Accepted copy for Publication at the 26th Euromicro Conference Series on Digital System Design (DSD), 2023

Secure Data Acquisition for Battery Management Systems

Fikret Basic, Christian Seifert, Christian Steger
Institute of Technical Informatics
Graz University of Technology
Graz, Austria
{basic, christian.seifert, steger}@tugraz.at

Robert Kofler
R&D Battery Management Systems
NXP Semiconductors Austria GmbH Co & KG
Gratkorn, Austria
robert.kofler@nxp.com

***Abstract*—The growing awareness of environmental sustainability has led to new investments in the field of electric vehicles. One of the most expensive and important components of electric vehicles are their batteries, with battery management systems (BMS) being responsible for their control. New regulations, such as those of the European Union, aim to introduce battery passports as a way to track battery lifecycle from manufacturing, over second-life use, to recycling. Given the vast amount of data generated during the lifecycle of a battery, the current research is focused on combining BMS with cloud connectivity. However, not much research has yet been done in the area of BMS cloud security and secure data logging. To address this gap, we propose a novel solution for secure BMS data acquisition for on-premise and cloud environments. In this paper, we make two main contributions: a secure data structure for BMS logging and a secure architecture for transferring BMS data from its source to cloud and end systems. We demonstrate the feasibility of the design by developing a prototype with real components and evaluate it in terms of security and performance.**

***Index Terms*—Battery Management System; Security; Cyber-physical; Cloud; Battery; Passport; Logging; Authentication.**

I. Introduction

The energy and environmental crisis caused by ever-increasing carbon emissions have led to an enormous increase in demand for electric vehicles. Battery management systems (BMS) play an important role in modern electric and hybrid vehicles by providing safety control over the use of batteries, their most important operating resource. They ensure the safety of the human driver by detecting and mitigating potential safety risks in advance [1]–[3]. As the use of electric vehicles, e-bikes, mopeds, etc. increases, so does the need for more batteries and thus BMS. In the recent market study published by Meticulous Research, the battery market is projected to reach $175.11 billion during the forecast period between 2021 and 2028 [4]. The exponential growth of the battery market between 2018 and 2025 and the importance of the second-life battery use case have also been observed by H. Melin [5]. The ever-growing market brings new challenges and requires solutions that would allow easier tracking and monitoring of batteries by their associated BMS. It is desirable to use batteries more efficiently and enable easier replacement at the end of their lifecycle to reduce global battery waste [6], [7]. We see two main challenges that need to be addressed in the development of modern BMSs.

BMS data reliance. The first challenge with modern BMSs lies in the need for efficient and easily accessible logging of monitoring and diagnostic processes, given the vast amount of generated data [8], [9]. Local logging devices may have limited capacity, be difficult to access, or even interfere with the regular operation of the BMS. On the other hand, with new regulations in the European Union (EU) and other countries around the world, the use of cloud systems for battery monitoring is slowly becoming a reality [10]–[12]. Several research papers have already proposed methods and models that use cloud services to extend the usability of BMS in a system [3], [9], [13]–[15]. Cloud systems enable the creation of battery lifecycle profiles, a concept that considers the collection and storage of important battery-related data from the BMS. This is done to extend user services and add external monitoring and diagnostic control by providing higher computing power and faster processing [3], [16].

Notably, the use of cloud connectivity with BMS offers:

- Battery life tracking and predictive support in the form of artificial intelligence or digital twins [3], [17].
- Increased computational power and faster processing of BMS-related diagnostic data, such as state of health (SoH) and state of charge (SoC) [18].
- Faster fault detection and battery age improvement [9].
- The use of ”swarming” to collect and use data for predictive maintenance of not just one but multiple systems in a group, e.g., for vehicle fleets [3], [9].

However, relying only on cloud services has three major disadvantages [8], [18]: (i) it requires a constant Internet connection e.g. if an accident occurs in a tunnel, there is no way to safely rely on the data during this transition, (ii) there may be delays due to the multi-level technological services that provide update control, and (iii) changes in data legislations and business models that may affect or complicate future ownership or access to the stored cloud data. An adequate BMS data service design should focus fully or partially on the use of local, i.e., on-premise, data services alongside conventional cloud connectivity. As Neubauer et al. [19] noted, it should be possible to handle both the batteries' on-site measurements, as well as to track the average use over time to facilitate the second life use case.

BMS security. The second challenge for modern BMS is to provide an adequate and lightweight security design. Most of the current BMS cloud research focuses on predictive estimation, digital twins, and machine learning [3], [17], [18], leaving the area of BMS cloud security largely unexplored. A security architecture for advanced BMS communications must address all data transmission layers. At the BMS level, it is important to consider the security of collected battery diagnostic data. Manipulation of diagnostic data by malicious parties can lead to hazards, such as thermal runaway in vehicles [20]. It is also important to ensure that BMS data is only processed by authorized parties to protect user privacy [21]. In addition, BMS cloud connectivity suffers from threats and vulnerabilities common to general networks. Thus, a BMS must always consider protection against man-in-the-middle (MitM) attacks, unauthorized access to storage, and the use of outdated protocols [8].

Contributions. Our goal is to propose a solution to the upcoming BMS challenges and present a hybrid logging architecture that combines both on-premise and cloud services for BMS data logging. To this end, we propose:

- A general BMS data structure independent of any topology or use case aimed at BMS monitoring and diagnostic data handling, while addressing security requirements.
- Furthermore, we present a layered model for a secure BMS cloud architecture based on a centralized gateway. While several papers have been published recently on cloud utilization with BMS, most of them are based on data-driven models, data propagation, or cloud-enhanced algorithms.

To the best of our knowledge, this is the first work in the field of cloud BMS that places data logging structure reliance and security as the primary focus.

II. Background

A. BMS and cloud computing

A BMS is a system responsible for the safety control of a large set of battery packs. Namely, they are accountable for battery cell monitoring, diagnostics, overall system safety, charging, cell balancing, and controlling the optimal discharging use of battery packs [1]–[3]. There are several established BMS network topologies, with modern ones being primarily based on modulated and distributed architectures having a central BMS controller with several battery pack controllers (BPC) [16], [22]. The majority of today's cloud systems are based on providing services for data storage, processing, and sharing. They aim to provide flexible and extensible services to end users behind a complex facade. As such, cloud systems provide modern solutions for processing the vast amount of BMS data and enable services such as remote monitoring and predictive maintenance [17], [18].

B. Logging BMS data

Modern BMS are responsible for processing a large amount of generated data. This data must be logged, either internally in the BMS with special modules or externally, e.g. in the cloud. The challenge here is to specify a flexible BMS data structure for efficient transmission. The size of the logged data depends on the tracked parameters, sampling frequency, and compression [8]. The BMS are responsible for interpreting the diagnostic data derived from the BPC or directly from the battery cells. For our analysis, we will focus on the following three main groups of data: (i) monitored data; which considers on-board read sensor or other measured data, e.g., battery cell voltage and temperature, (ii) diagnostic data, i.e. derived data based on observations, usually from a BPC, another module, or directly from the main BMS controller, and transmitted either as raw data or information such as SoC or SoH, (iii) fault diagnostic data, i.e., raw data derived from the register responsible for tracking the parameters of individual battery cells. Traditionally, most of the data generated on the BMS side was only stored locally for active monitoring. With the new initiatives related to the battery passport and secondary use, portions of this data would also need to be stored or sent to other systems to maintain tracking of the battery pack life cycle [11], [12]. For the remainder of this paper, we will refer to any BMS stored data as the BMS logging (log) data.

C. Battery passports and second life

Electric vehicle (EV) batteries reach the end of their life after 8 to 10 years of use, i.e., after they have dropped to 80% of their full capacity [7], [23]. After that, the batteries are either recycled or disassembled [5]. This is becoming a problem as the market for batteries keeps increasing, the rate at which they are recycled becomes limited and expensive, which also creates more environmental waste [5]. There is an initiative to allow batteries that have reached the end of their life in EV to be used for other applications, such as self-consumption in households or transmission deferral from EV to power grids [7], [24]. The EU Commission proposes the use of battery passports to track the lifecycle of batteries [10]–[12]. The battery passport is intended to be a digital representation of a battery that conveys all-important product information [10], [12]. An extension of this concept would be the battery e-passport, which could also dynamically record battery charge and discharge cycles, diagnostic information, faults, cell health, etc. This information could be used for rapid processing when the battery gets a second life in other applications during the disassembly process [5], [7]. Cloud systems provide such a solution, but there is currently no clear answer to secure BMS processing from the local to the cloud level, which we aim to extend in this work.

III. The Novel Secure BMS Data Structure Design

Based on the BMS lifecycle monitoring requirements, we propose a BMS data structure design based on a hierarchical distribution and differentiate between (i) log blocks, (ii) BMS blocks, and (iii) secure BMS blocks. A "*Log block*" (List. 3 with List. 1 & List. 2) contains a log header and a log body, where the log body contains the logged data based on a given structure. Log blocks form a payload that is represented by the "*BMS block*" (List. 4). To keep track of the sampling order,

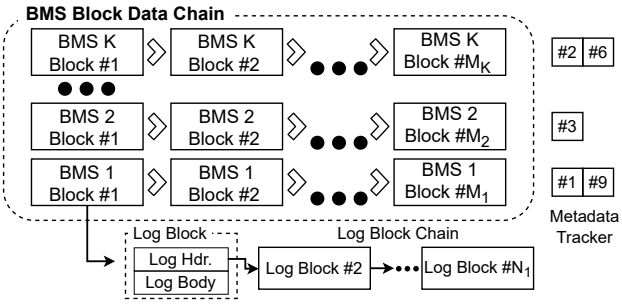

Fig. 1. BMS block data chain structure for logging of BMS lifecycle data.

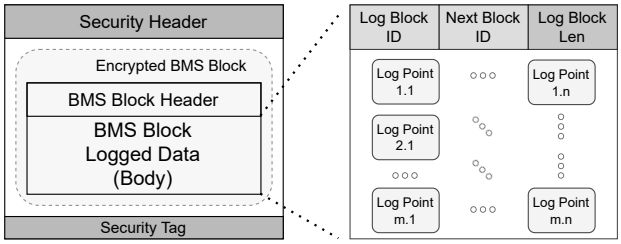

Fig. 2. Proposed design of the secure BMS data block.

the BMS blocks are intended to be stored in a data chain structure, as shown in Figure 1. Each BMS block contains a pointer to the first log block, which in turn contains a pointer to the next log block in the sequence. They are intended to encapsulate one BMS sub-system, with the "log chain" containing individual log samples per battery pack. The BMS block data chain identifies each individual sample point. In this example, '*M*' indicates the number of currently logged BMS blocks in a BMS sub-system, while '*N*' presents the total number of log blocks per one BMS block. '*K*' is the number of tracked BMS sub-systems. The sequence of the BMS blocks is determined by their timestamp field. Battery passport data is contained in the metadata field. To save space, metadata may be sent only when its contents change, for example, when the BMS sub-system changes its configuration or its host system. An array can be implemented that tracks BMS blocks with major status changes. Each time a new BMS block is received that contains metadata, the tracker array is incremented by the identifier of that block.

The advantage of the proposed data model structure is that it can be used across all different BMS topologies [22] with the following considerations for one full sample:

- Centralized: 1 BMS block, 1 Log block.
- Modulated: 1 BMS block, N Log blocks
- Distributed: 1 BMS block, N Log blocks
- Decentralized: K BMS blocks, $N_1, ..., N_K$ Log blocks

To guard against eavesdropping or other potential manipulations with logged BMS data, the application data exchange is protected via the "*Secure BMS block*" (List. 5), which uses the BMS block and attached log blocks as input for the encryption payload. Once the secure BMS block has successfully arrived at the end system, it can be decrypted and integrated into the log chain structure. Figure 2 shows the structure of the secure BMS block that is transmitted with each log sample. The secure BMS block consists of the unencrypted security header, the encrypted BMS block, and a security tag as a footer. The header contains information such as the current BMS secure identifier and cipher suite code. The security tag is used to verify the authenticity and integrity of the BMS block. It can be computed using the Message Authentication Code (MAC) or another security-related tag operation. The BMS block also contains its own header, logged data, and optional metadata. We propose that the header of the BMS block contain at least a unique identifier, a timestamp, the classifier ID, and the pointer to the first BMS log block. The protocol block contains at least its identifier, the identifier of the next block in the chain, and the block length.

Struct *LogSample* **contains**
  Log_Meas* measurements;
  Log_Diag* diagnostics;
  Log_Fault* fault_regs;
end

Listing 1: Data logging sample data structure.

Struct *LogBlockHdr* **contains**
  int block_id;
  int next_block_id;
  uint32 block_body_len;
end

Listing 2: Log block header data structure.

Struct *LogBlock* **contains**
  LogBlockHdr log_header;
  LogSample log_body;
end

Listing 3: Log block with log header and body entries.

Struct *BmsBlock* **contains**
  uint32 bms_block_id;
  uint32 timestamp;
  uint16 unit_id;
  int init_log_block_id;
  uint16 metadata_len;
  Bms_Metadata bms_metadata;
end

Listing 4: BMS block structure with optional metadata.

Struct *SecBmsBlock* **contains**
  uint16 version;
  uint16 length;
  uint32 sec_bms_block_id;
  uint32 sec_bms_block_serial;
  uint16 cipher_info;
  uint16 enc_bms_block_len;
  uint8* iv;
  uint8* enc_bms_block;
  uint8* mac;
end

Listing 5: Secure BMS block for symmetric crypto.

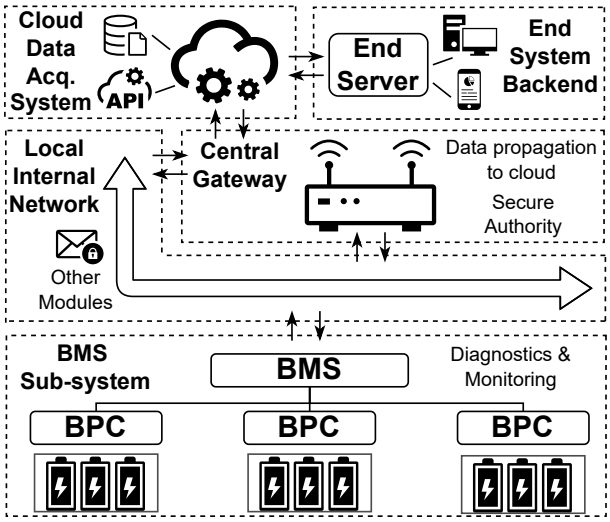

Fig. 3. Proposed BMS architecture for battery lifecycle monitoring.

IV. Secure BMS Data Acquisition Architecture

A. Security requirements

We observe the secure architecture from two perspectives: (i) the security of "cloud layers", i.e., the security on and from the central gateway, to the cloud acquisition system, and finally the end system, and (ii) BMS sub-system and its network.

For a BMS, security can be considered as (i) security during data transmission and (ii) security within the device. We want to ensure that the data from the source (battery sensors) to the end device (cloud system or end-user systems) is not compromised [6], [22]. Attacks targeting the BMS itself would be difficult to carry out because BMS communicate with battery packs that are enclosed and isolated. Still, various attacks could take place, usually in the form of spoofed devices or remote attacks if a vulnerability is found [6], [16]. MitM attacks are possible either between the BMS and the central gateway or when connecting to the cloud system. Data must be protected against these forms of attack by guaranteeing its authenticity, integrity, and confidentiality [21]. In addition, protection must be provided by message counters and inspections to ward off various replay attacks. Other attacks may take the form of denial-of-service (DoS) attacks, which would target either the cloud systems or the local networks. The attacker can also target the log content itself by launching attacks on delayed, reordered, or manipulated packets [25]. Accurate implementation and validation on the end system side should be performed to mitigate these types of attacks.

To support the proposed security architecture and secure BMS data structure, we observe the following design points:

- The security architecture with the cloud system is done over a trusted and verifiable service.
- The key generation and distribution by the original equipment manufacturer (OEM) are based on a trusted design, i.e., the end system device can securely receive the key necessary to decrypt the received BMS log data.
- Security operations are done over a trusted secure module to mitigate hardware-based vulnerabilities.

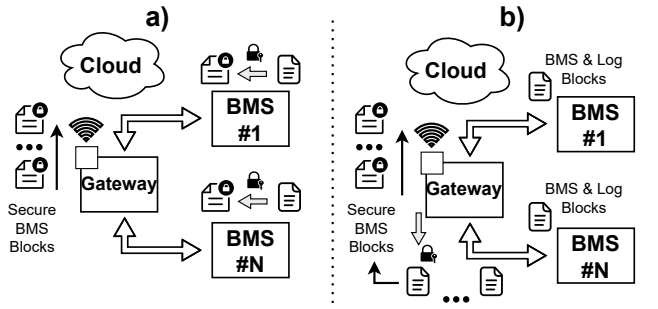

Fig. 4. Suggested secure BMS block data on-premise processing methods: a) security layer already added on BMS, b) security layer added on the gateway.

B. System layers

While different publications might refer to them with different notations, BMS cloud architectures generally consist of a perceptual layer, a network layer, and an end-user application layer [2], [15]. We further divide these layers to account for two additional middle layers to consider the on-premise BMS activities. Our main architecture design is presented in Figure 3. The system consists of five main layers: (i) BMS sub-systems, (ii) internal local network, (iii) central gateway, (iv) cloud data acquisition system, and (v) end system backend.

BMS sub-system. This layer considers the BMS and its connected battery packs as an independent entity. As an enclosed system, it is difficult to perform attacks from the outside. Nevertheless, it is recommended to provide authentication at the device level. The main BMS controller is responsible for data collection and preparation of the secure BMS blocks. This should be done on the device using a secure module. Due to widespread use, it is proposed to rely on symmetric encryption algorithms, specifically the traditional block ciphers, e.g., Advanced Encryption Standard (AES) with modes or authentication encryption (AE) primitives.

Internal local network. At the internal network system layer, device authentication, key derivation, and session establishment are performed using an appropriate security architecture [26]. This considers the communication between the BMS controller, the gateway, and any other local device. The secure session is established using either static or dynamic keys, taking into account perfect forward secrecy [27].

Central gateway (GW). The central device responsible for collecting BMS log data. It enables the connection with the cloud system. It is also the secure authority for the local network. For performance reasons, the rest of the data encapsulation is done on the GW side to prepare the BMS blocks for transmission over the cloud to the end system.

Cloud data acquisition system. Cloud systems rely on proven and robust security protocols. Cloud data communications for IoT solutions typically rely on the use of the underlying Transport Layer Security (TLS) or Datagram-TLS (DTLS) layers. At the application layer, DTLS is often used with the CoAP protocol and TLS with Message Queuing Telemetry Transport (MQTT) [28]. DTLS improves performance, but data might be lost, requiring retransmissions. The use of the right protocol depends primarily on the intended use case, with

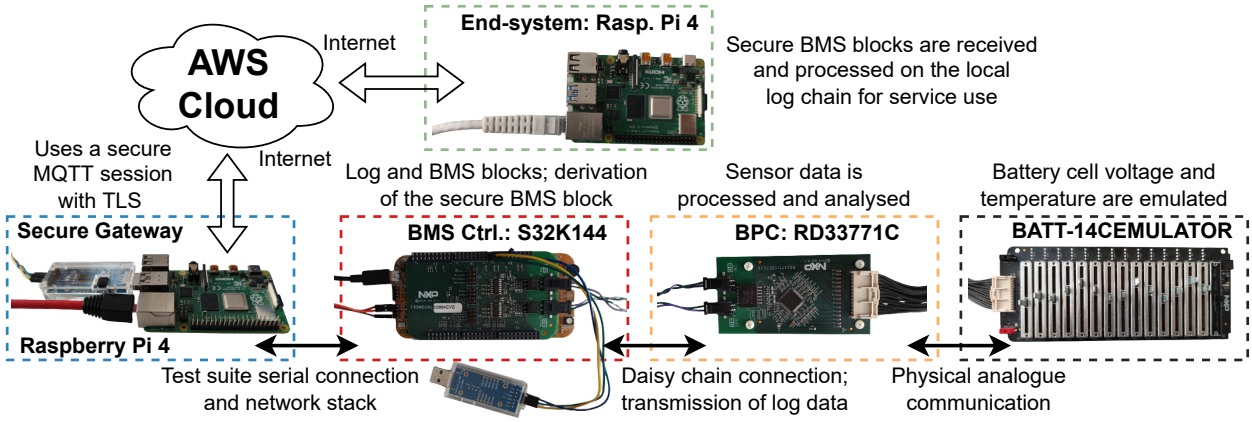

Fig. 5. Deployed BMS test suite. BMS sub-system: S32K144 BMS ctrl. with BPCs and battery emulators, Internal local network: implemented full local secure stack over a serial link, Central gateway: Rasp. Pi 4, Cloud data acq. system: Amazon web server with device shadow, End system backend: Rasp. Pi 4.

BMS relying on stricter safety standards and regulations that must be supplemented. It should be noted that both TLS and DTLS provide data protection from BMS to the gateway, but not for the end-to-end devices, i.e., from a BMS to the end device, which is why we also consider security at other layers.

End system backend. This considers any server or user device that handles the BMS log data for data processing, visualization, etc. Specifically, it is a device that relies on the battery e-passport services for lifecycle tracking and one that is authorized to access the encrypted secure BMS block data.

C. End system infrastructure discussion

For the presented architecture, an effort was made to keep the design generic and flexible regardless of the underlying session key exchange design. The end system receives the encrypted BMS data, but the means of processing this data is left open. Either the same or an additional external server would need to be used to disseminate and share the necessary certificates and other security-related configuration data. A similar concept could be adopted from the proposed standard ISO 15118 regarding the distribution of certificates to the respective OEM [29], [30]. Security solutions such as end-to-end encryption with adjustments could also be employed [31].

For processing individual BMS blocks to secure BMS blocks, we observe two approaches as seen in Figure 4:

a) Security functionality and storage are performed on the main BMS controller, i.e., it sends full encrypted data.
b) Security handling is performed on the gateway device.

We focus on the first approach, where data processing is performed on individual main BMS controllers. Here, the gateway acts as a buffer and bridge to ensure that each data block is correctly received and processed to the cloud service. This is an advantage for ad hoc BMS sub-systems that may store intermediate data between sessions or rely only on the on-premise use case. It is also more flexible for decentralized topologies where processing between BMS units is independent and therefore there is no bottleneck at the gateway. In addition, there are systems where there may not be a secure gateway and communication with cloud systems is done directly through the main BMS controller.

V. System Prototype Implementation

We implemented our proposed design on real hardware to demonstrate the feasibility of the presented methods, shown in Figure 5. A BMS emulator from NXP Semiconductors was used, consisting of an S32K144 microcontroller as the BMS controller, an RD33771C as the BPC, and a battery emulator that generates battery voltage and temperature data. To emulate local network communication, we used a Raspberry Pi 4 as the gateway device. Appropriate software was implemented and integrated for both the S32K144 and the Raspberry Pi to enable encapsulation of the test data and secure communication transmission. The security functionality was provided via the BearSSL library [32]. Security communication handling and secure BMS blocks were implemented based on an existing BMS diagnostic functionality. The security architecture for authentication and session key derivation is based on an Elliptic Curve Qu-Vanstone model. The communication between the gateway and the BMS controller is done over a serial link configured at a baud rate of 57 kbps, and uses the implemented network stack discussed in Section V-A. A symmetric cryptographic model was used for the secure BMS block, relying on the AES and hash-MAC (HMAC).

A. Test suite communication model

The test suite relies on the use of different network communication layers. The structures of the data packets used at each layer can be seen in Figure 6. The format of the data layer can be tailored to the needs of the target system. In our case, it is aimed at the communication between the secure gateway and the BMS controller, but it remains flexible and open.

The application layer is responsible for transmitting application-specific data, i.e., log data. The secure BMS block is included as a payload for our test cases. The application payload data is encrypted and tagged along with the added header to protect data confidentiality and integrity in the internal network. The transport layer allows for the fragmentation of large payloads. In our test case, the data-link layer packets can only contain up to 255 bytes in one packet, so the additional transport layer is required. It is modeled after the ISO 15765-2 standard, which is also used in similar environments [33].

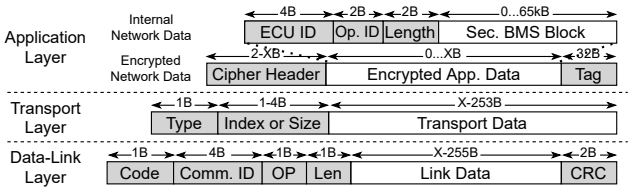

Fig. 6. Developed network communication model for the BMS test suite.

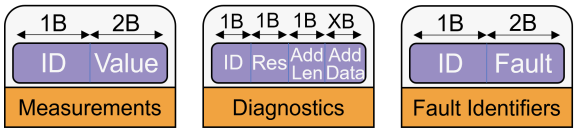

Fig. 7. Derived BMS log data model for the lifecycle monitoring.

B. Cloud setup

Cloud hosting is done on an Amazon web server (AWS). The gateway communicates with AWS and sends data using the secure MQTT protocol. After receiving the BMS block data, AWS propagates it directly to the assigned end system using an HTTPS REST push request. Both rely on the TLS protocol. In our test environment, the end system is represented by a Raspberry Pi 4 hosting a local server. The end system hosts a web application to display the captured BMS lifecycle data, running on a Flask server.

C. BMS data structure preprocessing

The battery log data collected by BPC and processed by BMS is based on the structure described in Section II-B and with List. 1, with the format shown in Figure 7. We optimize the processed data to incur as little overhead as possible. The monitored data includes a total of 3 bytes, one for the ID and two for the raw value. At least 3 bytes are allocated to the diagnostic data: one for ID, one for the diagnostic status, and one for the length of the additional data. The additional data is application specific. If there is no additional data per diagnostic entry, the length of the additional data is zero. The total amount of data for a log sample is 162 bytes. We assume that our system will have no more than 256 monitoring and diagnostic entries per BPC, so only one byte is reserved for identification. Otherwise, this entry could also be extended.

VI. Evaluation

A. Security analysis

We analyze the proposed design in terms of achieved security. The analysis is based on the security requirements described in Section IV-A. For our analysis, we derive *assets* {A}, *threats* {T}, *countermeasures* {C}, and *assumptions* {As}. The following assets are derived, i.e., the objects of protection: (A1) BMS log data, (A2) gateway-to-cloud payload, and (A3) cloud-to-end-system payload. To limit the security analysis to our proposed solution, we make the following assumptions: (As1) battery sensors, BPC and their channels are considered secure and trusted, (As2) no attack in the form of physical tampering is possible, (As3) security functions and keys are stored in a protected storage environment, (As4) cloud and end system are protected against

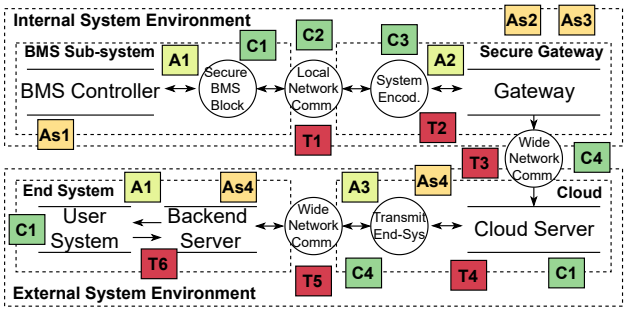

Fig. 8. Data flow diagram of BMS lifecycle monitoring security analysis.

common online threats. The security analysis model is built using the data flow diagram (DFD), with the derived model and results shown in Figure 8.

From the analysis, we list each potential threat with the targeted asset and analysed countermeasure strategies. Under "network attacks", we consider eavesdropping, tampering, replay, and MitM attacks on the messages. The threats are:

- [T1] Network attack on the BMS sent log data $\mapsto (A1)$, (C1) using secure BMS block with encryption and MAC, and with (C2) pre-established secure network session.
- [T2] Spoofing attack on the gateway $\mapsto (A1), (A2)$, (C3) the gateway is a secure authority with authentication.
- [T3] Network attack on the pushed cloud data $\mapsto (A2)$, (C4) uses secure MQTT, and TLS with certificates.
- [T4] Spoofing and privacy attacks on the cloud $\mapsto (A2)$, even if compromised, BMS data is protected with (C1).
- [T5] Network attack on the end system transfer $\mapsto (A3)$, (C4) again, with TLS and HTTPS for the pull request.
- [T6] BMS log data confidentiality compromise $\mapsto (A1)$, similar to [T4], (C1) mitigates any unauthorized readout.

B. Overhead analysis

The data overhead comes in the form of additional header data. It can be divided into two parts: static and dynamic data. The static data has a fixed size, regardless of the amount of processed log data for each secure BMS block. Here we refer to the variable sizes from Section III. Both secure and standard BMS blocks require 16 bytes each, with each additional log block requiring 12 bytes for its own header, regardless of the total size of the log body. Thus, the formula for calculating the total header size is $32+12*X$ bytes, where 'X' is the total number of log blocks. The dynamic component entails only the metadata in the BMS block, which is optional and depends on the implementation. It also depends on the underlying cypher protocols, i.e., larger block sizes mean larger key and MAC values. The total log block header size is also dynamic, as it depends on the total number of log blocks, i.e., BPC. Figure 9 shows the theoretical data overhead relative to the total secure BMS block size with variable log block length, where Table I shows the overhead analysis for the test suite with a variable number of log blocks, i.e., each with a size of 162 bytes. As noted, the secure BMS block data structure allows for minimal impact on overhead when deployed in a real-world environment, as it does not scale with the log block size.

TABLE I
OVERHEAD PER BMS BLOCK COMPARED TO THE NUM. OF LOG BLOCKS.

# of Log Blocks	1	2	4	8	12	16	32
Overhead (%)	21.4	14.7	11.0	9.0	8.3	8.0	7.4

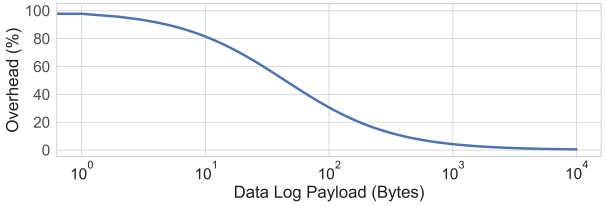

Fig. 9. Secure BMS block overhead in relation to log payload for one block.

TABLE II
BMS BLOCK ENCODING TIME WITH THE EMULATED DEVICES.

Encoding	Log body	BMS block	Secure block	Secure network
1 BPC	$0.09\,ms$	$0.24\,ms$	$18.16\,ms$	$20.68\,ms$
2 BPC	$0.17\,ms$	$0.38\,ms$	$23.63\,ms$	$26.33\,ms$

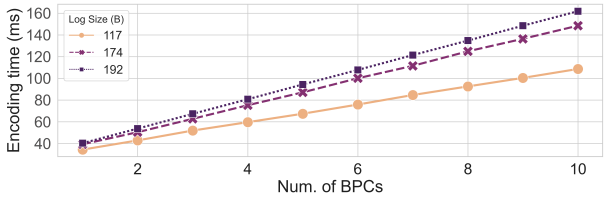

Fig. 10. Simulated BMS log encoding time for 117, 174 & 192 B log sizes.

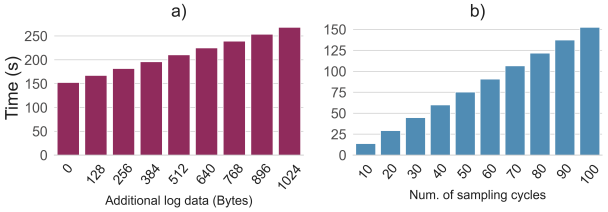

Fig. 11. Complete implementation run times for lifecycle monitoring with: a) variable additional log data, b) incrementing number of cycles.

C. BMS block encoding analysis

We analyze the encoding on the BMS controller using: (i) real hardware with emulated battery data, and (ii) simulated log input to the BMS controller for greater sizes. Under encoding, we consider the time of secure BMS block preparation after sampling and also the secure local network packet encoding. Table II shows the average results after one hundred test runs for individual encoding phases using emulators. The standard deviation is not included as it was negligible and $< 0.01\,ms$ for all test cases. As concluded, most of the encoding time is spent on security functions, which means that their optimization primarily affects the duration of the encoding process. Figure 10 shows the total encoding time for the simulated data. It shows the linear growth of the encoding time for three log block sizes compared to up to 10 BPC.

For the gateway, we want to ensure that the following goal is met: keep the transmission and processing time at a minimum, with the lowest time equal to the total sampling and processing time on the BMS controllers. The time for the decoding of the secure BMS blocks, i.e., the decryption from the application layer, MAC verification, and data extraction, is negligible compared to the total BMS processing time. Full network decoding accounts for $1.35\,ms \pm 0.11\,ms$, where decoding of the secure BMS block is $0.48\,ms \pm 0.01\,ms$. We note that this metric is highly dependent on the system and implementation used, but we can assume based on the devices used for this test suite that the same criteria would also be met in real systems. The sampling rate of the battery data is application dependent, but in our test suite, it was $\approx 112\,ms$ per BPC.

D. Transmission measurement

The tests were performed with one BPC on real hardware. The first step is the transmission from the BMS to the gateway, which takes $85.2\,ms \pm 3\,ms$. After decoding, the transmission via the gateway (Rasp. Pi 4) to the cloud (AWS) works on the *device shadow* principle, i.e., data is automatically forwarded when the BMS shadow is updated to ensure that log data is read only when needed. This step requires a total of $1.37\,s \pm 0.2\,s$ per request. After receiving the data from the gateway, the AWS forwards it to the end system (Rasp. Pi 4), which then decodes it and further processes the BMS block, requiring only $1.6\,ms \pm 0.4\,ms$. The Rasp. Pi measurements were averaged after one hundred runs. We have also tested decoding by adding additional data per message at various intervals up to 1 kB, but found only a small increase in processing time. The reported time for the additional 1 kB payload is $1.39\,s \pm 0.21\,s$ for the gateway and $2.2\,ms \pm 0.9\,ms$ for the end system. For a complete run, we took measurements from the battery sampling to the end system, as shown in Figure 11. The first plot shows the variation in time over the increase in additional logging data for one hundred cycles, while the second plot shows the total time for a different number of sampling cycles with no additional data.

We see that the main bottleneck is in the transmission of data to the cloud system, where multiple BMS sampling cycles can be performed during one cloud request. However, as mentioned earlier, even without considering further optimizations, the data is temporarily stored on the gateway device and can be pushed on the next request. The BMS can continue to safely sample new data without having to change its operating rate.

VII. RELATED WORK

The use of the cloud in conjunction with BMS has gained significant momentum in recent years, although many questions remain, especially those related to data storage and distribution of services [13]. While most of the publications are from recent years [15], some of the earlier proposals came from industry, notably from Fujitsu, where a system was envisioned that combines the use of cloud services for battery-sharing information between BMS [14]. More recently, Yang et al. [15] present a BMS cloud architecture based on the cyber hierarchy and interactive network framework, although they do not look into the BMS data acquisition design.

Digital twins are becoming increasingly associated with BMS in the context of cloud connectivity. In this area, Li

et al. [3] describe a model for comparing measured battery data and estimated digital twin data, with estimates based on the use of extended H-infinity filters and particle swarm optimization. Similarly, Wu et al. [17] present a cloud-side data-driven solution for BMS SoH estimation by focusing on machine learning methods for input noise reduction and using random forest regression to build a battery degradation model.

Concerning data logging, Mansor et al. [34] propose a secure logging approach for vehicles based on the use of mobile and cloud applications. Their focus is on the use of hardware security modules for in-vehicle units, such as those proposed in the EVITA project [35]. For solutions specifically targeting BMS, Zhou et al. [8] present a frequency division based storage and compression method that can be used for BMS log data. In their work, they also present three main requirements for using large battery storage. Among them, the limitations of the communication technology used in terms of data rate as well as the duration of data storage are argued. In our paper, we propose a design that is independent of these system constraints. However, we do consider the amount of argued data as one of the requirements which we discuss in Sect. III and implement in Sect. V. The paper also points out the possibility of bottlenecks when transmitting a large amount of BMS data. Our design provides a solution to this challenge by partitioning the task management of data acquisition and forwarding when a central gateway is considered.

VIII. Conclusion

In this paper, we have presented a novel approach to secure BMS lifecycle monitoring considering both on-premise and cloud environments. The proposed architecture has been developed in mind for the current and upcoming use cases concerning battery passports and regulations. The design allows for intermediate secure storage of BMS blocks on a local gateway device, before they are able to be further processed on the cloud and end systems. The BMS data is securely processed from the main BMS controller, over the internal network, to the cloud service and end system. A demonstrator has been successfully implemented to evaluate the design's performance in real-world environments. For future work, we see solutions such as OSCORE [28], aimed at constrained IoT devices, as a potential extension to the current security design on the local BMS network layer. Additionally, it is planned to cover modern solutions used for plug-and-charge services and create an adaptive layer with the current data structure design.

Acknowledgment

This paper is supported by the OPEVA project that has received funding within the Key Digital Technologies Joint Undertaking (KDT JU) from the European Union's Horizon Europe Programme and the National Authorities (France, Belgium, Czechia, Italy, Portugal, Turkey, Switzerland), under grant agreement 101097267. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or KDT JU. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] H. A. Gabbar *et al.*, "Review of Battery Management Systems (BMS) Development and Industrial Standards," *Technologies*, vol. 9, 2021.
[2] R. Xiong and W. Shen, *Advanced Battery Management Technologies for Electric Vehicles*. Wiley, 2019.
[3] W. Li *et al.*, "Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation," *Journal of Energy Storage*, vol. 30, p. 101557, 2020.
[4] "Electric Vehicle Battery Market - Global Forecast to 2028," tech. rep., Meticulous Research, Nov 2021.
[5] H. Eric Melin, "The lithium-ion battery end-of-life market – A baseline study," tech. rep., Global Battery Alliance, 2018.
[6] B. Jayaraman, "EVERLASTING D8.16 – White Paper 13," tech. rep., EU Project, 08 2020.
[7] L. C. Casals, B. Amante García, and C. Canal, "Second life batteries lifespan: Rest of useful life and environmental analysis," *Journal of Environmental Management*, vol. 232, pp. 354–363, 2019.
[8] L. Zhou *et al.*, "Massive battery pack data compression and reconstruction using a frequency division model in battery management systems," *Journal of Energy Storage*, vol. 28, p. 101252, 2020.
[9] M. S. Hossain Lipu *et al.*, "Smart Battery Management Technology in Electric Vehicle Applications: Analytical and Technical Assessment toward Emerging Future Directions," *Batteries*, vol. 8, no. 11, 2022.
[10] GBA, "Battery Passport." https://www.globalbattery.org/battery-passport/, 2023. Accessed: 07.03.2023.
[11] "Proposal for a regulation of the European Parliament and of the Council concerning batteries and waste batteries, repealing Directive 2006/66/EC and amending Regulation (EU) No 2019/1020." https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020PC0798, 2020.
[12] V. Halleux, "EU Legislation. New EU regulatory framework for batteries.," tech. rep., Members' Research Service, 2022.
[13] M.-K. Tran *et al.*, "Concept Review of a Cloud-Based Smart Battery Management System for Lithium-Ion Batteries: Feasibility, Logistics, and Functionality," *Batteries*, vol. 8, no. 2, 2022.
[14] T. Tanizawa *et al.*, "Cloud-connected Battery Management System Supporting e-Mobility," *Fujitsu Science Tech. Journal*, vol. 51, 2015.
[15] S. Yang *et al.*, "Implementation for a cloud battery management system based on the CHAIN framework," *Energy and AI*, vol. 5, 2021.
[16] A. Samanta and S. S. Williamson, "A Survey of Wireless Battery Management System: Topology, Emerging Trends, and Challenges," *Electronics*, vol. 10, no. 18, 2021.
[17] J. Wu, X. Liu, J. Meng, and M. Lin, "Cloud-to-edge based state of health estimation method for lithium-ion battery in distributed energy storage system," *Journal of Energy Storage*, vol. 41, p. 102974, 2021.
[18] K. Li *et al.*, "Battery life estimation based on cloud data for electric vehicles," *Journal of Power Sources*, vol. 468, p. 228192, 2020.
[19] J. Neubauer *et al.*, "A Second Life for Electric Vehicle Batteries: Answering Questions on Battery Degradation and Value," *SAE International Journal of Materials and Manufacturing*, vol. 8, 2015.
[20] W. Huang *et al.*, "Questions and Answers Relating to Lithium-Ion Battery Safety Issues," *Cell Reports Physical Science*, vol. 2, 2021.
[21] S. Kumbhar *et al.*, "Cybersecurity for Battery Management Systems in Cyber-Physical Environments," *IEEE ITEC*, pp. 761–766, 2018.
[22] A. Reindl *et al.*, "Scalable, Decentralized Battery Management System Based on Self-organizing Nodes," in *ARCS*, pp. 171–184, 2020.
[23] E. Wood *et al.*, "Investigation of battery end-of-life conditions for plug-in hybrid electric vehicles," *Journal of Power Sources*, vol. 196, 2011.
[24] R. Faria *et al.*, "Primary and secondary use of electric mobility batteries from a life cycle perspective," *Journal of Power Sources*, vol. 262, 2014.
[25] A. Ali *et al.*, "BCALS: Blockchain-Based Secure Log Management System for Cloud Computing," *Trans. E. Telecomm. Tech.*, vol. 33, 2022.
[26] F. Basic *et al.*, "Trust your BMS: Designing a Lightweight Authentication Architecture for Industrial Networks," in *IEEE ICIT*, 2022.
[27] F. Basic *et al.*, "Establishing Dynamic Secure Sessions for ECQV Implicit Certificates in Embedded Systems," in *DATE*, pp. 1–6, 2023.
[28] M. Gunnarsson *et al.*, "Evaluating the performance of the OSCORE security protocol in constrained IoT environments," *Internet of Things*, vol. 13, p. 100333, 2021.
[29] ISO, "ISO 15118-20:2022 Road vehicles — Vehicle to grid communication interface - Part 20," 2022.

[30] A. Fuchs *et al.*, "HIP-20: Integration of Vehicle-HSM-Generated Credentials into Plug-and-Charge Infrastructure," in *4th ACM CSCS*, 2020.
[31] B. Hale and C. Komlo, "On End-to-End Encryption." Cryptology ePrint Archive, Paper 2022/449, 2022.
[32] "BearSSL." https://bearssl.org/, 2018. Accessed: 25.03.2023.
[33] ISO, "ISO 15765-2:2016 Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2," 2016.
[34] H. Mansor *et al.*, "Log Your Car: The Non-invasive Vehicle Forensics," in *IEEE Trustcom/BigDataSE/ISPA*, pp. 974–982, 2016.
[35] "EVITA. E-safety vehicle intrusion protected applications." https://evita-project.org/, 2011. Accessed: 08.03.2023.