

BC-SBOM: Blockchain-based SBOM Management System

Ahyun Song*, Euseong Seo*, Heeyoul Kim**

* Department of Computer Science and Engineering, Sungkyunkwan University, Republic of Korea

**Division of Computer Science and Engineering, Kyonggi University, Republic of Korea

fialle@g.skku.edu, euseong@skku.edu, heeyoul.kim@kgu.ac.kr

Abstract—This paper presents BC-SBOM, a novel blockchain-based system designed to enhance the management of Software Bills of Materials (SBOMs). By leveraging blockchain technology, BC-SBOM ensures secure storage and sharing of SBOMs, while providing a comprehensive global view of dependencies among software components. The system also supports rapid propagation of alerts for newly discovered vulnerabilities, thereby increasing responsiveness to potential threats. Offering superior reliability, transparency, and availability compared to traditional SBOM tools, BC-SBOM aims to significantly improve the management of complex software systems and contribute to the advancement of software security practices.

Keywords— Blockchain, SBOM, Decentralization, Dependency, Vulnerability

I. INTRODUCTION

As seen in the case of the vulnerability in the Log4j library in 2021 [1], the complexity of software has increased, highlighting the importance of transparency and effective management of software components. A Software Bill of Materials (SBOM) provides a comprehensive list of all components included in a software product or service, offering a foundation for accurately identifying the source, version, and license information of software [2]. This ensures the transparency of software development, clarifying the origin and attributes of components, thereby enabling reliable software operations.

The necessity of SBOMs primarily manifests in enhancing security, complying with regulations, and managing licenses. Particularly in terms of security, SBOMs play a critical role in quickly identifying potential vulnerabilities in components and minimizing risks through the application of appropriate security patches. Moreover, in various industries, SBOMs are recognized as essential tools for meeting regulatory standards that demand traceability and transparency of software. Additionally, the accurate management of license information through SBOMs plays a crucial role in reducing legal risks and protecting intellectual property.

Key functionalities of an SBOM include the automated identification and listing of components, tracking vulnerabilities and monitoring update statuses, and managing software changes. These features enable swift and precise

responses to various issues that may arise during software development and maintenance, thereby enhancing the overall quality and stability of software. Furthermore, the utilization of SBOMs offers numerous advantages such as improved security and reliability, efficient risk management, and ease of regulatory compliance. Detailed information provided about each component helps effectively manage security risks, quickly resolve vulnerabilities and licensing issues, thus enhancing the operational efficiency of software.

This paper proposes a decentralized SBOM management system that involves software developers, operators, and users. Operating based on blockchain technology, this system offers higher reliability and availability compared to existing SBOM systems.

The rest of this paper is organized as follows. Section II provides an overview of the limitations of existing SBOM systems and introduces blockchain technology. Section III describes the core concepts, architecture, and primary operations of the proposed system. Section IV discusses considerations for implementing the system. Finally, Section V concludes the paper.

II. SBOM AND BLOCKCHAIN

SBOMs are typically generated using standards such as SPDX [3] and CycloneDX [4]. Tools like Dependency-Track [5] and FOSSA [6] leverage SBOMs to provide license compliance and vulnerability management. However, despite their crucial role in enhancing the transparency of software components, these tools possess structural limitations due to their standalone or centralized operations. Firstly, most SBOM systems rely on centralized servers or databases, creating single points of failure. Centralized systems facilitate data alteration or manipulation, thus compromising the integrity of SBOM data and potentially leading to incorrect decision-making. Secondly, the complexity of software updates and maintenance emerges. Centralized systems face constraints in reflecting continuous changes and updates in real time. Moreover, efficiently disseminating these changes to all relevant stakeholders presents further difficulties. Thirdly, a lack of transparency and traceability is evident. Centralized systems often restrict data accessibility and management

transparency, complicating the process for users or auditing bodies to access or verify the necessary information.

Blockchain technology, characterized by its decentralized ledger that stores transaction records in linked blocks containing transaction data, timestamps, and previous block hashes, complicates data tampering [7]. This ensures all participants verify network data, thus providing high levels of transparency and security. Additionally, the absence of centralized management enhances data integrity and stability.

Smart contracts stored on the blockchain autonomously execute contracts when conditions are met, ensuring reliability and transparency. Transactions can occur without intermediaries, reducing costs and time while minimizing the risk of errors. This technology is utilized across various sectors such as finance, real estate, and insurance to maximize operational efficiency and automation.

Blockchain's increasing utilization in supply chain management to enhance transparency, integrity, and reliability is noteworthy [8]. Research indicates blockchain's capability to facilitate real-time data tracking across all supply chain stages, offering visibility from product origin to consumer. For instance, in agriculture, it ensures food safety, and in manufacturing, it helps ascertain the origin of parts to prevent counterfeiting. In logistics, it improves transportation efficiency, and in pharmaceuticals, it verifies product authenticity to enhance patient safety. These studies demonstrate blockchain's potential to revolutionize traditional supply chain management methods.

Decentralized Identifiers (DID) [9] allow users to manage their identity information autonomously without central authority intervention, enhancing personal data protection and ensuring high reliability and transparency. Blockchain-based DID securely stores identity information on the blockchain, allowing users direct access and management, thus ensuring the integrity of identity information.

Recent initiatives have sought to integrate blockchain technology with SBOM, notably in the paper [10], which proposes a blockchain-based SBOM sharing architecture. This research introduces a new method using verifiable credentials to enable selective disclosure, enhancing software supply chain transparency and integrity. However, these efforts are still limited to using blockchain as a repository for SBOM documents, and they face challenges in understanding and sharing the global view of dependency relationships and software supply statuses.

III. BC-SBOM SYSTEM

We propose BC-SBOM, a highly trustworthy and automated SBOM management system that leverages the characteristics of blockchain technology. Figure 1 illustrates the architecture of the BC-SBOM system, which is composed of four distinct layers:

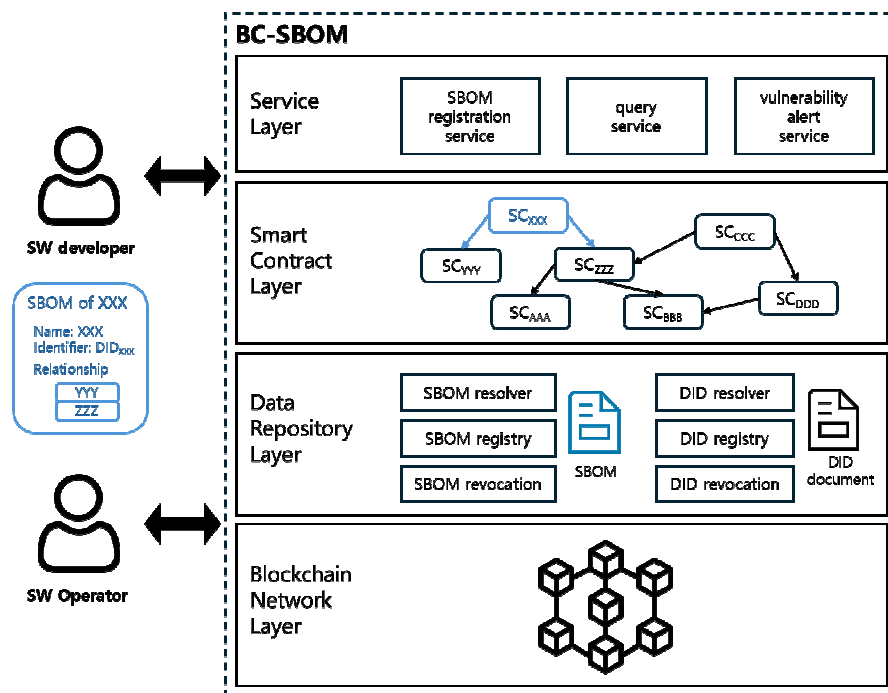
- **Blockchain Network Layer:** Ensures the integrity and transparency of transactions and blocks in a decentralized manner, providing high trustworthiness.
- **Data Repository Layer:** A blockchain-based repository

Figure 1 The Layer Architecture of BC-SBOM System

that stores and shares SBOM and DID documents, making them accessible.

- **Smart Contract Layer:** Manages dependencies between software components using smart contracts.
- **Service Layer:** Provides services for registering SBOMs, querying dependency information, and disseminating vulnerability alerts.

The proposed system not only facilitates the easy and secure sharing of registered SBOMs among participants but



also offers a global view that represents dependencies among all software components in the form of a Directed Acyclic Graph (DAG), reflecting changes in real-time. This view is constructed using blockchain's smart contracts, allowing participants to utilize it without concerns of tampering.

Using the proposed system, participants can easily share SBOMs and transparently obtain origin and related information about software components. Additionally, they can efficiently query comprehensive dependency information for specific components. Furthermore, when new vulnerabilities are detected, participants using the affected software can rapidly receive propagated alerts. The operational processes of these three services are described below.

A. SBOM registration

The proposed system registers SBOMs to construct a global view that displays dependencies and related information across all software components. Based on the characteristics of blockchain, this dependency view ensures high trustworthiness, integrity, and transparency, and is shared with all participants, providing high availability. The process in Figure 2 by which this view is expanded with the registration of a new SBOM is as follows:

- 1) Software developers provide and request registration of an SBOM for a new software component, XXX, through the registration service. A DID, DID_{XXX}, is assigned as the identifier for XXX and included in the SBOM. It is assumed that XXX has dependencies on software components YYY and ZZZ.
- 2) Once validated, the submitted SBOM is stored via the SBOM registry. Subsequently, participants can securely retrieve the desired SBOM without the risk of tampering by malicious attackers, using the SBOM resolver.

- 3) In this system, a smart contract is created for each software component to automate the management of dependency relationships. The registration service generates a new smart contract, SC_{XXX}, for XXX, which records key information about the XXX component (such as origin, license, and version) obtained from the SBOM.
- 4) SC_{XXX} requests the DID documents for components YYY and ZZZ, which are dependencies for XXX, from the DID resolver.
- 5) The DID resolver provides the DID documents for YYY and ZZZ, which include the addresses of the smart contracts that represent instances of these components.
- 6) SC_{XXX} adds the addresses of YYY and ZZZ to its relationship mapping, as illustrated. Conceptually, this means that links are created from SC_{XXX} to SC_{YYY} and SC_{ZZZ}, representing the dependencies.
- 7) Finally, the registration service creates the DID document for XXX, including the address of SC_{XXX}, and registers it through the DID registry.

B. SW dependency query

One of the key functionalities of using an SBOM is identifying dependencies between software components, which is particularly crucial in large-scale software systems. Tools like Dependency-Track offer this functionality; however, for accurate dependency analysis, it is essential to maintain reliable and up-to-date SBOMs for all relevant components, and the analysis performed for each query can be complex and resource-intensive. The proposed system, built on blockchain, leverages a shared global dependency view to support efficient querying, as follows:

- 1) Participants request a dependency analysis for XXX from the query service.

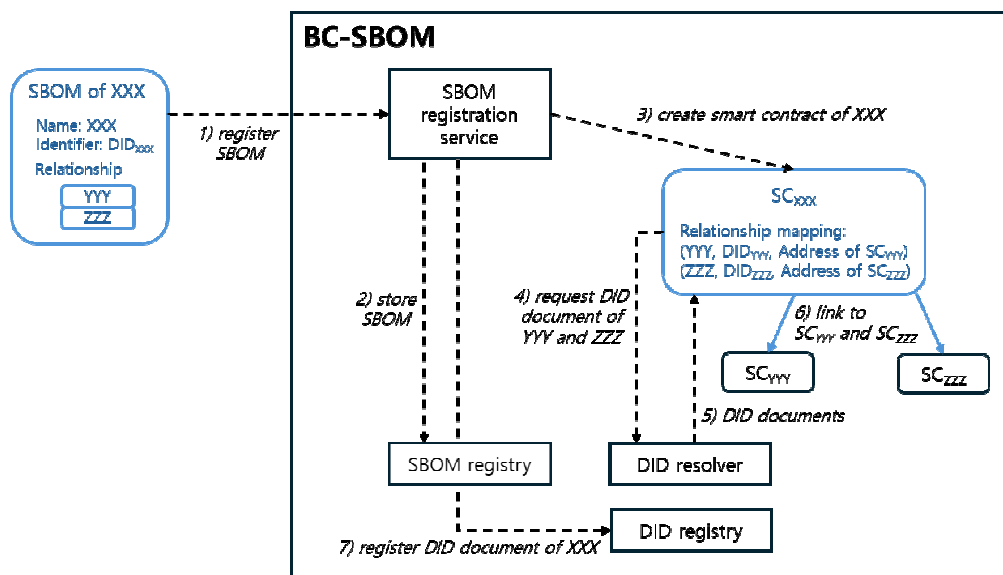


Figure 2 Process Flow of SBOM Registration

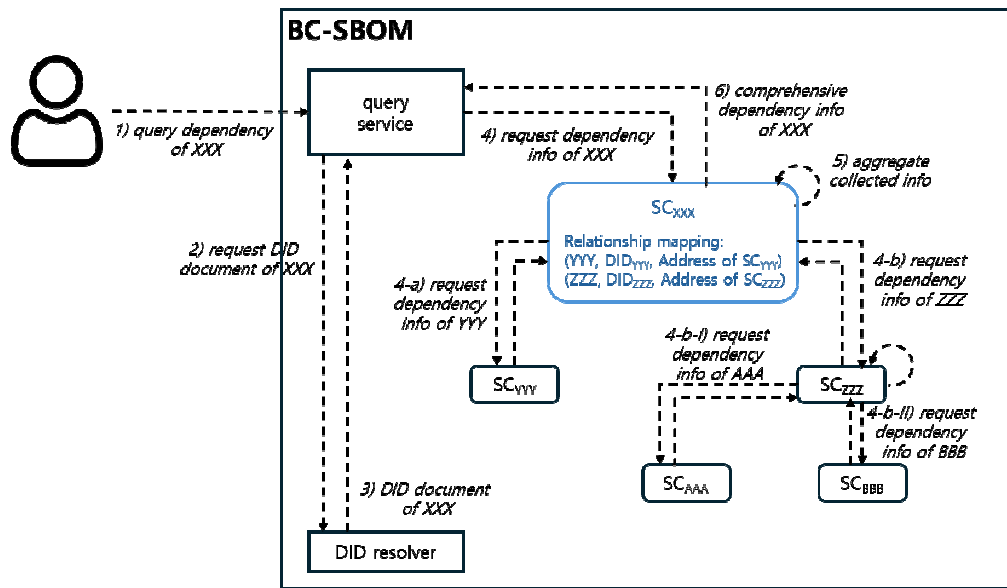


Figure 3 Process Flow of Dependency Query

- 2) The query service requests the DID document of XXX from the DID resolver to ascertain the address of SC_{XXX}.
- 3) The DID resolver returns the DID document of XXX, stored on the blockchain.
- 4) The query service requests dependency information for XXX from SC_{XXX}. SC_{XXX} retrieves the addresses of the smart contracts for components YYY and ZZZ, SC_{YYY} and SC_{ZZZ}, from its relationship mapping. SC_{XXX} then requests and receives dependency information from SC_{YYY} and SC_{ZZZ}. This querying process is recursive; for example, upon receiving a request from SC_{XXX}, SC_{ZZZ} requests and compiles dependency information from its direct dependency components AAA and BBB (illustrated as 4-b-I and 4-b-II in Figure 3), aggregates this data, and provides its complete dependency information back to SC_{XXX}.
- 5) SC_{XXX} aggregates the dependency information received from SC_{YYY} and SC_{ZZZ} to generate its own comprehensive dependency information. This information encompasses not only the direct dependencies of XXX but also all components in a transitive dependency relationship.
- 6) The comprehensive dependency information for XXX is provided back to the participant who initiated the request through the query service. This information is automatically generated by the blockchain's smart contracts, ensuring tamper-proof data and high trustworthiness.

blockchain's event handling mechanisms to automatically notify participants using the affected components regarding the new vulnerability. It is assumed that software developers, operators, and users are subscribed to receive events concerning the software they use. The process of propagating a vulnerability alert for component AAA in Figure 4 unfolds as follows:

- 1) When a security expert discovers a new vulnerability in AAA, he reports this vulnerability to the vulnerability alert service.
- 2) The alert service requests the DID document of AAA from the DID resolver to ascertain the address of SC_{AAA}.
- 3) The DID resolver returns the DID document of AAA, which is stored on the blockchain.
- 4) The vulnerability alert service notifies SC_{AAA} of the newly discovered vulnerability and triggers the creation and propagation of an alert.
- 5) SC_{AAA} emits an alert event containing vulnerability information. Components with a direct dependency on AAA listen to this event and then generate and emit their own alert events. This process continues recursively, propagating the alerts to all components that have transitive dependencies with AAA. For example, upon receiving the alert event on AAA, SC_{ZZZ} generates a new alert event on ZZZ, which is then conveyed to SC_{XXX}.
- 6) Participants who subscribed to receive events about XXX receive the alert event on XXX, becoming aware of the vulnerability in AAA that they depend on, and can promptly respond.

C. Vulnerability alert propagation

Another important functionality facilitated by SBOMs is the ability to quickly identify related software components when new vulnerabilities are discovered, thereby enabling prompt responses. The proposed system leverages the

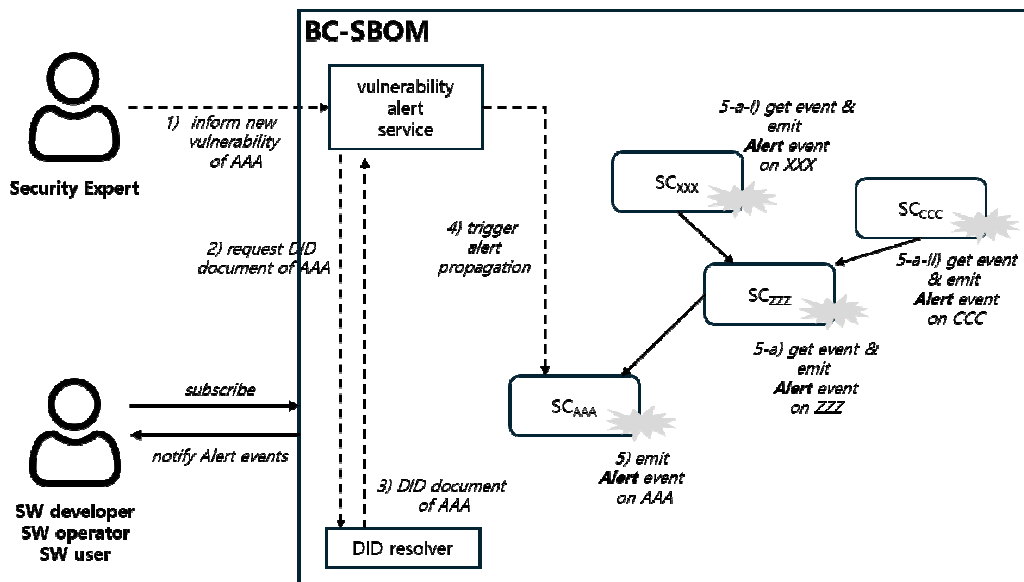


Figure 4 Process Flow of Alert Propagation

IV. DISCUSSION

In this paper, we have presented the core concepts and system architecture of the BC-SBOM system, which manages SBOMs based on blockchain technology, and described the operational processes of three services. For future work, we plan to implement and evaluate a prototype of the BC-SBOM. The following considerations are being made for effective implementation.

Firstly, the blockchain network used in BC-SBOM will be a hybrid of the public blockchain Ethereum and the consortium blockchain Hyperledger Indy. The management of DID documents and participant authentication will be conducted on Hyperledger Indy to enhance security. Dependency information will be managed using smart contracts on Ethereum, allowing anyone to easily query the dependency information of the software they are interested in.

Secondly, the cost of using the blockchain network needs to be considered. Particularly with Ethereum, where fees must be paid for each transaction, it is necessary to prevent excessive costs due to high transaction volumes. Registering new SBOMs and adding dependencies in BC-SBOM involve changing the state on Ethereum, thus generating transactions. However, for frequent dependency queries that do not alter the state information on Ethereum, a *call* is used instead of a *transaction*. This method does not incur fees unlike transactions, thereby minimizing costs.

V. CONCLUSION

We have proposed the BC-SBOM system, which effectively manages SBOMs based on blockchain technology. This system offers secure storage and sharing of SBOMs, provides a global view representing dependencies among software components, and facilitates rapid propagation of

alerts for new vulnerabilities. BC-SBOM possesses advantages over traditional SBOM tools in terms of trustworthiness, transparency, and availability. It is anticipated that the implementation of this system will assist in managing complex software systems and contribute to enhanced security.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(No. 2020R1A6A1A03040583)

REFERENCES

- [1] R. Hiesgen, M. Nawrocki, T. C. Schmidt and M. Wählisch, "The Log4j Incident: A Comprehensive Measurement Study of a Critical Vulnerability," in *IEEE Transactions on Network and Service Management*, 2024, doi: 10.1109/TNSM.2024.3440188.
- [2] B. Xia, T. Bi, Z. Xing, Q. Lu and L. Zhu, "An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead," 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, 2023, pp. 2630-2642, doi: 10.1109/ICSE48619.2023.00219.
- [3] Software Package Data Exchange (SPDX), [online] Available: <https://spdx.dev/about>.
- [4] OWASP CycloneDX Software Bill of Materials (SBOM) Standard, [online] Available: <https://cyclonedx.org/>.
- [5] Dependency-Track official site, [online] Available: <https://dependencytrack.org/>.
- [6] FOSSA: Comprehensive Open Source Security and SBOM management, [online] Available: <https://fossa.com/>.
- [7] M. Tabatabaei, R. Vitenberg, and N. Veeraragavan, "Understanding blockchain: Definitions, architecture, design, and system comparison," *Computer Science Review*, Volume 50, 2023, ISSN 1574-0137, <https://doi.org/10.1016/j.cosrev.2023.100575>.
- [8] P. Ducezyk, J. K. Dunston and G. V. Crosby, "Blockchain Technology for Global Supply Chain Management: A Survey of Applications, Challenges, Opportunities and Implications," in *IEEE Access*, vol. 12, pp. 70065-70088, 2024, doi: 10.1109/ACCESS.2024.3399759.

- [9] C. Brunner, U. Gellersdörfer, F. Knirsch, D. Engel, and F. Matthes, "DID and VC:Untangling Decentralized Identifiers and Verifiable Credentials for the Web of Trust," In Proceedings of the 2020 3rd International Conference on Blockchain Technology and Applications (ICBTA '20), pp. 61–66, <https://doi.org/10.1145/3446983.3446992>
- [10] B. Xia, D. Zhang, Y. Liu, Q. Lu, Z. Xing, and L. Zhu, "Trust in Software Supply Chains: Blockchain-Enabled SBOM and the AIBOM Future," In Proceedings of the 2024 ACM/IEEE 4th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS) and 2024 IEEE/ACM Second International Workshop on Software Vulnerability (EnCyCriS/SVM '24), pp. 12–19, <https://doi.org/10.1145/3643662.3643957>



Ahyun Song received the M.S. degree in Computer Science from KAIST, Korea, in 2005. From 2005 to 2015 she was a manager at Korea Financial Telecommunications & Clearings Institute. Since 2015 she has been a senior manager at Financial Security Institute in Korea. She is pursuing the Ph.D. degree in Computer Science and Engineering at Sungkyunkwan University. Her major research interests include security, blockchain, and DeFi.



Euseong Seo received his B.S., M.S., and Ph.D. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 2000, 2002, and 2007, respectively. He is currently a professor in Department of Computer Science and Engineering at Sungkyunkwan University, Rep. of Korea. Before joining Sungkyunkwan University in 2012, he had been an assistant professor at Ulsan National Institute of Science and Technology (UNIST), Rep. of Korea from 2009 to 2012, and a research associate at the Pennsylvania State University from 2007 to 2009. His research interests are system software, embedded systems, and cloud computing.



Heeyoul Kim received the B.E. degree in Computer Science from KAIST, Korea, in 2000, the M.S. degree in Computer Science from KAIST in 2002, and the Ph.D. degree in computer science from KAIST in 2007. From 2007 to 2008, with the Samsung Electronics as a senior engineer. Since 2009 he has been a faculty member of Department of Computer Science at Kyonggi University. His major research interests include cryptography, security and blockchain.