POSTER: Lightweight Code Assurance Proof for Wireless Software

Theo Gamboni-Diehl
University of Colorado
Colorado Springs, CO, USA
tgamboni@uccs.edu

Simeon Wuthier
University of Colorado
Colorado Springs, CO, USA
swuthier@uccs.edu

Jinoh Kim
Texas A&M University
Commerce, TX, USA
jinoh.kim@tamuc.edu

Jonghyun Kim
ETRI
Daejeon, South Korea
jhk@etri.re.kr

Sang-Yoon Chang
University of Colorado
Colorado Springs, CO, USA
schang2@uccs.edu

ABSTRACT

Software-defined radio (SDR) and the softwarization of the wireless and mobile systems enable intelligent processing and control in wireless networking. We design and build a lightweight code assurance proof scheme for wireless system software implementations. More specifically, our scheme assures that a wireless user/prover holds the correct software codes, e.g., the correct version, for its wireless networking implementations. In contrast to the previous research for code attestation in trusted computing, our scheme forgoes hardware-based security and real-time networking, thus substantially increasing the application feasibility. We further design our scheme to be efficient in computing by using a Merkle tree for the efficiency of the verification of the assurance proof. We implement our scheme for proof-of-concept on srsRAN (a popular open-source software for cellular technology) and conduct preliminary measurements to demonstrate the lightweight design. We envision our scheme to be orthogonal and supplementary to the previous trustworthy code attestation because it provides different properties (assurance vs. attestation) and because the lightweight aspect yields greater applicability and lower overheads in hardware and networking. Our scheme will therefore be appropriate for the wireless/mobile environment which uses broadcasting (where receiving/verifications occur more frequently than transmitting/generations) and whose devices are resource-constrained.

CCS CONCEPTS

• **Security and privacy → Information accountability and usage control**.

KEYWORDS

Software-Defined Radio, Software Assurance, Merkle Tree, Wireless Networking, Cellular Networking, srsRAN, Open RAN

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
WiSec '22, May 16–19, 2022, San Antonio, TX, USA.
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9216-7/22/05.
https://doi.org/10.1145/35076571.3529653

ACM Reference Format:
Theo Gamboni-Diehl, Simeon Wuthier, Jinoh Kim, Jonghyun Kim, and Sang-Yoon Chang. 2022. POSTER: Lightweight Code Assurance Proof for Wireless Software. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '22), May 16–19, 2022, San Antonio, TX, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/35076571.3529653

1 INTRODUCTION AND RELATED WORK

Software-defined radio (SDR) enables innovations and facilitates advancements in wireless networking systems. The softwarization of the radio control enables greater R&D by reducing the learning curve for researchers entering the field and providing easier distributions and dissemination of the research product. Highlighting the innovations in wireless systems include using radio-based metrics for information/data processing and machine learning (e.g., [2, 5, 12]), adaptive spectrum access (e.g., [3–5, 8, 12, 13]), and intelligent security control to protect the wireless link (e.g., [3, 4, 13]). The softwarization of RF technology also facilitates transparency and inter-operability across the wireless system vendors and implementations, e.g., Open RAN.

In this paper, we design and build an assurance proof scheme for the software code in wireless systems. Our scheme generates a proof by taking the code files (and the user ID) as inputs and outputs a proof. Our scheme prevents an attacker lacking the code to generate the correct proof and an attacker from reusing other users' proofs. We also prioritize the verification efficiency and make it lightweight in verification by using a Merkle tree for the proof generation and verification. The *root* or the Merkle root (which is the result of the final hash computation of the Merkle tree) is the proof that can be presented to other wireless users for the software code assurance.

Related to our research is previous work in trusted computing which studied the code attestation using hardware security (e.g., trusted platform module and Intel SGX) and real-time protocols for providing fresh evidence (e.g., challenge and response), e.g., [1, 6, 10, 11]. In addition to having a sharper focus on wireless system software, we distinguish our approach from previous research by making it lightweight in requirements (forgoing hardware-based trusted computing) and in networking (no back-and-forth real-time networking). In addition, we design our scheme to be lightweight in computing by using a Merkle tree construction to reduce the number of computations. However, our scheme for code *assurance*

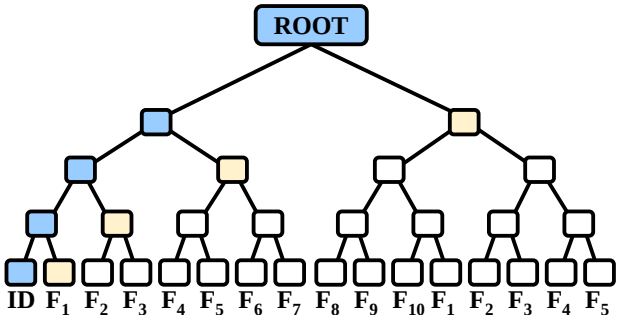


Figure 1: The binary Merkle tree structure of the proof of the code files, F_1 to F_{10}.

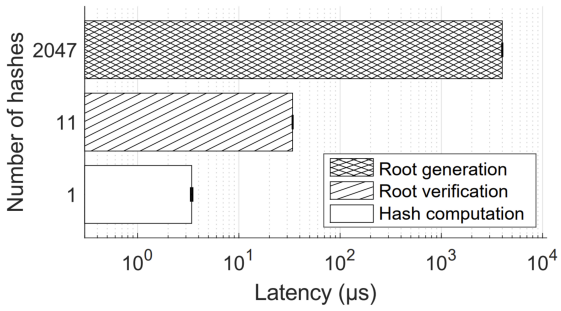


Figure 2: Time latency in the horizontal axis (measured) and the number of hash computations needed in the vertical axis (the implementation counter and the theoretical derivation yield consistent values).

has a tradeoff in the security property because it only assures that a wireless user/prover holds the software code files, in contrast to the code *attestation* providing stronger integrity assurance including the code execution.

Our work emphasizes the lightweight design and implementation, and thus provides greater applicability in enabling code assurance functionality to broader applications than code attestation. Specifically, the lightweight design will enable the functionality in the resource-constrained devices with limited hardware and power/networking capabilities; and the lack of real-time and user-specific networking makes our scheme broadcasting-friendly.

2 DESIGN AND REQUIREMENTS

Requirements and Approach The first two requirements are for security, while the last requirement is for correctness and efficiency. 1) To prevent an attacker without the files from generating the correct proof, we build on the cryptographic primitives for our scheme, e.g., one-way and the collision-resistance properties of the cryptographic hash function. 2) To disable an attacker from reusing the proof by other users (e.g., after eavesdropping), we take the following two approaches. First, we design our scheme so that the proofs are unique to each user; we include the user ID (e.g., the user's IP address) as the input for generating the proof. Second, we only network the proof output (the Merkle root) as opposed to the intermediate branch values, i.e., all the other intermediate values, except for the root-proof, stays and gets processed within the local host (similarly to a private key in public-key cryptography). 3) For correctness and consistency across the users (enabling the verification by other users and the utilization of the Merkle tree for efficiency), we establish a rule for ordering the files for input into our scheme (which, in our implementation, is based on the file sizes).

Prover and Verifier In our scheme, the *prover* (providing the proof for its code assurance) requires generating the full tree to retrieve the root. The *verifier* is required to have the same code files (e.g., the same software version implemented), and have previously generated the root. If the verifier does not have the same software files, our scheme is not applicable.

Root Generation at Prover Our design uses the Merkle tree [9] with the code files and the user ID as inputs. We denote F as an ordered set of code files to be used as leaf nodes in the binary tree, which yields $2^{\lceil \log_2(|\mathrm{F}|) \rceil}$ (i.e. next power of two) number of leaves for the Merkle tree for a balanced tree. For example, in Figure 1, there are 10 code files, i.e., $\mathrm{F} = \{\mathrm{F}_1, \mathrm{F}_2, ..., \mathrm{F}_{10}\}$. We round this to the next power of two such that we construct a new set $\hat{\mathrm{F}}$ that has a size of $|\hat{\mathrm{F}}| = 16$. The additional files are generated by cycling through the previous files in F so that $\hat{\mathrm{F}} = \{\mathrm{F}_1, \mathrm{F}_2, ..., \mathrm{F}_{10}, \mathrm{F}_1, \mathrm{F}_2, ..., \mathrm{F}_5\}$. The prover generates the full tree, and the number of hash computations required is $|\hat{\mathrm{F}}| - 1$.

Verification at Verifier During verification, the locally stored nodes, marked yellow in Figure 1, are used to reconstruct the root and are exclusively known to those who hold the software code and have completed the root computation. The nodes marked blue correspond to the set of hashes unique to the prover and verifier, and those which must be recomputed for verification. The verifier, having already generated the tree (e.g., for its own proof), only requires $\log_2(|\hat{\mathrm{F}}|)$ hash updates.

3 IMPLEMENTATION AND RESULTS

Our experimentation focuses on the feasibility of our scheme using our prototype on srsRAN release 21.10, an open-source tool used by researchers to implement with cellular technology [7]. Measurements were taken on Ubuntu 20.4.3 LTS using an Intel i7-11800H 2.30GHz processor with 16GB DDR4 at 3200MHz. For consistency with srsRAN, our implementation uses C++17, and the source code files of srsRAN are read into a vector representing $\hat{\mathrm{F}}$, with each node stored as three data fields: 1) the path to the file, 2) the corresponding hash of the file, and 3) the index mapping from files to the ordered set of leaf nodes, $\hat{\mathrm{F}}$. For hashing, we use Keccak-256, a member of the SHA-3 family of hashing algorithms.

Figure 2 plots the computational overhead to generate the Merkle root (done by the prover), verify the root (done by the verifier), and computation of a single hash, which is used in both root generation and root verification. The latency measurements are averaged over 10^5 samples, and the confidence intervals for latency are plotted but may not be visible. Verifying the tree takes 196.63% less time overhead compared to tree generation, and 197.86% less hashing overhead compared to tree generation. This reduction in overhead highlights the importance of using a Merkle tree for verification efficiency, which is often prioritized over the generation due to its

higher frequency in use, e.g., broadcasting with one prover and multiple verifiers.

4 CONCLUSION AND FUTURE WORK

Our scheme offers a lightweight solution, forgoing hardware-based trusted computing, for software assurance in wireless and resource-constrained devices. Leveraging the properties of the Merkle tree, we are able to reduce the computational complexity to further make the verification lightweight.

This work will facilitate further research for secure and lightweight wireless network solutions for software assurance. Potential future work in this direction includes: further analyses of our work (e.g., the Merkle tree structure parameter and its performance analyses) and the comparative analyses between assurance vs. attestation (e.g., the latency/power costs and hardware comparison and the practicality analyses on resource-constrained applications).

ACKNOWLEDGEMENT

This work was supported by National Science Foundation under Grant No. 1922410 and by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-02107, Collaborative research on element Technologies for 6G Security-by-Design and standardization-based International cooperation).

REFERENCES

[1] Mahmoud Ammar, Bruno Crispo, Ivan De Oliveira Nunes, and Gene Tsudik. 2021. Delegated Attestation: Scalable Remote Attestation of Commodity CPS by Blending Proofs of Execution with Software Attestation. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Abu Dhabi, United Arab Emirates) *(WiSec '21)*. Association for Computing Machinery, New York, NY, USA, 37–47. https://doi.org/10.1145/3448300.3467818

[2] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. 2008. Wireless Device Identification with Radiometric Signatures. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking* (San Francisco, California, USA) *(MobiCom '08)*. Association for Computing Machinery, New York, NY, USA, 116–127. https://doi.org/10.1145/1409944.1409959

[3] Sang-Yoon Chang and Yih-Chun Hu. 2017. SecureMAC: Securing Wireless Medium Access Control Against Insider Denial-of-Service Attacks. *IEEE Transactions on Mobile Computing* 16, 12 (2017), 3527–3540. https://doi.org/10.1109/TMC.2017.2693990

[4] Sang-Yoon Chang, Yih-Chun Hu, and Nicola Laurenti. 2012. SimpleMAC: A Jamming-Resilient MAC-Layer Protocol for Wireless Channel Coordination. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking* (Istanbul, Turkey) *(Mobicom '12)*. Association for Computing Machinery, New York, NY, USA, 77–88. https://doi.org/10.1145/2348543.2348556

[5] Charles Clancy, Joe Hecker, Erich Stuntebeck, and Tim O'Shea. 2007. Applications of machine learning to cognitive radio networks. *IEEE Wireless Communications* 14, 4 (2007), 47–52.

[6] George Coker, Joshua Guttman, Peter Loscocco, Amy Herzog, Jonathan Millen, Brian O'Hanlon, John Ramsdell, Ariel Segall, Justin Sheehy, and Brian Sniffen. 2011. Principles of remote attestation. *Int. J. Inf. Sec.* 10 (06 2011), 63–81. https://doi.org/10.1007/s10207-011-0124-7

[7] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D Sutton, Pablo Serrano, Cristina Cano, and Doug J Leith. 2016. srsLTE: An open-source platform for LTE evolution and experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. 25–32.

[8] William D Horne. 2003. Adaptive spectrum access: Using the full spectrum space. In *Proc. Telecommunications Policy Research Conference (TPRC)*.

[9] Ralph C Merkle. 2019. Protocols for public key cryptosystems. In *Secure communications and asymmetric cryptosystems*. Routledge, 73–104.

[10] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. 2004. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *13th USENIX Security Symposium (USENIX Security 04)*. USENIX Association, San Diego, CA. https://www.usenix.org/conference/13th-usenix-security-symposium/design-and-implementation-tcg-based-integrity-measurement

[11] E. Shi, A. Perrig, and L. Van Doorn. 2005. BIND: a fine-grained attestation service for secure distributed systems. In *2005 IEEE Symposium on Security and Privacy (S P'05)*. 154–168. https://doi.org/10.1109/SP.2005.4

[12] Karaputugala Madushan Thilina, Kae Won Choi, Nazmus Saquib, and Ekram Hossain. 2013. Machine learning techniques for cooperative spectrum sensing in cognitive radio networks. *IEEE Journal on selected areas in communications* 31, 11 (2013), 2209–2221.

[13] Tien Dang Vo-Huu, Triet Dang Vo-Huu, and Guevara Noubir. 2021. Spectrum-Flexible Secure Broadcast Ranging. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Abu Dhabi, United Arab Emirates) *(WiSec '21)*. Association for Computing Machinery, New York, NY, USA, 300–310. https://doi.org/10.1145/3448300.3467819