Black Block Recorder: Immutable Black Box Logging for Robots via Blockchain

Gianluca Caiazza and Agostino Cortesi
Ca' Foscari University of Venice
Venezia VE, Italy

Ruffin White and Henrik I. Christensen
Contextual Robotics Institute
UC San Diego, California, USA

***Abstract*—Logging plays a crucial role in robotic research, providing prolonged insight into a robots encountered environmental stimuli, internal behavioral state, and performance or outcome of actions taken; all necessary for profiling and debugging robotic application ex post facto. As robotic development matures into production, logging assumes an additional role in equipping auditors with the evidence necessary for investigating issues, accidents or fraud. Given robotic sectors such as drone delivery or autonomous transport must operate in the open world, ensuring the integrity, authenticity and non-repudiation of generated logs on these mobile cyberphical systems presents new threats that extend beyond those in traditional IT computing: such as physical system access or postmortem collusion between robot and OEM resulting in the truncation or alteration of previous records. In this work, we address the topic of immutabilized logs using integrity proofs and distributed ledgers with the additional consideration for mobile and public service robotic applications.**

***Index Terms*—Cryptobotics, Networked Robots, Industrial Robots, Robot Safety, Distributed Ledgers, ROS2**

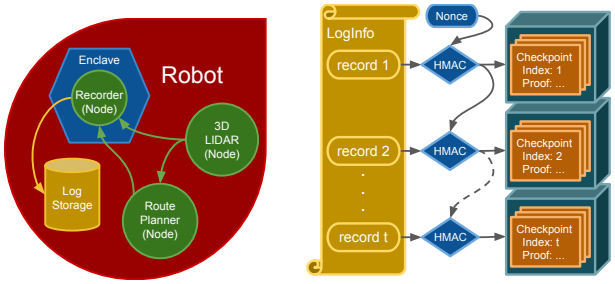

Fig. 1: High level overview of immutabile logging. Right, depicts an example deployment where an enclaved process generates the logs by capturing message traffic directly from each source. While streaming the log data out to arbitrary storage, the data is made immutable by submitting integrity proofs to the blockchain, comprising of linked HMAC digests that are indexed as checkpoint transactions, shown left.

I. Introduction

The spread of autonomous robotic applications and the ubiquity of Internet of Things (IoT) devices has narrowed our interaction boundary with computerized interfaces and has smooth the interplay with robotic system by defining the so called seamless interaction. The possibility of executing operations autonomously, without observable delays in response time and in the absence of human interaction, is inviting; however, the rush in the uptake of deploying robotic solutions in real world scenarios is not trivial.

As discussed by Morante *et al.* [**?**] the current cyber-security state of robots is not keeping up with the computers and smartphones counterpart. In fact, the developers of robotic applications has always overlooked cyber safety features in place of prioritizing the development of useful functions in 'secure' environment in which there were no possibility of external unsanitized inputs. Considering the change of scale in deployment from closed environment to open word, it's no surprise the grown in the amount of vulnerabilities that has has been recently reported.

Considering the spread of connected devices, ranging from one-purpose specific to more complex one, the amount of information that they process and the risk posed by their operation is not something that could be overlooked. In regards of debugging the information flow or an unexpected robot behaviour, the usage of logging is a pivotal feature. However, since we can't always guarantee the security of the robot, the correctness and completeness of the log is not certain. In the one hand, as presented by Mineraud *et al.* [**?**], today's IoT middleware platforms presents several challenges either from the security and privacy point of view. From their analysis has emerged how cloud-based IoT platforms are prone to traditional security attacks such as Denial of Service (DoS), man-in-the-middle (MIM), eavesdropping, spoofing, etc. Moreover, general IoT platforms has, among the other, several implementation flaws as unsuitable device authentication mechanisms, defective data storage protection, faulty trust managements, etc. On the other hand, more structured robotic framework as Robot Operating System (ROS) [**?**], as we will discuss later, presents to a malicious user various attack surfaces. In order to overcame some of this limitation and guarantee in a multi-robot distributed system a secure logging mechanism, we can use immutable logging. With the term immutable logging we generally refer to log files that posses some degree of protection from tampering and erroneous insertion. This tamper-resistant log record in an indelible manner all the operations that has occurred in the robot. The resulting content integrity logs allows us to perform static formal analysis verification [**?**] for security audit in multi-robot networks as depicted in Fig. 2. Thanks to the integrity guarantees of the immutable log, we can confidently store 'correct' critical data with a cryptography assurance of verification of the original.

Interesting application of such technique, is the deployment of historic archiving for medical information in bleeding edge emerging applications such as mobile Health (mHealth) [**?**]. In this case, data security for remote medical devices on the patient needs to be auditable for timely medical treatment. Moreover, in more complex life-critical machinery as surgery robot [**?**], been able to address accountability and ascertain possible attacks [**?**] also in remote connections is crucial.

Furthermore, as regards of the widespread interest for self-driving cars and autonomous drones the possibility of deploying real-world 'honeypot' is a serious concern [**?**]. Considering the recent history in automotive exploiting [**?**] and the already available attack surfaces, there can be no doubt that it represent a real threat [**?**], [**?**], [**?**].

Overview

- II **Related Work:** Discussion and background of immutable logs, distributed ledger technologies, and trusted execution models, as well as the limits of the existing approaches with respect to robotic requirements.
- III **Approach:** Formulates the integrity proof, smart contract and permissioned blockchain architecture implemented for the presented framework, including design mechanisms and development choices.
- IV **Implementation:** Details an implementation for evaluating the capability of our proposed framework with regards to integrity verification and runtime performance under mobile robotic like scenarios.
- V **Conclusion and Future Work:** Includes a discussion of the presented work and potential extensions with respect to newer available conscientious methods, improving the practicality and scalability for real world use.

II. Related Work

The Bitcoin Blockchain [**?**] represents a valid alternative solution to the usage of trusted third parties to process and mediate transactions. The idea of substitute thrust with a cryptographic proof in a peer-to-peer (p2p) network to allow willing parties to transact without the need of a third party, have opened novel research areas in several fields including Computer Science. Prior to Blockchain, horizontally scalable distributed databases were the leading techonolgy to store, under a central authority, a potentially infinite amount of information; still, being scattered across different devices, it's easy to see how this solution lacks the immutability and scalability feature that the Blockchain protocol guarantee by design. Starting from the original paper, several alternative usage of Blockchain has been proposed in the wild. In regards of those, we discuss below how we pose our approach. In order to ease the reader throughout the related work, we split the background in three sections.

A. Distributed Ledger

Blockchain is a peer-to-peer distributed ledger which leverage its security by means of public-key cryptography. Each peer in the network have a public address, derived by the hash of their self generated key, that identify the user among all the other peers. When users wants to perform an operation, they build a transaction by including as *input* their blockchain addresses, the amount of the involved resources and specify the hashes of the *outputs* of the previous Blockchain transactions. By means of a digital signature of the transaction, they provide the proof of ownership of the specified public address. The resulting transactions are then broadcasted in the p2p network and collected by *miners* that aggregates them in *blocks*. Once a block is ready, the miners mine it in order to add it to the chain of transactions. The mined blocks are then broadcasted in the network and distributed among the peers. The security of the approach is given by the fact that any peer can be a miner and that all the transactions are verified by all the peers. A malicious miner who tries to double-spend the transaction or modify it, will be detected by all the other peers provided that the malicious miner doesn't have more than 50% of the whole mining power

As discussed by BitFury and Garzik white papers [**?**] [**?**], blockchain-based ledgers has gained much popularity among banks and other financial institutions with the definition of several algorithms that leverage on blockchain's immutability and consensus for the validity of the transactions. However, nowadays Blockchain-based solutions are reaching physical limitations in terms of slow transaction throughput and a lesser degree of scalability in term of Proof-of-Work cost [**?**].

B. Immutable Logs

C. Trusted Execution

III. Approach

In our approach, the immutablizing of robotic logs using distributed ledgers requires the development of two main coupled components: the integrity proof and the smart contract specification. In this section, we detail the design and justification of both with respect to the constraints incurred by mobile robotic application an notable open frameworks incorporated.

A. Integrity Proof

To preserve the integrity of the logs without compromising system performance or publicly disclosing private log content, a similar procedure to [**?**] is adopted by chaining the log checkpoints together using Keyed-Hash Message Authentication Codes (HMACs). Borrowing terminology established in [**?**] we define a log checkpoint (Chk_i) to be linked with the previous one by using the prior digest (h_{i-1}) as the key bytes when computing the current HMAC digest (h_i) from the log content ($LogInfo_i$) (see E.g. 1).

$$Chk_i = (i, h_i) \quad h_i = HMAC(h_{i-1}|LogInfo_i) \\ where \quad h_0 \leftarrow^{\$} \{0,1\}^m \tag{1}$$

For privacy, a random nonce is included as the genesis digest (h_0) to inject initial entropy into the linked integrity proofs, ensuring separate log files with similar beginning contents do no repeat the same telltale signature of consecutive proofs.

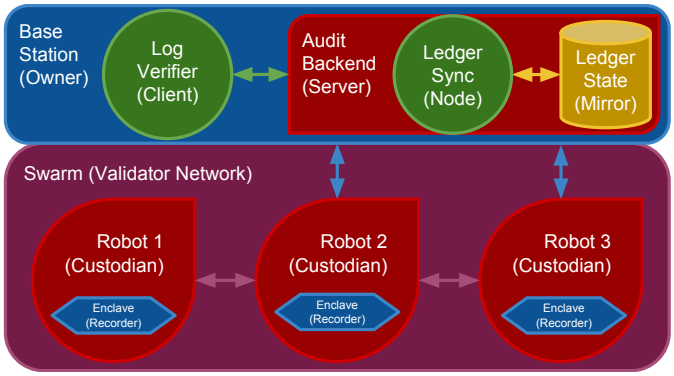

Fig. 2: Network perspective of swarm deployment. Log integrity of each robot is preserved via consensus of it's peers. An authority may assign a recorder to each custodian prior deployment, where the enclaved logger monitors it's charge on the auditor's behalf. As wireless connection with deployed swarm my be intermittent or constrained, a local mirror of the distributed ledger may be cached for offline auditing.

This deviates significantly from previous works that seeks to complicate the integrity proof to accommodate log authenticity and non-repudiation when relying on token based blockchains (designed primarily just for financial transactions) to achieve immutable yet distributed checkpoint archiving. We achieve these two properties differently through the use of Smart Contracts (SC) and Public Key Infrastructure (PKI) discussed in the next section. By keying with the previous unmodified (and potentially prior published) checkpoint digest, we reduce the validation of logs to the trivial task of checking a rudimentary meta-blockchain: I.E. sequentially iterating through the $LogInfo_i$ atomics in the log file and ensuring the order of digests correspond to the time series of proofs published into the global blockchain. Thus, any log checkpoint manipulation or deletion is stilled detected during the verification process.

By also including the index (i) into the checkpoint, we can ascertain a number of additional beneficial properties, such as partial validation or verification resumption in the face of missing or corrupted log elements. Additionally, provided log content includes an embedded indexing, as is common with sequential recordings, this affords parallelizable validation over large log files, accelerating the verification process. This perhaps comes at the cost of potentially leaking information such as the frequency of log elements being generated, however a degree of this can already be inferred by observing transaction activity in the public blockchain regardless.

Previous works such as [**?**], [**?**] make the distinction between two different types of checkpoint entries; the first being an incremental link in a chained proof, while the second being a anchor point that must always be published to commit to new secrets while revealing expired ones for later verification purposes. Our approach to checkpoints make no such distinction, thus any checkpoint or number of checkpoints generated may be published immediately or simultaneously. This ensure that checkpoints can always be submitted in short notice or without necessary waiting for previous transactions to be finalized in the global blockchain.

Given we do not require anonymity of recorder identity, we dispense the need of rotating private HMAC keys. Although key rotation does provide an element of forward security in the face of private key exposure, we attempt to mitigate this threat model in section **??** and by deliberately differing checkpoint write permissions to the SC where control policies of ownership and action access can be made far more expressive and dynamic.

For robotic applications in particular, where mobile processors may be subject to brownouts without warning due to self reliant energy supplies, integrity proofs that require stateful cryptography [**?**] could leave a recorder without recourse for resumption, as the previously published checkpoint would have included a commitment to future key (a secret lost if not written to persistent memory) that must be used and then revealed upon the next checkpoint. Given a recorder should wait for respective log IO completion before the publishing that checkpoint, the recorder can quickly re-derive the latest digest from the log data and head block to resume checkpointing wherever left off.

B. Smart Contract

In section III-A we detail our approach of indexed integrity proofs to ensure log file immutability, however this simplified method of verification does not alone offer the authenticity and non-repudiation properties required for securely logging distributed robotic applications. We instead utilize Smart Contracts (SC) that encapsulate the logic for blockchain validators to abide by when determining the validity of proposed checkpoint transactions.

Instead of relying on colored coins or token metadata in financial blockchains to encode checkpoints into the state of the distributed ledger, a dedicated transaction family can be defined to regulate ledger state, i.e. a SC enforcing governing or arbitrary computation. A common criteria however is that the validity of candidate transactions must be determiticly computable; i.e. no context external to the current state of the ledger and transaction payload in question should be used in deliberation. This ensure that the validity of any block in the chain can be independently verified regardless of time transpired since its creation.

To ensure authenticity of checkpoint committed into the blockchain, we consider transaction that must be signed using Digital Signatures (DAs) via PKI, effectively notarizing the identity of the signer. For our purposes, we also register the identity into the blockchain by enrolling it's public key into an access control policy stored in the distributed ledger to be used by SCs when inspecting candidate checkpoint transactions. Thus, limiting recorders permissions to append checkpoints on behalf of log files they are authorized for.

To ensure non-repudiation of transactions, our SC considers the index of the checkpoint, mandating it remain monotonically increasing. Skipping of indexes is permitted to enable recorders control in the rate at which they publish integrity

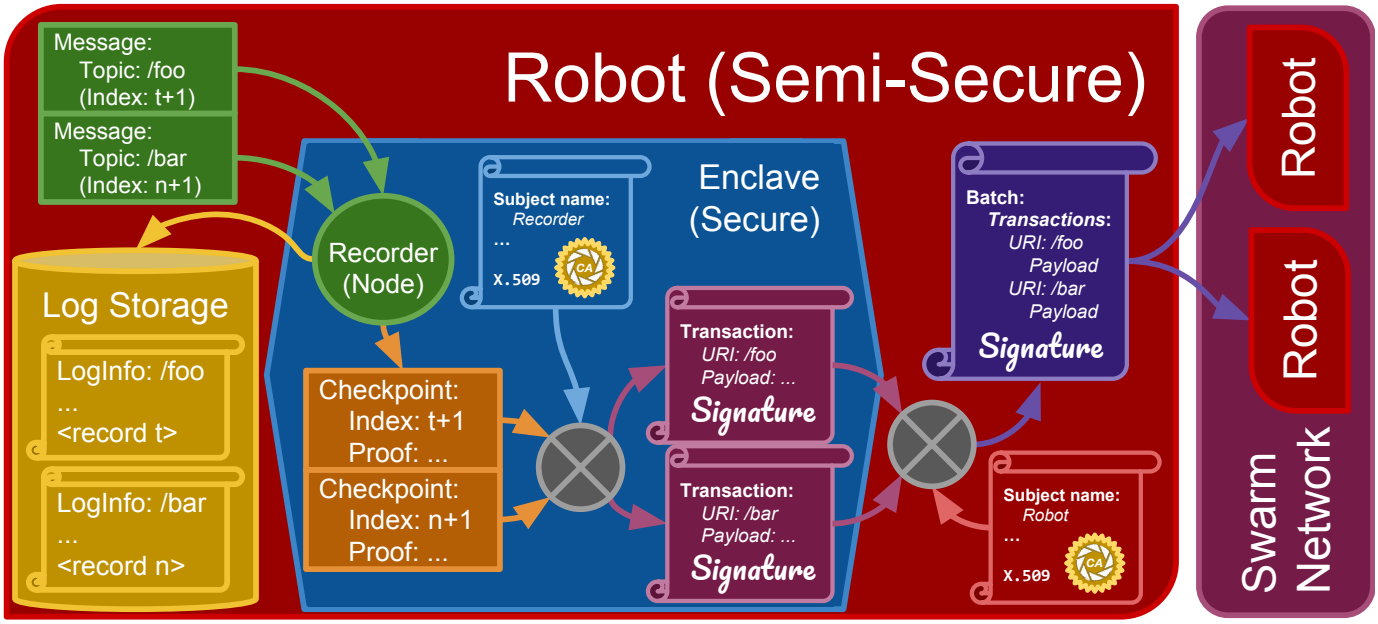

Fig. 3: Flow chart visualization of the immutable logging pipeline. While every robot platform is held suspect, a secure enclave (e.g. Trusted Execution Environment) is reserved for the recorder process. Logged input is securely received within the enclave and used to cryptographically derive a linked integrity proof specific for each input asset being tracked. As the log data may be streamed to external storage, respective checkpoint transactions are bound to the robot's public identity for batching and then signed by the recorder's private key sealed within the enclave. Thus only the robot's private key may be used to sign and relay batched transactions for validation: necessitating the collusion of both custodian matching recorder for append forgery.

proofs (while locally checkpointing log data the rate generated) to conserve energy or wireless network bandwidth and minimize the growth of the distributed ledger's state. To additionally curtail the memory growth of the ledger, effectively a database each validator must locally maintain to participate, a ring buffer is used here to keep track of the n latest checkpoints.

More advanced down-sampling and data retention methods could be applied; i.e. keeping a high precision of recent checkpoints while retaining an ever lower resolution of them as they age out, summarizing log integrity over increasing wider spans of time past. A SC could formulate a clock like gear reduction of nested ring buffers used as proxy for log rotations; e.g. a week-buffer overflows into the month-buffer only once every 7 days, and into a year-buffer once a month, etc.

Another means of non-repudiation such as timespamping could be used to subsume the role of indexing in providing an unique counter underneath transaction signatures. However, any reliance on time may be misplaced given that by design blockchains in general preserve only the order of events (i.e., weak freshness), while accurate timing of events (i.e., strong freshness) is questionable due to fundamental issues in resolving multiple time sources and relative timespaming. The work of [**?**] prescribes a workaround using a centralized third party, however this perhaps plays agents our objectives of distributed trust and scalability. Mobile swarm robots may roam autonomously beyond the network range of base stations or any one particular member, thus any agreed reference of time must arise from a distributed consensus in the swarm.

IV. Implementation

V. Conclusion

A. Future Work

Acknowledgment

We'd would like to thank the Hyperledger project for their open source contributions and documentation: both instrumental in developing this work and exploring its application.

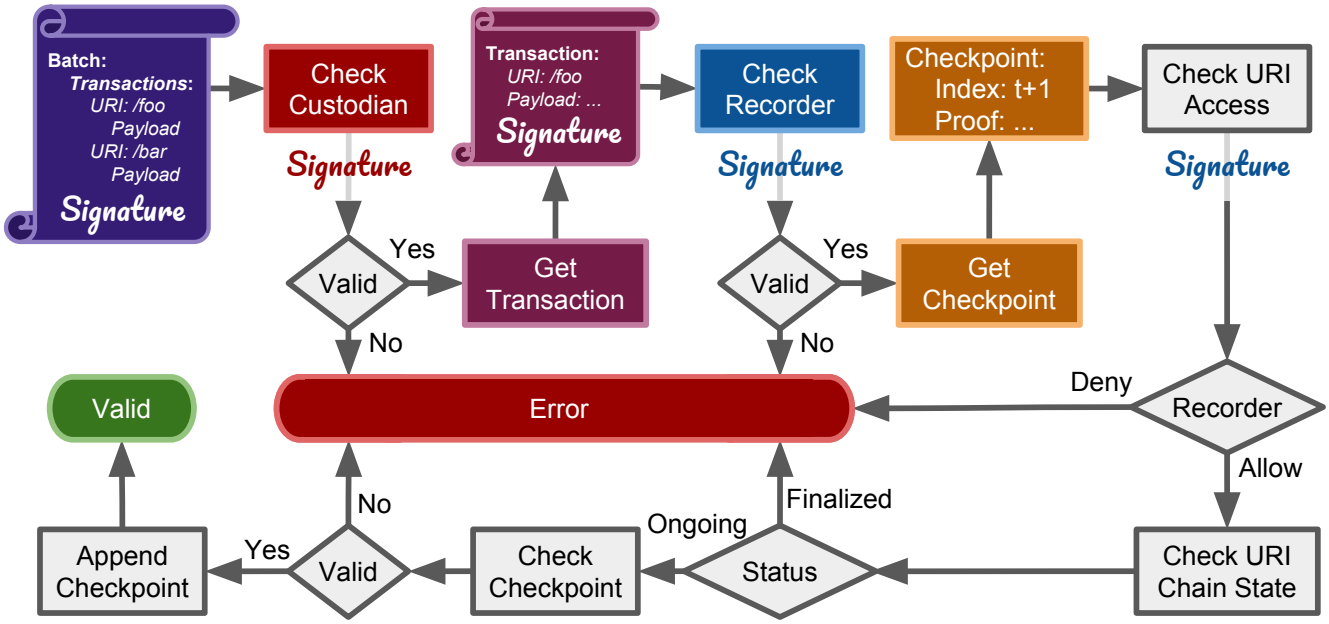

Fig. 4: Flow chart visualization of validating smart contracts for checkpoint transactions. Upon receiving a candidate block, the validator will check the corresponding batch's signature matches the custodian identity declared in each transaction. Each transaction signature is also check to validate the public identity of the recorder. The recorder identity is used when determining the authorization in appending new checkpoints for a specified asset to the ledger. If the status of the asset has already been finalized, any following append actions are rejected. Otherwise valid checkpoints may appended to logged asset's record.