Black Block Recorder: Immutable Black Box Logging via rosbag2 and DLTs

By: Ruffin White, Gianluca Caiazza
Motivation

With maturing R&D, robots have propagated into many industries and economies, including manufacturing, transport, logistics, and more.

As cyber-physical systems (CPS), the risks from and threats to robots can surpass that incurred by conventional Internet of Things (IoT) devices.

Yet, are the methods used to monitor/audit such CPS devices in pace with robotic proliferation?
Motivation

Event Data Recorders (EDRs) in automotive, or Black Box Flight recorders in aerospace industries serve a multitude of critical directives, including:

- Dataset & Model Acquisition
- Performance Benchmarking
- Quality Control and Testing
- Continuous Health Monitoring
- Diagnostic Debugging & Triage
- System Auditing & Verification
- Digital Forensic Investigations
Motivation

With high-stake deployments come high-stake consequences, thus the extraordinary incentives evident to secure vs. circumvent EDR devices:

- Hold liable parties accountable
- Guide regulatory & legislative policy
- Expunge incriminating evidence
- Conspire to falsify historic events

Directive: to verify the integrity, authenticity, and completeness of robotic logs under the threat of malicious/erroneous insertion, or omission.

Death by robot: the new mechanised danger in our changing world

As the use of autonomous machines increases in society, so too has the chance of robot-related fatalities.
Related Work


Table 1. Data immutabilized within the transaction output

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix</td>
<td>2</td>
<td>“SL”</td>
<td>OP_RETURN prefix</td>
</tr>
<tr>
<td>version</td>
<td>1</td>
<td>1</td>
<td>Version of the data structure</td>
</tr>
<tr>
<td>logId</td>
<td>16</td>
<td>-</td>
<td>Unique log identifier</td>
</tr>
<tr>
<td>Hj</td>
<td>32</td>
<td>-</td>
<td>Hash of the checkpoint</td>
</tr>
</tbody>
</table>
Related Work


Constraints

CPSs such as commercially viable mobile robots incur contains in addition to just cyber security:

- **Limited energy capacity**
  - Efficiency is key to prolong deployment
  - Minimize computational overhead

- **Build of Materials**
  - Practical costs and availability at scale
  - Consumer grade hardware compatible

- **Finite memory storage**
  - Event data recording must be continuous
  - Maximize data retention for auditability

- **Unreliable network connectivity**
  - Uplink bandwidth a precious commodity
  - Flexible QoS to minimize overhead
Roles & Obligated Parties

**Custodian:** Robot or autonomous vehicle OEM  
**Reporter:** Trusted Logger or Recorder Enclave  
**Owner:** End-User or Operator  
**Auditors:** Regulatory Agencies or Gov.  
**Attacker:** Malicious Actor
Party: Custodian

Obligated subject of log content and tasked with log preservation. e.g. Robot or autonomous vehicle OEM.
Party: Reporter

An independent party responsible for faithfully recording events. e.g. Trusted Logger or Recorder Enclave.
Party: Owner

Mediating party that has a stake in ensuring log integrity, authenticity, confidentiality. e.g. End-User or Operator.
Party: Auditor

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

I’m watching you, Dave.
Party: Attacker

Any party intent on modifying the record to falsify events. e.g. Malicious actor expunging incriminating evidence.

Noo, my Data!

HSSSSSS!!!
Storage Requirements

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

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

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

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

5. Data availability economy
   health and transparency of logs are contingent upon giving Auditors appropriate access.
System Properties

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

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

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

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

5. **Access management**
   - rights, obligations, and authorization of parties must be explicitly defined and enforceable.
Approach

- Introducing Black Block Recorder (BBR)
- To mediate separation of concerns, only the recording process need be enclaved
- Enclaved process generates the logs by capturing broadcasted message traffic
- Before streaming logs to arbitrary storage, data must first be rendered immutable
Approach

- Striding checkpoints are disseminated to the blockchain for eventual log auditing.
- Checkpoints are comprised of cryptographic linked integrity proofs that are indexed.
- Thus we explore EDRs based upon Distributed Ledger Technology (DLT).

\[ Chk_i = (i, h_i) \quad h_i = HMAC(h_{i-1}, LogMsg_i) \]
where \( h_0 \leftarrow \{0, 1\}^m \)
Information Flow

- While a robot platform is held suspect, an enclave (e.g. TTE) is reserved for the reporter
- Event data is received within the enclave via SROS2 using authenticated encryption
- Linked integrity proofs for each asset tracked are batched and signed via BBR rosbag plugin
- Batches are then signed within the enclave, then co-signed by the robot’s via BBR bridge
- Thus an append-only forgery to DLT record would necessitate the collusion of both the custodian and its respective assigned reporter
Schema Structure

- Recorded event data is structured using ROS2bag files and compatible SQLite schema.
- Data insertions into the database are achieved via 2D-array hash-chains of checkpoints.
- Where the primary axis checkpoints each bagfile 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, fostering parallelism.
Topics Schema

- The nonce and digest for each topic are retained to verify even IDL info & metadata.
- Here we’ll choose to hash the topic name metadata separate from the IDL info.
- This allows for message data to remain verifiable even when want to remap topics.
- As a superset of the default SQLite schema, bags from the BBR plugin remain compliant.

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>type</th>
<th>format</th>
<th>nonce</th>
<th>digest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/foo</td>
<td>gps</td>
<td>rtps</td>
<td>$\text{bits}_A$</td>
<td>$\text{bits}_{A0}$</td>
</tr>
<tr>
<td>2</td>
<td>/bar</td>
<td>imu</td>
<td>rtps</td>
<td>$\text{bits}_B$</td>
<td>$\text{bits}_{B0}$</td>
</tr>
<tr>
<td>3</td>
<td>/baz</td>
<td>lidar</td>
<td>rtps</td>
<td>$\text{bits}_C$</td>
<td>$\text{bits}_{C0}$</td>
</tr>
</tbody>
</table>

$\text{bits}_{bag} \leftarrow \{0, 1\}^m$

$\text{bits}_A \leftarrow \text{HMAC}(\text{bits}_{bag}, \text{Proto(name}_{bag}))$ (3)

$\text{bits}_{A0} \leftarrow \text{HMAC}(\text{bits}_A, \text{Proto(type}_{A}, \text{format}_{A}))$ (4)

$\text{bits}_B \leftarrow \text{HMAC}(\text{bits}_{A0}, \text{Proto(name}_{A}))$ (5)

$\text{bits}_{B0} \leftarrow \text{HMAC}(\text{bits}_B, \text{Proto(type}_{B}, \text{format}_{B}))$ (6)

$\text{bits}_C \leftarrow \text{HMAC}(\text{bits}_{B0}, \text{Proto(name}_{B}))$ (7)

$\text{bits}_{C0} \leftarrow \text{HMAC}(\text{bits}_C, \text{Proto(type}_{C}, \text{format}_{C}))$ (8)
Messages Schema

- Each message is checkpointed by the time of arrival and data payload received.
- Here we’ll choose not to hash the topic ID, given message is bound to the topic digest.

<table>
<thead>
<tr>
<th>Messages</th>
<th>BBR</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>topic</td>
</tr>
<tr>
<td>1</td>
<td>id 1</td>
</tr>
<tr>
<td>2</td>
<td>id 2</td>
</tr>
<tr>
<td>3</td>
<td>id 1</td>
</tr>
</tbody>
</table>

$bits_{A1} \leftarrow \text{HMAC}(bits_{A0}, \text{Proto}(time_{A1}, data_{A1}))$  \hspace{1cm} (9)

$bits_{B1} \leftarrow \text{HMAC}(bits_{B0}, \text{Proto}(time_{B1}, data_{B1}))$  \hspace{1cm} (10)

$bits_{A2} \leftarrow \text{HMAC}(bits_{A1}, \text{Proto}(time_{A2}, data_{A2}))$  \hspace{1cm} (11)

- This allows for message data to remain verifiable even when want to splice bags.
- Checkpoints link to previous message of the same topic, not necessarily previous insert.
Smart Contract

- Checkpoints ensure immutability, but not authenticity and non-repudiation by itself
- Smart Contracts (SC) encapsulate access control logic for state in permissioned DLTs
- Digital Asset Modeling Language (DAML) is used to formalize contract specification
- Record/Checkpoint data structures define the base abstractions used in SC modeling
Smart Contract

- An EDR SC is formed from parties involved voluntarily entering as obligated signatories
- External parties may also provisioned observational access to the SC state
- Control for creating new records, to retain checkpoints, is delegated to the reporter

```
1 template Edr -- Smart Contract for EDRs
2 with
3   auditors: [Party] -- Regulatory Agencies
4   custodian: Party -- Robot/Vendor Identity
5   owner: Party -- User/Operator Identity
6   reporter: Party -- Logger/TEE Identity
7 where
8   signatory custodian, owner, reporter -- obligated
9   observer auditors -- non-obligated parties
10  ensure unique (custodian :: owner :: reporter)
11  controller reporter can -- create many Records
12    non consuming Edr_Record : ContractEd Edr_Record
13      with record: Record
14      do create Record with edr = this; record
```
Smart Contract

- An Record SC is formed from a reference to the EDR SC and initial record struct object
- Only the **Reporter** is permitted the choice to append checkpoints into the ledger state
- Valid checkpoints are appended by asserting stamps remain monotonically increasing
- Both **Reporter** and **Owner** are permitted the choice to permanently finalize the record

```solidity
template Edr Record -- 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 : ContractEd Edr Record
  with checkpoints: [Checkpoint] -- [] for batching
  controller edr.reporter -- Only Reporter appends
  do let -- Update Record with added checkpoints
    is_valid = check Monotonic record checkpoints
    _record = append Checkpoints record checkpoints
  assert (is_valid == True) -- Error on invalid
  create Record with edr; record = _record
choice EdrRecord_Finalize : () -- Archives Contract
  controller edr.owner, edr.reporter
  do return () -- Finalized Record is un-appendable
```

Smart Contract

- An Record SC is formed from a reference to the EDR SC and initial record struct object
- Only the **Reporter** is permitted the choice to append checkpoints into the ledger state
- Valid checkpoints are appended by asserting stamps remain monotonically increasing
- Both **Reporter** and **Owner** are permitted the choice to permanently finalize the record
Smart Contract

- An EDR SC is preliminarily proposed via pending contract used to collect the necessary multi-party signatories.
- The pending SC is consumed to create the agreed EDR SC.
- The **Reporter** may create multiple referencing records and append checkpoints, that itself or the **Owner** may finalize.
Results

- Drop Rate & Load performance comparison of 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.

<table>
<thead>
<tr>
<th>Payload</th>
<th>Size (bytes)</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Chk_i$</td>
<td>36</td>
<td>Ledger Disk Storage</td>
</tr>
<tr>
<td>Signed Transaction</td>
<td>$\geq 629$</td>
<td>Network Bandwidth</td>
</tr>
<tr>
<td>Signed Batch</td>
<td>$\geq 965$</td>
<td>Network Bandwidth</td>
</tr>
</tbody>
</table>

TABLE I: BBR Payload Allocations
Conclusion

- Engineering trade offs always exist
  - Security vs. Performance
  - Immutability vs. Drop-Rate

- However trade offs can be balanced
  - Adjusting QoS for sensitive topics
  - Sparsity via Striding checkpoint

- Promising uses of DLTs in Cryptobotics
  - EDR applications leveraging distributed consensus
  - When Byzantine Fault Tolerance is advantageous

<table>
<thead>
<tr>
<th>Messages</th>
<th>BBR</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>topic</td>
</tr>
<tr>
<td>1</td>
<td>id 1</td>
</tr>
<tr>
<td>2</td>
<td>id 1</td>
</tr>
<tr>
<td>3</td>
<td>id 1</td>
</tr>
<tr>
<td>2</td>
<td>id 1</td>
</tr>
</tbody>
</table>
Future Work

- Benchmark with consensus
  - Profile validator overhead
  - Characterize scale limitations

- Explore alternative DLT paradigms, e.g.:
  - Directed Acyclic Graphs vs Blockchains
  - E.g. IOTA’s Tangle

- Record at the middleware level (DDS)
  - Capture all traffic like services/actions
  - E.g. Parameter updates, planner feedback
Acknowledgements
Links


GitHub
dledr/bbr_ros
Black Block Recorder with ROS2 | Immutable Logging via Blockchain - dledr/bbr_ros2

GitHub
ros2/sros2

2 tools to generate and distribute keys for SROS 2. Contribute to ros2/sros2 development by creating an account on GitHub.

Hyperledger
Hyperledger Sawtooth – Hyperledger
A modular platform for building, deploying, and running distributed ledgers.

Hyperledger
Hyperledger – Open Source Blockchain Technologies
Hyperledger is a multi-project open source collaborative effort hosted by The Linux Foundation, created to advance cross-industry blockchain technologies.

GitHub
digital-asset/daml
The DAML smart contract language. Contribute to digital-asset/daml development by creating an account on GitHub.
Questions?

**A113:** Full Autopilot Override
**BBR:** Black Block Recorder
**BFT:** Byzantine Fault Tolerance
**CPS:** Cyber-Physical System
**DAG:** Directed Acyclic Graph
**DFI:** Digital Forensic Investigation
**DLT:** Distributed Ledger Technology
**DSA:** Digital Signature Algorithm
**EDR:** Event Data Recorder
**MAC:** Message Authentication Code
**ROS:** Robot Operating System
**SROS:** Secure ROS
References


More about:
Ruffin: about.me/ruffin
Gianluca: about.me/caiazza