Security and Privacy

WILEY

RESEARCH ARTICLE OPEN ACCESS

Towards Sustainable IoT: A Digital Signature-Enhanced Federated Learning Approach

Shahida Hafeezan Qureshi[1] | Saif Ur Rehman Malik[1,2] | Junaid Haseeb[3] | Syed Atif Moqurrab[4] | Tanvir Fatima Naik Bukht[5] | Gautam Srivastava[6,7]

[1]Department of Computer Science, COMSATS University Islamabad, Islamabad, Federal Capital Territory, Pakistan | [2]School of Computer Science and Statistics, Trinity College Dublin, The University of Dublin, Dublin, Leinster, Ireland | [3]Department of Computer Science, University of Waikato, Hamilton, Waikato Region, New Zealand | [4]School of Computer Science and Technology, Luton, Bedfordshire, UK | [5]Department of Computer Science, Air University, Islamabad, Federal Capital Territory, Pakistan | [6]Department of Mathematics and Computer Science, Brandon University, Brandon, Manitoba, Canada | [7]Research Centre for Interneural Computing, China Medical University, Taichung, Taiwan

Correspondence: Syed Atif Moqurrab (syedatif.moqurrab@beds.ac.uk) | Gautam Srivastava (srivastavag@brandonu.ca)

Received: 6 March 2025 | **Revised:** 29 May 2025 | **Accepted:** 24 June 2025

Funding: The authors received no specific funding for this work.

Keywords: cryptography | ECC | ECDSA | federated learning | global model security | IoV | RSA

ABSTRACT

Federated Learning (FL) is emerging as a premier paradigm for privacy-preserved Machine Learning (ML), enabling devices to train models without central data pooling collaboratively. In the contemporary Internet of Things (IoT) landscape, characterized by escalating energy consumption and associated carbon footprint, FL is recognized not merely for its privacy features. Intrinsic to decentralized architectures such as FL, secure communication is based on digital signatures to guarantee integrity. This is particularly evident in sensitive sectors such as the Internet of Vehicles (IoV), banking, and healthcare. Integrating FL becomes imperative and intricate as these sectors are intertwined with the IoT fabric. Our study unveils "Secure Federated Learning Framework (SecFL)," a pioneering decentralized framework combining FL and sustainable computing. SecFL offers defences against adversarial attacks such as data poisoning and label flipping. Utilizing the Rivest-Shamir-Adleman (RSA) asymmetric encryption algorithm for securing digital communications and transactions, combined with ElGamal encryption and a private Ethereum blockchain, ensures enhanced client-specific security. Our research emphasizes the formal modeling of adversarial dynamics using High-Level Petri nets (HLPN) within the FL-IoT ecosystem, balancing system dynamics and energy conservation. Our model consistently outperforms contemporary solutions in accuracy and time efficiency after validation. As IoT burgeons into domains like environmental monitoring, smart cities, and energy grids, the SecFL framework, fostering FL, optimizes energy utilization and bolsters resource efficiency. In our comparative analysis, the Elliptic Curve Digital Signature Algorithm (ECDSA) algorithm demonstrates superior transaction latency and verification time compared to RSA and Elliptic Curve Cryptography (ECC).

1 | Introduction

The proliferation of the Internet of Things (IoT) shows both unparalleled opportunities and associated challenges. As billions of devices interlink, exchanging and processing data, the potential for revolutionizing sectors from healthcare to transportation is evident. Yet, this interconnected surge raises concerns over energy consumption, carbon footprint, and data privacy. Amidst this backdrop, Federated Learning (FL) stands as a promising beacon. This Machine Learning (ML) paradigm facilitates collaboration across diverse entities to hone a shared model, all while ensuring data remains localized, never migrating to a centralized

server or other participants. The essence of FL revolves around decentralization: Model updates journey to the central hub for aggregation, and the refined model is dispatched back to participants. This iterative progression persists until the model reaches a desired fidelity. However, the journey is strewn with challenges. While FL offers an antidote to the energy drain of centralized data exchange, it confronts potential data privacy breaches during model training. The sheer diversity of devices in FL amplifies the attack surface, making data privacy a paramount concern [1].

Existing research underscores the tenacity of adversarial attacks, especially within decentralized ecosystems. Tools like differential privacy, designed to cloak individual data points in vast datasets, grapple with integration challenges in FL. Crafty adversaries equipped with subtle strategies can induce perturbations that manipulate ML models. Given FL's applicability in sensitive sectors like healthcare and finance, even minute lapses can lead to serious consequences, necessitating robust defenses against threats like label flipping and data poisoning [2]. To address these concerns, FL enthusiasts are converging on an array of privacy-enhancing and energy-conserving strategies. Encryption emerges as a linchpin, safeguarding data during transmission and simultaneously curbing energy overheads. The surge in trusted execution environments, differential privacy, and Elliptic curve cryptography (ECC) represents a sophisticated approach to public-key cryptography, founded on the algebraic structure of elliptic curves over finite fields. Furthermore, Digital Signature encryption significantly enhances the security framework of FL. [3, 4]. Yet, as illuminated by Malik et al. [1], the path is fraught with challenges. ECC, despite its merits, is vulnerable due to its concise key sizes, making it susceptible to Pollard RHO attacks. The fusion of DS encryption with the Pollard-RHO technique, renowned for addressing large integer challenges, beckons further exploration [5, 6].

We present the Secure Federated Learning Framework (SecFL), a cutting-edge decentralized architecture specifically designed to enhance the security of FL. SecFL melds the Elliptic Curve Digital Signature Algorithm (ECDSA) with a private Ethereum blockchain, augmenting it with cryptographic protocols to fortify global parameters, safeguard client data, and authenticate clients. Striking a balance between decentralized parameters and energy thriftiness, we envision a paradigm that embodies security and sustainability. Our subsequent exploration, underpinned by High-Level Petri nets (HLPN), offers a mathematical perspective to dissect the dynamic interplay between adversarial and defensive entities [7]. Existing discourses pinpoint the gap in ECC vulnerability analysis. Our endeavor successfully orchestrates a Pollard RHO attack on ECC, demonstrating a model that champions ECDSA security and resists threats. Empirical evaluations on datasets from banks, hospitals, and IoVs corroborate our model's computational cost, accuracy, and security efficacy.

In summation, our contributions include

- A comprehensive review of contemporary research, spotlighting existing gaps.
- A robust security blueprint for global model parameters, tailored for IoT-centric environments, healthcare realms, and financial infrastructures.
- A counterstrategy to thwart Pollard RHO vulnerabilities in ECC, ensuring an ecosystem that harmoniously combines privacy with sustainability.
- A blockchain-driven framework for decentralized client validation, anchoring its strengths in digital signature-centric security and energy conservation.

Subsequent sections of this paper are structured as follows: Section 2 delves into related works. Section 3 shows the primary study, Section 4 analyzes the attacker model, and Section 5 elucidates the architecture of the proposed model in detail. Section 6 discusses the implementation of SecFL ventures into the Formal Modeling of HLPN. Section 7 offers a comparative analysis and discusses implementation results. Section 8 concludes the paper.

2 | Literature Evaluation

In this section, we review the existing research, primarily focusing on the vulnerability of ECC, especially through the use of the Pollard RHO algorithm. Additionally, this literature review emphasizes the security features of the ECDSA. Table 1 summarizes our findings.

In recent years, the adoption of ECC and ECDSA to enhance the security and efficiency of decentralized systems has increased. Authors like Genc [16] have identified ECC vulnerabilities and proposed countermeasures against fault attacks. In conclusion, while significant advancements have been made in ensuring decentralized systems' security and ML models' security, the existing vulnerabilities, especially those related to ECC and the Pollard RHO algorithm, need urgent attention and mitigation.

3 | Preliminary Study

This section discusses the key concepts utilized in our proposed methodology. These foundational insights will help readers understand the subsequent sections of the study.

3.1 | Federated Infrastructure

FL-based frameworks enhance collaborative learning of a shared model among devices without necessitating the exchange of confidential data, boosting both privacy and security. Traditional centralized ML approaches typically gather and transfer raw data to a central server for training. Such practices can result in significant communication overhead and delays, especially when dealing with large datasets or bandwidth-constrained devices. FL-based frameworks help mitigate these challenges by minimizing the data transmitted across the network, leading to reduced communication costs and latency [21].

Our proposed model secures federated learning with an elliptic curve digital signature. In Figure 1, the task publisher is tasked with defining and disseminating the ML objective to the FL clients. The SecFL framework permits a client or an external entity to act as the task publisher, coordinating with the main system to distribute tasks. FL clients contribute local data

TABLE 1 | This table discusses the existing research on the vulnerability of ECC using the Pollard Rho algorithm. Meanwhile, it emphasizes the security features of the ECDSA.

Authors	Year	Methodology	Achievements	Weaknesses	Limitations
R. Malik [1]	2023	Blockchain RSA ECC Algamal	Decentralized framework privacy and security	Vulnerable by Pollard RHO	Small ECC key size less computational cost.
R. Jin [8]	2022	Federated learning GAN	High performance and privacy	A malicious actor can compromise the ML model by submitting a tainted change to the system.	Attackers can upload a poisoned update to the server to alter the global model.
N. Yuvaraj [9]	2022	Literature review and analysis	Provides a comprehensive review of the (ECDSA)	Lack of empirical evidence or experimentation	Limited to the use of ECC.
F. Azam [10]	2021	Literature review and analysis	Provides a detailed analysis of the strengths and weaknesses of ECC	No empirical evidence or experimentation	Limited to the use of ECC.
U. Ghosh [11]	2021	Blockchain Federated learning	High performance and privacy	There is no data privacy between vehicles when they communicate. Transporter-to-transporter data transmissions are not private.	
Z. Wang [12]	2021	Federated learning	percentage of successful assaults	On the commonly shared models, there is the possibility of a model poisoning assault.	Globally shared models are vulnerable to model poisoning attacks.
M. S. Rahman [13]	2021	Security concerns with blockchain	Comprehensive overview of various security issues and challenges faced by blockchain-based systems	The failure to evaluate suggested solutions empirically and the inability to focus on particular applications or use cases	Disagreement on how to tackle security issues effectivelfy.
A. K. Yadav [14]	2021	Experimental	Higher security and efficiency compared to traditional algorithms	Limited experimental data	Limited scope of IoT devices tested.
S. Shukla [15]	2021	Experimental	Higher security and efficiency compared to using either ECC or RSA alone	Not appropriate for resource-constrained environments	Not appropriate for resource-constrained environments.
Y. Genc [16]	2021	Experimental research using fault injection techniques	Identified vulnerabilities in ECC and proposed countermeasures against fault attacks	Limited to fault attacks and may not cover other types of attacks	Higher security and efficiency compared to using either ECC or RSA alone.
W. Liang [17]	2020	Smart contracts, blockchain	Immutable, tamper-evident data storage and sharing, decentralized, improved data privacy and security	Excessive data storage and compute requirements, time and resource costs, and a slow transaction processing rate.	Low scalability, high latency, and inefficient energy consumption

(Continues)

TABLE 1 | (Continued)

Authors	Year	Methodology	Achievements	Weaknesses	Limitations
T. Duong [18]	2020	Blockchain and multi-agent reinforcement learning	Decentralized, secure, and transparent multi-agent reinforcement learning improved trust and privacy	Low scalability and a heavy workload	Due to the significant processing overhead of blockchain-based consensus techniques, there is limited scalability.
C. He and G. Liu [19]	2020	Federated learning SMC	Precision and response time	SMC's server-side Label flipping attack and high communication and calculation overhead are drawbacks.	The SMC has high communication and calculation costs, and there is a risk of Label flipping attacks from the server side.
T. Daisy Premila Bai [20]	2017	Theoretical	Provides a comprehensive analysis of security threats and their impact	No experimental validation	Limited analysis of security threats.

to the training process. Each client's dataset undergoes training on its respective device, and subsequently, model updates are forwarded to the central system for aggregation. The SecFL framework encompasses modules for data preparation, model training, and model updates. Crucially, FL enables collaboration without a centralized data repository, placing reliance on the client's end. In the SecFL system, client model updates are digitally signed to assure authenticity and data integrity. Moreover, Digital Signature provides additional protection for sensitive local data. A designated model aggregator does the aggregation of model updates from various clients. This aggregator processes the individual models into a consolidated global model relayed to the clients. Within the SecFL framework, DS ensures that the aggregator can function without directly accessing users' raw data. The aggregator employs a validation entity to verify model updates from clients. This setup, intrinsic to FL, facilitates collaborative model training without a centralized data repository, further fortifying data privacy.

3.2 | Rivest-Shamir-Adleman Algorithm

The Rivest-Shamir-Adleman algorithm (RSA), a renowned public-key encryption algorithm, offers the potential for secure FL [22]. This study introduces a secure decentralized system anchored on FL and the ECDSA. While RSA can be leveraged like ECDSA to safeguard the FL process, it is noteworthy that RSA employs a distinct mathematical approach to attain comparable outcomes. Both ECDSA and RSA generate key pairs consisting of a private and a public key, facilitating the creation and verification of digital signatures. RSA's computational complexity is a drawback, especially when dealing with large key sizes. Such complexity can impede the FL process, especially when numerous clients and data sources are involved [23]. Our proposed architecture prefers ECDSA due to its rapid execution and robust security features. Desired security levels, available computational resources, and specific use cases often influence the choice between ECDSA and RSA. While we've showcased the efficacy of ECDSA in securing FL, RSA remains a commendable alternative for similar applications.

3.3 | The Elliptic Curve Digital Signature Algorithm

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a public-key cryptography technique developed to generate digital signatures, ensuring data and communication security. Within FL, ECDSA authenticates model updates shared between clients and servers. ECDSA produces a pair of private and public keys. The private key remains confidential with the owner, while the public key is disseminated. Messages are signed using private keys and verified with public keys. In FL, ECDSA plays a pivotal role in verifying model updates. Absent such a mechanism, malicious clients might send spurious updates or interfere with genuine updates, potentially leading to flawed or compromised models. ECDSA boasts several advantages over traditional digital signature algorithms, especially regarding security. It is computationally efficient, features a compact key size—ideal for mobile devices—and offers resilience against signature forgery and key recovery attacks [24].

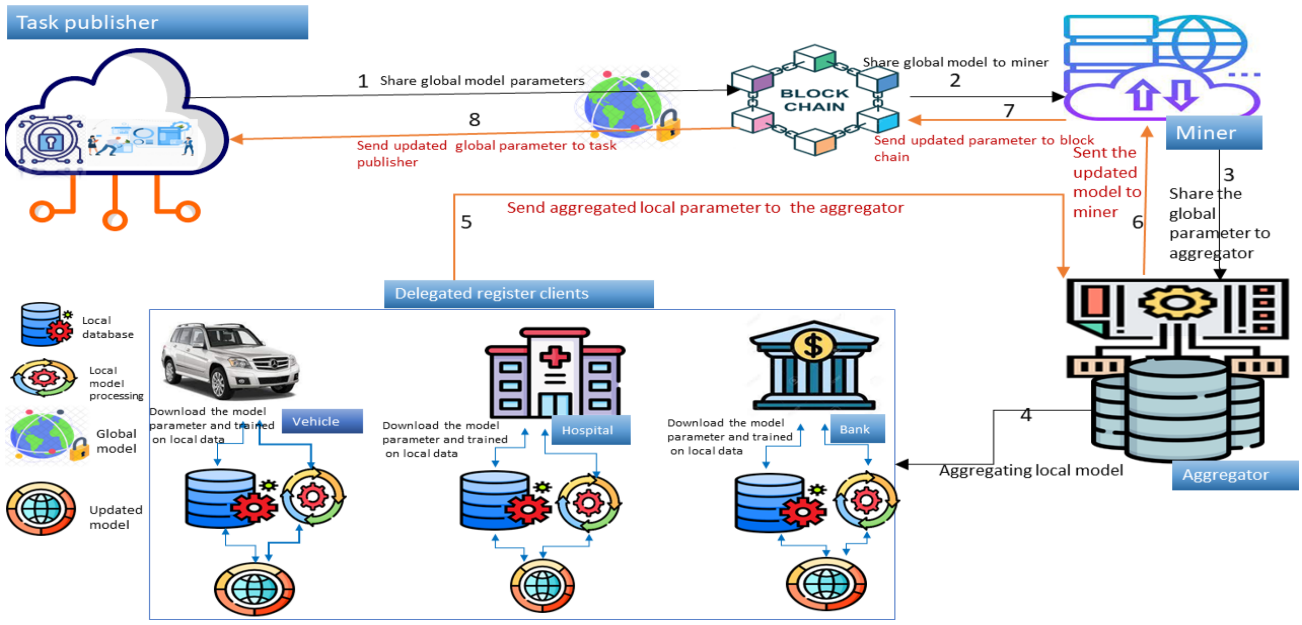

FIGURE 1 | Secure communication with clients using global parameters in the SecFL Network Model.

4 | Exploiting ECC With Pollard RHO

We executed the Pollard RHO attack, targeting the specific properties of the elliptic curve integral to the ECC technique. This attack facilitates the extraction of the private key using a compromised public key. Pollard's RHO algorithm is a popular choice for addressing the discrete logarithm problem inherent to ECC, which underpins many cryptographic systems [25]. An attacker can employ various strategies, such as label flipping and data poisoning, to compromise the integrity of the foundational dataset. Label flipping involves manipulating the data labels to misguide the model during training. On the other hand, data poisoning modifies the input data, leading the model to produce inaccurate predictions or outcomes. The subsequent section elaborates on the step-by-step implementation of the Pollard RHO attack.

4.1 | Analysis of Attack Detection in ECC Using Pollard RHO

The Pollard-RHO method offers a solution to the discrete logarithm problem on elliptic curves. It searches for a recurring point on the curve by stochastically cycling through points and curve sequences. Once such a recurring point is identified, it can be used to compute the discrete logarithm. Pollard RHO holds certain advantages over other methods addressing the discrete logarithm problem. By identifying recurring processes, the algorithm identifies curves. Its parallel execution across multiple processors diminishes computation durations and augments the likelihood of finding a solution. The method begins by randomly selecting two points on the curve and generating a series of points. It employs collision detection to spot sequence cycles, which in turn aids in solving the discrete logarithm problem. Distributing this computation across various devices further trims down the required time. Pollard's RHO Attack offers superior performance for ECC compared to other discrete logarithm techniques. A few notable advantages include:

- *Efficiency*: Parallel processing accelerates computations and bolsters the chances of a successful solution.
- *Memory Conservation*: The technique requires less memory than traditional methods, favoring devices with limited memory capacity.
- *Resistance to Side-Channel Attacks*: The approach is fortified against side-channel cryptographic attacks that exploit vulnerabilities in cryptographic system defenses.

Pollard's RHO Attack adeptly and efficiently tackles the discrete logarithm problem on elliptic curves.

4.2 | Compromising Secure Private Blockchain (Secprivchain) Security With Pollard RHO

The presented process, as depicted in Figure 2, outlines the structure of our framework. Within this structure, Algorithm 1 is proposed for the SecurePrivChain model, aiming to train global ML models securely and privately. This algorithm accepts a set of clients (K), encrypted model parameters (*Go*), and a designated number of iterations (T) as input. After T iterations, the output is the trained global model parameters (*Gt*).

The algorithm divides the points on an elliptic curve into three distinct subsets (S1, S2, and S3) while defining initial random values for two sequences (a and b) slated for subsequent iteration. It then generates a pseudorandom sequence of points on the elliptic curve, utilizing the initial values of a, b, and the subsets S1, S2, and S3. A cycle-finding mechanism detects the onset of repetitive sequences or cycles, identifying such occurrences as "collisions."

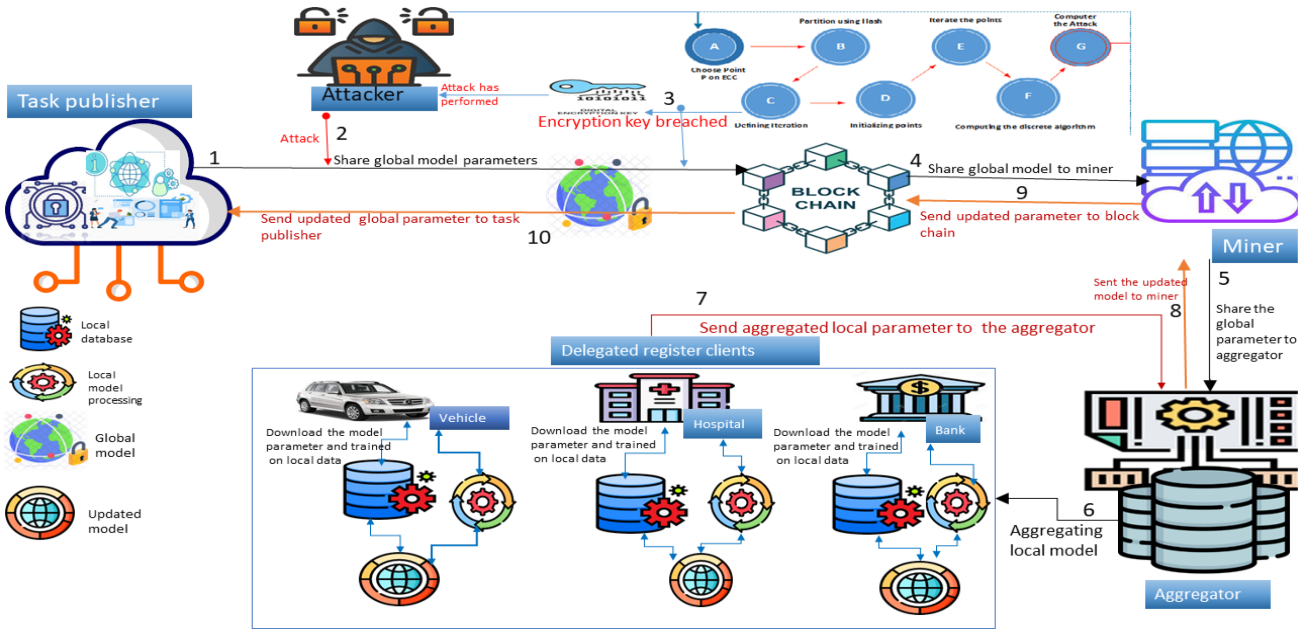

FIGURE 2 | The ECC attacker model devised by Pollard RHO, sending the global parameter susceptible to attacks.

ALGORITHM 1 | SecurePrivChain Vulnerability by Pollard's RHO Algorithm.

Require: K, G_0, T
Ensure: G_t
$TP \leftarrow G_0$ ▷ *Initialize global parameters on B-chain*
Divide elliptic curve into subsets: S_1, S_2, S_3
Define initial values a_i and b_i ▷ *To be iterated*
Set $X_0 \leftarrow a_0 P + b_0 Q$ ▷ *Initial point on the curve*
Generate points $X_1, X_2, \ldots$
for $i = 1$ to T **do**
  if $X_i \in S_1$ **then**
    $X_{i+1} \leftarrow X_i + x_a P$
  else if $X_i \in S_2$ **then**
    $X_{i+1} \leftarrow 2X_i$
  else if $X_i \in S_3$ **then**
    $X_{i+1} \leftarrow X_i + x_b Q$
  end if
  Check if X_{i+1} has started repeating ▷ *Cycle detection*
end for
for each delegated client in K **do**
  for each shared parameter $i \in K$ **do**
    $[w_i]^{(k)} \leftarrow [G_{t-1}]$ ▷ *Fetch previous weights*
    Encrypt $[w]^{(k)} \rightarrow \Omega_k$
    while $k > 0$ **do**
      $\Omega_{k+1} \rightarrow G_t$
      $G_t \rightarrow TP$ ▷ *Update to global model*
      $k \leftarrow k - 1$
    end while
    if Opt-acc is True **then**
      $G_t \rightarrow TP$ ▷ *Early stop if accuracy is achieved*
      break
    end if
  end for
end for

Upon detecting a collision, the algorithm computes the p value using the derived values of a and b.

Subsequently, the algorithm delegates the training of the global model to each client within set K. Every client decrypts the shared parameters, conducts local model training, and then encrypts the locally trained weights. The miner collects the aggregated weights from each client, updating the global model parameters accordingly. The algorithm verifies whether optimal accuracy has been attained. If achieved, the iteration halts; otherwise, the process recommences with a fresh set of initial values for a and b.

4.3 | Vulnerability Analysis of ECC Encryption Key Against Pollard RHO Attack

The Pollard RHO algorithm presents an effective method for compromising ECC encryption keys. It capitalizes on the realization that computing the discrete logarithm of a point on an elliptic curve is markedly simpler than executing its inverse operation, exponentiation. The algorithm identifies a collision point by generating pseudorandom points on the curve in a continuous sequence, marking the commencement of the sequence repetition. This collision point can be harnessed to discern the private key, which is pivotal for decrypting the encrypted content.

In Algorithm 2, the task publisher encompasses a set of untrained encryption parameters denoted as G_o. These are bifurcated into subsets C_1 and C_2, using the task publisher's public key (P_1, P_{2a}, P). ECC employs the private key P_b to facilitate the encryption of the parameter. The resultant ECC encryption is $C_m = \{KG, G_o, KP_b\}$. Notably, this specific ECC encryption key, $C_m = \{KG, G_o, KP_b\}$, is targeted for breach by Pollard RHO in its fifth step.

5 | SecFL: A Proposed ECDSA-Based Framework

In this section, we are discussing our proposed framework, SecFL, in detail. Our main contribution is ECDSA applied with FL. By combining the ECDSA algorithm with a decentralized framework, this research study proposes an innovative approach for protected FL. ECDSA has three major components: Key generation, message signing, and message verification; these components are discussed below. Figure 3 SecFL ECDSA-based framework involves a server, many clients, and other components like the task publisher, aggregation model, and global model. The ECDSA technique creates and verifies digital signatures, making FL more secure. To authenticate the model's updates before sending them to the server, each client creates a key pair that includes a private key and a public key. The server checks the digital signatures using the client's public keys to confirm that the updates are genuine and unmodified. Our results show that the suggested framework produces high accuracy while protecting the confidentiality and privacy of the client's data. We also demonstrate how the FL procedure is substantially more secure and resistant to many attacks when ECDSA is used. The suggested method offers a reliable and sustainable approach for collaborating on ML while addressing security and privacy issues related to FL.

ALGORITHM 2 | ECC Encryption key breach by Pollard RHO Algorithm.

Require: let n_a be the private key
Ensure: $n_a < n$
1: let $P_a = n_a \times G$ be the private key
2: **for** 3 subsets, S_1, S_2 and S_3 **do**
3: initial random values a_i and b_i iterated later
4: Let be a_0 and b_0. The initial point X_0 as $a_0P + b_0Q$
5: $X_1, X_2,$ ו
6:
7: $$X_{i+1} = \begin{cases} X_i + x_aP & \text{if } X_i \in S_1 \\ 2X_i & \text{if } X_i \in S_2 \text{ where } x_a \times x_b = 1 \\ X_i + x_bQ & \text{if } X_i \in S_3 \end{cases}$$
8: sequence will repeat
9: **if** a and b **then**
10: $aP + bQ$ must match the value of X
11: a and b iterated later
12: $$a_{i+1} = \begin{cases} a_i & \text{if } X_i \in S_1 \\ 2a_i & \text{if } X_i \in S_2 \\ a_i + x_b & \text{if } X_i \in S_3 \end{cases}$$
13: $$b_{i+1} = \begin{cases} b_i + x_b & \text{if } X_i \in S_1 \\ 2b_i & \text{if } X_i \in S_2 \\ b_i & \text{if } X_i \in S_3 \end{cases}$$
14: **end if**
15: points X_i starts repeating collision, two points X_j and X_k obtained $j = X_k$ and $j \neq k$. The compared iteration are X_j (called the tortoise steps) and X_{2j} (called the hare steps)
16: **if** $a_j = a_k$ and $b_j = b_k$ **then**
17: Start with different values for a_0 and b_0
18: **else**
19: p can be calculated values of a_j, a_k, b_j, and b_k
20: **end if**
21: **end for**

Client
22: Let n_b private key, $n_b < n$
23: Let $P_b = n_b \times G$ public key
24: $KG \times n_b - (1)$
25: Subtract (1) from second part of C_m: $G_o + KP_b - KG \times n_b = G_o$
26: After training parameters are $= G_o$
27: Encrypt again: $G_t = \{KG, G_o + KP_b\}$

Miner
28: Miner aggregates all locally trained weights G_t for each $\Omega_k \in \Omega$ until $k = 0$
29: $\Omega_{k+1} = \Omega, G_t$

TP
30: $G_o + KP_b - KG \times n_b = G_o$ after calculation

5.1 | Key Generation

This section discusses the Key Generation in ECDSA (as shown in Algorithm 3) generates an elliptic curve domain parameter, including the curve equation, base point, and prime modulus. Select a private key, a random integer within a specific range.

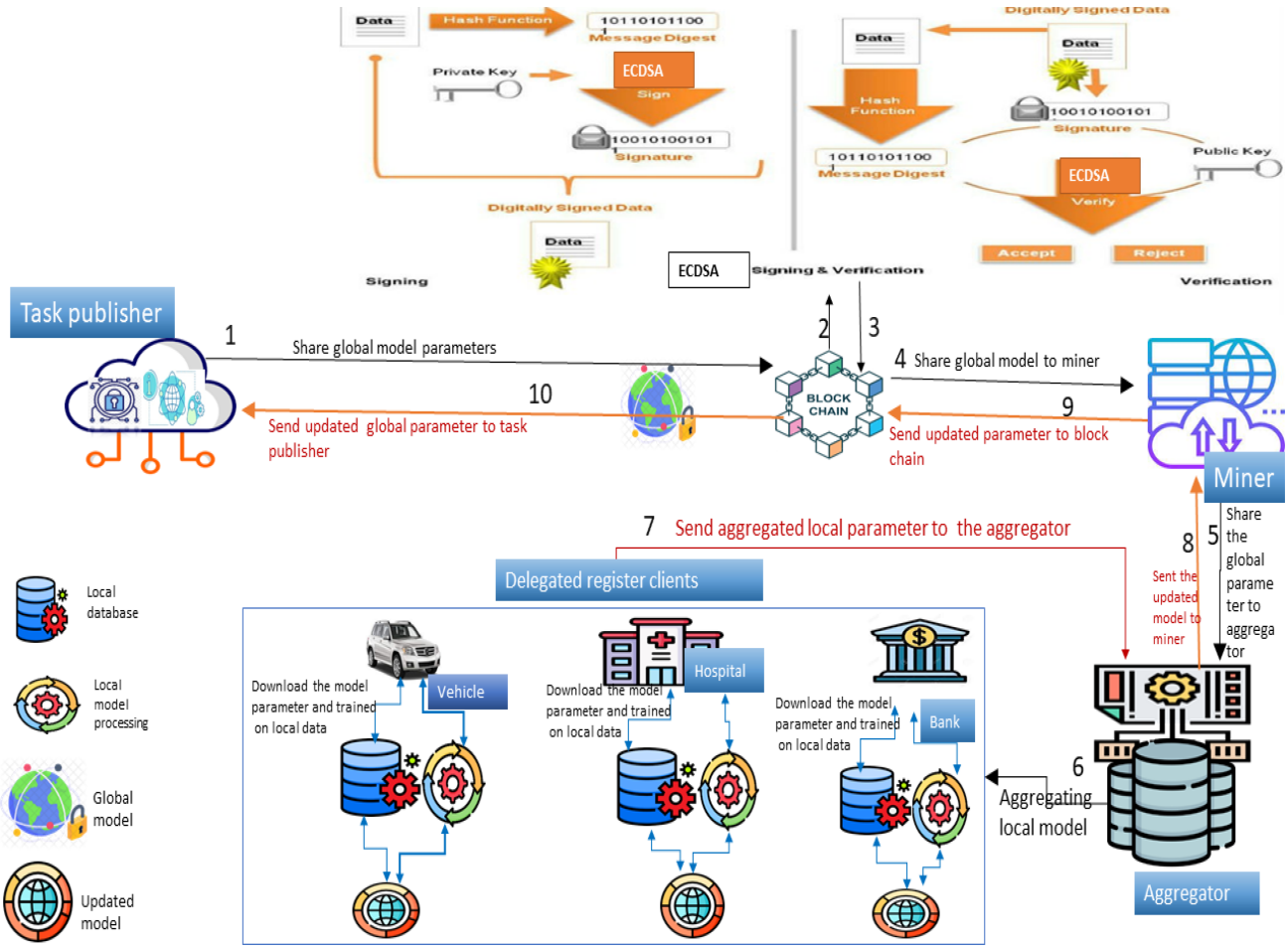

FIGURE 3 | A proposed decentralized secured SecFL model with ECDSA for secure communication with clients.

ALGORITHM 3 | ECDSA in IOV – Key Generation.

```
(E, P, n) = generate_elliptic_curve_domain_parameter()
d = generate_private_key()
Q = d * P
E, P, n = // generate elliptic curve domain parameters
return E, P, n
function GENERATE_PRIVATE_KEY
    d = // generate private key
    return d
end function
function COMPUTE_PUBLIC_KEY(d, P)
    Q = d * P
    return Q
end function
```

Compute the public key by multiplying the private key by the base point using elliptic curve scalar multiplication.

5.2 | Message Signing

In this section, we discussed the Message Signing in ECDSA. As shown in Algorithm 4, generating a hash of the message to be signed using a cryptographic hash function. Choose a random value called the "ephemeral key." Compute the ephemeral key's corresponding public key by multiplying it by the base point. Point 1: Calculate the signature parameters. Compute the value "r" by taking the x-coordinate of the temporary key's public key modulo the prime modulus. Calculate the value "s" using the formula: s = (hash + r * private key) / ephemeral key. The resulting signature is the pair (r, s).

5.3 | Message Verification

In this section, we discussed Message Verification in ECDSA in detail. The Algorithm 5 receives the message and its associated signature. Generate the hash of the received message using the same cryptographic hash function. Verify that the signature parameters "r" and "s" are within a valid range. Compute the value "w" using the formula: $w = s^{-}1$ (mod prime modulus). Calculate two points on the elliptic curve:

Point 1: Multiply the hash value by "w" and compute the result modulo the prime modulus.

Point 2: Multiply the signature parameter "r" by "w" and compute the result modulo the prime modulus.

Compute the resulting point on the elliptic curve by adding Point 1 and Point 2. If the x-coordinate of the resulting point is equal to "r," the signature is valid; otherwise, it is invalid.

ALGORITHM 4 | ECDSA in IOV – Message Signing.

(E, P, n) = generate_elliptic_curve_domain_parameter()
$h \leftarrow$ hash_function(message)
$k \leftarrow$ generate_ephemeral_key()
$K \leftarrow k \cdot P$
$r \leftarrow$ x-coordinate$(K) \mod n$
$s \leftarrow (h + d \cdot r) \cdot \text{modInverse}(k, n) \mod n$
signature $\leftarrow (r, s)$
function GENERATE_EPHEMERAL_KEY
    $k \leftarrow$ // generate ephemeral key
    return k
end function
function COMPUTE_PUBLIC_KEY(k, P)
    $K \leftarrow k \cdot P$
    return K
end function
function COMPUTE_R(K, n)
    $r \leftarrow$ x-coordinate$(K) \mod n$
    return r
end function
function COMPUTE_S(h, r, d, k, n)
    $s \leftarrow (h + d \cdot r) \cdot \text{modInverse}(k, n) \mod n$
    return s
end function
function MODINVERSE(a, m)
    for $i = 1$ to $m - 1$ **do**
        if $(a \cdot i) \mod m == 1$ **then**
            return i
        end if
    end for
end function
function HASH_FUNCTION(message)
    $h \leftarrow$ // apply cryptographic hash function to message
    return h
end function

ALGORITHM 5 | ECDSA in IOV – Message Verification.

Require: message, signature, Q, P, n
Ensure: true or false
$h \leftarrow$ hash_function($message$)
$r \leftarrow$ signature[0]
$s \leftarrow$ signature[1]
if $r \leq 0$ or $r \geq n$ or $s \leq 0$ or $s \geq n$ **then**
    return false
end if
$w \leftarrow \text{modInverse}(s, n)$
$u1 \leftarrow (h \cdot w) \mod n$
$u2 \leftarrow (r \cdot w) \mod n$
$U \leftarrow u1 \cdot P + u2 \cdot Q$
if U is the point at infinity **then**
    return false
end if
if $r =$ x-coordinate$(U) \mod n$ **then**
    return true
else
    return false
end if

6 | Formal Modeling of Adversarial Dynamics in HLPN Models

The complexity of the distributed system encompasses several components and various types of data. The existence of adversarial model parameters and encryption keys suggests that the system might have specific security and privacy requirements. Additionally, global model parameters imply using ML or other predictive modeling techniques within the system. High-Level Petri Nets (HLPN) provide a robust mathematical framework for modeling and analyzing complex systems, which is especially beneficial in federated learning frameworks focused on security. As detailed in [26], HLPNs extend classical Petri nets by incorporating data, types, and hierarchical structures, enabling detailed and scalable representations of distributed and concurrent processes. Security and privacy become paramount concerns in federated learning, where multiple decentralized nodes collaboratively train machine learning models without sharing raw data. By leveraging HLPNs, it is possible to formally model the intricate interactions and communication protocols among federated clients and servers, capturing potential vulnerabilities and enabling systematic verification of security properties. Considering the complex nature of our proposed SecFL framework, we model the problems of attack and defence scenarios as HLPNs. For better understanding, first we outlined the primary sections, including "Place" and "Mapping," in Table 2. Their abbreviations are provided in Tables 3 and 4.

In Figure 4, the second parameter in the tuple of $(T.Pub)X_1$, which contains the model parameters, must correspond to the value of the first feature in the tuple of $(GMP)X_2$, representing the global model parameters. This stipulation ensures that the fourth element in the tuple of $(CHL)X_3$ (X_4), $(CVL)X_4$ (X_5), and (X_{12}) matches the value of the second element in the tuple of $(T.Pub)X_1$. Equation (1) ensures that the secure global model parameters $(SGMP)$ receive accurate inputs from all associated components.

$$
\begin{aligned}
\boldsymbol{R}(\textbf{SGMP}) &= \forall x_1 \in X_1 \wedge \forall x_2 \in X_2 : x_2[1] = x_2[0] \\
\boldsymbol{R}(\textbf{AGG}) &= \forall x_2 \in X_2 \wedge \forall x_3 \in X_3 \wedge \forall x_4 \in X_4 \wedge \forall x_5 \in X_5 : \\
&\quad x_2[0] = x_3[3] = x_4[4] = x_5[5] \\
&\quad x_2[1] = x_3[2] = x_4[2] = x_5[2] \\
&\quad \forall x_3 \in X_3 \wedge \forall x_4 \in X_4 \wedge \forall x_5 \in X_5 \\
&\quad \wedge \forall x_{12} \in X_{12} \wedge \forall x_1 \in X_1 : \\
&\quad x_3[4] = x_{12}[0] = x_1[1] \\
&\quad x_4[4] = x_{12}[0] = x_1[1] \\
&\quad x_5[4] = x_{12}[0] = x_1[1] \qquad (1)
\end{aligned}
$$

The specified Equation (1) constraints on the permissible values of certain model variables post-attack. These variable values are constrained according to a predefined set of rules, particularly for x_2, x_7, and x_8 in the first equation, $R(AGG)$. The symbol $\wedge$ denotes "and," while $\forall$ signifies "for all."

The Equation (2) $R(AP)$ states that the values of x_2 and x_7 should match the value of x_1 in the first position, specifically $x_2[0] =$

TABLE 2 | Places, Mapping, and their description of formal analysis of HLPN models.

Places	Mapping	Mapping description
$\phi(T.Pub)X_1$	$P(ID, MP)$	Place holding the task publisher's identification and a set of model parameters (MP)
$\phi(GMP)X_2$	$P(MP, Ekey)$	Place holding a set of global model parameters (MP) and an encryption key (Ekey)
$\phi(SGMP)X_3$	$P(SMP, Ekey)$	Place holding a set of secure global model parameters (SMP) and an encryption key (Ekey)
$\phi(GMP_{Man})X_8$	$P(MMP, Ekey)$	Place holding a set of manipulated global model parameters (MMP) and an encryption key (Ekey)
$\phi(ATT)X_7$	$P(ID, AMP, Ekey)$	Place holding the attacker's identification (ID), an adversarial model parameter (AMP), and an encryption key (Ekey)
$\phi(CBL)X_5$	$P(ID, Ekey, TData, MP)$	Place holding the client bank's identification (ID), an encryption key (Ekey), transaction data (TData), and a set of model parameters (MP)
$\phi(CHL)X_3, X_4$	$P(ID, Ekey, TData, MP)$	Place holding the client hospital's identification (ID), an encryption key (Ekey), transaction data (TData), and a set of model parameters (MP)
$\phi(CVL)X_4, X_5$	$P(ID, Ekey, TData, MP)$	Place holding the client vehicle's identification (ID), an encryption key (Ekey), transaction data (TData), and a set of model parameters (MP)
X_7	$P(MP)$	Location (MP) containing a list of model parameters
X_{12}	$P(MP)$	Location (MP) containing a list of model parameters
X_9	$P(MP)$	Location (MP) containing a list of model parameters
X_{10}	$P(MP)$	Location (MP) containing a list of model parameters
X_{11}	$P(MP)$	Location (MP) containing a list of model parameters

TABLE 3 | Places and their abbreviation of formal analysis of HLPN models and rules.

Places	Abbreviation
SGMP	Secure global model parameters
T.Pub	Task publisher
E.key	Encryption key
GMP	Global model parameter
ATT	Attacker
GMP (Man)	Global model parameter manipulated
CBK	Client bank
CHL	Client hospital
CVL	Client vehicle

TABLE 4 | Transactions and their abbreviation of formal analysis of HLPN models and rules.

Transaction	Abbreviation
T.inp	Task input
SGMP	Secure global model parameter
AP	Attack performed
A.int	Attack initialized
AGG	Aggregator/aggregation
ECDSA	Elliptic curve digital signature algorithm

$x_7[0] = x_1[0]$. Additionally, the first position value of x_8 should be identical to the second position value of x_2, that is, $x_8[0] = x_2[1]$. These rules encapsulate the ramifications of a system breach.

The Equation (2), represented by $R(AGG)$, constrains the values of the variables x_3, x_4, x_5, and x_8. It mandates that the first position value of x_8 equates to the third position values of x_3, x_4, and x_5, that is, $x_8[0] = x_3[3] = x_4[3] = x_5[3]$. Additionally, the second position value of x_8 should match the first position values of x_3, x_4, and x_5. The rules further specify that the third position value of x_3 should correspond to the first position value of x_9, the third position value of x_4 should align with the first position value of x_{10}, and the third position value of x_5 should be identical to the first position value of x_{11}. Moreover, the rule necessitates that the first position values of x_9, the sixth position value of x_{10}, and the first position value of x_{11} should all equate to the first position value of x_{12}, which in turn should be identical to the second position value of x_1. These constraints are also formulated to reflect the aftermath of a system intrusion.

$$
\begin{aligned}
\boldsymbol{R}(\mathbf{AP}) = \forall x_2 \in X_2 \wedge \forall x_7 \in X_7 \wedge \\
\forall x_8 \in X_8 : x_7[0] = x_2[0] = x_1[0] \\
\wedge \; x_8[0] = x_2[1] = x_8[1] \\
\boldsymbol{R}(\mathbf{AGG}) = \forall x_8 \in X_8 \wedge \forall x_3 \in X_3 \wedge \\
\forall x_4 \in X_4 \wedge \forall x_5 \in X_5 : \\
x_8[0] = x_3[3] = x_4[3] = x_5[3] \\
x_8[1] = x_3[1] = x_4[1] = x_5[1]
\end{aligned}
$$

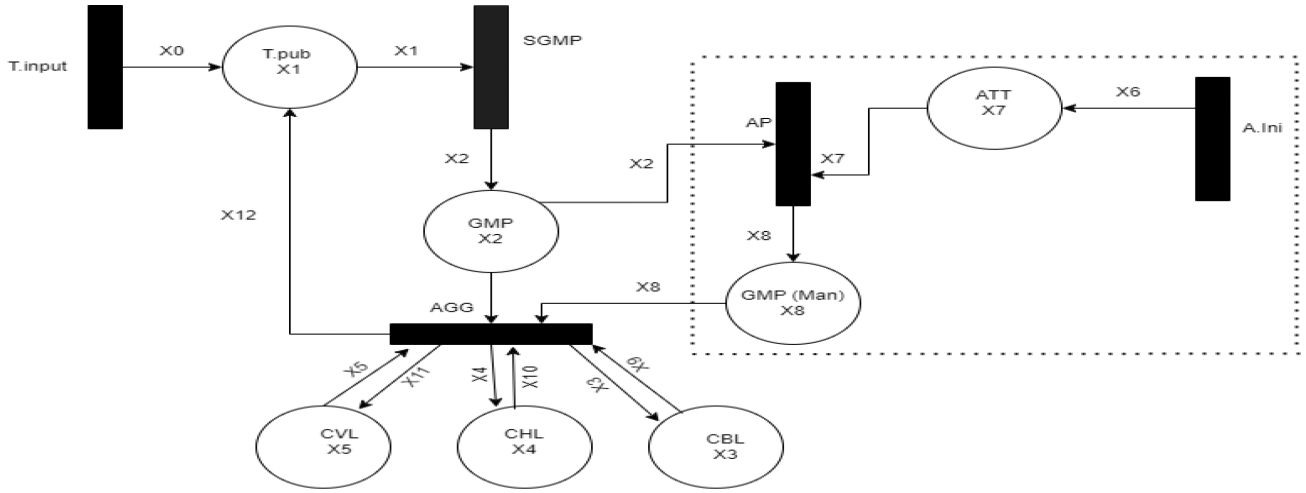

FIGURE 4 | HLPN after attack of Pollard RHO on ECC.

$$\begin{aligned}
&\forall x_3 \in X_3 \wedge \forall x_4 \in X_4 \wedge \forall x_5 \in X_5 \wedge \\
&\forall x_{12} \in X_{12} \wedge \forall x_1 \in X_1 : \\
&\quad x_3[3] = x_9[0] = x_4[3] = x_{10}[0] = x_5[3] \\
&= x_{11}[0] = x_9[0] \\
&\quad x_{12}[0] = x_1[1] \wedge x_{10}[6] = x_{12}[0] = x_1[1] \\
&\quad x_{11}[0] = x_{12}[0] = x_1[1] \qquad (2)
\end{aligned}$$

In Figure 5, $T.Inp$ provides input to $T.Pub$. Subsequently, $SGMP$ shares the global model parameters with GMP without any attack. GMP then shares these parameters with CBK, CHL, and CVL via AGG. When an attack is initiated through AP, with rules ranging from ATT to $A.Ini$, the parameters shared by GMP will be manipulated via AP and stored in $GMP(Man)$. After this manipulation, the updated parameters will be disseminated by AGG to CBL, CHL, and CVL. These parameters are subsequently shared with $T.Pub$.

The rules governing the values of variables in an HLPN model are articulated by the set of equations referenced in Figure 4. These constraints are crafted to capture the system's behavior after an ECDSA operation.

The values of variables x_1 and x_2 are governed by the Equation (3) $R(SGMP)$. In the provided table, x_1 represents a set of public keys, while x_2 symbolizes a set of private keys. The stipulation in $R(SGMP)$ mandates that the second position value of x_2 should align with the first position value of x_1. This constraint ensures a secure linkage between network devices by limiting access to network devices and traffic.

In Equation (3) labeled $R(ECDSA)$, the variables x_2 and x_3 have conditions imposed on their values. The table denotes that x_3 represents a set of signatures. The Equation $R(AGG)$ elucidates the relationships between the variables x_3, x_4, x_5, x_6, x_7, and x_1 and is nested within $R(ECDSA)$. Following the ECDSA operation, the system's behavior is characterized by the stipulations in $R(AGG)$.

$$\mathbf{R(SGMP)} \leftarrow \forall x_1 \in X_1 \wedge \forall x_2 \in X_2 | x_1[1] = x_2[0]$$

$$\mathbf{R(ECCDSA)} \leftarrow \forall x_2 \in X_2 \wedge \forall x_3 \in X_3$$

$$\begin{aligned}
\mathbf{R(AGG)} \leftarrow \forall x_3 \in X_3 \wedge \forall x_4 \in X_4 \wedge \forall x_5 \in X_5 \wedge \forall x_6 \in X_6 : \\
x_3[0] = x_4[3] = x_5[3] = x_6[3] = x_3[1] = x_4[1] \\
= x_5[1] = x_6[1] \\
x_4[3] = x_7[0] = x_1[1] \wedge x_5[3] = x_7[0] = x_1[1] \\
x_6[3] = x_7[0] = x_1[1] \qquad (3)
\end{aligned}$$

According to the constraints in $R(AGG)$, the value of x_3 in the first position must match the third position values of x_4, x_5, and x_6 post the ECDSA operation. This ensures the security and immutability of the signatures generated by the ECDSA process. The rules further dictate that the first position values of x_4, x_5, and x_6 should be consistent with the second position value of x_3. This provides updated and secured global model parameters, which are further shared from these locations. For ECDSA, $T.inp$ furnishes input to $T.pub$. Subsequently, $SGMP$ shares the global model parameter with GMP. Following this, another transition, labeled ECDSA, secures the global model $SGMP$ post-signature and verification and dispatches it to the aggregator. The aggregator then transmits the secure global model parameters to CBL, CHL, and CVL. From these locations, the updated and secured global model parameters are further shared with $T.pub$.

7 | Experimentation

To evaluate the efficacy of the SecFL framework, this section first delineates its experimental implementation, followed by the resulting outcomes and their analysis.

7.1 | Experimental Analysis and Deployment Approaches

The implementation utilized Python as the primary programming language. It offers various time-related functions to measure the duration required for signing and validating messages. Data visualization was achieved using Matplotlib, while numpy facilitated numerical operations on the data, including support for arrays and matrices. Keras datasets provide a variety of machine-learning datasets, with the MNIST dataset employed

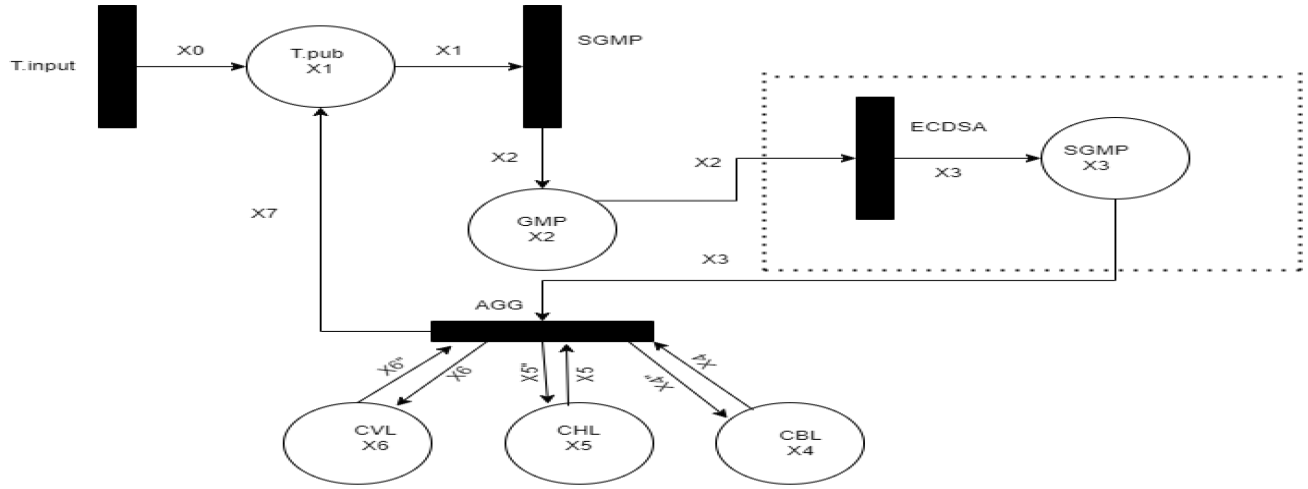

FIGURE 5 | HLPN model after applying the ECDSA security to the proposed SecFL model.

to generate messages. An extensive subset of their corresponding labels is exploited for this experiment, sourced using the Keras datasets package. This application's utilized digital signatures serve as authentication mechanisms, guaranteeing message authenticity, integrity, and non-repudiation.

7.2 | Evaluation of Signature Schemes for Secure Communication

This section assesses the viability of a secure FL architecture by emphasizing pivotal factors such as security, privacy, and cost. The primary objective is to ascertain the framework's real-world applicability. The analysis commences with a comprehensive security review, probing the strengths and vulnerabilities of the proposed framework. A subsequent cost analysis is presented to shed light on the financial ramifications of the framework's deployment. Moreover, we underscore our approach's unique contributions by contrasting our SecFL framework with previously published research on digital signatures. Our analytical discourse offers insights, guiding practitioners towards devising more robust and secure communication frameworks.

7.3 | Security Analysis of ECDSA

This segment conducts a rigorous security analysis of the ECDSA signature technique, prevalently utilized in secure communication systems. The intent is to elucidate how ECDSA fulfills its privacy objectives and to deliberate on strategies for countering potential threats and breaches. Our investigation reveals that ECDSA effectively meets the security benchmarks of confidentiality and integrity, thereby providing a robust mechanism for ensuring authentication, security, privacy, and non-repudiation. By leveraging public and private keys, it fortifies model parameters against adversarial activities. Our comprehensive security analysis accentuates the pivotal role of ECDSA in ensuring secure communication and safeguarding data, offering invaluable insights to industry professionals.

7.4 | Privacy Innovative Analysis of ECDSA FL Framework

This section emphasizes the paramount role of privacy in FL, presenting an innovative examination of a decentralized FL architecture fortified with ECDSA encryption for enhanced security. The objective is to illustrate the framework's resilience against unauthorized breaches and its prowess in safeguarding user privacy. Within the context of FL, the discussion accentuates privacy, proposing security objectives centered around confidentiality and integrity. The narrative then elucidates how the proposed design achieves these objectives, leveraging ECDSA encryption for parameter encryption and decryption and employing a private Ethereum setup to ensure exclusive access to authorized, delegated clients for decryption of shared parameters. This mechanism ensures that only authenticated participants access shared global model parameters, preventing unauthorized entities from deciphering private client data. The SecFL Framework assures that these parameters remain inaccessible without authenticated client registration on the Ethereum blockchain. Unlike many extant FL systems that resort to centralized approaches for global model training—exposing them to threats and centralized control—our proposed system synergizes Ethereum with horizontal FL. This melding obviates centralization and mitigates poisoning attacks. Collectively, our in-depth analysis underscores the significance of privacy and security in FL and the advantages of a decentralized approach using ECDSA encryption, proffering invaluable insights for researchers and industry practitioners.

7.4.1 | Cost Evaluation of SecFL

This segment furnishes a meticulous assessment of SecFL, positioning it in comparison to other established cryptographic techniques like RSA, ECC, and ECDSA. Evaluative criteria encompass computational costs, transactional latency, and accuracy. The subsequent discourse delves into a comprehensive cost analysis of the framework, appraising the efficacy of the proposed prevention mechanisms. Utilizing publicly accessible datasets, including those from the vehicle, bank, hospital, and MNIST domains,

SecFL's performance is juxtaposed against other cryptographic methods to facilitate a detailed comparison. This comparative analysis elucidates the security, privacy, and performance merits of SecFL, delineating each technique's strengths and weaknesses.

7.4.2 | Computation Cost

For a comprehensive analysis, evaluate the efficiency of various encryption, decryption, and digital signature algorithms, including RSA, ECC, and ECDSA. Measure the computational costs of each algorithm by gauging the time and accuracy necessary for encrypting, decrypting, signing, and verifying model parameters. Initially, the code imports the MNIST dataset. The mathematical relationship represented by Equation (4) illustrates the computational cost utilized in this code:

$$\text{Average Computational Cost} = \frac{1}{n}\sum_{i=1}^{n}(TE_{\text{enc}} + TD_{\text{enc}}) \quad (4)$$

where:

- n denotes the total number of iterations.
- TE_{enc} signifies the time or cost for encryption per iteration.
- TD_{enc} represents the time or cost for decryption per iteration.

Subsequent to this formulation, the code generates the ECC and ECDSA key pairs using the ECDSA module and the SECP256k1 curve. It then produces the RSA key pair leveraging the cryptography module with a public exponent of 65 537 and a key size of 4096 bits. The code creates signatures for each message via each algorithm and then verifies them. Padding for RSA signatures is facilitated using the padding module from the cryptography library. This code harnesses digital signature algorithms to bolster secure communication in the Internet of Vehicles (IoV), hospitals, and banks. Three cryptographic algorithms have been implemented, namely RSA, ECC, and ECDSA. The graphical representations illustrate the average times for signing and verifying across the three cryptographic algorithms: RSA, ECC, and ECDSA. The initial trio of graphs illustrates the signing times, while the subsequent graph elucidates the verification durations. The x-axis enumerates the algorithmic types, while the y-axis showcases time in seconds. The blue bars are indicative of the RSA algorithm, the green ones represent the ECC algorithm, and the red bars correspond to the ECDSA algorithm. Figure 6 offers a comparative view of cryptographic algorithms tailored for secure IoV communication. Specifically, Figure 6a portrays the signing durations, while Figure 6b elucidates the verification intervals. For the RSA algorithm, the signing time clocks at 0.0060s and verification at 0.0050 s. ECC demands 0.0017 s for signing and 0.0047s for verification. Contrastingly, ECDSA requires a mere 0.0009 s for signing and 0.0002s for verification, making it more efficient than both RSA and ECC.

$$\text{Signing Time} = \frac{\sum_{i=1}^{N}\text{signing time}_i}{N} \quad (5)$$

Where:

- signing time$_i$ denotes the time taken to perform the signing operation for each message i.
- N represents the total number of messages.

Equation (5) calculates the average signing time for the iteration by summing up the signing times for all messages and dividing by the total number of messages.

For verification, let's define another equation:

$$\text{Verification Time} = \frac{\sum_{i=1}^{N}\text{verification time}_i}{N} \quad (6)$$

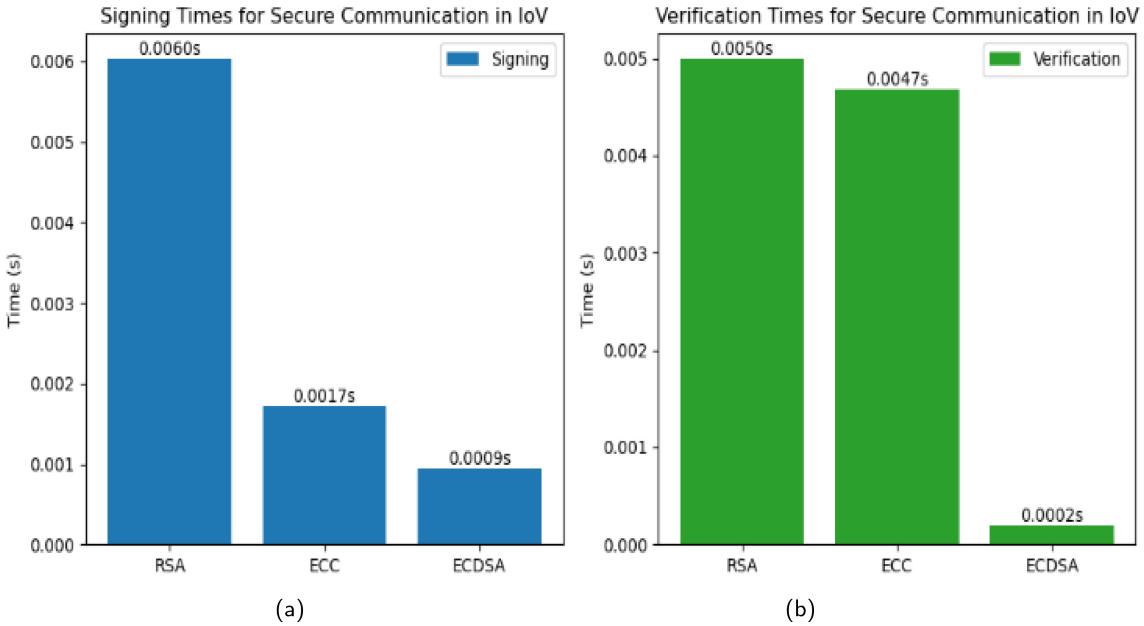

FIGURE 6 | Average computational cost on IoV.

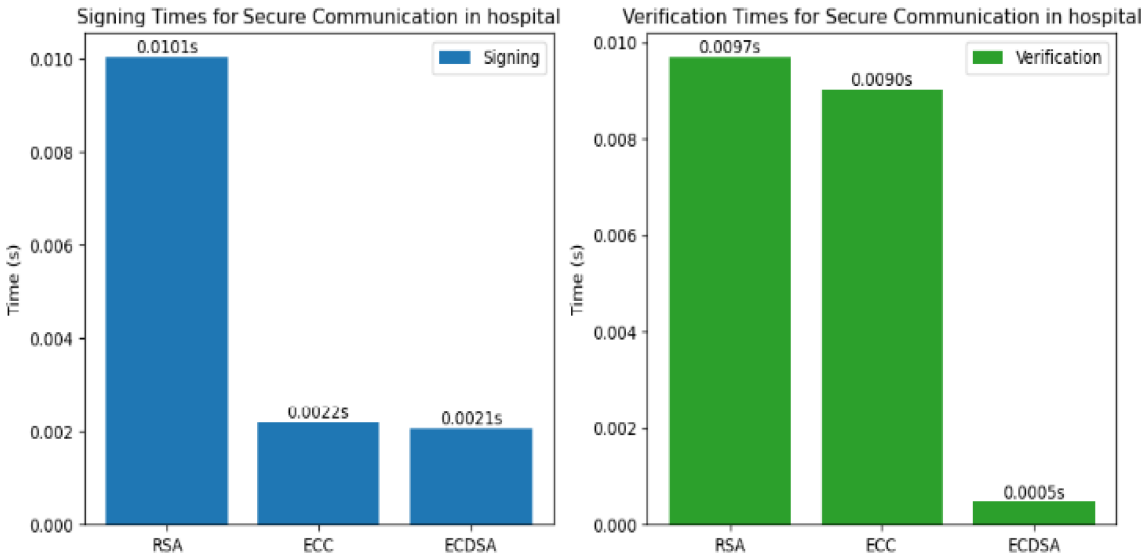

FIGURE 7 | Average computational cost in hospitals.

Where:

- verification time$_i$ represents the time taken to perform the verification operation for each message i.
- N again stands for the total number of messages.

Equation (6) calculates the average verification time for the iteration by summing up the verification times for all messages and dividing it by the total number of messages. Banking dataset: Contains 10 000 records with 20 features per entry, including customer demographics, transaction patterns, and service usage indicators. The target variable is binary (loan approval).

Figure 7 demonstrates a comparison of cryptographic algorithms for secure communication in hospitals. Hospital dataset: Comprises 15 000 patient records with 25 clinical and demographic features (e.g., age, diagnosis codes, medication history). The target variable is disease risk classification (multi-class).

IoV dataset: Includes 12 000 entries capturing vehicular telemetry, traffic flow metrics, and geographic data. The target variable is vehicle anomaly detection (binary classification).

Each dataset was partitioned in a non-IID manner to simulate realistic federated settings, where client data distributions vary significantly.

Figure 6a shows the signing time, and Figure 6b displays the verification time. RSA takes 0.0101 s for signing and 0.0097 s for verification. ECC requires 0.0022 s for signing and 0.0090 s for verification. ECDSA takes 0.0021 s for signing and 0.0005 s for verification. Among the three, ECDSA has the shortest signing and verification times.

Figure 8 compares cryptographic algorithms for secure communication in banking organizations. Figure 8a shows the signing time, while Figure 8b indicates the verification time. RSA takes 0.0126 s for signing and 0.0115 s for verification. ECC requires 0.0042 s for signing and 0.0102 s for verification. ECDSA takes 0.0025 s for signing and 0.0002 s for verification. Again, ECDSA proves to be the fastest among the three.

Transaction latency in an FL system is the time a transaction takes between a client and server. Reducing transaction latency is crucial for enhancing the system's overall efficiency and performance.

The mathematical Equation (7) for the average transaction latency used in this code is given by:

$$A = \frac{1}{n}\sum_{i=1}^{n}\text{latency}_i \tag{7}$$

Where:

- A denotes the average latency.
- n stands for the total number of transaction measurements.
- latency$_i$ signifies the latency measurement for each transaction.

Figure 9 illustrates the average transaction latency in Figure 9a and the verification time in Figure 9b for three cryptographic algorithms: ECDSA, ECC, and RSA. The x-axis represents the algorithms, while the y-axis indicates the average time in seconds. ECDSA has the lowest average transaction latency of 0.0008 s. However, its average verification time of 0.0052 s is slightly longer. ECC's average transaction latency is 0.0016 s, but it verifies faster at 0.0042 s. RSA has the highest transaction latency of 0.0058 s but offers the quickest verification at 0.0002 s.

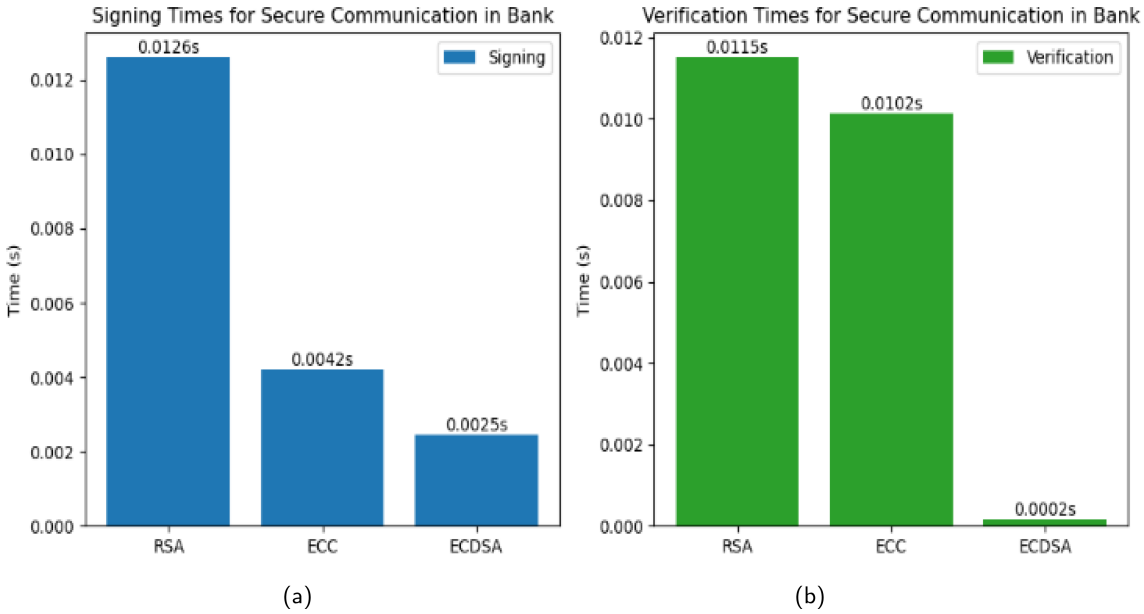

FIGURE 8 | Average computational cost in banking.

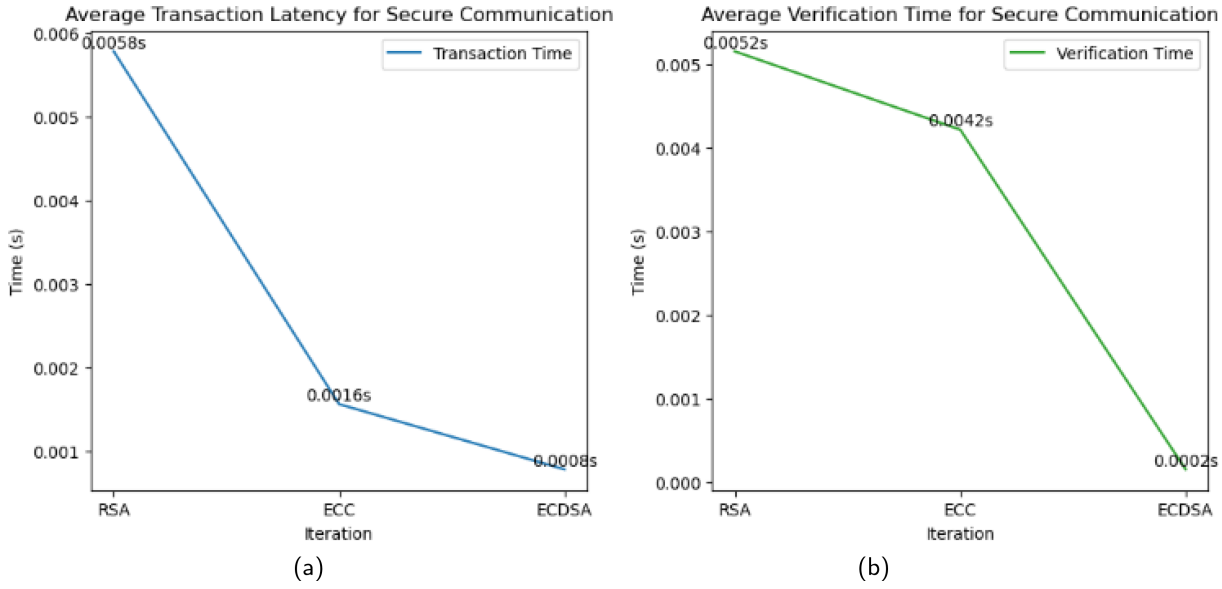

FIGURE 9 | Average transaction latency and verification comparison.

The mathematical Equation (8) for the categorical cross-entropy loss function used in this code is:

$$L(y_{\text{pred}}, y_{\text{true}}) = -\frac{1}{C}\sum_{i=1}^{C} y_{\text{true}_i} \log(y_{\text{pred}_i}) \tag{8}$$

- y_{pred} is the predicted output from the model.
- y_{true} is the true output (ground truth).
- C represents the number of classes.
- Σ is the summation symbol.
- i denotes the index of the class.
- log represents the natural logarithm function.

In Figure 10, the y-axis depicts the loss, ranging from 0.0s to 1.8 s, while the x-axis enumerates the model's iterations, totaling 700 evaluations. The graph illustrates that the SecFL architecture achieves a reduced gradient loss for each iteration compared to the SecPriveChain system. Furthermore, the loss for SecFL is considerably lower than that of SecPriveChain, suggesting that SecFL is more proficient in minimizing loss during FL.

7.4.3 | Experimental Dataset Validation

In this section, we discuss SecFL, 2 different datasets that we use in our experiments, and a secure FL framework focused on safeguarding the privacy of data shared during the learning process. This framework ensures secure communication channels

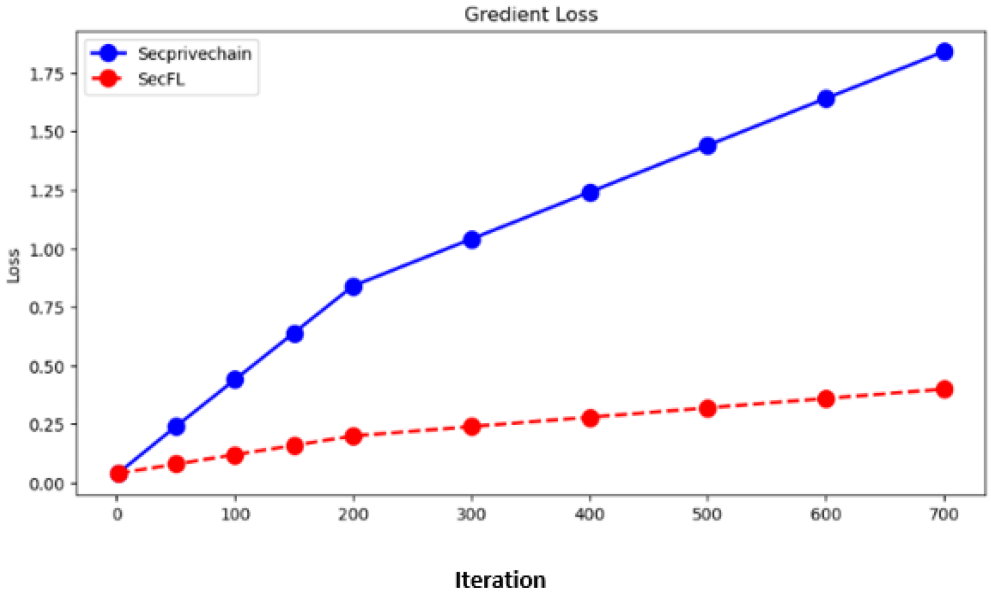

FIGURE 10 | Loss value comparison.

between clients and the server, maintaining data confidentiality and privacy. Utilizing a blockchain-based architecture, SecFL fortifies data integrity and communication.

- *Number of clients*: 100
  This reflects a moderately large FL environment, simulating a typical IoT deployment scenario with a distributed set of clients.
- *Batch size*: 32
  A standard mini-batch size that balances model convergence stability and computational efficiency on resource-constrained IoT devices.
- *Learning rate*: 0.01
  This value was selected to provide steady convergence during training, avoiding both divergence and overly slow learning across communication rounds.
- *Number of local epochs*: 5
  Each client performs 5 local updates before synchronizing with the global model, which is a common configuration to reduce communication overhead while maintaining learning progress.
- **Client sampling rate:** 10% per communication round
  In each round, 10 out of the 100 clients are randomly selected to participate. This setting reflects realistic constraints where not all clients may be available or online at once, while also reducing the communication load per round.

CIFAR-10 (image classification, 10 classes) and FEMNIST (handwritten character recognition, 62 classes, 3550 clients).

The primary objective of the proposed framework is to emphasize the protection of client data privacy and address data poisoning issues effectively (Table 5). Additionally, the framework integrates client authentication as an essential feature [1]. The chain-PPFL system presents concerns about potential breaches in client data privacy during data distribution [27]. Notably, it showcases the advantage of having a low computational cost. Both SAFELearn and PrivacyFL tackle inference attacks by employing secure Multi-Party Computation (MPC) or fully DS encryption (FHE) techniques [28, 29]. However, these methods lead to increased computational complexity and costs, necessitating a careful evaluation of these trade-offs in practical scenarios.

8 | Conclusion

The primary objective of this study was to bolster the security of global model parameters during the training phase of FL. We aimed to safeguard the sensitive initial values of global model parameters, especially in applications such as the IoV, Health, and Banking sectors. By leveraging a blockchain-based architecture to authenticate clients in a decentralized manner, we achieved enhanced security through digital signatures, all while maintaining a minimal transaction delay. FL, an innovative ML technique, allows multiple entities to collaboratively train a shared model without disclosing their local data to a central server or other participants. Each user retains their training data on their respective device or local server, with only the model updates being communicated to the central server. The central server aggregates these updates, refines the shared model, and redistributes it to the clients for further training. This iterative process continues until the model achieves the desired level of performance or accuracy. However, privacy in FL is not absolute. While FL enables multiple devices to train a model using decentralized and private data, it remains susceptible to challenges like data poisoning and label flipping attacks. Specifically, label-flipping attacks mislabel inputs, leading to erroneous model updates. To protect our global parameters from such threats, we employed various cryptographic techniques, including RSA, differential privacy, and ECC. Nonetheless, ECC remains vulnerable to the Pollard RHO attack due to its smaller key size. To mitigate this, we utilized ECDSA, addressing the discrete logarithm problem. A mathematical model based on HLPN was proposed

TABLE 5 | Comparison of the proposed framework to the existing works.

Features	PrivacyFL	ChainPPFL	SAFELearn	SecurePriveChain	Proposed
Time efficacy	Low	Low	Low	Low	High
Computation cost	High	High	High	High	Low
Accuracy	Low	Low	Low	Low	High
Privacy	Low	Low	Low	Low	High
Security	Low	Low	Low	Low	High

to depict the relationship between attack and defense mechanisms. The initial step in using HLPN for adversarial modeling involves understanding the system's components and their interrelationships. Once the HLPN model is meticulously constructed, formal methods will validate and test it. During the verification phase, the model's adherence to specific requirements, such as the absence of deadlock or livelock, is rigorously assessed. Post-verification, the model is simulated to observe its behavior under varied conditions.

For future endeavors, we plan to develop Adaptive Defensive Mechanisms: Frameworks capable of real-time responses to evolving attacks throughout the training process.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. K. Sanam, S. U. R. Malik, T. Kanwal, and Z. U. I. Adil, "SecurePrivChain: A Decentralized Framework for Securing the Global Model Using Cryptography," *Future Generation Computer Systems* 142 (2023): 364–375.

2. T. Q. Dinh, D. N. Nguyen, D. T. Hoang, T. V. Pham, and E. Dutkiewicz, "In-Network Computation for Large-Scale Federated Learning Over Wireless Edge Networks," *IEEE Transactions on Mobile Computing* 22 (2022): 5918–5932.

3. L. Campanile, S. Marrone, F. Marulli, and L. Verde, "Challenges and Trends in Federated Learning for Well-Being and Healthcare," *Procedia Computer Science* 207 (2022): 1144–1153.

4. R. M. Savithramma, R. Sumathi, and H. S. Sudhira, "Reinforcement Learning Based Traffic Signal Controller With State Reduction," *Journal of Engineering Research* 11 (2023): 100017.

5. M. J. Josodipuro, K. V. I. Saputra, and S. Lukas, "Statistical Analysis of Pollard's Rho Attack on Elliptic Curve Cryptography," in *Proceedings of the 2022 1st International Conference on Technology Innovation and Its Applications (ICTIIA)* (IEEE, 2022), 1–6.

6. S. Kanzawa, H. Miura, Y. Kodera, Y. Nogami, and T. Kusaka, "Effectiveness of a Method to Eliminate Fruitless Cycles for Pollard's Rho Method," in *Proceedings of the 2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)* (IEEE, 2022), 1–4.

7. S. U. Malik, S. U. Khan, and S. K. Srinivasan, "Modeling and Analysis of State-Of-The-Art Vm-Based Cloud Management Platforms," *IEEE Transactions on Cloud Computing* 1 (2013): 1.

8. R. Jin and X. Li, "Backdoor Attack Is a Devil in Federated GAN-Based Medical Image Synthesis," in *Simulation and Synthesis in Medical Imaging: 7th International Workshop, SASHIMI 2022, Held in Conjunction With MICCAI 2022, Singapore, September 18, 2022, Proceedings* (Springer, 2022), 154–165.

9. N. Yuvaraj, K. Praghash, and T. Karthikeyan, "Data Privacy Preservation and Trade-Off Balance Between Privacy and Utility Using Deep Adaptive Clustering and Elliptic Curve Digital Signature Algorithm," *Wireless Personal Communications* 124 (2022): 655–670.

10. F. Azam, S. K. Yadav, N. Priyadarshi, S. Padmanaban, and R. C. Bansal, "A Comprehensive Review of Authentication Schemes in Vehicular Ad-Hoc Network," *IEEE Access* 9 (2021): 31309–31321.

11. U. Ghosh, H. Maziku, H. P. Gupta, B. Sikdar, and J. J. P. C. Rodrigues, "Security, Trust, and Privacy Solutions for Intelligent Internet of Vehicular Things–Part I," *IEEE Consumer Electronics Magazine* 11 (2022): 39–40.

12. Z. Wang, Q. Kang, X. Zhang, and Q. Hu, "Defense Strategies Toward Model Poisoning Attacks in Federated Learning: A Survey," in *Proceedings of the 2022 IEEE Wireless Communications and Networking Conference (WCNC)* (IEEE, 2022), 548–553.

13. M. S. Rahman, M. A. Islam, M. A. Uddin, and G. Stea, "A Survey of Blockchain-Based IoT eHealthcare: Applications, Research Issues, and Challenges," *Internet of Things* 19 (2022): 100551.

14. A. K. Yadav, "Significance of Elliptic Curve Cryptography in Blockchain IoT With Comparative Analysis of RSA Algorithm," in *Proceedings of the 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)* (IEEE, 2021), 256–262.

15. S. Shukla, S. Thakur, and J. G. Breslin, "Secure Communication in Smart Meters Using Elliptic Curve Cryptography and Digital Signature Algorithm," in *Proceedings of the 2021 IEEE International Conference on Cyber Security and Resilience (CSR)* (IEEE, 2021), 261–266.

16. Y. Genç and E. Afacan, "Design and Implementation of an Efficient Elliptic Curve Digital Signature Algorithm (ECDSA)," in *Proceedings of the 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)* (IEEE, 2021), 1–6.

17. J. Chen, J. Xue, Y. Wang, L. Huang, T. Baker, and Z. Zhou, "Privacy-Preserving and Traceable Federated Learning for Data Sharing in Industrial IoT Applications," *Expert Systems With Applications* 213 (2023): 119036.

18. T. Duong, K. K. Todi, U. Chaudhary, and H.-L. Truong, "Decentralizing Air Traffic Flow Management With Blockchain-Based Reinforcement Learning," in *Proceedings of the 2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1 (Ieee, 2019), 1795–1800.

19. C. He, G. Liu, S. Guo, and Y. Yang, "Privacy-Preserving and Low-Latency Federated Learning in Edge Computing," *IEEE Internet of Things Journal* 9 (2022): 20149–20159.

20. T. D. P. Bai, K. M. Raj, and S. A. Rabara, "Elliptic Curve Cryptography Based Security Framework for Internet of Things (IoT) Enabled Smart Card," in *Proceedings od the 2nd World Congress on Computing and Communication Technologies* (WCCCT, 2017), 43–46.

21. J. Zhu, J. Cao, D. Saxena, S. Jiang, and H. Ferradi, "Blockchain-Empowered Federated Learning: Challenges, Solutions, and Future Directions," *ACM Computing Surveys* 55 (2023): 1–31.

22. W. Yang, B. Liu, C. Lu, and N. Yu, "Privacy Preserving on Updated Parameters in Federated Learning," in *Proceedings of the ACM Turing Celebration Conference-China* (Association for Computing Machinery (ACM), 2020), 27–31.

23. Y. Shang, "Efficient and Secure Algorithm: The Application and Improvement of Ecdsa," in *Proceedings of the 2022 International Conference on Big Data, Information and Computer Network (BDICN)* (IEEE, 2022), 182–188.

24. R. Behnia, A. Riasi, R. Ebrahimi, S. S. Chow, B. Padmanabhan, and T. Hoang, "Efficient Secure Aggregation for Privacy-Preserving Federated Machine Learning," in *Proceedings of the 2024 Annual Computer Security Applications Conference (ACSAC)* (IEEE, 2024), 778–793.

25. N. Hristov-Kalamov, R. Fernández-Ruiz, C. Conde, et al., "Partially Homomorphic Framework for Secure Privacy-Preserving Id Creation," *Integrated Computer-Aided Engineering* (2024): 10692509251342680.

26. M. Diaz, *Petri Nets: Fundamental Models, Verification and Applications* (John Wiley & Sons, 2013).

27. T. Nguyen, "Advancing Privacy and Accuracy With Federated Learning and Homomorphic Encryption," *Authorea Preprints* (2023).

28. Y. Li, Y. Zhou, A. Jolfaei, D. Yu, G. Xu, and X. Zheng, "Privacy-Preserving Federated Learning Framework Based on Chained Secure Multiparty Computing," *IEEE Internet of Things Journal* 8 (2020): 6178–6186.

29. V. Mugunthan, A. Peraire-Bueno, and L. Kagal, "Privacyfl: A Simulator for Privacy-Preserving and Secure Federated Learning," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Association for Computing Machinery (ACM), 2020), 3085–3092.