PK-PoMLO: Public Key Proof of ML Ownership System

Joyeon Park, Jinah Seo, Do. KyoungHwa* and Soo Yong Park

Sogang University, Republic of Korea
E-mail: joanne4p@gmail.com; jinah12@sogang.ac.kr; doda0905@gmail.com; sypark@sogang.ac.kr
**Corresponding Author*

Received 07 December 2025; Accepted 06 January 2026

Abstract

In this study, we propose an on-chain-based ML ownership proof system (PK-PoMLO), which combines a digital signature and a blockchain timestamp value to generate a certificate of ownership that is publicly disclosed on-chain, enabling strong claim of ML ownership. First, the owner creates a certificate signed with their private key using the hash value of the ML model and a structured message, and includes a timestamp. This is then used to generate an ML ownership certificate and registered on-chain. At this time, the owner uses their private key to create a standard signature value as a 128-bit mark and embeds it in the ML model. Anyone wishing to verify ML ownership then uses the owner's public key to compare the hash value of the on-chain ML ownership certificate with the timestamp value to verify ML ownership. In other words, we can verify the authenticity of the owner by testing whether the bit error rate (BER) between the mark extracted from the ML ownership certificate and the internally stored mark string satisfies BER $\leq \tau$, and verifying it with the signature value of the ML ownership certificate. To verify the results of this study, we implement and evaluate a prototype on the MNIST MLP and the Ethereum Sepolia test network.

Keywords: Machine Learning, ML Ownership, blockchain, timestamp.

Journal of Web Engineering, Vol. 25_1, 51–66.
doi: 10.13052/jwe1540-9589.2514
© 2026 River Publishers

1 Introduction

As digital transformation accelerates, large scale machine learning (ML) models have become core intangible assets across industries. A model is not a mere bundle of code it embodies substantial value accumulated through extensive data collection, compute expenditure, and expert time over long periods. However, once a model is distributed externally as a weight file, container image, or prediction application programming interface (API), it is immediately exposed to disputes arising from unauthorized redistribution, model extraction, or minor modifications followed by re asserted ownership. Recent studies have shown that the functionality of a machine learning model can be closely replicated using only its prediction API. This makes model extraction not just a theoretical concern, but a practical threat even in commercial environments [1].

To defend model ownership there are a lot of ways such as collecting internal artifacts such as file creation timestamps, operational logs, or deployment histories. However, these evidence assume trust in the model operator and are susceptible to tampering or loss. Traditionally, machine learning watermarking has been actively explored as a way to embed ownership information directly into the model. These watermarks usually use bit string patterns embedded in the parameter space, intermediate activations, or input, output behavior can help identify the rightful owner [2–5].

However, two key challenges remain unresolved in real world ownership disputes. First, it's unclear whether existing watermarking methods cryptographically bind the mark to a specific identity. Second, there's no strong mechanism to objectively prove when a claim was first made. Evidence stored in centralized systems or private repositories offers limited trust guarantees, and weak links between identity and watermark leave the system vulnerable to attacks, such as key substitution.

To address this, this paper suggest strong ownership claim mechanism that combines cryptographic identity binding and blockchain based timestamps. Public blockchains is a decentralized and tamper resistant alternative. By anchoring data hashes through a consensus mechanism. This allows third parties to independently verify the specific data actually exist at a specific point in time, asserting model ownership.

Also designed to be reproducible and verifiable by third parties. We provide strong attribution by embedding a 128-bit internal watermark derived from the owner's digital signature into the model. By committing the model hash and structured message to a public blockchain, anyone can

independently verify the transaction to determine who owns the model and when it was objectively.

The embedded watermark is used as evidence through a judgment process based on the Bit Error Rate (BER). A match is determined when the BER between the extracted watermark and the expected watermark is below a predefined threshold τ. Using practical parameters ($T = 128$, $\tau \leq 0.2$), we show that the false acceptance rate (FAR) from the attacker's perspective is offset by the binomial tail probability and is negligibly small in real-world environments.

2 Background & Related works

2.1 Machine Learning Ownership Techniques

In Machine Learning ownership techniques there are ML watermarking, Backdoor, model fingerprinting etc. These techniques broadly divided into black-box and white-box categories. Both focus on proving the existence of a watermark, but they differ significantly in terms of access rights, deployment environment, verification procedure, and reproducibility. Below, we summarize the key ideas, advantages, and disadvantages of each category [4, 5].

The black box category assumes an environment without access to the model internals, and determines the existence of a watermark based on the outputs for a specific query set (triggers). Representative work by Adi et al. proposed backdoor based watermarking, which induces predefined responses to trigger inputs, thereby enabling verification in remote environments [6]. Le Merrer et al. presented a method that verifies the watermark through query response patterns by manipulating the decision boundary [7].

White box techniques directly insert bit string into the model and then extract them using internal statistics such as projection techniques [8]. Uchida et al. proposed a method which encode a watermark by adjusting the weight distribution of a specific layer and verify its presence through statistical testing [2]. Chen et al. presented a general framework for imposing projection constraints on intermediate activations and comparing them to the target bit string. This method has demonstrated high applicability across a variety of network architectures and tasks [3].

In summary, black box watermarking techniques offer the advantage of being deploymented, and suffer from limitations such as trigger input managements. Verification threshold settings, and often the lack of attribution and timestamp evidence. White box techniques offer high reliability in watermark

extraction, but require access to the model file, which can pose operational limitations.

Previous research has explored the presence and robustness of watermarks in these two categories, that the specific owner isn't bind to this watermarked machine learning or providing non repudiation by using public timestamping.

In this study, we adopt the white box approach, which is more compatible with cryptographic binding via private keys. For implementation, we build upon the open source DeepSigns framework [9].

2.2 Digital Signature (ECDSA, EIP-712)

A digital signature provides message integrity, signer authentication, and non-repudiation. This study employs elliptic curve digital signatures (ECDSA) over secp256k1, where for a message m the signer generates σ = Sign sk(m), and verification accepts the signature if Verify (PK, σ, m) = true. Since ECDSA is sensitive to the entropy of the nonce k, we adopt the deterministic signing procedure of RFC 6979, which derives k from an HMAC-DRBG so that identical (sk, m) pairs yield identical signatures [10]. In addition, the Ethereum ecosystem enforces low S normalization ($s \leq n/2$) under EIP-2, thereby eliminating representation variability of signatures for the same message [11, 12]. To make the semantics of signed data explicit to both users and verifiers, this work adopts the structured data hashing and signing specification of EIP-712 [13].

2.3 Blockchain Stamping (Ethereum, Keccak-256)

A public blockchain functions as a distributed timestamping server by linking blocks containing hashes through decentralized consensus, thereby allowing anyone to verify that specific data existed prior to a given point in time [14]. For convenience of verification and ecosystem tool support, this study employs Ethereum (Sepolia testnet). Ethereum uses Keccak-256 as its default hash function for key data such as block state, transactions, and roots; commitments are aggregated into the block header via the RLP/Patricia Trie structure, forming part of the chain's immutability [15, 16]. To balance privacy and gas costs, only summary hashes (e.g., H(m), H(σ)) are minimally recorded on-chain, while the original data are kept and submitted off-chain to provide a reproducible verification path [17, 18].

3 System Overview

In this section, we will describe the system overview briefly. The full proposed process will be explained in Section 4.

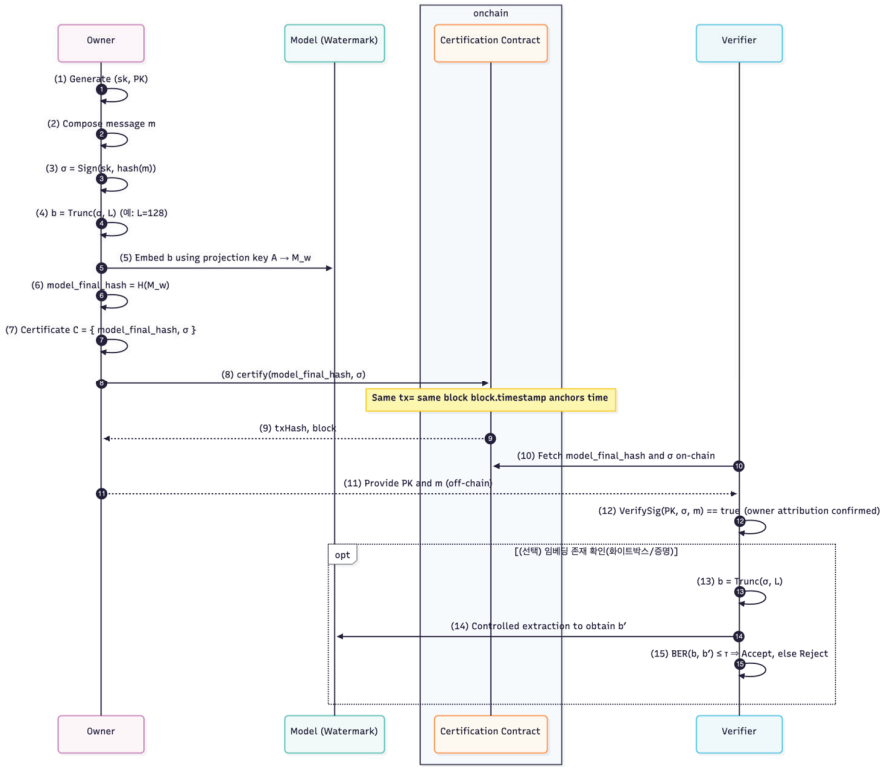

Figure 1 System overview.

As shown in Figure 1 the core of the system is that the owner's signature value is directly used to derive an internal mark b bound into the model, while its counterpart sig(m) and PK are committed on-chain, thereby providing attribution and timestamping simultaneously. Attribution arises from the fact that b is not an arbitrary secret but is deterministically derived from sig(m).

Since sig(m) can only be generated with the corresponding private key, the probability that a third party could independently produce the same b is essentially eliminated. Conversely, timestamping is achieved through the minimal on-chain commitment of sig(m) and PK. Such a commitment, without revealing the original content, records in a public ledger which public

key claimed which signature value at what time, thereby enabling objective determination of priority in case of disputes.

In this design, the computations and embedding namely, the generation of b and its binding within M are performed entirely off-chain, ensuring that sensitive information is not exposed. The on-chain layer focuses solely on notarization (anchoring) with minimal data. A verifier checks the existence of the on-chain commitment to establish the timestamp, validates the signature (PK, m, σ), and, within allowed access, extracts b' from M. Ownership is then confirmed if the condition BER $(b, b') \leq \tau$ holds. The fact that b is derived from the signer's own sig(m) provides the technical basis for attribution, while the commitment of sig(m) and PK on-chain serves as the technical basis for timestamping.

This approach offers several key advantages over conventional watermarking techniques. Traditional watermarking can confirm the presence of a watermark but typically provides only weak attribution, as the mark is not inherently tied to a specific owner.

In contrast, the proposed system cryptographically binds the watermark to the owner's public key by deriving the embedded bit string b from the signature value sig(m), thereby enabling strong ownership attribution.

Furthermore, by committing this signature on-chain, a publicly verifiable timestamp is created, which can be used as a powerful piece of evidence to resolve ownership disputes, especially when identical or competing claims arise. Moreover, the activation-based extraction mechanism tied to the input set A ensures that b' can only be retrieved under controlled conditions.

4 Proposed Method

This section follow the Figure 2 steps (①–⑪), focusing on why each step is necessary and what guarantees are obtained. The notation is kept identical to the diagram.

4.1 Create Ownership Proof

The owner first prepares a key pair, then generates sig(m) for message m using the private key, and derives b = Trunc (sig(m), L) from this signature value. In the Select layer, projection A = Target layer activation key (...) step, the embedding target layer, projection A, and activation conditions are determined. During the Embed signature by fine-tune step, fine-tuning is performed so that b is stably bound to the internal representation of M without

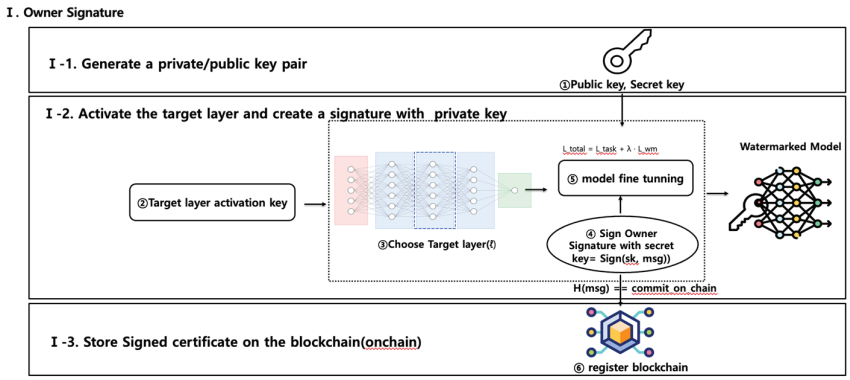

Figure 2 Proposed method.

degrading the model's original functionality. Once embedding is completed, the process outputs the watermarked model M, and then commits sig(m) and PK on-chain. This commitment anchors equivalence and timestamp without revealing the original data, and allows objective proof of priority for the corresponding signature–public key pair in the event of disputes. The essence of issuance is binding b within M and committing the supporting evidence sig(m) and PK on-chain. By deriving b directly from sig(m), the embedded value becomes a structure generable only by the private key, rather than a mere identifier or random token. As a result, it is practically infeasible for a third party to independently reproduce the same b, and even if the model is replicated or redistributed, the internal mark is automatically attributed to the original owner.

4.2 Verification of Ownership

The purpose of the verifier's procedure is to confirm who, when, and what (i.e., the mark b derived from sig(m)) is being claimed as ownership. The verifier first checks in step ⑥ whether Commit (sig(m), PK) exists on-chain. This confirmation anchors the claim to a specific time prior to a given block height, serving as the prerequisite for subsequent procedures. Next, for the submitted (publickey, m, sig), the verifier performs step ⑦ verify (publickey, m, sig) and confirms the outcome of step ⑧ (True). This step guarantees that sig(m) is indeed a valid signature under PK, ensuring that all subsequent conclusions are grounded in attribution to the owner's key. With these two checks alone, the core guarantees intended by the design attribution (who the claim belongs to) and timestamping (when the claim was made) are satisfied.

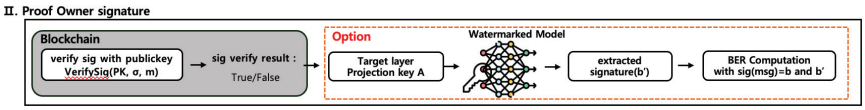

Figure 3 Verification of ownership.

As shown in Figure 3 for dispute resolution or in contexts demanding additional assurance, the actual existence of the embedding can be lightly verified. In such cases, the verifier initiates step ⑨, a restricted verification procedure, and when necessary derives b' from M according to A in step ⑩. This process is performed in a controlled environment under the owner's supervision, without releasing the model externally; only the procedure and logs may be shown to outsiders. The final judgment is made in step ⑪ by checking whether BER $(b, b') \leq \tau$ holds.

5 Experiments & Evaluation

5.1 Experimental Environment

We conduct our experiments on the MNIST handwritten digit dataset. As the target model, we use a three-layer fully-connected neural network with architecture $784 \rightarrow 256 \rightarrow 128 \rightarrow 10$ and ReLU activations. For watermark embedding, we reuse the open-source implementation of DeepSigns [15] as our baseline. In the original DeepSigns setting, the embedded bit string is sampled uniformly at random. In our prototype, we instead derive the embedded bit string from the owner's public key, so that the internal watermark is cryptographically tied to a specific public key rather than to a random identifier.

All experiments are performed on a desktop PC. AMD Ryzen 5 5600X CPU, an NVIDIA GeForce RTX 3070 Ti GPU (8 GB), 8 GB of DDR4-3200 RAM, and a 1 TB SK hynix Gold P31 NVMe SSD, with a Gigabyte B550M AORUS PRO-P motherboard running Windows.

For on-chain anchoring and verification, we implement and deploy the smart contract using the Remix IDE on the Ethereum Sepolia test network, and we inspect transactions and gas usage through Etherscan.

5.2 Offchain Evaluation

Off-chain, we first generate a digital signature σ = sig(m) on a message m, and derive the L-bit internal mark b=Trunc (sig(m), L). The projection matrix A is

fixed in advance, and watermark embedding is performed by fine tuning the designated hidden layer of the model M using the DeepSigns procedure [3, 15]. We set the watermark length to L=128 bits, which we found to be the maximum length that does not noticeably degrade the model's classification accuracy; in our prototype, the watermarked model achieves about 96% accuracy on MNIST, comparable to the non watermarked model. After training, we repeatedly extract b' from the marked model under the same projection A and compute the bit error rate BER (b, b'). In our experiments with the public key derived mark, the observed BER (b, b') for the legitimate owner remained below 0.01, which is well within the decision threshold τ that we consider acceptable. Thus, for the intended owner, the ownership verification test BER $(b, b') \leq \tau$ consistently succeeds, indicating that using a public key derived bit string as the internal watermark does not hinder reliable extraction.

5.3 False accept rate under random-key attacks

We next examine how likely it is for an adversary, who does not know the embedded mark b, to succeed by random guessing in Figure 4. In our threat model, the adversary can freely choose a key pair and derive a candidate bit string b^* from the public key using the same rule as the legitimate owner (secp256k1 key generation, SHA-256 of the uncompressed public key, and truncation to 128 bits). The question is how often such a candidate bit string could accidentally satisfy BER $(b, b^* \leq \tau)$. For the analytical estimate, we model the number of matching bits between b and b^* as a binomial random variable X∼Binomial (L, 1/2), since each bit matches with probability 1/2

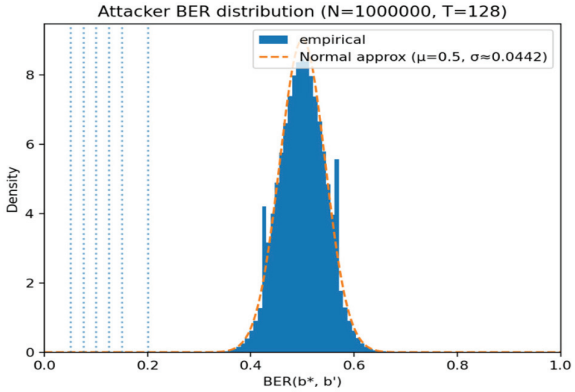

Figure 4 False accept rate.

under random guessing. The probability that a random wrong mark passes the test BER $(b, b^* \leq \tau)$ is given by the left tail of this distribution.

$$\Pr[\mathrm{BER}(b, b^*) \leq \tau] = \sum_{k=0}^{\lfloor \tau L \rfloor} \binom{L}{k} 2^{-L}$$

For $L = 128$ and $\tau \leq 0.2$, this probability becomes extremely small, and for stricter thresholds $\tau = 0.1$ it rapidly approaches a negligible value. To empirically confirm this behavior under our key-derivation rule, we perform a Monte Carlo experiment with $T = 128$ bits and $N = 1{,}000{,}000$ trials. In each trial, we generate a random secp256k1 key pair, compute the corresponding public key, hash it with SHA-256, truncate the result to 128 bits to obtain a candidate b^*, and compute BER (b, b^*) against a fixed reference bit string b. The resulting BER values form a bell-shaped distribution centered near 0.5, consistent with the binomial model. For thresholds τ up to 0.2, the empirical false accept rate is extremely small; in particular, when $\tau = 0.01$, the probability that a random key derived bit string passes the test effectively converges to zero in our experiments.

These results indicate that, under random-key attacks, PK-PoMLO's ownership test has a negligible false accept rate at the parameter sizes considered, and a random third party is extremely unlikely to generate a key that yields a bit string close enough to the embedded mark to be falsely accepted.

6 Conclusion

This paper presented a concise framework that simultaneously achieves attribution and timestamping by binding a mark b derived from a private-key signature value into the model, while anchoring the corresponding sig(m) and PK on-chain. All computations are performed off-chain prior to commitment, and the blockchain is used solely as a minimal anchor through Commit (sig(m), PK), which objectively fixes the claim's timestamp. Verification proceeds by confirming the on-chain commitment and validating verify (pubkey, m, sig), thereby resolving the essential questions of who and when. If needed, the restricted procedure extracts b' according to A from M, and only checks whether BER $(b, b') \leq \tau$, serving as auxiliary confirmation of embedding.

This approach separates attribution and timestamping by anchoring the signed message sig(m) and the public key PK on-chain. In the real world, this method is particularly will be suitable where model ownership potentially holds significant value, such as AI model experimentation, research,

compliance during AI deployment, and record-keeping for proprietary models. For example, startups or companies distributing models such as APIs or edge devices can leverage this mechanism to embed ownership without exposing secret data.

But this study have some limitations. To verify whether the inserted bit string b actually exists within the model, b' must be extracted, a process that currently requires controlled access or specific procedures. However, these limitations can be seen as a deliberate trade off that prioritizes simplicity and privacy over widespread public accessibility. However this does not compromise the system's core guarantee that the watermark exists within Model M as a cryptographically bound form to the owner's signature, and that the ownership claim is publicly and verifiably committed on-chain at a specific point in time.

For future work, we plan to use zero-knowledge techniques such as zkml, zkVM to leave verifiable evidence of extracting b'. This would allow the construction of a complete ownership certificate that can be registered on-chain without requiring direct inspection of the model.

Acknowledgement

This work was supported by the IITP (Institute of Information & Coummunications Technology Planning & Evaluation) – ITRC (Information Technology Research Center) grant funded by the Korea government (Ministry of Science and ICT) (IITP-2026-RS-2023-00259099)

References

[1] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs", Proc. USENIX Security, 2016, pp. 601–618.

[2] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in Proc. ACM Int. Conf. on Multimedia Retrieval (ICMR), 2017, pp. 269–277.

[3] B. D. Rouhani, H. Chen, and F. Koushanfar, "DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural networks," in Proc. ASPLOS, 2019, pp. 485–497.

[4] F. Boenisch, "A Systematic Review on Model Watermarking for Neural Networks," Frontiers in Big Data, vol. 4, Art. no. 729663, 2021, doi: 10.3389/fdata.2021.729663.

[5] Y. Li, H. Wang, B. Wang, and Z. Zhang, "A Survey of Deep Neural Network Watermarking Techniques," Neurocomputing, vol. 461, pp. 171–193, 2021, doi: 10.1016/j.neucom.2021.07.051.
[6] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring," in Proc. 27th USENIX Security Symp. (USENIX Security '18), 2018, pp. 1615–1631.
[7] F. Le Merrer, B. Perez, and G. Trédan, "Adversarial Frontier Stitching for Remote Neural Network Watermarking," Neural Computing and Applications, vol. 32, no. 13, pp. 9233–9244, 2020, doi: 10.1007/s00521-019-04434-z.
[8] Y. Yan et al., "Rethinking White-Box Watermarks on Deep Learning: Are They Robust to Neural Structural Obfuscation," in Proc. 32nd USENIX Security Symp. (USENIX Security '23), 2023, pp. 2347–2364.
[9] RorschachChen, DeepSigns-torch (GitHub repository), 2018–. (Accessed: 2026-01-07).
[10] T. Pornin, "RFC 6979: Deterministic Usage of the Digital Signature Algorithm (DSA) and ECDSA," IETF RFC 6979, 2013.
[11] V. Buterin, "EIP-2: Homestead Hard-fork Changes," Ethereum Improvement Proposals, no. 2, 2015. (Accessed: 2026-01-07).
[12] Z. Wang, "Blockchain-Assisted Robust Subgroup ECDSA Multisignature for Consensus," IEEE Internet of Things Journal, vol. 12, no. 4, pp. 4525–4535, 2025, doi: 10.1109/JIOT.2024.3485215.
[13] R. Bloemen, L. Logvinov, and J. Evans, "EIP-712: Typed structured data hashing and signing," Ethereum Improvement Proposals, no. 712, 2017. (Accessed: 2026-01-07).
[14] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," White Paper, 2008. (Accessed: 2026-01-07).
[15] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger (Yellow Paper)," Technical Report, 2014. (Accessed: 2026-01-07).
[16] X. Lin, L. He, and H. Yu, "Practical Preimage Attacks on 3-Round Keccak-256 and 4-Round Keccak [r=640, c=160]," IACR Trans. Symmetric Cryptology, vol. 2025, no. 1, pp. 328–356, 2025, doi: 10.46586/TOSC.V2025.I1.328-356.
[17] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "The Keccak Sponge Function Family–Main Document," Ver. 2.0, Sep. 10, 2009. (Accessed: 2026-01-07).

[18] K. Bak, H. Salin, K. Niczyj, and L. Krzywiecki, "Enhancing Tunnel Safety for Dangerous Goods Vehicles through Blockchain-Based Time-Stamping," in Proc. IEEE 22nd Int. Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom), 2023, pp. 1312–1317, doi: 10.1109/TrustCom60117.2023.00179.

Biographies

Joyeon Park is a last-semester master's student in computer science at Sogang University. She received her bachelor's degree in intellectual property from Kyonggi University in 2022. Her research interests are in the convergence of blockchain technology and intellectual property. She is currently working on zero-knowledge machine learning (ZKML) technologies. Also, working on a ITU-T Techical Report related to ZKML.

Jinah Seo is a 5th semester Ph.D. student in the Department of Computer Science and Engineering at Sogang University. She has over 10 years of experience in communication network deployment and security consulting and received her M.S. degree from the Graduate School of Information and Communication at Sogang University in 2022. Her current research focuses

on the security aspects of blockchain technology, and she has contributed to ITU-T SG17 standardization meetings.

Do. KyoungHwa received her Ph.D. in Computer Communication and Information Security from Soongsil University in 2004. She currently serves as a professor in the College of Engineering at Sogang University. Her research areas include artificial intelligence (AI), blockchain, physical AI, information security, and cloud computing. She has also served as a reviewer for several prominent academic journals. She served as the head of the New Technology Team at the Ministry of the Interior and Safety for 15 years, researching and applying new technologies to e-government.

Soo Yong Park received his Ph.D. degree from George Mason University in 1995. He has held several prestigious positions, including serving as a Professor in the Department of Computer Science at Sogang University since March 1998 and as the Dean of the College of Software Convergence at Sogang University since July 2024. He is currently serving as the Chair of the Distributed Ledger Standard Forum and has been the Director of the Web 3.0 Research Center (ITRC) since 2023. Previously, he served as the President and CEO of the National IT Industry Promotion Agency (NIPA) from

September 2012 to November 2014. Since January 2019, he has also served as the President of the Korea Society of Blockchain and as the Director of the Intelligent Blockchain Research Center. His accolades include the 10-Year Most Influential Paper Award from the Asia–Pacific Software Engineering Conference (APSEC) in December 2018 and the Minister of Science and ICT Award for Best Project Evaluation in November 2021.