Trusted Single-Source Sensors using SNARKs

Saad Bin Shams
Siemens AG
Munich, Germany
saad.shams@siemens.com

Emanuel Regnath
Siemens AG
Munich, Germany
emanuel.regnath@siemens.com

Andreas Bogner
ex-Siemens AG
Munich, Germany
mail@andreasbogner.de

Sebastian Steinhorst
Technical University of Munich
Munich, Germany
sebastian.steinhorst@tum.de

***Abstract*—The trustworthiness of sensor data readings is crucial for IoT applications. The trend of decentralized and distributed architectures give rise to multi-party scenarios where mutual trust between different parties might not be present. Current approaches to increase trust in sensor readings include cryptographic authentication, redundancy of sensors, and plausibility verification of received signals. However, these approaches can often only defend against certain types of attacks.**

In this paper, we propose a multi-layer approach to increase the trust in single data sources, such as wireless sensors, by using a trusted execution environment (TEE) and succinct non-interactive arguments of knowledge over authenticated data (AD-SNARKs). First, we bring several trust metrics as close to the sensor as possible to reduce the surface of attacks. Second, we develop an optimized constrained system for AD-SNARKs that allows offloading statistical operations on the sensor data, such as moving average, to a non-trusted constrained device. By lowering the number of constraints to 6, our implementation is able to generate proofs in 60ms on a Raspberry Pi 3(B) offering 128 bit of security with all validation data fitting into 1023 bytes of payload. Compared to other security approaches, this is a small overhead for achieving provable sensing *and* processing of data from source to consumer, which is a major step towards *distributed trust* for IoT applications.

***Keywords*—ADSNARKs, Trust, IoT, IIoT**

I. Introduction

Smart devices are used for sensing changes in the environment, perform computation onboard and send the results to a consumer of such data. The trend of pushing the computation towards the edge and bringing decentralized and distributed architectures to the forefront, brings new challenges regarding security and trust.

To explain the problem we are trying to tackle, consider the following example in context of Industrial Internet of Things (IIoT): In a blockchain-based supply chain management application, the party receiving the shipment is interested to know if the temperature of the freight is in the range that complies to the requirements. Such a shipment could be of vaccines where the temperature needs to be sub-zero to ensure its integrity. Wireless sensors can be used to monitor the temperature and the data generated by the sensors is sent to a blockchain. Since the receiving party does not have control over the shipment process, it has limited trust in the sensor data in the blockchain. In the real world, there can be multiple suppliers and the sensing capabilities can be provided by some other service providers, which complicates the situation.

979-8-3503-4647-3/23/$31.00 ©2023 IEEE

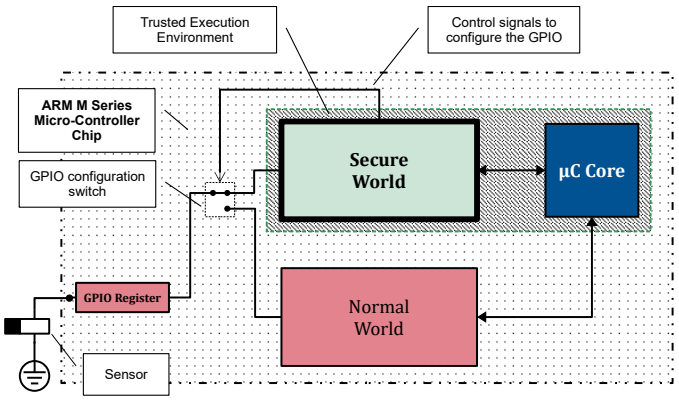

Figure 1: The architecture of the micro-controller shows how the value from the sensor is read directly into the secure world. This diagram does not show the exact internal architecture, but is developed to aid in explanation of the concept. The blocks colored in red show that they are not inside the TEE and can be malicious.

At first glance, this problem can be solved by authenticating and performing integrity protection on data from the wireless sensors. However, the computation onboard can still be manipulated and the receiver would have to recalculate the data of interest from the raw sensor data to enhance trust. When the amount of data and computation grows, recalculating them is not an attractive proposition. Redundant sensors can be added to the wireless device, but this would only protect against a sensor failure because the computation on the data can still be manipulated.

Instead of focusing on redundancy, we aim for a solution that works even with a single sensor. First, this simplifies the problem domain to find and evaluate approaches. Second, it allows our solution to be applied to low-cost systems consisting of constrained device. Third, a solution for a single-sensor scenario can easily be scaled to a more complex system but not necessarily vice versa.

A. Contributions

In this paper, we propose a multi-layer trust-enhancing methodology to increase the trust in single data sources, such as individual sensors. We aim to bring security mechanisms for data readings as close to the source as possible and enable verifiable computation on constrained device or on devices between the sensor and consumer, such as a gateways. In particular, we

- incorporate several trust metrics, such as Distributed trust (DT), close to the data source (sensor) using a TEE, as shown in Figure 1. The closeness is numerically measured and is shown in (Section II-B),
- develop an optimized Succinct Non-Interactive Arguments of Knowledge (SNARKs) constraint system for operations, such as moving average and sliding variance, that enable even embedded devices to generate SNARKs proof, shown in (Section V-B), and
- compare our approach with the state of the art regarding closeness to the source of the individual trust metrics (Section VI).

II. Metrics for comparison

A. Trust Metrics

We use the trust evaluation scheme from [1]. They have performed an extensive survey of the trust techniques used in the field of Internet of Things (IoT) and they have categorized them into what they call *trust metrics*. They use these trust metrics to evaluate entire systems. Three important metrics relevant for us are:

1) Data perception trust (DPT) which focuses on the plausibility verification of the streaming sensor data.
2) Identity trust (IT) which is a trust metric that shows that the sensor values are authenticated.
3) System security and robustness (SSR) which is assigned to a hardened system in terms of security.

Further trust metrics and their explanation can be reached in the aforementioned paper [1]. We extend their trust metrics by what we call **DT**. It is the ease of incorporating trust in multiple parties of an ecosystem. This trust metric is becoming increasing relevant in the trend of distributed and decentralized architectures.

B. Introducing closeness to source factor

This factor is used to numerically calculate the closeness to source of the respective trust metric. This is presented in Figure 2. Here a single-source sensor node (SSN) is sending data to the *consumer* via a *gateway*. $Closeness\ to\ source\ factor = H - h$ where H is the total number of hops according to the network architecture from the consumer to the sensor node (in the Figure 2 the total number of hops is $H = 4$). The measure h is the entity from which a particular trust technique is introduced in the data with the consumer as the reference point. Our aim is to minimize this factor for each trust metric, effectively introducing trust from as close to the source as possible.

III. Related Work

We evaluated the trust technique approaches in our literature review, based on the trust metrics and closeness to source factor introduced in the previous section II. We have formulated the comparison in Table I. It shows the trust metrics for each approach along with the number of sources they use, the architecture they adopt, and whether the approach is a data oracle (service that transport data securely to or from a blockchain) or not. It also shows the minimum closeness to source factor a particular trust metric achieves in the approach. Lastly, at the end of table we also present *our approach* to show how it compares with other approaches.

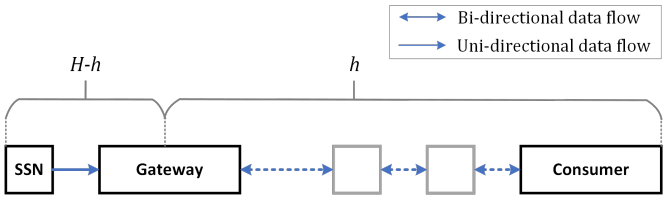

Figure 2: A single-sourced sensor node (SSN) is connected to a gateway, and is sending data sensor data to the consumer. The data reaches the consumer after a few more network hops. The letter H represents the total number of network hops between the source and the consumer. The small h is used to describe the network component from where the trust technique is incorporated.

The first approach, which we consider state of the art, Reliable, Resilient and Secure IoT for Smart City Applications (RERUM) project, presents statistical computations to achieve DPT trust metric. The also authenticate the data from the sensor nodes. Helping them achieve IT trust metric right at the source of the data. However, the threat of manipulation of data by malicious parties is not addressed by their approach. *Javed12* [3] suggests involving signal processing to enhance the reliability of the data from the sensors but does not focus on constrained devices. *Sicari13* [4] admittedly, introduces many levels of trust from the source but they rely on having multiple sources of data and do not focus on the aspect of DT.

In the domain of data oracles, *Witnet* [5] and *Astrea* [6] address the aspect of DT in the content of data using their protocol. Which relays data between the source and the consumer. The source of data for their approach is websites. Therefore, their approach does not incorporate trust from the actual source which might be a sensor on site sending daily temperatures to the website. Similarly, Town Crier (TC)/*Chainlink* [7], [8] and *provable* [9] provide trust techniques for secure transportation of data to the blockchain. They also utilize TEE in their approach. However, manipulation of data by a malicious actor is still possible. *Zk-AuthFeed* [10] is similar to our approach, where the authors use AD-SNARKs feeding data into smart contracts on the Ethereum blockchain. However, their approach is focused on the performance aspects of AD-SNARKs and how they scale with more constraints. Their focus is not on bringing this technique to constrained devices that are generating sensor data. Lastly, we present our approach which achieves multiple trust metrics in comparison to other techniques. The details on how our approach achieves them is explained in section VI-B.

IV. Background

This section provides a background on the building blocks of our approach; Public Key Infrastructure (PKI), TEE, SNARKs, and time series statistics.

A. Public key infrastructure

PKI is a system that is used to create, manage, store, and distribute digital certificates of entities in a networked

Technique Name	Trust Metrics	No. of sources	Arch. type	Oracle Type	H	$H-h$
RERUM [2]	PP, SSR, IT, DPT	single, multiple	centralized	✘	≥ 2	0: IT, DPT
Javed12 [3]	DPT	multiple	✘	✘	✘	✘
Sicari13 [4]	TRD, DPT, DFMT (only fusion), SSR (Partial)	multiple	✘	✘	≥ 3	0: DFMT, PP
Witnet [5]	**DT**, DTCT	single, multiple	decentralized	consensus	2	1: DTCT
Astrea [6]	**DT**	single	decentralized	consensus	2	1: **DT**
TownCrier, Chainlink [7], [8]	DTCT, IT, SSR	single, multiple	decentralized	hardware + software	2	1: DTCT
Provable [9]	DTCT, SSR	single	centralized	software	2	1: DTCT
Zk-AuthFeed [10]	**DT**, IT, PP	single	decentralized	software	2	1: 0: IT
Our Approach	**DT**, SSR, DPT, IT, PP (Partial)	single	decentralized	hardware + software	2	0: **All**

Table I: Evaluation of the current state of the art in trust techniques in comparison to *our approach*. Trust metrics are based on the work of the authors in [1] and is extended by our trust metric DT introduced in subsection II-A. The table highlights the number of data sources and the architecture type each approach adopts. It also presents the oracle type if applicable. Lastly, for each approach, the table presents the trust metric with the minimum closeness to source, shown in the last column.

application [11]. The key goal of the PKI system is to provide a trustworthy process of binding the public key to an entity. It is achieved with the issuance of X.509 certificates to the entity with the digital signature of the *Certificate Authority*. PKI enables us to achieve *identity trust* by authenticating the data packets. Ensuring us that the data is coming from a trusted source by verifying the signature and the certification path.

B. Trusted Execution Environment

For mobile or low powered devices, ARM TrustZone (ARM TZ) is the industry leader by the ARM Holdings [12]. Since most of the constrained device are powered by ARM microcontrollers ARM TZ is our TEE of interest. ARM TZ has the concept of a *non-secure world* and a *secure world*. The non-secure world is restricted by hardware to access the secure world. A codebase for critical tasks, typically called trusted computing base (TCB), is loaded inside the secure world. Creating an extra layer of defense from the outside world while the code is executed. Our approach is not limited to the use of ARM TZ, any other equivalent technology can be used. However, to develop trust in the sensor readings from as close to the source as possible, the presence of a TEE on the sensor node is imperative.

C. Succinct Non-Interactive Arguments of Knowledge

SNARKs are a category of non-interactive proving systems under the umbrella of Zero-Knowledge Proofs (ZKPs). In our approach, we use AD-SNARKs [13], which is a variant of this proving system based on BCTV14 [14]. SNARKs are a two party system, where one party is the *prover*. Which creates proof of knowledge of a mathematical calculation to the other party known as the *verifier*. The mathematical calculation is agreed upon in the *generation phase* of the SNARKs. The generator algorithm produces two keys; circuit evaluation key EK_C and circuit verification key VK_C. They are used to generate and verify the proofs of the mathematical calculation, respectively. Both of these keys do not have to be confidential and not restricted to any particular entity. Anyone can use these keys to create and verify proofs of the mathematical computation they are set for.

The state-of-the-art SNARKs protocol, uses an optimization and reduces the computation in circuits to the form called quadratic arithmetic program (QAP). Which is a tuple $(\vec{A}, \vec{B}, \vec{C}, Z)$ of size m and degree d over a field $\mathbb{F}$, where $\vec{A}, \vec{B}, \vec{C}$ are three vectors each of size $m+1$ polynomials in $\mathbb{F}^{\leq d}[z]$ and $Z \in \mathbb{F}[z]$ has degree of exactly d. Here $\mathbb{F}[z]$ denotes the ring of univariate polynomials over the field $\mathbb{F}$ and $\mathbb{F}^{\leq d}[z]$ denotes to the subring of polynomials of degree $\leq d$. This is known as the circuit satisfaction, which is as follows:

Definition 1: The **satisfaction problem** of size-m $QAP(\vec{A}, \vec{B}, \vec{C}, Z)$ is a relation $\mathcal{R}_{(\vec{A},\vec{B},\vec{C},Z)}$ of pairs $(\vec{x}, \vec{s})$ such that (i) $\vec{x} \in \mathbb{F}^n, \vec{s} \in \mathbb{F}^m$ and $n \leq m$; (ii) $x_i = s_i$ for $i \in [n]$ and (iii) the polynomial $Z(z)$ divides the following one completely:

$$(A_0(z) + \sum_{i=1}^{m} s_i A_i(z)) \cdot (B_0(z) + \sum_{i=1}^{m} s_i A_i(z)) - (C_0(z) + \sum_{i=1}^{m} s_i C_i(z)). \tag{1}$$

The relation $\mathcal{R}_{(\vec{A},\vec{B},\vec{C},Z)}$ language is denoted by $\mathcal{L}_{(\vec{A},\vec{B},\vec{C},Z)}$.

Entry point to the SNARKs is through the language called Rank-1 Constraint System (R1CS) (commonly known as constraints or constraint system), which are a group of vectors $\vec{f}, \vec{g}, \vec{h}$. Each vector $\vec{f}, \vec{g}$ and $\vec{h}$ represents the operand according to the decided mapping from the flattened statements. The R1CS is transformed into vector of polynomials turning the circuit satisfaction problem into the QAP satisfaction problem, represented in equation 1. Mathematical circuit represented in the form of constraints are turned into polynomials. Which are eventually used to prove the honest computation of the constraints.

For our approach, we want our sensor data to be digitally signed at the source which would help us achieve IT trust metric. Therefore, we use the AD-SNARKs [13] variant, since it allows signing of data sources used in constraints. The authors of AD-SNARKs incorporated the signing and verification process of the signature as a part of the SNARKs protocol.

Consequently, achieving performance roughly equivalent to the original SNARKs protocol.

D. Time Series Statistics

We use statistical measures that provide information about streaming data from the sensor attached to the constrained device. The measures which we use for our approach are inspired by the RERUM project [2] and are introduced below.

a) Exponential Moving average: The simplest measure that can be calculated is called the *mean* of the given data. However, this measure when calculated for the entire lifetime of the sensor node would not prove to be useful because the environment is constantly changing and the past values tend to lost their importance. Therefore, a measure that is able to capture this change over a predefined window of past values is called *moving average*. *Exponential smoothing* is applied to moving average resulting in a smooth trend line which follows the data from the sensor, avoiding the erratic behavior and we end up with a measure called exponential moving average (EMA). It is calculated according to the following equation. $\text{EMA} = \bar{x}_t = \eta x_t + (1-\eta)\bar{x}_{t-1}$.

Where η is the smoothing factor $0 < \eta < 1$, and is usually in the order of 0.05 or smaller.

b) Sliding Variance: A measure that would provide us with the information about the spread of the data set would be by calculating the *variance*. The exponential moving average x_t introduced in the last paragraph is a stream where we do not have a fixed data set. Therefore, we use a streaming value that provides the spread of the data in the chosen window. This measure is called sliding variance (SV) and is described by the mathematical relationship $\text{SV} = \hat{x}_t = \eta(x_t - \bar{x}_t)^2 + (1-\eta)\hat{x}_{t-1}$

Using both of these statistical measures, we can represent the streaming data in a condensed form that is suitable for many applications such as forecasting of the underlying data. Based on these statistical measures we develop our constraints for AD-SNARKs. If there is a requirement then this approach can be extended to other statistical measures as well, consequently extending the number of constraints for the AD-SNARKs system. As an example, for a seismograph a peak detection computation might be more relevant and the relevant constraints can be developed for it.

V. Our Approach

In this section, we use the building blocks from the previous sections to explain our approach of introducing multiple layers of trust in the data from the single-sourced sensors.

A. Reading sensor value securely

We secure the reading of sensor data right from the source by utilizing the TEE, specifically the ARM TZ. It allows the configuration of General Purpose Input Output (GPIO)s and associates them with a particular *world* (secure or normal). Reading the sensor value directly in the ARM TZ enables us to minimize the closeness factor to 0 with respect to SSR trust metric. To explain the process of reading the sensor values better, we have a high level block diagram of the physical

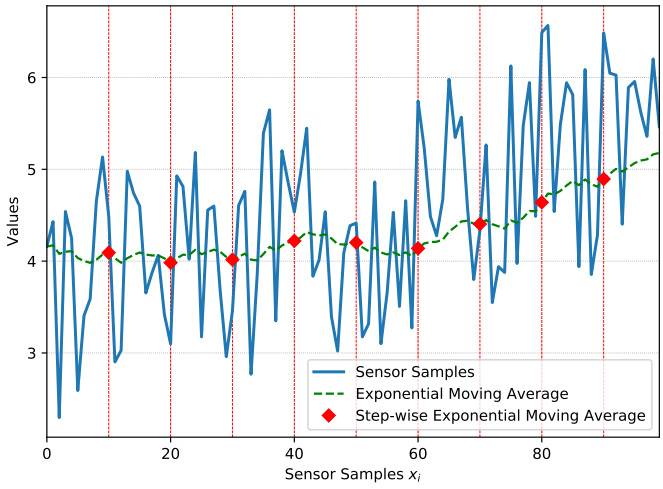

Figure 3: Plot showing step-wise exponential moving average for sensor samples. It has a window of $N = 10$ samples per step j. With each step shown as red vertical dashed lines.

architecture of the sensor node in Figure 1. The TEE is made up of the secure chunks of memory called the *secure world*, attached to the micro-controller core, which runs the code from the secure memory locations. The sensor is physically attached to one of the GPIO pins of the micro-controller. Which is configured to send the contents of the GPIO register to the secure world. The software to read sensor values from the sensor should be part of the TCB so that it is executed from a secure memory location, isolated from the normal malicious world. The configuration and development for the TEE depends on the micro-controller of the constrained device, therefore it would vary with the choice of micro-controller. Our focus was on ARM TZ therefore we found this tutorial to be quite helpful [15].

B. Developing the constraint system for AD-SNARKs

In the previous section (IV-D) we introduced the time series statistics which are used to develop the constraints for AD-SNARKs. We can use the mathematical representation of EMA and SV directly to create a constraint system for AD-SNARKs. Ending up with a total of 11 constraints and we call this constraint system as constraint system 1 (CS1). However, if we want to increase the data density, the use of this constraint system significantly increase the packet size and the time required to create proofs. Data density is defined as $Data\ density = kx_t/m$. Here k is the number of sensor readings, x_t is the sensor readings and m are the number of packets sent from the sensor node. The units of data density for 1 reading per packet are defined as $1\,\text{rds}\,\text{pkt}^{-1}$.

In order to mitigate these issues we use an alternative representation of the same statistical measures in the form of summation notation. This enables us to provide the same information in a step-wise fashion and achieve a higher data density in the packets. There is data loss in between steps but this is not an issue because this provides partial privacy preserving capabilities since all of the sensor data never leaves

the constrained device. The concept is visualized in Figure 3, where the step-wise exponential moving average (SEMA) of the streaming sensor samples are shown. Instead of calculating the EMA at every sampled value, we accumulate the sensor samples depending on window we want to have in each step and calculate the SEMA from these values. The EMA can be transformed in to SEMA and it is represented with the equation below. Where N are the number of sensor samples and j is the step number.

$$\bar{x}_{jN} = \sum_{i=0}^{N-1} \eta(1-\eta)^i x_{(jN-i)} + (1-\eta)^N \bar{x}_{(j-1)N} \quad (2)$$

Similarly for SV we have step-wise sliding variance (SSV) which is represented as follows:

$$\hat{x}_{jN} = \sum_{i=0}^{N-1} \eta(1-\eta)^i (x_{(N-i)} - \bar{x}_{(N-i)})^2 + (1-\eta)^N \hat{x}_{(j-1)N} \quad (3)$$

To develop our constraint system we expand our equations into flattened statements. These are three operand based equations, such as $a + b = c$, with a mathematical $(+)$ and an equality operator $(=)$. Using the equations for SEMA and SSV above, we reach at the following flattened statements:

$$\begin{aligned} etaWindow * \bar{x}_{t-1} &= sema1 \\ xFused + sema1 &= seMovAvg \\ etaWindow * \hat{x}_{t-1} &= ssv1 \\ avgSqFused + ssv1 &= sSliVar \end{aligned} \quad (4)$$

Once we have the flattened statements we need to take care of two important aspects before we reach our R1CS compliant constraints. Firstly, there should be no linear dependencies between the flattened statements. If it exist then only one such equation should be present. Secondly, the variables and constants for the constraint system need to be integers because they are elements of the prime field $\mathcal{F}_p$. Therefore, we need to scale our flattened statements for our variable because sensor data is represented in floating point. The final scaling is represented by a term which we call scaling factor (SF), which is required by the consumer to get to the actual value with decimals.

Once we ensure these two aspects, we get the constraint system in the R1CS form as an input for our AD-SNARKs generator algorithm. Our optimized constraint system (CS Optimized) for the equations above is given below:

$$\begin{aligned} etaWindow * \bar{x}_{t-1} &= sema1 \\ xFused * 100 &= scaled\_xFused \\ (scaled\_xFused + sema1) * 1 &= \mathbf{seMovAvg} \\ etaWindow * \hat{x}_{t-1} &= ssv1 \\ avgSqFused * 100 &= scaled\_avgSqFused \\ (scaled\_avgSqFused + ssv1) * 1 &= \mathbf{sSliVar} \end{aligned} \quad (5)$$

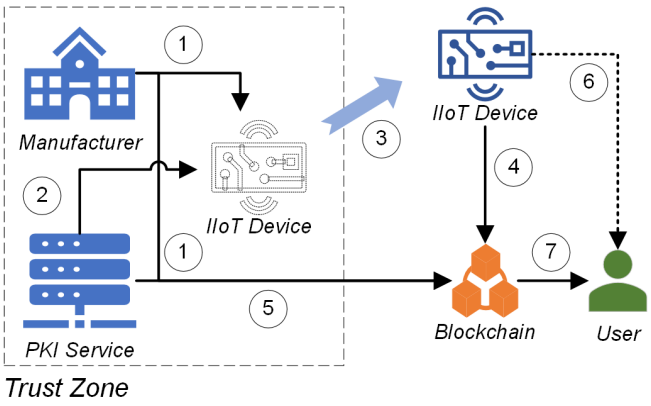

Figure 4: An illustration showing the lifecycle of an IIoT device after its production. The Manufacturer is trusted to run the trusted setup of AD-SNARKs and places the circuit evaluation key EK_C in the device. It places the circuit verification key VK_C and SF in the blockchain. The manufacturer also issues a PKI certificate for the public key of the IIoT device so that the data source can be authenticated. Once the IIoT device is transported and configured, it starts to gather sensor data and pushes it to the consumer along with the proofs.

Please note that the colors of each operand is based on the vectors of R1CS system as it was introduced in the SNARKs introduction in Section IV-C. The variables in bold are public output values and would be transmitted in the packets for the consumer. They are accompanied by the authenticated data and a proof, which would prove the provenance of data and verification of the computation respectively.

C. System Overview

In this section, we describe our approach using the system architecture in Figure 4 and how all these components work together. We have multiple parties in the system (similar to our problem in the introduction I): 1. Manufacturer of the IIoT device, 2. PKI system operated by the manufacturer, 3. the IIoT device, 4. the consumer of data from the IIoT device, which in our case is the blockchain and 5. the user which needs trustworthy data. The entire process is divided into 7 steps which are show in Figure 4 and are explained below.

a) Step 1: The entire process starts with the manufacturer of the device who is the root of trust in our system. The manufacturer burns the device with its initial TCB. The authentication keys should be generated on the device inside the secure world, only the public key leaves the secure world which is required as an input for the trusted setup of AD-SNARKs. This way the private key never leaves the secure world and consequently its leakage is prevented. Manufacturer executes the trusted setup of AD-SNARKs, which generates circuit evaluation key EK_C and circuit verification key VK_C for the CS Optimized that we have defined in Section V-B. The manufacturer sends the circuit verification key VK_C, SF and the authentication public key to the consumer (blockchain) so that they are available for all interested parties to verify the proofs and get the trustworthy values of SEMA and SSV.

b) Step 2: The PKI service issues an X.509 certificate to the device. The authentication public key is embedded in the

certificate so that any party communicating with the device can verify that it is an authentic device from a trusted manufacturer. This public key is also utilized in verifying the proof itself which would be available at the consumer and the end user.

c) Step 3: The IIoT device is transported to the field where it is deployed. It is usually connected to an IoT gateway (which is not shown in the diagram for brevity). Once the device is active and configured to connect to the consumer, it starts reading the sensor data directly inside the TEE on the device.

d) Step 4: The input values for the proving algorithm are signed within the TEE and then passed to the normal world, where the signature is verified and if this check passes then the proof is generated using the circuit evaluation key EK_C. The proof, authenticated data and the public values of SEMA and SSV are pushed to the consumer (blockchain). If the device is not powerful enough to compute the proof, then this process can be shifted to the more powerful IoT gateway. The sensor device only needs to provide the authenticated data which would be used as an input for proof generation at the gateway.

e) Step 5: Using a micro-service at the consumer end (or smart contracts in the blockchain), the device is on-boarded by validating the signature on the certificate of the device and performing a proof of private key possession. The signature on the certificate can be validated by requesting the public key of the manufacturer from the PKI service exposed to the consumer. On-boarding needs to be done only once. Once the device is authenticated, the consumer can verify the proof by using circuit verification key VK_C, public authentication key and the authenticated data. If this check passes, it stores the current proof along with the public values for any interested party to read. If an online connection is not available, the proofs can be stored at the gateway, or on the device for later verification.

f) Step 6: This is an optional step for the end user that wants to on-board the device in its system. The device can be on-boarded by performing the actions written in *Step 5*.

g) Step 7: The end user pulls the proof, circuit verification key VK_C, authentication public key (of device) and the authenticated data to verify that the public SEMA & SSV values available at the consumer. It uses this data to authenticate the sensor reading to determine that the source is trustworthy. It then verifies the proof to ascertain that the mathematical operations on the sensor readings were done properly. The details of the algorithms used and their implementation are explained in the section VI.

Applying this approach to our example problem of the "vaccine transport" from section I we can highlight the benefits of our approach. The end user from our system overview in Figure 4 would be the party purchasing the vaccines. By using our proposed solution for recording the temperature of the shipment throughout its journey, the end user will be able to determine conclusively how the temperature varied throughout the journey by verifying the proofs at each step along with the public values of SEMA and SSV. Having the proof enables us to achieve the trust metric of DT, where any entity can create or verify the proofs using the circuit evaluation key EK_C and circuit verification key VK_C. Furthermore, the end user can be certain that the temperature readings were coming from a trusted device because input data to SNARKs is signed by the prover and verifiable. With the use of TEE, signing of sensor data and providing proofs of computation we are able to provide multiple layers of trust in our approach.

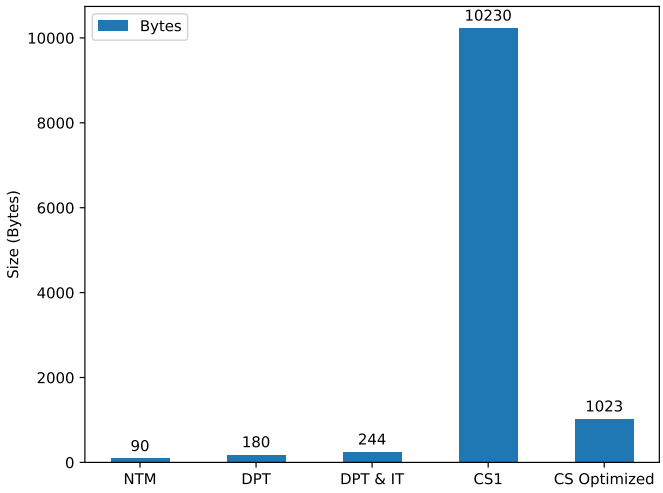

Figure 5: This bar graph compares the payload sizes as trust techniques are added to the packet. Each payload has a data density of $10\,\mathrm{rds\,pkt^{-1}}$. From left to the right, the bars represent, the size of the payload with *no trust metrics* applied. Followed by addition of *data perception trust*, and then *identity trust* is added to the payload. Lastly, we compare our payload of our CS Optimized with the payload of CS1. The exact contents of packets are explained in Subsection VI-A

VI. Implementation Details and Evaluation

A. Implementation details

Each trust technique was implemented and tested on a general purpose laptop running Ubuntu 18.04 LTS. The specifications of the laptop are Intel Core i7-7500U, 2.7GHz, 8GB RAM and x64 Architecture. The sensor was a simulated temperature sensor that was used as an input for all trust techniques. Furthermore, no data transmission timings were measured because our approach is packet based but protocol agnostic. Lastly, all implementations were carried out in C/C++ and compiled using the `gcc` compiler.

We compared the packet's payload sizes and the timing performance of our approach with other common trust techniques that we explored in our literature review. For each trust technique we construct the payload with the data represented as strings and a data density of $10\,\mathrm{rds\,pkt^{-1}}$.

For control we use a payload with no trust metric (NTM) applied and it is constructed with ten consecutive values from the sensor stacked in front of each other. The first trust technique, that is commonly used, is to add DPT. The payload has ten consecutive EMA and SV values stacked in front of each other. The state of the art is to add IT on top of the previous trust technique as shown by the authors in [2]. This is achieved by adding a digital signature to the payload. For

	Moving Average (ns)	Sliding Variance (ns)	Auth. (ms)	AuthVer. (ms)	Proof Gen. (ms)	Proof Ver. (ms)	Total (ms)
DPT	512	563	✘	✘	✘	✘	0.251
DPT & IT	444	465	.63	1.57	✘	✘	2.452
CS1	5433	7256	27.24	28.03	98.63	362.92	517.084
CS Optimized	5157	34080	2.85	3.07	7.94	37.28	51.407

Table II: Comparison of the timing performance of each trust technique introduced in subsection VI-A. Every timing measurements started from the measurement of reading data from the simulated temperature sensor till the packet's payload construction and it is an average of ten values. The specification of the laptop are: Intel Core i7-7500U, 2.7GHz, 8GB RAM and x64 Architecture. Timing of data transmission was excluded from the measurements.

signing we use *Ed25519* elliptic curve signatures so that we can make a fair comparison with the use of AD-SNARKs because they also use elliptic curve signatures for authentication.

Now coming to our approach with the CS Optimized, the payload includes the public values of SEMA and SSV calculated with ten consecutive sensor readings. Furthermore, it includes the authenticated data and the proof generated with the AD-SNARKs algorithm. We also compare our CS Optimized to the CS1 to show the improvement that we are able to achieve.

The library used for implementing AD-SNARKs is libsnark [16] from Succinct Computational Integrity and Privacy Research (SCIPR) lab. This library has multiple variants of the SNARKs protocol. The GROTH 16 [17] protocol achieves the best performance but we are dependent on BCTV14 [14] because AD-SNARKs extends on this implementation. However, the timing performance improvements by using the GROTH 16 would be minimal for us because of the low number of constraints.

Furthermore, the pairing operation used by AD-SNARKs protocol is based on elliptic curves. Changing the elliptic curve also influences the sizes of the payload. Since our approach is a proof of concept, we focus on the curve that provides the maximum bit security. We used the *Barreto Naehrig* curve which provides us with 128 bits of security. In comparison to other curves, it produces the largest payload and slower timing performance.

B. Evaluation

The sizes of the payload with each trust metric being applied, is shown in Figure 5. Going from the left bar to the right, we can see that as more layers of trust are added the packet size gradually increases. With our approach using the CS Optimized we get multiple layers of trust. Namely they are DPT, IT and DT.

Furthermore, the Figure 5 also compares the payload sizes of the two constraints systems CS1 and CS Optimized. We can see that our CS Optimized achieves a tenfold size reduction compared to CS1. Therefore, our CS Optimized is much well suited for constrained devices. With an additional 779 Bytes, our approach offers another layer of trust on top of DPT & IT. We do want to point out that there is data loss in between steps but this is not an issue because this provides a partial privacy preserving capabilities because the exact sensor values never leave the constrained device.

In our payload for CS Optimized, the major chunk is taken by the authenticated data elements. This takes up 576 Bytes of the payload. The proof for calculating the public values of SEMA and SSV is of constant size, take up slightly less than 383 Bytes. Lastly, we have the public values for which we have designed this protocol. Each take up 32 Bytes, hence a total of 64 Bytes. Bringing the total payload size to 1023 Bytes.

The timing performance of the aforementioned trust techniques is tabulated in Table II. The sensor read times were similar for all of them because we were aiming at a data density of $10\,\mathrm{rds}\,\mathrm{pkt}^{-1}$ and it amounts to roughly 250 ms. This would be the time required for NTM but it is not shown in the table to save space. Furthermore, the measurement of time began with the first sensor data read and ended when the complete packet's payload was constructed for a particular trust technique. In the table the time values are an average of ten values to average out the variations by the operating system.

By comparing the payload size and the timing performance of the trust techniques we can see that as another layer of trust is added to the communication packet, the time to construct a packet and the payload size increases. With our proposed approach of CS Optimized, it would need an additional 779 Bytes and an extra time of 11.68ms on top of the state of the art trust techniques. With this additional cost we get a huge benefit of adding another layer of trust, which is (DT). That provides, any party reading the data, additional trust in the correctness of computation of the DPT trust metrics. Additionally, with the use of TEE in reading the sensor data we are also able to achieve another layer of trust which is SSR. We can see that our approach enhances the trust in the sensor data in the scenario where the sensors are operated by other parties and can be potentially malicious.

In our approach we were able to achieve the following trust metrics with the closeness to source factor:

- Achieving SSR with $H - h = 0$: Our approach promotes the use of a TEE. The ARM TZ on the ARM M-23 microcontroller enables us to achieve SSR trust metric right from the source.
- Achieving DPT with $H - h = 0$: By calculating SEMA and SSV on raw sensor data. This is computed on the sensor node itself.
- Achieving IT with $H - h = 0$: With the help of the PKI, we are able to achieve the trust metric IT right at the

source because the public values are signed inside the TEE.

- Achieving privacy preserving (PP) (partial) with $H - h = 0$: The way the constraint system is designed, we are able to fuse together the consecutive input values x_t, therefore making it difficult for the attacker to find the fine-grained values. This is helpful in scenarios where the sensor data is measuring private data such as the power consumption of an apartment.
- Achieving DT with $H - h = 0$: The proof generation lets us achieve the trust metric of DT because it is able to provide any party with a proofs that ensures proper computation. The proof can be generated at the device achieving the closeness to source of 0. If this trust metric is computed at the gateway or at a later stage then the closeness to source would be 1 or higher depending on the hop where it is computed.

Our approach shows how SNARKs implementation can be made feasible for constrained devices. However, it is not yet fully portable to devices that cannot run an operating system. The libsnark library [16] that we use, is written in C/C++ language but it cannot be ported to micro-controllers because it uses a number of operating systems helper functions for memory profiling and timing operations. There is a SNARKs library for embedded systems, but it does not support the AD-SNARKs variant and adding its support to the library would be significant work in itself [18].

The libsnark library does shows that it can be compiled on ARM processors that are capable of running Linux operating systems. Hence, we can generate SNARKs on IoT gateway class processors. However, the AD-SNARKs variant that we are using, cannot be compiled for ARM processors because of its dependency on *SUPERCOP* library requires x86 or x64 processors. Notwithstanding, we did compile our CS Optimized on an ARM Cortex A53 processor using the BCTV14 SNARKs variant. We did lose the support for authenticated data but we were able to measure the performance which should roughly be equivalent to that of AD-SNARKs. The platform we used was a Raspberry Pi 3 Model B with a quad core processor running at 1.2 GHz with 1GB RAM. The platform was running Ubuntu server 22.04. We recorded a proving time of nearly 60 ms and the verification of proof takes roughly 230ms. This result conforms to our expectations because as the processing power reduces, the proving time would increase.

VII. Conclusion

In this paper, we aimed to present a prototype for incorporating trust in a single sourced IoT device. Our approach uses multiple trust metrics including the use of AD-SNARKs, which is a variant of SNARKs. We presented an optimized constraint system that can be used to generate proofs on a constrained device for the SEMA and SSV statistical measures.

Our approach shows promising results in context of constrained devices with proof generation times in the range of milliseconds. Although, we were unable to port the approach to a Raspberry Pi 3 completely, our implementation shows the feasibility and potential of our method.

We expect libraries that are optimized for constrained devices to appear in the near future, which would enable further research directions for even better constraint systems.

We believe that we will see a rise in decentralized and distributed architectures, because of the large number of small connected devices that are constantly being produced. To cater to security and privacy concerns in such architectures, the technology of SNARKs can play a key role as we have demonstrated in this paper.

Acknowledgment

We express our gratitude to Dr. Martin Wimmer and Dr. Prabhakaran Kasinathan for their help in refining the paper.

References

[1] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for internet of things," *Journal of network and computer applications*, vol. 42, pp. 120–134, 2014.

[2] D. López, J. Cuellar, R. Staudemeyer, P. Charalampidis, A. Fragkiadakis, P. Kasinathan, H. Pöhls, S. Suppan, E. Tragos, and R. Weber, "Modelling the trustworthiness of the iot," 04 2016.

[3] N. Javed and T. Wolf, "Automated sensor verification using outlier detection in the internet of things," in *2012 32nd International Conference on Distributed Computing Systems Workshops*. IEEE, 2012, pp. 291–296.

[4] S. Sicari, A. Coen-Porisini, and R. Riggio, "Dare: evaluating data accuracy using node reputation," *Computer Networks*, vol. 57, no. 15, pp. 3098–3111, 2013.

[5] "The witnet protocol," https://witnet.io/, accessed: 2023-06-15.

[6] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain oracle," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1145–1152.

[7] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, 2016, pp. 270–282.

[8] "Chain link," https://chain.link/, accessed: 2023-06-15.

[9] "The provable blockchain oracle for modern dapps," https://provable.xyz/, accessed: 2023-06-15.

[10] Z. Wan, Z. Guan, Y. Zhou, and K. Ren, "Zk-authfeed: How to feed authenticated data into smart contract with zero knowledge," in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 83–90.

[11] J. Weise, "Public key infrastructure overview," *Sun BluePrints OnLine, August*, pp. 1–27, 2001.

[12] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[13] M. Backes, M. Barbosa, D. Fiore, and R. M. Reischuk, "Adsnark: nearly practical and privacy-preserving proofs on authenticated data," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 271–286.

[14] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 781–796.

[15] D. Ehnes, "Building secure iot with arm cortex m23 microcontroller and trustzone," https://medium.com/magicofc/gas-for-your-secure-iot-fire-a-tutorial-for-crypto-on-a-cortex-m23-%C2%B5c-bfb27017df21, accessed: 2023-06-15.

[16] S. Lab, "Libsnark," https://github.com/scipr-lab/libsnark, accessed: 2023-06-15.

[17] J. Groth, "On the size of pairing-based non-interactive arguments," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2016, pp. 305–326.

[18] X. Salleras, "Zpie: Zero-knowledge proofs in embedded systems," https://github.com/xevisalle/zpie, accessed: 2023-06-15.