Extending On-chain Trust to Off-chain – A Trustworthy Vaccine Shipping Example

***Abstract*—Blockchain creates a secure environment on top of strict cryptographic assumptions and objective security proofs. It permits on-chain interactions to have many trustworthy properties such as traceability, transparency, and accountability. However, current blockchain trustworthiness is only confined to on-chain, creating a "trust gap" to the physical, off-chain environment. This is due to the lack of a scheme that can truthfully reflect the physical world in a real-time and consistent manner. Such an absence hinders further real-world blockchain applications, especially for security-sensitive ones.**

In this paper, we propose a scheme to extend blockchain trust from on-chain to off-chain, and implement the full system taking trustworthy vaccine transportation as an example. Our scheme consists of 1) a TEE-enabled Trusted Environment Monitoring System that continuously senses the inside of the vaccine box through trusted sensors and generates anti-forgery data; and 2) a consistency protocol to upload the environment status data from the TEE system to blockchain in an orderly, consistent, and fault-tolerating fashion. Our experiment records the internal status of the vaccine shipping box during transportation. The results indicate that our system incurs a maximum of 8 seconds to have the sensed data available in the blockchain, and no adversary can tamper with the vaccine in any way without being captured.

I. Introduction

Blockchain is a secure environment that permits certain interactions within it to be trustworthy. It is built on cryptographic assumptions and proofs to guarantee objective security and trustworthiness. However, currently, its trustworthiness is confined within the blockchain environment only, and can be hardly extended into the off-chain physical world. This is manifested by the lack of strong guarantee that blockchain data truthfully reflects the physical object it is bound, or that blockchain control instructions are always correctly deployed to the device it is managing. This "trust gap" between on-chain and off-chain undermines the trustworthiness property of blockchain, especially when applying it to security-sensitive physical world related scenarios.

Taking vaccine transportation as an example – vaccine is critical to protect citizen health, thus keeping its physical distribution process highly traceable, transparent, and trustworthy is of paramount significance to public interest. It is one of the critical scenarios where digital data collected for monitoring the vaccine transportaiton should strictly reflect the physical world activities. However, we have seen many tragic cases where the vaccines were damaged or counterfeited due to the lack of highly trustworthy monitoring systems. Data were mostly manually reported and entered, thus we don't know if the vaccine was *actually* exposed to hazardous environments, defective or tampered with. During a surprise inspection to the rabies vaccine produced by Changchun Changsheng Ltd, China's National Drug Administration (NDA) found more than 250,000 doses of a vaccine were out of the production standards and the online data did not match with the actual product [1][2]. The Centers for Disease Control and Prevention of America (CDC) mandates the vaccine to be stored between 2°C and 8°C. However, Nelson *et al.* found that Hepatitis B vaccines in Indonesia was common to inadvertently freezing and thus harming the bioactivity, though the use of trusted vaccine vial monitors can significantly reduce the risk of heat-damage of the vaccines [3]. This "physical intraceability" challenge are common in many other security-sensitive applications in real life, and blockchain is not the silver bullet to the problem due to the current trust gap between on-chain environment and off-chain physical world.

To address this problem, we resort to periodical trustworthy sensing, creating data anti-forgery proofs, and achieving real-time upload guarantee to blockchain and error recovery against data losses. Our scheme consists of 1) a TEE-enabled Trusted Environment Monitoring System that periodically senses the physical environment, creates anti-forgery data records, and uploads the records to blockchain; and 2) a consistency protocol to upload the records from the TEE system to the blockchain in an orderly, consistent and fault-tolerating fashion. We make careful security designs to ensure 1) no adversary can forge a fake data as all are verified authentic and 2) no adversary can physically violate the security requirements without being captured.

The contributions of this paper are summarized as follows.

1) We fully implement a trusted monitoring system to support our trustworthy vaccine transportation in real world. We develop the system using the low-cost Cortex M33 Trusted Execution Environment (TEE) microcontroller, which enforces strict physical isolation, thus achieving high security guarantee while keeping the system practically cheap for IoT applications.
2) We extend the on-chain trust to off-chain by designing a protocol to achieve the real-time data consistency between physical objects and their corresponding digital entities. We also analyze the time upper bound for data consistency considering general blockchain systems.
3) Our system is invulnerable to long-range vulnerabilities. Unless physically destroyed, any tampering with the system or the monitored physical object is to be recorded with non-repudiation and traceability. We also provide a fault-tolerance and error recovery solution against record losses caused by transmission failures without relaxing

any security requirement.

The rest of this paper is arranged as follows. Section II presents the necessary definitions and basic assumptions. Section III outlines the vaccine transportation system under our consideration to illustrate our design. Section IV describes the in-depth technical details of our TEE-enabled Trusted Environment Monitoring System and the consistency protocol. Section V demonstrates our full system and visualizes the critical data. We finally summarize the related work in Section VI and conclude the paper in Section VII.

II. Definitions and Models

In this paper, we intend to extend the on-chain trust to off-chain, and realize real-time data consistency between them, with high security confidence. To achieve this goal, we need to define what is real-time data consistency, what is a physical event of interest and its corresponding digital entity, and how to achieve the consistency goal.

A. Basic Definitions

We say a physical event of interest is an object that can be described as a combination of different types of sensor outputs such as binary, numerical, or multimedia. We denote this combinatory sensor output as C, which is a description towards the event. The selection of the sensors is application-specific, and we actualize our design in Section III. Pattern Γ is defined accordingly to examine whether or not data C follows a correct pattern. In practice, Γ and C share the same structure. The examination is done by computing the deviation between C and Γ. The combination of C and Γ can describe the basic detail of a physical event and how desirable or how legal it is. The physical event of interest then can be recorded as a timed object, denoted as (t, C, π), and uploaded to blockchain, where t is the timestamp when this event object is created and π is a proof of the authenticity of the event or (partly) the trustworthiness of the data. We say a prover P generates such a timed event and the corresponding proof, while a verifier V verifies it. A prover is usually a physical sensor or an IoT device, while the verifier can be any public node in the blockchain system. We then define the trustworthiness of a timed event as follows.

Definition 1 (Trustworthiness of a timed event:)**.** *We say a timed event is trustworthy if it is:*

1) *generated by a trustworthy program, and*
2) *uploaded to the blockchain complying with the real-time consistency requirement.*

Here real-time data consistency between on-chain and off-chain is defined as:

Definition 2 (Real-time On-Chain Off-Chain Data Consistency)**.** *For any event of interest occurred in the physical world at time t, it takes at most δ more time for the blockchain system to return the same timed event, where δ is a small real number that is application-specific.*

As one can see, these definitions mandate a truthful, robust, and highly-responsive system that continuously enforces monitoring towards a physical object, and then uploads the data to blockchain with low latency. This process must also be efficient enough to achieve the real-time requirement. We realize this goal by designing a consistency protocol and implementing the trusted environment monitoring system on top of Trusted Execution Environment (TEE) in Section IV.

A digital entity α can then be formulated as a vector of chronologically consecutive timed events. In other words, a digital entity α fully describes a physical object and its state transitions in 4-dimensional space, by scheduling periodical sensing and data reporting every Δt time.

$$\alpha = (t_0, C_0, \pi_0), \cdots, (t_i, C_i, \pi_i) \tag{1}$$

In order to continuously ensure this digital entity α to truthfully describe the physical object, we define the validity of a digital entity as follows.

Definition 3 (Validity of a digital entity)**.** *We say a digital entity α is valid if for each timed event (t_i, C_i, π_i) in α, the real-time on-chain off-chain data consistency is strictly maintained, and the time interval between two consecutive timed events in blockchain is confined within $\hat{\delta}$, where $\hat{\delta}$ is a real number that does not significantly deviate from Δt, the sensing interval.*

We lastly define a blockchain system as a distributed nework consisting of n processes (or nodes). We consider a general type of blockchain that uses partially-synchronous mode [4][1], where the blockchain is mandated to synchronize blocks at periodical deadlines called Global Standardization Time (GST). Within the time between two consecutive GSTs, denoted as ΔGST, the blockchain transactions are packed into one or more blocks, proposed and verified by most-but-not-all nodes, and not until the next GST will these blocks be guaranteed to be synchronized into every non-faulty node's ledger.

B. Trust Model

In this study, we trust the TEE hardware is secure against any long-range vulnerabilities, and blockchain cannot be manipulated by an adversary.

Blockchain: We say a blockchain system is secure if its consensus output is secure against adversaries' malicious manipulations. The blockchain produces a block every ΔB time and mandates global block synchronization every ΔGST time. With the general partially synchronous assumption mentioned earlier, and the fact that most non-trivial blockchain systems or consensus algorithms provide such proofs against malicious manipulations, one can reasonably make this trust assumption.

TEE: A Trusted Execution Environment (TEE) physically divides a secure zone and a non-secure zone (some may call

[1]Solving fault-tolerant consensus in asynchronous network is considered a difficult problem [5]. While many works claim achievement of asynchronous consensus, they are in fact using a weak, or say partially-synchronous assumption, requiring the network to synchronize at periodical GSTs like Delegate PoS [6] or exist-but-unknown-a-priori upper bounds like PoW [7].

rich-environment zone). Programs in secure zone can only be called by non-secure zone but cannot be modified or explicitly inspected. It is a general consensus among security community that programs within the secure zone are invulnerable against long-range tampering. There is one master secret key mas_sk that is unique to each TEE system and can be used to exclusively authenticate this trusted device to the public. This master secret key cannot be explicitly retrieved or tampered with. We also assume the TEE hardware has a public/private key pair, denoted by TEE_pk and TEE_sk, as a blockchain client needs to be implemented in TEE to communicate with blockchain. Note that we consider physical damage towards TEE out of scope.

III. A Vaccine Transportation Monitoring System

In this section we introduce the vaccine transportation system as an example to demonstrate our design of extending trust from on-chain to off-chain physical world. More specifically, our scheme is designed to make sure that the environment within the vaccine shipping box is monitored in a periodical and trustworthy manner.

We consider an important vaccine is stored inside an insulation box. This box must be transported from A to B via an approved route. The box is sealed and is prohibited from being opened. The temperature inside the box must be kept low and stay stable, in order to preserve the biological activity of the vaccine. We start by defining possible situations that violate the security requirements:

1) the box is opened (may destroy or replace the vaccine),
2) the temperature within the box is abnormal (may nullify the vaccine biological activity),
3) the transportation route is deviated from the predefined one (same as 1),
4) records may be lost (same as 1).

To capture these violations, we place a photosensor, a temperature sensor, and a GPS locator inside the box. As discussed earlier, pattern Γ needs to be predefined to determine whether the data C complies with the security requirements or violates them. In this vaccine transportation system, the photosensor should always output 0 for constant darkness, indicating the box is sealed, thus we can define $\Gamma_P = \{0\}$; the actual binary data of the photosensor is denoted by L, and $L = 1$ if the ambient brightness is greater than the luminous threshold θ and $L = 0$ otherwise. The temperature sensor should always report a steady internal temperature with a max deviation tolerance σ, so $\Gamma_T = [\bar{K} - \sigma, \bar{K} + \sigma]$, where $\bar{K}$ is the target ideal temperature. The GPS sensor records the physical shipping trace of the box, and should follow a geographic pattern from the true origin, following a reasonable trace, to the final destination. We define $\Gamma_G = \{(X_i, Y_i), r_i\}$, which includes a series of checkpoints (X_i, Y_i) by latitudinal and longitudinal coordinates and a safe radius between the box and the closest checkpoint. If the Euclidean distance between each location upload (x_i, y_i) and the nearest checkpoint (X_i, Y_i) is less than r_i, the box is considered in a safe area. The actual sensing data to be uploaded for each sensing is denoted by:

$$C = (L, K, x, y) \tag{2}$$

We would like to point out that we could use more types of sensors to monitor the box during transportation. For example, one can use a smart lock, a motion sensor, a humidity sensor, to enhance the monitoring of the vaccine box. These sensors are not hard to add-on, as one can see from next section that our design philosophy can be easily applied to them. This study represents our exploratory effort towards trustworthy vaccine shipment and for demonstration purpose we focus on a simple example and design the most essential components in this paper, leaving other opportunities to future interested real world system developers.

IV. Main Scheme: Trust Extension from On-Chain to Off-Chain Physical World

In this section, we detail the design and implementation of our trust extension scheme using the vaccine transportation as an example. Our scheme consists of a TEE-enabled Trusted Environment Monitoring System and a consistency protocol for uploading data from the TEE system to blockchain in an orderly fashion. As discussed earlier, vaccines may suffer from counterfeit, physical damage, being unsealed or replaced by fake ones during transportation. Nevertheless, it is extremely challenging to seamlessly monitor the transportation of vaccines in a trustworthy way in practice. This "physical untraceability" is common in many security-sensitive real world applications. We propose a system that permits secure and trustworthy vaccine shipping, which can capture any tampering and violation to the physical object with non-repudiation and traceability. We also present a fault tolerance mechanism that can recover from lost packets due to transmission failures, ensuring the seamless monitoring of vaccine transportation.

A. TEE-Enabled Trusted Environment Monitoring

We first introduce our full system of TEE-enabled Trusted Environment Monitoring System. We develop our system from the bare metal level for best security guarantee and efficiency/cost performance.

As discussed earlier, the secure zone inside a TEE hardware has the highest security privilege through physical isolation. Programs implemented in the secure zone can only be called by non-secure zone through a callable-API, but cannot be modified or inspected by the non-secure zone. Therefore, the secure zone should execute security-critical jobs such as trusted data collection, pattern extraction, encryption and decryption. The non-secure zone can do non-security jobs such as user interface or packet routing. To make the system function as expected, we first develop drivers of security-critical sensors within the secure zone, directly connecting them to the corresponding devices by wire; then we assign higher system interrupt priorities to security-critical tasks in order to achieve high real-time performance. Fig. 1 demonstrates the system block diagram with basic components introduced as follows.

Algorithm 1 Main System Utilities

Symmetric Key Derivation and Distribution
Function KDF($mas\_sk, recp\_pk, recp\_addr$)
//master secret key of the system, recipient (blockchain client) public key, recipient address.
//Symmetric encryption for efficiency, asymmetric signature for public blockchain verification.
    sym_sk = $Enc_{\text{AES}}(mas\_sk||recp\_pk||\text{TRNG}())$
    $distribution$
        $=Sign_{\text{RSA}}(Enc_{\text{RSA}}(sym\_sk, recp\_pk), TEE\_sk)$
    4Gsend($distribution$, $recp\_addr$)
return sym_sk

Retrieving Sensor Data
Function Sensors_get(θ)
Initialization: $L = 0, K = 0, x = 0, y = 0$
    if ($app\_ambient\_lum() > \theta$) **then** L=1
     //Hardware level logic
    end if
    K = $app\_temp\_get()$
    (x, y) = $app\_gps\_get()$
return $C = (L, K, x, y)$

Violation Detection
Function Violation_check($C, \Gamma_P, \Gamma_T, \Gamma_G$)
Initialization: $flag_P = flag_T = flag_G = 0$, $msg = null$
    if ($C.L == 1$) **then**
     $flag_P$=1 $msg.append$(“Box opened”)
    end if
    if ($C.K > \Gamma_{T \cdot Kmax}$ || $C.K < \Gamma_{T \cdot Kmin}$) **then**
     $flag_T = 1$, $msg.append$(“Abnormal Temperature”)
    end if
    if ($\min dist\{(C.x, C.y), (\Gamma_G.X_i, \Gamma_G.Y_i)\} > \Gamma_G.r_i$) **then**
     $flag_G = 1$, $msg.append$(“Route Deviated”)
    end if
    if ($flag_P||flag_T||flag_G$) **then**
        t = $get\_sys\_clock()$
        4Gsend($(t, msg), recp\_addr$)
    end if
return 0

Send Data Packets via 4G and Remote Agent Client to Blockchain
Function 4GSend($msg, recp\_addr$)
    $app\_4G\_send(msg, recp\_addr)$
    if (not receiving ACK) **then return** ‘fail’
        else return ‘ok’
    end if

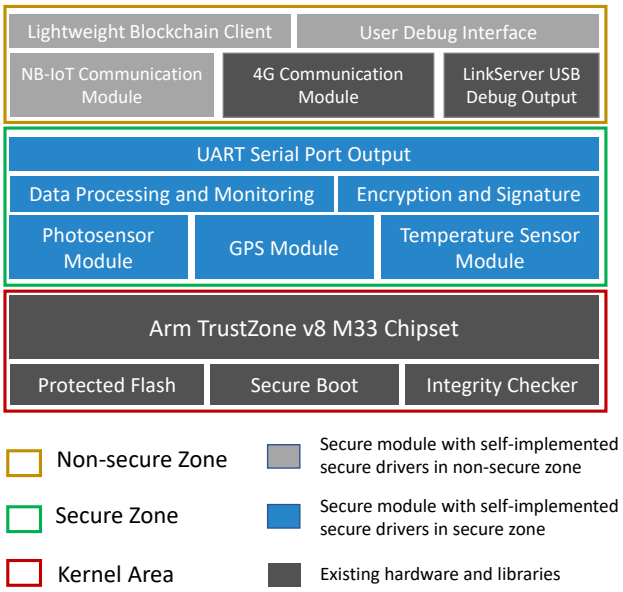


Fig. 1: Block Diagram of the System

At the lowest kernel level, a secure boot module and an integrity checker are first activated after booting. These two modules validate the current image (burnt-in executable) by computing its RSA signature and comparing it with the pre-stored correct one. If no corruption is found, the CPU control is transferred to the secure zone user space. The Protected Flash Region (PFR) stores the master secret key mas_sk and the key pair TEE_pk and TEE_sk, and must be checked for integrity by secure boot. If passing the check, which means that mas_sk and the key pair in PFR are intact, they are retrieved from PFR and loaded into the secure zone RAM. One can see that these keys cannot be faked, corrupted, or stolen from the PFR and the secure zone. The master secret key mas_sk is used to derive session keys for AES encryption while the public key pair is used for message authentication and signature verification.

At the secure zone we develop and assemble the drivers of the photosensor, GPS, camera, and temperature sensor in C language on our own. We also define critical logics and parameters such as the raw data collecting procedure, buffer size, inner capture frequency, debug procedures, and so on. This is reflected by **Function** Sensors_get(θ) in Algorithm 1, where θ is a threshold to ensure that the photosensor returns binary 1 if and only if its reading is above θ.

The data processing and monitoring module collects data from the sensors as C then compares the data with the predefined legit pattern Γ. If this checking finds any violation against Γ defined in Section III, an immediate alarm message along with the current timestamp is generated. Note that this alarm message is directly sent out and does not affect normal data uploading. This is shown as **Function** Violation_check($C, \Gamma_P, \Gamma_T, \Gamma_G$) in Algorithm 1. Next we retrieve the current system clock t, and packs all into a correct data structures (t, C, π), where π is the publicly verifiable

signature computed using TEE_sk to authenticate the trusted source of data C. Finally we check if there are lost history data (if so, includes them as f), encrypt $f||(t, C, \pi)$ with the session key sym_sk, and call `4GSend()` presented in Algorithm 1 to upload the data to the blockchain. This procedure is summarized by Algorithm 2 and the flow chart of the data uploading procedure is illustrated in Fig. 2.

The encryption and signature module performs encryption, signature signing, and verification. Recall that the system has a master secret key mas_sk and a public/private key pair TEE_pk and TEE_sk. To save computational resource, we need a symmetrical session key for encryption with AES. This symmetrical key sym_sk can be derived as an AES encryption output of concatenation of the master secret key mas_sk, the recipient public key $recp\_pk$, and a True Random Number which is generated by the embedded True Random Number Generator (TRNG). AES encryption is a common source of randomness to derive symmetric key. Since it is CCA-secure, attackers cannot have more than negligible probability to infer mas_sk providing sym_sk. We then encrypt symmetric key sym_sk once using the remote blockchain agent's public key to securely deliver sym_sk. This function is shown in **Function** `KDF`($mas\_sk, recp\_pk, recp\_addr$).

Finally in the secure zone, a Universal Asynchronous Receiver/Transmitter (UART) module receives a message and sends it out to the non-secure zone. The 4G or NB-IoT communication module, or LinkServer local USB debug probe receives the data output from the UART module. For time-sensitive applications it can use 4G module to transmit; for energy-efficiency-sensitive applications it can use NB-IoT module to transmit. The lightweight blockchain client in TEE provides the current remote blockchain agent server's address to which data can be redirected after entering the Internet. A user debug interface connects UART to USB and permits output via PC at the specific IDE. Fig. 3 shows another view of the system.

We notice that there exist other works that use TEE to perform trustworthy operations. However, they are either prohibitively expensive for using the more-easy-to-implement but expensive chipsets such as the Intel SGX, or macro-controller Cortex A-series that does not strictly enforce secure zone physical isolation. In this paper, to the best of our knowledge, we are the first to implement the system using Cortex-M series TrustZone Chipset as it is the first series that physically divides secure zone from the non-secure zone. It also poses great engineering challenge and overhead for having very few usable libraries, kernels, and operating systems. We build the system from the bare metal level using C and Assembly. To the best of our knowledge, we are the first to implement such a trustworthy system using the challenging yet cheap Cortex M33 MCU, which is priced around $40, while most other Intel SGX works cost around $300 $\sim$ $400.

B. Consistency Analysis

As discussed earlier, to achieve trusted physical traceability, we must ensure

Algorithm 2 Data Uploading Protocol

Input: $recp\_addr, max\_f$
//Sensing cycle length, remote blockchain agent address, max number of packet resend tolerance.
Initialization: Initialize system hardware. Activate blockchain. Synchronize system clock and blockchain client with external GPS time.
`KDF`($mas\_sk, recp\_pk, recp\_addr$)
while It is time for periodic reporting **do**
    C = `Sensors_get`(θ)
    `Violation_check`($C, \Gamma_P, \Gamma_T, \Gamma_G$)
    t = $get\_sys\_clock()$
    π = $sign_{\text{RSA}}(C, \text{TEE}_{sk})$
    if queue!='null'&& len(queue)$\leq max\_f$ **then**
        f=queue.pop()
    else
        if queue!='null'&& len(queue)$> max\_f$ **then**
            return 'Exceed maximum recovery tolerance'
        end if
    end if
    $msg = Enc_{\text{AES}}(f||(t, C, \pi), sym\_sk)$
    status=4Gsend(msg, $recp\_addr$)
    if status=='fail' **then** queue.push($f||(t, C, \pi)$)
    end if
end while
return 0

1) any event that happens must be uploaded and available within δ time;
2) a constant and periodical monitoring every Δt time,

which are respectively the *real-time data consistency* and *digital entity validity*. Algorithm 2 shows the pseudocode of the data uploading protocol. We now analyze the worst latency performance of this protocol.

Time latency is a sum of three random system delays, i.e., $\delta = \epsilon_1 + \epsilon_2 + \epsilon_3$, which are described as follows.

1) ϵ_1 is the delay of local data sensing and processing in the system. It begins at the moment the system starts sensing, and ends by the time the secure zone outputs a signed ciphertext data $D = (t, C, \pi)||f$ as a digital record.
2) ϵ_2 is the delay of transmission between the TEE system to the remote blockchain agent, who then decrypts the data and relays to the whole blockchain. It starts by the non-secure world receiving D and ends by the remote blockchain agent obtaining D.
3) ϵ_3 is the delay of synchronizing the records in the whole blockchain system. It starts by the first blockchain node receiving D and ends when all up-to-date blockchain nodes retrieving D from the blockchain.

In our system, we do not require strict synchronization between blockchain and the sensing system, which is a strong assumption and may disqualify many real-world applications. Instead, we allow $\mathcal{K}$ blocks to be evenly produced between two GST times. Each of this small block proposing time has

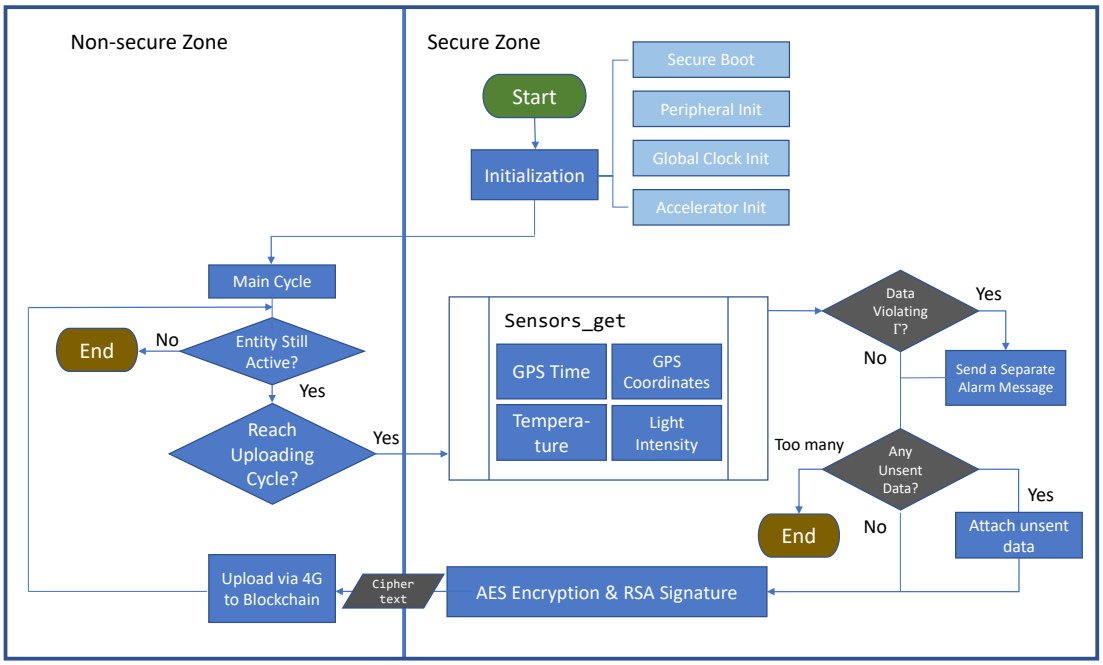


Fig. 2: Flowchart of the System

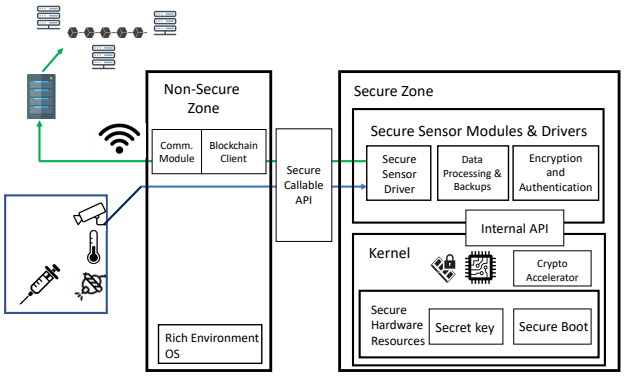


Fig. 3: Abstract Architecture of System

a length B and each interval between B is denoted by ΔB. We set $\mathcal{K} = 3$ in this study for illustration purpose. We require that all constantly active and non-faulty blockchain nodes follow every block production at every B time and synchronize at every GST. For most lightly-weighted or oblivious nodes that may disconnect or fail from time to time, they can catch up blocks afterwards as long as they are active before next GST.

Fig. 4 shows a normal operation of the system. One can see that time t_i is the local system timestamp at TEE when the event record *should* be created, and T_i is the time when blockchain has this event publicly available. The time gap Δt is the desired period between two planned sensings and ΔT is the actual time period between two records. We know that the system latency ϵ_1 is stable for a properly developed system, and the transmission latency ϵ_2 should also be stable for a high performance transmission protocol like 4G. The only unstable part that may affect the latency is ϵ_3, as a record may miss a block production time B. We use ϵ_s to denote the clock difference between the TEE system and the remote blockchain agent, as the agent actually proposes the record in blockchain.

Theorem 1. *Assume that there is no record loss. For a sensing started at time t, it takes at most $\delta + \Delta B + \epsilon_s$ time to have this sensed event available in blockchain.*

Proof. In the worst case, the sensing starts at time t, and after δ time the record reaches the blockchain. But unfortunately this record just misses one block production time B so it must wait one more ΔB time to be included in blockchain. Considering the time synchronizing difference one can get the result. □

Note that for lightly-weighted and oblivious nodes that only synchronize at GST time, the result in Theorem 1 is further relaxed to $\delta + \Delta GST + \epsilon_s$.

Theorem 2. *Assume that there is no record loss. For two consecutive timed events recorded in blockchain, the maximum time difference is $\Delta t + \delta + \Delta B + \epsilon_s$.*

Proof. We consider the worst case again. After the first timed event took place, we wait Δt which is the planned period between two sensing actions for the next record. Then according to Theorem 1, it takes at most $\delta + \Delta B + \epsilon_s$ time to have the next timed event to become available in blockchain. □

We can also relax ΔB to ΔGST for weakly synchronized or oblivious nodes. Note that we do not pose any further requirements on blockchain or our system. For a blockchain with a higher TPS or a shorter finalization time, the latency upper bound can be further decreased as the blockchain latency ΔB and ΔGST may decrease.

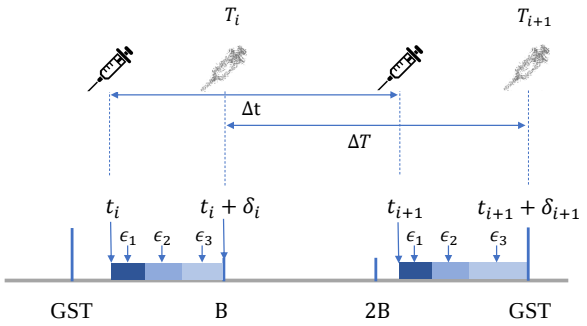


Fig. 4: Data uploading illustration

C. Lost Record Tolerance and Recovery

The whole system can malfunction in many ways; thus we need to consider fault tolerance as well as recovery. The malfunction or halt of the TEE system itself must require manual inspection and restart, and that of the blockchain is out of the scope of this paper. Therefore we consider fault tolerance and recovery for the transmissions between the TEE system and the blockchain system, which is common in real world and doable in a software level. On the other hand, if communication fails and the system can prove to blockchain that it did create the data record back in time as expected, those lost or late packets can be accepted without distrusting the device.

Recall that for each digital record $D_i = (t_i, C_i, \pi_i)||f_i$, the lost packets are stored in a queue and can be retrieved by

$$f_i = \text{queue.pop()} \tag{3}$$

Note that whenever a record D_j is not acknowledged by blockchain, we have

$$\text{queue.push}(f_{j-1} \;||\; (t_j, C_j, \pi_j)). \tag{4}$$

In other words, if the system does not receive an ACK for a record from blockchain, it pushes the data into the backup queue and try to resend it at the next uploading cycle.

By this design, we claim that the system is invulnerable to the following two common attacks, where 1) an adversary commits a jamming attack and therefore the 4G module cannot send to or receive from the outer world any information; and 2) the 4G signal is generally weak or the link is disconnected. Both cases are common in practice, and appear to have the device disconnected causing the records to be missed for a period of time. Note that even when the transmission signal is down, the secure zone is still constantly monitoring the environment inside the box and this backup queue cannot be modified as it stays inside the secure zone.

We allow a maximum of max_f number of consecutive missed packets to be backed up and resent. If the packets are correctly recovered, at packets then still follow the correct Δt patterns; thus we consider it acceptable and not conflicting with the validity defined earlier.

Note that in case 1), the adversary may compromise the inner environment of the TEE system while jamming the communications. Nevertheless, the records still stay in the secure zone (inside the backup queue), though jamming prevents them from being sent out. Unless the adversary jams the system forever or physically destroys the system, this attack action will sooner or later be publicly revealed.

One last case exists where the system may be attacked without being noticed, that is, an adversary jams the system for less than max_f number of record time and meanwhile 1) successfully breaks into the system and retrieves the secret key then creates a valid signature, or 2) forges a signature without the secret key and impersonates this device to upload fake data. These two cases can be nullified by the following security assumptions: 1) a trusted hardware (TEE) protects its programs in the secure zone from inspection or modification and 2) the unforgeability proof of a secure digital signature states that no polynomial time adversary can forge a valid signature with a negligible probability.

V. Experiments

In this section, we put our system into an actual test. We implemented the system as described earlier, attached it into a vaccine shipping box, and performed consistency monitoring towards the vaccine transportation.

A. Setup

The system was implemented on an LPC55S69-EVK development board from the NXP Semiconductor. The board consists of a LPC55S69 dual-core Arm Cortex-M33 microcontroller, running at 150MHz, and supports Arm TrustZone technology. For more information regarding LPC55S69-EVK please refer to the user's manual [8][9].

We developed our own blockchain system using Golang [10] for the best flexibility support. Golang is quite popular in security community for its memory-safe, high-concurrent, and high-usable properties. We adopted the Tendermint-BFT consensus algorithm, which is the Delegate Proof-of-Stake version of the original Practical Byzantine Fault Tolerance mechanism [11]. Specifically, Tendermint-BFT assigns different weights to different nodes during BFT voting, while PBFT assigns equal weights to all nodes. This weight, in practice, can be used to represent different trustworthiness of different nodes. We implemented the main blockchain system with about 4000 line of codes in Golang. The PC we used is a 8-Core Intel i7-6700HQ @ 2.6GHz with 16G memory and Ubuntu 18.04.1 GNU/Linux. We simulated 6 blockchain nodes on top of this PC as our blockchain system.

B. Evaluation

In our evaluation, we burned the project (containing codes for both the secure zone and the non-secure zone) into the LPC55S69-EVK board system. By providing a 5V external power bank, we made the system a standalone one. The board was positioned to the right side of the inside vaccine shipping box, leaving all critical sensors on the left side. As one can see from Fig. 5, the temperature sensor was the silver textured

probe, and the main part of the photosensor was a light-sensitive LED. The GPS antenna was small in size and its signal power was not affected when sealed inside the box. The 4G antenna was attached to the exterior top-left of the box. We set Δt to be 10 seconds, which means that we sensed the environment and uploaded to the blockchain one record every 10s. The maximum number of blocks for error recovery was fixed to 5. In our experiment, we attached the vaccine shipping box at the back seat of a motorcycle. The safe vaccine transportation temperature was ranged from 13 to 15 degrees.

We rode the motorcycle along a street and continuously collected the sensor data. To decrease the inside temperature of the vaccine box we placed two ice bags. The following results represent the data collected from 19:15:19 to 19:26:47 after the temperature dropped to the safe range, during which the 4G antenna was unplugged from 19:18:48 to 19:19:08 and from 19:25:48 to 19:26:18 to mimic two short jamming attacks. The vaccine box was opened at 19:25:27 and remained open until 19:26:47. As the sensing interval is 10 seconds, we collected in total 92 data points. Due to jamming we missed 3 records from the first attack and 4 records from the second one.

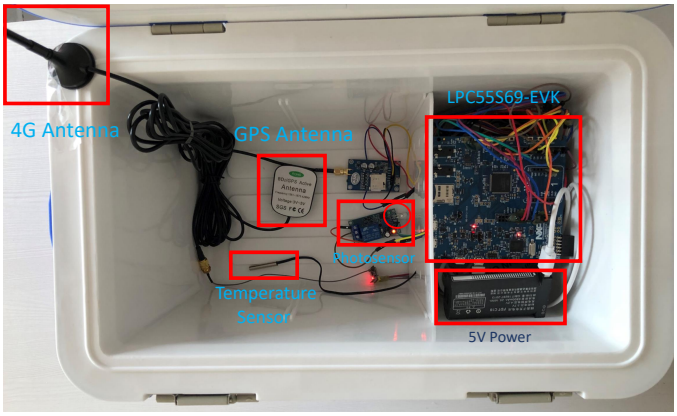


Fig. 5: Photo of the vaccine shipping box with sensing devices

From the 92 data points one can experimentally investigate the critical latencies of the system, i.e., ϵ_1, ϵ_2, and ϵ_3. We observed that ϵ_1 was between 5 to 7 seconds, with a mean of 6.6244 seconds and a variance of 0.2284. This is due to the low computation power of the embedded IoT devices. When 4G signals were stable, the 4G transmission latency ϵ_2 was between 50 and 80 milliseconds, with a mean of 76.3 ms and a variance of 2.2×10^{-5}. Each data record was about tens of KB so it did not inflict huge transmission overhead. The blockchain latency ϵ_3 was less than 60 milliseconds with a mean of 53.85 ms and a variance of 2.36×10^{-7}, as we had only 6 blockchain nodes. The maximum latency among all the data was 7.78 seconds. So one can safely assert that the total latency upper limit was 8 seconds for our test system.

Fig. 6a visualizes the actual shipping trace. We setup five predefined security checkpoints (gold stars) and fixed a radius of 500 meters (purple big circles). The security checkpoints and radius together define an approved ground transportation trace, which is a road shown as a faint white trace from left to right[2]. As one can see from the figure, the location was recorded, uploaded and authenticated regularly (red dots), demonstrating a trusted actual transportation trace. Although there were 3+4 planned jamming losses (green crosses), the records were all resent and recovered by the next following one. The 3 dropped records in the middle were clear to see as green crosses, with one of them overlapping with another data. The 4 in the end were all overlapping with others as we were circling in a small area. As one can see from the figure, the driver clearly did not deviate from the predefined route.

During transportation, we first kept the temperature low and made it stay steady at 14 degrees Celsius. Figure 6b shows the recordings of our actions, where we kept the box closed and sealed during time 0-600 (blue stars), and then opened it (at a summertime) after time 600 (red stars). Note that the x-axis uses the relative time from 0 to 700 instead of the actual data collection time for better illustration. After the box was opened, its internal temperature was quickly increased for 10 consecutive records and stayed stable at around 24 degrees. When these records were synchronized at all blockchain nodes, it was easy to observe and prove that this vaccine was exposed to undesirable temperature and should be no longer acceptable for customer use. Our planned jamming losses at time 200 (yellow stars) and 600 (orange stars) were also recovered nicely, as shown in this figure.

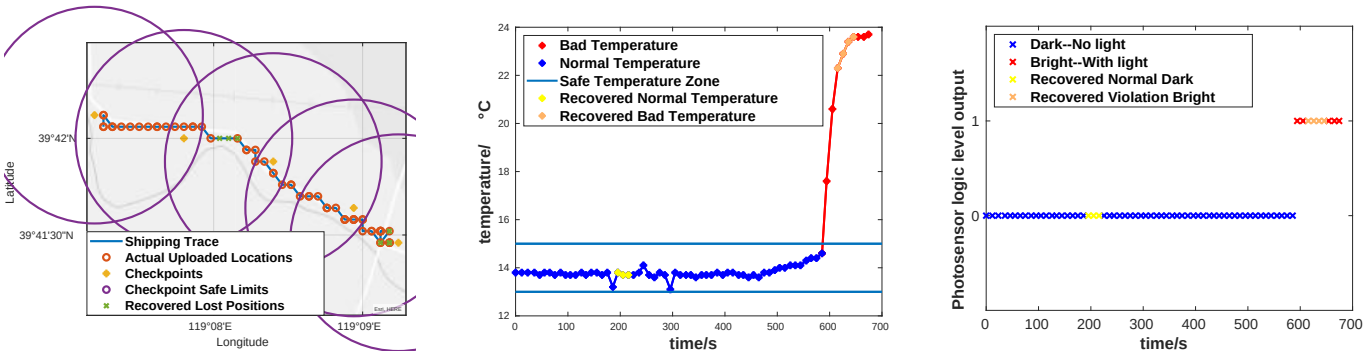
The outputs of the photosensor were illustrated in Fig. 6c. One can see clearly that the sealed vaccine box was opened at time 600. When the box was sealed, it remained dark (with a proper threshold that can negate faint LEDs inside the box) and hence the photosensor constantly output 0 (blue crosses); while when the box was opened at time 600 and was exposed to the sun, the photosensor snapped the ray and output the 3.3V high logic voltage level to the board (red crosses indicating the logical output of 1 in Fig. 6c). The photo and temperature sensors together can enhance the confidence that the box and its inside were indeed compromised. Fig. 6c also demonstrates that our 3+4 planned jamming losses at around time 200 and 600 were recovered correctly.

VI. Related Work

There exist a few attempts to extend the blockchain trustworthiness from on-chain to the physical world, by either applying blockchain directly to the physical scenarios making use of blockchain security, or using secure hardware or cryptographic primitives to enhance blockchain's internal security and trustworthiness.

Blockchain technologies have been applied in the fields of smart city, autonomous driving, smart home, etc. Zhou *et al.* proposed a hierarchical IoT architecture for smart home environments, where each home maintains a private chain that can hierarchically construct into a public one. However there were no countermeasures or discussions regarding how to stop the home owner from faking the record and lying to the public chain and how to validate the subchain data

[2]Not to confuse with the grey one, which is a river.



(a) GPS movement trace and security checkpoints

(b) Temperature fluctuation

(c) Photosensor and its logic voltage level output

Fig. 6: Experimental Sensing Data Separately Visualized as GPS Location, Temperature and Brightness.

authenticity at the public chain. Guo *et al.* developed an event recording system for autonomous vehicles with blockchain, and presented the concept of "proof of event", which is a hash digest of a combination of event locations, embedded vehicle event recorder readings and timestamps. This digest does not include any unforgeable data or data from an authenticated source, thus it can be easily faked afterwards [12]. It is also vulnerable to sybil attacks, where the fake data is shared among all sybil identities and uploaded by all, in order to increase the confidence of the fake data.

The above examples demonstrate a common problem shared by most blockchain applications: the blockchain trustworthiness is confined within the on-chain environment. There is no strong guarantee that the on-chain records continuously and truthfully reflect the *true* physical world, and that the blockchain control instructions are forcibly deployed on time.

Secure hardware or cryptographic primitives were also employed to enhance the blockchain security. Lind *et al.* proposed TEEChain, a TEE-based secure payment blockchain network in which TEE was used as *treasuries* to manage the off-chain funds and payments inside the TEE secure zone [13]. Ayoade *et al.* used the Intel SGX to securely offload data from on-chain to SGX-powered databases, which can hash the stored data and compare to the correct hash stored in the blockchain for data correctness verification. Dang *et al.* managed to scale blockchain via sharding and Intel SGX. They made use of the protected enclave module in Intel SGX to enhance security and increase the performance of the Proof-of-Elapsed-Time consensus and the PBFT consensus algorithms. Chainlink [14] and Provable [15] adopted various secure hardware such as Intel SGX to build a trusted inter-blockchain data exchange solution or website-to-blockchain trusted data input oracle. Some of these works made impressive progress by achieving trust extension between blockchains or between a blockchain and other digital environments, but they cannot extend trust from on-chain to the physical world. Not to mention that Intel SGX is a server-level hardware that typically costs around $300 to $400, which highly restricts its adoptions by IoT applications.

In this paper we propose a scheme to extend trust from on-chain to the off-chain physical world by developing a TEE-enabled trusted environment monitoring system and a fault tolerance uploading protocol to achieve real-time high data consistency between on-chain digital world and off-chain physical world. As far as we know, we are the first to overcome all engineering challenges and develop such a trustworthy system over the Cortex M33 MCU, which is highly competitive in price – around $40, approximately one-tenth of that of Intel SGX.

VII. Conclusions

In this paper, we propose a scheme to extend blockchain trust from on-chain to off-chain, and implement the full system taking trustworthy vaccine shipping as an example. Our scheme consists of a TEE-enabled Trusted Environment Monitoring System that continuously senses and generates anti-forgery data, and a consistency protocol that uploads data from the system and blockchain in an orderly, consistent, and fault-tolerating way. Our experiment records the internal status of the vaccine shipping box during the whole shipping process. One can see that our system is quite efficient with approximately 7-second of system processing time, 70ms of transmission time, and 40ms of blockchain synchronizing time. Our planned jamming attacks and physical violations are also captured and errors are recovered as expected.

As an exploratory work, we select vaccine transportation as an example and choose the photosensor, GPS sensor and temperature sensor to describe the status of a moving vaccine box. As we have our full system available, which is a relatively general framework that can customize critical parameters and tools, researchers can extend to any application of their interest by developing their own additional sensing devices based on our system template. We keep this work open-sourced at https://github.com/zhuaiballl/TEE-enabled_Trusted_Environment_Monitoring_System.

References

[1] Nature, "Chinese maker of faulty rabies vaccines fined bhttps://golang.org/doc/illions of yuan," Avail-

able at https://www.nytimes.com/2018/07/23/world/asia/china-vaccines-scandal-investigation.html, July 23, 2018.
[2] T. N. Y. Times, "In china, vaccine scandal infuriates parents and tests government," Available at https://www.nature.com/articles/d41586-018-07136-z, October 23, 2018.
[3] C. M. Nelson, H. Wibisono, H. Purwanto, I. Mansyur, V. Moniaga, and A. Widjaya, "Hepatitis b vaccine freezing in the indonesian cold chain: evidence and solutions," *Bulletin of the World Health Organization*, vol. 82, pp. 99–105, 2004.
[4] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, 1988.
[5] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.
[6] block.one, "Eos whitepaper, v2," Available at https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md/, Mar 16 2018.
[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.
[8] N. Semiconductor, "Nxp semiconductors lpc55s6x arm cortex-m33 microcontrollers," Available at https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc5500-cortex-m33/high-efficiency-arm-cortex-m33-based-microcontroller-family:LPC55S6x, 2020.
[9] ——, "Lpc55s69evk: Lpcxpresso55s69 development board," Available at https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc5500-cortex-m33/lpcxpresso55s69-development-board:LPC55S69-EVK, 2020.
[10] Golang, "Golang documentation," Available at https://golang.org/doc/, 2020.
[11] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on bft consensus," *arXiv preprint arXiv:1807.04938*, 2018.
[12] H. Guo, E. Meamari, and C.-C. Shen, "Blockchain-inspired event recording system for autonomous vehicles," in *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*. IEEE, 2018, pp. 218–222.
[13] J. Lind, O. Naor, I. Eyal, F. Kelbert, E. G. Sirer, and P. Pietzuch, "Teechain: a secure payment network with asynchronous blockchain access," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 63–79.
[14] ChainLink, "The chainlink network provides reliable tamper-proof inputs and outputs for complex smart contracts on any blockchain." Available at https://chain.link, 2020.
[15] Provable, "The provable blockchain oracle for modern dapps," Available at https://provable.xyz, 2020.