Article

Not peer-reviewed version

A Secure and Integrated Approach to Software Code and Docker Image Signing: Introducing the Hybrid Chain of Trust (HCoT) Algorithm

Jamshir Qureshi *

Posted Date: 10 February 2025

doi: 10.20944/preprints202502.0684.v1

Keywords: HCoT; Docker image signing; code signing; artifact signing; Software code

 doi:10.20944/preprints202502.0684.v1

Article

A Secure and Integrated Approach to Software Code and Docker Image Signing: Introducing the Hybrid Chain of Trust (HCoT) Algorithm

Jamshir Qureshi

Purdue University Global, West Lafayette, IN 47906, USA; jamshirqureshi@alumni.purdue.edu

Abstract: Securing the software supply chain is critical in an increasingly interconnected development environment, and vulnerabilities can have implications at a widespread level. Hybrid Chain of Trust (HCoT) algorithm, a proposed new solution, integrates cutting-edge technology to address such concerns. AI-facilitated code analysis in HCoT identifies a broader range of vulnerabilities compared to traditional methods. HCoT simplifies workflows through aggregation of code and image signatures under a single, unified process. HCoT also introduces transparency to the software delivery pipeline through having recordable, verifiable proofs of vulnerability and integrity of code. With these, security professionals and developers can make trust in software delivered to them a reality.

Keywords: HCoT; Docker image signing; code signing; artifact signing; Software code

Introduction

Software supply chain vulnerabilities are a ticking time bomb, and each explosion creates widespread disruption and data breaches. Recent examples with a victim base in the range of millions and an impact in terms of dollars in the range of billions paint a picture of the severity of such an attack.

Existing methodologies have a tendency to follow discrete workflows for code sign and Docker image sign and, in the process, introduce vulnerabilities at each stage in between. Common vulnerabilities like privilege escalation vulnerabilities in container images and insecure cryptographic algorithms for codesigning certificates add justification for security at all development stages.

The Hybrid Chain of Trust (HCoT) algorithm overcomes such obstacles with a single converged approach leveraging improvements in AI and blockchain technology. Smart analysis through AI-powered analysis, tamper-evidence through tamper-evident code bundle, and secure key management through decentralized key management work together to streamline workflows, enhance security, and enable transparency in the software delivery pipeline.

One-Sentence Summary: Hybrid Chain of Trust (HCoT) algorithm proposes a unified, AI and blockchain-based method for securing a software supply chain with efficient workflows, security, and transparency enhancement.

Existing Challenges: Securing a software supply chain is a complex problem with many obstacles. Traditional methods have two workflows for signing codes and Docker images, generating additional complexity and opportunity for human vulnerability [1]. Traditional methods also use a centralised Certificate Authority (CA) for issuing sign certificates, offering one single point of failure such as in NotPetya [2].

Other than these signing factors, several other factors contribute towards overall software supply chain risk:

1. Lack of Visibility:

 doi:10.20944/preprints202502.0684.v1

Following software parts and dependencies through a supply chain is challenging. Transparency in lack of origin and dependencies in software parts is not an easy one, and hence vulnerability detection and mitigation in third-party libraries or dependencies is a challenge [3].

1. Open-Source Vulnerabilities:

Open-source parts, with a function and efficiency, can introduce security vulnerabilities in case vulnerabilities in them go undetected and go unpatched in a timely manner. Maintaining an updated record of vulnerabilities and deploying timely patches is a must [4].

1. Limited Automation:

Security audits and updates in a manual manner can take a long and error-prone path. Security checking in the development life cycle through automation detects and corrects vulnerabilities early in life [5].

The drawback with these is that they require new and innovative approaches that make workflows easier, improve security, and improve transparency in the software delivery pipeline. An admission must, however, be made that HCoT can introduce its own constraint, such as increased computational cost in AI-powered analysis and the potential for added complexity in working with decentralized keys.

AI-powered analysis of code for vulnerability detection in the codebase.

Decentralized key management for elimination of single point of failure with CAs.

Tamper-evident bundles of code for integrity safeguarding of code and accompanying analysis reports.

By overcoming these challenges, HCoT presents a secure and efficient software supply chain security solution.

Proposed Solution: HCoT Algorithm

The HCoT Algorithm introduces a new model for securing the software supply chain. It builds a secure and verifiable trust chain for both Docker images and code, fixing key development issues for today's environment. HCoT Algorithm does this through a range of key capabilities, including AI analysis for code, secure bundle creation, and key management through a decentralized model. All these capabilities can be seen in the below diagram, a visualization of the HCoT workflow.

HCoT Workflow

 doi:10.20944/preprints202502.0684.v1

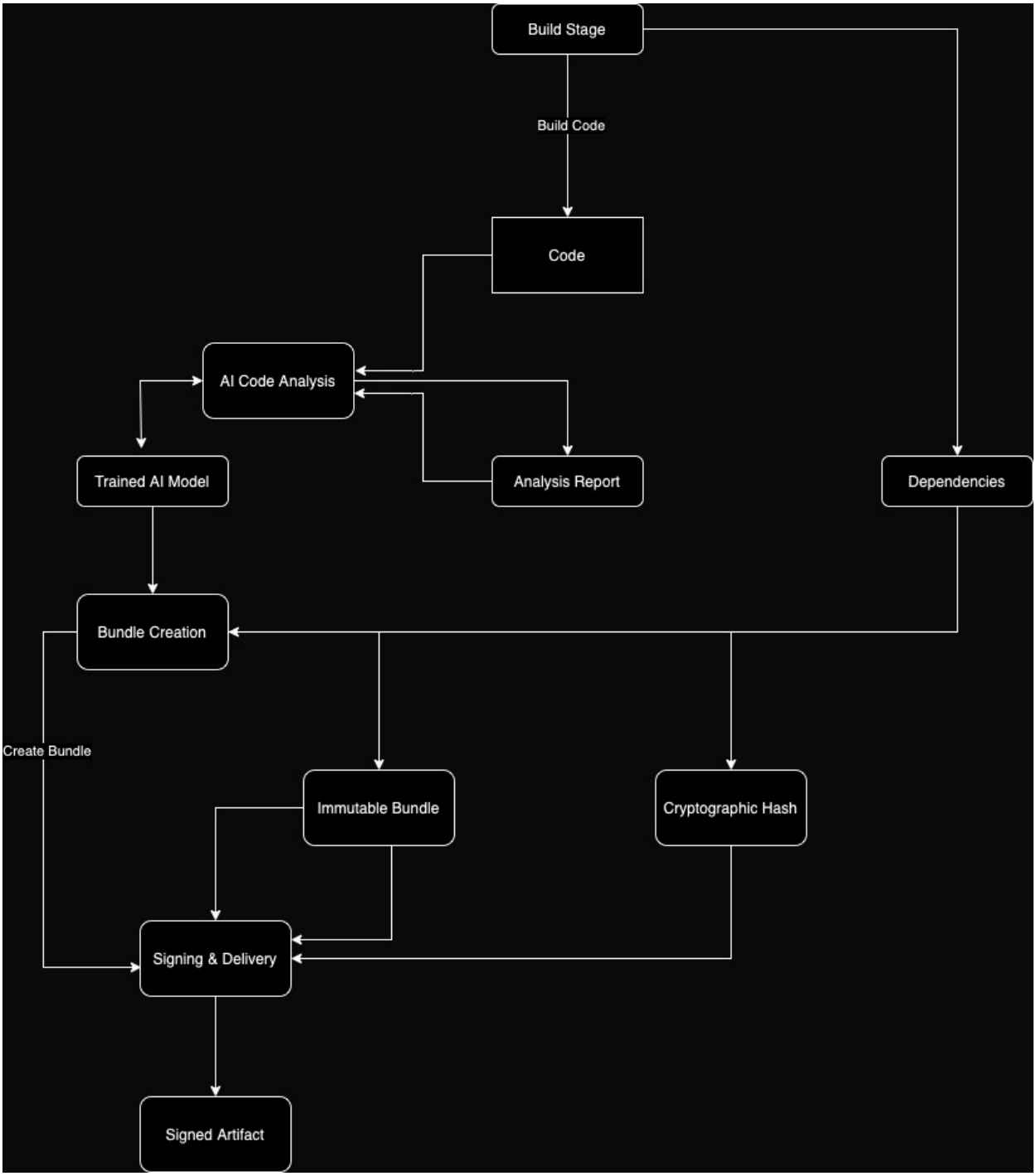

Figure. HCoT workflow.

A software developer creating a new software application. The developer codes and then pushes onto a versioning system. HCoT then takes over and performs the following:

1. ***Code Fetching:***
   HCoT retrieves the latest code from the version control system.

2. ***AI Analysis:***
   The code is subjected to comprehensive AI analysis to identify vulnerabilities and security weaknesses.

 doi:10.20944/preprints202502.0684.v1

3. ***Bundle Creation:***

Upon successful analysis, a secure bundle is created using containerization technologies, encapsulating the verified codebase.

4. ***Decentralized Key Management:***

Cryptographic keys are generated and securely distributed across a decentralized network for bundle encryption.

5. ***Secure Distribution:***

The signed and encrypted code bundle is made available for deployment in various environments.

Benefits of HCoT

The HCoT Algorithm brings many key advantages in securing software supply:

1. Increased Security:

HCoT significantly strengthens the overall security posture through its core functionalities:

AI-Powered Code Analysis:

This can identify a broader range of vulnerabilities compared to traditional signature-based methods, potentially reducing the likelihood of undetected security flaws. Research suggests that AI-powered code analysis can achieve high accuracy in vulnerability detection [6].

The HCoT Algorithm offers a multitude of benefits for developers. By leveraging AI-powered analysis, HCoT proactively identifies security risks early in the development process. This allows developers to address vulnerabilities before they become critical security breaches, saving time and resources. The AI analysis also improves code quality by detecting coding errors and inefficiencies, leading to more robust and secure software. This translates to a reduction in development time, as developers spend less time fixing vulnerabilities discovered later in the cycle. Overall, HCoT streamlines the development process by promoting secure coding practices and proactive risk mitigation.

HCoT can integrate with existing code scanning tools or leverage custom-built AI models trained on a vast dataset of vulnerabilities and secure coding practices.

Decentralized key management:

Eliminates the single point of failure for traditional Certificate Authorities (CAs). It is not possible for a compromised certificate of a CA to sign malicious software, but HCoT eliminates such an opportunity.

Decentralized key management forms a secure security mechanism for HCoT. By distributing its cryptographic keys in a peer-to-peer network, security overall drops to a significant level. With a system in distribution, it is incredibly challenging for hackers to penetrate and exploit, for hackers will have to penetrate several nodes at one go. HCoT is also more fault tolerant. In case one of a network's single node is compromised through an attack, supporting nodes can still maintain security for packages of codes. With such redundancy, security continues uninterrupted even in case a security attack occurs. Lastly, decentralized key management is highly scalable. As more and more users and codebases join in, the system can expand with ease to serve heightened demand, and HCoT becomes a future-proof device for an ever-changing software development environment.

The HCoT can use current decentralized key management tools such as Trezor or GPG to secure and manage its cryptographic keys to and from its peers.

Tamper-evident code bundles:

HCoT surpasses secure bundling by incorporating tamper-evident mechanisms to secure the code and the associated AI analysis report. This is very important because a compromised codebase

 doi:10.20944/preprints202502.0684.v1

or a manipulated report of the analysis will break the whole security framework. Here's how HCoT achieves a tamper-proof code bundle:

Examples include cryptographic hashing, such as SHA-256, which is used to ensure the integrity of the code and the AI analysis report. When either change, the signatures are invalidated, and tampering is easily detected.

HCoT employs a cryptographic hashing function, like SHA-256, on both the code and the AI analysis report to create a unique digital fingerprint. This fingerprint, also called a hash, is like a checksum in that it captures the exact content of the data.

The beauty in cryptographic hashes is that they are one-way. It's quite easy to generate a hash from any data, but highly computationally infeasible to recreate the original data from a given hash. Additionally, any small change in data will completely result in a different hash value.

The generated hashes of the code and the AI analysis report are securely stored within the code bundle itself. This way, they will travel with the code and the report and always be available for verification.

Here's how HCoT uses these tamper-evident mechanisms to ensure the integrity of the code bundle:

Upon receiving the code bundle, the recipient can recalculate the hash for the code and the analysis report using the same cryptographic hashing function (e.g., SHA-256).

The newly calculated hashes are then compared with the ones stored within the bundle.

If the calculated hashes match the stored ones, it confirms that the code and report haven't been tampered with during transit or storage. Any discrepancy between the values indicates a potential tampering attempt, alerting the recipient to a potential security breach.

HCoT's tamper-evident code bundles offer a trifecta of security benefits. Firstly, they foster trust by guaranteeing the authenticity of the code and the accompanying AI analysis report. Recipients can be confident that they haven't received tampered data, ensuring they're working with the genuine software and a reliable security assessment. Secondly, these bundles enable early breach detection. Any attempt to alter the code or report will be flagged, allowing for swift action to mitigate the potential security threat. Finally, tamper-evident bundles contribute to a more robust security posture throughout the software development lifecycle. By safeguarding the integrity of these critical elements, HCoT helps build a more secure foundation for software development.

HCoT's tamper-evident code bundles, powered by cryptographic hashing, play a vital role in safeguarding the integrity of the software supply chain. This ensures that developers and end-users can trust the code they deploy and rely on the validity of the AI analysis report for informed decision-making.

When a code bundle is delivered to the end user, the recipient can utilize the same cryptographic hashing function to re-compute the hashes for the code and the analysis report, for example, SHA-256.

The end user then proceeds to compare the newly computed hashes with the ones that are available within the bundle.

If the computed hash value is the same as the ones saved, then it means that the code and the report have not been modified after they were sent. Any differences in the values could indicate the received item was altered, which should set an alarm for a possible security breach.

Code bundles that are sensitive to tampering as produced by HCoT offer an opportunity for multiple security advantages. It fosters trust since it guarantees that the code and the AI analysis report were not altered. Recipients are certain that the data they have received is unmodified, which means that the software, and the security assessment have never been compromised. These bundles also allow for early breach detection. Any attempt to compromise the code or report will trigger an alert which can then decorrelate the security threat. Lastly, tamper-evident bundles allow me to have a stronger security stance during the entire software development process. By protecting these sensitive elements, HCoT fortifies the base of software development.

 doi:10.20944/preprints202502.0684.v1

2. Streamlined Workflows

The process should allow Docker images and code signing to be integrated and streamlined into one single process. The easier will the tool be as it functions with only one software package. In this way, the developers can sign both the code and the Docker image in their building pipeline, hence this reduces the workflow.

3. Enhanced Transparency

The non-reversible code package which gets AI-analyzed is a guarantee for reflection of the total state of the code and the possible problems. While different departments may be operating in silos, this transparent approach allows the development teams to keep track of changes, assess risks, and come to a conscious decision about the trustworthiness of the project.

4. Smaller Attack Surface

Potential attack vectors reduce because CA's are not the only single point of an attack on a piece of the infrastructure anymore. Malicious actors can always aim an attack on a CA server, yet due to the distribution of the HCoT infrastructure, it becomes much more secure.

An illustration of the advantages and accomplishments of a holistic, simple, and transparent HCoT method, as an example of possibilities, could prove attractive in the direction of software supply chain security.

The different advantages and the achievements that flow from an HCoT approach must be taken into account also because there are several possible limitations to contend with:

Increase in Computation Cost

AI-driven code analysis is extremely resource-heavy and overwhelm large codebases. This could have an associative build time effect or additional resource cost.

Decentralized Key Management is Complex

Some of the features of decentralized key management are interfacing compared to traditional centralized CA models. This intensity of complex integration requirements sanctions developers and security teams to access additional training and tooling.

Use cases that are scanned with HCoT should be subjected to the same limitations of consideration set forth here.

Future Work

The HCoT algorithm offers hope for the possibilities of securing the software supply chain; however, there are various aspects that keep returning for further attentiveness:

Real-time Threat Intelligence Integration

Research involving integrating real-time threat intelligence feeds into the actual AI analysis will hold credible very recent vulnerability detection capabilities. HCoT would, therefore, enable the suggestion of zero-day vulnerabilities not incorporated in traditional signature-based detection mechanisms. Another potential downside that may rear its head will be the latency involved in those cases where data are streaming from the real-time feed and final output processing, and the susceptibility of the intelligence sources to commercial bias.

Federated Learning for AI Model Training

Taking into the fold of federated learning methods, the ability to enable collaborative training of the AI model across the developer community offers a variety of options. This collaborative turn

 doi:10.20944/preprints202502.0684.v1

would bolster code analysis due to the richness of datasets and skills available within the community. Nevertheless, federated learning does pose issues around data privacy and security during the training process.

Reference Implementation and Intuitive Interfaces

Creating a reference implementation of the HCoT algorithm with explicit documentation and a user-friendly interface for developers will be crucial to adoption. This reference implementation will facilitate integration into an environment with HCoT to take advantage of its security features.

Conclusion

The HCoT algorithm is a revolutionary approach to stealing the software supply chain security space. The use of artificial intelligence (AI) blended with elements of blockchain technology enables HCoT to create a multidimensional platform to secure software development life cycle. Looking into these aspects brings about how HCoT will potentially impact practitioners in the field through its good for developers, security, and the future of software development.

Living the new Development Workflows

HCoT offers developers a very exciting opportunity to build sustainable workflows and secure them within the development lifecycle. The AI-driven analysis engine allows proactive identification of vulnerabilities from the very early stages of coding. Developers can now address vulnerabilities before they snowball into major breaches. Thus, they will make great time savings as they prevent vulnerabilities from coming back to them later in the software development life cycle. They will avoid the delays that are often involved in hunting down and foreseeing an old problem that repeats itself in the middle or advanced stages of development. This leads to an expedited release cycle and better efficiency overall.

Empowering Security Teams

Security teams are vital in safeguarding software from possible threats. HCoT equips them with various tools to contribute to this light. The enhanced vulnerability detection means that the HCoT allows the teams to proactively detect and remediate security risks early enough before exploitation by malicious actors. HCoT also encourages greater transparency within the entire software delivery pipeline. The ability to establish tamper-evident code bundles, combined with a secure key management practice, ensures that no code is altered or corrupted in its life cycle. This allows a security team to completely keep tabs over the entire development life cycle by affirming the integrity of whatever software had its path cleared. This aids in promoting trust and allows them to make informed decisions with respect to the deployment of software.

Creating the Future of Software Development

Strenuous research and development are still required for HCoT to enhance its functions and parameters of real-world implementation about software. Fresh times take cent-stages, thus calling for advanced security. HCoT can take forms to ensure security within self-dependable software, ultimately making its appeal in terms of software development and delivery. It will let an AI-driven analysis work with the management of keys on a decentralized platform for a much-enhanced, secured, trustworthy software ecosystem.

Beyond the Horizon

A fruitful future will consider possibilities outside its core functionalities for HCoT. Future evolutions in the domain could deal with the integration of HCoT with existing CI/CD pipelines extending solutions targeting better optimization of the development cycle from integration to

 doi:10.20944/preprints202502.0684.v1

automation of security checks. The decentralized key management will continue to blossom towards supplying secure collaboration among development teams and third-party vendors. In the ongoing horizon of HCoT research as it matures, it might very well turn into a foundation where developing software elite on security and trustworthiness will be its norm.

In general terms, the HCoT Algorithm projects a future vision by packing AI potential and bits of blockchain technology. Its approach is promising—securing the software supply chain through empowering both the development and security teams. Indeed, research and refinement of HCoT need to be exercised. Yet, this will prove to be a transformational force in all software development and delivery aspects of a digital future world being safe and trusted.

References

1. BlackBerry. (n.d.). Software supply chains: A major challenge for cybersecurity. Retrieved from https://www.blackberry.com/content/dam/bbcomv4/blackberry-com/en/campaigns/2022/na/Software-Supply-Chain-Security-Research-Report.pdf
2. Rodgers, J., & Winkler, J. (2017, June 28). NotPetya malware outbreak: Quick start guide. Secureworks. Retrieved from https://www.secureworks.com/resources/vd-pre-ransomware-attack-simulation
3. GitHub. (n.d.). Software supply chain best practices. Retrieved from https://docs.github.com/en/code-security/supply-chain-security
4. Synopsys. (n.d.). 2023 open-source security and risk analysis report. Retrieved from https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html
5. Chawla, N., & Krishna, S. (2022, June). A comprehensive survey on software supply chain security: Challenges and solutions. In 2022 11th International Conference on Cloud Computing and Big Data (CCBD) (pp. 127-134). IEEE.
6. Bissyandri, E., Feitelson, D., & Khurshid, S. (2019). Evaluating repair patches generated by deep learning models. In 2019 34th IEEE/ACM International Conference on Software Engineering (ICSE) (pp. 805-816). IEEE.