applied system innovation

MDPI

Article

Beyond Histotrust: A Blockchain-Based Alert in Case of Tampering with an Embedded Neural Network in a Multi-Agent Context

Antonio Pereira [1], Dylan Paulin [2] and Christine Hennebert [2,*]

[1] IMT Atlantique, Campus de Brest, F-29280 Plouzané, France; antonio.pereira@imt-atlantique.net
[2] CEA, Leti, Univ. Grenoble Alpes, F-38000 Grenoble, France; dylan.paulin@cea.fr
* Correspondence: christine.hennebert@cea.fr

Abstract

An intrusion into the operational network (OT) of a production site can cause serious damage by affecting productivity, reliability, and quality. The presence of embedded neural networks (NNs), such as classifiers, in physical devices opens the door to new attack vectors. Due to the stochastic behavior of the classifier and the difficulty of reproducing results, the Artificial Intelligence (AI) Act requires the NN's behavior to be explainable. For this purpose, the platform HistoTrust enables tracing NN behavior, thanks to secure hardware components issuing attestations registered in a blockchain ledger. This solution helps to build trust between independent actors whose devices perform tasks in cooperation. This paper proposes going further by integrating a mechanism for detecting tampering of embedded NN, and using smart contracts executed on the blockchain to propagate the alert to the peer devices in a distributed manner. The use case of a bit-flip attack, targeting the weights of the NN model, is considered. This attack can be carried out by repeatedly injecting very small messages that can be missed by the Intrusion Detection System (IDS). Experiments are being conducted on the HistoTrust platform to demonstrate the feasibility of our distributed approach and to qualify the time required to detect intrusion and propagate the alert, in relation to the time it takes for the attack to impact decisions made by the AI. As a result, the blockchain may be a relevant technology to complement traditional IDS in order to face distributed attacks.

Keywords: blockchain; smart contract; bit-flip attack; embedded AI; weight tampering; intrusion detection system

Academic Editor: Andrzej Białas

Received: 16 October 2025
Revised: 10 December 2025
Accepted: 17 December 2025
Published: 8 January 2026

1. Introduction

The report named *The Cost of a Data Breach* [1], sponsored by IBM, has shown that, on average, companies take more than 100 days to identify and more than 50 days to contain a data breach. This is the case in a highly monitored environment, such as the internet. But when it comes to cyber–physical systems, the scenario can be really different. Attacks such as the one Ukraine suffered in 2015, leading to a blackout [2], occurred with their systems being infected for over 6 months without detection. With regard to AI systems, there is a lot of talk about Large Language Models (LLMs). Recent studies have shown that some of the latest GPUs available and commonly used for this purpose are vulnerable to bit-flips [3,4]—a type of attack where protected bits in memory are flipped using techniques like the Row Hammer Attack (RHA) [5–7], voltage glitching, or fault injection, with the

purpose of tampering with the internal weights of the neural network. Even though this type of attack mostly targets cloud-based infrastructures, the same idea also applies to embedded systems, which can also induce unpredictable behavior.

1.1. Motivation: AI in Embedded and Industrial Systems

As embedded systems become increasingly powerful, embedded AI models are being deployed in a wide range of domains and entrusted with high-impact decision-making for image recognition, time series prediction, and so on. However, this adoption needs extra precaution due to a lack of transparency and unpredictable behavior.

In order to overcome these issues, certain guidelines [8] have already been made, and some studies have been conducted with the aim of improving AI robustness and traceability. HistoTrust, for example, applies secure hardware along with blockchain technology to audit AI behavior, bringing trustworthy tools to the manufacturing line use case [9,10]. Being a sequel to these two past projects, we aim to develop a new complementary methodology, enabling the deployment of secured embedded AI models in multi-stakeholder industrial contexts.

While these tools provide explainability to AI-driven systems under normal conditions, by themselves alone, they do not protect against silent parameter tampering. It's also important to find ways to detect possible anomalies occurring in these systems. In this paper, we explore a novel approach that uses blockchain technology as a security enabler and, especially, as a distributed alerting mechanism to propagate alerts of AI model tampering.

1.2. Security Challenges in Embedded AI Environments

The increasing adoption of connected embedded systems in Industry 4.0 has created several security challenges to be solved, even more so with the presence of several independent actors cooperating in given ecosystems. Furthermore, the integration of AI models into physical devices requires the implementation of security measures that take into account the resource constraints of embedded systems. Recent studies have shown that attacks on the weights of an NN model, such as bit-flips, can reduce accuracy, misclassify critical inputs, or even create backdoors in AI models by flipping vulnerable weight values [4,11].

Even though this type of attack has already been studied for years, and most of the focus has been aimed at x86 architectures in cloud-based infrastructures, it is also possible to exploit these vulnerabilities in ARM-based processors [12]. This boosts the search for robust solutions that may help prevent and detect these types of threats, especially in the industrial field. Companies have already suffered various types of targeted attacks on their embedded applications in recent years [13,14], and now, with the spread of AI-based systems, they could be even more affected.

1.3. Overview of the Proposed Approach

To address these challenges, we propose an approach that exploits a blockchain infrastructure already present, along with secure hardware and software, in order to generate alerts propagated across a distributed network. The core idea is that each device continuously verifies the integrity of its NN model by computing the digest of the memory area storing the model at each inference, and checks in a TrustZone environment if the digest is correct. In case of tampering, a transaction is sent to a smart contract to trigger an alert. Instead of relying on a centralized server to collect and forward these alerts, we use a blockchain network as a decentralized publish–subscribe medium using Solidity events in order to reach the peer physical devices directly.

1.4. Research Questions and Article Organization

The research conducted as part of this study addresses the following questions:

- **Q1:** How to rely on hardware security components to detect tampering with the embedded NN model?
- **Q2:** How quickly does a bit-flip attack impact the decision of an embedded classifier?
- **Q3:** Despite its latency, is the use of blockchain relevant for spreading an alert?
- **Q4:** How has blockchain been integrated into intrusion detection frameworks for cyber–physical and embedded systems, and what roles does it play in such architectures?

To answer these questions, the remainder of this paper is structured as follows. Section 2 describes the industrial use case considered in the context of our study. Section 3 provides background on security in embedded AI, detection, and prevention mechanisms to deal with threats. Section 4 introduces the blockchain concepts and shows how a blockchain infrastructure, deployed in addition to existing elements, enhances security. Section 5 then articulates the event-driven intrusion detection concept. Section 6 presents the embedded design based on hardware security components. Section 7 proposes an implementation scenario and a discussion on the speed of the bit-flip attack based on the attacker's knowledge. Finally, Section 8 concludes the paper by summarizing our contributions.

2. Description of the Use Case

The use case studied is in the context of a digitized industry, also known as Industry 4.0, where several robots and automatons cooperate on different operational tasks on a production line [15].

2.1. HistoTrust Use Case

In a previous paper [9], HistoTrust is presented as a solution for providing trust and fairness within an ecosystem of actors involved in a factory's production process.

On the production line, many robots and automatons are performing planned tasks and increasing productivity. These devices often embed artificial intelligence, which has been trained to understand the context in which the robot is evolving and provide support and decision-making assistance to control the actuators and carry out the required task appropriately. The robots are designed, trained, manufactured, maintained, and supervised by independent companies, generally external to the factory. As a result, when an incident occurs on the production line, it can be difficult to find the cause and, most of all, to establish who is accountable for the robot's operation. Artificial intelligence is software, possibly embedded, that is not legally accountable for anything. Moreover, the behavior of an embedded classifier may be difficult to reproduce. In this multi-agent context, as depicted in Figure 1, HistoTrust provides each ecosystem actor with the means to verify the authenticity of operations performed by robots and/or automatons on the production line.

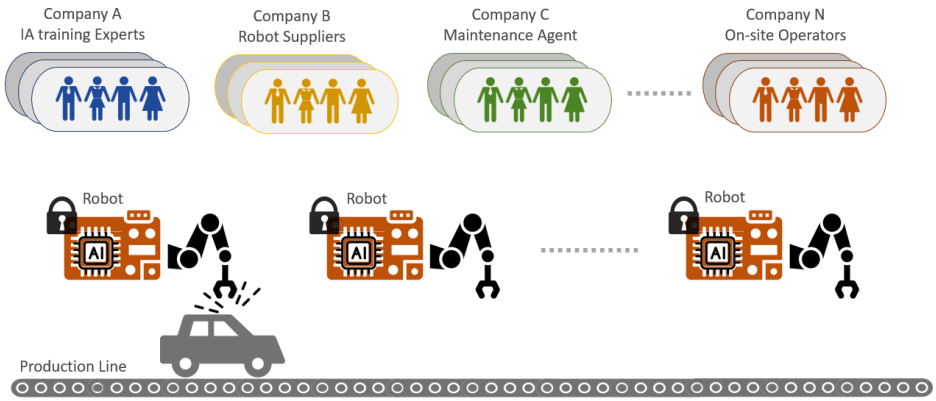

Figure 1. Endorsement of the tasks performed by the robots.

To achieve this, the logs of the performed operations are stored securely and kept by their owner, while the evidence of the authenticity of these logs is registered in a transparent ledger shared among all the actors via a blockchain. The blockchain provides a decentralized means of linking devices and the people involved.

However, a misbehavior of an embedded classifier may be caused by a software or hardware attack, and not by human error. The present paper deals with this use case and introduces a solution for detecting the tampering of an embedded NN model, and using the blockchain to propagate the alert to all connected devices and people, thereby strengthening transparency, trust, and fairness between the players in the ecosystem.

2.2. Architecture of an Industrial Network

An industrial network is usually composed of an Information Technology (IT) network and an Operational Technology (OT) network separated by firewalls, as depicted in Figure 2. The IT network includes computers and servers for administrative, management, and communication tasks. It is connected to the World Wide Web. The OT network includes SCADA (Supervisory Control And Data Acquisition), which is an Industrial Control System (ICS) consisting of computers, networked data communications, and graphical user interfaces for high-level supervision of equipment and processes. It also includes Intrusion Detection System capabilities and Programmable Logic Controllers (PLCs), which interface with the robots acting on the production line.

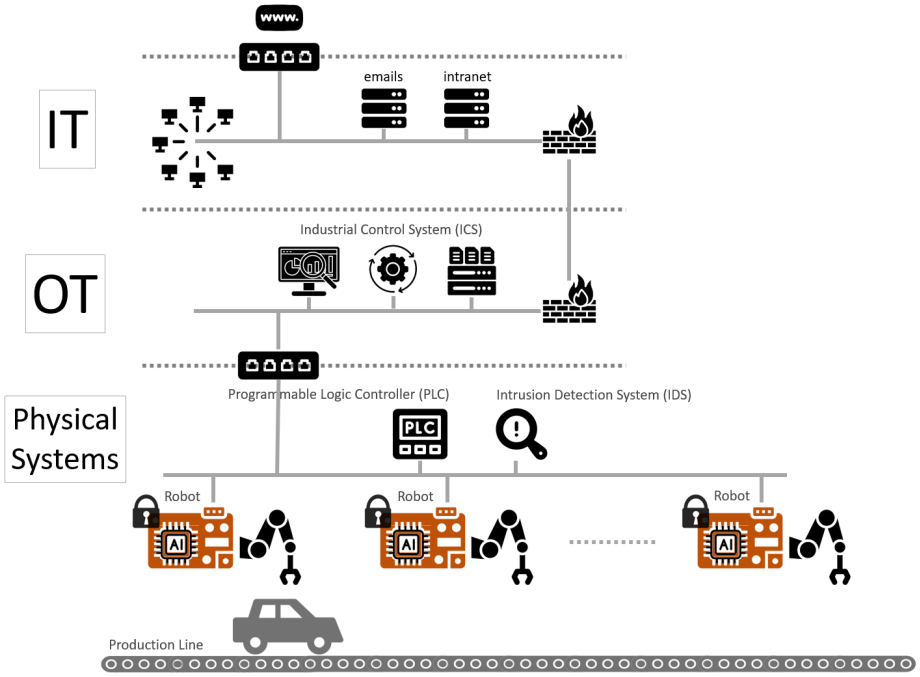

Figure 2. Overview of an industrial network.

The OT network is not directly connected to external networks or the World Wide Web in order to isolate the internal local network that interconnects the equipment, and, in particular, the devices that control the actuators. Thus, an attack on the OT network may be launched either from the local internal network, i.e., by a person present in the factory and with access to this network, or from outside with resources large enough to penetrate firewalls. This latter type of attack consists of several stages, from gaining access primarily to the IT network and then escalating the permissions [16] into the OT network using various techniques, before exfiltrating and/or injecting data while remaining undetected by control systems.

2.3. Recent Attacks on Industrial Sites

Recently, several industrial sites have been the target of coordinated attacks that have affected actuators or halted production. The reports [13,14,17,18] provide feedback and detail the modus operandi of these attacks, to help protect against them in the future.

Attacks such as Stuxnet [17], commonly known as the first to specifically target ICS, began their sequence of infections mainly through USB devices. Once infected, the attackers began to expand their privileges through zero-day vulnerability exploits, reaching the OT network and damaging approximately 1000 centrifuges at an Iranian nuclear enrichment facility.

In the case of the TRITON attack [18], the target was a Safety Instrumented System that is the last line of automated safety defense for industrial facilities. The attackers started by gaining access to the IT network and then moving to the OT network through systems without permissions. Once in the OT network, the attack infects the engineering workstation, usually situated in an isolated network segment. As a result, systems from a Middle Eastern oil and gas petrochemical facility have entered safe mode, which resulted in an unexpected halt of the plant's operation.

2.4. Attacker Profile, Feared Events, and Targeted Attack

In the following, we consider that the attacker has gained access to the OT network. It involves remote access that penetrates the IT network and passes through firewalls, or physical access to the OT network in the factory.

The feared event considered in this paper is a bit-flip attack on an embedded NN model, which causes an unexplainable malfunction of the robots' actuators. Tampering with the NN model has the effect of misleading the classifier.

Thus, we will simulate bit-flip attacks on the weights of the embedded Neural Network (NN) learned for HistoTrust to recognize digits. The attack is illustrated in Figure 3. It may be the consequence of a RowHammer attack or a software injection. Laser hardware injections can also cause bit inversions, but this attack is difficult to carry out on an industrial site, as the equipment has to be disassembled to gain access to the electronic chips. The attack will be simulated using a small script, reverting only one bit at each injection. The script should be as short as possible in order to disappear under the radar of network traffic monitoring systems. On the other hand, the rate of occurrence of this script will vary, successively affecting several bits of the NN model. Therefore, we assume that the attacker knows the address range of the NN model in the device's flash memory.

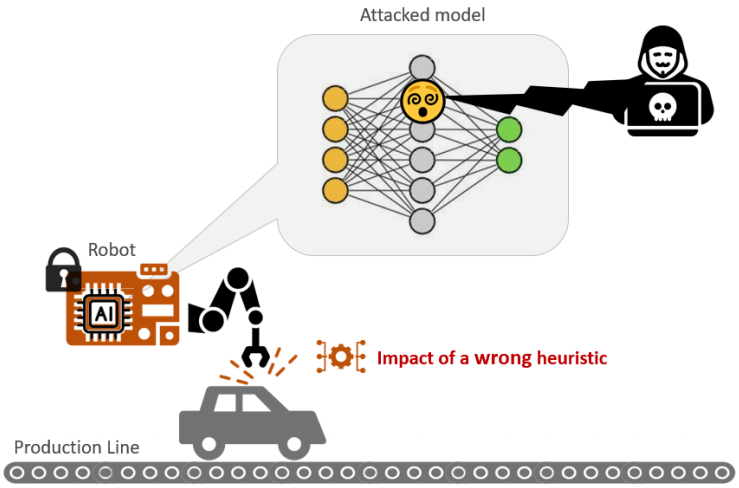

Figure 3. Impact of a tampered neural network on actuator behavior.

The goal of the attacker is to tamper with the NN model in order to generate wrong inferences, i.e., incorrect classifications in the output. As the inference result may be taken into account to command the actuators, the aim is to generate an inappropriate action causing financial losses or safety defects.

2.5. Beyond HistoTrust Use Case

HistoTrust has demonstrated the value of using a blockchain to guarantee the authenticity of the logs generated by the equipment operating on the production line, enabling each player involved in the manufacturing process to prove the tasks carried out and for which they are accountable. In this way, if an incident occurs, evidence of the actions carried out is recorded in an orderly, time-stamped, and immutable manner in the blockchain ledger and is available for consultation with all players, as well as an authorized third party, such as a legal officer, for example. Thus, no logs may be altered to mask liability.

Now that a blockchain is deployed and accessible by robots that record evidence from their logs at the rate at which blocks are generated, the blockchain may be used for other purposes.

With a few additions to the embedded software, the devices are able to detect any tampering of their NN model in real time. They also embed all the facilities to easily send a transaction to the blockchain to record the event and propagate an alert that can be consulted immediately by all client devices connected to the blockchain in a decentralized manner. This anti-tampering protection embedded in physical devices can complement the capabilities of the IDS to enhance system security. Moreover, the propagation of the alert to peers on the blockchain network provides greater transparency between actors and enables hardware countermeasures to be considered if necessary. The use of blockchain and smart contracts to record incoming events in an immutable and orderly way, in order to propagate alerts to subscriber peer devices, enhances security at the OT level, while new attacks target the transport of alert messages in the OT network to the centralized ICS.

3. Background and Related Works

This section presents existing work on bit-flip attacks on embedded artificial intelligence (AI) models, means of detecting and/or preventing these attacks, and blockchain-based detection solutions. The aim is to highlight their contributions and limitations.

3.1. Bit-Flips Attacks

A bit-flip attack involves a hacker flipping a few bits in the memory address range where the NN model is stored to cause unexpected behavior in the AI. In [19], Leemann et al. make the distinction between the attacks of NN models on black-box versus white-box. Attacks on white-box mode presume an attacker with full knowledge of the model, its training pipeline, and parameters. The full knowledge of the NN model is not available at runtime once the neural network has been embedded in operating electronic devices. This corresponds to the black-box model, meaning that an attacker can only query the model. Such attacks are more realistic in our use case. Depending on the time and resources available to the attacker and the performance of the surveillance system, it may be possible to dump the flash memory hosting the NN model into the targeted device and exfiltrate this data to the outside world for remote analysis.

Nevertheless, most scientific papers published so far consider attacks in the white-box, where the attacker has full knowledge of the neural network architecture and memory layout. Therefore, precise bit-flips may be performed through fault injection techniques to deteriorate the model until it behaves like a random guess level [20] or misclassifies specific inputs [21]. This approach is very effective but not very realistic for our use case.

When the attack is carried out in black-box mode, the bits to revert are targeted randomly. Studies such as [22,23] showed that random inversion of weight values requires several random guess levels to reduce the classifier accuracy in black-box or semi black-box contexts. However, these works were carried out on large NN models, with no size constraints. In embedded systems, NN models are very small in comparison, and their performance is a matter of compromise. In the following part of this paper, we consider a black-box attack scenario to target an embedded NN of size 25 KB.

Bit-flip attacks can be performed with mainly two approaches. The first one is a common bit-flip, where the attacker has little to no idea of the memory addresses to target in order to cause the desired effect. Thus, the attacker will perform random attacks. Experiments showed that this approach led to strong results for certain NN models, such as the ResNet-18 model, for example. However, when quantization is applied [24], the results are not as profound. Moreover, this attack may be detected through periodic verification of the NN metrics.

The second approach is the Target Bit-Flip (T-BFA). This technique targets model weights identified as having a strong impact on the inference result [21]. This requires prior identification of the memory addresses where these weights are located. In this way, it is possible to significantly reduce the accuracy of the model with a single bit inversion.

In order to perform these bit-flips, many techniques have already been developed through the exploitation of some vulnerabilities or physical degradation of the chip. For example, the Row Hammer technique is not easily detectable, as no input data, training, or network traffic is affected [25].

3.2. Defense Mechanisms

Detection: A bit-flip detection mechanism suggested as a possible solution for manufacturers is the use of Error Correction Codes [26]. This involves adding parity bits to the model so that its integrity can be restored if an attack occurs. Although it has been in use for a long time, and even with newer technologies such as DDR5 bringing On-Die Error Correction Code (OD-ECC), it has been shown that bit-flips are still feasible [3]. Moreover, integrating the NN model into an embedded system requires compromises to satisfy the constraints of the embedded system while ensuring correct performance. Adding parity bits to the model is not a viable option for embedded systems, as it greatly increases the size of the model.

Therefore, it is necessary to seek complementary approaches that can help reduce the applicability of this type of vulnerability.

Some of the state-of-the-art approaches to detect model tampering include the use of runtime verification systems such as checksum [27] or hash signature [5,6]. Even though both techniques are used to verify data, they serve different purposes. A checksum is a simple value calculated using a basic mathematical operation to detect accidental or intentional errors, while a hash is a complex algorithm to convert data into a fixed-size string.

These solutions have demonstrated mostly the highest tampering detection rates due to their characteristics, but have also added some overload to the process. It is important to find a proper balance between these two metrics.

Prevention: Some traditional defense mechanisms for this problem include adding some type of quantization to the model [28]. Quantization is an optimization technique that aims at reducing computational effort and memory usage while keeping or minimizing the loss of accuracy. In practice, several Machine Learning models use high-precision floating-point, which is solid in terms of accuracy, but not in terms of processing power. Thus, by converting these floating points into integers, for example, it is possible to maintain good performance while reducing the overall requirements of the model. In addition,

other approaches have applied changes to the neural network structure, adding random dummy operations during model inference [11], which have decreased the effectiveness of bit inversions in general.

The experiments described in this paper were performed on a model with floating-point weights that are not quantized.

3.3. Taxonomy of Intrusion Detection Systems

The two main tasks of an Intrusion Detection System (IDS) are to detect an intrusion and to propagate the alert to the Industrial Control System (ICS), which will take the appropriate measures.

An Intrusion Detection System (IDS) is a security tool that performs extensive analysis of network traffic and devices, searching for known and unknown malicious activities, along with security policy violations. Liang introduces in [29] a taxonomy of the IDS into three characteristics: (1) the class, Host-based or Network-based, (2) the detection method, Signature-based, Anomaly-based, Specification-based, or Hybrid, (3) the placement strategy, Centralized, Distributed, or Hybrid.

On one hand, Host-based IDS consists of mechanisms embedded in critical devices that scan logs and audit records to detect intrusions or anomalies. It may provide detailed information, according to the capacity of the host. It can only see and analyze what is happening in the host being monitored. On the other hand, Network-based IDS is more complex and aims to detect abnormal behaviors and unexpected data flow in the network traffic.

These systems can usually be defined as Signature-based methods, where the system looks for known patterns in order to define if an activity is malicious or not, and Anomaly-based methods, where it searches for deviations from a known behavior. Specification-based detection relies on rules defined in advance, and Hybrid-based methods refer to the use of any combination of the above-mentioned detection methods.

The centralized IDS is composed of several agents monitoring hosts and/or the network. The alerts and/or logs are sent to the ICS for further analysis. However, these IDS are not designed to detect a distributed attack taking place across multiple hosts of the network. In the Distributed IDS (DIDS), the analysis of events is spread across the network, with each element able to monitor beyond itself and to analyze the received events. Recently, a novel strategy has emerged with Collaborative IDS (CIDS). These IDSs rely on blockchain and deep-learning techniques to detect intrusions in a collaborative way [30]. They have the ability to correlate events that may occur on different hosts.

3.4. Blockchain-Based Platforms for Monitoring NN Decision-Making in Embedded Systems

The EU AI Act (https://artificialintelligenceact.eu/ accessed on 10 December 2025) published by the European Commission requires decisions taken by artificial intelligence to be explainable. In the context of Industry 4.0, this will make it easier to determine who is accountable in the event of incidents caused by bad decisions involving AI. However, if the classifier has been poisoned during the training phase, accountability may be more complex to assign.

In [31], the authors highlight the advantages of a decentralized CIDS topology, with peer-to-peer exchanges facilitating data sharing, correlation, and aggregation of data between all connected devices. They relate that these exchanges come at the cost of peers' interconnection, which consumes a lot of communication resources. Their contribution consists of using a blockchain to reduce this communication overhead and improve trust. In this solution, the logs of the peer devices are recorded in the blockchain ledger. Thus, every peer has access to the other's logs. However, we argue that the logs are confidential data,

covered by the contract between each device supplier and the production site manager. The scheme introduced by Koumidis in [32] suffers from the same limitations linked to the confidential nature of logs.

The work described in this article presents a Host-based Intrusion Detection System capable of detecting anomalies in the behavior of an embedded AI, according to a decentralized Collaborative IDS (CIDS) strategy. Each device is capable of monitoring its own embedded AI, to register alerts in a blockchain ledger, and to subscribe to events emitted by the blockchain through smart contracts. The event receipt allows each device to handle an alert on its own.

The papers [33,34] describe the use of a blockchain to supervise NN decision-making. In [33], blockchain is used to trace the decision of several AIs operating off-chain, while the inference details are stored in an IPFS off-chain storage. The goal is to make inference aggregation explainable, and to attribute a reputation score to the various AIs acting off-chain. In [33], the framework is envisaged for the healthcare field. It includes large AI systems and manipulates voluminous data. In [34], the authors describe an information system that takes advantage of a blockchain to protect the authenticity of explanations of the behavior of AI operating on Intel Core i7 processors https://en.wikipedia.org/wiki/Intel_Core#Core_i3/i5/i7/i9 (accessed on 10 December 2025).

HistoTrust [9], presented in Section 2.1, provides a suitable solution to trace and assure the authenticity of embedded AI's decision-making operating in collaboration in a decentralized network. This Blockchain-based solution offers an already deployed infrastructure to integrate the detection of embedded NN models, according to a decentralized Collaborative IDS (CIDS) strategy, with little overhead.

Other papers introduced blockchain-based mitigation techniques for AI model tampering attacks [30,35–38]. These studies focus on Federated Learning techniques using the blockchain to share refinements on the model. Each peer device embeds learning capacities and shares with others the result of its learning process in order to build a better model in collaboration. The accuracy of each recording model update is checked, leading to the assignment of a reputation score to each participating device, while simultaneously combating tampering attacks that reduce accuracy. However, this technique does not apply to our use case, firstly because each robot performs a different task and its embedded AI has been trained precisely for its own task, and secondly because the model of the embedded AI is confidential data belonging to the supplier. Also, the models cannot be shared with peer devices, possibly supplied by competing companies, on a ledger.

Other work focuses on the use of smart contracts operated on a blockchain to replace the publish–subscribe mechanism with events managed by smart contracts developed using the Solidity language [39,40]. The Proof-of-Concept depicted in the following part of this paper relies on smart contract events to alert peer devices to the occurrence of a bit-flip attack on a peer host.

4. Blockchain as a Security Enabler

A blockchain is a network of peer validators agreeing by consensus to add incoming transactions to a common ledger. The ledger is composed of successive blocks, each containing several transactions, linked cryptographically. The blockchain brings new properties, such as transaction ordering, immutability of the ledger, and transparency between connected peer devices, that reinforce security in a decentralized context [41].

4.1. Blockchain Fundamentals

In the blockchain ecosystem, we can define three functional roles: (1) consensus participants (e.g., miners in PoW, validators in PoS), referred to in this model as validator peers;

(2) ledger maintainers (e.g., full nodes), referred to as verifier peers; and (3) transaction generators, referred to as client peers.

Clients are peer devices, capable of generating and sending transactions to validator peers via a dedicated protocol called Web3. They integrate a wallet with a secret of their own.

Peer validators and verifiers form the infrastructure of the blockchain, aimed at producing the immutable ledger and making it available to clients. Validators write in the ledger, whereas verifiers read from the ledger. Peer validators take part in the consensus, with the aim of agreeing on the content to be added to the ledger. Peer verifiers hold a complete copy of the ledger and are reachable by clients.

The blockchain infrastructure is based on a consensus algorithm. Proof-of-Stake has been chosen for the Ethereum network since 2022 [42]. The idea behind this consensus algorithm is for peer validators to agree on the next block to be added. This consensus is fundamental to security: it ensures that even if some peers are malicious, the blockchain infrastructure remains available and safe as long as the majority of peers are working properly.

Thus, there is no central authority in a blockchain network. The ledger is replicated across many peers, and additions are only accepted if there is consensus. This avoids single points of failure. If a peer is compromised, it cannot falsify or delete entries on its own. For our use case, this ensures that the alert mechanism itself is robust against compromise by a minority of peers.

Once the records have been written to the ledger and finality has been achieved, i.e., a sufficient number of blocks have been added afterwards, any modification of the records is impossible without being detected. This would break the cryptographic links with the following blocks, which would alter the ledger. In our context, this means that an alert recorded in the ledger cannot be modified or deleted retroactively by an attacker, which provides immutability.

Transactions on the blockchain are digitally signed by the private key of the issuing client. This makes it possible to trace the device that issued an alert, without any confidential information being divulged in the network. In addition, the blockchain guarantees the ordering of events in the ledger, protecting the integrity of the records.

4.2. Smart Contracts

Blockchain is used to record evidence and transfer ownership of an asset. This may be performed through the use of smart contracts. Smart contracts are pieces of code, deployed in the ledger and executed by the peer validators [43], which enable a defined operation to be carried out on transactions submitted by clients. The smart contract is executed on receipt of a transaction.

4.3. Deployment of a Trusted Private Blockchain in an Industrial Site

Blockchain has been studied and applied to various fields, from distributed energy systems to healthcare [44]. However, its application to Industry 4.0 has yet to reach its full potential due to existing technological challenges [45] and the high costs involved in implementing new technologies. Yet blockchain could solve the difficulties encountered when it comes to automating processes involving several independent players, such as suppliers or service providers, for example. If this technology is to be adopted, the players in the ecosystem need to feel confident that their interests will be protected and that it will deliver added value [46].

In a factory, the OT network is not connected to the outside world for reasons of security and isolation; therefore, the use of blockchains that have been scaled up, such as Ethereum 2.0, is not an option. The deployment of a blockchain on a private network or

a consortium network will be preferred for our use case. The main difference between blockchains operating in private or consortium networks lies in governance. While all peers in a private network blockchain are operated by the same legal entity, a blockchain in a consortium network is governed by a set of independent actors [47]. Fairness among actors is ensured when each holds the same number of peer validators and, therefore, voting power within the consensus.

The deployment of a blockchain in a consortium that equitably involves the various stakeholders participating in the industrial process would be the most desirable. The stakeholders enrolled in the consortium may include robot suppliers, experts who have trained the AI, maintenance companies, operators, etc. However, this mode of governance is difficult to implement by design, as several stakeholders involved are not physically present in the factory and need to maintain their peers remotely, on a site disconnected from the outside.

Also, the deployment of a blockchain on a private network, composed of peers operated by the factory manager, is more realistic. However, with this solution, the distributed trust, a priori provided by the blockchain, is lost because a single actor operates all the peer validators and holds 100% of the voting rights within the consensus.

In the following, we propose a solution for deploying a blockchain on a private network, with a trust anchor. This anchor makes it possible to provide stakeholders in the ecosystem with trust in the deployed infrastructure and the immutability of the produced ledger.

Ethereum Foundation provides a framework for deploying an Ethereum technology-based blockchain on a private network with Proof-of-Stake consensus [48]. It enables building a blockchain infrastructure composed of a limited number of peer validators and verifiers through Kurtosis [49]. For the implementation, each peer validator consists of a consensus node that operates in coordination with an execution node. The consensus node takes part in the consensus protocol with the validators of the other peers. The execution node handles transaction processing, state management, smart contract execution, and offers a connection port to the clients. Each peer verifier is composed of an execution node. Through this approach, it is possible to define various types of configurations, including elapsed time between blocks, number of deployed nodes, choice of consensus and execution nodes development language, among other configurations.

Figure 4 illustrates the private network blockchain infrastructure, which is composed of peers acting either as validator nodes or verifier nodes. Their role is to generate and maintain the ledger, which is a chain of valid blocks. The validator nodes, depicted in red, are responsible for appending new transactions in the ledger through the generation of a new block to add to the chain of cryptographically linked blocks. The validator nodes agree by consensus. The verifier nodes, depicted in violet, are responsible for transmitting the incoming transactions from the clients to the validator nodes. They also check the consistency of the whole ledger exposed to the clients.

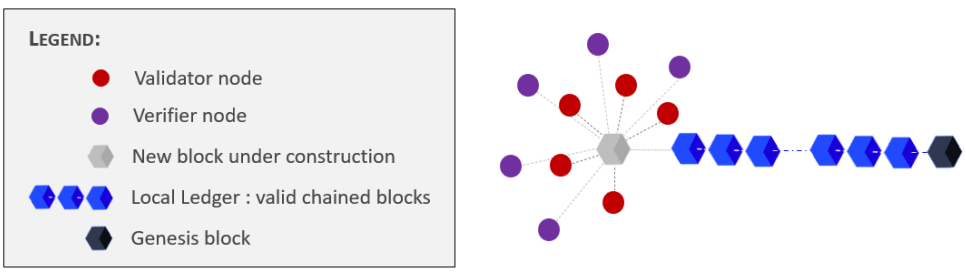

Figure 4. Overview of a private network blockchain.

However, with such an infrastructure, where all the nodes are operated by the same actors, trust and ledger immutability are not ensured. To recover these main properties, we suggest deploying this private network blockchain as Layer 2 of a scaled public network blockchain, such as the mainnet of Ethereum 2.0. To perform this, a digest of the private network's blockchain ledger is calculated at regular intervals, using a hash function such as SHA256, as illustrated in Figure 5. This digest is encapsulated in a transaction sent to Ethereum 2.0 mainnet through the proxy and the IT network of the industrial site. This anchoring mechanism ensures the integrity of the private ledger and provides a verifiable proof of activity of the private network without exposing sensitive details. This mechanism ensures the anti-tampering of the information contained in the private ledger only once the digest has been registered in Ethereum 2.0 and thus made public.

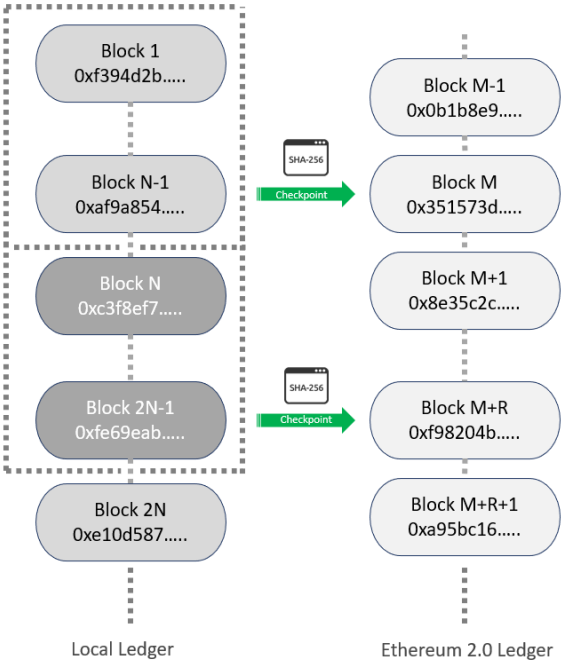

Figure 5. Deployment of private network blockchain as Layer 2 of the public network Ethereum 2.0.

In other words, the private network blockchain appears as a new element of the industrial network, located in the OT network and producing a private ledger that is available to the physical devices, acting as clients of this blockchain. The transaction, including the digest of successive blocks of the local ledger, is built by a verifier node of the private blockchain infrastructure. The resulting digest, denoted *digest*(i), is obtained by hashing with the function SHA256, the previous digest, denoted *digest*(i-1), concatenated with the current block identifier, denoted *blockID*(i) , as explained in Equation (2). For a given block, *blockID* is the footprint of the block, and the first block of the set of N blocks is denoted *blockID*(1). The signed transaction is routed through the OT network to reach a host located in the IT network. The role of the host device is to send this transaction to the public Ethereum 2.0 blockchain via the Ethereum JSON RPC protocol. If the transaction is modified while being routed within the industrial network, Ethereum 2.0 will reject it. Figure 6 shows how the private blockchain network can be integrated into the industrial network.

$$digest(i) \;=\; SHA256(\; digest(i-1) \;||\; blockID(i) \;)\; for\; i \in [2; N] \tag{1}$$

$$digest(1) \;=\; blockID(1) \tag{2}$$

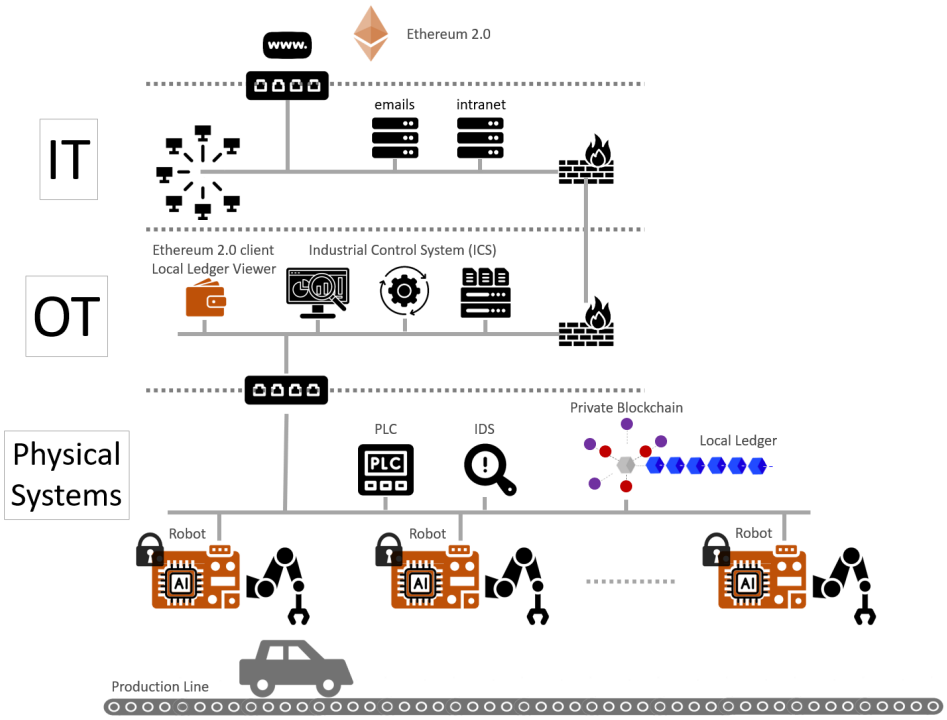

Figure 6. Overview of an industrial network, including a trustworthy private network blockchain.

5. Distributed Event-Driven Intrusion Detection

5.1. Need for Distributed Alert Propagation

With the modernization of industry and the increasing use of distributed connected devices, cyber-attacks are evolving, and some operate in a distributed manner, requiring new defense mechanisms to be adapted [50]. When a security flaw is detected in a device, host-based detection systems forward the alert to the ICS. Using blockchain and smart contracts, the alert can be propagated directly to other devices operating in the factory. In the multi-agent scenario we are working on, the detection of tampering of an embedded NN in a given device should incite the other devices to redouble their vigilance or switch to secure mode. In the following, we present the solution implemented to propagate an alert regarding the detection of a tampered embedded NN, via a smart contract operating on a private blockchain network. This is performed through a publish–subscribe-like mechanism, using event-driven smart contracts.

5.2. Event Mechanisms in Smart Contracts

Smart contracts make it possible to define rules to automate the processing, recording, or transfer of a digital asset in a blockchain ledger. Once the result of the operations linked to the content of the incoming transaction is recorded in the ledger, smart contracts also have the ability to issue events. Through the RPC interface of the Ethereum client protocol, the connected devices can subscribe to these events. This mechanism, which is similar to publish–subscribe techniques, has the advantage of operating in a distributed context.

There are already mechanisms for distributing messages in the form of publish–subscribe, such as Apache Kafka [51] or MQTT [52]. However, these mechanisms are integrated into a data-centric information system, which is limited in terms of trust in a multi-agent system [40]. Blockchain offers a device-centric system architecture. The ledger, which contains the data published by customers, is immutable, and its content is transparent to all connected devices. By using a blockchain, we avoid creating a single point of failure, and the data stored in the ledger is secured by cryptography and visible to all

the actors involved. By subscribing to chosen events, each device is notified in real-time of new additions to the ledger centered on its points of interest, without the need for polling.

Through any execution node, the devices acting as clients are able to send transactions encapsulating public data and to subscribe to events. It is a smart contract that orchestrates the mechanism for publishing the alert by executing the code targeted by the incoming alert transaction, and submitting the result of this execution to the validating peer consensus, as well as the mechanism for propagating the alert by issuing an event. Devices subscribing to the 'alert' event type will be notified each time a new alert is added to a ledger block. This orchestration is illustrated in Figure 7.

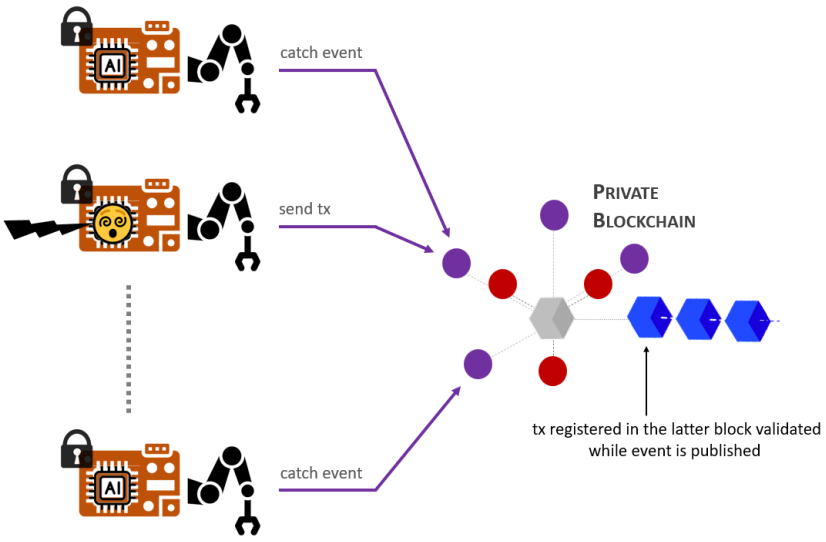

Figure 7. Alert propagation through the blockchain clients.

5.3. Smart Contract Design for Event-Based Alerts

The smart contract, deployed in the ledger and executed by the blockchain's validating peers, contains the rules and constraints for the considered use case. A simplified version in the Solidity language is shown in Listing 1. The keyword of Solidity language are written in blue. The general overview of how it works is shown in Figure 8.

Listing 1. Smart contract example code.

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.5;

/// @notice This contract is intended for illustrative purposes only
contract MnistNNTamperAlert {
    mapping (address => uint256) public nn_hash;

    event registerTampering(
        address indexed sender,
        uint256 nnModelHash
    );

    function registerTampering(uint256 _nn_hash) external {
        nn_hash[msg.sender] = _nn_hash;
        emit registerTampering(msg.sender, _nn_hash);
    }
}
```

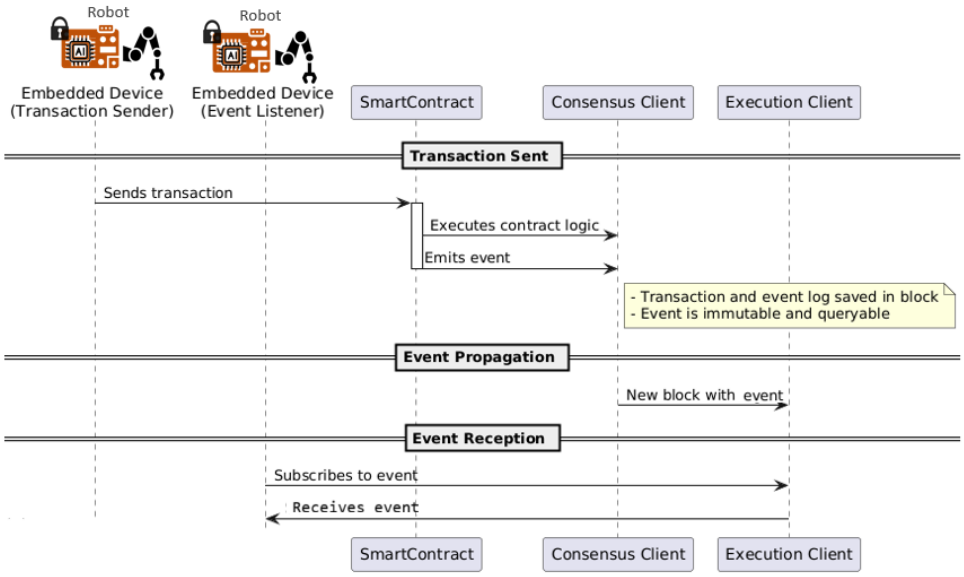

Figure 8. Overview of the Solidity event.

The diagram in Figure 8 details the decentralized alert propagation process. The first robot, named *transaction sender*, emits a transaction that includes an alert to the targeted smart contract on the blockchain. The record of the alert in the ledger is submitted to the consensus of validating peers. If there is an agreement, recording the alert in the ledger raises an event. The other robots, represented in Figure 8 by *event listeners*, receive the alert if they have subscribed to the event. In this way, the alert is propagated.

Our use case consists of detecting the tampering of an embedded NN model and propagating the alert in a decentralized context. It could be extended to the propagation of other malfunction alerts detected by the IDS [53]. Each alert issued by the IDS would then be encapsulated in a transaction, processed by a smart contract, and submitted to the blockchain. This is fully compatible with the architecture introduced in this paper, and could be deployed at low cost, as all the components are already present.

6. Embedded Design

In our use case, the clients of the blockchain infrastructure are robots. They embed MPU-based electronic boards, including secure hardware components.

6.1. Hardware Architecture Overview

The electronic board STM32MP157-EV1 from ST Microelectronics is used to achieve a Proof-of-Concept of HistoTrust and beyond HistoTrust use cases. The core of this MPU-based board is a low-power dual-core microprocessor, an ARM Cortex-A7 https://en.wikipedia.org/wiki/ARM_Cortex-A7 (accessed on 10 December 2025) , which hosts both a normal world running an embedded Linux and a secure world running the OP-TEE (https://github.com/OP-TEE accessed on 10 December 2025) (Open Portable Trusted Execution Environment) operating system within ARM TrustZone technology. The board also integrates an ARM Cortex-M4 microcontroller, which hosts the embedded NN model.

The ARM Cortex-M4 microcontroller is controlled by the embedded Linux. It is used as additional computing power or for real-time tasks. A shared memory between the ARM Cortex-A7 and the ARM Cortex-M4 enables the exchange of a large volume of data, such as images and inferences, through the protocol RPMSG (Remote Processor Messaging). A complete overview is shown in Figure 9.

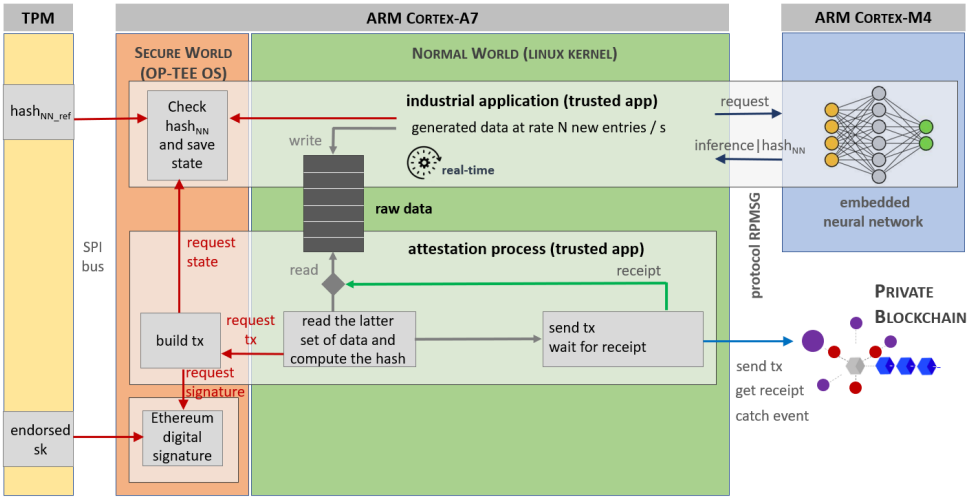

Figure 9. Overview of the embedded design.

In addition, the TPM expansion board STPM4RASPI is added as a daughter board to the STM32MP157-EV1. Operated from the TrustZone through a secure SPI bus, the TPM provides hardware security to store secrets in its vault and to authenticate the device based on the TPM standard endorsement hierarchy.

6.2. Embedded Software Architecture

For the Proof-of-Concept, the neural network (NN) was trained using Tensorflow https://en.wikipedia.org/wiki/TensorFlow (accessed on 10 December 2025) [54] Keras to recognize handwritten digits from zero to nine taken from the MNIST dataset [55]. However, any other dataset can be considered to train the AI.

The board embeds two independent processes operating asynchronously. The first one consists of an industrial application, based on an embedded NN, running in real-time to operate a dedicated industrial task. The second one is a distributed application (dApp) running an attestation process, which is based on an embedded hardware wallet, connected to the blockchain infrastructure through an Ethereum RPC client protocol stack in the normal world. These two processes are linked through a buffer. While the industrial application writes to the buffer at the rate of operations, the dApp reads the data in the buffer as it receives receipts from the blockchain. The buffer is sized to take into account that the transaction sent to the blockchain may not be included in the next block, and to wait a little while. However, after several minutes or even several hours, if the transaction is still not validated, the industrial application stops. In principle, this problem should not arise because, in the case of a concrete deployment, several parameters must be adjusted to avoid it, including the rate of data generated, the amount of data recorded in the buffer, the size of the buffer, and the time interval between two consecutive blocks in the ledger. The use of a private blockchain network avoids block congestion, in principle, which can sometimes occur on a public blockchain.

Each new digit image collected at the Linux level is presented to the neural network for recognition. As output, the NN provides 10 probabilities as inferences, one for each possible recognized digit. In addition, the footprint of the memory hosting the NN model within the ARM Cortex-M4 is computed. All 10 probabilities and the current NN hash are returned to the industrial application embedded on Linux via the RPMSG bus. From here, the inference is written in the buffer, whereas the current NN hash is sent to a trusted application running in the TrustZone to be compared with the initial NN hash that serves as a reference. If the two hashes are identical, everything is correct; if not, the NN model

has been tampered with. In this latter case, an alert is raised, triggering the build of a new transaction while bypassing the buffer read.

A trusted application, located in TrustZone, is dedicated to building transactions according to the prototype of the smart contract's function. More precisely, the transaction may target the function `registerTampering` with the smart contract sample presented on Listing 1. The NN current digest (different from the reference digest) is retrieved from the TrustZone and encapsulated in the transaction under construction. Then, the signature of the transaction is performed into the TrustZone using a private key stored in the TPM. Once completed, the transaction is sent to the blockchain infrastructure through the embedded Ethereum RPC client communication stack.

Figure 10 depicts the process of the embedded industrial application. As long as the NN is integer, the process continues to make inferences. However, if the NN is tampered, the fault is attested and sent to the smart contract through a transaction, as illustrated in Figure 11.

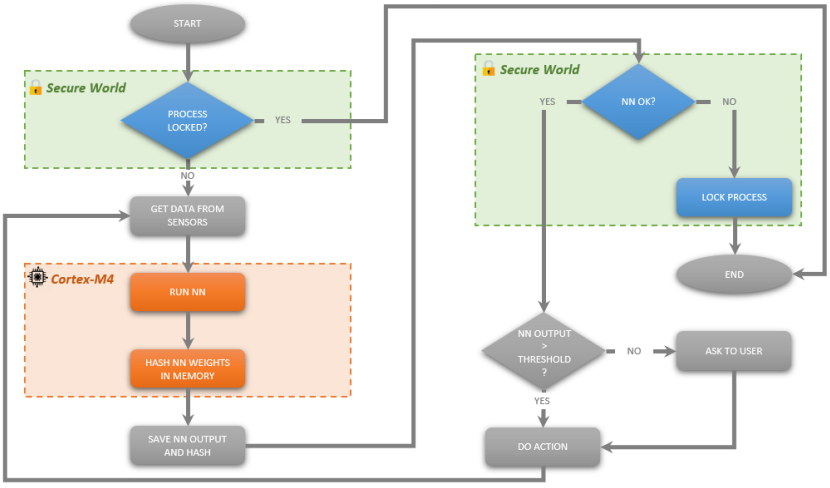

Figure 10. Client's application flow.

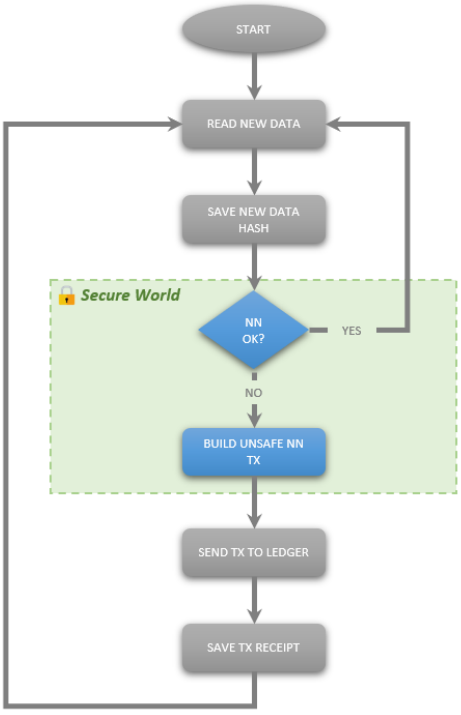

Figure 11. Attestation application flow.

6.3. The Proof-of-Concept

The Proof-of-Concept of this distributed scheme consists of the detection of embedded NN model tampering and the propagation of the intrusion alert through an event emitted by a smart contract operated in a blockchain infrastructure. It was integrated into the HistoTrust platform, shown in Figure 12.

Four electronic boards, STM32MP157-EV1, are used as clients of a private blockchain network. Each board emulates the control unit embedded in a robot. In the following attack scenario, one board suffers an intrusion affecting its embedded NN model, while the others are peer clients subscribed to the intrusion alerts issued by the blockchain. For each board, the digit photo presented at the input of the classifier is displayed on the screen.

Figure 12. Photo of the HistoTrust platform.

7. Attack Scenarios, Detection and Alert Propagation

7.1. Presentation of the Attack Scenario

In the scenario considered, an attacker gains physical or remote access to the ARM Cortex-A7 processor, which shares peripherals with the ARM Cortex-M4 https://en.wikipedia.org/wiki/ARM_Cortex-M accessed on 10 December 2025) that hosts the neural network. Using a malicious script, the attacker launches a random bit-reversal attack targeting the flash memory where the model is stored. It intends to disrupt the inference resulting from the neural network, leading to an inappropriate command being sent to the actuators.

The first attack is carried out on the assumption that the attacker has knowledge of the address range where the neural network model is stored in the flash memory of the ARM Cortex-M4. This knowledge enables him to launch a random bit-flip attack.

A second attack is launched on the assumption that the attacker also knows the structure of the neural network, enabling him to deduce the memory locations where the first and last layers of neurons are stored.

The neural network architecture used in this study consists of a 2D convolutional layer containing two filters of size 3×3, which are designed to process a single-channel input. This layer includes 18 kernel weights and two bias terms. The resultant output from the convolution process is flattened and subsequently fed into a fully connected layer featuring 16 neurons, with a total of 6272 weights and 16 biases being employed. The final layer is characterized by its dense composition, featuring 10 output neurons, which renders it particularly well-suited for classification tasks involving 10 distinct classes, one for each digit of the MNIST dataset. The output layer of the model contains 160 weights and 10 bias

values. It should be noted that no quantization has been applied to embed the model under consideration into the ARM Cortex-M4.

7.2. Impact of Random Bit-Flip Attacks on Inference Accuracy

To evaluate the damage caused by a random bit-flip attack targeting any bit of the neural network, the following experiment was carried out. Once the address range hosting the neural network model has been identified, a script injected through an open ARM Cortex-A7 peripheral randomly targets a bit in this address range and reverses it. One thousand runs were attempted. Four of which are randomly selected and presented in Figure 13. For each run, several trials are launched with 1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, and 1000 random bit-flips performed successively. After each bit is reverted, the memory layout is affected. The NN model is reloaded into the memory layout at each new run. Thus, it is always the same at the start. For each run, the same 100 pictures are presented as input to the neural network in order to measure the accuracy of the output inference. The accuracy results are reported in the graph of Figure 13. The curve is then smoothed using the PchipInterpolator of the Python 3.12 scipy package. The scale of the x-axis is logarithmic. It represents the number of bits reverted. On the y-axis, several benchmarks are indicated. A total of 98% accuracy corresponds to the minimum expected performance of the neural network; 90% corresponds to the threshold below which performance is no longer acceptable. For the 50% benchmark, the model is wrong one out of two times. As there are 10 possible output digits, once it finds the right one, whereas the other time it prefers one of the other nine digits. The 10% accuracy corresponds to a random selection of the chosen digit.

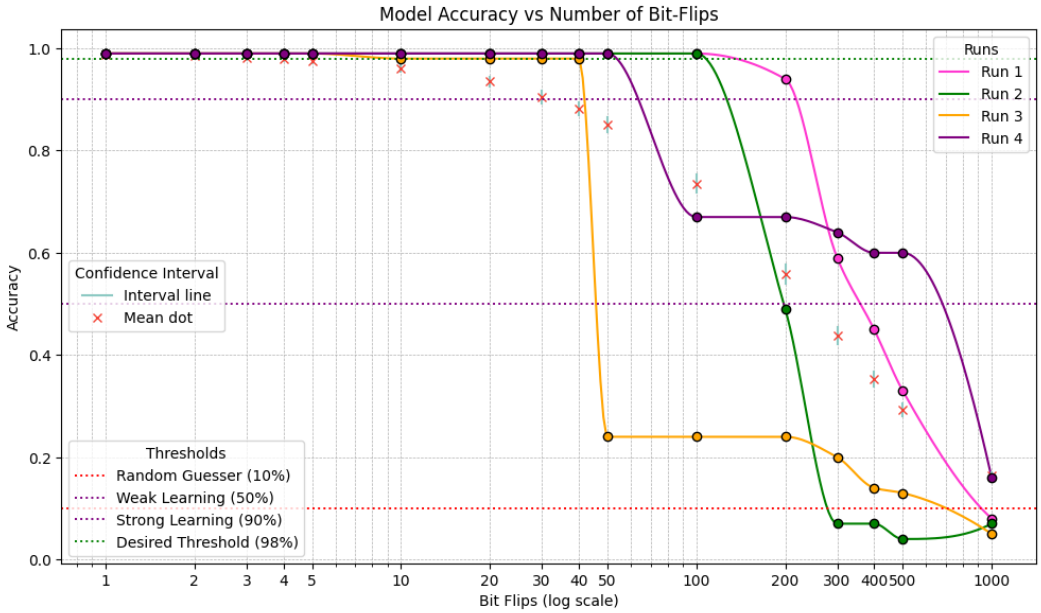

Figure 13. Accuracy of inferences when the NN is under random bit-flip attacks.

In addition, the confidence interval is computed for each measure, i.e., number of bits reverted, for a confidence level of 95%. For each measure, one thousand samples x_i coming from the 1000 runs are considered, so $n = 1000$ and $i \in [1; n]$ in Equation (3). The mean of this set of 1000 samples is denoted by $\bar{x}$, and the standard deviation is denoted by σ.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} |x_i - \bar{x}|^2}{n-1}} \tag{3}$$

The confidence interval yields the range of plausible values for the entire population mean. Computed with $n = 1000$ samples of accuracy values, the margin of error is around 3%. Thus, for a confidence level of 95%, the bounds of the confidence interval for a given number of bits reverted are given by Equation (4):

$$[\ \bar{x} - 1.96\ \frac{\sigma}{\sqrt{n}}\ ;\ \bar{x} + 1.96\ \frac{\sigma}{\sqrt{n}}\ ] \tag{4}$$

In Figure 13, the mean of the thousand samples x_i, with $i \in [1; n]$, for a given number of bits reverted, is drawn with red crosses, and the confidence interval is drawn with a blue line.

The results show that with only four bits reverted, the mean accuracy falls under the desired threshold of 98%. To fall under 90% of accuracy, 40 bits need to be reverted on average. However, there are large fluctuations between the different runs, which means that for some runs, the resulting accuracy is well below 90%. Beyond 40 reverted bits, the accuracy value declines significantly. Moreover, with 1000 bit-flips, the performance of the model is equivalent to that of a random guess.

Table 1 summarizes the statistics for each of the measurement points for random bit-flip attacks.

The gap between the minimum and maximum suggests that reversing certain bits could have a very strong impact, while others would have less impact. Looking at the minimum column in Table 1 confirms this hypothesis, as the model may be completely broken as soon as the first bit is reverted. This topic is explored in more detail in Section 7.5, where we are looking for those impacting bits in the layout of the flash memory.

Table 1. Statistics for each point of measurement for random attacks.

No. of Bit-Flip	Standard Deviation	Mean	Median	Minimum	Maximum
1	0.038	0.987	0.99	0.28	0.99
2	0.044	0.985	0.99	0.28	0.99
3	0.064	0.981	0.99	0.2	0.99
4	0.070	0.978	0.99	0.2	0.99
5	0.078	0.975	0.99	0.2	0.99
10	0.115	0.961	0.99	0.08	0.99
20	0.156	0.934	0.99	0.08	1.0
30	0.187	0.904	0.99	0.08	1.0
40	0.202	0.881	0.99	0.08	1.0
50	0.224	0.850	0.98	0.08	1.0
100	0.278	0.735	0.86	0.07	0.99
200	0.292	0.558	0.57	0.01	1.0
300	0.266	0.437	0.37	0.01	1.0
400	0.232	0.352	0.28	0.01	0.99
500	0.200	0.293	0.23	0.01	0.98
1000	0.097	0.164	0.14	0.01	0.73

7.3. Speed of Attack

To assess the speed of the attack, i.e., the time taken for the attack to impact the operation of the actuators, several points of interest are selected on the curve of the mean of the one thousand runs illustrated in Figure 14, i.e., the red crosses. Each point corresponds to the accuracy for a given number of reverted bits in the NN model. The curve is smoothed using the PchipInterpolator of the Python scipy package. Only four reverted bits led to a degradation of accuracy below 98%. With about 244 reverted bits, the inference is correct only one out of two times, and with 1000 reverted bits, the model is completely broken. Thus, the experiment consists of measuring the time needed to reach these points of interest, according to several hypotheses on the attacker's side. In the first experiment, the attacker is able to inject successive malicious scripts to revert one additional bit with "no delay", which means that the scripts are small enough and sent at a rate that cannot be detected by the IDS. In the second experiment, the malicious scripts are injected at the same rate as the inference output, which means that one bit is reversed at each new inference output. The

last experiment simulates an attack where a malicious script is injected every second, to remain undetected by the IDS.

For the sake of accuracy, we would like to point out that our NN model is completely broken when 1000 bits are reversed. However, this is a small model designed to be embedded in an ARM Cortex-M4. NN models running on servers are generally much larger, and the number of bits that need to be reverted to break the model is specific to each one.

For each experiment, the timings are reported in the Table 2. The targeted device detects the attack as soon as the first bit is reverted, i.e., as soon as t_{1BF}. The rate at which the malicious script is injected depends on the attacker's resources and the IDS's ability to detect the intrusion. The finer the analysis of network traffic by the IDS, the slower the attack will be. Also, the analysis of the timings reported in Table 2 can be used to decide on reaction strategies from the attacked device, the ICS, and peer devices.

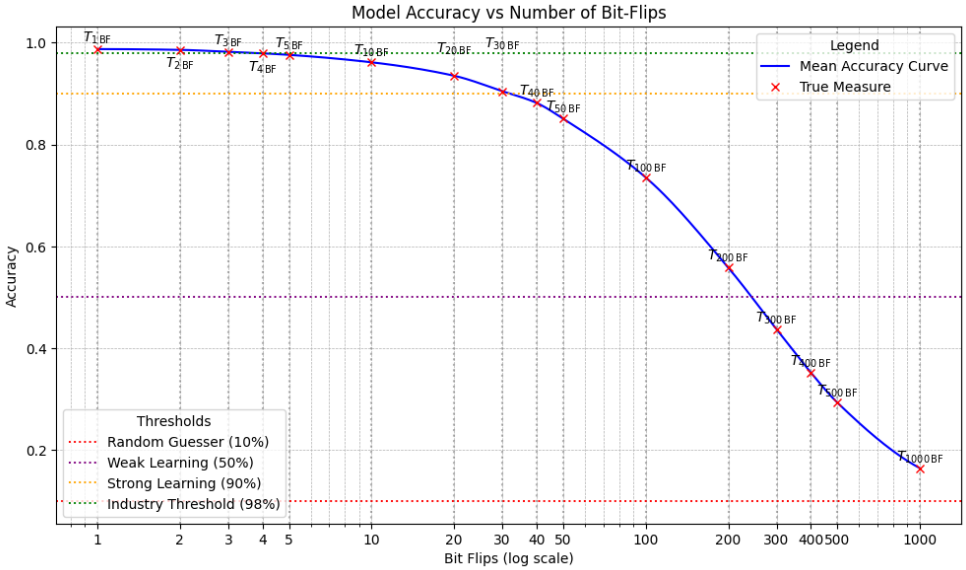

Figure 14. Points of interest for timing measurement.

Table 2. Timing of random bit-flip attacks for several experiments.

Bit-Flips	No Delay (T_{nBF})	=Inference Time	Slow Attack
T_{1BF}	0.07 ms	3.15 ms	1 s
T_{2BF}	0.14 ms	6.29 ms	2 s
T_{3BF}	0.21 ms	9.44 ms	3 s
T_{4BF}	0.28 ms	12.59 ms	5 s
T_{5BF}	0.35 ms	15.73 ms	5 s
T_{10BF}	0.71 ms	31.47 ms	10 s
T_{20BF}	1.40 ms	62.94 ms	20 s
T_{30BF}	2.11 ms	94.41 ms	30 s
T_{40BF}	2.80 ms	125.88 ms	40 s
T_{50BF}	3.51 ms	157.35 ms	50 s
T_{100BF}	7.10 ms	314.67 ms	1.6 min
T_{200BF}	14.20 ms	629.34 ms	3.2 min
T_{300BF}	21.30 ms	944.01 ms	4.8 min
T_{400BF}	28.40 ms	1.26 s	6.4 min
T_{500BF}	35.50 ms	1.57 s	8 min
T_{1000BF}	71.00 ms	3.14 s	16.6 min

7.4. Scheduling of Attack Detection and Alerting

Each device that might be targeted by the bit-flip attack is able to detect the intrusion when the first bit is flipped. It can then put itself in a safe position and propagate an alert in two independent ways: to the ICS by sending a message on the OT network and to the blockchain by issuing a transaction that will be registered in the ledger. While the ICS is a centralized supervisory and decision-making entity within a hierarchical infrastructure,

the blockchain offers transparent, decentralized information linking all connected devices. Additionally, the ICS is controlled by one stakeholder, typically the plant owner, whereas the ledger can be accessed by all parties involved. Through the ledger, even the smallest component of the physical system can receive an alert as soon as the event is appended to the ledger. Figure 15 provides an overview of the timing related to alert propagation.

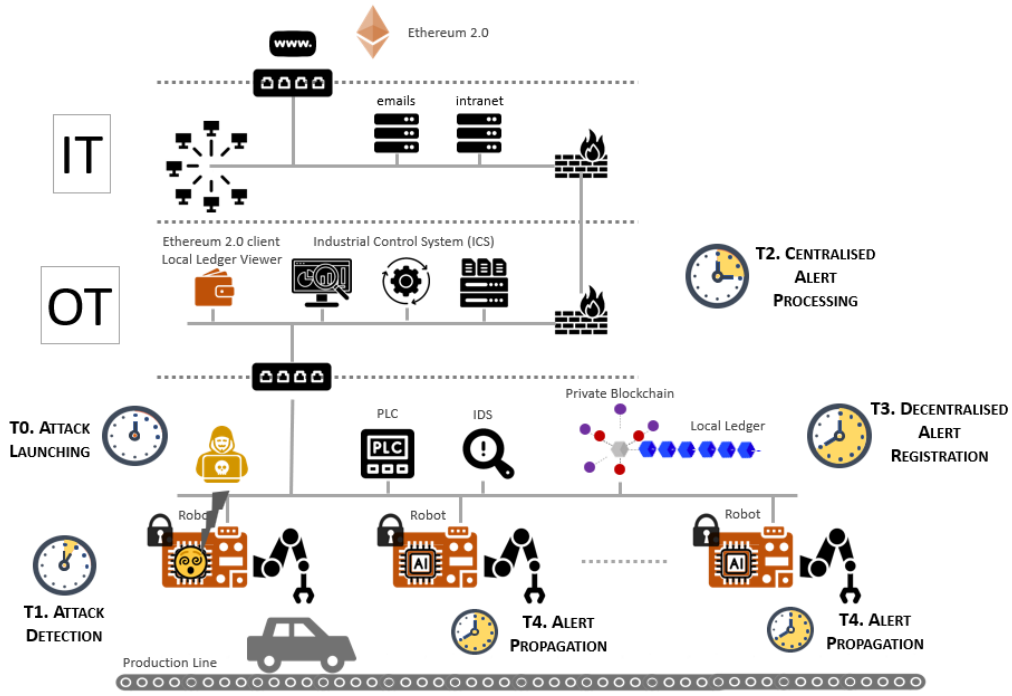

Figure 15. Scheduling of the attack detection and alert propagation.

The ICS makes decisions at the OT network level. In contrast, a physical device alerting through an event can protect itself and record relevant events in its own logs. This may be useful for in-depth analysis of what happened. The events are transparently shared with all parties involved.

Thus, with our implementation, the device detects the bit-flip attack 3.2 ms after the attack occurs, i.e., the first bit is reverted. If the attack occurs, it means that the IDS has not intercepted it. We are not able to provide the propagation time of an alert to the ICS, but it is significantly faster than recording an event in the ledger via the blockchain. For this latter process, the best timing scenario is from the attack detection to the event registration in the ledger is 2.4 s. The worst timing, when the interval between two blocks is fixed to 5 s, is measured at 7.4 s.

However, the random bit-flip attack does not impact the inference accuracy at the first bit being reverted. If the IDS is sufficiently powerful to avoid a speedy attack, that gives time to the ICS to react with safety actions in a brief delay and security actions if the attack is considered rapid or severe. For example, a safety action may consist of closing the input peripheral of the ARM Cortex-A7 while deferring the decision of the classifier to a human supervisor (or other defined mechanisms) until the attack is under control. Alerted by the event recorded in the ledger, the network's peer devices can apply decisions at their own level, depending on their embedded technology. They can also report the alert in their own logs for traceability purposes.

7.5. Impact of Targeted Bit-Flip Attacks on Inference Accuracy

To deepen this study, we consider a more powerful attacker who succeeds in gaining knowledge of the neural network model. Assuming that the attacker has succeeded in

dumping the memory range hosting the neural network model, he is able to know the topology of the network, the content of each of layer, and the weights of the various neurons, using tools like Netron 8.7.8 (https://netron.app/ accessed on 10 December 2025) for example. Then, a memory analysis enables the location of the physical address of each parameter in the flash memory. In the following, we assume that the first and last layers have the greatest impact on the inference result.

Analysis of the memory content extracted from the ARM Cortex-M4 shows that the parameters for a given layer are not stored in an organized manner but are located in the same addressing region. Figure 16 illustrates the organization of the parameters of the first and the last layers in a simplified way.

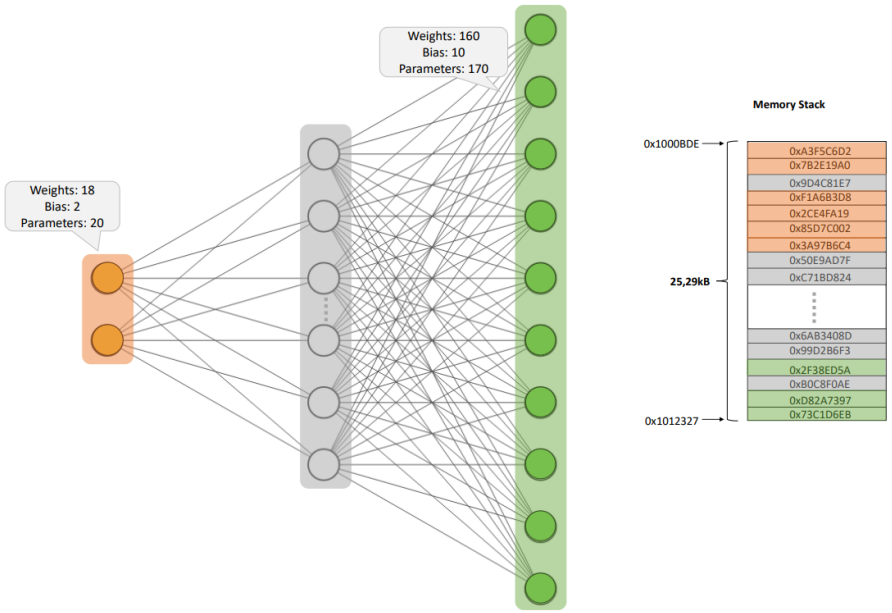

Figure 16. Overview of the NN model organization in the flash memory.

The analysis of the first layer provides the knowledge that it is composed of 20 parameters, two biases of nine weights each. Figure 17 illustrates the impact of a unique bit-flip on bit 30s MSB of the 32-bit value of the most impacting parameters. Since bit 31 encodes the sign of the 32-bit floating point value, bit 30 is the most significant bit of the floating point value.

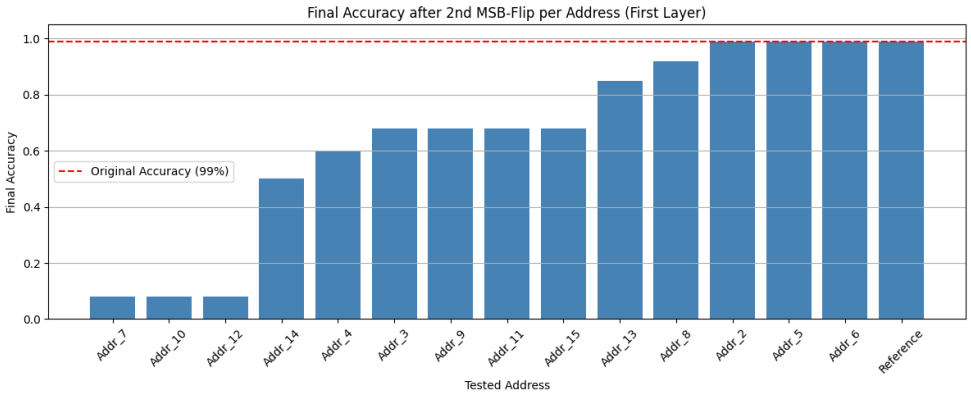

Figure 17. Impact of the flip of the second bit MSB of the 32-bit floating-point value at the targeted addresses of the parameters of the first layer.

These results show that by precisely targeting the bits with the greatest impact, it is possible to completely break the model by inverting a single bit. Targeting the parameters

at the addresses *Addr_7*, *Addr_10* and *Addr_12* yields the same performance as a random guess level, while targeting *Addr_14*, *Addr_4*, *Addr_3*, *Addr_9*, *Addr_11* and *Addr_15* renders the model inoperative. The reference index is a benchmark that shows the accuracy of the normal behavior of the NN model.

In the following, the analysis of the last layer, which is the classifier layer, composed of 10 biases of 16 weights each, is conducted in the same way. The impact of reverting bit 30 of the bias parameters is illustrated in Figure 18.

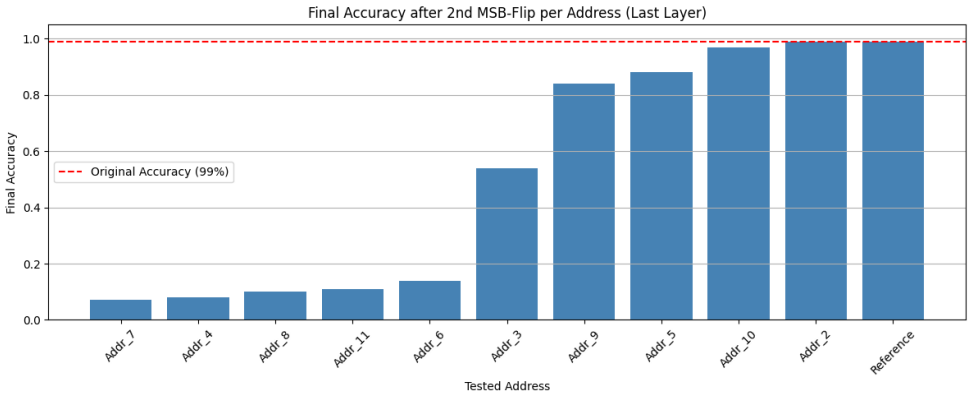

Figure 18. Impact of the flip of the second bit MSB of the 32-bit floating-point value at the targeted addresses of the bias of the last layer.

This analysis shows that parameters with negative values are less sensitive to bit-flips. When a neural network uses ReLU activation functions, for example, negative weighted inputs are often suppressed because ReLU outputs zero for negative values. If a negative weight becomes more negative, it usually does not change the output since the neuron remains inactive. However, when a negative weight flips to positive, it can activate previously inactive neurons, allowing signals to pass through the neural network and potentially alter the final prediction.

The final experiment simulates the most feasible scenario with the attacker's knowledge. This one does not know what the most impacting addresses are, but knows which region to attack in order to achieve the greatest accuracy reduction with the fewest bit-flips. For each selected address, only bit 30 of the 32-bit value is targeted. For this experiment, one thousand runs were launched under the same operational conditions as in Section 7.2, on a scale ranging from 1 to 100 for the number of bit-flips. Figure 19 depicts the results.

As a result, as soon as the first bit is reverted, the NN model can be severely degraded, as shown by the average accuracy value of 0.862. As soon as three bits are reversed, the model's performance falls to 50%, with a one in two chance of obtaining a classification error. The model is completely broken after 10 bit-flips. The confidence interval for measurements ranging from 1 to 10 bit-flips is quite wide, revealing a significant disparity in the impact of a given reverted bit. Some bits play a much more significant role than others in classification. The proportion of bits that have a large impact on the set of samples is much greater with this targeted attack than with the random selection attack shown in Figure 13. This demonstrates that having a basic understanding of the model's architecture is sufficient to improve attack performance and break the NN model with a few bit-flips.

Table 3 summarizes the statistics of the 1000 runs for targeted bit-flip attacks.

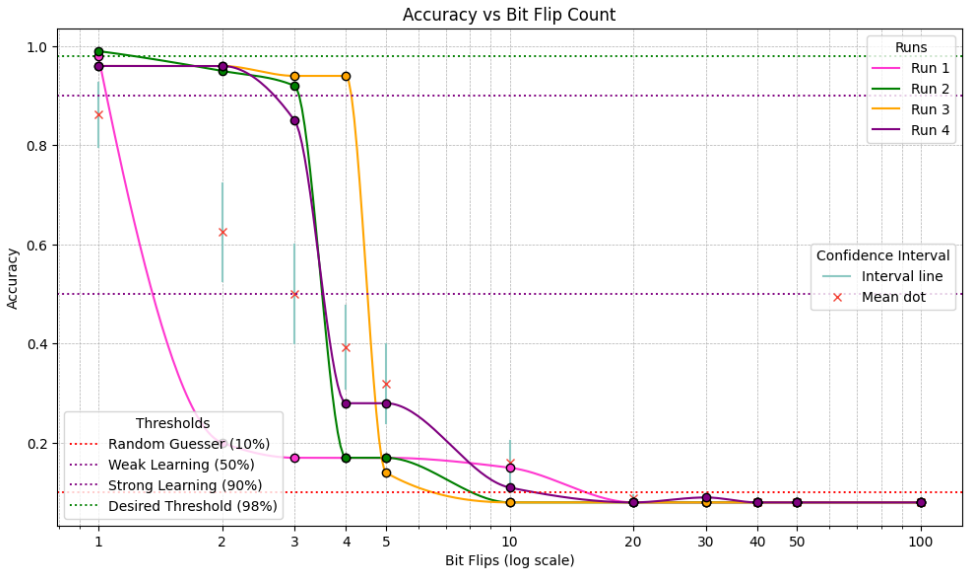

Figure 19. Accuracy of inferences when the NN is under targeted bit-flip attacks.

Table 3. Statistics for each point of measurement for targetted attacks.

No. of Bit-Flip	Standard Deviation	Mean	Median	Minimum	Maximum
1	0.231	0.862	0.97	0.08	1.0
2	0.353	0.624	0.81	0.07	1.0
3	0.353	0.500	0.415	0.07	0.99
4	0.301	0.393	0.265	0.07	0.95
5	0.283	0.320	0.17	0.07	0.93
10	0.154	0.160	0.08	0.06	0.79
20	0.020	0.088	0.08	0.05	0.16
30	1.31×10^{-2}	8.48×10^{-2}	0.08	0.08	0.15
40	2.77×10^{-17}	8.00×10^{-2}	0.08	0.08	0.08
50	1.40×10^{-3}	8.02×10^{-2}	0.08	0.08	0.09
100	8.40×10^{-3}	8.01×10^{-2}	0.08	0.08	0.14

7.6. Detection of the Severity of the Attack

In order to determine the severity of the attack, we would like to detect when a bit-flip occurs if the targeted bit significantly impacts the inference accuracy or not. This will provide information on the speed and severity of the attack and its impact on the command control decision.

For that, an 8-bit CRC, taking as input the 32-bit values of the most impacting addresses and storing the parameters of the first and the last layer of the NN model, is provided with the inference output and the hash of the whole NN model. In this way, the severity of the attack may be propagated with the alert in order to adjust the reactions.

7.7. Distributed Bit-Flip Attack

The experiments carried out target a specific physical device that is the victim of the bit-flip attack, assuming that the attacker knows the architecture of the software embedded in this device. In reality, the attacker's knowledge is generally more vague. Therefore, the attacker may adopt a strategy of targeting several different devices, hoping that the bit-flip attack will affect at least one of them. The attack may randomly affect a bit of an embedded NN or have no effect at all. Furthermore, depending on the integrated electronic components, the structure of the embedded code, and the physical and software protections deployed, some devices may be more vulnerable to bit-flip attacks.

In this context of distributed attacks, decentralized alert propagation makes perfect sense. It allows the various devices on the network to be directly notified of a risk and, if necessary, to adapt their level of security and vigilance.

7.8. Discussion

About Q1: The proposed approach provides a decentralized trust mechanism. Each device verifies its model integrity, and in case of a mismatch, generates an alert. The integrity check is performed in TrustZone. It consists of the comparison of the reference digest of the NN model stored in the TPM and the current digest of the NN model. Storing the reference digest in the TPM protects this value from tampering, including physical attack attempts. Since the comparison is performed in TrustZone, the result is trustworthy. This provides a security-by-design. In case of a mismatch, the alert is encapsulated in a transaction built and signed in TrustZone, using a private key that authenticates the device. The recipient of the transaction is the smart contract, thus ensuring end-to-end security. Therefore, once recorded in the ledger, the alert becomes immutable and non-repudiable. It can then be captured by other devices asynchronously.

About Q2: The speed with which the classifier is impacted by a bit-flip attack depends on the attacker's knowledge. The impact is greater if the reversed bit is the most significant bit in the first layer or the last layer of the neural network. Without this knowledge, with a random guess level attack, several bits must be reversed to impact the classifier. Statistically, the classifier makes a mistake 1 time out of 10 when 30 bits have been reverted in the NN model. The speed of this attack then depends on the rate at which a malicious person is able to inject these errors into the NN model.

About Q3: To go undetected by IDS, an attacker must be very discreet and generate very little traffic on the network. This is why attacks on random bits are much more likely and less complex to carry out than attacks on targeted bits. The latter also acts more slowly, allowing time for the alert to be propagated to peer devices on the network via the smart contract. The use of blockchain and smart contracts thus reinforces the defense provided by the IDS. Moreover, the blockchain infrastructure is resilient to failure, which reinforces the security in complement to the IDS. In addition, the blockchain infrastructure is fault-tolerant, which enhances security in addition to the IDS. If a peer validator fails, the network remains available, continues to operate, and the alert is propagated anyway. The time interval between two blocks of the ledger affects the latency of alert propagation. A short interval results in rapid propagation but requires more processing resources. A balance must be found based on the criticality of the environment.

The response adopted by a physical device when notified of an alert is not addressed in this paper. However, the advantage of our solution is that it allows operations to continue, possibly in degraded mode with increased vigilance, knowing that the impact of the attack is not immediate. For example, some decisions based on the classifier result may be deferred to humans until the attack is neutralized.

About Q4: In this paper, we suggest deploying the blockchain as a private peer-to-peer network, located as close as possible to operational physical devices. The peers that make up the blockchain infrastructure are not directly connected to the outside world or exposed to the internet, which enhances security. To strengthen trust in a multi-actor context, we suggest anchoring the content of the local ledger in a public blockchain, such as Ethereum 2.0, for example. To perform this, a transaction including the digest of the recent history of the local ledger is transmitted from a verifying peer to a host device on the IT network, encapsulated in a message with high privilege. The host's role is then to send the transaction to Ethereum 2.0 via the standard Ethereum Client protocol.

8. Conclusions and Future Work

The alert detection and propagation approach presented in this paper builds on the HistoTrust platform [9], leveraging the presence of blockchain technology. Blockchain connects all devices on the network without intermediaries and enables distributed infor-

mation dissemination through the use of smart contracts. The use case considered is a bit-flip attack targeting the weights of the NN model embedded in the ARM Cortex-M4.

The attack is detected immediately at the targeted device, as soon as the first bit is affected within an embedded architecture based on hardware security components. The experiments described in this paper focus on the delay in propagating the alert via the smart contract versus the delay at which the attack will impact the AI's inference result.

Blockchain provides an infrastructure that records events and allows information to be disseminated through the use of smart contracts based on an event emission mechanism. It is an irreplaceable tool for scheduling and maintaining an immutable history of events. Its main limitation lies in the latency of recording in the ledger. This latency can be reduced by reducing the interval between consecutive blocks that constitute the ledger.

In a real-world scenario, the defenses already deployed on the industrial network limit the attacker's options, preventing data leaks. With little prior knowledge of the embedded software architecture of the devices involved, a bit-flip attack will only have an impact after several relevant bits on the same device have been affected.

The study presented in this paper demonstrates the value of a blockchain that operates on a distributed network as a complement to centralized defense infrastructures. The experiments conducted make it possible to scale the various elements of the system in order to reduce the attacker's chances and slow down the attack. The bit-flip attack was chosen for this study, and other attacks that could affect an industrial network may be considered in the future.

Author Contributions: Conceptualization, A.P., D.P. and C.H.; Methodology, A.P., D.P. and C.H.; Software, A.P., D.P. and C.H.; Validation, A.P., D.P. and C.H.; Investigation, A.P., D.P. and C.H.; Writing—original draft preparation, A.P., D.P. and C.H.; Writing—review & editing, A.P., D.P. and C.H.; Project administration, C.H.; Funding acquisition, C.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is a collaborative research action that is supported by the French National Research Agency (ANR) in the framework of the "investissements d'avenir" program ANR-10-AIRT-05, irtnanoelec.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: During the preparation of this manuscript, the authors used buildroot v2020.08, linux kernel v5.4, uboot v2020.01, optee-os v3.9.0 and arm_trusted_firmware v2.2 to build the board support package of the electronic board STM32MP157f-DK2. A computer with Linux Mint 22LTS, with python 3.12 was used for the data analysis. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ICS	Industrial Control System
IDS	Intrusion Detection System
IT	Information Technology
LLM	Large Language Models
NN	Neural Network
OT	Operational Technology
PCR	Platform Configuration Registers
PLC	Programmable Logic Controller

PKI	Public Key Infrastructure
RHA	Row Hammer Attack
SPoF	Single Point of Failure
TPM	Trusted Platform Module
TEE	Trusted Execution Environment

References

1. IBM. Cost of a Data Breach Report. 2024. Available online: https://www.ibm.com/reports/data-breach (accessed on 17 July 2025).
2. Liang, G.; Weller, S.R.; Zhao, J.; Luo, F.; Dong, Z.Y. The 2015 Ukraine Blackout: Implications for False Data Injection Attacks. *IEEE Trans. Power Syst.* **2017**, *32*, 3317–3318. [CrossRef]
3. Lin, C.S.; Qu, J.; Saileshwar, G. GPUHammer: Rowhammer Attacks on GPU Memories are Practical. In Proceedings of the 34th USENIX Conference on Security Symposium (SEC), Seattle, WA, USA, 13–15 August 2025.
4. Cheng, Qian; Zhang, M.N.Y.L.S.C.H. A survey of bit-flip attacks on deep neural network and corresponding defense methods. *Electronics* **2023**, *12*, 853. [CrossRef]
5. Javaheripi, M.; Koushanfar, F. HASHTAG: Hash Signatures for Online Detection of Fault-Injection Attacks on Deep Neural Networks. In Proceedings of the IEEE/ACM International Conference On Computer Aided Design (ICCAD), Munich, Germany, 1–4 November 2021. [CrossRef]
6. Javaheripi, M.; Chang, J.W.; Koushanfar, F. AccHashtag: Accelerated Hashing for Detecting Fault-Injection Attacks on Embedded Neural Networks. *J. Emerg. Technol. Comput. Syst.* **2022**, *19*, 1–20. [CrossRef]
7. Li, S.; Xue, M.; Zhang, Z.; Wu, W. Yes, One-Bit-Flip Matters! Universal DNN Model Inference Depletion with Runtime Code Fault Injection. In Proceedings of the 33rd USENIX Security Symposium, Philadelphia, PA, USA, 14–16 August 2024; pp. 1315–1330.
8. Tabassi, E. Artificial Intelligence Risk Management Framework (AI RMF 1.0). 2023. Available online: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=936225 (accessed on 10 December 2025).
9. Paulin, D.; Joud, R.; Hennebert, C.; Moëllic, P.A.; Franco-Rondisson, T.; Jayles, R. HistoTrust: Tracing AI behavior with secure hardware and blockchain technology. *Ann. Telecommun.* **2023**, *78*, 413–427. [CrossRef]
10. Paulin, D.; Hennebert, C.; Franco-Rondisson, T.; Jayles, R.; Loubier, T.; Collado, R. HistoTrust: Ethereum-Based Attestation of a Data History Built with OP-TEE and TPM. In Proceedings of the Foundations and Practice of Security (FPS), Paris, France, 7–10 December 2021; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2022; Volume 13291, pp. 130–145. [CrossRef]
11. Yan, X.; Qiu, H.; Zhang, T. ObfusBFA: A Holistic Approach to Safeguarding DNNs from Different Types of Bit-Flip Attacks. *arXiv* **2025**, arXiv:2506.10744.
12. van der Veen, V.; Fratantonio, Y.; Lindorfer, M.; Gruss, D.; Maurice, C.; Vigna, G.; Bos, H.; Razavi, K.; Giuffrida, C. Drammer: Deterministic Rowhammer Attacks on Mobile Platforms. In Proceedings of the SIGSAC Conference on Computer and Communications Security (CCS), Vienna, Austria, 24–28 October 2016; pp. 1675–1689. [CrossRef]
13. Dragos. PIPEDREAM: Chernovite's Emerging Malware Targeting Industrial Control Systems. 2022. Available online: https://hub.dragos.com/whitepaper/chernovite-pipedream (accessed on 17 July 2025).
14. MITRE Corporation. Industroyer2-Software S1072. 2023. Available online: https://attack.mitre.org/software/S1072/ (accessed on 17 July 2025).
15. Cañas, H.; Mula, J.; Díaz-Madroñero, M.; Campuzano-Bolarín, F. Implementing Industry 4.0 principles. *Comput. Ind. Eng.* **2021**, *158*, 107379. [CrossRef]
16. Yang, S.J.; Holsopple, J.; Sudit, M. Evaluating Threat Assessment for Multi-Stage Cyber Attacks. In Proceedings of the IEEE Military Communications Conference (MILCOM), Washington, DC, USA, 23–25 October 2006; pp. 1–7. [CrossRef]
17. Langner, R. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Secur. Priv.* **2011**, *9*, 49–51. [CrossRef]
18. NOZOMI Networks. TRITON: The First ICS Cyber Attack on Safety Instrument Systems. 2018. Available online: https://www.nozominetworks.com/resources/triton-the-first-ics-cyber-attack-on-safety-instrument-systems (accessed on 17 July 2025).
19. Leemann, T.; Prenkaj, B.; Kasneci, G. Is My Data Safe? Predicting Instance-Level Membership Inference Success for White-box and Black-box Attacks. In Proceedings of the Next Generation of AI Safety Workshop, Vienna, Austria, 26 July 2024.
20. Chen, Y.; Yuan, Y.; Liu, Z.; Hu, S.; Li, T.; Wang, S. BitShield: Defending Against Bit-Flip Attacks on DNN Executables. In Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 24–28 February 2025. [CrossRef]
21. Rakin, A.S.; He, Z.; Li, J.; Yao, F.; Chakrabarti, C.; Fan, D. T-BFA: Targeted Bit-Flip Adversarial Weight Attack. *arXiv* **2021**, arXiv:2007.12336 . [CrossRef]

22. Ghavami, B.; Sadati, M.; Shahidzadeh, M.; Shannon, L.; Wilton, S. A Semi Black-Box Adversarial Bit-Flip Attack with Limited DNN Model Information. *arXiv* **2024**, arXiv:2412.09450.
23. Wang, Z.; Tang, D.; Wang, X.; He, W.; Geng, Z.; Wang, W. Tossing in the Dark: Practical Bit-Flipping on Gray-box Deep Neural Networks for Runtime Trojan Injection. In Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, 14–16 August 2024; pp. 1331–1348.
24. Rakin, A.S.; He, Z.; Fan, D. Bit-Flip Attack: Crushing Neural Network with Progressive Bit Search. *arXiv* **2019**, arXiv:1903.12269 . [CrossRef]
25. Bai, J.; Wu, B.; Zhang, Y.; Li, Y.; Li, Z.; Xia, S.T. Targeted Attack against Deep Neural Networks via Flipping Limited Weight Bits. *arXiv* **2021**, arXiv:2102.10496. [CrossRef]
26. Kim, Y.; Daly, R.; Kim, J.; Fallin, C.; Lee, J.H.; Lee, D.; Wilkerson, C.; Lai, K.; Mutlu, O. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. *ACM SIGARCH Comput. Archit. News* **2014**, *42*, 361–372. [CrossRef]
27. Li, J.; Rakin, A.S.; He, Z.; Fan, D.; Chakrabarti, C. RADAR: Run-time Adversarial Weight Attack Detection and Accuracy Recovery. In Proceedings of the IEEE Design, Automation, Test in Europe Conference Exhibition (DATE), Grenoble, France, 1–5 February 2021; pp. 790–795. [CrossRef]
28. Eslami, M.; Liu, Y.; Ullah, S.; Salehi, M.E.; Hosseini, R.; Ahmad Mirsalari, S.; Kumar, A. MONO: Enhancing Bit-Flip Resilience With Bit Homogeneity for Neural Networks. *IEEE Embed. Syst. Lett.* **2024**, *16*, 333–336. [CrossRef]
29. Liang, C.; Shanmugam, B.; Azam, S.; Karim, A.; Islam, A.; Zamani, M.; Kavianpour, S.; Idris, N.B. Intrusion Detection System for the Internet of Things Based on Blockchain and Multi-Agent Systems. *Electronics* **2020**, *9*, 1120. [CrossRef]
30. AI-Ajlan, M.; Ykhlef, M. DeepBlock: A Collaborative Intrusion Detection Framework Based on Blockchain and Deep Learning. In Proceedings of the 5th International Conference on Blockchain Computing and Applications (BCCA), Kuwait, Kuwait, 24–26 October 2023; pp. 180–185. [CrossRef]
31. Gupta, R.K.; Chawla, V.; Pateriya, R.K.; Shukla, P.K.; Mahfoudh, S.; Shah, S.B.H. Improving Collaborative Intrusion Detection System Using Blockchain and Pluggable Authentication Modules for Sustainable Smart City. *Sustainability* **2023**, *15*, 2133. [CrossRef]
32. Koumidis, K.; Kolios, P.; Ellinas, G.; Panayiotou, C.G. Secure Event Logging Using a Blockchain of Heterogeneous Computing Resources. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6. [CrossRef]
33. Nassar, M.E.B.; Salah, K.; Rehman, M.H.U.; Svetinovic, D. Blockchain for explainable and trustworthy artificial intelligence. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2019**, *10*, e1340. [CrossRef]
34. Malhotra, D.; Srivastava, S.; Saini, P.; Singh, A.K. Blockchain based audit trailing of XAI decisions: Storing on IPFS and Ethereum Blockchain. In Proceedings of the International Conference on COMmunication Systems & NETworkS (COMSNETS), Bangalore, India, 5–9 January 2021; pp. 1–5. [CrossRef]
35. Aich, S.; Sinai, N.K.; Kumar, S.; Ali, M.; Choi, Y.R.; Joo, M.I.; Kim, H.C. Protecting Personal Healthcare Record Using Blockchain & Federated Learning Technologies. In Proceedings of the 24th International Conference on Advanced Communication Technology (ICACT), Pyeongchang, Republic of Korea, 13–16 February 2021; pp. 109–112. [CrossRef]
36. Short, A.R.; Leligou, H.C.; Papoutsidakis, M.; Theocharis, E. Using Blockchain Technologies to Improve Security in Federated Learning Systems. In Proceedings of the IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 1183–1188. [CrossRef]
37. Zhang, J.; Liu, Y.; Qin, X.; Xu, X.; Zhang, P. Adaptive Resource Allocation for Blockchain-Based Federated Learning in Internet of Things. *IEEE Internet Things J.* **2023**, *10*, 10621–10635. [CrossRef]
38. Prathiba, S.B.; Govindarajan, Y.; Pranav Amirtha Ganesan, V.; Ramachandran, A.; Selvaraj, A.K.; Kashif Bashir, A.; Reddy Gadekallu, T. Fortifying Federated Learning in IIoT: Leveraging Blockchain and Digital Twin Innovations for Enhanced Security and Resilience. *IEEE Access* **2024**, *12*, 68968–68980. [CrossRef]
39. Yap, S.K.; Dong, Z.; Toohey, M.; Lee, Y.C.; Zomaya, A.Y. Smart Contract Data Monitoring and Visualization. In Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Dubai, United Arab Emirates, 1–5 May 2023; pp. 1–8. [CrossRef]
40. Ramachandran, G.S.; Wright, K.L.; Zheng, L.; Navaney, P.; Naveed, M.; Krishnamachari, B.; Dhaliwal, J. Trinity: A Byzantine Fault-Tolerant Distributed Publish-Subscribe System with Immutable Blockchain-based Persistence. In Proceedings of the International Conference on Blockchain and Cryptocurrency (ICBC), Seoul, Republic of Korea, 14–17 May 2019; pp. 227–235. [CrossRef]
41. Shinde, R.; Patil, S.; Kotecha, K.; Ruikar, K. Blockchain for Securing AI Applications and Open Innovations. *J. Open Innov. Technol. Mark. Complex.* **2021**, *7*, 189. [CrossRef]
42. Kapengut, E.; Mizrach, B. An Event Study of the Ethereum Transition to Proof-of-Stake. *Commodities* **2023**, *2*, 96–110. [CrossRef]
43. Kushwaha, S.S.; Joshi, S.; Singh, D.; Kaur, M.; Lee, H.N. Ethereum Smart Contract Analysis Tools: A Systematic Review. *IEEE Access* **2022**, *10*, 57037–57062. [CrossRef]

44. Bodkhe, U.; Tanwar, S.; Parekh, K.; Khanpara, P.; Tyagi, S.; Kumar, N.; Alazab, M. Blockchain for Industry 4.0: A Comprehensive Review. *IEEE Access* **2020**, *8*, 79764–79800. [CrossRef]
45. Javaid, M.; Haleem, A.; Pratap Singh, R.; Khan, S.; Suman, R. Blockchain technology applications for Industry 4.0: A literature-based review. *Blockchain Res. Appl.* **2021**, *2*, 100027. [CrossRef]
46. Stodt, J.; Reich, C. Data Confidentiality in P2P Communication and Smart Contracts of Blockchain in Industry 4.0. In Proceedings of the Computer Science Information Technology, 10th International Conference on Computer Science, Engineering and Applications (CCSEA 2020), London, UK, 25–26 July 2020; AIRCC Publishing Corporation: Chennai, India , 2020; pp. 1–10. [CrossRef]
47. Shifferaw, Y.; Lemma, S. Limitations of proof of stake algorithm in blockchain: A review. *Zede J. Ethiop. Eng. Archit.* **2021**, *39*, 81–95.
48. ethPandaOps. Ethereum Private Blockchain. 2025. Available online: https://github.com/ethpandaops/ethereum-package (accessed on 17 July 2025).
49. Kurtosis. 2025. Available online: https://github.com/kurtosis-tech/kurtosis (accessed on 17 July 2025).
50. Duo, W.; Zhou, M.; Abusorrah, A. A Survey of Cyber Attacks on Cyber Physical Systems: Recent Advances and Challenges. *CAA J. Autom. Sin.* **2022**, *9*, 784–800. [CrossRef]
51. Apache Software Foundation. Apache Kafka. 2025. Available online: https://kafka.apache.org (accessed on 17 July 2025).
52. V, S.; A, V.; Pattar, S. MQTT based Secure Transport Layer Communication for Mutual Authentication in IoT Network. *Glob. Transitions Proc.* **2022**, *3*, 60–66. [CrossRef]
53. Meng, W.; Tischhauser, E.W.; Wang, Q.; Wang, Y.; Han, J. When Intrusion Detection Meets Blockchain Technology: A Review. *IEEE Access* **2018**, *6*, 10179–10188. [CrossRef]
54. Abadi, M.e.a. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. *arXiv* **2016**, arXiv:1603.04467.
55. Lecun, Y.; Cortes, C.; Burges, C.J. MNIST Handwritten Digit Database. 2010. Available online: http://yann.lecun.com/exdb/mnist (accessed on 10 December 2025).