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

Ruffin White[1], Gianluca Caiazza[2], Agostino Cortesi[2], Young Im Cho[3], and Henrik I. Christensen[1]

***Abstract*—Event data recording is crucial in robotics research, providing prolonged insights into a robot's situational understanding, progression of behavioral state, and resulting outcomes. Such recordings are invaluable when debugging complex robotic applications or profiling experiments ex post facto. As robotic developments mature into production, both the roles and requirements of event logging will broaden, to include serving as evidence for auditors and regulators investigating accidents or fraud. Given the growing number of high profile public incidents involving self-driving automotives resulting in fatality and regulatory policy making, it is paramount that the integrity, authenticity and non-repudiation of such event logs are maintained to ensure accountability. Being mobile cyber-physical systems, robots present new threats and vulnerabilities beyond traditional IT: unsupervised physical system access or postmortem collusion between robot and OEM could result in the truncation or alteration of prior records. In this work, we address immutablization of log records via integrity proofs and distributed ledgers with special considerations for mobile and public service robot deployments.**

***Index Terms*—Robot Safety, Networked Robots, Software Middleware, Cryptobotics, Distributed Ledgers**

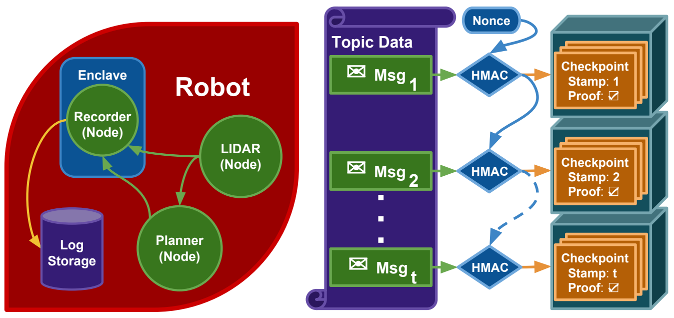


Fig. 1: High level overview of immutable logging. Left, depicts an example deployment where an enclaved process generates the logs by capturing message traffic directly from each source. While streaming the log data out to arbitrary storage, the data is made immutable by submitting striding checkpoints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

I. Introduction

ROBOTS, being cyber-physical systems (CPS), are increasingly deployed as part of a cyber-infrastructure, becoming ever more interconnected with the Internet of Things (IoT). The use of robots in a connected eco-system is far from trivial. How do we make these robots safe? How can we verify correct operation? How can we document operations for the purpose of traceability or document operations in the presence of failures? An essential part of the design, deployment and verification of distributed robot systems is the ability to monitor and record runtime event data.

The possibility of deploying real-world 'honeypots' is a serious concern given the widespread interest in exploiting self-driving cars and autonomous drones [1]. Considering the recent history in automotive exploitation [2] , there can be no doubt that lack of security represents a real threat [3]–[6].

Documenting the operation of an integrated robotic system composed of multiple components while generating a comprehensive trace of the operation is essential to quality control, debugging, systems verification, etc. For debugging information flow or in cases of unexpected robot behaviour, event logging is fundamentally integral. However, when the absolute security of a robotic CPS can not be guaranteed, the correctness of such event logs is subsequently tenuous.

Digital forensic investigations (DFIs) [7] use digital logs as evidence in post-event analysis or in intrusion detection systems (IDS) in electronic devices. By continuously broadcasting abridged cryptographic commitments of system state, devices are constantly under the sword of Damocles which incentivizes honesty via enforced accountability without relying on specific hardware. Considering the mass-manufacturing restrictions for mobile robots including build-of-materials, serviceability, immense data rates, and the utilization of low-volume high-cost tamper-proof storage devices such as Write Once Read Many (WORM) memory would be financially unprofitable.

Additionally, considering the limitations for mobile robotic platforms such as: restricted computational power, networking bandwidth, on-board energy capacity, extensive transmission of encrypted logs would be not only technologically impractical, but also in violation of international data retention policies. As detailed by Veitas et al. [8], though shared server-centric data retention is favored by OEMs for its straightforward architecture, it is also in direct contention with governmental regulatory agencies and privacy advocacy groups.

Thus, our goal is in verifying the integrity, authenticity, and completeness of robotic event data while under the threat of malicious/erroneous insertion, omission, or replacement. To this end, we explore the application of a Event Data Recorder (EDR) based upon cryptographic linked integrity proofs, disseminated via distributed ledgers, shown in Fig 1.

Manuscript received: February, 24, 2019; Revised May, 24, 2019; Accepted July, 8, 2019. This paper was recommended for publication by Editor Paolo Rocco upon evaluation of the Associate Editor and Reviewers' comments.

[1]Ruffin and Henrik are with UC San Diego, USA rwhitema@ucsd.edu
[2]Gianluca and Agostino are with Ca' Foscari University of Venice, Italy
[3]Young is with Gachon University Seoul, South Korea

In this work, we present Black Block Recorder (BBR), an approach combining the use of Digital Signature Algorithms (DSA), keyed-hash Message Authentication Codes (HMAC), and Smart Contract (SC) via Distributed Ledger Technology (DLT) to enable tamper-evident logging, while considering the limited resources available for mobile robotic deployments.

This is the structure of the paper: Section II (Related Work) discusses the literature on immutable and tamper-evident logs, distributed ledger technologies, and the limits of using existing approaches with robots. Section III (EDR Roles, Requirements and Primitives) discusses the details of the properties we want to enforce and what trust settings are addressed with the proposed framework. Section IV (Approach) formulates the integrity proof, smart contract and permissioned blockchain architecture as implemented in our framework, including design mechanisms and development choices. Section V (Implementation) discusses the details of an implementation to evaluate our proposed framework as capable of integrity verification and runtime optimizations for mobile robotic scenarios. Finally, Section VI (Conclusion and Future Work) provides a discussion of the work and extensions w.r.t. newer available consensus methods for the practicality and scalability in real world deployments.

II. Related Work

First, we give a brief introduction to token-based ledgers and their main properties; then, we discuss in greater detail the concept of distributed ledgers technology (DLT), immutable logs, trusted computing, and their relevance to Event Data Recording for autonomous systems.

A token-based blockchain is a peer-to-peer (p2p) distributed ledger which derives its security from public-key cryptography. Each participant in the network has a public address within the Merkle Tree [9], e.g. derived by the hash of its public key, which identifies the user uniquely among all the other participants. Transactions between users are defined by providing as *input* the users' blockchain addresses, the balance transfer and the hashes of the *outputs* of the last accepted block. Candidate transactions are signed and then broadcast in the p2p network and collected by *validators* that aggregate them in blocks. A candidate block is produced when validators "mine" it - by solving the challenge of the consensus algorithm - whereupon it will be proposed and added to the chain of previous transaction blocks. A proposed fork is only adopted by a validator after it is determined to be the longest chain among the network where all transactions remain valid. The security of the approach is assured by the Byzantine Fault Tolerance (BFT) of the consensus algorithm used, and by relying on the difficulty or inherent cost in subverting the consensus algorithm as a deterrent against malicious actors. Readers unfamiliar with this DLT architecture are referred to the seminal work [10] for an approachable introduction.

A. Distributed Ledgers Technology

Prior to DLTs, horizontally scalable Distributed Databases (DDB) were commonly used to replicate record state across trusted storage devices. However when relying upon CPS infrastructures for data retention, auditing the integrity of classical DDB updates in face of transiently available or compromised devices can deteriorate into an under-constrained problem. Reconstructing postmortem consensus of chronological changes across remaining DDB replications with potentially revoked credentials are classes of issues that can be avoided when disseminating data integrity using DLTs instead.

As an example, Bitcoin [10] provides an alternative to the use of trusted third parties to process and mediate transactions; i.e. the main focal point being the introduction of distributed trust even under mutually distrusting validators. The resulting distributed ledger contains an chronological evidentiary trail of consensus that every participant can easily audit.

As discussed by BitFury and Garzik white papers [11] [12], blockchain-based ledgers have gained popularity among banks and other financial institutions with the ongoing development of several applications that leverage upon Blockchain's immutability and consensus to validate transactions. However, public finance blockchains are constrained due to latent/limited transaction throughput and scalability due to energy and opportunity costs consumed by traditional Proof of Work (PoW) [13] consensus. To overcome these limitations and enforce enterprise-level security mechanisms, alternate variants have emerged by defining *public* and *private* distributed ledgers.

In public ledgers there are no restrictions on submitting transactions. Private ledgers limit those actions to a predefined list of entities. Ledgers are further classified as *permissioned* and *permissionless*. In permissioned ledgers vs. permissionless, the identity of peers that act as validators is restricted; e.g. whitelisted public keys. Public permissionless ledgers are used for cryptocurrencies like Bitcoin; public permissioned ledgers are used to keep control on 'certified' validators; private permissioned ledgers work in ways similar to enterprise distributed databases; private permissionless ledgers are not possible. Even more novel approaches to ledgers have emerged in the Hyperledger Project [14] from Linux Foundation, which seeks to improve the performance of the distributed ledgers by creating open source enterprise standard libraries.

B. Immutable Logs

Immutable logs require robust tamper-proof logging. Using cryptographic functions we are able to enforce integrity, authenticity, and non-repudiation of the logs' entry. Several proposals to achieve immutable logs already exist in the literature; the usual general idea is to use a combination of DSAs and Message Authentication Code (MAC) to unambiguously validate log entries. It is possible to enforce accountability [15] in an heterogeneous distributed environment and reduce the number of trusted devices. However, the needs of central authorities to store and verify the logs makes it necessary to build an additional chain of trust and deploy a distributed storage system for logs (e.g. distributed databases). The use of a distributed versioning implementation such as IPFS [16] can also be a valid option. Still, the use of Merkle DAG does not incorporate verification mechanisms such as smart contracts which are vital to apply validation logic to the system.

Following the discussion in II-A, considering the similarity with Blockchain and its intrinsic security features, leveraging on Bitcoin presents an appealing solution [17]. Snow *et al.* [18] present how Factom [1] distributes immutable logs on Bitcoin chain using an OP_RETURN transaction to store the entry of their client logs. Similarly, Cucurull *et al.* [19] discuss how at Scytl [2] it was possible to incrementally secure electronic voting machine results on Bitcoin blockchain. However, cryptocurrencies developers regard this as among the more dubious emerging trends in the wild and an abuse of the OP_RETURN to piggy-back arbitrary data for storage on the Bitcoin Blockchain [20]. As discussed by Matzutt *et al.* [21] the impact of this abuse to store non-financial content on original cryptocurrency blockchains is unsustainable.

On the other hand, Sutton *et al.* [22] follow the concept of checkpoints presented by Cucurull *et al.*, to propose a model using Linked Data to optimize the use of Blockchain by constructing a hashing tree rather than continuously dumping logging hashes into the chain. This becomes necessary since the misuse of OP_RETURN has several disadvantages either from the protocol point of view discussed above or because of the transaction fees incurred. All the transactions that need to be published in cryptocurrency blockchains need to pay a fee that will be '*burned*', deducting the limited balance from the account. Considering the volatile increase of Bitcoin's exchange rate over the years, it's clear that this costly operation is not viable for large scale deployments.

Another barrier to the use of blockchains for storing immutable logs is presented by the freshness property of the Blockchain [23]. By design, Blockchain preserves the order of events (i.e. weak freshness), however the accurate time of events (i.e. strong freshness) is not guaranteed. The work of Szalachowski [24] offers a workaround using a centralized third party, however this plays somewhat against our own objectives of distributed trust and scalability. Mobile robots may roam autonomously beyond the network range of centralized base stations or any one particular neighbor, so any agreed reference to time must arise from a distributed consensus.

One notable work preceding much of the others thus far using DLT is that of Crosby *et al.* [25] and presents efficient data structures for tamper-evident logging using history-trees. Although the validation using history-trees is efficient, $O(log_2 n)$, the runtime time for adding checkpoints is no longer constant, $O(log_2 n)$ rather than $O(1)$ for hash-lists. Thus given the lopsided computing resources between robots and off-line auditing infrastructure, our approach opts for hash-lists given the constant overhead in terms of log length, while introducing indexing to enable the parallelizion of auditing.

C. Event Data Recorders

Event Data Recorders (EDR) have become prevalent within the automotive industry, due in part to regulatory compliance from governmental safety legislation, as well as OEM incentives w.r.t. insurable liability and risk management. Reminiscent of Black Box Recorders in aviation, EDRs are used to log internal and external vehicular data during deployment, such as engine health and status, steering and brake operation, and accident reporting such as obstacle distances or inertial forces from impact. Among the list of transportation infrastructure primed to fully incorporate EDR deployments, autonomous driving vehicles are perhaps first among them. Questions now from both industry and regulatory agencies are being brought forth as per the privacy and security of such EDRs given the pervasive yet critical nature of the data they retain.

The work by Veitas *et al.* includes a two part series pertaining to these particular issues; the first presents Policy Scan [26], a methodology for technology strategy design; i.e. developing concrete actions and products for guiding technology adoption. Policy Scan was developed for the purpose of addressing specific types of ill-defined problems in terms of observing, analyzing and integrating technology developments with policy requirements, social governance and societal expectations. The second paper [8] applies Policy Scan to the domain of autonomous driving and smart mobility, presenting a proposal for making future autonomous vehicles within collaborative intelligent transportation systems (C-ITS) using EDR as more socially acceptable and legally compliant.

Building upon the above groundwork and also that from Taurer *et al.* [27], a bio-inspired approach to secure data recording for robots, we have designed BBR as an EDR implementation that conforms to the in-vehicle data recording, storage and access management requirements as specified, while also remaining extendable to general autonomous AI applications using open source robotic middleware and distributed ledger software.

III. EDR Roles, Requirements and Primitives

Here we formally define EDR systems in terms of the roles, requirements and primitives adapted from prior work [8], [27] to enumerate our design/implementation conformity. Boldface terms are later referenced when demonstrating compliance.

A. Obligated Roles and Observing Parties

Auditors: observing parties called upon to investigate and validate record archives. e.g. Regulatory Agencies or Gov.

Custodian: obligated subject of log content and tasked with log preservation. e.g. Robot or autonomous vehicle OEM.

Owner: mediating party that has a stake in ensuring log integrity/authenticity/confidentiality. e.g. End-User or Operator.

Reporter: an independent party responsible for faithfully recording events. e.g. Trusted Logger or Recorder Enclave.

B. Recording, Storage and Access Requirements

R1 *Data provision conditions*: requires consent on behalf of the *Owner* who transitively controls the log assets tracked.

R2 *Fair and undistorted competition*: trust should be distributed and shared across all validators (a.k.a *Custodians*).

R3 *Data privacy and data protection*: the co-location of logs external to that of the *Custodian* must be prevented.

R4 *Tamper-proof access and liability*: integrity and authenticity of logs must derive from an independent *Reporter*.

R5 *Data availability economy*: health and transparency of logs are contingent upon giving *Auditors* appropriate access.

[1]https://www.factom.com
[2]https://www.scytl.com/en

C. Defined Primitives and System Properties

P1 *Secure identification of physical data sources*: attestation between devices trusted by the *Custodian* and *Reporter*.

P2 *Metadata enrichment*: log event context may be associated to respective *Owner*, *Custodian* and *Reporter* parties.

P3 *Data exchange and messaging*: authenticated encryption is used in establishing secure connectivity between parties.

P4 *Data recording & storage*: reporting remains flexible in terms of QoS as well as reasonable in resource consumption.

P5 *Access management*: rights, obligations, and authorization of parties must be explicitly defined and enforceable.

IV. Approach

In our approach, using DLT in EDR requires the development of two principal components: the integrity proof coupled with the smart contract specification. This section details the design of and the justification for both to accommodate the constraints of mobile robots and open source frameworks.

A. Incremental Integrity Proof

To preserve the integrity of the logs without compromising system performance or publicly disclosing private log content, as in [19] we leverage the collision and pre-image resistance of HMAC [28] by chaining the log checkpoints together with key rotations, accommodating **R3**. Borrowing terminology established in [19] 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 message ($LogMsg_i$):

$$\begin{gathered} Chk_i = (i, h_i) \quad h_i = HMAC(h_{i-1}, LogMsg_i) \\ where \quad h_0 \xleftarrow{\$} \{0,1\}^m \end{gathered} \tag{1}$$

For privacy, a random nonce is included as the genesis digest (h_0) to inject initial entropy into the linked integrity proofs, ensuring that separate records with similar beginning contents do not repeat the same telltale signature of consecutive proofs.

This deviates from previous work that convolutes the log integrity proof with token based blockchains and previous financial transaction outputs to achieve immutability. By keying the HMAC with the previous checkpoint digest instead, we reduce the validation of logs to the trivial task of checking a simple hash-chain: i.e. sequentially iterating through $LogMsg_i$ in the log file, ensuring the last linked digest corresponds to the final proof published into the ledger, satisfying **R4**.

By including the index (i) into checkpoints, partial validation or triage discrepancies in the face of missing or corrupted log events can be fine-grained. Provided indices are similarly embedded in log content, validation over large log files is easily parallelizable, accelerating the total verification process.

Previous works such as [19], [22] make the distinction between two different types of checkpoint entries; the first being an incremental link in a chained proof, with the second being an anchor point that must always be published to commit to new secret keys while unveiling expired ones for later verification of integrity and authenticity. Our approach to checkpoints makes no such distinction, thus any checkpoint or sequence of checkpoints may be immediately published. This ensures that the latest checkpoints can always be submitted on short notice or without necessarily waiting for previous transactions to be finalized in the global blockchain.

For robotic applications in particular, where mobile computing may be subject to instantaneous brownouts due to self reliant energy supplies, integrity proofs that require stateful cryptography [19] could leave a recorder without recourse for resumption, as the previously finalized transactions would have included a commitment to a future temporary key that must be revealed upon the next checkpoint. Our approach permits the recorder to quickly recover from last known integrity proof and resume checkpointing the log wherever it left off (**P4**).

B. Smart Contract

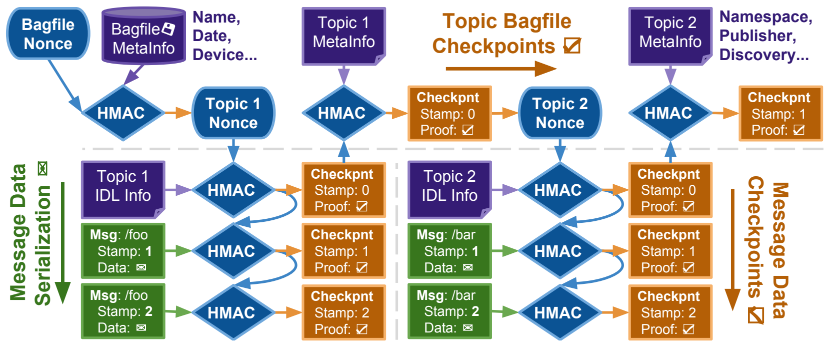
Section IV-A formalized the incremental integrity proof to ensure log file immutability; however, this efficient method of verification does not in and of itself offer the authenticity and non-repudiation properties still required. Smart Contracts (SC) encapsulate the access control logic for DLT validators to abide by when determining the validity of proposed checkpoint transactions, addressing **R1** and facilitating **R4**.

Instead of relying on colored coins or token metadata in financial blockchains to encode ownership, a dedicated transaction family is defined to regulate write access to ledger state. A common criteria however is that the validity of candidate transactions must be deterministically computable; i.e. no context external to the current state of the ledger and transaction payload in question should be used in deliberation. This ensures that the validity of any block in the chain can be independently verified in the future.

To ensure authenticity of a checkpoint committed into the blockchain, transactions are signed via an Elliptic Curve Digital Signature Algorithm (ECDSA), effectively notarizing the identity of the signer. For our purposes, we also register the identity into the blockchain by enrolling its public key into an access control policy stored in the distributed ledger to be used by SCs when verifying candidate checkpoint transactions. Thus, we limit recorders' permissions to append checkpoints only for log files priorly-authorized (**P2**).

To ensure non-repudiation of transactions, our SC mandates that checkpoint indices remain monotonically increasing. The striding of published checkpoint indexes is permitted to enable recorders to down-sample the rate at which integrity proofs are transmitted, versus rate locally generated, as a Quality of Service (QoS) to conserve energy or wireless network bandwidth and ensure sustainable size of the distributed ledger's state. To curtail the memory growth of the ledger, effectively a database each validator must maintain locally to participate, a paging ring buffer is adopted to keep rotating the n latest checkpoints for a given log file. The ring buffer size may also be allocated to comply with data retention window requirements per **R3**. However, the genesis digest is always preserved to ensure indefinite immutability of the entire log unto its first record.

A particular problem presented in previous work includes the open ended issue of finality of checkpoint termination, i.e. preventing further checkpoints for a given log from appending



Bagfile Nonce
Bagfile MetaInfo
Name, Date, Device...
Topic 1 MetaInfo
Topic Bagfile Checkpoints ☑
Topic 2 MetaInfo
Namespace, Publisher, Discovery...
HMAC
Topic 1 Nonce
Checkpnt Stamp: 0 Proof: ☑
Topic 2 Nonce
Checkpnt Stamp: 1 Proof: ☑
Message Data Serialization ⊠
Topic 1 IDL Info
Msg: /foo Stamp: 1 Data: ✉
Msg: /foo Stamp: 2 Data: ✉
Checkpnt Stamp: 0 Proof: ☑
Checkpnt Stamp: 1 Proof: ☑
Checkpnt Stamp: 2 Proof: ☑
Topic 2 IDL Info
Msg: /bar Stamp: 1 Data: ✉
Msg: /bar Stamp: 2 Data: ✉
Message Data Checkpoints ☑

(a) Dependency diagram of linked integrity proof for single bag file database.

Topics				BBR	
id	name	type	format	nonce	digest
1	/foo	gps	rtps	$bits_A$	$bits_{A_0}$
2	/bar	imu	rtps	$bits_B$	$bits_{B_0}$
3	/baz	lidar	rtps	$bits_C$	$bits_{C_0}$

Messages				BBR
id	topic	time	data	digest
1	id 1	1.0s	bits	$bits_{A_1}$
2	id 2	3.1s	bits	$bits_{B_1}$
3	id 1	4.5s	bits	$bits_{A_2}$

(b) Database tables for topics and messages.

Fig. 2: Commitments to data and its insertion into the database is achieved via 2D-array hash-chains, where the primary axis checkpoints each topic's genesis-block and meta-info, while the secondary axis checkpoints the insertion of respective message data. This coupling affords a holistic integrity proof of the entire database, while preserving topics as a time series atomic.

to the ledger after the log file is intentionally concluded. Such actions could be taken by recorders that are threatened or suspect intrusion, providing a self destructive deterrent and reducing utility of private keys recovered by an adversary.

Previous works using token based blockchains could conclude a checkpoint record via output transactions that are addressed to random public identities, for which no private key is known. This extreme all-or-nothing ownership forfeit is probabilistically final, but doesn't afford any other status for provenance, such as 'stalled', 'critical', or methods for reversal, useful for conditional resumption of logs after a situation is resolved. SCs instead provide greater granularity in this regard for regulating the life cycle of checkpoint records.

V. Implementation

As a proof of concept, we implement Black Block Recorder using existing open source robotic middleware and distributed ledger software. ROS2 was chosen, given its support for secure multicast networking (**P1**) and modular ROSBag2 plugin design, enabling secure and efficient tapping of internal/external robotic networks (**P3**). Hyperledger Sawtooth[3] was chosen as the ledger framework for its energy efficient yet BFT consensus algorithm, multilingual SC processors, permissioned DLT support, and parallelizable transaction architecture.

As both custodian and reporter parties manifest as physical CPS devices, their identities are particularly susceptible to attack. Here, both are used to co-sign batched transactions for validator submission; thus appended forgery checkpoints necessitates the corruption of both the custodian and reporter.

A. Checkpoint Integration

To integrate our linked checkpoint approach into ROSBag2, we extend the existing SQLite default storage plugin to additionally compute and broadcast the checkpoints. A 2D-array hash-chain is created to render bagfile databases into append only data structures. The following equations in conjunction with the table and color coded flow diagram in Fig 2 depict the process for checkpointing topic insertions:

[3]Hyperledger Sawtooth: hyperledger.org/projects/sawtooth

$$bits_{bag} \xleftarrow{\$} \{0,1\}^m \tag{2}$$
$$bits_A \leftarrow HMAC(bits_{bag}, Proto(name_{bag})) \tag{3}$$
$$bits_{A_0} \leftarrow HMAC(bits_A\ , Proto(type_A, format_A)) \tag{4}$$
$$bits_B \leftarrow HMAC(bits_{A_0}, Proto(name_A)) \tag{5}$$
$$bits_{B_0} \leftarrow HMAC(bits_B\ , Proto(type_B, format_B)) \tag{6}$$
$$bits_C \leftarrow HMAC(bits_{B_0}, Proto(name_B)) \tag{7}$$
$$bits_{C_0} \leftarrow HMAC(bits_C\ , Proto(type_C, format_C)) \tag{8}$$

The nonce (in blue) for bagfile is combined with bagfile metadata (in purple), deterministically serialized via protobuf to avoid the ambiguity in hashing a list of items, to generate the nonce for the first inserted topic. This is then combined with IDL information of the topic to generate the genesis digest. The previous topic's genesis digest and meta data is then combined to seed the nonce for the next topic, which is also reported as the checkpoint (in orange) for the bagfile itself (**P2**). Thereafter, the cycle repeats for each additional topic.

For messages (in green), the previous digest for the respective topic is combined with the message and its time of arrival to compute the current digest, shown below and in Fig 2:

$$bits_{A_1} \leftarrow HMAC(bits_{A_0}, Proto(time_{A_1}, data_{A_1})) \tag{9}$$
$$bits_{B_1} \leftarrow HMAC(bits_{B_0}, Proto(time_{B_1}, data_{B_1})) \tag{10}$$
$$bits_{A_2} \leftarrow HMAC(bits_{A_1}, Proto(time_{A_2}, data_{A_2})) \tag{11}$$

In this way, bagfile and message checkpoints are loosely coupled enough for auditing data provenance while remaining independent for concurrent computation and atomic record keeping, even across separate topic streams. An architecture diagram of a robot deployment using the BBR storage and bridging plugin can be viewed in Figure 3. Note the separable stages for recording vs signing checkpoints within an enclave in blue. This allows for modular integration for swapping database storage drivers or alternate ledger infrastructures.

B. Transaction Family for EDR Smart Contracts

To develop BBR's SCs, we extend from Sawtooth's reference supply chain Transaction Family (TF), used for tracing the

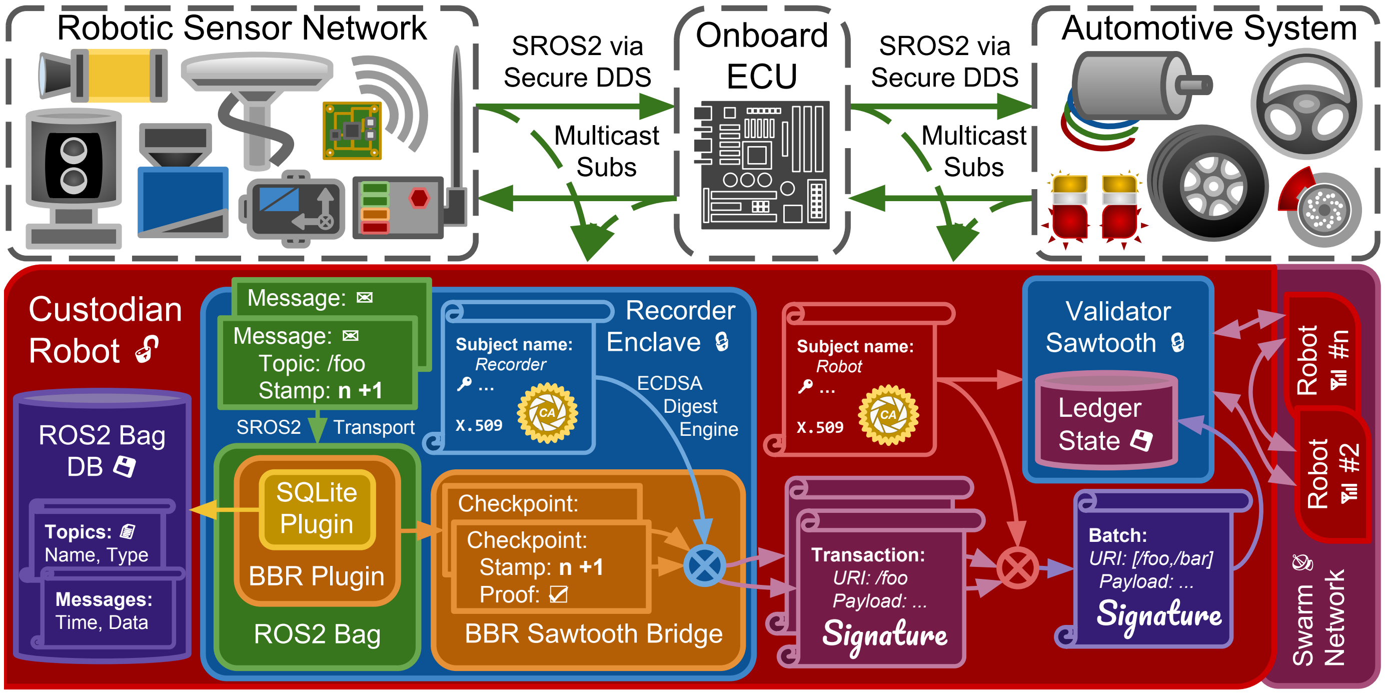


Fig. 3: Flow chart visualization of the immutable logging pipeline. While every robot platform is held suspect, a secure enclave (e.g. TTE) is reserved for the recorder process. Logged input is securely received within the enclave and used to cryptographically derive a linked integrity proof specific 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; to append a forgery would necessitate the collusion of both the custodian and its assigned recorder.

provenance and other contextual time series information of assets. This is formalized in Figure 4 by specifying our SC using the Digital Asset Modelling Language (DAML)[4], a open source domain specific language for expressing contracts, parties, rights, obligations, and authorization directly (**P5**).

The main SC for an EDR agreement is captured in lines 1-14 in Figure 4a, where the primary parties involved enter as signatories, while a set of external parties are provisioned observational access to the SC's state. Control for creating associative records is solely delegated to the recorder. Lines 23-40 capture the secondary SC specific to a particular record; i.e. log checkpoints for a single topic. Again the recorder is given the choice to append or finalize the record, while under the assertion that submitted checkpoints remain monotonic. The owner may also choose to finalize the record, whereupon the SC is archived and left entirely immutable in the DLT.

Lastly, lines 16-22 and 42-44 specify the structural data a recorder must submit upon choosing actions for aforementioned SCs. The complete DAML model, including the pending SC for establishing the multiple party agreement, as well as the test scenario depicted in Figure 4b has been open sourced and made publicly available[5]. As integration between DAML and Sawtooth is still in early development, the TF for BBR remains implemented in the Rust programming language. The DAML model is a faithful representation of the SC logic.

[4]DAML Specification: daml.com
[5]EDR DAML Model: github.com/dledr/edr_daml

C. Performance Profiling and QoS Tuning

As a preliminary validation for the tractability of using BBR in robotic systems, we provide a quantitative benchmark comparison in the overhead introduced by utilizing the BBR storage plugin and bridging interface by evaluating drop rate performance and CPU load over a range of common sensor message sizes and frequencies. Test results show BBR's current performance falls closely in line with the default driver plugin whilst single thread workload remains unsaturated. See Fig 5. Marginal performance gains during midrange workloads are likely artifacts attributed to fewer CPU cache misses, due to reduced process idle time given continuous overhead.

In regards to the depicted drop-off in throughput, though BBR seeks to checkpoint events at the write-rate to database, in practice the signing and transmission of those checkpoints over the bridge interface should be rate limited for purposes stated before, such as local QoS restrictions or moderating workloads for external validators (**R2**); given that ECDSA transaction signatures remain the predominate cryptographic bottleneck in the pipeline. Recall that as long as each event is incorporated into the hash-chain, down-sampling checkpoint publication would not inhibit the tamper-evident properties of the log segment with unpublished checkpoints, merely the resolution at which alterations may be pinpointed in the log.

In regards to uplink network usage, validator traffic is conditional upon consensus algorithm, gossip protocol, number of participants, and frequency/size of submitted transactions

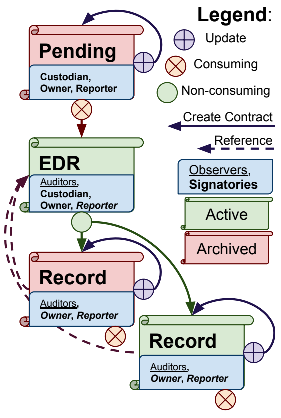
```
template Edr -- Smart Contract for EDRs
 with
   auditors: [Party] -- Regulatory Agencies
   custodian: Party  -- Robot/Vendor Identity
   owner: Party      -- User/Operator Identity
   reporter: Party   -- Logger/TEE Identity
 where
   signatory custodian, owner, reporter -- obligated
   observer auditors -- non-obligated parties
   ensure unique (custodian :: owner :: reporter)
   controller reporter can -- create many Records
     nonconsuming Edr_Record : ContractId EdrRecord
       with record: Record
       do create EdrRecord with edr = this; record

data Record = Record with  -- data type struct
   r_name: Text   -- "/sensors/exteroceptive/gps"
   r_type: Text   -- "gps"
   r_format: Text -- "rtps"
   r_nonce: Text  -- "<bits_A>"
   r_digest: Text -- "<bits_A_0>"
   r_checkpoints: [Checkpoint] -- monotonic list
```

```
template EdrRecord -- Smart Contract for Records in EDR
 with
   edr: Edr       -- Reference EDR of origin
   record: Record -- Initial Record state
 where
   signatory edr.owner, edr.reporter
   observer edr.auditors -- custodian can be excluded
   choice EdrRecord_Append : ContractId EdrRecord
       with checkpoints: [Checkpoint] -- [] for batching
       controller edr.reporter -- Only Reporter appends
       do let -- Update Record with added checkpoints
           is_valid = checkMonotonic record checkpoints
           _record = appendCheckpoints record checkpoints
         assert (is_valid == True) -- Error on invalid
         create EdrRecord with edr; record = _record
   choice EdrRecord_Finalize : () -- Archives Contract
     controller edr.owner, edr.reporter
       do return () -- Finalized Record is un-appendable

data Checkpoint = Checkpoint with -- data type struct
   c_proof: Text -- "<bits_A_1>"
   c_stamp: Int  -- 1
```

(a) Formal specification of Smart Contract model for Event Data Recording captured in DAML



(b) Smart Contract Diagram

Fig. 4: Shown is a simplistic contract model for EDRs using DLT specified via a domain specific modeling language for SCs. An EDR SC is first preliminarily proposed via a pending contract used to collect the necessary multi-party signatories. Once finalized the pending contract is consumed to create the agreed EDR SC, allowing the recorder to create multiple referencing records and append checkpoints, that itself or the owner may finalize. Respective SC diagram depicts a basic example scenario.

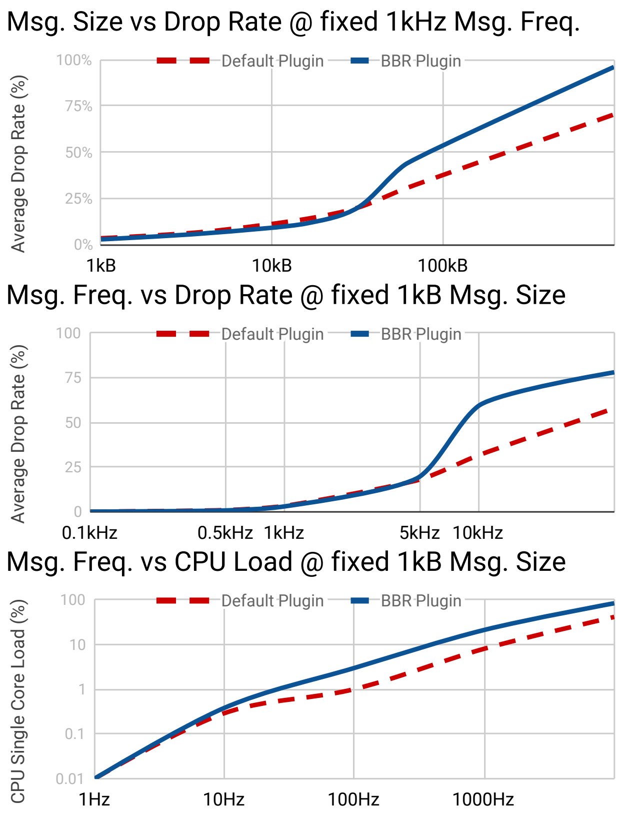


Fig. 5: Droprate/Load performance comparison of our BBR storage/bridge plugins with regard to ROSBag2's default SQLite plugin. Benchmarked via ROS2 Crystal, 2.6GHz Intel i7-6700HQ, with RTI Connext RMW on loopback interface.

specific to DLT implementation/framework used. However, to profile the network bandwidth usage specific to our BBR bridge, Table I includes the minimal payload requirements as calculated using the current serialization schema implemented.

This depicts the lower bounds given a signed batch transmission with a list of one signed transaction containing an array of one checkpoint. While transactions in general may contain arbitrary topic metadata, a checkpoint is simply a tuple of a 256bit hash and 32bit unsigned integer. As a reference, rosbag recording all 20 unique topics on a typical TurtleBot3 running ROS1 navigation stack writes to disk at about 1.4MB/s at 1k messages/sec., publishing every checkpoint batched at 1Hz results in approximately 400Kbps in BBR uplink overhead. In practice, a more sensible striding of one checkpoint per topic per sec reduces this to around 110Kbps instead, with a sustainable <1KB/s of ledger state growth (**P4**).

Payload	Size (bytes)	Requirements
Chk_i	36	Ledger Disk Storage
Signed Transaction	≥629	Network Bandwidth
Signed Batch	≥965	Network Bandwidth

TABLE I: BBR Payload Allocations

VI. Conclusion

In this work, we established a secure logging framework for robots using distributed ledgers and linked integrity proofs to insure the immutability of continuous event data records. Authenticity and non-repudiation are achieved via dissemination of checkpoint proofs and smart contracts that respect the nature of mutually distrusting parties involved while enforcing a contractual symbiosis between regulators, robots, and users.

If fact, while the BTF holds for the DLT network, compromising any single party involved in the EDR SC would not itself afford an untraceable appended forgery, nor would the exploitation of all EDR SC parties afford forgery of the past.

With respect to the whether the marginal overhead incurred in recording logs via BBR vs. conventional rosbag exceeds the benefits in ensuring event records remain tamper resistant, we conclude that given the appropriate QoS for reporting topics of significance, the practicality and utility of applying BBR in security sensitive robotic domains remains advantageous.

We expect this application domain for robotic EDRs will be one among many exciting frontiers to be explored along the intersection of cryptobotics and distributed ledgers, as the methods presented generalize across future DLT architectures.

Security of distributed ledgers can hinge upon the number and health of validators that govern global state, influencing **R2, R5**. If the validator pool becomes too small or dominated by a single party, community trust in the ledger can falter. To mitigate the monopolization of validation, more fog level IoT devices, ranging from stationary C-ITS infrastructure or mobile field robots, could be conscripted as additional validators to help bolster device diversity. To incentivize participation and avoid the tragedy of the commons, an alternate consensus protocol could be adopted. Tangle [29], a Directed Acyclic Graph (DAG) consensus algorithm used in IOTA, a DLT for IoT domains, leverages an auspicious pay-it-forward strategy: to issue transactions, an identity must first work to validate a greater number of transactions than it wishes to submit, thereby sustainably contributing to the network's security.

Recently, DAGs have been gaining traction over blockchain data structures due to their greater throughput of asynchronous transactions and relaxed connectivity requirements. For large scale robot deployments connected over temporally disjointed or semi-partitioned networks, DAGs could alleviate the continuous global connectivity requirements incurred when using traditional synchronous distributed data structures instead.

Lastly, given the designs presented for linked integrity proofs supporting parallel checkpoint amendments, the lack of any ROSBag2 storage plugin capable of taking full advantage of such concurrency remains a shortcoming. Thus, the development of such a storage plugin compatible with SQLite alternatives supporting parallel writes would be a boon for high-performance high-bandwidth message capture as well as immutable event data recording.

Acknowledgment

We would like to thank the Hyperledger project for their open source efforts and documentation; both were instrumental in developing this work and exploring its applications.

References

[1] J. McClean, C. Stull, C. Farrar, and D. Mascareñas, "A preliminary cyber-physical security assessment of the Robot Operating System (ROS)," vol. 8741, p. 874110, 2013. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2016189

[2] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," *Defcon 23*, vol. 2015, pp. 1–91, 2015. [Online]. Available: http://illmatics.com/RemoteCarHacking.pdf

[3] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Snachám, and S. Savage, "Experimental security analysis of a modern automobile," in *Proceedings - IEEE Symposium on Security and Privacy*, 2010, pp. 447–462.

[4] S. Checkoway, D. Mccoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," *System*, pp. 6–6, 2011. [Online]. Available: http://www.usenix.org/events/security/tech/full{_}papers/Checkoway.pdf

[5] C. Miller and C. Valasek, "A Survey of Remote Automotive Attack Surfaces," *Defcon 22*, pp. 1–90, 2014. [Online]. Available: http://illmatics.com/remoteattacksurfaces.pdf

[6] S. Morante, J. G. Victores, and C. Balaguer, "Cryptobotics: Why Robots Need Cyber Safety," *Frontiers in Robotics and AI*, vol. 2, no. September, pp. 23–26, sep 2015. [Online]. Available: http://journal.frontiersin.org/Article/10.3389/frobt.2015.00023/abstract

[7] R. Rowlingson and Q. Ltd, "A Ten Step Process for Forensic Readiness," *International Journal of Digital Evidence Winter*, vol. 2, no. 3, 2004.

[8] V. K. Veitas and S. Delaere, "In-vehicle data recording, storage and access management in autonomous vehicles," *arXiv preprint arXiv:1806.03243*, 2018.

[9] G. Becker, "Merkle signature schemes, merkle trees and their cryptanalysis," *Ruhr-University Bochum, Tech. Rep*, 2008.

[10] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: www.bitcoin.org

[11] BitFury Group and J. Garzik, "Public versus Private Blockchains. Part 2: Permissionless Blockchains," pp. 1–23, 2015. [Online]. Available: http://bitfury.com/content/5-white-papers-research/public-vs-private-pt2-1.pdf

[12] ——, "Public versus Private Blockchains. Part 1: Permissioned Blockchains," pp. 1–23, 2015. [Online]. Available: http://bitfury.com/content/5-white-papers-research/public-vs-private-pt1-1.pdf

[13] K. J. O'Dwyer and D. Malone, "Bitcoin mining and its energy footprint," *IET Conference Proceedings*, pp. 280–285(5), January 2014. [Online]. Available: http://digital-library.theiet.org/content/conferences/10.1049/cp.2014.0699

[14] V. Dhillon, D. Metcalf, and M. Hooper, *The Hyperledger Project*. Berkeley, CA: Apress, 2017, pp. 139–149. [Online]. Available: https://doi.org/10.1007/978-1-4842-3081-7_10

[15] D. Butin, M. Chicote, and D. L. Mtayer, "Log design for accountability," pp. 1–7, May 2013.

[16] J. Benet, "IPFS-content addressed, versioned, P2P file system," *arXiv preprint arXiv:1407.3561*, 2014. [Online]. Available: http://arxiv.org/abs/1407.3561

[17] N. Anderson, "Blockchain Technology: A Game-Changer in Accounting?" pp. 1–5, 2016. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/de/Documents/Innovation/Blockchain_A%20game-changer%20in%20accounting.pdf

[18] P. Snow, B. Deery, J. Lu, D. Johnston, and P. Kirby, "Factom business processes secured by immutable audit trails on the blockchain," *Whitepaper, Factom*, November 2014. [Online]. Available: https://github.com/FactomProject/FactomDocs/raw/master/Factom{_}Whitepaper.pdf

[19] J. Cucurull and J. Puiggalí, "Distributed Immutabilization of Secure Logs," ser. Lecture Notes in Computer Science, G. Barthe, E. Markatos, and P. Samarati, Eds. Springer International Publishing, 2016, vol. 9871, no. 2, pp. 122–137.

[20] M. Bartoletti and L. Pompianu, "An analysis of bitcoin op_return metadata," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 218–230.

[21] R. Matzutt, J. Hiller, M. Henze, J. H. Ziegeldorf, D. Müllmann, O. Hohlfeld, and K. Wehrle, "A quantitative analysis of the impact of arbitrary blockchain content on bitcoin," in *Proceedings of the 22nd International Conference on Financial Cryptography and Data Security (FC). Springer*, 2018.

[22] A. Sutton and R. Samavi, "Blockchain enabled privacy audit logs," in *International Semantic Web Conference*. Springer, 2017, pp. 645–660.

[23] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the Delivery of Blocks and Transactions in Bitcoin," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, 2015, pp. 692–705. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2810103.2813655

[24] P. Szalachowski, "(short paper) towards more reliable bitcoin timestamps," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, June 2018, pp. 101–104.

[25] S. A. Crosby and D. S. Wallach, "Efficient data structures for tamper-evident logging," in *Proceedings of the 18th Conference on USENIX Security Symposium*, ser. SSYM'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 317–334. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855768.1855788

[26] V. K. Veitas and S. Delaere, "Policy scan and technology strategy design methodology," *arXiv preprint arXiv:1806.03235*, 2018.

[27] S. Taurer, B. Dieber, and P. Schartner, "Secure data recording and bio-inspired functional integrity for intelligent robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 8723–8728.

[28] M. Bellare, "New proofs for nmac and hmac: Security without collision resistance," *Journal of Cryptology*, vol. 28, no. 4, pp. 844–878, Oct 2015. [Online]. Available: https://doi.org/10.1007/s00145-014-9185-x

[29] S. Popov, "The tangle," *Whitepaper, IOTA*, February 2018. [Online]. Available: https://assets.ctfassets.net/r1dr6vzfxhev/4i3OM9JTleiE8M6Y04Ii28/d58bc5bb71cebe4adc18fadea1a79037/Tangle_White_Paper_v1.4.2.pdf