A Zero-Knowledge Framework for Confidential and Verifiable SBOM Validation

Jiarou Deng
Johns Hopkins University
Baltimore, MD, USA
jdeng33@jh.edu

Yang Yang
Johns Hopkins University
Baltimore, MD, USA
yyang296@.jh.edu

Michael Rushanan
Harbor Labs
Pikesville, MD, USA
mike@harborlabs.com

***Abstract*—Recent regulatory initiatives require medical device manufacturers to produce and maintain Software Bills of Materials (SBOMs) to enhance visibility into the software supply chain and associated vulnerability risks. However, public release of SBOMs introduces a new attack surface by exposing component-level information that adversaries can exploit. Our prior work demonstrates that even de-identified SBOMs can be paired with public vulnerability databases and large language models (LLMs) to generate functional attack blueprints, reducing adversarial effort and successfully exploiting 77.8% of known vulnerabilities in a controlled environment. This transparency v. exposure dilemma motivates the need for privacy-preserving validation mechanisms. We propose a Zero-Knowledge Proof (ZKP) framework that enables stakeholders to verify SBOM properties, such as the absence of critical vulnerabilities, without disclosing sensitive supply chain details.**

***Index Terms*—SBOM, computer security, medical devices, supply chain, regulatory compliance, zero knowledge, AI, LLM**

I. Introduction

Software supply chain attacks, such as those involving QNX [1] and Log4j [2], have underscored the need to improve visibility into software dependencies, particularly in critical infrastructure and life-critical technologies. SBOMs have emerged as a regulatory and technical response, with the U.S. Food and Drug Administration (FDA) now requiring SBOMs in premarket submissions for medical devices. These inventories help medical device manufacturers (MDMs) and regulators identify vulnerabilities across complex ecosystems of open-source and third-party software. SBOMs also provide a cybersecurity signal or input that can be monitored for marketed products to identify potential vulnerabilities throughout the product's lifecycle.

Regulatory guidance further intends for MDMs to make the SBOM available to device users, including hospital delivery organizations (HDOs), medical professionals, and patients. This increased transparency comes with *risk*—an attacker could use the information disclosed in the SBOM to exploit the associated device.

To convey this risk in the real world, we implemented a case study that demonstrates that even minimally detailed, real-world SBOMs can act as attack blueprints [3]. By combining SBOM disclosures with vulnerability databases and prompting LLMs, we showed that adversaries can semi-automate exploit generation and successfully validate 77.8% of the nine tested vulnerabilities. The attack cost was negligible, requiring just minutes of analyst time and costing under $0.20 in compute resources per CVE. We concluded that public access to SBOMs can significantly enhance adversarial capabilities, enabling faster and lower-cost exploit development.

In this paper, we present our ongoing work to address the transparency v. exposure dilemma using a zero-knowledge proof (ZKP) framework for validating SBOMs. We describe both the theoretical foundations and practical design details of our framework, and outline the open challenges that are currently under active investigation.

II. ZKP for SBOM Validation

We propose a ZKP framework as a cryptographic mechanism to address SBOM transparency. Our framework enables MDMs to commit their SBOM to our vulnerability dataset, then produce a ZKP attesting that the SBOM satisfies specific security policies (e.g., "no critical CVEs present"). Regulators or customers can then cryptographically verify this proof without needing to access the original or raw SBOM, which contains sensitive information such as software names, ensuring compliance while protecting sensitive details.

This design directly addresses three interrelated challenges:

- Tamper-evidence—SBOMs are bound by cryptographic primitives, such as Merkle tree roots, making any modification detectable at rest or in transit. Succinct zero-knowledge proof systems (e.g., Groth16) allow attestations to be tied to these commitments, ensuring proofs are anchored to authentic SBOM data.
- Confidentiality—Sensitive component details (e.g., software names and versions) are kept confidential. Zero-knowledge techniques enable participants to demonstrate the security or compliance properties of the SBOM without disclosing the raw contents.
- Validation—Regulators and auditors can verify that an SBOM satisfies correctness or compliance rules (e.g., absence of high-severity CVEs) solely from the proof and commitment.

A. ZKP Framework Design

We are exploring two potential ZKP approaches for SBOM validation. We outline both below.

1) TreeGap: TreeGap is a Merkle-tree-based algorithm that provides non-membership proofs by showing that a queried element lies within a gap—the interval between two consecutive neighbors in a sorted, committed set.

We build a Merkle tree on a snapshot of vulnerable components (i.e., the latest list of CVEs from the NVD). Each SBOM component is checked against the tree; if it is not in the set, we prove it as a non-member using a TreeGap witness; otherwise, we prove it as a member.

The gap in our implementation is defined by the two closest neighbors in the sorted vulnerability set. For membership proofs, we compute the Merkle path from the queried component's leaf node to the published root, demonstrating inclusion. For non-membership proofs, we provide valid Merkle paths for the two neighboring entries, showing that the queried component lies strictly between them and therefore cannot be in the vulnerability set.

2) RSA accumulator: An RSA accumulator is a cryptographic primitive that commits to a set of elements using modular exponentiation under an RSA modulus, enabling succinct proofs of membership (or non-membership) for individual elements without revealing the entire set.

We utilized the RSA accumulator implementation provided in Xin et al. [4]. In particular, we constructed an RSA accumulator over SBOM entries by mapping each identifier to a prime representative using a hash-to-prime function. These primes are sequentially exponentiated under an RSA modulus $N = p \times q$ to obtain a single accumulator value that succinctly commits to the entire dataset.

For membership proofs, let $S = \{e_1, e_2, \ldots, e_n\}$ denote a finite set of SBOM entries, where each e_i is either a prime number or the output of a hash-to-prime function. The accumulator value for the set S is defined as

$$\mathrm{acc}(S) = g^{\prod_{i=1}^{n} e_i} \bmod N,$$

where g is a generator modulo N.

To prove that a particular SBOM entry $e' \in S$ is included in the accumulator, the prover computes the witness

$$\pi = g^{\prod_{e_i \in S \setminus \{e'\}} e_i} \bmod N.$$

Verification is performed by checking the equation

$$\pi^{e'} \stackrel{?}{=} \mathrm{acc}(S) \pmod{N}.$$

If the equality holds, then e' is confirmed to be a member of the set S.

B. Experimental Setup

We prototyped our circuits using the open-source ZKP toolchains (Circom [5] and gnark [6]) with Poseidon hashing [7] and Merkle trees. SBOMs were generated in CycloneDX format, and vulnerability datasets constructed over the NVD published vulnerability database were committed as daily snapshots for membership and non-membership proofs. Scripts are adapted from public repositories circomlib [8], rsa_accumulator [9], and gnark examples.

III. Design Challenges

While we have limited our design to two candidate implementations, both with their own pros and cons, we have also faced several challenges, of which we enumerate below.

- Input normalization. SBOMs feature inconsistent naming conventions and version formats; scopes (e.g., $< 2.1.3$) and pre-release versions complicate deterministic matching. We apply minimal normalization (basic parsing and CPE mapping) and treat ambiguous cases as out-of-scope or conservatively flagged. Edge cases are not fully handled in the current pipeline.
- Scalability and efficiency—ZKP verification must scale to large SBOMs with thousands of components without prohibitive overhead. We employ batch proofs, Merkle forests, RSA accumulators, and compact proof systems, such as Groth 16, in circuits to ensure efficiency.
- Updatability and Maintenance—Vulnerability data updates hourly. We enable verifiers to precisely determine the "point in time" when claims become effective by binding proofs to snapshot dates and supporting reissuance based on incremental snapshot roots.
- Adoption and trust models—Defining who generates, verifies, and trusts ZKP attestations remains an organizational and governance challenge.

By addressing these challenges, our work seeks to establish a practical ZKP-based attestation framework that preserves the regulatory and security value of SBOMs while significantly reducing exposure risks.

IV. Future Work

We are working to solve the challenges enumerated in this paper and present empirical findings in the next iteration.

References

[1] Cybersecurity and Infrastructure Security Agency. (2021) Top routinely exploited vulnerabilities. CISA Cybersecurity Advisory AA21-229A. [Online]. Available: https://www.cisa.gov/news-events/cybersecurity-advisories/aa21-229a

[2] ——. (2021) Apache releases log4j version 2.15.0 to address critical rce vulnerability under exploitation. CISA Alert. [Online]. Available: https://www.cisa.gov/news-events/alerts/2021/12/10/apache-releases-log4j-version-2150-address-critical-rce-vulnerability-under-exploitation

[3] J. Deng, Y. Yang, and M. Rushanan, "Sbom transparency v. exposure dilemma: A case study on adversarial access to public sboms in healthcare," 2025, manuscript in submission to HealthSec.

[4] J. Xin, A. Haghighi, X. Tian, and D. Papadopoulos, "Notus: Dynamic proofs of liabilities from zero-knowledge RSA accumulators," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 1453–1470. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/xin

[5] Iden3, "Circom: Circuit compiler for zero-knowledge proofs," 2020. [Online]. Available: https://github.com/iden3/circom

[6] Consensys. (2025) gnark: a zksnark library in go. Accessed: 2025-09-15. [Online]. Available: https://github.com/Consensys/gnark

[7] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for Zero-Knowledge proof systems," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 519–535. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/grassi

[8] Iden3, "Circomlib: Library of circom templates," 2020. [Online]. Available: https://github.com/iden3/circomlib

[9] Notus Project. (2025) Rsa accumulator. Accessed: 2025-09-15. [Online]. Available: https://github.com/notus-project/rsa_accumulator