Future Generation Computer Systems 164 (2025) 107562

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

SeCTIS: A framework to Secure CTI Sharing

Dincy R. Arikkat [a], Mert Cihangiroglu [b], Mauro Conti [c], Rafidha Rehiman K.A. [a], Serena Nicolazzo [d], Antonino Nocera [b,*], Vinod P. [c,a]

[a] Department of Computer Applications, Cochin University of Science and Technology, India
[b] Department of Electrical, Computer and Biomedical Engineering, University of Pavia, Italy
[c] Department of Mathematics, University of Padua, Italy
[d] Department of Computer Science, University of Milan, Italy

ARTICLE INFO

Keywords:
Cyber threat intelligence
CTI sharing
Swarm learning
Federated learning
Blockchain
Zero knowledge proof
Internet of Things (IoT)

ABSTRACT

The rise of IT-dependent operations in modern organizations has heightened their vulnerability to cyber-attacks. Organizations are inadvertently enlarging their vulnerability to cyber threats by integrating more interconnected devices into their operations, which makes these threats both more sophisticated and more common. Consequently, organizations have been compelled to seek innovative approaches to mitigate the menaces inherent in their infrastructure. In response, considerable research efforts have been directed towards creating effective solutions for sharing Cyber Threat Intelligence (CTI). Current information-sharing methods lack privacy safeguards, leaving organizations vulnerable to proprietary and confidential data leaks. To tackle this problem, we designed a novel framework called SeCTIS (Secure Cyber Threat Intelligence Sharing), integrating Swarm Learning and Blockchain technologies to enable businesses to collaborate, preserving the privacy of their CTI data. Moreover, our approach provides a way to assess the data and model quality and the trustworthiness of all the participants leveraging some *validators* through Zero Knowledge Proofs. Extensive experimentation has confirmed the accuracy and performance of our framework. Furthermore, our detailed attack model analyzes its resistance to attacks that could impact data and model quality.

1. Introduction

With the advent of Industry 5.0, organizations tend to include smart technology in their systems with the objective of delegating repetitive and time-consuming activities to support devices, such as cobots or smart objects, and digital twins. As a consequence, the attack surface and the potential harm to the safety of cyber–physical systems is expanding exponentially [1]. Indeed, as reported by Parachute,[1] the year 2022 saw a significant resurgence in malware attacks, with the number soaring to an astonishing 2.8 billion. Moreover, the Anti-Phishing Working Group (APWG)[2] reported that the second half of 2023 alone saw five million phishing attacks, and from March to May 2023, threat actors initiated an average of 11.5 attacks per minute, incorporating 1.7 newly developed malware samples per minute. Hence, in the event of a cyber incident, having access to timely and relevant threat intelligence can greatly aid in reaction and mitigation efforts. It can provide valuable context about the attacker's Tactics, Techniques, and Procedures (TTPs), helping organizations prevent and remediate the incident more effectively.

In this context, Cyber Threat Intelligence (CTI, hereafter) has emerged as a powerful tool to gain knowledge and insights about cyber threats and adversaries. CTI refers to systematically collecting, analyzing, and interpreting data related to vulnerabilities, threat reports, and attack trends observed across various sectors. CTI enables industries to understand the evolving threat landscape better, anticipate potential cyberattacks proactively, and carry out possible defenses [2].

In the pervasive environment, connected smart objects and sensors produce an enormous amount of CTI in multiple forms and types. On the other hand, organizations cannot rely solely on their internally generated CTI to protect themselves; they need to benefit from the knowledge coming from external sources such as network traffic, hacker forums, APT reports, technical blogs, etc. CTI Sharing plays a crucial role in enabling organizations to disseminate both raw and

* Corresponding author.
E-mail addresses: dincyrarikkat@cusat.ac.in (D.R. Arikkat), mert.cihangiroglu01@universitadipavia.it (M. Cihangiroglu), mauro.conti@unipd.it (M. Conti), rafidharehimanka@cusat.ac.in (Rafidha Rehiman K.A.), serena.nicolazzo@unimi.it (S. Nicolazzo), antonino.nocera@unipv.it (A. Nocera), vinod.p@cusat.ac.in, vinod.puthuvath@unipd.it (Vinod P.).

[1] https://parachute.cloud/cyber-attack-statistics-data-and-trends
[2] https://apwg.org/trendsreports

https://doi.org/10.1016/j.future.2024.107562
Received 20 June 2024; Received in revised form 17 September 2024; Accepted 12 October 2024
Available online 19 October 2024

processed information. This helps organizations access external threat intelligence and enhance their collective ability to defend against cyber threats [3].

However, in practice, CTI Sharing is challenging due to a variety of factors [4]. First, participants may not want to disclose their identity to avoid damage to the organization's reputation. Unfortunately, this implies that if the source of certain data is unknown, the credibility of shared information is harmed, and trusting relationships among the participating entities cannot be easily established. Another factor linked to entities' trustworthiness that may discourage collaboration is the problem of incomplete or false information. This can contaminate or mislead the algorithms or the results of analysis. Furthermore, not all organizations are inclined to invest additional resources both for interoperability and to ensure that the shared CTI can be automated and easily reusable by the participating entities. Moreover, legal questions should be considered if the information to be shared contains materials protected under data protection and privacy law, antitrust law, or intellectual property law. For instance, in Germany, IP addresses are considered personal information; therefore, any disclosure of CTI, including them, must comply with German privacy laws [5]. Instead, in the UK, they can be freely shared. These legal and regulatory obligations can pose a significant barrier to international business cooperation. Although several solutions exist to provide organizations with an environment for sharing and consuming CTI [6,7], they are not designed to assess the privacy and trust of the participants [5].

We propose a novel framework, Secure CTI Sharing (SeCTIS), to contribute to this setting. SeCTIS is an architecture that allows organizations to share CTI data in a privacy-preserving way. Our framework is the first that collaboratively trains Machine Learning (ML) models on CTI data and assesses both the quality of data and models and the trustworthiness of all participants. To do this, SeCTIS integrates several technologies, such as Swarm Learning (SL), Blockchain Smart Contracts, and Zero Knowledge Proof mechanisms.

SL has been recently introduced [8] as a novel paradigm for collaborative and privacy-preserving Machine Learning. Similar to Federated Learning (FL), SL participants jointly train a global model and update their local model contribution. Instead of relying on a single aggregator, SL leverages a Blockchain to coordinate the model aggregation and securely onboard members. The aggregation of local model updates and the subsequent alignment of the local nodes are handled in a decentralized manner. In our framework, the different organizations represented by Swarm Edge Nodes compose a Swarm Network. They aim to collaboratively train a Global Model using their private CTI data and build the Local Models independently without revealing them to other participants. Only model parameters are shared via the Swarm Network. In this way, data security and confidentiality are preserved.

SeCTIS also includes some additional steps to assess the quality of the employed CTI data and model and compute the participants' trustworthiness. For this, in each iteration, a set of *validators* nodes is randomly selected among the participants to verify the performance of all the local model updates before their aggregation. To protect against malicious actions from the validators, a Zero-Knowledge Proof (ZKP, hereafter) mechanism is also employed. Finally, an elected *Swarm Aggregator* computes the reputation score of all local models using validator nodes and aggregates the parameter updates from the top-k local models. Subsequently, the aggregated Global Model is uploaded into IPFS and sent back to SL nodes to start the next iteration and continue until convergence. Through this mechanism, the quality of the model and the CTI data is assessed. SeCTIS also evaluates the reputation of participating organizations during the SL model training. Organizations' reputation is estimated by considering the quality of their contribution (local model updates) during each SL iteration.

Hence our framework significantly advances traditional CTI sharing methods in several aspects. Indeed, combining Swarm Learning with Blockchain and ZKPs creates a robust system where *(i)* data privacy is guaranteed (through the use of Swarm Learning), and both *(ii)* the quality of the model is assessed and *(iii)* trust among participants is ensured (thanks to Blockchain and ZKPs). Moreover, minimizing data exposure and ensuring secure, verifiable transactions address many of the limitations of existing regulatory frameworks that demand stringent data privacy and security measures. Hence, our comprehensive approach can significantly improve collective cybersecurity efforts, making organizations more resilient against threats. To the best of our knowledge, no current frameworks provide the possibility to keep data private during CTI sharing, maintaining model quality and assessing participants' trustworthiness at the same time.

In summary, the main contributions of this paper, intended to solve the major challenges in CTI Sharing, are as follows:

- **Privacy-preserving CTI data sharing**: our framework adopts an SL Network to generate a CTI Model in a distributed manner collaboratively. Since SL does not require data to be shared with a central entity, this decentralization protects data privacy as the raw data never leaves the local node, ensuring the confidentiality of each organization's data.
- **Trust among participants and quality of CTI data**: SeCTIS provides a process based on *validator* nodes to assess CTI data and model quality using reputation scores. In addition, through the ZKP mechanism, *validator* activities can be verified ensuring that malicious entities cannot compromise the system. These mechanisms make SeCTIS also a collaborative trust framework.
- **Interoperability and automation**: SeCTIS provides a middleware that can manage heterogeneous data formats and establish a unique methodology to be employed. Indeed, only model parameters (weights and biases) are shared, not the raw data. This decoupling allows systems with different data formats and structures to contribute to the collective model without compatibility issues. Moreover, different ML frameworks can be used to train local models as long as they can produce compatible model parameters for aggregation, thus enhancing interoperability across different platforms and tools. Furthermore, automation is achieved through both the decentralized and autonomous model training and the automated validation of participants, thus reducing the need for manual oversight and intervention.
- **Scalability**: SeCTIS is also a scalable solution to secure CTI sharing, indeed it leverages some mechanisms to improve the efficiency of the employed Blockchain. Moreover, the workload of training models is distributed across multiple nodes which reduces the burden on any single entity and allows the network to expand as needed.
- **Legal liabilities**: Keeping CTI data confidential may reduce legal risks concerning information disclosure, making organizations more willing to participate in SL-sharing schemes. By exchanging only the model updates or gradients, the amount of information that could potentially reveal sensitive data is minimized. Moreover, SeCTIS minimizes legal liabilities because it leverages Zero-Knowledge Proofs for verification without data exposure, encouraging broader participation by lowering legal barriers. Also, the use of Blockchain provides transparency and accountability through the shared ledger to assess the trustworthiness of all the steps of the framework for the different participants. All these features collectively help organizations mitigate the legal risks associated with data sharing and collaborative learning.

Our paper is organized as follows. Section 2 describes the main works related to our approach. Section 3 delves into the details about CTI, Blockchain, Federated Learning, Swarm Learning, and Zero-Knowledge Proof concepts that are essential to the understanding of our solution. Section 4 describes the main components of our framework and the steps we performed to secure CTI Sharing. In Section 5, we present our attack model, demonstrating our approach as robust to possible attacks against data and model quality. Section 6 deals with the experimental campaign used to assess the performance of our solution,

including the setup, results, and possible limitations of our system. An analysis of the security of validators' operations through ZKP is reported in Section 7. Section 8, instead, discusses some example attack scenarios and analyze the behavior of our solution. The performance impact of the inclusion of ZKP in our solution is evaluated in Section 9. Finally, Section 11 concludes the work and presents possible future directions.

2. Related work

In this section, we describe the related systems currently adopted to share CTI data and their characteristics.

Although lots of organizations still rely on informal means (i.e., phone calls or emails) for sharing CTI-related information, recently, there has been a growing interest in dedicated platforms to facilitate the automated or semi-automated sharing of CTI data inside connected communities [9]. Threat Intelligence Platforms (TIPs, hereafter) are specialized software that helps industries collect and analyze real-time threat information from various sources to support defensive tactics. While numerous TIPs are available in the market, most are offered under commercial licenses. For instance, VirusTotal[3] is one of the leading CTI service able to analyze suspicious files, domains, IPs, and URLs to detect malware and other breaches, produce threat reports, and automatically share them with the security community. Another similar open-source solution, known as MISP (Malware Information Sharing Platform) [6], aims to gather, store, and distribute cybersecurity IoCs and CTI reports, both within the security community and beyond. MISP offers a range of features, such as an indicator database, automated correlation, sharing capabilities, a user-friendly interface, and compatibility with various data formats and standards.

The authors of [10] assess nine TIPs such as ThreatStream,[4] ThreatQ,[5] ThreatConnect,[6] Open Threat Exchange (OTX),[7] MISP, IBM X-Force Exchange,[8] Falcon X Intelligence Crowdstrike,[9] Collective Intelligence Framework (CIF),[10] and Collaborative Research into Threats (CRITs)[11] by examining how they align with the CTI life cycle. Their investigative case studies uncovered that the current focus of these platforms, similar to the ones described previously, is primarily on the pre-processing and dissemination stages.

However, Jollès et al. [7] analyzed ThreatFox,[12] a free platform for IoCs Sharing similar to VirusTotal and MISP. Their findings revealed that building collaborative cybersecurity on an established network of trust is a crucial dynamic for this kind of platform.

A similar platform called ETIP (Enriched Threat Intelligence Platform) [11] focuses on the collection and processing of structured data sourced from external sources, encompassing OSINT feeds, along with information originating from an organization's network infrastructure. ETIP includes the following components: *(i)* an input module responsible for the collection and standardization of IoCs from OSINT feeds and the monitoring infrastructure; *(ii)* an operational module, which produces enriched IoCs and evaluates threat data using a threat score; and *(iii)* an output module for the presentation of the outcomes and their sharing with external entities to strengthen cybersecurity defenses. This platform eliminates duplicate IoCs, creates composed IoCs, and assigns a threat score to each IoC to help Security Operations Center (SOC) analysts prioritize security incident investigations.

Haque et al. [12] underline the significance of adopting an automated strategy for sharing CTI while emphasizing the effectiveness of Relationship-Based Access Control for facilitating this sharing. Their approach aims to identify, generate, and disseminate structured CTI and implement these concepts through a prototype Automated Cyber Defense System in a cloud-based environment. In response to the challenges related to trust in the source and integrity of threat intelligence data, Preuveneers et al. [13] improved the security framework TATIS [14]. Both TATIS [14] and the framework in [12] guarantee that only authorized individuals can access sensitive data when it is being transferred between various threat intelligence systems. Moreover, in the proposal of [14], encryption is applied using the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) cryptographic scheme to protect the shared data.

Sharing CTI data can greatly improve IT security, but it faces several challenges, such as expenses, risks, and legal requirements. To overcome these issues, Riesco et al. [15] proposed an approach to encourage the sharing of CTI among various stakeholders by leveraging blockchain technology and smart contracts. Their research suggests creating a marketplace on the Ethereum blockchain where participants can exchange CTI tokens as digital assets, thereby incentivizing sharing while addressing potential storage limitations and transaction costs. Menges et al. [16] presented DEALER, a system promoting secure CTI sharing by providing incentives and addressing compliance concerns. Like our approach, DEALER relies on Blockchain technology and an InterPlanetary File System (IPFS) distributed hash table. It employs unbiased quality metrics for reputation assessment and protects buyers and sellers through dispute resolution and cryptocurrency rewards. However, the authors recommended limiting the platform's use to sharing noncritical data, as it may not be suitable for sharing highly sensitive and critical CTI. However, these studies [15,16] do not employ Federated Learning. The BFLS [17] approach ensures the security of CTI data sharing by combining FL for training threat detection models and Blockchain for decentralized aggregation. Specifically, the consensus protocol of the Blockchain is enhanced to filter and select high-quality CTIs for participation in FL [18]. Smart contracts are utilized to automate the aggregation and updation of models, ensuring efficient and secure CTI sharing. Sarhan et al. [19] proposed a Hierarchical Blockchain-based Federated Learning (HBFL) framework for collaborative IoT intrusion detection. The framework ensures secure and privacy-preserved collaboration by leveraging a permissioned blockchain and smart contracts. Moulahi et al. [20] used blockchain technology and FL to safeguard data integrity and aggregation in detecting cyber-threats within Vehicular Ad Hoc Networks (VANET) and Intelligent Transportation Systems (ITS). Their approach involves uploading vehicle-generated models onto a blockchain-based smart contract for aggregation before returning them to the vehicles.

Table 1 provides a summary analysis of the contributions of the existing related works compared to ours. This table shows the key attributes of all the analyzed systems, namely: *(i)* if they provide threat detection, *(ii)* if the threat computation is performed in a collaborative way *(iii)* if the system allow CTI sharing, *(iv)* if it guarantees data privacy, model quality, and participant reputation. As shown by this table and to the best of our knowledge, a complete framework providing secure CTI Sharing still does not exist in the present literature. Indeed, no current frameworks provide the possibility to keep data private during CTI sharing through the collaboration of all the participants, maintaining model quality and assessing participants' trustworthiness at the same time.

3. Background

In this section, we delve into some useful basic concepts to understand the framework described in our paper. In particular, we define the CTI scenario and the phases that compose its lifecycle. Then, we illustrate the fundamental notions of Blockchain. Moreover, we describe

[3] https://www.virustotal.com
[4] https://api.threatstream.com/
[5] https://www.threatq.com/
[6] https://threatconnect.com/
[7] https://otx.alienvault.com/
[8] https://exchange.xforce.ibmcloud.com/
[9] https://go.crowdstrike.com/
[10] https://csirtgadgets.com/collective-intelligence-framework
[11] https://crits.github.io/
[12] https://threatfox.abuse.ch

Table 1
CTI Sharing Systems features.

System	Threat detection	Collaborative computation	CTI sharing	Data privacy	Model quality	Participant reputation
VirusTotal	✓	–	✓	–	–	–
MISP [6]	✓	–	✓	–	–	–
ThreatFox	✓	–	✓	–	–	–
ThreatStream	✓	–	✓	–	–	–
ThreatQ	✓	–	✓	–	–	–
ThreatConnect	✓	–	✓	–	–	–
OTX	✓	–	✓	–	–	–
IBM X-Force Exchange	✓	–	✓	–	–	–
Falcon X Intelligence Crowdstrike	✓	–	✓	–	–	–
CIF	✓	–	✓	–	–	–
Collaborative CRITs	✓	–	✓	–	–	–
Haque et al. [12]	✓	–	✓	–	–	✓
ETIP [11]	✓	–	✓	–	–	–
DEALER [16]	✓	–	✓	–	✓	–
TATIS [14]	✓	–	✓	✓	–	✓
Riesco et al. [15]	✓	–	✓	–	–	–
BFLS [17]	✓	✓	✓	✓	✓	–
HBFL [19]	✓	–	✓	✓	–	–
Moulahi et al. [20]	✓	✓	✓	✓	–	–
SeCTIS	✓	✓	✓	✓	✓	✓

Table 2
List of the acronyms used in the paper.

Acronyms	Description
CTI	Cyber Threat Intelligence
EVM	Ethereum Virtual Machine
FL	Federated Learning
GM	Global Model
IoT	Internet of Things
IPFS	InterPlanetary File System
LM	Local Model
ML	Machine Learning
SC	Smart Contract
SL	Swarm Learning
SMC	Secure Multiparty Computation
TIP	Threat Intelligence Platforms
TTP	Tactic, Technique, and Procedure
ZKP	Zero-Knowledge Proof

the main concepts related to FL and Swarm Learning, their workflow, and such approaches' principal differences and challenges. Finally, we provide details about the Zero-Knowledge Proof and Zero-Knowledge Machine Learning approaches. Table 2 summarizes the acronyms used in the paper.

3.1. Cyber threat intelligence

With the term Cyber Threat Intelligence (CTI), we refer to a set of data regarding security threats, threat actors, exploits, malware, vulnerabilities, and indicators of compromises that can help organizations, governments, and individuals in decision-making for proactive cybersecurity defense [21,22]. CTI data is usually shared in a textual and unstructured form in several online data sources, such as blogs, forums, Online Social Networks (OSNs, for short), or Dark Net Marketplaces. Hence, an iterative process consisting of several phases should be followed to provide valuable insights and transform raw data into actionable intelligence. The main CTI phases can be grouped into six stages [23], namely:

- **Planning and Direction**: phase identifies the main stakeholders and defines the organization's objectives, priorities, and requirements.
- **Data Collection**: phase in which the data sources are identified and CTI data, including IoCs, malware samples, and network traffic logs, are collected through automated tools and manual research.
- **Data Processing**: a phase that involves the transformation and cleaning of raw data into a structured format.

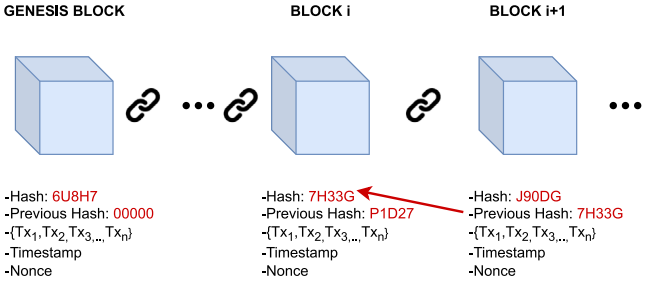


Fig. 1. Example of a Blockchain.

- **Analysis**: phase in which patterns, trends, and potential threats are identified.
- **Dissemination**: phase that consists of sharing data with relevant stakeholders.
- **Feedback**: phase in which the effectiveness of the actions taken and the overall intelligence process are considered to refine and improve future iterations of the CTI lifecycle.

3.2. Blockchain

Blockchain technology refers to a decentralized solution based on a distributed ledger mechanism. It records immutable transactions across multiple parties to provide tamper resistance and security without relying on any centralized trusted third party. This technology has been originally conceptualized as the underlying framework for Bitcoin, the first cryptocurrency, but its possible applications have developed far beyond this initial scenario [24–26]. Transactions, as visible in Fig. 1, are grouped into blocks and are the fundamental units of a Blockchain. Each transaction represents the transfer of value or digital assets from one participant to another.

Each block includes the transactions, the previous block's hash value, a timestamp, and a nonce (a random number for verifying the hash). Due to the presence of a unique hash value, once generated, the information within each block cannot be altered. This ensures the network's immutability. In the first generation of Blockchain technology that handles cryptocurrencies, whenever a new transaction is created, it undergoes validation and verification through a consensus protocol carried out by the *miners*. Miners generate a new block of transactions after solving a mathematical puzzle called *Proof of Work* (PoW) and then propagate that block to the network. Other nodes in the network can validate the correctness of the generated block and only build upon

it if both the transactions included in a new block and of the block itself are considered valid.

Ethereum[13] has emerged as the second generation of Blockchain to allow the building of complex distributed applications beyond the cryptocurrencies through the development of Smart Contracts. A smart contract is an executable code that automatically runs and enforces the terms of an agreement once the specified conditions are met [27]. After being deployed on the Blockchain, the contract operates autonomously, and its code cannot be altered. Usually, it is initiated by activating its constructor function via a transaction submitted to the network. This constructor function is then executed, and the resulting smart contract code is permanently stored on the Blockchain [28]. The execution of the smart contract is validated by a consensus mechanism called *Proof of Stakes* (PoS). Validator nodes with significant cryptocurrency holdings and willing to 'stake' them as collateral are chosen in a fair and transparent manner to participate in block creation and transaction validation. Validators are incentivized by earning transaction fees.

The following Blockchain categories can be defined [29]:

- **Permissioned Blockchains** usually entails a set of participants who must obtain authorization to join the network, perform transactions, and validate blocks. Transactions are grouped, accessed, and verified by a designated group of nodes instead of anonymous miners.
- **Permissionless Blockchains** allow anyone to join and participate without demanding prior authorization. Transaction verification relies on the work of many anonymous miners competing to solve a complex mathematical algorithm for that block of transactions via a trial-and-error approach.
- **Private Blockchains** are restricted to authorized participants. A single entity decides who can join the network and has full authority over the blockchain's management.
- **Public Blockchain** is an open and permissionless network accessible to anyone. Control is shared among all participants through consensus mechanisms. Since transactions are open to the public to verify, the risk of hacking and data manipulation is low even if information privacy can be menaced [29].
- **Hybrid Blockchain** integrates public and private Blockchain elements.
- **Consortium Blockchains**, like Hybrid Blockchains, have private and public features. Access to the network is restricted to predetermined organizations or entities who jointly control the network. Decisions require consensus among the consortium participants.

3.3. Federated learning and swarm learning

Both Federated Learning (FL) and Swarm Learning (SL) are decentralized approaches to machine learning that enable model training across distributed devices without the need to centrally share raw data. Since data is not transferred and centralized, these two methods have advantages regarding privacy preservation and network traffic reduction.

As for FL, the main actors of this protocol are C client devices (or "workers"), owning sensitive data and running local training on them; and a central server (or "aggregator"), that organizes the whole FL process aggregating the local updates. In particular, FL's goal is to train a global model $\mathbf{w}$ by uploading the weights of local models from workers $\{\mathbf{w}^i | i \in C\}$ to the parametric aggregator optimizing a loss function:

$$\min_{\mathbf{w}} l(\mathbf{w}) = \sum_{i=1}^{n} \frac{s_i}{C} L_i(\mathbf{w}^i) \tag{1}$$

[13] https://ethereum.org

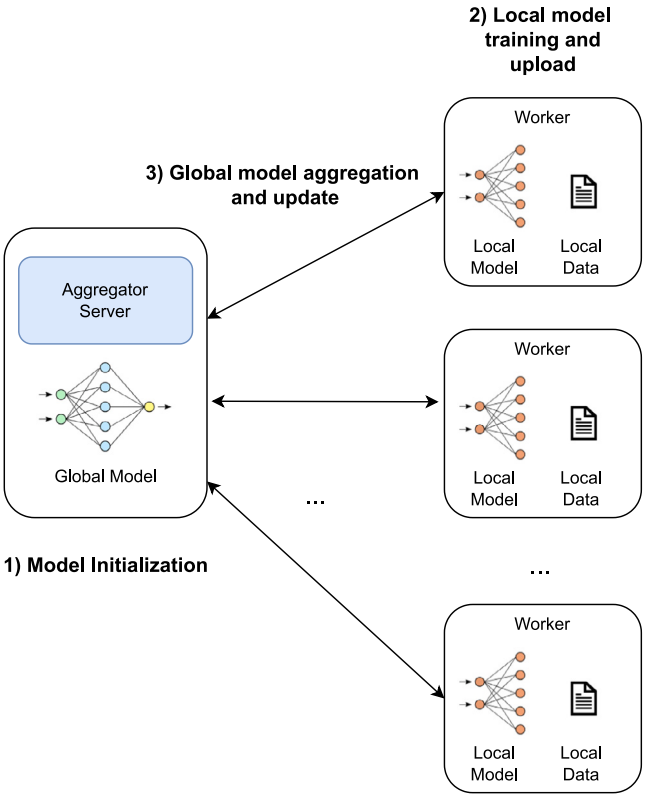


Fig. 2. The Federated Learning workflow.

where $L_i(\mathbf{w^i}) = \frac{1}{s_i} \sum_{j \in I_i} l_j(\mathbf{w}^i, x_i)$ is the loss function, s_i is the local data size of the i-th worker, and I_i identifies the set of data indices with $|I_i| = s_i$, and x_j is a data point.

As visible in Fig. 2, the basic FL workflow consists in the following phases [30]:

1. **Model initialization**: phase in which the aggregator or server sets all the parameters useful for the global ML model $\mathbf{w}$ to their initial status. Moreover, this step also selects the random workers to be included in the process.
2. **Local model training and upload**: phase in which the workers execute local training using their private data after downloading the current global model. Then, each client computes the model parameter updates and sends them to the aggregator or server. The local training typically implicates multiple iterations of gradient descent, back-propagation, or other optimization methods to enhance the local model's performance. Specifically, at the t-iteration, each client updates the global model with the contributions coming from their datasets: $\mathbf{w}_t^i \leftarrow \mathbf{w}_t^i - \eta \frac{\partial L(\mathbf{w}_t, b)}{\partial \mathbf{w}_t^i}$ (where η identify the learning rate and b is local batch).
3. **Global model aggregation and updation**: phase in which the central aggregator collects and aggregates the model parameter updates from all the workers, $\{\mathbf{w}^i | i \in C\}$. The central server can employ different aggregation approaches like averaging, weighted averaging, or Secure Multi-party Computation (SMC) to combine the received updates from each client.

The security and fault tolerance of FL have been increasingly discussed because the central aggregator, keeping model parameters, can be vulnerable to malicious attacks or system failures [31,32]. To address these problems, Swarm Learning (SL) has been recently introduced [8].

As visible in Fig. 3, SL exploits a Blockchain instead of a central aggregator server to securely onboard members and dynamically elect

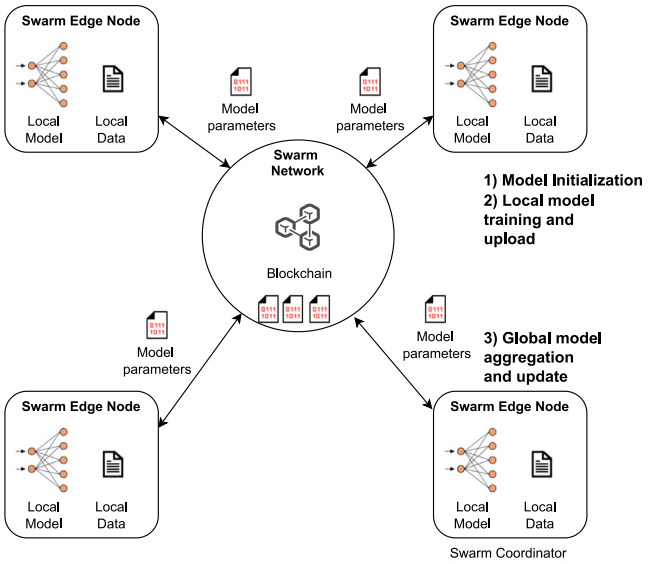


Fig. 3. The Swarm Learning workflow.

the leader. This allows for the performance of DL to be extremely decentralized.

Moreover, it shares the model parameters via the Swarm Network and builds the models independently on private data at *Swarm Edge Nodes*, without the need for a central aggregator. In the workflow of SL, a new edge node enrolls via a Blockchain smart contract, obtains the model, and performs localized training until a given interval. Then, local model parameters are exchanged between participants and combined to update the global model before the next training round. A *Swarm coordinator* can be elected randomly[14] during each iteration and is responsible for maintaining metadata like the model state, training progress, and licenses without model parameters.

3.4. Zero-knowledge machine learning

Zero-Knowledge Proof or Zero-Knowledge Protocol (ZKP, hereafter) is a cryptographic protocol, originally presented in [34], that enables one party (called the *prover*) to prove to another party (called the *verifier*) that a piece of information is true without disclosing the actual details of that statement (without revealing any information beyond the statement's validity).

The main features of the ZKP system include the following properties [35]: *(i) Completeness*, which involves the fact that if the statement is correct, the verifier will always accept it; *(ii) Soundness*, which enforces the fact that if the statement is incorrect, the verifier will always reject it; *(iii) Zero Knowledge* means that no (malicious) verifier can get any extra information from the proof, except the correctness of the statement defined before.

These properties make ZKP widely used in the context of Secure Multiparty Computation (SMC), privacy and authentication. Zero Knowledge Machine Learning (ZKML) is the application of ZKP on ML models in which parameters and operations are concealed from the verifier. The prover can demonstrate the computational correctness of the ML models without disclosing undesired information, promoting transparency and trust. The potential application of ZK to ML could determine that a particular piece of content is produced by applying a specific ML model to a given input [36]. Interestingly, ZKML allows users to specify the desired information included in the proof, such as model parameters, input, output, or none.

[14] Due to deterministic characteristics of blockchains, randomization can be achieved through third-party services such as Chainlink available for public blockchain [33].

4. Proposed approach

In this section, we present a general overview of our approach. The framework aims at secure CTI sharing, allowing participants to train a Global Model (GM, hereafter) collaboratively without revealing confidential information. Furthermore, our reputation approach provides an additional feature, indeed a set of *Validator* nodes is in charge of evaluating the quality of the local model contributions. The reputation mechanism detects low-quality models and prevents them from joining the aggregation. Moreover, the trustworthiness of *Validator* nodes is assessed by Zero-Knowledge Proof in our strategy.

A general architecture of our solution is reported in Fig. 4 with the following actors:

- **Swarm Edge Nodes**. These nodes represent different organizations that are the basic participants of our framework. They are both consumers and producers of shared CTI information, holding private local data and training the local model according to the SL mechanism. They are also called "clients" or "workers" of the SL model.
- **Validator Nodes**. They are chosen among the Swarm Nodes according to their computed reputation (see Section 4.3 for details on the selection strategy). They rank all the local contributions based on reputation score for each iteration to obtain a GM with the best performance. Their trustworthiness is assessed via a mechanism based on ZKP.
- **Swarm Aggregator Node**. A randomly selected node is in charge of aggregating the best local models at each iteration.
- **IPFS**. This distributed File System stores the Global Model and the different Local Models for each iteration.
- **Blockchain**. It is employed to provide a distributed ledger that stores the results of ZKP algorithms and transactions. Moreover, it allows the execution of several Smart Contracts, such as `Coordinator` SC and `Verifier` SC. `Coordinator` SC contains the necessary functions to orchestrate the framework, like storing the verification contract addresses for all the local models at each iteration. It also maintains IPFS model addresses (for both the local and global models), computes trust and reputation scores, and keeps track of the iteration number. `Verifier` SC, instead, receives the proof generated by the validator and produces a response based on its validity.

In practice, our solution comprises three main steps, namely:

1. **Local Models Training**. In this phase, the SL training iterations take place for each client.
2. **Validators Verification**. In this phase, for each iteration, a ZKML algorithm is computed to verify the validators and assess their trustworthiness.
3. **Global Model Aggregation**. In this step, the Global Model is aggregated by a selected node. This Aggregator ranks nodes according to a reputation score and aggregates the top-k model.

In the following sections, we detail each phase of our framework SeCTIS.

4.1. Local models training

This phase starts after the system setup and model initialization step, in which the different organizations register to the Blockchain and join the network.

For each iteration, the workers (i.e., the different organizations) download the Global Model from the IPFS. After that, local training with the CTI private data can be conducted for every client node.

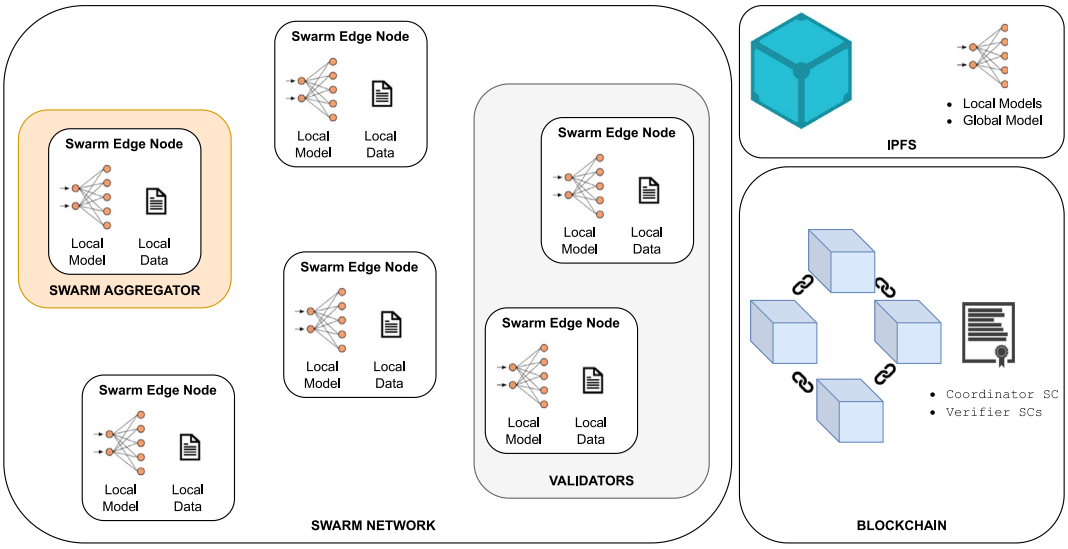


Fig. 4. The General SeCTIS Architecture.

When the Local Model (LM) is computed, it is uploaded to the IPFS by the corresponding node. Moreover, the node submits the hash of the LM that identifies the model itself to the `Coordinator` SC for the subsequent phases. Also, the `Verifier` SC is deployed to the Blockchain, and its contract address is submitted to the `Coordinator` SC.

In summary, the Local Model Training phase comprises the following steps:

1. Download the Global Model from IPFS.
2. Execute local training using their private CTI data.
3. Upload the model into IPFS.
4. Submit the IPFS address (the hash) of their local models to `Coordinator` SC.
5. Deploy the `Verifier` SC to the Blockchain.
6. Get the `Verifier` SC address.
7. Submit the `Verifier` SC address to `Coordinator` SC.

Fig. 5 shows a sequence diagram of all the steps performed during this first phase of the framework.

4.2. Validators verification

In this phase, *Validator* nodes are in charge of testing all the LMs and producing proof of their trustworthiness. At each iteration of the framework, validators are chosen randomly among all the participant nodes. They have to query the `Coordinator` SC and get the hashes of all the LMs. This is to be tested and identify them in the distributed filesystem and download them. After that, they execute their test data on all the LMs to assess the quality of these models. Since the validators can be malicious or malfunctioning and their test data can be corrupted, an immutable proof has to be run for each of them to assess the quality and trustworthiness of these nodes. In particular, each validator generates a Zero-Knowledge Proof containing: *(i)* a digest of the input data batch, and *(ii)* a digest of the model weight. Through this proof, we can assess that the validator's behavior is equal for all the clients. Specifically, this proof verifies that a given validator has tested all the models with the same data points in the same order. After the proof generation, for each tested model, the validator submits these proofs to associated `Verifier` SC. Observe that for each iteration, there is a `Verifier` SC for each participant node. In summary, during this phase of the framework, each *validator* node does the following steps for each iteration:

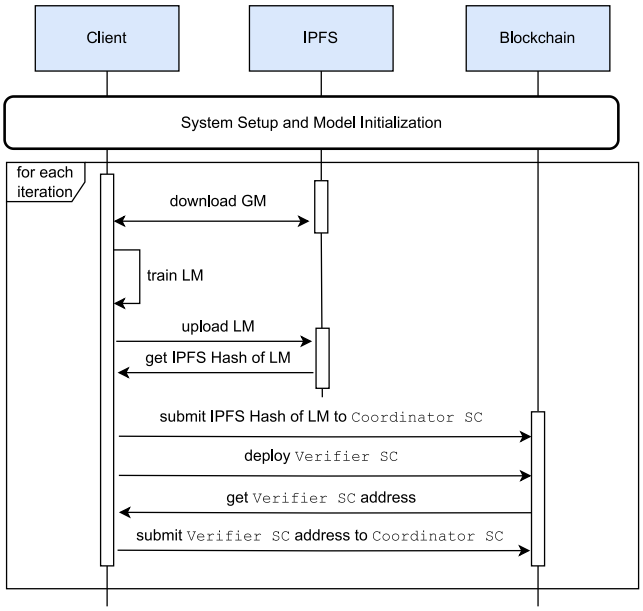


Fig. 5. Local Models Training Sequence Diagram.

1. It queries the `Coordinator` SC and gets the hashes of all the LMs to be tested.
2. It downloads all the LMs from the IPFS.
3. It executes its test data on LMs .
4. It generates the ZKP.
5. It submits the proof along with its results (i.e., the outputs of the tested models) to the `Verifier` SC for each LM update.

Fig. 6 shows a sequence diagram of all the steps performed in the second phase of SeCTIS.

4.3. Global model computation

During the last phase of our framework, a GM is computed by aggregating the local model updates. However, a data quality mechanism is applied to produce a GM that considers only the best local contributions. To do this, as a first step, the `Coordinator` SC randomly selects an Aggregator node among all the participants. This node has

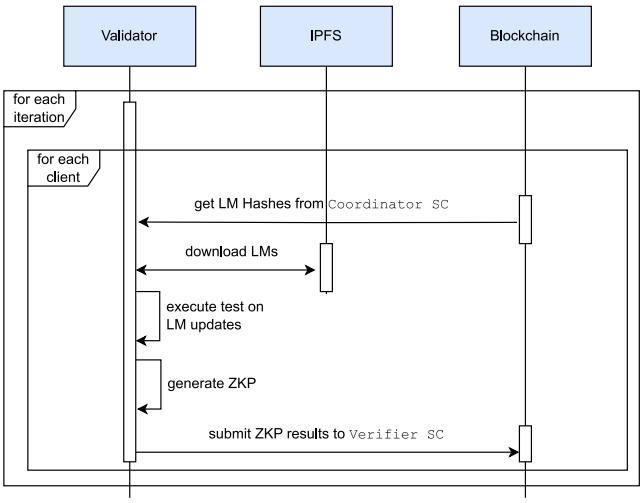


Fig. 6. Validators' Verification Sequence Diagram.

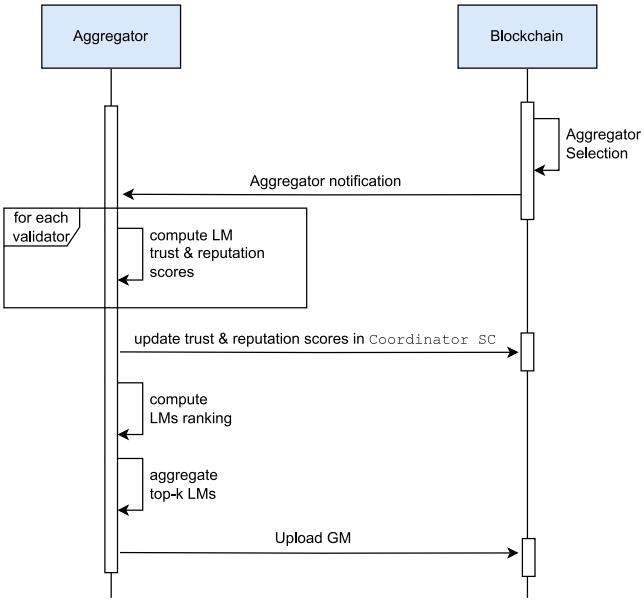


Fig. 7. Global Model Computation Sequence Diagram.

to compute the LMs' trust and reputation scores for all the validators. Then, it submits all these scores to the `Coordinator` SC. After that LMs are ranked and the top-k models are aggregated. Finally, the aggregator publishes the results in the Blockchain.

In summary, the final stage of our framework consists of the following steps:

1. An aggregator is randomly selected by the `Coordinator` SC.
2. The Aggregator collects the outputs of validators for each LM.
3. The Aggregator computes the trust and reputation scores from the outputs that each validator has submitted.
4. It updates the trust and reputation values in `Coordinator` SC.
5. A global rank of LM is computed.
6. The top-k LMs are used to update the GM.
7. The Aggregator publishes the results in the Blockchain.

Fig. 7 shows a sequence diagram of all the steps performed for the Aggregator selection and computation of the GM.

In the following, we define our trust and reputation model to assess the trustworthiness of nodes. These metrics are useful to select the *validator* nodes for the subsequent iteration. Indeed, If a participant's reputation is damaged over a set number of interactions, the organization they represent may be suspended or even removed from the system. Hence, it can no longer be selected as a validator or participate in the framework. The node reputation is computed through the model trust scores for each iteration and is defined as follows.

Given the set of local models $\{m_1, m_2, \dots, m_z\}$, let m_i be the i_{th} local model. For each validator j, test results are deployed into the Blockchain in the form of a $n-tuple$ containing the following information:

1. a digest of the employed test set TS_j with size $|TS_j| = s$,
2. a digest of the verified model $\mathcal{H}(m_i)$,
3. the output of the model for each data point in TS_j.

The output of the i_{th} model is a vector of probabilities $\mathcal{P}_i^d[p_1, p_2, \dots, p_n]$ for each data point d of TS_j, where n is the number of possible classes. This vector contains the probabilities that d belongs to each considered class (or label). The result can be represented as follows:

$$\langle \mathcal{H}(TS_j), \mathcal{H}(m_i), \mathcal{M}_{n\times s} \rangle$$

where $\mathcal{H}$ is the hash function to compute the digest, and $\mathcal{M}_{n\times s}$ is the matrix containing all the probabilities vectors returned by the i_{th} model for the data points of TS_j.

To compute the trust score, we start by defining the average error P_{ij}^k, where i is the i_{th} model, j is the j_{th} validator and k is the iteration, according to the following equation:

$$P_{ij}^k = \frac{\sum_{d\in TS_j} \|\mathcal{P}_i^d - \mathcal{P}_c^d\|}{|TS_j|} \tag{2}$$

where $\mathcal{P}_c^d$ is the centroid of the different output vectors produced by all the local models for the data point d. In practice, we consider each probability vector as coordinates of a point in an Euclidean n-space. Hence, given a point d of the test set TS_j of j, we locate all the outputs produced by the available local models on d in the Euclidean space and identify the most central one as the reference centroid for d. Finally, $\|\mathcal{P}_i^d - \mathcal{P}_c^d\|$ denotes the distance between the output of the i_{th} model for the data point d and the related centroid. Higher values of P_{ij}^k mean that the results of the i_{th} model differ greatly from the average.

Now we define P_i^k as the average value of P_{ij}^k for each validators, for the k_{th} iteration and for the i_{th} model, as:

$$P_i^k = \frac{\sum_{j\in V} P_{ij}}{|V|} \tag{3}$$

where V is the set of all validators, and $|V|$ is its size.

The trust value for the i_{th} model at the k_{th} iteration can be computed as:

$$T_i^k = 1 - P_i^k \tag{4}$$

Finally, the reputation score for the i_{th} model at the k_{th} iteration is represented by the following equation:

$$R_i^k = (1-\alpha)R_i^{k-1} + \alpha T_i^k \tag{5}$$

where α is a weight parameter ranging from [0, 1]. The lower the parameter α the higher the importance of past reputation scores for the given node.

5. Attack model

In our attack model, we focus on possible attacks to the mechanism of assurance of data and model quality rather than addressing common threats like poisoning attacks, backdoors, or denial-of-service (DoS) in Swarm Learning, as discussed in [37,38].

We can identify two main phases in which data can be altered in our framework, namely: *(i)* during the data collection and *(ii)* during the labeling phases. As for the first case, the different organizations usually

gather data from IoT sensors or custom applications. These sources can be compromised or malfunctioning, and data can present errors, be incomplete, or be false. Moreover, the labeling phase, since it has to be carried out manually by some experts, can produce unintentional errors and noisy labels [39]. In our attack model, the only type of malicious attack to model and data quality we consider is *label flipping* attack. The *label flipping* attack is a form of data poisoning where an adversary intentionally mislabels data. This threat aims to corrupting the model's training data and potentially leading to degraded performance or incorrect predictions [40] (see Section 6.2 for further detail on this attack).

Moreover, the *validator* nodes pose potential threats. There is a risk of collusion between certain validators and Byzantine clients [41] - clients containing noisy labels that can behave differently. This collusion enhances the reputation of the Byzantine clients while diminishing that of benign clients. In this attack model, a malicious validator selects a set of verification data tailored to a specific colluding malicious client, as explained in Section 6.2.

To fortify our framework against this form of attacks, we incorporate both the Zero-Knowledge Proof (ZKP) and a reputation mechanism designed to foster consensus among validators. These approaches ensure that the behavior of validators remains consistent across all clients. Additionally, this approach ensures that each validator tests all models using the same data points and in the same sequence, thereby maintaining the integrity of the validation process and mitigating the risk of collusion.

Finally, by combining ZPK, Smart Contracts, and Blockchain technologies, we can guarantee that SeCTIS is secure and can assess the trustworthiness of all the participants of our framework. Hence, classical attacks on trust and reputation systems (such as Slandering, Whitewashing, or Sybil attacks [42]) are implicitly solved by design.

6. Experimental results

In this section, we discuss the experiments carried out to assess the performance of our framework. Specifically, in Section 6.1, we describe the dataset, the evaluation metrics, and the environment used for our experiments. Section 6.2 is dedicated to analyzing the findings and performance of our reputation approach. Lastly, in Section 6.3 we talk about scalability of SeCTIS.

6.1. Testbed description

In our experimentation, we have employed a testbed comprising elements of Swarm Learning, integrating both Federated Learning and Blockchain technologies.

6.1.1. Dataset

In this study, we utilized four distinct datasets to evaluate the proposed Swarm Learning framework for secure CTI sharing. Each dataset represents a different facet of cybersecurity threats, providing a comprehensive testbed for our approach.

- **CIC-Darknet2020:** Darknet traffic frequently encompasses communications linked to malicious activities such as botnet, command and control, malware distribution, and phishing. Analyzing this traffic can help organizations identify patterns and signatures of such malicious activities and proactively defend them. Therefore, we conducted our experiments utilizing the publicly accessible CIC-Darknet2020 [43] dataset by the Canadian Institute of Cybersecurity (CIC). This dataset includes traffic from various categories: Non-Tor, Non-VPN, Tor, and VPN. The dataset contains 93,356 Non-Tor samples, 23,863 Non-VPN samples, 1392 Tor samples, and 22,919 VPN samples.
- **KronoDroid:** The KronoDroid dataset [44] provides both benign and malicious Android applications with features for the detection and classification of mobile malware. This dataset includes 41,382 malicious apps and 36,755 benign apps. Each sample has 200 static and 289 dynamic attributes extracted from apps running on real devices.
- **CSE-CIC-IDS2018:** Anomaly detection helps an organization to identify novel attacks. For the intrusion detection scenario, we employed the CSE-CIC-IDS2018 dataset, which encompasses seven distinct attack scenarios, including Heartbleed, DDoS, Botnet, Infiltration, Web, DoS, and Brute-force attacks. It captures network traffic and system logs from each machine, with 83 features extracted using CICFlowMeter-V3. The dataset is labeled with 14 different attack types, such as DoS Golden Eye, Heartbleed, DoS Hulk, DoS Slow HTTP, DoS Slowloris, DDoS, SSH-Patator, FP, Patator, Brute Force, XSS, Botnet, Infiltration, PortScan, and SQL Injection. As a preprocessing step, we removed features like SrcPort, Flow ID, Timestamp, and IP addresses, as well as duplicates rows. Additionally, we excluded the Benign class label to focus on attacks, and removed the labels SQL Injection, DoS attacks-SlowHTTPTest, and FTP-BruteForce due to their counts being below 100. After excluding certain labels, we have a dataset consisting of 575,364 samples of DDoS attacks-LOIC-HTTP, 198,861 of DDoS attack-HOIC, 145,199 of DoS attacks-Hulk, 144,535 of Bot, 140,610 of Infiltration, 94,048 of SSH-Bruteforce, 41,406 of DoS attacks-GoldenEye, 9908 of DoS attacks-Slowloris, 1730 of DDoS attack-LOIC-UDP, 555 of Brute Force-Web, and 228 of Brute Force-XSS, with a total of 78 features.

To conduct SL and evaluate the model, we partitioned each dataset into an 80:20 ratio, with 80% allocated for collaborative training and the remaining 20% for testing. We carried out our experiments under an Independent and Identically Distributed (IID) data distribution, i.e., the training data is evenly divided among each client. The testing data is solely employed for model evaluation purposes.

6.1.2. Evaluation metrics

We employed various evaluation metrics to assess the impact of the label-flipping attack to evaluate the performance of the proposed framework. These metrics include:

- **Model F1-score**. Due to the imbalanced nature of our datasets, we adopted the F1-score as a metric to evaluate the model's performance. The F1-score considers both precision ($\mathcal{P}$) and recall ($\mathcal{R}$), and it is computed by Eq. (6).

$$F1\text{-score} = \frac{2 \times \mathcal{P} \times \mathcal{R}}{\mathcal{P} + \mathcal{R}} \tag{6}$$

  where $\mathcal{P}$ is the ratio of true positive predictions to the total predicted positives and $\mathcal{R}$ is the ratio of true positive predictions to the total actual positives.
- **Class Transition Misclassification Rate**. The Class Transition Misclassification Rate (CTMR) measures the percentage of instances from the source class, s, that are incorrectly classified as the target, t, class. CTMR is calculated using Eq. (7). A higher CTMR indicates a greater proportion of instances were misclassified.

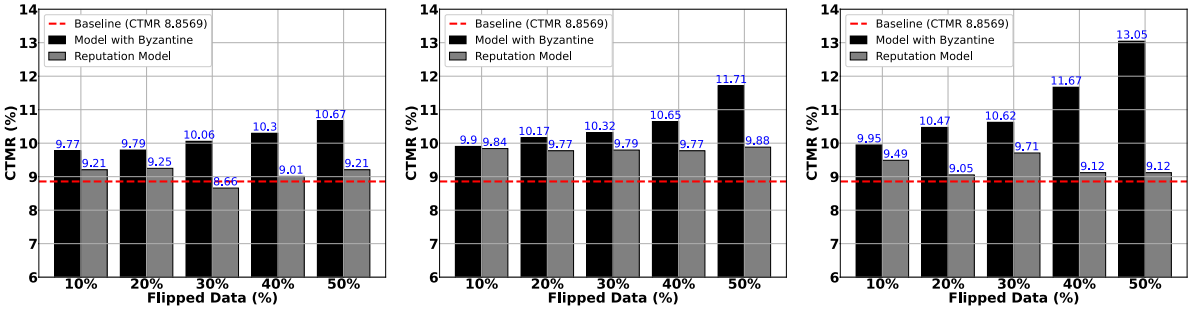
$$\text{CTMR} = \frac{\sum_{j=1}^{N_s} [(y_j == s) \wedge (\hat{y}_j == t)]}{N_s} \tag{7}$$

  where N_s is the total number of instances with the class label s, y_j is the true label of instance j, and $\hat{y}_j$ is the predicted label of instance j.
- **Source Recall**. Source Recall measures the ability of the model to correctly identify instances of a particular class s. A decline in recall for a specific class indicates that an attack has successfully

Table 3
Model architecture settings for each dataset.

Dataset	Layers	η	$\mathcal{B}$	Optimizer	$\mathcal{L}$
CIC-Darknet2020	Hidden: (64, 32), Output: 4	0.01	32	SGD	CrossEntropy
KronoDroid	Hidden: (64, 32), Output: 2	0.001	16	Adam	BCEWithLogits
CSE-CIC-IDS2018	Hidden: (64, 32), Output: 11	0.01	64	SGD	CrossEntropy

where, η represents learning rate, $\mathcal{B}$ indicates batch size, and $\mathcal{L}$ represents loss function.



(a) 10% Byzantine Client (b) 20% Byzantine Client (c) 30% Byzantine Client

Fig. 8. Class Transition Misclassification Rate of CIC-Darknet2020 Dataset.

misled the model, causing it to incorrectly categorize instances of that class. Eq. (8) determines the recall of particular class s.

$$\mathcal{R}_s = \frac{\mathcal{X}}{\mathcal{X} + \hat{\mathcal{X}}} \tag{8}$$

where $\mathcal{X} = \sum_{j=1}^{N_s}[(y_j == s) \wedge (\hat{y}_j == s)]$; $\hat{\mathcal{X}} = \sum_{j=1}^{N_s}[(y_j == s) \wedge (\hat{y}_j \neq s)]$; N_s is the total number of instances belonging to class s, y_j is the true label of instance j, and $\hat{y}_j$ is the predicted label of instance j.

6.1.3. Environment setup

The experiments were conducted on a Windows 11 Pro system featuring an Intel Core i9 processor, 32 GB of RAM, and an NVIDIA Quadro P2000 with 5 GB of GDDR5X memory. We implemented the SeCTIS using the *PyTorch* framework. Additionally, the visualization of results was facilitated by employing the *Matplotlib* library.

In our research, we focused on testing the individual components of our solution on a single machine, utilizing a local blockchain environment with Foundry Forge Anvil [45]. While we did not implement a production-level environment, our approach effectively demonstrates the feasibility and functionality of the proposed system in a controlled setting.

The following pseudocode illustrates the workflow we followed during our experiments:

```
# Dataset split among clients
split_dataset = split_data(dataset,
    num_clients)

# Simulate training on each client
for client_data in split_dataset:
    local_model = train_model(global_model,
        client_data)
    onnx_model = convert_to_onnx(local_model)
    circuit = setup_circuit(onnx_model)
    deploy_smart_contract(circuit,
        local_blockchain)

# Validators process
for validator in validators:
    for model in deployed_models:
        results = validator.run_model(model,
            private_data)
        proof = validator.generate_proof(
            results)
        submit_to_blockchain(proof,
            local_blockchain)

# Aggregator collects results and aggregates
    the selected models
good_models =
    calculate_trust_and_reputation_scores(
    local_blockchain)
aggregated_model = aggregate_results(
    local_blockchain)
next_round(global_model)
```

Listing 1: Pseudocode for SeCTIS

High-level overview of the processes are as follows:

1. **Dataset Distribution**: The dataset is split among multiple clients.
2. **Client Training**: Each client is simulated to train the global model with their own data. The trained models are then converted to ONNX format.
3. **Circuit Setup and Smart Contract Deployment**: Clients set up the circuit for their models and deploy the smart contract to the local blockchain.
4. **Validation**: In a separate loop, validators retrieve the models, run them with their private data points, generate proofs, and submit the results to the local blockchain.
5. **Aggregation**: The aggregator collects the results from the blockchain, performs the necessary calculations(trust and reputations scores), and then aggregates the models, preparing the global model for the next training round.

6.1.4. Model settings

We implemented Deep Neural Networks (DNNs) across all datasets and conducted global model training for 50 rounds. In each round, organizations trained their models for 5 local epochs. The DNN architecture, including the number of layers, neurons, learning rate, batch size, and optimizer, is detailed in Table 3.

6.2. Experimental analysis of reputation model

The data collection and labeling phases of CTI generation within the organization may unintentionally produce noisy labels. This may

potentially obstruct the collaborative learning process. As stated in Section 5, we randomly flipped labels to simulate instances of incorrect labels. These simulations involved $t\%$ of the N participating organization acting as *Byzantine client*. Additionally, to analyze how the affect of noisy labels on the model's performance, we conducted experiments where $x\%$ of samples from specific classes were changed to another class.

Initially, we tested the CIC-Darknet2020 traffic data within a Swarm Learning setup involving 10 participating organizations, excluding Byzantine clients. Our Swarm Learning model for darknet traffic classification achieved an F1-score of 0.928. Subsequently, we conducted experiments with 10%, 20%, and 30% of the total participating organizations as Byzantine clients. Since we have to assess whether our reputation model could identify low-quality models, we designated Byzantine nodes as fixed clients. For the scenario with $t = 10\%$, Client 5 was designated as the Byzantine client. In the case of 20%, Clients 2 and 5 were chosen as Byzantine, while for the 30% setup, Clients 2, 5, and 8 were selected Byzantine. For each scenario, we also conducted experiments with different percentages of label flipping from VPN class to NonVPN, where x ranges from 10% to 50%.

Then, we analyzed the CTMR rate on the test set under two cases: *(i)* when flipping is performed and *(ii)* when the SL model is implemented with the reputation technique. Fig. 8 illustrates the CTMR rates across three different scenarios of CIC-Darknet2020 dataset: with one Byzantine client, two Byzantine clients, and three Byzantine clients. In Fig. 8(a), the horizontal dashed line represents the SL model's CTMR (with a value of 8.8569) without any Byzantine clients. This indicates that approximately 8.9% of the VPN data is misclassified to the NonVPN class within the baseline model. Additionally, as shown in the figure, the misclassification rate increases when the Swarm Learning setup includes one Byzantine client. Also, the misclassification rate rises proportionally with the increased percentage of incorrect labels. Specifically, when 10% of VPN samples are changed to the NonVPN class, the misclassification rate increases to 9.77%. When 50% of VPN class samples are flipped, the misclassification rate reaches 10.67% in the test set. The results for two and three Byzantine clients follow the same pattern as those for one Byzantine client. Figs. 8(b), 8(c) depict a clear increase in the misclassification rate as the number of Byzantine clients increases. For instance, when three Byzantine clients are present in the SL training round, and 50% of samples are changed to another class, the misclassification rate reaches 13.05%.

We also experimented with the same SL scenario on the KronoDroid dataset with the same initial assumptions (10 participating organizations and a maximum of three Byzantine clients). The baseline FL model for the KronoDroid dataset achieves an F1 score of 0.980. To introduce noisy labels, we flipped the labels of malware samples to benign. Fig. 9(a) illustrates the results of the CTMR for the KronoDroid dataset. When no Byzantine clients are present, the SL model for Android malware detection on the KronoDroid dataset produced a misclassification rate of 1.7413%. Similar to the CIC-Darknet2020 dataset, the CTMR increases with the number of flipped samples, as shown in Fig. 9. Moreover, the presence of Byzantine clients significantly affects the CTMR as depicted in Fig. 9(a), 9(b), and 9(c). Specifically, with one Byzantine client, the CTMR increases to 1.86%. When the number of Byzantine clients rises to two, the CTMR further escalates to 2.08%. With three Byzantine clients, the same reaches 2.22%. Moreover, when 50% of the samples are flipped and three Byzantine clients are involved, the CTMR sharply increases to 5.63%, as illustrated in Fig. 9(c).

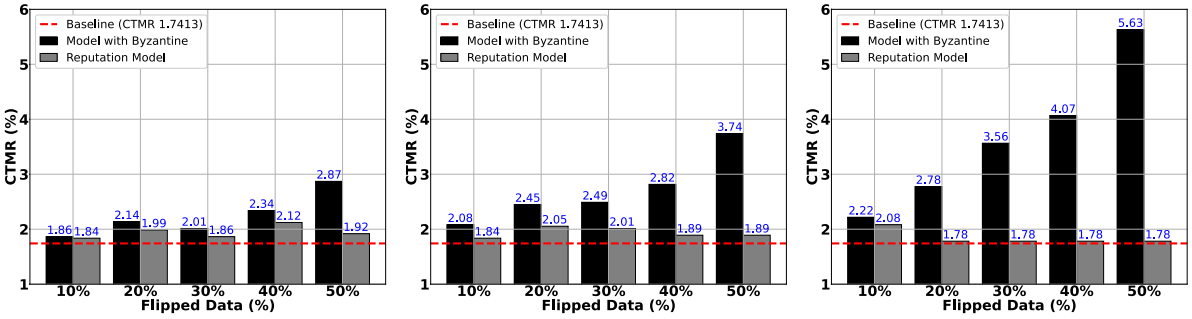
Furthermore, experimentation on the CSE-CIC-IDS2018 dataset achieved an F1 score of 0.999. After testing various combinations of label flipping, we specifically altered samples from the Infiltration class to the DDoS attacks-LOIC-HTTP class. In the baseline scenario, without any Byzantine clients, only 89 out of 28,122 samples were misclassified, yielding an exceptionally low misclassification rate of 0.3165%, as depicted by the horizontal dashed line in Fig. 10. This low misclassification rate likely reflects the dataset's high quality and linear separability under normal conditions. However, when the experiment included Byzantine clients and noisy labels, the number of misclassifications surged to between 128 and 164, resulting in a CTMR of 0.46% to 0.58%. These findings, plotted in Fig. 10, highlight the significant impact of varying attack intensities and the presence of Byzantine clients.

The results of the above experiments demonstrate that noisy labels degrade the performance of the global model and increase misclassification rates. To avoid the inclusion of low-quality models in aggregation, we implemented SL and computed a reputation score for each model using validators, as explained in Section 4.2. Within this framework, we examined two scenarios. First, we assessed the computation of the reputation score under the assumption that all validators are honest within the SL system, even when confronted with noisy labels. Subsequently, we examined the performance of the reputation model when faced with dishonest validators.

6.2.1. Scenario 1: Reputation model with noisy labels and honest validators

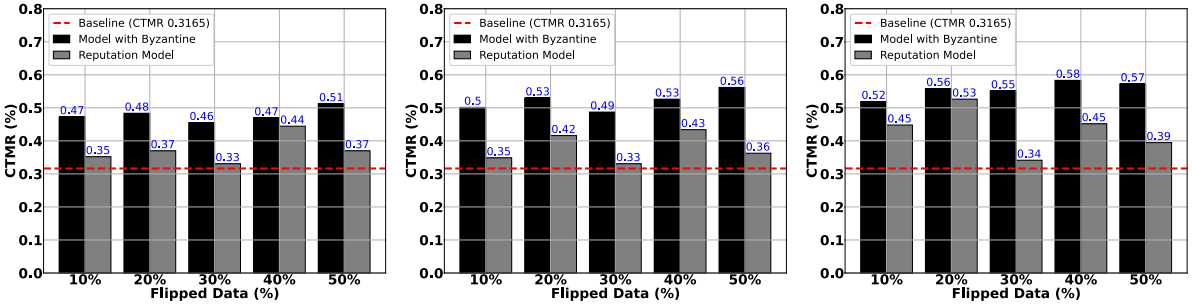
In this, all the validators are honest, and each validator uses the same data to evaluate different models, but different validators use different validation data. Fig. 11 illustrates the reputation scores of 10 participating organizations in each round of our GM training. In this Figure, Model 5, which is highlighted by a dashed line, represents the Byzantine client, while the legitimate models are depicted in solid lines. Also, the five different cases, corresponding to varying percentages of label flipping, are presented in the subfigures. In the initial rounds of SL training, the reputation scores are slightly lower than in subsequent rounds. Specifically, the reputation score of Model 8 is notably low in the first few rounds. However, following the reception of the aggregated model, Model 8 shows improvement, with its reputation score steadily increasing. As the rounds of SL progress, the models converge, and the reputation scores appear to stabilize. Furthermore, it is evident that the reputation scores of Byzantine clients are consistently lower than those of legitimate clients. With an increase in the number of incorrect labels, there is a noticeable divergence in the reputation scores between honest and Byzantine clients. Particularly, when 30%, 40%, and 50% of samples are flipped, a clear distinction emerges, showing the quality of legitimate clients. Similarly, the scenario involving two Byzantine clients shows a identical pattern to that of a single Byzantine client. From Fig. 12, it is evident that both Byzantine models (Model 2 and Model 5) have lower reputation scores than others. Also, when 50% labels of samples are altered, the reputation scores of Byzantine clients range between 0.80 to 0.85, whereas others range from 0.90 to 0.95. The SL system effectively detects and isolates the three Byzantine clients by assigning them low reputation scores. Fig. 13 depicts the reputation scores of each model across multiple GM training rounds containing three Byzantine clients. In this case, Models 2, 5, and 8 are identified as Byzantine, with their reputation scores notably lower than others. The reputation score of Model 8 varies depending on whether it behaves as Byzantine or not. Specifically, with 50% of samples flipped, the reputation score of Model 8 drops to approximately 0.85, whereas it is above 0.90 when not considered as a Byzantine node. Byzantine clients exhibit lower reputation scores than others for 10% and 20% flipping. However, the difference is insignificant due to the limited number of label alterations. This indicates that in scenarios where organizations with a higher proportion of incorrect labels exhibit comparatively lower reputation scores. So, based on the reputation score, we opted to include only 70% of the total participating organizations with high reputation scores for aggregation.

The misclassification rate of SL with reputation, compared to the model without reputation scheme, is illustrated in Fig. 8. Fig. 8(a), 8(b), and 8(c) clearly demonstrate a reduction in the misclassification rate when low-quality models are excluded based on the reputation score. For instance, with three Byzantine clients and 50% of label flipping, the misclassification rate reduced to 9.12% from 13.05%. As the reputation model excludes low-quality models from aggregation, our SeCTIS



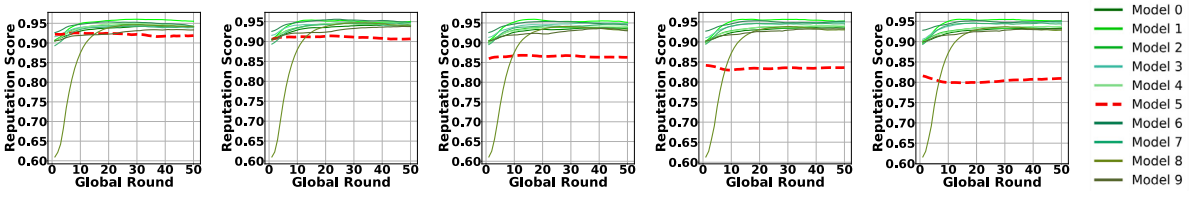
(a) 10% Byzantine Client (b) 20% Byzantine Client (c) 30% Byzantine Client

Fig. 9. Class Transition Misclassification Rate of KronoDroid Dataset.



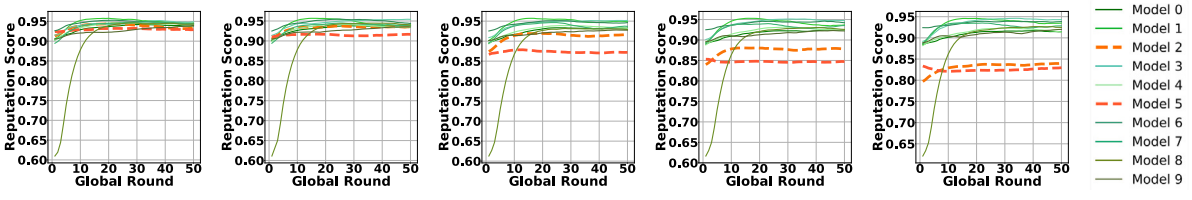
(a) 10% Byzantine Client (b) 20% Byzantine Client (c) 30% Byzantine Client

Fig. 10. Class Transition Misclassification Rate of CSE-CIC-IDS2018 Dataset.



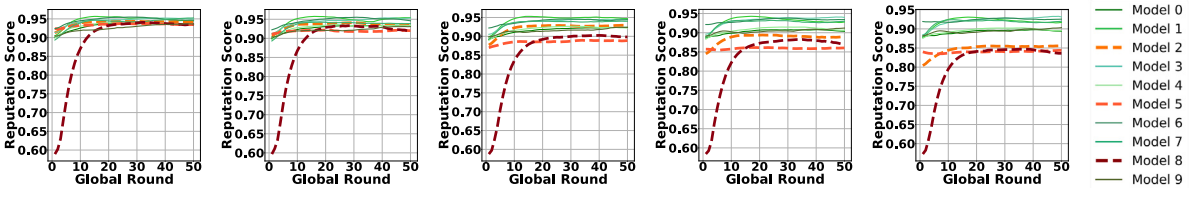
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 11. Reputation scores for each model experimented on CIC-Darknet2020 dataset with 10% Byzantine clients. Byzantine client (Model 5), highlighted in red, contrasts with benign participants represented in green.



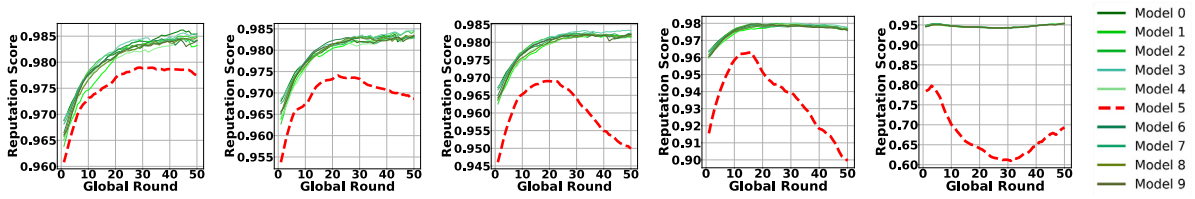
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 12. Reputation scores for each model experimented on the CIC-Darknet2020 dataset across 50 rounds of Swarm Learning involving 20% Byzantine clients. Two Byzantine models (Model 2 and 5) are emphasized in red variant colors, and legitimate participants are highlighted in green.



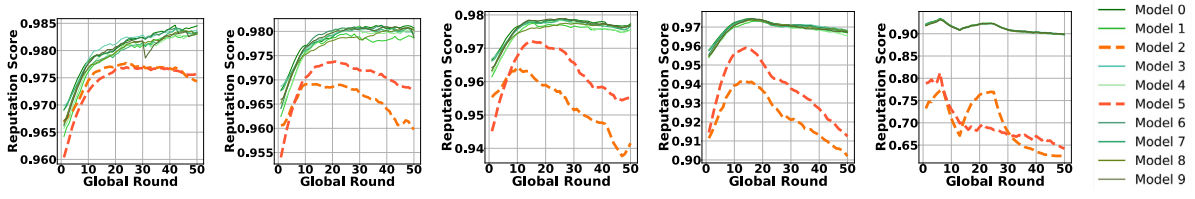
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 13. Reputation scores for each model experimented on the CIC-Darknet2020 dataset across 50 communication rounds with a presence of 30% Byzantine clients. Three Byzantine models (Models 2, 5, and 8) are vivid in red, while benign models are represented in green.



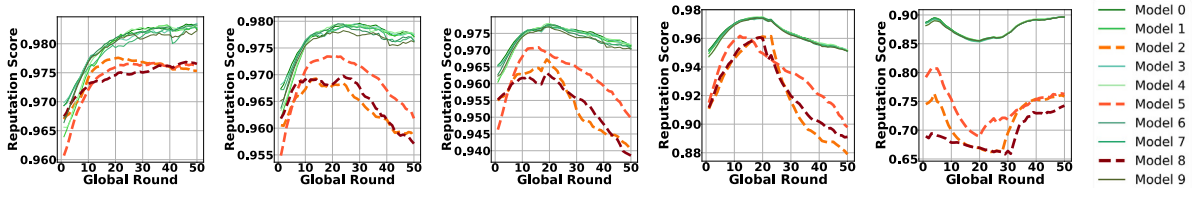
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 14. Reputation scores for each model tested on the KronoDroid dataset with 10% Byzantine clients. Byzantine clients (Model 5), highlighted in red, contrast with benign participants represented in green.



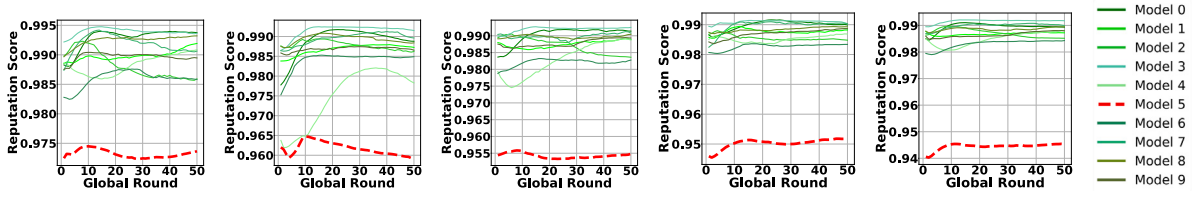
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 15. Reputation scores for each model tested on the KronoDroid dataset across 50 rounds of Swarm Learning involving 20% Byzantine clients. Two Byzantine models (Model 2 and 5) are emphasized in red variant colors, and legitimate participants are highlighted in green.



(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 16. Reputation scores for each model tested on the KronoDroid dataset across 50 communication rounds with a presence of 30% Byzantine clients. Three Byzantine models (Models 2, 5, and 8) are vivid in red, while benign models are represented in green.



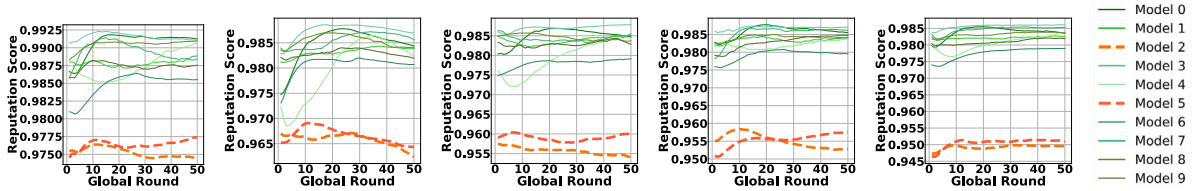
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 17. Reputation scores for each model tested on the CSE-CIC-IDS2018 with 10% Byzantine clients. Byzantine clients (Model 5), highlighted in red, contrast with benign participants represented in green.

framework reduces misclassification rates. Achieving the same results as FL without Byzantine clients becomes challenging. However, in some cases, the misclassification rate reaches the baseline. We also examined the recall of the VPN class under two conditions: when the reputation scheme is not utilized and when SL with reputation is applied, as depicted in Fig. 20. In this figure, the recall of the baseline model is denoted by a dashed line, with a value of 88.4162. The presence of Byzantine clients during collaborative learning will diminish the source recall; for instance, with SL involving three Byzantine clients and 50% of label alterations, the source recall drops to 84.36%. However, with aggregation based on the reputation scheme, the recall consistently improves across all scenarios, closely approaching the performance of the baseline model. Specifically, when three Byzantine clients are present, and 50% of the labels are flipped, our SeCTIS framework enables a significant increase in the recall rate to 88.35%.

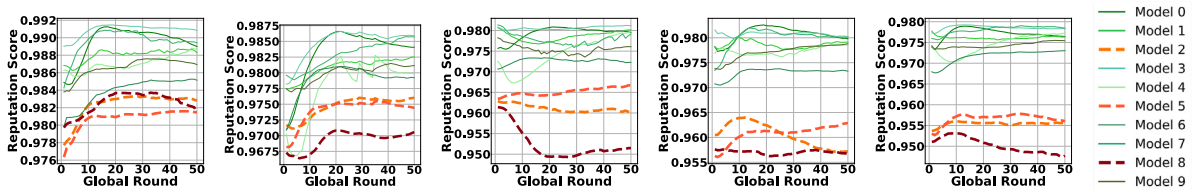
We also evaluated the proposed reputation scheme on the KronoDroid dataset. Similar to the CIC-Darknet2020 dataset, our reputation scheme effectively identifies low-quality models and excludes them from aggregation in the KronoDroid dataset. Figs. 14, 15, and 16 show the reputation scores of each of the 10 models across three scenarios: one Byzantine client, two Byzantine clients, and three Byzantine clients, respectively. In these figures, Byzantine clients are represented by dashed lines, while non-Byzantine clients are shown with solid lines. For the KronoDroid dataset, there is a clear distinction between Byzantine and non-Byzantine nodes. In the presence of one Byzantine client and with 50% of samples flipped, the reputation score of Model 5 ranges from 0.60 to 0.80, whereas non-Byzantine nodes have reputation scores close to 0.95. Similarly, with two and three Byzantine clients, the reputation scores of Byzantine nodes remain low compared to those of non-Byzantine nodes, following the same pattern observed with one Byzantine client.

Based on these reputation scores, we excluded low-quality models from aggregation and then computed the CTMR and source recall, as discussed for the CIC-Darknet2020 dataset. Fig. 9 illustrates the CTMR



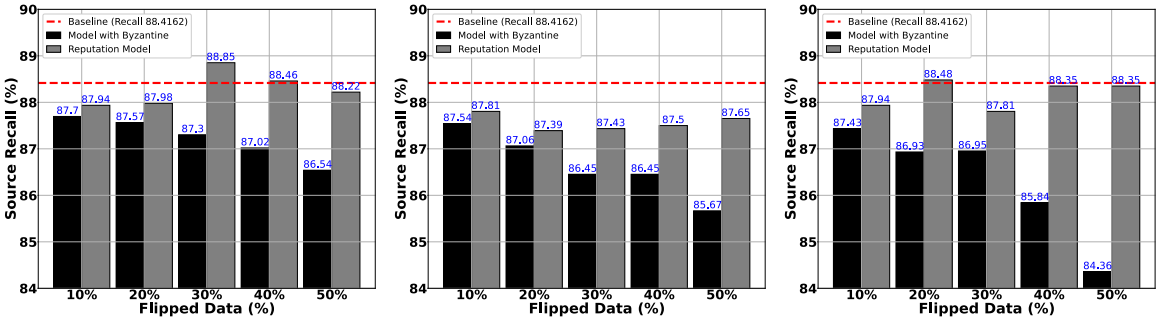
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 18. Reputation scores for each model tested on the CSE-CIC-IDS2018 dataset across 50 rounds of Swarm Learning involving 20% Byzantine clients. Two Byzantine models (Model 2 and 5) are emphasized in red variant colors, and legitimate participants are highlighted in green.



(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 19. Reputation scores for each model tested on the CSE-CIC-IDS2018 dataset across 50 rounds of Swarm Learning involving 30% Byzantine clients. Three Byzantine models (Models 2, 5, and 8) are emphasized in red variant colors, and legitimate participants are highlighted in green.



(a) 10% Byzantine Client (b) 20% Byzantine Client (c) 30% Byzantine Client

Fig. 20. Source Recall of CIC-Darknet2020 Dataset.

when the global model is implemented with the reputation scheme. As shown in Fig. 9, the CTMR is reduced across all scenarios. Specifically, as previously discussed, when 3 Byzantine clients and 50% of the samples flipped, the CTMR is 5.63%. However, after excluding low-quality models, the CTMR is reduced to 1.78%, which is close to the baseline model's performance.

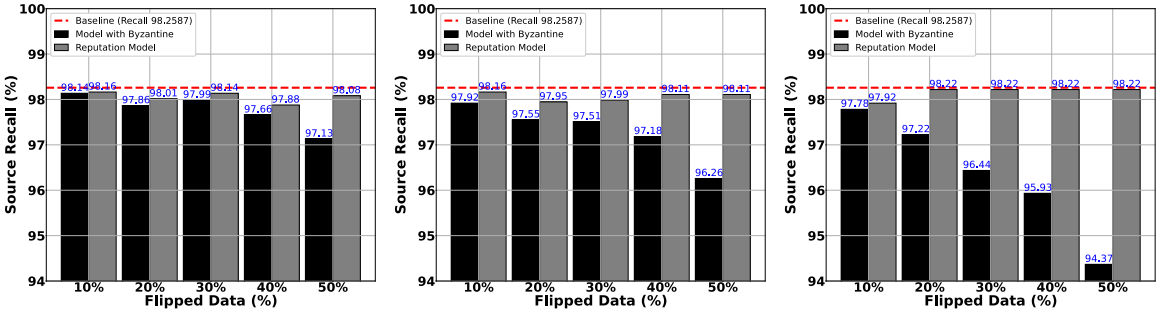
Moreover, Fig. 21 shows that the source recall for the Malware class decreases when noisy labels are introduced. However, when low-quality models are excluded based on the reputation scheme, the source recall increases, approaching the performance of our baseline model. For instance, with three Byzantine clients and 50% of malware samples flipped, the source recall for the Malware class is 94.37%. After removing the low-quality models and creating a global model, the source recall improves to 98.22%, which is very close to the baseline recall of 98.26%.

Similar to the CIC-Darknet2020 and KronoDroid datasets, the CSE-CIC-IDS2018 dataset exhibit the same behavior. The low-quality models in the CSE-CIC-IDS2018 dataset were identified by the reputation score, as illustrated in Figs. 17, 18, and 19. As shown in Fig. 17, which depicts the scenario with one Byzantine client, the reputation score of Byzantine client Model 5 ranges from 0.94 to 0.975, while for the other models, it exceeds 0.975. Similarly, in the scenario with two Byzantine clients (illustrated in Fig. 18), the reputation scores of Models 2 and 5 fall below 0.970, whereas for non-Byzantine clients, the scores are above 0.970. This pattern is also observed in the scenario with three Byzantine clients, as illustrated in Fig. 19. After excluding these low-quality models from aggregation, the CTMR decreases and returns to baseline levels, as shown in Figs. 10. Additionally, source recall improves and aligns with the baseline value of 99.6266%, as shown in Fig. 22. When three Byzantine clients flip 50% of the samples, source recall drops to 99.33%. However, after applying the reputation scheme and excluding low-quality models during aggregation, source recall improves to 99.54%. Our experimental findings show that integrating a reputation scheme into SL effectively reduces the impact of Byzantine clients, leading to lower misclassification rates and improved recall rates. Furthermore, this integration helps to identify and exclude low-quality models during the GM aggregation. Thus enhancing the overall performance and resilience of collaborative learning.

6.2.2. Scenario 2: Reputation model with noisy labels and faulty validators

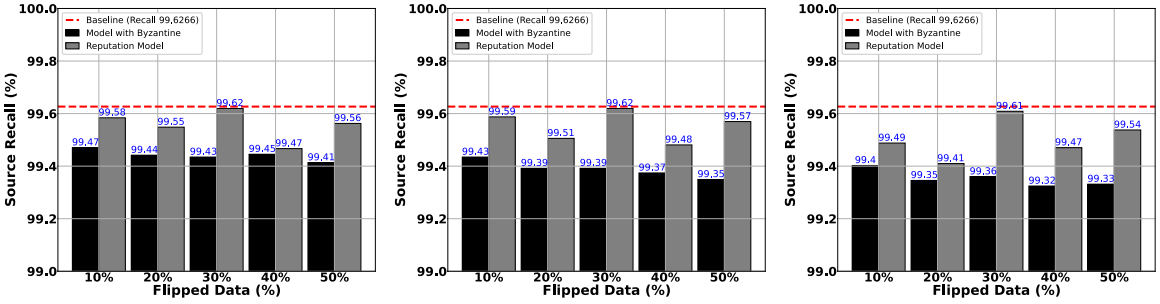
We introduced faulty validators that collude with Byzantine clients to evaluate the effectiveness of our framework in the presence of dishonest validators. Our consensus mechanism is specifically designed to ensure fault tolerance and robustness in these scenarios. It functions effectively as long as a majority of the nodes remain honest and operational. For a system with N nodes with f faulty nodes, the system ensures correctness if at least $2f + 1$ nodes are honest and operational.

In our framework, we assessed scenarios involving a total of 10 validators, which allows the system to tolerate up to three faulty validators. To test this, we simulated a scenario where three faulty validators colluded with Byzantine clients. Specifically, we examined a case with one Byzantine client and three faulty validators–Validators 0,



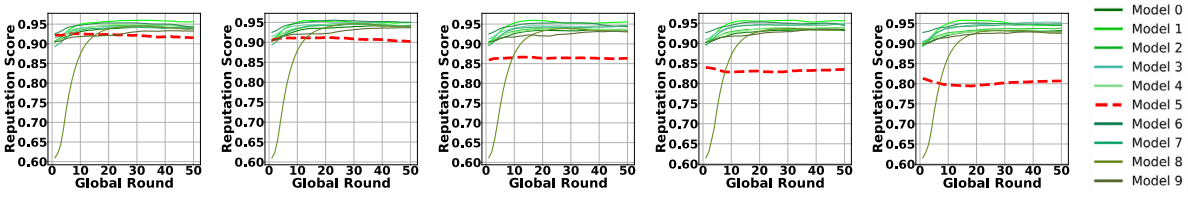
(a) 10% Byzantine Client (b) 20% Byzantine Client (c) 30% Byzantine Client

Fig. 21. Source Recall of KronoDroid Dataset.



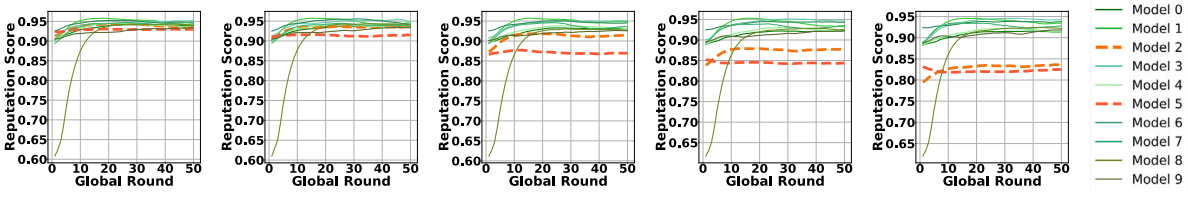
(a) 10% Byzantine Client (b) 20% Byzantine Client (c) 30% Byzantine Client

Fig. 22. Source Recall of CSE-CIC-IDS2018 Dataset.



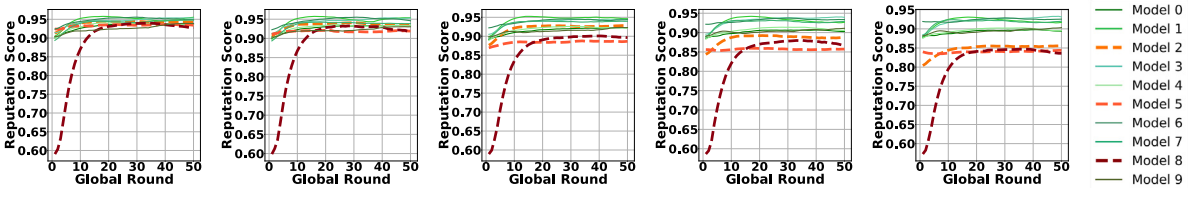
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 23. Reputation scores of each model with 10% Byzantine clients and dishonest validators on the CIC-Darknet2020 dataset. Byzantine clients (Model 5), highlighted in red, contrast with benign participants represented in green.



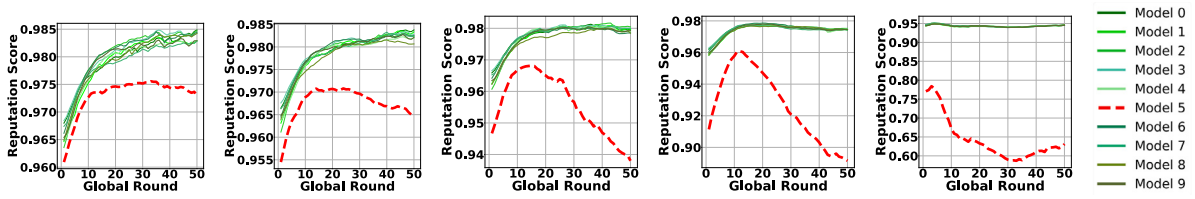
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 24. Reputation scores throughout rounds of Swarm Learning involving 20% Byzantine clients dishonest validators on the CIC-Darknet2020 dataset. Two Byzantine models (Model 2 and 5) are emphasized in red variant colors, and legitimate participants are highlighted in green.



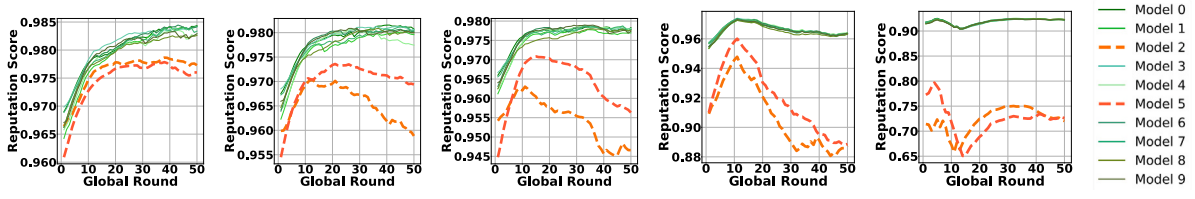
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 25. Reputation scores across communication rounds with a presence of 30% Byzantine clients dishonest validators on the CIC-Darknet2020 dataset. Three Byzantine models (Models 2, 5, and 8) are vivid in red, while benign models are represented in green.



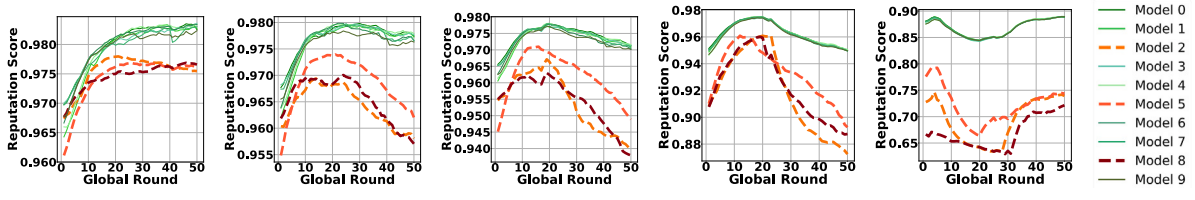
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 26. The reputation scores of all models were evaluated under conditions of 10% Byzantine clients and dishonest validators using the KronoDroid dataset. Byzantine clients (Model 5), highlighted in red, contrast with benign participants represented in green.



(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 27. The reputation scores of all models were tracked during multiple rounds of Swarm Learning, conducted with 20% Byzantine clients and dishonest validators on the KronoDroid dataset. Two Byzantine models (Model 2 and 5) are emphasized in red variant colors, and legitimate participants are highlighted in green.



(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 28. The reputation scores of all models were evaluated throughout multiple communication rounds in a Swarm Learning environment with 30% Byzantine clients and dishonest validators, using the KronoDroid dataset. Three Byzantine models (Models 2, 5, and 8) are vivid in red, while benign models are represented in green.

1, and 5–colluding with Byzantine Client 5 and holding Client 5's noisy data. This setup was designed to artificially inflate the reputation score of Model 5. However, since the majority of validators are honest, the overall system can still identify Model 5 as low quality. The reputation scores calculated in the presence of faulty validators are illustrated in Fig. 23. When comparing Figs. 11 and 23, there is no significant difference in the overall reputation scores of the clients. This consistency is achieved with three faulty validators as the system reaches a consensus on the correct state without exceeding the tolerated threshold.

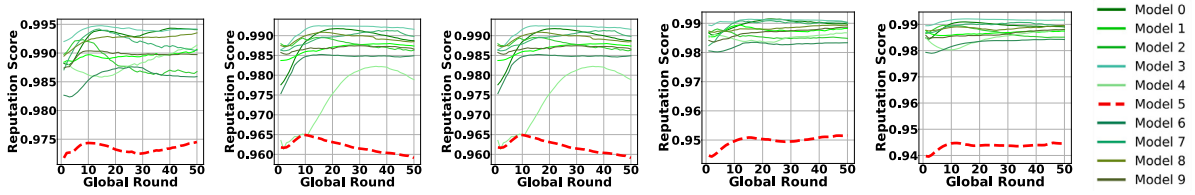
Similarly, we simulated scenarios with two Byzantine clients and three Byzantine clients. With two Byzantine clients, Validators 0, 2, and 5 were faulty; Validators 0 and 5 colluded with Byzantine Client 5, while Validator 2 colluded with Byzantine Client 2. In three Byzantine clients case, Validators 2, 5, and 8 were faulty and colluded with Byzantine Client 2, Byzantine Client 5, and Byzantine Client 8, respectively. In these cases also, we got the same pattern as when all validators are honest, as shown in Figs. 12, 13 and 24, 25. This indicates that our framework can handle up to three faulty validators and discards the Byzantine clients, ensuring reliable consensus and maintaining the system's integrity. We also conducted similar tests using the KronoDroid, and CSE-CIC-IDS2018 datasets. Figs. 26, 27, and 28 illustrates the reputation scores for various cases in the Kronodroid dataset, while Figs. 29, 30, and 31 depicts those for the CSE-CIC-IDS201 dataset. In both cases, our framework successfully identifies low-quality models, even with the presence of up to three faulty validators. In our evaluation, our framework maintains safety and liveliness, ensuring system integrity and responsiveness. Safety mechanisms guarantee correctness and consistency, even when faced with adversarial actions. Despite challenges posed by Byzantine clients and colluding validators, the majority of honest validators consistently identified and rejected malicious data injections. For example, in the simulated scenario featuring one Byzantine client and three faulty validators, attempts to manipulate the reputation score of Model 5 were thwarted by the vigilance of honest validators.

6.3. Scalability of SeCTIS

While an increase in data volume affects the training time of local models in the Swarm Learning setup and, hence, represents a standard scaling issue, it is not a major concern for our solution. SeCTIS is built on top of a Swarm Learning setup and, therefore, its scalability is primarily affected by the model size, i.e., the ONNX model generated by the learners.

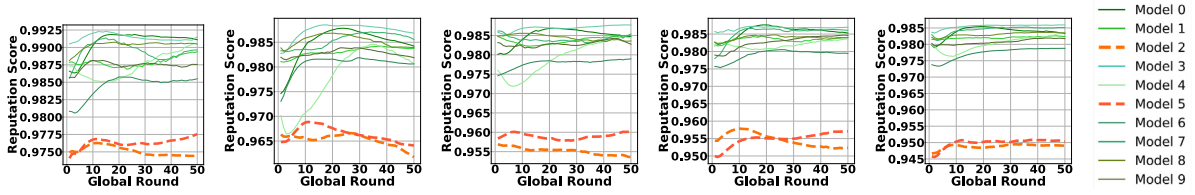
The key factors in our framework's scalability are:

- **Model Size**:
  - **Circuit Creation Time**: as the model size increases, the time required to create the circuit on the learner side increases as well. This affects the initial setup phase and is managed locally by each learner.
  - **Proof Generation Time**: larger models result in longer proof generation times on the validator side.
- **Number of Learners**: the main effect is on the total time a validator spends, as the higher the number of learners the higher the number of executions required to verify the quality of their models. On the other hand, in Swarm or Federated Learning, the number of learners cannot scale exponentially due to inherent system limits like bandwidth, synchronization, and computational constraints, which keep validation manageable [46,47].



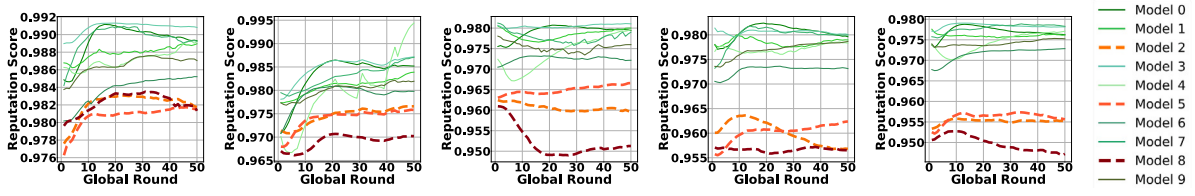
(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 29. Reputation scores of each model with 10% Byzantine clients and dishonest validators on the CSE-CIC-IDS2018 dataset. Byzantine clients (Model 5), highlighted in red, contrast with benign participants represented in green.



(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 30. Reputation scores throughout rounds of Swarm Learning involving 20% Byzantine clients dishonest validators on the CSE-CIC-IDS2018 dataset. Two Byzantine models (Model 2 and 5) are emphasized in red variant colors, and legitimate participants are highlighted in green.



(a) 10% Flipped Data (b) 20% Flipped Data (c) 30% Flipped Data (d) 40% Flipped Data (e) 50% Flipped Data

Fig. 31. Reputation scores across communication rounds with a presence of 30% Byzantine clients dishonest validators on the CSE-CIC-IDS2018 dataset. Three Byzantine models (Models 2, 5, and 8) are vivid in red, while benign models are represented in green.

In light of the reasoning above, the main impacting factor for the scalability of our framework is the model size. For this reason, we performed experiments to assess the performance of our solution by using models of varying sizes. The models used in the experiments have the following architectures:

- **M3:** 3 layers (32 → 16 → 4 neurons);
- **M4:** 4 layers (64 → 32 → 16 → 4 neurons);
- **M5:** 5 layers (128 → 64 → 32 → 16 → 4 neurons).

Our goal was to observe how increasing model complexity affects the key performance metrics in our framework, namely: calibration time, setup time, and proving time. The results, as shown in Table 4, indicate a clear trend: as the model size increases, so do the time and resource demands. For the M3 model, the calibration took 18.56 s, the setup took 52.13 s, and proving took 63.90 s. As we scaled up to the M4 model, the calibration time rose to 26.56 s, the setup to 81.82 s, and proving to 102.17 s. The M5 model further escalated these times to 63.70 s for calibration, 204.54 s for the setup, and 199.02 s for proving.

These results show the increase in computational effort and resource allocation as model complexity grows. Larger models demand more time for calibration, setup, and proving and require larger proving keys, thus showing a direct correlation between model size and resource consumption.

SeCTIS validates the proofs via Blockchain, therefore it is important to discuss the scalability of the Blockchain component of our framework. A large percentage of the time spent on the Blockchain is for the verification of the proofs generated by the validators (Swarm Learning validators). Each validator submits N proofs, where N is the number of models they are testing. As a result, the number of transactions is related to the number of learner nodes involved.

Table 4
Execution times with different model architectures (in seconds).

Model	Calibration time	Setup time	Proving time
M3	18.56	52.13	63.90
M4	26.56	81.82	102.17
M5	63.70	204.54	199.02

In a real-world setting, SeCTIS would be deployed on a public Blockchain like Ethereum. Therefore, following the reasoning above concerning the implicit limitation for the number of learners in a Swarm Learning scenario, as long as each transaction is included in the first block after it is sent, the overall time for all transactions will be approximately 12 s.

7. Security analysis of validator operations

We employed the EZKL library in our framework to implement and manage ZKPs [48]. EZKL is specifically designed to facilitate efficient and secure proof generation and verification, making it an integral component in ensuring the privacy and integrity of our computational processes. This section details how the underlying mechanisms upheld the security guarantees using EZKL. In the following subsections, we will provide a high-level overview of the workflow EZKL in our solution.

Table 5
List of Terms used in ZKP concept.

Elliptic Curve Point	A point on an elliptic curve
Commitment	Cryptographic binding of data to a fixed value
KZG-commitments	A polynomial commitment scheme used in ZKPs [49]
Poseidon Hash	A cryptographic hash function used in ZKPs
Witness	Data used to generate a zero-knowledge proof
SRS	Structured Reference String [50]
Circuit	A set of equality constraints for ZKP computations
EZKL	Python library for zero-knowledge proof systems [48]
HALO2	A recursive Zero-Knowledge Proof protocol [51]
ZKSNARK	A Zero Knowledge Proof System [52]

7.1. Setup phase

This is the initial phase which configures Zero Knowledge solution. The objective of this phase is the generation of a key pair, $\langle proving\ key,\ verification\ key \rangle$, and setting the circuit as done in [53]. The key pair and circuit settings are used in the proof generation and, subsequently, in the verification phases, which are handled by the validators and the `Verifier SC`, respectively. For this objective, based on the aforementioned key pair and circuit setting, the federated learning clients build an Ethereum Virtual Machine (EVM) verifier and deploy it to the blockchain. In our solution, we inherit the functionalities of the EZKL framework and use them to carry out the steps above.

The input to the setup phase includes:

- The local model in the ONNX format.
- The Structured Reference String (SRS) contains all the information to generate and verify the proofs [50,54]. In our framework, we use a global SRS for all ZKP operations across all learner nodes. The SRS is stored in IPFS and its address is available in the `Coordinator SC`.

The outputs of the setup phase are:

- The *proving key* used to generate the proof. This key can either be provided to the validators by a trusted third party or built through a Secure Multiparty Computation performed by the involved organizations [53].
- The *verification key*. `Verifier SC` takes the proof submitted by the validator as input and uses this *verification key* to assess whether the proof is valid or not. The *verification key* is pre-embedded in `Verifier SC` smart contract on the blockchain.
- The circuit settings. The circuit is created based on the EZKL framework. Circuit settings include tolerance, input scale, and lookup range. Tolerance defines the acceptable error margin in computations, while the input scale specifies the scaling factor for input values. Lookup range setting the range of values for lookup operations. By changing these settings, we can fine-tune the performance and accuracy of cryptographic computations.

7.2. Commitment creation

Validators generate proofs using HALO2 and KZG commitments (see definitions in Table 5). These commitments include the input data hash, model hash, and public output. The hash is created using the *Poseidon hash function*, with elliptic curve commitments facilitated by EZKL.

The Structured Reference String (SRS) defined during the Setup Phase includes a generator g and its powers based on a secret scalar s. The SRS ensures the security of the commitments. In our solution, we define the following commitments:

7.2.1. Input data commitment

This represents the input data x to the ONNX model used by the validator. The dataset Poseidon hash is used to build the commitment as follows:

$$C(PH(x)) = g^{PH(x)} \tag{9}$$

7.2.2. Model commitment

Similarly, the commitment for the ONNX model, denoted as M, is defined as follows:

$$C(PH(M)) = g^{PH(M)} \tag{10}$$

The model hash maps the commitment to the target ONNX model.

7.2.3. Public output commitment

The public output y, generated by running the model M on its input data x, is committed as follows:

$$C(y) = g^{y} \tag{11}$$

These commitments bind the data to specific values while keeping them hidden, except for the public model results.

In our solution, a dedicated verifier contract is in charge of checking and detecting the correctness of the proofs submitted by validators. If an attacker attempts to alter the input data, model, or public output, the resulting commitments $C(PH(x'))$, $C(PH(M'))$, and $C(y')$ will be different from the original $C(PH(x))$, $C(PH(M))$, and $C(y)$. Due to the properties of the hash function and the elliptic curve commitments, any deviation will be detected by the verifier:

$$g^{PH(x')} \neq g^{PH(x)},\ g^{PH(M')} \neq g^{PH(M)},\ g^{y'} \neq g^{y}$$

Also, to forge a valid proof using different points, an attacker would need to find consistent polynomial evaluations for points $g^{PH(x')}$, $g^{PH(M')}$, and $g^{y'}$ that correspond to valid inputs. This is equivalent to solving the polynomial evaluation problem at secret points defined by the SRS, which is computationally infeasible. This ensures that any change in the inputs or attempts to generate fake proofs will result in a detectable change in the results, maintaining the binding property. The security of our commitments relies on the computational infeasibility of deriving the secret s from the SRS. Additionally, the Poseidon hash function ensures collision-resistance and preimage-resistance, further securing the integrity of the commitments.

7.3. Proof generation and verification

After the generation of the commitments as explained in Section 7.2, the proof can be generated and verified as follows.

7.3.1. Proof generation

The generation of the proof leverages the *proving key* for the validator generated during the setup phase. As mandated in the HALO2 ZK-SNARK proof system, the prover (i.e., the validator in our scenario) uses the *proving key* to generate a zero-knowledge proof, namely π, which includes the commitments.[15] The proof demonstrates that the prover uses $PH(x)$, $PH(M)$, and obtains y as output.

7.3.2. Verification of proof

To validate the proof π returned by a validator, `Verifier SC` uses the *verification key* to ensure the proof π correctly proves knowledge of $PH(x)$, $PH(M)$, and obtained y, without having access to execution details. This step ensures that the commitments are valid under the HALO2 ZK-SNARK protocol used by EZKL.

[15] We do not report the details about the proof generation as they are part of the basic HALO2 ZK-SNARK framework and, hence, not defined in our solution.

Table 6
Scenario 1.

	Valid	Not valid
Data Hash	cfbacc	0fbacc
Proof Bytes	0 × 10513eb…	0 × 10513eb…
Output	1	The constraint system is not satisfied

Table 7
Scenario 2.

	Valid	Not valid
ECP	30,189; …	40,189; …
Output	1	The constraint system is not satisfied

Table 8
Scenario 3.

	Valid	Not valid
Model Hash	96cbd4d3	06cbd4d3
Proof Bytes	0 × 10513eb…	0 × 10513eb…
Output	1	The constraint system is not satisfied

8. Validation robustness against example attack scenarios

As we stated before, our solution guarantees robust and secure model validation through the use of zero-knowledge proofs. In particular, such proofs certify that each validator has used a defined set of input data, has tested target models, and has generated specific outputs during its validation activity. To demonstrate the efficacy of our solution, in this section, we focus on four possible attack strategies that are derived from the attack model described in Section 5. In particular, we focus on scenarios where the attacker controls some validator nodes and tries to compromise their standard behavior. To achieve this objective, the attacker can try to exploit the following components: *(i)* input data; *(ii)* secure parameters in the Proof; *(iii)* model weights; *(iv)* public output.

Without losing generality, we focus on the case in which the attacker controls a single validator, and we consider four scenarios, namely Scenario 1, Scenario 2, Scenario 3, and Scenario 4. In each scenario, the adversary focuses on one of the above-mentioned components to forge the attack. The adversary objective is to elude the `Verifier SC` so as to jeopardize the estimated trust score for target models. This causes model exclusion (resp., inclusion) in the subsequent aggregation step.

In Scenario 1, a malicious validator tries to deceive the `Verifier SC` by exploiting different input data to generate the public output for a target model. The rationale behind this strategy is to penalize a target model by using special input data to validate it.

As discussed in Section 7, changing the input to the proof will generate a different commitment value (see Eq. (9)). Therefore, our `Verifier SC` will return an error as visible in Table 6.

In Scenario 2, a malicious validator tries to tamper with the proof by changing the Elliptic Curve Points (ECP). However, again, due to the properties of the ZKP Scheme and Hash Function, this action will generate different values with respect to the content of the SRS, thus making the proof invalid. Therefore, once again, the `Verifier SC` will return an error message as visible in Table 7.

In Scenario 3, a malicious validator attempts to use a different model to generate results. This action will cause the violation of the Model Commitment (see Eq. (10)), which will hence make the proof not valid. The output produced by the `Verifier SC` is visible in Table 8.

Finally, in Scenario 4, a malicious validator tries to modify the public output produced by the execution of a target model before submitting it to the blockchain. However, because this forged output is not generated by the given input data hash and model, such modification would violate the Public Output Commitment (see Eq. (11)), which will, hence, invalidate the proof. In such a case, the `Verifier SC` will return the error visible in Table 9.

Table 9
Scenario 4.

	Valid	Not valid
Public Output	0	$1E+63$
Proof Bytes	0 × 1308d6f…	0 × 1308d6f…
Output	1	The constraint system is not satisfied

9. Execution performance for ZKP operations

To provide insights about the execution time of the cryptographic operations included in our EZKL-based solution, we conducted a final experiment to measure the execution times of even distinct actions from two key actors: Validators and Learners. In particular, we focused on the following operations.

1. Calibration of input data. (Learner)
2. Circuit compiling. (Learner)
3. Circuit setup. (Learner)
4. Circuit deployment on the blockchain.(Learner)
5. Witness file generation. (Validator)
6. Proof generation. (Validator)
7. Proof verification on the blockchain. (Validator)

We ran this experiment by using our previously discussed neural network on a 2021 Apple M1 Pro with 8 CPU cores at 3200 MHz and 16 GB RAM. Our analysis found out that, without considering ZKP operations, each client spent approximately 100 s for a global FL round, while each validator spent around 95 s. Table 10 provides a comprehensive breakdown of the time duration associated with the various operations performed during the execution of ZKP processes by both Learner and Validator entities. In particular, for Learner operations, it shows that the most time-intensive task is data calibration, requiring an average of 223.14 s. Conversely, the compilation of the circuit demonstrates negligible duration, taking merely 0.0052 s. Setting up the circuit follows as the next significant operation, consuming an average of 72.89 s. The circuit is deployed, and the final operation is completed within an average time of 12 s. When it comes to validator operations, witness generation is the quickest task, requiring an average of 0.39 s. Conversely, the process of proving is the most time-consuming operation, averaging 83.25 s, followed by the verification process, which takes an average of 12 s.

We are also aware of other zero-knowledge protocols commonly adopted by researchers and developers, such as Orion and Risc0 [55, 56]. According to the EZKL official GitHub repository, in which they compare different frameworks and their benchmark results on various models, EZKL uses 63.95% less memory than Orion and 99.14% less than RISC0. Additionally, it is 2.92 times faster in proving compared to Orion and 77.29 times faster compared to RISC0.

Our implementation ensured that both privacy and efficiency were upheld during the proof generation and verification processes. Through a series of experiments, we validated the effectiveness of our framework while preventing any attempts by dishonest provers to deceive the verifier. In cases where the proof held true, honest provers were able to demonstrate its truth to the verifier. Conversely, attempts by dishonest provers to falsify proofs were effectively thwarted. The cryptographic capability of EZKL's underlying algorithms acted as a robust safeguard against tampering and falsifying proofs, ensuring our framework's integrity and security. Furthermore, EZKL's sophisticated cryptographic techniques obscured the underlying data, preventing any leakage of additional information beyond the truth of the statement during the verification process.

Table 10
Times taken for learner and validator operations.

	Operation	Time (s)
Learner	Data Calibration	223.14 s
	Compile Circuit	0.0052 s
	Setup Circuit	72.89 s
	Deployment	12 s
Validator	Witness Generation	0.39 s
	Prove	83.25 s
	Verify	12 s

10. Discussion

Sharing CTI has been proposed as an effective way to enhance overall cyber intelligence and defense. However, loss of data confidentiality may act as a disincentive to disclose information and impede data sharing. Moreover, various sources of liability may discourage private entities from participating in CTI data distribution. The most commonly cited source of liability is privacy and data protection law, though antitrust law, negligence law, and intellectual property law are also potential concerns if any information they intend to share contains material that is potentially protected [4].

Several features of SeCTIS are intended to address these issues.

1. Data processing is decentralized, hence each participant processes its data on-site without sending it to a central server. The fact that only the learned model parameters (like weights and gradients) are shared reduces the risk of data breaches and ensures compliance with privacy regulations (such as GDPR [57]). Indeed, Intellectual Property (IP) on data, data ownership, and data protection are crucial points for sharing data to train ML models in a distributed way [58].
2. Since SeCTIS architecture relies on Swarm Learning the collaborating entities train a shared model without exchanging their underlying datasets. This ensures that sensitive data remains within its country of origin, avoiding potential legal and ethical challenges associated with cross-border data transfers.
3. Since SeCTIS does not require data to be moved across borders, it allows organizations to collaborate while adhering to their own country's regulations and still contributing to a global model.
4. SeCTIS exploits Blockchain to add an additional layer of security, ensuring that the whole process is transparent, immutable, and tamper-proof. Thanks to this technology a trustless environment where participants do not need to trust each other is also guaranteed and this can incentivize international collaboration and foster a more cooperative and equitable environment.

In summary by keeping data local, *(i)* companies can comply with data sovereignty laws that require data to remain within specific jurisdictions, *(ii)* the risk of non-compliance with data protection regulations is reduced, and *(iii)* the potential legal liabilities and penalties associated with data breaches or unauthorized data sharing are minimized.

11. Conclusion

CTI Sharing provides businesses with access to CTI data that ordinarily they may not have been able to obtain without collaboration with other organizations. This information can be exploited to improve their overall security posture by using the knowledge, experience, and capabilities of the participating entities. This ensures that the detection and previous knowledge of one organization becomes the future prevention of another one. Unfortunately, most organizations are hesitant to share their private CTI data for several reasons, such as the possible loss of credibility, the lack of trust in other peer organizations, possible risks in using external data that may be false or wrong, and strict law regulations.

To provide a contribution in this setting, we propose a complete framework called SeCTIS (Secure CTI Sharing) that aims to tackle different challenges. Firstly, it performs a collaborative training of ML models between different organizations through a Swarm Learning approach. Furthermore, SeCTIS assesses models and data quality through the use of *validator* nodes and the Zero-Knowledge Proof, thus developing a robust reputation model to estimate the trustworthiness of all participants. To evaluate our reputation model experimentally, we introduced byzantine clients with noisy labels and simulated different attack scenario. Our results show that integrating a reputation scheme into Swarm Learning effectively mitigates the influence of noisy labels. Moreover, this integration streamlines the identification and exclusion of low-quality models during aggregation, thereby strengthening the collective performance and resilience of collaborative learning.

To the best of our knowledge, our proposal is novel, and SeCTIS is the first framework to provide complete assurance of data confidentiality, Organization privacy, data, and model quality, and the trustworthiness of participants.

As a limitation of our approach, we identify two main points. First, our framework relies on the EZKL library to generate and verify the required cryptographic proofs. This library allows our framework to run on any blockchain compatible with the Ethereum Virtual Machine (EVM). However, in the current most diffused public blockchains, such as Ethereum, the use of such technology leads to an increase in gas costs for the execution of involved Smart Contracts. On the other hand, opting for a private permissioned blockchain would require additional solutions for basic operations included in our solution. For example, the simple random selection of validators and aggregators among available clients may become problematic due to the lack of third-party services, such as Chainlink, that provide verifiable random numbers. This would require the deployment of additional, possibly costly, solutions. As a second point, we showed that secure operations included in EZKL require relatively high execution times. For both of the points mentioned above, we emphasize that within our application context, which involves (even large) organizations, these limitations seem insignificant. However, they could become impacting if our solution should be extended to other application contexts, such as scenarios in which Internet of Things devices are directly exposed and involved in the learning task.

In the future, we aim to improve our SeCTIS framework by considering also Organization-Specific Threat Intelligence. In particular, we want to help Security Operations Center (SoC) analysts exchange and train ML models to suit a peculiar organization's needs. As for the limitations mentioned above, we are also planning to adapt and test our solution with newer layer 2 blockchains, like Arbitrum, Optimism, and Base, which could allow us to enhance the obtainable performance.

CRediT authorship contribution statement

Dincy R. Arikkat: Writing – original draft, Visualization, Validation, Software, Investigation, Data curation. **Mert Cihangiroglu:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Investigation, Data curation. **Mauro Conti:** Writing – review & editing, Supervision, Conceptualization. **Rafidha Rehiman K.A.:** Writing – review & editing, Writing – original draft, Validation, Conceptualization. **Serena Nicolazzo:** Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Conceptualization. **Antonino Nocera:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Conceptualization. **Vinod P.:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Funded by the European Union

Finanziato dall'Unione europea NextGenerationEU

This work was supported in part by the following projects:

(i) the HORIZON Europe Framework Programme partly supported this work through the project "OPTIMA - Organization sPecific Threat Intelligence Mining and sharing" (101063107). Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or granting authority. Neither the European Union nor the granting authority can be held responsible for them.

(ii) The PRIN 2022 Project "HOMEY: a Human-centric IoE-based Framework for Supporting the Transition Towards Industry 5.0" (code: 2022NX7WKE, CUP: F53D23004340006) funded by the European Union - Next Generation EU.

(iii) The project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor the Italian MUR can be held responsible for them.

Data availability

The authors used publicly available data.

References

[1] Yuchong Li, Qinghui Liu, A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments, Energy Rep. 7 (2021) 8176–8186.

[2] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, Raheem Beyah, Acing the IoC game: Toward automatic discovery and analysis of open-source cyber threat intelligence, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 755–766.

[3] Chris Johnson, Lee Badger, David Waltermire, Julie Snyder, Clem Skorupka, et al., Guide to cyber threat information sharing, NIST Special Publ. 800 (150) (2016) 35.

[4] Livinus Obiora Nweke, Stephen Wolthusen, Legal issues related to cyber threat information sharing among private entities for critical infrastructure protection, in: 2020 12th International Conference on Cyber Conflict (CyCon), vol. 1300, IEEE, 2020, pp. 63–78.

[5] Kealan Dunnett, Shantanu Pal, Zahra Jadidi, Challenges and opportunities of blockchain for cyber threat intelligence sharing, Secur. Trust. Cyber Phys. Syst. Recent Approaches Future Dir. (2022) 1–24.

[6] Cynthia Wagner, Alexandre Dulaunoy, Gérard Wagener, Andras Iklody, MISP: The design and implementation of a collaborative threat intelligence sharing platform, in: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, 2016, pp. 49–56.

[7] Eric Jollès, Sébastien Gillard, Dimitri Percia David, Martin Strohmeier, Alain Mermoud, Building collaborative cybersecurity for critical infrastructure protection: Empirical evidence of collective intelligence information sharing dynamics on threatfox, in: International Conference on Critical Information Infrastructures Security, Springer, 2022, pp. 140–157.

[8] Jialiang Han, Yun Ma, Yudong Han, Demystifying swarm learning: A new paradigm of blockchain-based decentralized federated learning, 2022, arXiv preprint arXiv:2201.05286.

[9] Thomas D. Wagner, Khaled Mahbub, Esther Palomar, Ali E. Abdallah, Cyber threat intelligence sharing: Survey and research directions, Comput. Secur. 87 (2019) 101589.

[10] Clemens Sauerwein, Daniel Fischer, Milena Rubsamen, Guido Rosenberger, Dirk Stelzer, Ruth Breu, From threat data to actionable intelligence: an exploratory analysis of the intelligence cycle implementation in cyber threat intelligence sharing platforms, in: Proceedings of the 16th International Conference on Availability, Reliability and Security, 2021, pp. 1–9.

[11] Gustavo González-Granadillo, Mario Faiella, Ibéria Medeiros, Rui Azevedo, Susana González-Zarzosa, ETIP: An enriched threat intelligence platform for improving osint correlation, analysis, visualization and sharing capabilities, J. Inf. Secur. Appl. 58 (2021) 102715.

[12] Md Farhan Haque, Ram Krishnan, Toward automated cyber defense with secure sharing of structured cyber threat intelligence, Inf. Syst. Front. (2021) 1–14.

[13] Davy Preuveneers, Wouter Joosen, Jorge Bernal Bernabe, Antonio Skarmeta, Distributed security framework for reliable threat intelligence sharing, Secur. Commun. Netw. 2020 (2020).

[14] Davy Preuveneers, Wouter Joosen, TATIS: trustworthy apis for threat intelligence sharing with UMA and CP-ABE, in: Foundations and Practice of Security: 12th International Symposium, FPS 2019, Toulouse, France, November (2019) 5–7, Revised Selected Papers 12, Springer, 2020, pp. 172–188.

[15] Raúl Riesco, Xavier Larriva-Novo, Víctor A. Villagrá, Cybersecurity threat intelligence knowledge exchange based on blockchain: Proposal of a new incentive model based on blockchain and smart contracts to foster the cyber threat and risk intelligence exchange of information, Telecommun. Syst. 73 (2) (2020) 259–288.

[16] Florian Menges, Benedikt Putz, Günther Pernul, DEALER: decentralized incentives for threat intelligence reporting and exchange, Int. J. Inf. Secur. 20 (5) (2021) 741–761.

[17] Tongtong Jiang, Guowei Shen, Chun Guo, Yunhe Cui, Bo Xie, BFLS: Blockchain and federated learning for sharing threat detection models as cyber threat intelligence, Comput. Netw. 224 (2023) 109604.

[18] K.M. Sameera, Serena Nicolazzo, Marco Arazzi, Antonino Nocera, K.A. Rafidha Rehiman, P. Vinod, Mauro Conti, Privacy-preserving in blockchain-based federated learning systems, Comput. Commun. (2024).

[19] Mohanad Sarhan, Wai Weng Lo, Siamak Layeghy, Marius Portmann, HBFL: A hierarchical blockchain-based federated learning framework for collaborative IoT intrusion detection, Comput. Electr. Eng. 103 (2022) 108379.

[20] Tarek Moulahi, Rateb Jabbar, Abdulatif Alabdulatif, Sidra Abbas, Salim El Khediri, Salah Zidi, Muhammad Rizwan, Privacy-preserving federated learning cyber-threat detection for intelligent transport systems with blockchain-based security, Expert Syst. 40 (5) (2023) e13103.

[21] Dave. Shackleford, Who's using Cyberthreat Intelligence and how, SANS Institute, 2015.

[22] Nan Sun, Ming Ding, Jiaojiao Jiang, Weikang Xu, Xiaoxing Mo, Yonghang Tai, Jun Zhang, Cyber threat intelligence mining for proactive cybersecurity defense: a survey and new perspectives, IEEE Commun. Surv. Tutor. (2023).

[23] Marco Arazzi, Dincy R. Arikkat, Serena Nicolazzo, Antonino Nocera, Mauro Conti, et al., NLP-based techniques for cyber threat intelligence, 2023, arXiv preprint arXiv:2311.08807.

[24] Nakamoto S. Bitcoin, Bitcoin: A peer-to-peer electronic cash system, 2008.

[25] Marco Arazzi, Serena Nicolazzo, Antonino Nocera, A novel IoT trust model leveraging fully distributed behavioral fingerprinting and secure delegation, Pervasive Mob. Comput. (2024) 101889.

[26] Marco Arazzi, Serena Nicolazzo, Antonino Nocera, A fully privacy-preserving solution for anomaly detection in IoT using federated learning and homomorphic encryption, Inf. Syst. Front. (2023) 1–24.

[27] Vitalik Buterin, et al., A next-generation smart contract and decentralized application platform, White Paper 3 (37) (2014) 1–2.

[28] Shafaq Naheed Khan, Faiza Loukil, Chirine Ghedira-Guegan, Elhadj Benkhelifa, Anoud Bani-Hani, Blockchain smart contracts: Applications, challenges, and future trends, Peer-to-peer Netw. Appl. 14 (2021) 2901–2925.

[29] Rebecca Yang, Ron Wakefield, Sainan Lyu, Sajani Jayasuriya, Fengling Han, Xun Yi, Xuechao Yang, Gayashan Amarasinghe, Shiping Chen, Public and private blockchain in construction business process and information integration, Autom. Constr. 118 (2020) 103276.

[30] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, Yuan Gao, A survey on federated learning, Knowl.-Based Syst. 216 (2021) 106775.

[31] Milad Nasr, Reza Shokri, Amir Houmansadr, Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning, in: 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 739–753.

[32] Marco Arazzi, Mauro Conti, Antonino Nocera, Stjepan Picek, Turning privacy-preserving mechanisms against federated learning, in: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2023, pp. 1482–1495.

[33] Mudabbir Kaleem, Weidong Shi, Demystifying pythia: A survey of chainlink oracles usage on ethereum, in: Financial Cryptography and Data Security. FC 2021 International Workshops: CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers 25, Springer, 2021, pp. 115–123.

[34] Shafi Goldwasser, Silvio Micali, Chales Rackoff, The knowledge complexity of interactive proof-systems, SIAM J. Comput. 18 (1) (1989) 186–208.

[35] Huixin Wu, Feng Wang, et al., A survey of noninteractive zero knowledge proof system and its applications, Sci. World J. 2014 (2014).

[36] Bianca-Mihaela Ganescu, Jonathan Passerat-Palmbach, Trust the process: Zero-knowledge machine learning to enhance trust in generative AI interactions, 2024, arXiv preprint arXiv:2402.06414.

[37] Kongyang Chen, Huaiyuan Zhang, Xiangyu Feng, Xiaoting Zhang, Bing Mi, Zhiping Jin, Backdoor attacks against distributed swarm learning, ISA Trans. 141 (2023) 59–72.

[38] Ridhima Bector, Hang Xu, Abhay Aradhya, Chai Quek, Zinovi Rabinovich, Poisoning the well: can we simultaneously attack a group of learning agents? in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23, 2023, pp. 3470–3478.

[39] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, Jae-Gil Lee, Learning from noisy labels with deep neural networks: A survey, IEEE Trans. Neural Netw. Learn. Syst. (2022).

[40] Mahdi Abrishami, Sajjad Dadkhah, Euclides Carlos Pinto Neto, Pulei Xiong, Shahrear Iqbal, Suprio Ray, Ali A. Ghorbani, Classification and analysis of adversarial machine learning attacks in IoT: a label flipping attack case study, in: 2022 32nd Conference of Open Innovations Association, FRUCT, IEEE, 2022, pp. 3–14.

[41] Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, Byzantine machine learning: A primer, ACM Comput. Surv. 56 (7) (2024) 1–39.

[42] Kevin Hoffman, David Zage, Cristina Nita-Rotaru, A survey of attack and defense techniques for reputation systems, ACM Comput. Surv. 42 (1) (2009) 1–31.

[43] Arash Habibi Lashkari, Gurdip Kaur, Abir Rahali, Didarknet: A contemporary approach to detect and characterize the darknet traffic using deep image learning, in: Proceedings of the 2020 10th International Conference on Communication and Network Security, 2020, pp. 1–13.

[44] Alejandro Guerra-Manzanares, Hayretdin Bahsi, Sven Nõmm, KronoDroid: time-based hybrid-featured dataset for effective android malware detection and characterization, Comput. Secur. 110 (2021) 102399.

[45] Foundry, Forge anvil: A local ethereum blockchain for testing and development, 2021, https://getfoundry.sh/.

[46] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, Virginia Smith, Federated learning: Challenges, methods, and future directions, in: IEEE Signal Processing Magazine, IEEE, 2020, pp. 50–60.

[47] Jing Wen, Zhaoxiang Zhang, Yu Lan, et al., A survey on federated learning: challenges and applications, Int. J. Mach. Learn. Cybern. 14 (2023) 513–535.

[48] EZKL Documentation, Ezkl documentation, 2024, https://docs.ezkl.xyz/, n.d. (Accessed 6 October 2024).

[49] Aniket Kate, Gregory M. Zaverucha, Ian Goldberg, Constant-size commitments to polynomials and their applications, in: Advances in Cryptology - ASIACRYPT 2010-16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December (2010) 5–9. Proceedings, Springer, 2010, pp. 177–194.

[50] EF Protocol Support Team, Announcing the KZG ceremony. Ethereum foundation blog, 2023, Posted by EF Protocol Support Team on January 16 2023.

[51] Zcash Foundation, Halo2 documentation, 2024, https://zcash.github.io/halo2/index.html, n.d. (Accessed 6 October 2024).

[52] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, Madars Virza, Succinct Non-Interactive Zero Knowledge for a Von Neumann Architecture, Technical Report, 2019, Updated version.

[53] Tobin South, Alexander Camuto, Shrey Jain, Shayla Nguyen, Robert Mahari, Christian Paquin, Jason Morton, Alex'Sandy' Pentland, Verifiable evaluations of machine learning models using zksnarks, 2024, arXiv preprint arXiv:2402.02675.

[54] Mary Maller, Sean Bowe, Markulf Kohlweiss, Sarah Meiklejohn, Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 2111–2128.

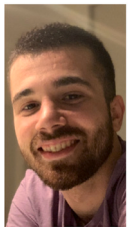
[55] Tiancheng Xie, Yupeng Zhang, Dawn Song, Orion: Zero knowledge proof with linear prover time, in: Annual International Cryptology Conference, Springer, 2022, pp. 299–328.

[56] Erkan Tairi, Pedro Moreno-Sanchez, Matteo Maffei, A 2 l: Anonymous atomic locks for scalability in payment channel hubs, in: 2021 IEEE Symposium on Security and Privacy (SP), IEEE, 2021, pp. 1834–1851.

[57] General Data Protection Regulation GDPR, General data protection regulation, in: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 2016 on the Protection of Natural Persons with Regard To the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC, 2016.

[58] Mauritz Kop, Machine learning & eu data sharing practices, in: Stanford - Vienna Transatlantic Technology Law Forum, Transatlantic Antitrust and IPR Developments, 2020.

Dincy R Arikkat is presently a Full Time Research Scholar at the Department of Computer Applications, Cochin University of Science & Technology, Cochin, Kerala, India. She received a Master's in Computer Applications from Cochin University of Science & Technology. Her main research interests include Cyber Threat Intelligence, Network Traffic Analysis, Machine Learning, and Data Mining.



Mert Cihangiroglu is currently a Ph.D. Student in Computer Engineering at the University of Pavia. He received his BsC in Computer Science at Antalya Bilim University and MsC in Data Science at the University of Pavia. During his studies, he worked part-time in various organizations as a data scientist. His research interests include Data Science, Machine Learning, Blockchain, Zero Knowledge Proofs, Privacy, and Security. He is also the author of one scientific paper in these fields.

Mauro Conti is Full Professor at the University of Padua, Italy. He is also affiliated with TU Delft and University of Washington, Seattle. He obtained his Ph.D. from Sapienza University of Rome, Italy, in 2009. After his Ph.D., he was a Post-Doc Researcher at Vrije Universiteit Amsterdam, The Netherlands. In 2011 he joined as Assistant Professor at the University of Padua, where he became Associate Professor in 2015, and Full Professor in 2018. He has been Visiting Researcher at GMU, UCLA, UCI, TU Darmstadt, UF, and FIU. He has been awarded with a Marie Curie Fellowship (2012) by the European Commission, and with a Fellowship by the German DAAD (2013). His research is also funded by companies, including Cisco, Intel, and Huawei. His main research interest is in the area of Security and Privacy. In this area, he published more than 550 papers in topmost international peer-reviewed journals and conferences. He is Editor-inChief for IEEE Transactions on Information Forensics and Security, Area Editor-in-Chief for IEEE Communications Surveys & Tutorials, and has been Associate Editor for several journals, including IEEE Communications Surveys & Tutorials, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, and IEEE Transactions on Network and Service Management. He was Program Chair for TRUST 2015, ICISS 2016, WiSec 2017, ACNS 2020, CANS 2021, CSS 2021, WiMob 2023 and ESORICS 2023, and General Chair for SecureComm 2012, SACMAT 2013, NSS 2021 and ACNS 2022. He is Fellow of the IEEE, Fellow of the AAIA, Distinguished Member of the ACM, and Fellow of the Young Academy of Europe.

Rafidha Rehiman K.A. is presently an Assistant Professor at the Department of Computer Application, Cochin University of Science & Technology, Cochin, Kerala, India. She holds a Ph.D. in Data security, an M. Tech in Information system security, and a Master's in Computer Applications. She has several research articles published in peer-reviewed Journals and International Conferences. She is also supervising Ph.D. scholars in privacy-preserving data sharing and distributed machine learning. Her research interests include Cryptography, Information Security, and Cyber forensics Analysis. Her research work focuses on Malware Analysis and Cyber Threat Intelligence.

Serena Nicolazzo is currently a Research Fellow (RTDA) at the University of Milan. She got a Ph.D. in Information Engineering at the University Mediterranea of Reggio Calabria in 2017. Her research interests include IoT, Security, Privacy, and Social Network Analysis. She is an Editorial Board Member of Online Social Networks and Media (OSNEM) and she is involved in several TPCs of prestigious International Conferences in the context of Data Science and Cybersecurity. She is the author of about 50 scientific papers. She was a Visiting Researcher at the Middlesex University of London.

Antonino Nocera is an Associate Professor at the University of Pavia. His research interests span over Art ificial Intelligence, Cybersecurity, and Data Science. The results of his work in these domains are collected in about 100 research papers published in prestigious International journals and conferences. He is a member of the DCALab laboratory of the University of Pavia in which he leads a research group, characterized by several international collaborations, focusing on Artificial Intelligence solutions applied to the Cybersecurity domain. He is Associate Editor of Information

Sciences (Elsevier) and of the IEEE Transaction on Information Forensics and Security (T-IFS). Moreover, he is involved in the TPC of many renowned International Conferences focusing both on cybersecurity and artificial intelligence. Furthermore, he is the director of the local node of the University of Pavia for the CINI "Data Science" National Lab and a member of the local node of the University of Pavia for the CINI "Cybersecurity" National Lab.

Vinod P. is presently a Marie Curie fellow at the University of Padua and a Professor in the Department of Computer Applications at Cochin University of Science & Technology. He was a Postdoctoral Researcher at the Department of Mathematics, University of Padua, where he is part of the EU-H2020 TagitSmart project. He was a Postdoctoral researcher at Malaviya National Institute of Technology, India, under the ISEA project on Mobile Security. In 2020, he was awarded the Seal of Excellence for a Marie SkłodowskaCurie Individual Fellowship by the European Commission. His area of interest is Adversarial Machine Learning, Malware Analysis, Data Mining, and NLP.