Towards More Practical Software-based Attestation

Rodrigo Vieira Steiner, Emil Lupu

Department of Computing, Imperial College London, London, U.K.

Abstract

Software-based attestation promises to enable the integrity verification of untrusted devices without requiring any particular hardware. However, existing proposals rely on strong assumptions that hinder their deployment and might even weaken their security. One of such assumptions is that using the maximum known network round-trip time to define the attestation timeout allows all honest devices to reply in time. While this is normally true in controlled environments, it is generally false in real deployments and especially so in a scenario like the Internet of Things where numerous devices communicate over an intrinsically unreliable wireless medium. Moreover, a larger timeout demands more computations, consuming extra time and energy and restraining the untrusted device from performing its main tasks. In this paper, we review this fundamental and yet overlooked assumption and propose a novel stochastic approach that significantly improves the overall attestation performance. Our experimental evaluation with IoT devices communicating over real-world uncontrolled Wi-Fi networks demonstrates the practicality and superior performance of our approach that in comparison with the current state of the art solution reduces the total attestation time and energy consumption around seven times for honest devices and two times for malicious ones, while improving the detection rate of honest devices (8% higher TPR) without compromising security (0% FPR).

Keywords: software-based attestation, wireless, security, internet of things

Email addresses: r.v.steiner@imperial.ac.uk (Rodrigo Vieira Steiner), e.c.lupu@imperial.ac.uk (Emil Lupu)

1. Introduction

With each passing day, we interact with more and more devices throughout the Internet. Analysts predict that by 2020 at least 20 billion connected devices will be in use across the globe [1]. The idea of an Internet of Things (IoT) promises that countless *things* will work autonomously and collaboratively form huge distributed systems to provide a series of services. However, the introduction of these devices into our daily life, across smart environments, autonomous vehicles, industry 4.0, critical national infrastructure, and medical and healthcare systems raises serious concerns about their integrity. A malicious device can steal information, report delusive data, or even disrupt system operations, all of which could lead to disastrous situations. We have already seen cases such as Stuxnet, a malware targeting industrial control systems [2], the Mirai botnet, mainly constituted of IoT devices and believed to be the biggest distributed denial of service (DDoS) attack to date [3], and more recently the WannaCry ransomware hitting organizations such as the British National Health Service (NHS) [4]. Therefore, at any given point, one must be able to tell whether or not a *thing* is operating as it should or if it has been compromised.

Nonetheless, in the future IoT not all *things* will have a reliable connection to the network, a last-generation processor equipped with tamper-resistant hardware, and unlimited amount of energy. On the contrary, most *things* will have little processing power, severe energy constraints, and will communicate over a wireless medium [5]. Besides, many existing devices will continue to work for a long period of time and will also take part in the IoT.

In this context, software-based attestation poses as a promising solution as it supposedly allows a trusted entity to validate the memory contents of an untrusted device without requiring any specific hardware. Due to its high applicability, it has received much attention [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. However, proposed schemes have been evaluated under simulation or well-controlled environments. They rely on strong, and sometimes unrealistic, assumptions which may hinder their deployment and has led to much discussion

about their real security [18, 19, 20, 21, 22]. For instance, it is commonly assumed that using the maximum known network round-trip time (RTT) to define the attestation timeout allows all honest devices to reply in time — the timeout is critical to distinguish an honest from a compromised device. While this assumption usually holds in controlled laboratory environments, it is typically untrue in real deployments and particularly so in the IoT context where many devices communicate over an intrinsically unreliable wireless medium and can suffer from issues such as external interference. Furthermore, the higher the attestation timeout, the higher the number of computations the untrusted device has to perform, consuming extra time and energy and restraining it from performing its primary tasks. Hence, using the maximum known network RTT incurs a high overhead and still does not guarantee an honest device will always pass attestation.

In this work we propose a novel stochastic software-based attestation approach that breaks the standard single challenge-response protocol used by previous schemes into a sequence of challenges with shorter timeouts, requiring only one of them to be correctly answered within time. We take advantage from the fact that most of the time the actual network RTT is much smaller than the maximum known RTT. Thus, we can configure a sequence of challenges in such a way that there is a high probability at least one will be replied in time. While this approach significantly improves the attestation performance, it remains secure against all known attacks. Moreover, it is less susceptible to collusion, impersonation, proxy, and high execution time variance attacks as we discuss in Section 4.4. The main contributions of our proposal are: (I) reviewing a critical overlooked assumption made by software-based attestation schemes; (II) proposing a new interaction pattern that uses a sequence of short challenges, instead of a single large one, allowing for the reduction of the overall attestation time and energy consumption, and improving its detection rate; and (III) an experimental evaluation over real-world uncontrolled Wi-Fi networks comparing the proposed scheme to the current state of the art solution. To the best of our knowledge, this is the first time software-based attestation has been evaluated

under real-world Wi-Fi networks. One additional, minor, contribution is the implementation of a software-based attestation routine to a hardware platform designed for the IoT, the Intel Edison [23], which we used in our experiments. At this stage, our attestation routine implementation is just a prototype that has been used to validate the proposed stochastic attestation approach and is not the focus of the paper.

2. Background

Software-based attestation mechanisms follow a challenge-response protocol depicted in Figure 1. A trusted device, usually called *Verifier*, validates the integrity of an untrusted device, commonly named *Prover*, by inspecting its memory contents. The verifier challenges the prover, with a random number, and waits for it to respond for a limited amount of time. Once the prover receives a challenge it uses it as a seed to an attestation routine that traverses its memory in a pseudo-random fashion and computes a checksum of the accessed memory addresses. It is assumed that the verifier knows the memory contents of the prover, so that it can perform the same computation and compare the result with the response received. If the prover replies with a correct value within the time limit then it is considered to be honest. Otherwise, it may have been compromised. By definition, a malicious prover should never be able to pass attestation.

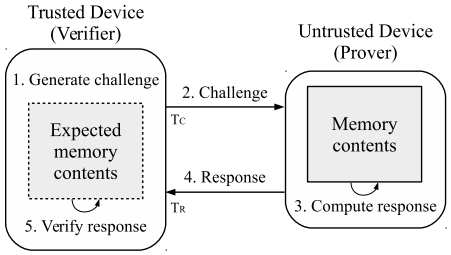


Figure 1: Attestation overview.

Among software-based attestation mechanisms there are two different approaches regarding the time limit for the prover's response [24]. The timeout

obviously has to incorporate the time necessary for the prover to execute the attestation routine. But it also has to include the time necessary to send and receive the challenge and response messages. Some approaches have very strict timing conditions while others have looser requirements.

Schemes with a looser time condition assume that the prover's memory can be completely filled with random noise, leaving no space for an adversary to install its malicious code and still manage to calculate the correct response since it will have to erase the original memory contents [8, 9, 10, 15, 17]. Nevertheless, an adversary can still compress the application code in program memory releasing enough space for its own malicious code and decompress the original code during attestation [18]. Unless the application code is written in a compressed form to the program memory [25], which requires special hardware to decompress and execute it, an adversary will always be able to perform this attack. Thus, software-based attestation schemes with loose time conditions are inherently insecure against compression attacks and cannot be used in practice.

On the other hand, approaches with strict timing conditions assume that the attestation routine is implemented in a time-optimal way, such that an adversary cannot modify it and still compute the correct response in time [6, 7, 11, 12]. The attestation routine contains a checksum computation loop that must be executed several times. Since the memory reads are inside the loop, to hide any modified memory content an adversary would have to introduce new instructions inside the loop, increasing the computation time. Therefore, such schemes have to limit the response time, which includes the RTT, in order to detect any time discrepancy. Nonetheless, these approaches have also been targeted by attacks. Shankar et al. show that incorporating CPU execution side effects, such as translation lookaside buffer (TLB) misses, is not sufficient to authenticate software [26]. Wurster et al. explore the use of virtual memory mapping to overcome attestation without increasing the computation time [27]. Castelluccia et al. presents a rootkit-based attack that uses return-oriented programming (ROP) to hide the malware during attestation and install it again just after its end [18]. Castelluccia et al. also describes an attack that is faster than

the threshold detection of a previous attestation mechanism. Notwithstanding, the work of Seshadri et al. [7] addresses the first two attacks and is not vulnerable to the third [19]. Gardner et al. [28] describes an attack that is faster than the one depicted by Seshadri et al. [7], however, they also present a countermeasure. Furthermore, as observed by Yan et al., the problem of threshold overestimation can be resolved by simply updating the threshold as new attacks are discovered [11]. Each time the threshold is updated, it becomes increasingly harder for an adversary to come up with a new attack.

While approaches with strict timing conditions have not been formally proven to be completely secure, they do not suffer from inherent vulnerabilities like schemes with loose timing conditions, and thus, appear to be a viable alternative. At the very least such schemes can be used to slow down attackers who would not be able to compromise devices without avoiding detection before coming up with a way of circumventing attestation.

3. Problem Definition

The problem with attestation mechanisms with strict timing conditions is that they rely on strong assumptions to limit the adversary capabilities.

Time-optimal implementation. Assuming that the implementation of the attestation routine is time-optimal such that the best attack has a fixed overhead per iteration. Despite recent advances in code optimization [29, 30, 31], proving code optimality is still an open problem. In this case, one can draw a parallel with cryptosystems which rely on unproven assumptions, e.g. the RSA problem [32]. By making the algorithm public, under the scrutiny of a large community, one can have a higher confidence in the security achieved.

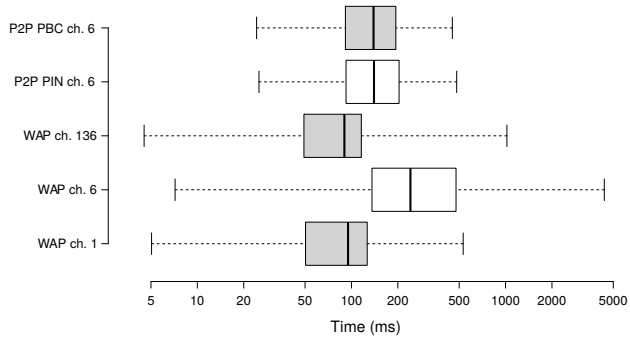
No physical attacks. Another assumption is that the adversary cannot modify the prover's hardware. Otherwise an attacker could increase the prover's clock speed and compute a modified version of the attestation routine within the time limit. Nonetheless, it is impractical to perform such physical attacks in large-scale.

Authentication. The communication channel used for attestation is also assumed to provide message authentication. One possible solution is the use of Radio Frequency fingerprint techniques to identify devices by the unique characteristics of their radio transmitters [33, 34, 35]. More recently, Feng et al. [17] proposed an attestation scheme that uses the microcontroller SRAM as a Physical Unclonable Function (PUF) to provide authentication without requiring any hardware changes.

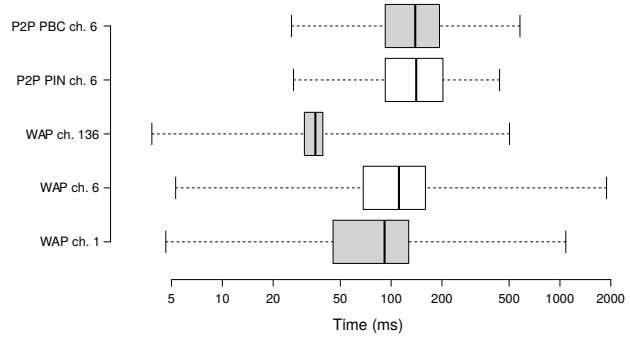
Restricted communication. A fourth assumption is that during attestation the prover can only communicate with the verifier and therefore they must be within communication range. This prevents the so called proxy attack, in which the prover relays the challenge to a more powerful device that is able to compute a valid response within the timeout. Again we can imagine possible solutions, such as jamming the prover or having devices monitoring the communication of their neighbors during attestation. However, to the best of our knowledge, no previous approach has put these to practice.

Maximum RTT known a priori. Finally, these mechanisms also assume that by using the maximum known network RTT to specify the challenge timeout guarantees all honest devices can pass attestation.

In this paper, we focus on this last assumption. To show that it is not practical, we went to the Imperial College Central Library to collect real data. First, we scanned for available Wireless Access Points (WAPs) and Ad-Hoc connections in range and found a total of thirty six connections. Many of them were using the same frequency, which leads to interference and performance degradation. From these we had access to three different channels: 1, 6, and 136. We used two Intel Edison devices to measure the RTT. The Edison also supports Wi-Fi Peer to Peer (P2P) allowing devices to connect directly without the need for a WAP. It provides two modes of connection, Push Button Control (PBC) and PIN mode (PIN), which we also evaluated. Additionally, we measured the RTT with the Edison Wi-Fi driver power management, which is used to reduce power consumption, both on and off. Figure 2 shows the results obtained by using one device to ping the other a thousand times for each channel.



(a) Power Management On



(b) Power Management Off

Figure 2: Box plot of the network round-trip time under different channels.

The variable transmission delays experienced are likely due to packet collisions, retransmissions, and the use of CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) supplemented by RTS/CTS (Request to Send/Clear to Send) frames[1], all of which are influenced by the total number of devices concurrently using the network, the traffic generated by them, as well as external interference. Note that we had no control over these factors since we only owned

[1]While the use of RTS/CTS frames partially avoids the hidden terminal problem it also introduces the exposed terminal problem.

two devices in the network and the measurements were not done all at the same time. With exception to channel 1, all channels present a smaller mean and standard deviation for measurements done with the power management turned off. Given all variables into play, it is difficult to give an explanation for the odd behaviour of channel 1, which is already an indicator that assuming a maximum RTT is unpractical and inefficient. To make the network RTT more deterministic, schemes such as SCUBA [36] assume that the prover has exclusive access to the radio channel during attestation. However, due to external interferences, the actual RTT may be worse than anticipated resulting in honest provers failing attestation. Nevertheless, for now, let us work with this assumption to estimate the necessary amount of attestation routine iterations.

Since the attestation routine is assumed to be implemented in a time-optimal way, any modification an adversary makes to it will incur in computation time overhead. To compensate this overhead, the adversary may try to reduce the network RTT between the compromised device and the verifier in order to still reply within the attestation timeout. In theory, an adversary can have zero RTT, in which case its time advantage would be the maximum RTT allowed by the attestation process. Seshadri et al. also consider the time an adversary gets from warming up the cache to avoid cache misses during execution [7]. Nevertheless, when compared to the measured RTT samples in Figure 2, this gain is negligible. Because even the best adversary has an overhead per iteration, it is possible to compensate its time advantage, which is fixed, by increasing the number of iterations required.

To compute the number of iterations, three factors must be taken into account: c, the prover's CPU clock speed; a, the adversary time advantage; and o, the adversary overhead per iteration of the attestation routine [7]. Furthermore, because the memory is being traversed in a pseudo-random fashion, to guarantee, with high probability, that each memory address is accessed at least once, based on the Coupon Collector's Problem, the number of iterations for a memory of size m has to be at least $O(m \ln m)$ [37]. Hence the number of

iterations i can be computed as below:

$$i \geq \frac{(c \cdot a)}{o} \geq m \cdot \ln(m) \tag{1}$$

The prover's CPU clock speed and the assumed adversary overhead per iteration are constants. Consequently the number of iterations becomes a linear function of the allowed maximum RTT as shown in Figure 3. The Edison CPU operates at 500 MHz, and the best known attack has an overhead of 1.037 CPU cycles per iteration [2]. The scenario that requires the smaller number of iterations, P2P PIN ch. 6 from the measurements in Figure 2 (b), would still need to execute 211463783 iterations. Based on the measurements we have done of the time taken by each iteration of our implementation of the attestation routine it would take approximately 24.6 seconds to execute this number of iterations.

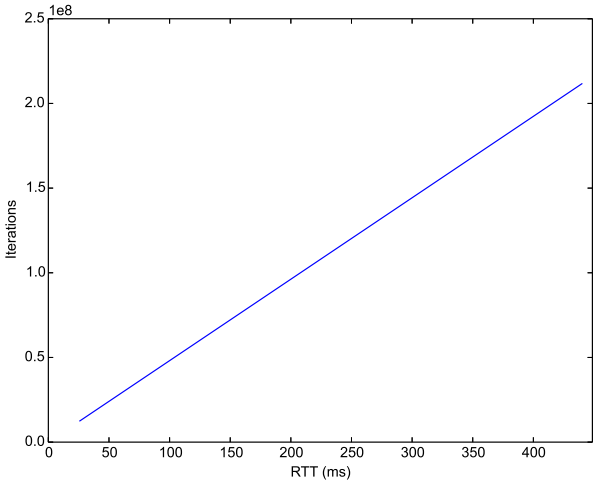


Figure 3: Number of iterations based on the RTT.

According to existing approaches, regardless of the actual network condi-

[2] We implemented the memory copy attack as described in [28], which has an overhead of two `xor` instructions per iteration. We refer the reader to their original work for further details.

tions, the assumed maximum RTT is always used to compute the number of iterations of the attestation routine. Therefore, even if the network conditions are better than estimated, the prover still has to compute a large number of iterations. For instance, as can be seen in Figure 2 (b), 75% of the RTT samples taken from P2P PIN ch. 6 take less than 202 ms and 50% of the samples are within the range of 26 to 139 ms. Nevertheless, a traditional attestation mechanism would always consider the maximum RTT of 438.682 ms to compute the number of iterations, wasting both time and energy. Furthermore, the higher the number of iterations, the higher the execution time jitter due to clock, cache, and TLB fluctuations. As observed by Li et al. [12], to avoid false negatives (honest provers not replying in time) the challenge timeout should also include the execution time jitter. However, an adversary could take advantage of this time extension to execute additional instructions and still pass attestation with non-negligible probability (false positive). Thus, a high execution time jitter forces a tradeoff between false negatives and false positives, both of which are not desirable.

4. Stochastic Software-based Attestation

The single challenge-response protocol followed by traditional schemes limits the amount of flexibility of the attestation process. Using a fixed timeout based on the assumed network maximum RTT always demands a high number of attestation routine iterations to be computed, and, due to external interference on the communication channel and execution time jitter, still does not guarantee that an honest prover is always able to respond within the time limit. Therefore, we propose a probabilistic attestation approach that allows us to break the single challenge into a series of challenges with shorter timeouts whenever the breakage is expected to be cost beneficial and there is a high probability that at least one challenge in the series will be replied in time.

For instance, suppose that when estimating the network conditions we observe a maximum RTT of x but most RTT samples observed are smaller or

equal to $x/2$. In this case, rather than sending a single challenge with a timeout based on x, it is better to send a series of challenges each of which has a timeout based on $x/2$. We name this series of challenges an attestation round. Note that the verifier only sends a new challenge in the series after the previous challenge timeout expires and the prover did not respond with a valid response. If the prover correctly replies to a challenge in time then the attestation round is over and there is no need for the verifier to send the remaining challenges since the prover has already demonstrated it has not been compromised. In practice, our approach is similar to performing traditional attestation multiple times in a row using a shorter timeout and stopping either when the prover passes, or when all challenges of the attestation round have been sent.

In summary, our stochastic software attestation mechanism has two different outputs. It outputs ACCEPT whenever a challenge is correctly replied within the time limit, and outputs REJECT if all challenges in a series expire without receiving a valid response. It is important to note that while ACCEPT confirms that the prover is honest, REJECT does not confirm that it has been compromised, but only serves as an indication that it might have been.

4.1. Assumptions

With exception to assuming a maximum network RTT a priori, we share most assumptions made by software-based attestation mechanisms with strict timing conditions: the attestation routine is implemented in a time-optimal way; the verifier cannot be compromised by the adversary and knows the prover's hardware and software configuration; the prover is in communication range of, and can only communicate with, the verifier during attestation; authentication is provided either by radio frequency fingerprint or SRAM PUF; the adversary has full control of the prover's software but cannot modify its hardware.

Ideally, the WAP would perform the role of the verifier. It is in direct communication range with the provers and could easily keep track of the network RTT, it has unlimited energy, and if it gets compromised then essentially the entire network can be considered to be compromised. Furthermore, it could

block any outgoing traffic from devices under attestation.

4.2. Network Conditions Estimation

Before starting attestation, the verifier has to (i) collect a sufficient amount of network RTT samples, (ii) find the probability distribution function that best fit these samples, and (iii) use the distribution to calculate the attestation parameters under the desired confidence level. Section 4.5 describes these last two steps in further details exemplifying our approach using the RTT samples from Figure 2.

We gather network RTT samples by making the verifier ping the provers. This can be done in two different ways: either (i) *online*, when we collect new samples every time before performing attestation; or (ii) *offline*, obtaining all samples only once during an initialization phase.

On the one hand, the online collection allows us to estimate the actual network conditions before triggering attestation enabling the use of a different timeout for each round. As a result, if conditions are good, we can decrease the timeout, consequently reducing the number of iterations a prover has to perform. While if conditions are bad, we can increase the timeout, which has the opposite effect but also increases the chances of honest provers replying in time. To achieve this dynamic timeout behaviour we amend the challenge message to include, besides a nonce, the number of iterations to be executed and modify the attestation routine to also use this information as a parameter. In order to find a distribution that best represents the actual network conditions during attestation, we allow the use of a dynamic number of RTT samples. We collect ten samples, find the best fitting distribution, calculate the attestation parameters and repeat this process until the time spent collecting samples is higher than the estimated challenge timeout.

On the other hand, the offline collection benefits from not having to collect samples and compute the attestation parameters every time. In this scenario, a large sample set is preferable as it will be more likely to capture the average channel conditions.

4.3. Attestation Round

An attestation round may consist in a series of challenges which respect the following specification:

1. All challenges in a round have the same timeout, and thus, require the same number of iterations to be computed by the prover.
2. Each challenge in the series has a different nonce from the previous ones. Otherwise, an adversary would receive a challenge, solve it expiring its timeout, but then receive it again having the correct answer already computed and would be able to respond in time.
3. The verifier sends a new challenge of the series immediately after the timeout of the current challenge expires without the reception of a valid response.
4. Whenever a challenge is correctly responded within its timeout, attestation is concluded, and the prover is considered to be honest.

4.4. Security Evaluation

To analyse the security of the proposed scheme we first present the existing attacks on attestation previously described in the literature and then explain how they relate to our approach.

Precomputation attack. If the adversary is able to predict the challenges sent by the verifier then it can precompute valid responses.

Replay attack. The adversary might eavesdrop the response of a non-compromised prover and replay it when challenged.

Forgery attack. The adversary may alter the attestation routine to forge a valid response or modify the prover's memory contents in a way that these modifications cancel one another during the checksum computation.

Memory copy attack. The adversary might keep a copy of the original memory contents and compute the checksum over this copy.

Data substitution attack. Instead of keeping a copy of the entire original memory contents, the adversary can copy only the parts of memory that it

has overwritten and modify the attestation routine to redirect memory reads whenever necessary.

Compression attack. The adversary can keep a copy of the original contents in a compressed format and decompress it during attestation.

Collusion attack. Multiple compromised nodes might try to parallelize the response computation or each node might install the malicious software in different memory locations and communicate during attestation to recover the original contents.

Impersonation attack. A compromised node might impersonate genuine nodes and send invalid attestation responses or might impersonate the base station and forward challenges directed to itself to genuine nodes and relay the response.

Proxy attack. A compromised node might relay the challenge to a more powerful device that is able to compute a valid response in its behalf.

Return-oriented programming attack. The adversary may modify the control-flow of the prover's original code to execute arbitrary operations without making any changes to the program memory.

High execution time variance attack. The higher the number of iterations performed by the attestation routine, the higher its execution time variance, which must be accounted for in the timeout, allowing an adversary to execute extra instructions with considerable changes of still replying in time.

By using a cryptographically secure pseudo-random number generator to produce a different nonce for each challenge both the *precomputation* and *replay* attacks are no longer feasible.

The *collusion*, *impersonation*, and *proxy* attacks are not valid under the adopted threat model. As with previous schemes, we assume the prover can only communicate with the verifier during attestation, which prevents both the *collusion* and *proxy* attacks, and that some authentication mechanism is in place, which prevents the *impersonation* attack. Nevertheless, our scheme reduces the success chances of these attacks. Since we use shorter timeouts it becomes harder for nodes to exchange or relay messages and still reply to the

verifier in time. Furthermore, the use of shorter timeouts reduces execution time jitter, thus also preventing the *high execution time variance* attack.

The *forgery*, *memory copy*, *data substitution*, *compression*, and *return-oriented programming* attacks aim to overcome the attestation routine itself. While these attacks are implementation specific, our scheme can be used with different implementations. We only require the routine to also take the number of iterations as a parameter, which has no security implications. In traditional approaches, since the timeout is fixed, the adversary always knows the number of iterations, while in our scheme the adversary only gets to know this information when it receives the challenge. From the attestation routine perspective, the fact that we may execute the routine multiple times in a row cannot be exploited since each run is independent of the previous. There is however, a new attack that an adversary could try to perform against our scheme, which we describe below:

Delay attack. Knowing that we estimate the network conditions with RTT samples, an adversary may deliberately delay all messages handled by compromised nodes. In doing so, the adversary can influence our estimations and increase the calculated challenges timeout in an attempt to gain more time to perform extra computations.

Whenever we increase a challenge timeout we also increase the number of iterations of the attestation routine to compensate for any time advantage the adversary may have, as shown in Equation 1 and Figure 3. Therefore, the *delay* attack would be ineffective in overcoming attestation. In the worse case scenario, it would force the use of a large timeout even though the network conditions are good, similar to what happens with traditional attestation approaches. Nevertheless, a possible countermeasure to this attack is to collect RTT samples from multiple provers and apply anomaly detection techniques to exclude the malicious data introduced by the attacker.

Ultimately, our approach does not make attestation vulnerable to any existing or foreseeable new attacks. In reality, the use of shorter timeouts hinders the execution of *collusion*, *impersonation*, *proxy*, and *high execution time variance*

attacks.

4.5. Analytical Evaluation

We analyse the performance of our proposal using the RTT samples from the same channel we used to discuss traditional approaches in Section 3 — P2P PIN ch. 6 in Figure 2 (b). We chose this in order to compare results. Table 1 summarizes the notation adopted in this paper.

Table 1: Notation summary.

Term	Description
γ	Time of a single iteration of the attestation routine
δ	Maximum RTT allowed
a	Adversary time advantage
c	CPU clock speed
i	Number of iterations
o	Adversary's overhead per iteration of the attestation routine in CPU cycles
N	Maximum number of challenges in a round
$\mathcal{C}$	Number of challenges sent during attestation
$\mathrm{E}[\mathcal{X}]$	Expected value of random variable $\mathcal{X}$
$\mathcal{E}$	Attestation energy consumption
$\mathcal{R}_{\mathcal{T}}(0,\delta)$	RTT observed during attestation
$\mathcal{T}$	Attestation time
$\mathcal{X}_{\mathcal{T}}(a,b)$	Truncated random variable $\mathcal{X}$ within interval $a \leq x \leq b$
E_A	Attestation routine energy consumption
E_P	Prover energy consumption
E_{rx}	Reception energy consumption
E_{tx}	Transmission energy consumption
P_A	Attestation routine power consumption
P_{rx}	Reception power consumption
P_{tx}	Transmission power consumption
T_A	Attestation routine execution time

Having the network RTT samples, the next step in our proposed approach is to find the corresponding best fitting distribution. First we use the Maximum Likelihood Estimation (MLE) method to fit different distributions to the RTT samples. Then, to find the best fitting distribution, we calculate the Sum of Squared Errors (SSE) for each distribution and select the one with the smallest SSE. Figure 4 shows the Probability Density Function (PDF) of the five best fitting distributions over the RTT samples. In this example the Exponential

Power Distribution (labeled exponpow) is the one with the smallest SSE.

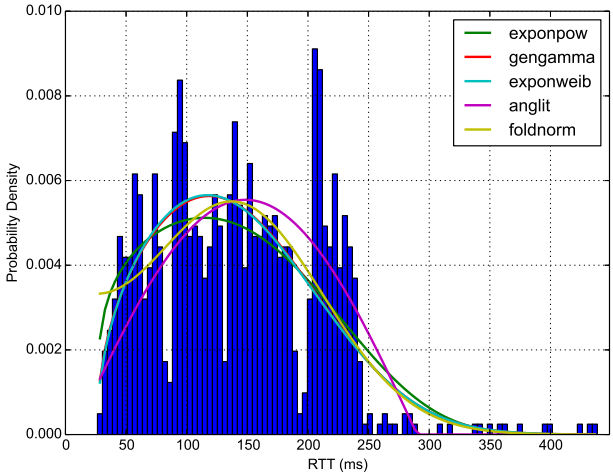


Figure 4: Best fitting distributions for the RTT samples.

Once a distribution is chosen we can use its corresponding Cumulative Distribution Function (CDF) to calculate the probability of successfully sending and receiving the challenge and response messages for any given RTT limit δ. Note that here we are just calculating the probabilities for the messages transmission and reception without taking into account the execution time T_A of the attestation routine. However, since T_A is determined by the number of iterations i, which in turn are determined by δ, the same calculation applies. Therefore, we can use these values to compute the probabilities of sending two, three, up to N challenges in a row, where at least one of them receives a response within δ according to the equation below:

$$\begin{aligned} P(\textsc{accept} \text{ at least 1 challenge in } N \text{ attempts} \mid \delta) = \\ 1 - P(\textsc{reject} \text{ all challenges in } N \text{ attempts} \mid \delta) = \\ 1 - P(\textsc{reject} \mid \delta)^N \end{aligned} \tag{2}$$

Figure 5 shows the results of Equation 2 for $N = \{1, ..., 10\}$. It is possible to

notice that the higher the number of challenges allowed, the faster the probability of success grows. Which is intuitive, the more messages we send, the higher is the chance that at least one of them will be replied in time.

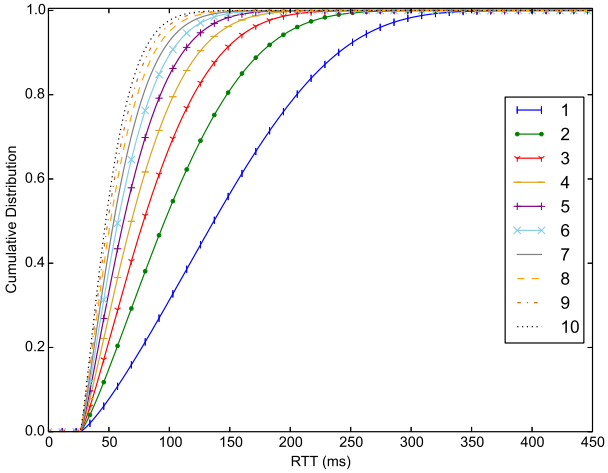


Figure 5: Probabilities of sending from 1 to 10 challenges in a row where at least one is replied within δ.

We can represent the effective number of challenges sent during attestation as a discrete random variable $\mathcal{C} = \{1, 2, 3, ..., N\}$. Both the attestation time and energy consumption depend on the expected value of $\mathcal{C}$, which can be calculated as follows:

$$\mathrm{E}[\mathcal{C}] = \sum_{j=1}^{N} c_j \cdot P(\mathcal{C} = c_j \mid \delta) \tag{3}$$

Where $P(\mathcal{C} = c_j \mid \delta)$ is:

$$P(\mathcal{C} = c_j \mid \delta) = \begin{cases} P(\textsc{reject} \mid \delta)^{c_j - 1} \cdot P(\textsc{accept} \mid \delta) & \text{if } c_j < N \\ P(\textsc{reject} \mid \delta)^{c_j - 1} & \text{if } c_j = N \end{cases} \tag{4}$$

The attestation time depends on $\mathcal{C}$, δ, and on the execution time of the attestation routine T_A, which also depends on δ. We can denote the effective attestation time as a continuous random variable $\mathcal{T}$ defined as:

$$\mathcal{T} = \mathcal{C} \cdot (\mathcal{R}_T(0, \delta) + T_A(\delta)) \tag{5}$$

Where $\mathcal{R}_T(0,\delta)$ denotes a truncated random variable, in the interval $[0,\delta]$, that represents the RTT observed during attestation. We truncate it by δ because once the time limit expires the verifier stops waiting for the response and immediately transmits the next challenge if there is one.

Let us denote the time taken to execute a single iteration by γ, then we can compute T_A as follows:

$$T_A(\delta) = i \cdot \gamma = \frac{(c \cdot \delta)}{o} \cdot \gamma \tag{6}$$

Now, assuming that $\mathcal{C}$ and $\mathcal{R}_T(0,\delta)$ are independent, we can calculate the expected value of $\mathcal{T}$ as:

$$\begin{aligned} \mathrm{E}[\mathcal{T}] &= \mathrm{E}[\mathcal{C} \cdot (\mathcal{R}_T(0,\delta) + T_A)] \\ &= \mathrm{E}[\mathcal{C} \cdot \mathcal{R}_T(0,\delta)] + \mathrm{E}[\mathcal{C} \cdot T_A] \\ &= \mathrm{E}[\mathcal{C}] \cdot \mathrm{E}[\mathcal{R}_T(0,\delta)] + T_A \cdot \mathrm{E}[\mathcal{C}] \end{aligned} \tag{7}$$

Where, given its CDF $F(r)$, its PDF $f(r)$, and the auxiliary function $g(r) = f(r)\ \forall\ 0 \leq r \leq \delta$, the expected value of $\mathcal{R}_T(0,\delta)$ can be computed as:

$$\mathrm{E}[\mathcal{R}_T(0,\delta)] = \frac{\int_0^\delta r \cdot g(r)\,\mathrm{d}r}{F(\delta) - F(0)} \tag{8}$$

Figure 6 shows the maximum and expected attestation time when determining δ under different confidence levels. It is possible to see that the higher the confidence level, the higher is $\mathcal{T}_{max}$. The reason is that $\mathcal{T}$ increases as δ increases, and higher confidence levels require higher δ. On the other hand, as N increases, $\mathrm{E}[\mathcal{T}]$ decreases until it reaches a minimum and then starts to grow. The explanation is that as N increases δ decreases, so it is possible for the prover to reply earlier. However, when δ gets too small, it is more likely that the prover will not be able to respond in time. Thus, more challenges are required and the total expected time starts to increase.

The prover's energy consumption E_P for a single challenge can be calculated as:

$$E_P = E_{rx} + E_A + E_{tx} \tag{9}$$

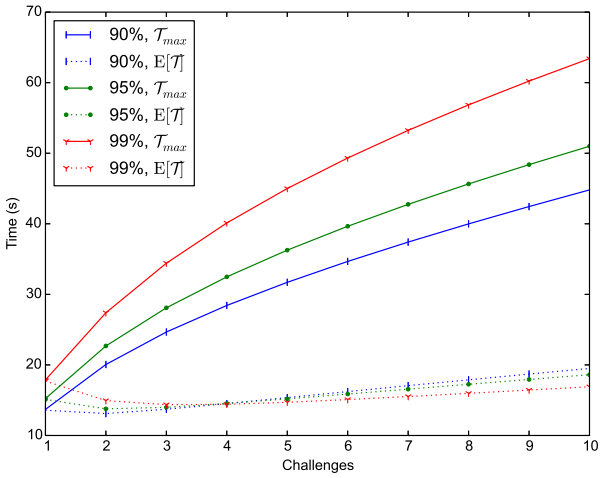


Figure 6: Maximum and expected attestation time under different confidence levels.

Where the challenge reception energy consumption E_{rx} can be computed as:

$$E_{rx} = \frac{\text{Packet size}}{\text{bit rate}} \cdot P_{rx} \tag{10}$$

And, similarly, the response transmission energy consumption E_{tx}:

$$E_{tx} = \frac{\text{Packet size}}{\text{bit rate}} \cdot P_{tx} \tag{11}$$

While the attestation routine energy consumption E_A follows:

$$E_A = T_A \cdot P_A \tag{12}$$

With these equations we can define the prover's effective energy consumption as a discrete random variable $\mathcal{E}$:

$$\mathcal{E} = \mathcal{C} \cdot E_P \tag{13}$$

And its expected value:

$$\mathrm{E}[\mathcal{E}] = \sum_{j=1}^{N} c_j \cdot E_P \cdot P(\mathcal{C} = c_j \mid \delta) \tag{14}$$

To calculate the energy consumption, we have measured the power consumption of the Intel Edison and observed the values shown in Table 2. The packet size, 90 bytes, is the same for both challenge and response. It includes the 802.11 data frame (38 bytes), the IP header (20 bytes), ICMP header (8 bytes), and payload (24 bytes). The channel bit rate is 54 Mbit/s. Figure 7 shows the prover's maximum and expected energy consumption with its Wi-Fi driver power management turned off. The same analysis we did for the attestation time is valid here. The higher the confidence level, the higher is $\mathcal{E}_{max}$, and as N increases $\mathrm{E}[\mathcal{E}]$ decreases until it reaches a minimum and then starts to grow.

Table 2: Edison power consumption (mW).

	Power Management	
	On	Off
P_{Tx}	535	716
P_{Rx}	604	649
P_A	714	839

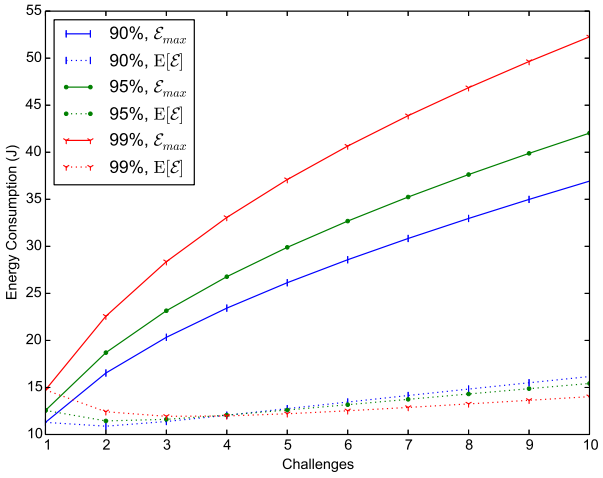


Figure 7: Prover's maximum and expected energy consumption (power management off).

Since both $\mathrm{E}[\mathcal{T}]$ and $\mathrm{E}[\mathcal{E}]$ reach minimum values, we can set N to optimize either the attestation time or energy consumption. Furthermore, to find their

minimums we do not need to compute for arbitrary values of N. Since both $\mathrm{E}[\mathcal{T}]$ and $\mathrm{E}[\mathcal{E}]$ continually decrease as N increases until they reach a minimum, we can stop the computation whenever we find the turning point.

Table 3 summarizes how $E[C]$, i, and T_A change under the same N by the use of different confidence levels to estimate δ. We can also use the values from this table to compare our performance against the one we analysed in Section 3. While traditional attestation schemes are limited to the maximum RTT estimated a priori our approach enables a wide range of options for defining the number of iterations and timeout. We demonstrated that for these network conditions traditional approaches would require 211463783 iterations to be performed by the attestation routine, which would take approximately 24.6 seconds to execute plus the RTT to send the challenge and receive the response. In our best scenario, we could choose $N = 10$, with 90% confidence level, and have our first challenge in the series be responded within the timeout. This would take only 4.4 seconds plus the messages RTT. Even if we limit ourselves to $N = 1$, instead of using the worst RTT measured, we can with 99% confidence level reduce the attestation routine execution time to 17.6 seconds.

4.6. Experimental Evaluation

We designed our stochastic attestation to work either with online or offline RTT samples collection. To provide a fair comparison with the current state of the art solution, we have also implemented an online and offline version of the traditional single challenge-response protocol, even though works in the literature only take the offline approach. Thus, we test four different configurations: our proposed *stochastic online* and *stochastic offline* solutions, the current state of the art solution ([6, 7, 11, 12]) *maximum offline*, and, for completeness, an adaptation of it which we name *maximum online*.

To experiment with real devices, we have implemented an attestation routine, based on the work developed by Seshadri et al. [7], for the Intel Edison platform. We also consider the case of an attacker trying to bypass attestation without being detected and have thus implemented a malicious version of the

Table 3: Parameters according to confidence level.

N	%	E[$\mathcal{C}$]	δ(ms)	i	T_A(s)
	90	1.000	239.939	115660904	13.455
1	95	1.000	266.945	128679161	14.970
	99	1.000	313.216	150983554	17.564
	90	1.316	175.723	84706215	9.854
2	95	1.224	198.747	95804546	11.145
	99	1.100	239.939	115661161	13.455
	90	1.680	144.025	69426404	8.077
3	95	1.504	164.056	79082148	9.200
	99	1.262	200.983	96882741	11.271
	90	2.056	124.506	60017477	6.982
4	95	1.802	142.263	68577185	7.978
	99	1.448	175.724	84706473	9.854
	90	2.439	111.091	53550469	6.230
5	95	2.108	127.071	61253624	7.126
	99	1.645	157.687	76011791	8.843
	90	2.824	101.226	48795302	5.677
6	95	2.417	115.778	55810064	6.493
	99	1.848	144.025	69426501	8.077
	90	3.211	93.628	45132632	5.250
7	95	2.729	107.007	51581855	6.001
	99	2.054	133.255	64234759	7.473
	90	3.598	87.574	42214613	4.911
8	95	3.041	99.973	48191165	5.606
	99	2.262	124.507	60017824	6.982
	90	3.987	82.624	39828466	4.633
9	95	3.355	94.187	45402390	5.282
	99	2.472	117.240	56514707	6.575
	90	4.376	78.493	37836967	4.402
10	95	3.670	89.338	43064981	5.010
	99	2.683	111.091	53550622	6.230

attestation routine performing the memory copy attack as described in [28]. At the current stage, our implementation for the Intel Edison is still a prototype. For instance, although the Edison is a multi-core platform, currently our implementation executes on a single core. This is by no means a limitation imposed by our scheme. In fact, our proposed stochastic approach can be applied to other platforms and to different attestation routines, as long as the latter can take the number of iterations as a parameter (only necessary for the online configurations). Nevertheless, our proof of concept is enough to show the feasibility of our approach.

We performed our experiments in three different locations using real-world Wi-Fi networks: (i) the already mentioned Imperial College Central Library; (ii) a residential building where different apartments share a communal Wi-Fi; and (iii) a coworking office space. Once again, we emphasize we had no control over the ongoing network traffic in any of these locations.

We used Wireshark to capture the packets exchanged between the prover and the verifier, which allowed us to picture missing packets and responses out of bound. In addition, by putting the wireless card in monitor mode, we were able to observe all wireless traffic during attestation rounds, which allowed us to tell how noisy each location was during our tests. By checking MAC addresses, we have also counted the total number of distinct WAP and Ad-Hoc connections in range and unique devices observed during our experiments — note that these numbers were not constant. Table 4 summarizes our findings. It is interesting to see that even though the library was by far the noisiest environment, it was the building location which presented the highest observed RTT. We also noticed that the Edison devices were getting disconnected from the library WAP every time the attestation routine took more than 34 seconds to execute. Whenever this happened, the device would need to first reconnect to the WAP and only then send the attestation response, thus elapsing the timeout. Therefore, we had to limit the maximum RTT allowed to 600 ms to bound the attestation routine execution time and consequently avoid this from happening. The reason for the disconnections was most likely a configuration in the library WAP as the same did not happen in the building location. Effectively, this is another limitation of using the maximum RTT to define the attestation timeout given that the stochastic approach is much less likely to use long timeouts.

Table 4: Observed wireless conditions.

	Building	Library	Office
Average bytes/s	27.7 K	2.67 M	23 K
Connections in range	26	213	118
Min. RTT (ms)	2.33	2.83	2.69
Max. RTT (ms)	985.76	695.13	389.56
Unique devices	625	4203	2111

In each location, we first evaluated the *online* configurations and then used the RTT samples they collected to evaluate the *offline* configurations. To find the best fitting distribution for the samples collected we use the Scipy library, which has more than eighty distributions. To reduce the computation time, we experimented and found a subset of distributions which performed better than others [3]. We then calculate the attestation parameters with 99% confidence level and select the maximum number of challenges in a round (N) with the smallest expected energy consumption ($\mathrm{E}[\mathcal{E}]$). On average, the *stochastic online* configuration collected 16.5 RTT samples per attestation round, which took around 6.4 seconds and spent roughly more 2 seconds to find the best fitting distribution and compute the attestation parameters. Whereas, the *maximum online* configuration did not need to fit a distribution, but rather simply use the maximum RTT observed. Thus, we used a fixed RTT sample of size ten, which took on average 3.6 seconds to collect.

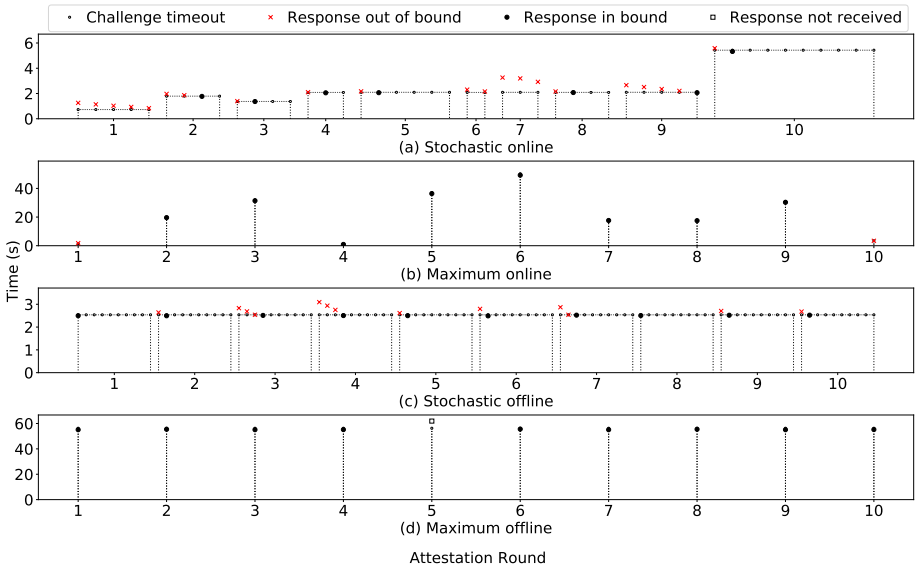


Figure 8: Interaction pattern of the different approaches.

[3] Those were: alpha, foldcauchy, halfcauchy, powerlognorm, fatiguelife, lomax, johnsonsu, cauchy, and gilbrat.

For all locations we attested both the honest and malicious devices fifty times each. To better illustrate how each configuration behaves, Figure 8 shows the first ten attestation rounds performed in the building location. It is possible to see how the maximum number of challenges and their timeout change from round to round in *online* configurations, which is a reflection of the network conditions changing over time. Whereas the *offline* configurations use the same parameters for all rounds. We explicitly show all the challenges that could be sent during an attestation round by plotting their timeout. This enables us to visualize how many other chances the prover still had when it passed attestation.

Figure 9 shows the percentage of times the honest device was able to pass attestation, in other words, the True Positive Rate (TPR), achieved by each configuration in all locations. As can be seen, the *stochastic offline* configuration achieved the best results in every location with an average of 99%, followed by the *stochastic online* (95%), *maximum offline* (91%), and lastly *maximum online* (82%). All configurations presented a 0% False Positive Rate (FPR), meaning that not even once the malicious prover was able to pass attestation. This is expected, since we know the overhead of the adversary attack and set the number of iterations accordingly (Equation 1), such that a malicious prover can only respond after the timeout expires. The only way a false positive could happen in this scenario is if we had a sufficiently high execution time jitter allowing the attacker to reply within time occasionally. This did not occur in our experiments.

Figure 10 displays the attestation time measured by the verifier and energy consumption of the honest prover for all configurations and locations. Bear in mind that it does not take into account the time and energy spent collecting RTT samples, which is a disadvantage of *online* configurations. It depicts the minimum, maximum, average (where the error bar shows the standard deviation), and total (sum of all fifty attestation rounds) values observed during our experiments. Note how small is the standard deviation of the *maximum offline* configuration in comparison to the others. That is because it is the only configuration that does not make use of either multiple challenges or different timeouts

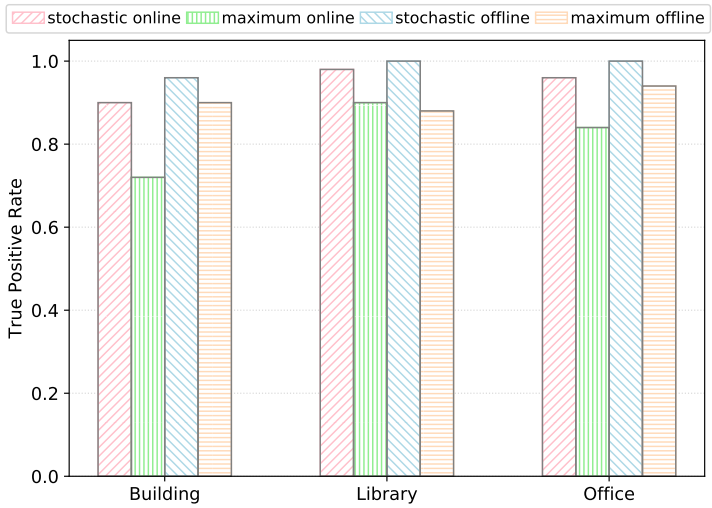


Figure 9: Attestation TPR achieved under different environments.

per attestation round. Thus, the only things changing per round are the attestation routine execution time jitter and the actual challenge-response RTT. It is also possible to see that the *stochastic offline* configuration outperformed the others (except for the *stochastic online* in the library) presenting the best overall results with an aggregate (all three locations) time of 845.55 s and energy consumption of 648.45 J, followed by the *stochastic online* (1025.08 s and 815.98 J), *maximum online* (1616.54 s and 1350.37 J), and *maximum offline* (5548.35 s and 4646.87 J).

Figure 11 shows the attestation time and energy consumption of the malicious prover, again not considering the time and energy consumed collecting RTT samples. In this case, the *stochastic offline* configuration also has a tiny standard deviation. That is because the malicious prover always fails all challenges in a round and all rounds use the same parameters. Note that we are assuming an adversary that tries to bypass attestation by sending a correct answer. If the compromised device were to send a wrong answer, then it would be detected and the attestation would end sooner. This time the *maximum online* configuration outperformed the others in every location presenting the

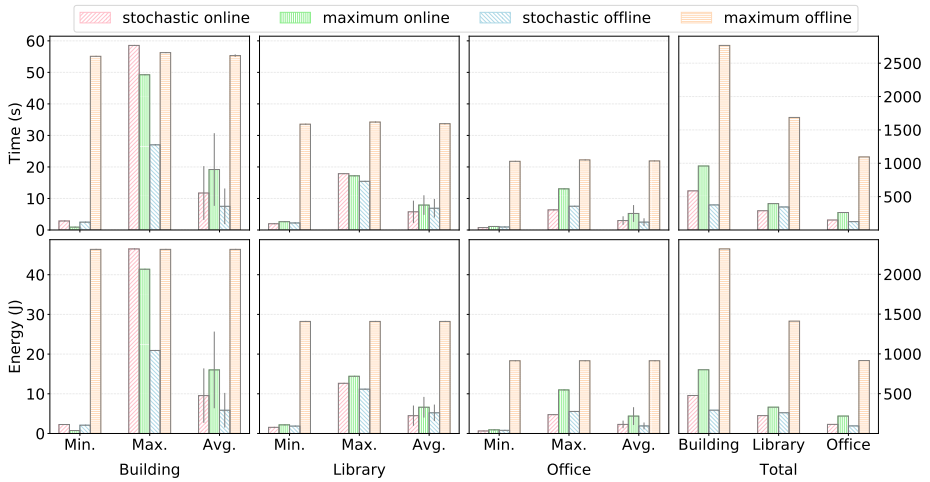


Figure 10: Attestation time and energy consumption (honest prover).

best results (1963.65 s and 1618.64 J), while the *stochastic offline* is the second best (2975.28 s and 2239.85 J), *stochastic online* third (3524.89 s and 2747.35 J), and *maximum offline* fourth (5637.77 s and 4646.87 J).

It is worth mentioning that we expect the majority of devices in a network not to be compromised, and if that is not the case, then the time and energy spent attesting malicious devices is certainly not a primary concern. Therefore, reducing the attestation time and energy for honest devices is much more valuable than doing the same for malicious ones.

In summary, the *stochastic offline* configuration presented the best overall results. In comparison to the current state of the art solution (*maximum offline* configuration), it presented an 8% TPR increase in detecting honest provers and reduced the attestation time and energy consumption around seven times for honest devices and two times for malicious ones. The considerably lower TPR achieved by the *maximum online* configuration is reason enough to disregard it as a good option. Finally, the *stochastic online* configuration is a valid second choice, but just not as good, especially when we consider the extra time and energy required to collect RTT samples before each attestation round.

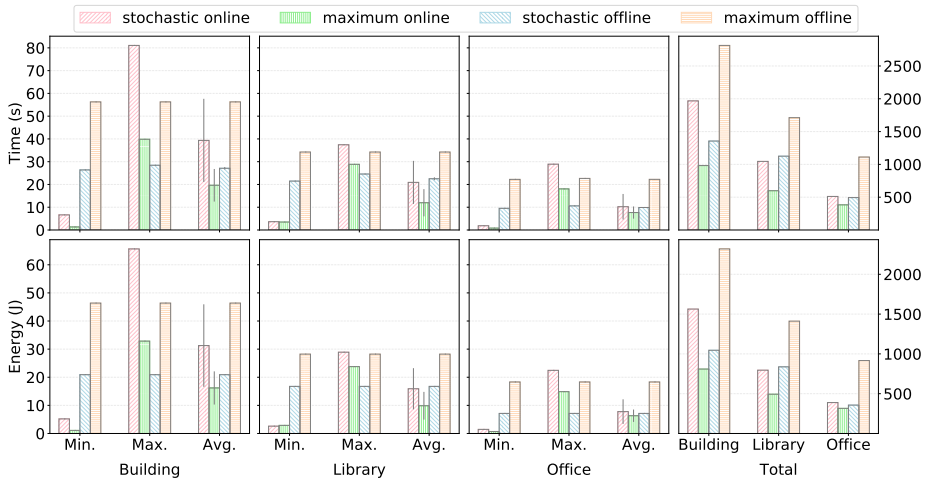


Figure 11: Attestation time and energy consumption (malicious prover).

5. Related Work

Several attestation mechanisms have been proposed over the last years [6, 38, 39, 7, 36, 8, 9, 40, 41, 42, 11, 43, 25, 44, 12, 45, 13, 46, 47, 48, 15, 16, 17, 49, 50]. However, they take different approaches to acquire evidence from the untrusted device [24]. On the one hand, hardware-based attestation relies on tamper-resistant hardware such as the Trusted Platform Module (TPM) [51]. On the other hand, software-based techniques do not require any particular piece of hardware. Lastly, hybrid techniques do not depend on secure hardware but require specific hardware such as ROM and a Memory Protection Unit (MPU).

Each approach has its advantages and disadvantages. The use of secure hardware facilitates attestation since all information provided by it can be considered reliable. Nevertheless, it excludes many devices that do not possess such hardware, and that does not mean only legacy devices. Whether it is due to cost reduction, size minimization, or even the fact that security usually comes as an afterthought when companies try to push their products into the market, the reality is that many new devices are being released without any tamper-resistant hardware. Alternatively, software-based approaches can, in theory, be applied to any device. However, they are hard to implement and rely on strong

assumptions that may undermine the achieved security [18]. Hybrid mechanisms present themselves as a middle ground. An adversary may not modify code written to ROM, and may not even be able to access it if an MPU is protecting it. Nonetheless, devices might not have enough space available in ROM or even an MPU. Also, when a vulnerability on the attestation mechanism is discovered, the use of specific hardware, such as a TPM or ROM, may become a restraint hindering its update.

To the best of our knowledge the only other software-based attestation works that target the network RTT are [52, 13], with the later being an extension done by the same authors. Their approach differs from ours as they propose a multi-hop attestation scheme, while our scheme performs only single-hop attestation. To attest a remote device, the verifier sends a challenge that is relayed across the network until it reaches the prover. The prover's response has to go back through the same path as the challenge. Each relay node records the time when they receive the challenge and response and report this information to the verifier. The verifier then can estimate the average single-hop RTT and detects compromised nodes by using a Bayesian classifier. We identify three drawbacks of this scheme. First, since both the challenge and response have to go through the same path the approach does not tolerate neither node failures or node mobility. Second, to prevent a compromised relay node from modifying the reports of other nodes, they assume the use of cryptographic keys to protect transmitted messages. However, software-based attestation mechanisms are explicitly designed for devices that do not possess any hardware capable of protecting such keys. Third, they assume that all hops in the network have a similar RTT. This is not realistic since each hop may suffer from different sources of interference. Furthermore, the experiments we realized in this paper have shown that even a single-hop may drastically change its conditions over time.

6. Conclusions

With the ever-increasing number of connected devices that are becoming part of our daily lives, software-based attestation can play a very important role in the near future. It is a fact that the *correct* design and implementation of such technique is eminently challenging, and many proposed solutions rely on strong assumptions that may not hold under real-world scenarios. Nonetheless, we still consider that its use is far from being infeasible or worthless and that, at the very least, it can be used as an extra layer of security.

In this paper, we present a novel way of performing attestation which is a step further in the development of practical software-based mechanisms. Instead of using the maximum known network RTT to send a single large attestation challenge, which demands a high number of computations to be performed, we use a series of short challenges that require fewer computations and can be finalized whenever a correct response is given in time. As can be seen from our experimental results, when compared to the current state of the art solution (*maximum offline* configuration) our proposal (*stochastic offline* configuration) considerably reduces the overall attestation time and energy consumption (around seven times for honest devices and two times for malicious ones) while improving the detection rate of honest devices (8% higher TPR) without compromising its security (0% FPR). Nevertheless, all previous proposals can be adapted to follow this new interaction pattern and benefit from its advantages.

As future work, we plan to examine the use of machine learning techniques to estimate the channel conditions and compute the attestation parameters which could potentially improve the performance of the proposed approach. Also, a more meticulous investigation is needed to analyse how effective is the use of smaller timeouts in preventing the execution of collusion, impersonation, and proxy attacks.

7. Acknowledgments

This work was supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under Grant No. 206707/2014-2 awarded to Rodrigo Vieira Steiner.

8. References

[1] A. Nordrum, Popular internet of things forecast of 50 billion devices by 2020 is outdated, IEEE Spectrum [online], Available: http://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated.

[2] R. Langner, Stuxnet: Dissecting a cyberwarfare weapon, IEEE Security & Privacy 9 (3) (2011) 49–51.

[3] N. Woolf, Ddos attack that disrupted internet was largest of its kind in history, experts say, The Guardian [online] Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.

[4] C. Graham, NHS cyber attack: Everything you need to know about 'biggest ransomware' offensive in history, The Telegraph [online] Available: http://www.telegraph.co.uk/news/2017/05/13/nhs-cyber-attack-everything-need-know-biggest-ransomware-offensive.

[5] S. Tozlu, M. Senel, W. Mao, A. Keshavarzian, Wi-Fi Enabled Sensors for Internet of Things: A Practical Approach, IEEE Communications Magazine 50 (6) (2012) 134–143. doi:10.1109/MCOM.2012.6211498.

[6] A. Seshadri, A. Perrig, L. Van Doorn, P. Khosla, SWATT: SoftWare-based ATTestation for Embedded Devices, in: Proceedings of the 25th IEEE Symposium on Security and Privacy, SP '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 272–282. doi:10.1109/SECPRI.2004.1301329.

[7] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, P. Khosla, Pioneer: Verifying Code Integrity and Enforcing Untampered Code Execution on Legacy Systems, ACM SIGOPS Operating Systems Review 39 (5) (2005) 1–16. doi:10.1145/1095809.1095812.

[8] Y.-G. Choi, J. Kang, D. Nyang, Proactive Code Verification Protocol in Wireless Sensor Network, in: Proceedings of the 2007 International Conference on Computational Science and Its Applications, ICCSA '07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 1085–1096. doi:10.1007/978-3-540-74477-1_97.

[9] Y. Yang, X. Wang, S. Zhu, G. Cao, Distributed Software-based Attestation for Node Compromise Detection in Sensor Networks, in: Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems, SRDS '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 219–230. doi:10.1109/SRDS.2007.31.

[10] T. AbuHmed, N. Nyamaa, D. Nyang, Software-Based Remote Code Attestation in Wireless Sensor Network, in: Proceedings of the 28th IEEE Conference on Global Telecommunications, GLOBECOM '09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 1–8. doi:10.1109/GLOCOM.2009.5425280.

[11] Q. Yan, J. Han, Y. Li, R. H. Deng, T. Li, A Software-based Root-of-trust Primitive on Multicore Platforms, in: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS '11, ACM, New York, NY, USA, 2011, pp. 334–343. doi:10.1145/1966913.1966957.

[12] Y. Li, Y. Cheng, V. Gligor, A. Perrig, Establishing Software-Only Root of Trust on Embedded Systems: Facts and Fiction, in: Cambridge International Workshop on Security Protocols, Springer, 2015, pp. 50–68. doi:10.1007/978-3-319-26096-9_7.

[13] X. Yang, X. He, W. Yu, J. Lin, R. Li, Q. Yang, H. Song, Towards a Low-

Cost Remote Memory Attestation for the Smart Grid, Sensors 15 (8) (2015) 20799–20824.

[14] V. P. Illiano, R. V. Steiner, E. C. Lupu, Unity is Strength!: Combining Attestation and Measurements Inspection to Handle Malicious Data Injections in WSNs, in: Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '17, ACM, New York, NY, USA, 2017, pp. 134–144. doi:10.1145/3098243.3098249.

[15] H. Tan, G. Tsudik, S. Jha, MTRA: Multiple-tier remote attestation in IoT networks, in: IEEE Conference on Communications and Network Security, CNS, IEEE, 2017, pp. 1–9.

[16] B. Chen, X. Dong, G. Bai, S. Jauhar, Y. Cheng, Secure and Efficient Software-based Attestation for Industrial Control Devices with ARM Processors, in: Proceedings of the 33rd Annual Computer Security Applications Conference, ACSAC '17, ACM, New York, NY, USA, 2017, pp. 425–436. doi:10.1145/3134600.3134621.

[17] W. Feng, Y. Qin, S. Zhao, D. Feng, AAoT: Lightweight attestation and authentication of low-resource things in IoT and CPS, Computer Networks 134 (2018) 167–182. doi:10.1016/j.comnet.2018.01.039.

[18] C. Castelluccia, A. Francillon, D. Perito, C. Soriente, On the Difficulty of Software-based Attestation of Embedded Devices, in: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS '09, ACM, New York, NY, USA, 2009, pp. 400–409. doi:10.1145/1653662.1653711.

[19] A. Perrig, L. Van Doorn, Refutation of "On the Difficulty of Software-Based Attestation of Embedded Devices" (2010).

[20] A. Francillon, C. Castelluccia, D. Perito, C. Soriente, Comments on "Refutation of On the Difficulty of Software-Based Attestation of Embedded Devices" (2010).

[21] F. Armknecht, A.-R. Sadeghi, S. Schulz, C. Wachsmann, A Security Framework for the Analysis and Design of Software Attestation, in: Proceedings of the 20th ACM Conference on Computer and Communications Security, CCS '13, ACM, New York, NY, USA, 2013, pp. 1–12. doi:10.1145/2508859.2516650.

[22] A. Francillon, Q. Nguyen, K. B. Rasmussen, G. Tsudik, A Minimalist Approach to Remote Attestation, in: Proceedings of the 2014 Conference on Design, Automation and Test in Europe, DATE '14, European Design and Automation Association, 3001 Leuven, Belgium, 2014, pp. 1–6.

[23] C. Intel, Intel Edison Module Documentation, Available at: https://software.intel.com/en-us/iot/hardware/edison/documentation, Last accessed: August 2018 (2015).

[24] R. V. Steiner, E. Lupu, Attestation in Wireless Sensor Networks: A Survey, ACM Computing Surveys 49 (3) (2016) 51:1–51:31. doi:10.1145/2988546.

[25] B. Vetter, D. Westhoff, Simulation Study on Code Attestation with Compressed Instruction Code, in: Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 296–301. doi:10.1109/PerComW.2012.6197498.

[26] U. Shankar, M. Chew, J. D. Tygar, Side Effects Are Not Sufficient to Authenticate Software, in: Proceedings of the 13th Conference on USENIX Security Symposium, SSYM '04, USENIX Association, Berkeley, CA, USA, 2004.

[27] G. Wurster, P. C. Van Oorschot, A. Somayaji, A Generic Attack on Checksumming-Based Software Tamper Resistance, in: Proceedings of the 26th IEEE Symposium on Security and Privacy, SP '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 127–138. doi:10.1109/SP.2005.2.

[28] R. Gardner, S. Garera, A. D. Rubin, On the Difficulty of Validating Voting Machine Software with Software, in: Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology, EVT'07, USENIX Association, Berkeley, CA, USA, 2007, pp. 11–11.

[29] E. Schkufza, R. Sharma, A. Aiken, Stochastic Superoptimization, in: Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '13, ACM, New York, NY, USA, 2013, pp. 305–316. doi:10.1145/2451116.2451150.

[30] N. P. Lopes, J. Monteiro, Weakest precondition synthesis for compiler optimizations, in: International Conference on Verification, Model Checking, and Abstract Interpretation, Springer, 2014, pp. 203–221.

[31] N. P. Lopes, D. Menendez, S. Nagarakatte, J. Regehr, Provably Correct Peephole Optimizations with Alive, in: Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '15, ACM, New York, NY, USA, 2015, pp. 22–32. doi:10.1145/2737924.2737965.

[32] R. L. Rivest, B. Kaliski, Rsa problem, in: Encyclopedia of cryptography and security, Springer, 2005, pp. 532–536.

[33] O. Ureten, N. Serinken, Wireless security through rf fingerprinting, Canadian Journal of Electrical and Computer Engineering 32 (1) (2007) 27–33. doi:10.1109/CJECE.2007.364330.

[34] S. U. Rehman, K. W. Sowerby, C. Coghill, Analysis of Impersonation Attacks on Systems Using RF Fingerprinting and Low-end Receivers, Journal of Computer and System Sciences 80 (3) (2014) 591–601. doi:10.1016/j.jcss.2013.06.013.

[35] D. A. Knox, T. Kunz, Wireless Fingerprints Inside a Wireless Sensor Net-

work, ACM Transactions on Sensor Networks 11 (2) (2015) 37:1–37:30. doi:10.1145/2658999.

[36] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, P. Khosla, SCUBA: Secure Code Update By Attestation in Sensor Networks, in: Proceedings of the 5th ACM Workshop on Wireless Security, WiSe '06, ACM, New York, NY, USA, 2006, pp. 85–94. doi:10.1145/1161289.1161306.

[37] M. Mitzenmacher, E. Upfal, Probability and computing: Randomized algorithms and probabilistic analysis, Cambridge University Press, 2005.

[38] T. Park, K. G. Shin, Soft Tamper-Proofing via Program Integrity Verification in Wireless Sensor Networks, IEEE Transactions on Mobile Computing 4 (3) (2005) 297–309. doi:10.1109/TMC.2005.44.

[39] M. Shaneck, K. Mahadevan, V. Kher, Y. Kim, Remote Software-based Attestation for Wireless Sensors, in: Proceedings of the 2nd European Conference on Security and Privacy in Ad-Hoc and Sensor Networks, ESAS '05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 27–41. doi:10.1007/11601494_3.

[40] C. Kil, E. C. Sezer, A. M. Azab, P. Ning, X. Zhang, Remote Attestation to Dynamic System Properties: Towards Providing Complete System Integrity Evidence, in: Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks, DSN '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 115–124. doi:10.1109/DSN.2009.5270348.

[41] D. Perito, G. Tsudik, Secure Code Update for Embedded Devices via Proofs of Secure Erasure, in: Proceedings of the 15th European Conference on Research in Computer Security, ESORICS'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 643–662. doi:10.1007/978-3-642-15497-3_39.

[42] D. Zhang, D. Liu, DataGuard: Dynamic Data Attestation in Wireless Sensor Networks, in: Proceedings of the 2010 IEEE/IFIP International Con-

ference on Dependable Systems and Networks, DSN '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 261–270. doi:10.1109/DSN.2010.5544307.

[43] K. Eldefrawy, G. Tsudik, A. Francillon, D. Perito, SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust, in: Proceedings of the 19th Network and Distributed System Security Symposium, NDSS '12, Internet Society, 2012, pp. 1–15.

[44] P. Koeberl, S. Schulz, A.-R. Sadeghi, V. Varadharajan, TrustLite: A Security Architecture for Tiny Embedded Devices, in: Proceedings of the 9th European Conference on Computer Systems, EuroSys '14, ACM, New York, NY, USA, 2014, pp. 1–14. doi:10.1145/2592798.2592824.

[45] H. Tan, W. Hu, S. Jha, A remote attestation protocol with Trusted Platform Modules (TPMs) in wireless sensor networks, Security and Communication Networks 8 (13) (2015) 2171–2188. doi:10.1002/sec.1162.

[46] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, C. Wachsmann, SEDA: Scalable Embedded Device Attestation, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, ACM, New York, NY, USA, 2015, pp. 964–975. doi:10.1145/2810103.2813670.

[47] T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, G. Tsudik, C-FLAT: Control-FLow ATtestation for Embedded Systems Software, arXiv preprint arXiv:1605.07763.

[48] A. Ibrahim, A.-R. Sadeghi, G. Tsudik, S. Zeitouni, DARPA: Device Attestation Resilient to Physical Attacks, in: Proceedings of the 9th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '16, ACM, New York, NY, USA, 2016, pp. 171–182. doi:10.1145/2939918.2939938.

[49] P. Maene, J. Gtzfried, R. de Clercq, T. Mller, F. Freiling, I. Verbauwhede, Hardware-Based Trusted Computing Architectures for Isolation and Attestation, IEEE Transactions on Computers 67 (3) (2018) 361–374. doi:10.1109/TC.2017.2647955.

[50] X. Carpent, K. Eldefrawy, N. Rattanavipanon, A.-R. Sadeghi, G. Tsudik, Reconciling Remote Attestation and Safety-critical Operation on Simple IoT Devices, in: Proceedings of the 55th Annual Design Automation Conference, DAC '18, ACM, New York, NY, USA, 2018, pp. 90:1–90:6. doi:10.1145/3195970.3199853.

[51] T. C. G. TCG, TPM Main Specification Level 2 Version 1.2, Revision 116, Available at: http://www.trustedcomputinggroup.org/resources/tpm_main_specification, Last accessed: August 2018 (2011).

[52] X. He, X. Yang, R. Li, Q. Yang, A novel delay-resilient remote memory attestation for smart grid, in: Wireless Algorithms, Systems, and Applications, Springer, 2013, pp. 88–99.