

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Anonymous Tokens with Designated-Reader Metadata Bit

Aisha Tu, *Wuhan University;* Meng Jia, *The Hong Kong Polytechnic University;*
Kun He, Jing Chen, and Ruiying Du, *Wuhan University*

https://www.usenix.org/conference/usenixsecurity26/presentation/tu

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology



USENIX Security '26 Artifact Appendix: Anonymous Tokens with Designated-Reader Metadata Bit

Aisha Tu[†], Meng Jia[‡], Kun He[†*], Jing Chen[†], Ruiying Du[†]
[†]Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University
{aishatu, hekun, chenjing, duraying}@whu.edu.cn
[‡]Department of Computing, The Hong Kong Polytechnic University
mengjia@polyu.edu.hk

A Artifact Appendix

A.1 Abstract

Anonymous tokens with private metadata bit convey hidden signals to verifiers when presented by the user and are under discussion in standardization. Existing solutions only allow the token issuer to read the signals, placing a heavy burden on the issuer and making it challenging to support issuer-hiding, as verifiers must contact the issuer. In this paper, we propose an anonymous token scheme with designated-reader metadata bit, named HinToken, allowing the user to specify an issuer-accepted verifier to read the signal from the token directly. We also extend our scheme to support reader-hiding (HinToken-R) and issuer-hiding (HinToken-I).

This artifact provides implementations and benchmarks of HinToken, HinToken-R, and HinToken-I, and can be run on a single machine with a Rust development environment. In our artifact evaluation, for each construction, we mainly consider the token size and the time costs of key generation, token issuance, and token verification.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

We believe that executing our artifact will not raise any machine security, data privacy, or ethical concerns for evaluators.

A.2.2 How to access

This artifact can be found at https://doi.org/10.5281/zenodo.17946122.

A.2.3 Hardware dependencies

This artifact can be run on most laptops and desktops. We run it on a laptop with Apple M4 Pro chip and 24 GB RAM.

*Corresponding author

A.2.4 Software dependencies

This artifact can be run on a single machine with Rust version 1.88 or later. Any mainstream operating system is compatible (we have tested them on macOS Tahoe, Ubuntu 24.04, and Debian 13; the data in our paper was tested on macOS Tahoe).

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

Run `cargo build --release` to compile the implementations of HinToken, HinToken-R, and HinToken-I.

A.3.2 Basic Test

Run `cargo test --release` to test the implementations of HinToken, HinToken-R, and HinToken-I.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): The runtime of IKeyGen, RKeyGen, and Read in HinToken, HinToken-R, and HinToken-I is roughly the same. The runtime of User_0 and Sign in HinToken-R is slower than that in HinToken and HinToken-I, and the runtime of User_1 in HinToken-I is slower than that in HinToken and HinToken-R. Unless the parallel optimization mentioned in Section 7 of our paper fails due to environmental reasons, the runtime of Verify in HinToken-I is roughly the same as that in HinToken and HinToken-R; otherwise, it will be slower than the other two. The running time and token size of our constructions are both worse than those in PMBT and ATHM. All results are shown in Table 1.

A.4.2 Experiments

(E1): Benchmarks of this artifact.

Execution: Run `python3 bench.py` to benchmark the implementations of HinToken, HinToken-R, and HinToken-I. This script will run `cargo bench` to benchmark each implementation and summarize all results in the file `res.md`.

Results: The raw data for each implementation is stored in its respective folder. Taking HinToken as an example, the raw runtime results are displayed in the `index.html` file under the folder `HinToken/target/criterion/report`, including the algorithms IKeyGen (IKeyGen), RKeyGen (RKeyGen), User_0 (user0), Sign (sign0), User_1 (user1), Verify (vrfy), and Read (readbit). The token size is stored in the `_summary_token_size.txt` file under the folder `HinToken/target/criterion`.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.