internal adversaries (malicious verifiers) that use the RA protocol to harm the prover. External adversaries that seek to harm the provers by bypassing the RA protocol are out of the paper's scope. The same applies to man in the middle attacks, malware, and eavesdropping on network traffic.

3.2 The BARRETT Architecture

The BARRETT architecture comprises three entities. (1) The PEN which is the main mechanism that protects the provers from CDoS attacks and network congestion. (2) The Verification Nodes (VNs) which are members of the PEN and act as the verifiers in BARRETT. There are two types of VNs: a) Full VNs that send ARs, verify R, store a copy of the blockchain and mine it. b) Light VNs that only send ARs and verify R. (3) Finally, we have the provers which are IoT devices and members of the PEN. The provers measure their internal state and produce R, which they submit to the blockchain as well as send it to the VN. Figure 1 depicts the interactions between each one of these parties.

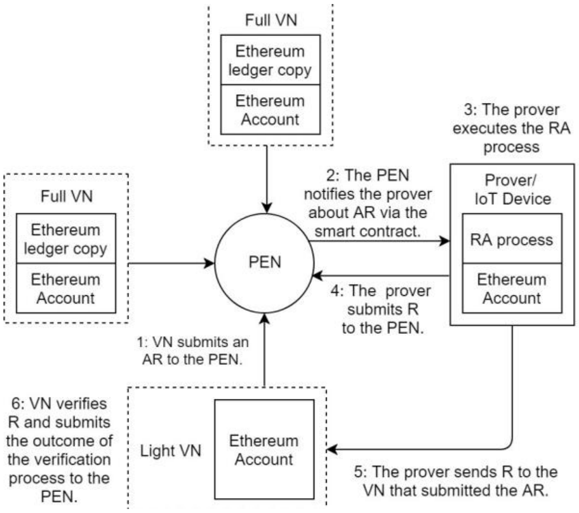
Full VN
Ethereum ledger copy
Ethereum Account
Full VN
Ethereum ledger copy
Ethereum Account
PEN
2: The PEN notifies the prover about AR via the smart contract.
3: The prover executes the RA process
Prover/ IoT Device
RA process
Ethereum Account
4: The prover submits R to the PEN.
1: VN submits an AR to the PEN.
6: VN verifies R and submits the outcome of the verification process to the PEN.
Light VN
Ethereum Account
5: The prover sends R to the VN that submitted the AR.

Figure 1: BARRETT Conceptual Architecture

The core component of the BARRETT architecture is the PEN since it is the main mechanism that prevents CDoS attacks against the provers and network congestion. It achieves that as follows. First by imposing a fee for every AR that a VN submits thus making CDoS prohibitively expensive in terms of monetary cost. Therefore, the potential loss of funds which is proportional to the number of ARs used in a CDoS attack, de-incentivizes the verifier from performing CDoS attacks. Second, by reducing the frequency on which ARs reach a prover in a period of time and as a result the number of times it invokes the RA process. BARRETT makes this possible through the mining delays that occur in PENs, which also delay the inclusion of transactions that contain ARs to the blockchain. Third, by having a smart contract impose a limit to the number of verifiers that can send ARs to a prover concurrently as well as to the number of ARs that can reach a prover in a period of time. The PEN comprises nodes that submit and in some cases mine various types of transactions. Some of these nodes are VNs that send ARs to the provers in the form of Ethereum transactions. Besides the VNs, the provers are also members of the PEN and a smart contract, deployed on the PEN, notifies them about ARs addressed to them.

In addition to the above, the PEN also acts as an immutable event log, which makes possible to detect when a VN behaves dishonestly. BARRETT achieves that since the IoT device and the VN submit as transactions to the blockchain R and the outcome of the verification process respectively. As a result, in case a VN labels on the blockchain an honest IoT device as compromised, another VN will eventually detect this behavior, since the blockchain also stores the prover's R.

The second component of BARRETT are the VNs, which are general-purpose computers that possess considerably higher computational resources in comparison to the provers. For a VN to be a member of the PEN, it must have an Ethereum account. To each account, an address corresponds, that uniquely identifies it in the blockchain network, as well as a public cryptography key pair. Moreover, we assume that the Ethereum account of every VN has some balance in Ether cryptocurrency. There are two types of VNs, the light VNs, and the full VNs. The light VNs only submit transactions that contain ARs to the blockchain and verify R a prover produced and sent to them. The full VNs, on the other hand, submit ARs on the blockchain and participate in the mining process. Furthermore, they store a local copy of the blockchain and verify R. The outcome of the verification process is submitted as a transaction to the blockchain. Both types of VNs must pay a fee each time they wish to submit an AR to the blockchain. Note that every VN stores the secret key K, which shares with a prover and uses in an RA session. The sharing mechanism of K is not within the paper's scope. It is worth noting that the full VNs may possess higher processing power and storage capabilities in comparison to the light VNs. That enables them to cope with the requirements that the mining process and the storage of the blockchain copy impose.

The beneficiaries of all these restrictions BARRETT imposes on the VNs are the provers, which are IoT devices and members of the PEN. The provers have an Ethereum account with some Ether cryptocurrency balance and can submit transactions. As in VN's case, the Ethereum account of each prover has an address that uniquely identifies it and a public cryptography key pair. Moreover, the RA process is stored and executed on the prover. Additionally, the prover stores in its ROM the RA process's code and the shared secret key K. The prover's security architecture protects K and the Ethereum account's private key from unauthorized use. After the smart contract notifies the prover that a VN has submitted an AR on the blockchain concerning it, the prover computes the checksum of their internal state and produces R. Then the prover submits R to the blockchain in the form of a transaction and at the same time sends it to the VN who verifies it.

Regarding the smart contract we mentioned earlier in this section, it is responsible for handling and regulating the submission of ARs to the PEN. Specifically, it sets a maximum limit to the total number of ARs that all the verifiers can submit to the blockchain in a predefined period of time. That happens only in case all these ARs concern the same prover. If the submitted ARs reach this limit, then the smart contract accepts no more ARs concerning that prover to the blockchain until that period of time elapses. To perform these functions, the smart contract stores the Ethereum address that corresponds to the account of each prover and each VN. It also stores the addresses to impose control w.r.t. which PEN nodes can participate in BARRETT and send ARs. Details regarding the deployment of the smart contract are outside the paper's scope.

3.3 Flow of Data

Initially, a VN generates a transaction, that contains an AR, which submits to the PEN via the smart contract and pays the respective fee. After the transaction undergoes validation, it is inserted into a block, and then the mining process adds that block to the Ethereum blockchain. As soon as the mining of the block completes, the smart contract notifies the prover about the AR, which in turn authenticates it. If the authentication of the AR is successful, then the prover proceeds with executing the RA process and produces R. The prover submits R as a transaction to the PEN as well as sends it directly to the VN that submitted the AR. The VN verifies R, and if it is correct it accepts it otherwise rejects it and takes corrective action. In both cases, it publishes its decision on the blockchain. A high-level depiction of the flow of data in BARRETT provides Figure 1.

3.4 Design Decisions

The main reason we chose a PEN is because we wish to deter malicious verifiers from sending multiple (thousands) ARs to a prover. In BARRETT, the PEN helps deter such behaviors primarily by requiring from the VNs to pay fees to submit ARs. Additionally, it offers higher data integrity and robustness in comparison to private as well as consortium Ethereum blockchains since thousands of nodes around the world store copies of the ledger. Moreover, the average block mining time in a PEN that uses Proof of Work [32] consensus algorithm is 13.27 seconds [29] at the time of this writing. This delay, just like the fees, plays a role in the protection of provers from CDoS attacks and network congestion. This delay helps since it significantly reduces the frequency on which a malicious verifier can send ARs via transactions. Note that the actual mining delay may vary significantly from the average one as there have been instances of its duration exceeding 40 seconds or being as short as 5 seconds [30].

Regarding the RA protocol that BARRETT uses, we chose not to design our solution around a specific type of RA. The main rationale behind this decision is to enable our architecture to work with a variety of RA types and be compatible with them. Thus, our solution will be able to protect provers in a variety of RA scenarios from malicious verifiers.

4 Performance and Security Analysis

4.1 Performance Analysis

In this section, we examine performance issues that the combination of a PEN with an RA protocol causes. As mentioned in section 3, the verifiers in our architecture are nodes in the PEN. While some of them can only submit transactions containing ARs, others participate in the mining process and store a local copy of the PEN's blockchain ledger. As stated in [31] the size of the PEN's ledger is 132.57 GB, thus storing a local copy of it imposes a significant storage overhead on the Full VNs. In addition to the storage overhead, the mining function of Ethereum imposes a considerable performance overhead to the full VNs. It is worth noting that full VNs can cope with storing a local copy of the blockchain since their storage space can range between 500 GB and 1 TB. Moreover, it is well within the full VNs' capabilities to cope with the performance burden of the mining process. Additionally, since both types of the VNs, as well as the provers, have Ethereum accounts each of them must dedicate a total of 928 bits from their respective memory for their Ethereum account information; 256 bits for its private key, 160 bits for their Ethereum address, and 512 bits for the public key [32]. Consequently, our architecture imposes a storage overhead of 928 bits on the prover, which is acceptable even in the case of class 1 IoT devices, that possess 10 KB of RAM and 100 KB of ROM.

Another issue that the use of PEN in our solution causes is the relatively long delays due to the mining process. These delays also hold in the inclusion of an AR to the blockchain. The duration of these delays makes the BARRETT architecture unsuitable for scenarios in which there is very low tolerance for delays w.r.t. the time it takes for an AR to reach the prover. In some cases, these delays can even exceed 40 seconds, which are added to the total time it takes for R to reach the verifier and consequently the system administrator. If the system administrator needs to take informed action about an IoT device immediately, then these delays may indirectly cause damage (e.g., in cases immediate action must be taken about a compromised sensor of an Industrial Control System). Therefore, the BARRETT architecture is only suitable for RA scenarios that are tolerant to delays. Note that mining delays with a duration close to 40 seconds frequently occur in PENs.

4.2 Security Analysis

In this section, we examine the security of BARRETT architecture as well as the security properties it obtains from the PEN. As section 3 explains, the PEN acts as an immutable event log in the BARRETT architecture that records in the form of transactions the actions of every party (VNs and provers) that participates in BARRETT. The parties cannot refute their transactions since they sign those with the private key of their respective Ethereum account. Consequently, BARRETT also provides accountability and non-repudiation of actions. Moreover, since, there are thousands of nodes around the world that store a copy of the public blockchain, single points of failure are non-existent. Thus, we also obtain high and constant data availability.

Another feature of our solution that stems from using a PEN is that transaction fees deter a malicious verifier from performing CDoS. This happens because to send thousands of ARs as blockchain transactions would require from the verifier to pay a prohibitively high fee. Even if the attacker possesses enough ether balance to pay for the transaction fees to carry out the attack, it has to consider what is the benefit and the cost of such an attack. Additionally, the delays imposed by the mining process in our case contribute to significantly reducing the frequency by which ARs reach a prover. As a result, if a malicious verifier has a relatively significant amount (e.g., 100) of Ether at its disposal which aims to use them all to generate multiple transactions containing ARs, these ARs will not reach the targeted prover at the same time. Furthermore, as section 3 states, the average duration of mining delays is 13.27 seconds. Consequently, one AR per 13.27 seconds on average is not enough to cause congestion to the part of the network to which a prover is connected. It is worth mentioning that the actual mining delay in many cases differs from the average as there have been instances of it being as short as 5 seconds or even exceeding 40 seconds. Therefore, the time an AR reaches a prover may vary.

It is worth noting that the delays and the fees are not enough in case a prover receives many ARs from different verifiers concurrently. These ARs will eventually force the prover to invoke the RA process, thus preventing it from performing its tasks. One may claim that an RA approach which allows interruptions of the RA process by time-critical tasks, can mitigate this. The problem with such approaches is that the RA process is only interrupted when the IoT device must perform its task. So, in case the device under attack does not have to perform its time-critical task for a long time (e.g., eight hours) the RA process will keep executing until it is eventually interrupted by the task. Our architecture mitigates this issue by using the smart contract we described in section 3 to impose a limit to the number of ARs that can reach a prover in a period of time. Moreover, the smart contract sets a maximum limit to the number of VNs that can send ARs to a prover within that period of time.

5 Conclusion and Future Work

This paper presented the BARRETT architecture, which proposed combining an RA protocol with a PEN to protect provers from CDoS attacks performed by malicious verifiers. Towards this end, we suggested using Ethereum transaction fees to deter adversarial verifiers from sending multiple ARs via the blockchain. We described some performance issues that arise by using BARRETT and discussed how our architecture impacts the security of RA settings. Moreover, we concluded that BARRETT is not suitable for scenarios where there is low tolerance for delays in RA, due to the delays the mining process in PENs causes. Future work will examine replacing the public Ethereum with a private as well as the consequences of such modification to the BARRETT architecture. Additionally, we will present ways through which BARRETT can help an RA protocol to detect malware that an adversary has installed on a prover. Moreover, we will expand the threat model of BARRETT to accommodate adversaries that act outside the protocol. We will also examine the case in which the monetary cost that the fees of BARRETT impose can become prohibitively expensive to honest VNs. This issue can occur in case a VN needs to monitor frequently (e.g., every hour) the status of a prover and thus must send a proportional number of requests. Additionally, we will propose ways to mitigate this issue. Finally, we aim to create a methodology which will determine the amount of the fee that a verifier will be required to pay in order to send an AR. The methodology will determine that amount by taking into account the value and importance of a device to its stakeholders.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the anonymous reviewers for their valuable comments. In this paper, the research of the authors M. Bampatsikos and S. C. A. Thomopoulos is supported by Stavros Niarchos Foundation (SNF) in conjunction with EXODUS Ltd, under Grant No. 12149 "Support of scholarships for industrial PhD's and post-doc industrial positions and adjunct industrial researcher" whereas the research of the authors C. Xenakis and C. Ntantogian in this paper is supported by the EU as part of the CUREX project (H2020-SC1-FA-DTS-2018-1 under grant agreement No 826404).

REFERENCES

[1] Checkpoint, "UltraHack: The security Risks of Medical IoT," 2019. [Online]. Available: https://blog.checkpoint.com/2019/03/07/ultrahack-the-security-risks-of-medical-iot/ . [Accessed 30 July 2019].

[2] K. Defrawy, N. Rattanavipanon and G. Tsudik, "HYDRA: hybrid design for remote attestation (using a formally verified microkernel)," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, Massachusetts, 2017.

[3] F. Brasser, K. B. Rasmussen, A. Sadeghi and G. Tsudik, "Remote attestation for low-end embedded devices: the prover's perspective," in *Proceedings of the 53rd Annual Design Automation Conference*, 2016.

[4] Trusted Computing Group, Trusted Platform Module Part 1: Architecture, 2016.

[5] W. A. Arbaugh, D. J. Farber and J. M. Smith, "A secure and reliable bootstrap architecture," in *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, Oakland, 1997.

[6] V. Costan and D. Srivinas, "Intel SGX Explained," *IACR Cryptology*, 2016.

[7] J. Park and K. Kwangjo, "TM-Coin: Trustworthy management of TCB measurements in IoT," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kona, 2017.

[8] A. Seshadri, A. Perrig, L. van Doorn and P. Khosla, "SWATT: softWare-based attestation for embedded devices," in *Proceedings of the IEEE Symposium on Security and Privacy 2004*, Berkeley, 2004.

[9] Y. Li, J. M. McCune and A. Perrig, "VIPER: verifying the integrity of PERipherals' firmware," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011.

[10] L. Yanlin, M. M. Jonathan and A. Perrig, "SBAP: Software-Based Attestation for Peripherals," in *Proceedings of the Trust and Trustworthy Computing 2010*, 2010.

[11] A. Seshadri, M. Luk and A. Perrig, "SAKE: Software Attestation for Key Establishment in Sensor Networks," in *Proceedings of the International Conference on Distributed Computing in Sensor Systems*, 2008.

[12] A. Seshadri, M. Luk, A. Perrig, L. van Doorn and K. K. Pradeep, "SCUBA: Secure Code Update By Attestation in sensor networks," in *Proceedings of the 5th ACM workshop on Wireless security*, 2006.

[13] A. M. Azab, C. Kill, E. Sezer, P. Ning and X. Zhang, "Remote attestation to dynamic system properties: Towards providing complete system integrity evidence," in *Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, 2009.

[14] A. Seshadri, M. Luk, A. Perrig, L. van Doorn and P. K. Khosla, "Using FIRE & ICE for Detecting and Recovering Compromised Nodes in Sensor Networks," 2004. [Online]. Available: https://www.researchgate.net/publication/235102807_Using_FIRE_ICE_for_Detecting_and_Recovering_Compromised_Nodes_in_Sensor_Networks. [Accessed 1 May 2019].

[15] A. Francillon, Q. Nguyen, K. B. Rasmussen and G. Tsudik, "Systematic Treatment of Remote Attestation," *IACR Cryptology,* 2014.

[16] K. Defrawy, A. Francillon, D. Perito and G. Tsudik, "SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust," in *Proceedings of NDSS Symposium 2012*, 2012.

[17] P. Koeberl, A. Sadeghi and V. Varadharajan, "TrustLite: A Security Architecture for Tiny Embedded Devices," in *Proceedings of the Ninth European Conference on Computer Systems*, 2014.

[18] F. Brasser , B. Mahjoub , A. Sadeghi , C. Wachsmann and P. Koeberl, "TyTAN: Tiny trust anchor for tiny devices," in *Proceedings of the IEEE Design Automation Conference (DAC)*, 2015.

[19] X. Carpent, N. Rattanavipanon and G. Tsudik, "Remote Attestation of IoT Devices via SMARM: Shuffled Measurements Against Roving Malware," in *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust*, 2018.

[20] N. Asokan, F. Brasser, A. Ibrahim, A. Sadeghi, M. Schunter, G. Tsudik and C. Wachsmann, "SEDA: Scalable Embedded Device Attestation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[21] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A. Sadeghi and M. Schunter, "SANA: Secure and Scalable Aggregate Network Attestation," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[22] X. Carpent, K. Eldefrawy, N. Rattanavipanon and G. Tsudik, "Lightweight Swarm Attestation: A Tale of Two LISA-s," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017.

[23] J. Dheeraj and S. Gurubharan, "DDoS Mitigation Using Blockchain," *International Journal of Research in Engineering, Science and Management,* October 2018.

[24] Z. Ahmed, N. Afaqui and O. Humayan, "Detection and Prevention of DDoS attacks on Software Defined Networks Controllers for Smart Grid," *International Journal of Computer Applications,* March 2019.

[25] B. Rodrigues, T. Bocek, A. Lareida, D. Hausheer, S. Rafati and S. Burkhard, "A Blockchain-Based Architecture for Collaborative DDoS Mitigation with Smart Contracts," in *Proceedings of the IFIP International Conference on Autonomous Infrastructure, Management and Security 2017*, 2017.

[26] B. Rodrigues, T. Bocek and B. Stiller, "Multi-domain DDoS Mitigation Based on Blockchains," in *Proceedings of the IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security*, 2017.

[27] U. Javaid, A. Siang, M. Aman and B. Sikdar, "Mitigating loT Device based DDoS Attacks using Blockchain," in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, 2018.

[28] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access,* May 2016.

[29] Etherscan, "Ethereum Block Time History (2019)," 2019. [Online]. Available: https://etherscan.io/chart/blocktime. [Accessed 2019].

[30] Ethstats, "Ethstats," 2019. [Online]. Available: https://ethstats.net. [Accessed 30 July 2019].

[31] Etherscan, "Ethereum Chain Data Size Growth," 2019. [Online]. [Accessed 30 July 2019].

[32] G. Wood and A. M. Antonopoulos, Mastering Ethereum, Sebastopol, California, USA: O'Reily Medica Inc, 2018.