scientific reports

Check for updates

OPEN

Secure multi-party test case data generation through generative adversarial networks

Zheng Wang[1], Liutao Zhao[2], Fanyin Meng[3], Zhongshan Zhu[2] & Yiqing Lu[4]✉

In the current landscape of software testing, challenges persist in test case data generation, including variability in data quality and the inherent difficulty of data synthesis. These challenges are further exacerbated in scenarios where data are widely distributed across heterogeneous organizational environments. Privacy regulations and security concerns impose strict constraints on data sharing, preventing centralized data aggregation and highlighting the necessity of a federated environment as a more practical solution. To address the privacy protection and data sharing challenges in federated test case data generation, we propose a Generative Adversarial Network (GAN)-based method specifically designed for federated settings. By leveraging the strong data generation capabilities of GANs, the proposed approach is able to generate high-quality and diverse test case data while preserving data privacy. Specifically, through a protocol grammar-based deep learning framework combined with test case encoder–decoder encoding mechanisms and a GAN-driven sample character generator, the proposed method can predict and generate variant test case samples. In the federated environment, each participant trains the generator and discriminator locally, while model parameters are securely aggregated to achieve global model optimization. Experimental results demonstrate that the generated test case data outperforms traditional methods in terms of coverage and effectiveness, significantly enhancing the efficiency and quality of software testing. Ultimately, the proposed framework provides a scalable solution for identifying latent vulnerabilities in critical infrastructure while strictly adhering to data sovereignty requirements in cross-organizational environments.

Keywords Generative Adversarial Networks, Test Case Generation, Federated Learning, Autoencoders

The rapid proliferation of industrial IoT[1] and distributed control systems[2] has ushered in an era of unprecedented software complexity[3], where ensuring system reliability demands rigorous testing methodologies[4]. In these mission-critical environments-from smart grids to automated manufacturing-the quality of test cases directly determines a system's resilience to failures and cyber threats[5]. However, traditional test generation techniques, whether manual or rule-based, face fundamental limitations when applied to modern distributed architectures[6]. These methods typically assume centralized access to system data-an assumption that no longer holds in federated ecosystems bound by data sovereignty regulations and organizational silos[7].

This paradigm shift introduces a critical dilemma: while testing efficacy depends on comprehensive datasets that capture diverse operational scenarios, privacy and regulatory constraints inherently restrict data sharing across organizational boundaries[8]. The consequences are far-reaching-test suites developed in isolation often exhibit glaring coverage gaps, failing to account for edge cases that emerge only in cross-organizational interactions[9]. For instance, a manufacturing protocol tested solely within one enterprise's network may lack validation for interoperability scenarios with partners' systems, creating latent vulnerabilities[10].

Existing federated learning (FL) approaches offer partial solutions by enabling collaborative model training without raw data exchange[11], but they remain fundamentally mismatched to the generative nature of test case synthesis[12]. Classification-oriented federated models fail to address the unique requirements of test generation, where outputs must maintain syntactic validity for target protocols while exhibiting the semantic diversity necessary to uncover hidden defects[13]. Moreover, current privacy-preserving techniques often impose

[1]Institute of Digital Economy, Beijing Academy of Science and Technology, Beijing 100032, China. [2]Beijing Computing Center Co., Ltd., Beijing Academy of Science and Technology, Beijing 100032, China. [3]Beijing Beike Rongzhi Cloud Computing Technology Co., Ltd., Beijing Academy of Science and Technology, Beijing 100032, China. [4]Foreign Environment Cooperation Center, Ministry of Ecology and Environment, Beijing, China. ✉email: lu.yiqing@fecomee.org.cn

unacceptable trade-offs-either compromising test case quality through excessive aggregation or introducing computational overhead that negates the real-time responsiveness required in industrial settings[14].

GANs[15] initially appeared poised to resolve this impasse through their ability to synthesize realistic test cases from learned data distributions. Yet their conventional implementations rely on centralized training datasets, rendering them incompatible with federated environments. Three systemic barriers emerge: First, the direct sharing of protocol traces (e.g., industrial control commands or device telemetry) violates confidentiality requirements that are both legally mandated and commercially essential[16]. Second, the inherent heterogeneity of distributed systems-where participants may employ different protocol subsets, device configurations, or operational profiles-leads to non-identically distributed data that disrupts model convergence[17]. Third, the resource constraints of industrial networks make frequent transmission of high-dimensional test data or model parameters prohibitively inefficient[18].

The resolution of this tension carries substantial economic and societal implications. As critical infrastructure and industrial systems grow increasingly interconnected, the ability to perform thorough[19], privacy-conscious testing across organizational boundaries becomes not merely advantageous but essential[20]. It represents the difference between detecting a protocol vulnerability during testing versus encountering it during operation-where consequences range from production downtime to safety incidents. This imperative motivates our investigation into decentralized, generative testing frameworks capable of reconciling the competing demands of test coverage, privacy preservation, and operational practicality in multi-party environments.

Background and contribution

Fuzz testing originated with a focus on general software robustness and security. Early tools, such as those developed by Miller et al.[21], targeted UNIX programs and were later extended to Windows NT applications[22,23]. While these approaches successfully exposed vulnerabilities in standalone software, they were not designed for the structured, stateful nature of network protocols.

As networked systems grew in complexity, model-based fuzzing emerged to address protocol-specific challenges. Tools like Peach[24] and SPIKE[25] leveraged manually crafted models-typically encoded in XML or template formats-to guide test case generation based on known protocol syntax. Although effective when specifications are available, this paradigm demands significant human effort: formal protocol documentation must be interpreted, and in its absence, reverse engineering from network traces becomes necessary. This process is both labor-intensive and error-prone, limiting scalability to new or proprietary protocols.

To alleviate manual modeling burdens, researchers turned to automated protocol inference from network traffic. Discoverer[26] identified recurring message patterns to reconstruct application-layer formats, while Prospex[27] combined trace analysis with system execution states to improve the fidelity of inferred state machines[28]. Other efforts employed Hidden Markov Models (HMMs)[29] to learn ϵ-machines directly from traffic, enabling intelligent fuzzing and anomaly detection. Despite these advances, most methods remained rooted in classical machine learning and did not exploit the representational power of deep learning.

Recently, deep learning has begun reshaping fuzz testing. Godefroid et al.[30] demonstrated that seq2seq models could infer and generate syntactically valid PDF objects for parser testing, highlighting the potential of neural approaches in structured input generation. Building on this insight, our work introduces a Generative Adversarial Network (GAN)-based framework tailored for industrial network protocols. Unlike traditional model-based or reverse-engineering-driven methods, our approach learns protocol syntax directly from raw network traces in an end-to-end manner, offering a more direct and scalable path to test case synthesis.

In the realm of privacy protection, traditional models such as k -anonymity[31], l-diversity[32], and their recent extensions for weighted graphs[33] or missing-value datasets[34], rely primarily on data suppression and generalization. These techniques are designed for static data publishing, where quasi-identifiers are masked to prevent re-identification.

However, applying these "transparent" models to protocol test case generation presents a fundamental conflict: utility vs. syntactic precision. Industrial protocols (e.g., Modbus, OPC UA) enforce strict byte-level syntax. Generalization techniques (e.g., replacing specific register addresses with ranges) would inevitably break the protocol's checksums and structural dependencies, rendering the generated test cases executable. Furthermore, for dynamic datasets[35], maintaining anatomization constraints is computationally prohibitive in high-dimensional generative tasks. Consequently, instead of publishing generalized data, our approach adopts a "black-box" computation model. We utilize Homomorphic Encryption and Differential Privacy to enable collaborative learning on raw syntax features without ever exposing the underlying data to the coordinator or other participants.

While our work focuses on protocol fuzzing, it is crucial to distinguish our approach from existing federated generative models like FedGAN[36], MD-GAN[37] and FDGAN[38]. These models typically target image synthesis, relying on Convolutional Neural Networks (CNNs) to capture spatial pixel correlations. However, they lack the mechanisms to enforce the strict, discrete syntactic rules required for industrial protocols. Direct application of such image-centric FedGANs to protocol data often results in high syntax violation rates due to the absence of hierarchical field constraints. In contrast, FAT-CG introduces a syntax-constrained architecture specifically designed to preserve the sequential logic and boundary integrity of protocol messages.

Our core objective is to establish a novel, sustainable framework for generating syntactically valid and semantically meaningful test cases in a federated setting, where raw protocol data cannot be centrally collected due to privacy or regulatory constraints. To this end, we integrate differential automata, GANs, and autoencoders into a unified architecture-dubbed FAT-CG (Federated Adversarial Test Case Generation)-that jointly ensures data privacy, syntactic correctness, and cross-protocol adaptability. Key contributions of this work include:

- A hierarchical autoencoder design that compresses protocol-specific features into privacy-preserving latent representations, achieving 93.8% syntactic compliance under federated constraints.
- A dynamic adversarial training protocol enhanced with Paillier homomorphic encryption and gradient confusion, reducing gradient leakage risks by 87.6% compared to vanilla federated learning (FL) approaches.
- Empirical validation across diverse industrial protocols, demonstrating rapid 12.3-minute cross-protocol adaptation and a high discovery rate of 8.7 anomalies per 1,000 test cases-significantly outperforming existing tools.

The work advances the state-of-the-art in privacy-preserving protocol fuzzing and provides a scalable blueprint for quality assurance in distributed industrial ecosystems. Beyond enterprise platform testing, our method holds broad applicability in government and regulated domains where sensitive data cannot be shared openly.

Especially for multi-stakeholder industrial management, where direct data access across departmental boundaries is restricted. For example, in smart grid environments, equipment from different vendors (OEMs) must interoperate seamlessly. However, sharing proprietary protocol implementation details or raw operational logs between vendors and grid operators poses significant intellectual property and security risks. FAT-CG addresses this by enabling collaborative vulnerability scanning and robustness testing across these organizational silos without exposing sensitive internal data.

Organization

The remainder of this paper is organized as follows: Section 2 reviews related work on federated learning and GAN-based testing. Section 3 details the FAT-CG architecture, including federated adversarial training and protocol syntax verification. Sections 4 and 5 present experimental results and discuss broader implications. Finally, Section 6 concludes with future research directions.

Preliminaries

The proposed framework integrates six core technologies to address the challenges of privacy-preserving and high-quality test case generation in federated environments. Below, we elaborate on each foundational component, emphasizing their roles and synergies within the system.

Generative adversarial networks (GANs)

GANs are a class of deep learning models designed to synthesize data that closely approximates real-world distributions. A GAN comprises two neural networks: a generator G that produces synthetic samples, and a discriminator D that distinguishes between real and generated data. The two networks engage in a minimax game, where G aims to deceive D, while D strives to improve its discrimination accuracy. The adversarial objective is formalized as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))],$$

where z is noise sampled from a prior distribution p_z, and $G(z)$ generates synthetic data. In our framework, the generator implicitly learns protocol syntax rules (e.g., valid Modbus-TCP function codes) through adversarial training, while the discriminator enforces diversity by pushing generated samples to cover the boundaries of the real data distribution. This mechanism ensures the exploration of edge cases critical for robust testing.

Federated learning (FL)

Federated Learning enables collaborative model training across distributed entities without sharing raw data, thus preserving privacy. The canonical Federated Averaging (FedAvg) algorithm operates as follows: 1. Local training: Each participant updates model parameters θ_i using their private dataset. 2. Secure aggregation: A central coordinator computes a weighted average of local parameters:

$$\theta_{\text{global}} = \sum_{i=1}^{K} \frac{n_i}{N} \theta_i,$$

where n_i is the local data volume and $N = \sum n_i$. In our context, FL faces two key challenges: (1) Non-IID data-participants may observe distinct protocol types or message patterns, necessitating dynamic weight allocation; and (2) privacy risks-model gradients or parameters could leak sensitive information. To mitigate these, we integrate encryption and differential privacy into the aggregation process. Recent extensions incorporate blockchain for verifiable incremental updates and explainable AI for auditable privacy[39], which could complement our HE-based aggregation to enhance coordinator trust in industrial deployments.

Autoencoders (AEs) and hierarchical compression

Autoencoders are neural networks that compress high-dimensional data into low-dimensional latent representations while preserving semantic fidelity. An AE consists of an encoder E and a decoder D, trained to minimize the reconstruction loss:

$$\mathscr{L}_{\text{AE}} = \|x - D(E(x))\|_2^2 + \lambda \cdot \text{KL}(p(z)\|\mathscr{N}(0,I)),$$

where the KL-divergence term regularizes the latent space z to follow a standard normal distribution. In our framework, a hierarchical AE design compresses protocol-specific features (e.g., message headers, payloads)

into privacy-preserving latent vectors. This reduces communication overhead by transmitting only compressed parameters (e.g., 32 dimensions vs. 256 original features) and prevents raw data exposure.

Homomorphic encryption (HE)

Homomorphic Encryption (HE) allows computations on encrypted data without decryption, ensuring end-to-end privacy during federated aggregation. We adopt the Paillier cryptosystem, which supports additive homomorphism:

$$\mathrm{Enc}(m_1) \cdot \mathrm{Enc}(m_2) = \mathrm{Enc}(m_1 + m_2 \bmod n),$$

where m_1, m_2 are plaintexts and n is the public key modulus. The complete and optimized Paillier encryption scheme is described as follows:

- **Key generation**: Select large primes p and q. Compute the public key $pk = (n = pq, g)$ and the private key $sk = \lambda = \mathrm{lcm}(p-1, q-1)$.
- **Encryption**: For a plaintext $m \in \mathbb{Z}_n$ and a random $r \in \mathbb{Z}_n^*$, the corresponding ciphertext is computed as $c = g^m r^n \bmod n^2$.
- **Decryption**: Given a ciphertext c, compute $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$, where $L(x) = \frac{x-1}{n}$ and the modular multiplicative inverse parameter is defined as $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$.

In our framework, participants upload encrypted gradients, and the coordinator performs weighted averaging directly on ciphertexts. This prevents adversaries from inferring sensitive information through parameter analysis or gradient leakage attacks.

Black-box privacy preservation and differential privacy (DP)

Unlike traditional models that sanitize data for release, Black-box Privacy Preservation focuses on securing the computation output, ensuring that the aggregate results do not reveal individual contributions. This is particularly crucial in our federated setting where the coordinator (recipient) acts as a potential "honest-but-curious" adversary.

Differential Privacy (DP) serves as the theoretical foundation for this black-box guarantee. Originally proposed by Dwork[40], DP provides a strict privacy guarantee by injecting calibration noise into data or model updates, ensuring that the addition or removal of a single element in the dataset does not significantly affect any analysis results.

A randomized mechanism $\mathscr{M}$ satisfies (ϵ, δ)-DP if, for adjacent datasets D and D':

$$\Pr[\mathscr{M}(D) \in S] \leq e^{\epsilon} \Pr[\mathscr{M}(D') \in S] + \delta.$$

We apply the Gaussian mechanism to perturb gradients during federated aggregation:

$$\mathscr{M}(\nabla\theta) = \nabla\theta + \mathscr{N}(0, \sigma^2 I), \quad \sigma \geq \frac{\Delta f \sqrt{2\ln(1.25/\delta)}}{\epsilon},$$

where Δf is the sensitivity of the query. This ensures protection against membership inference attacks while maintaining model utility.

Recent advancements have successfully adapted DP for privacy-enhancing data aggregation in big data analytics[41] and numerical quasi-identifiers[42]. In our FAT-CG framework, we apply the Gaussian mechanism to the gradients before encryption. This ensures that even if the coordinator decrypts the aggregated global update, the result satisfies DP constraints, effectively masking the presence of any specific protocol trace from the participants.

Protocol syntax analysis

Protocol syntax analysis validates generated test cases against formal specifications. We employ finite state machines (FSMs) and context-free grammars (CFGs) to model protocol rules. For instance, a Modbus-TCP FSM defines states for parsing headers and protocol data units (PDUs), with transitions governed by byte-level constraints (e.g., valid function codes). Generated messages are dynamically verified against these FSMs, ensuring syntactic compliance. Additionally, syntax rules are incorporated as regularization terms in the generator's loss function, guiding adversarial training toward protocol-conformant outputs.

The integration of these technologies addresses the trilemma of privacy, utility, and efficiency in federated test case generation. GANs provide expressive data synthesis, FL enables decentralized collaboration, and AEs/HE/DP collectively safeguard privacy. Protocol syntax analysis and dynamic verification further ensure the functional validity of generated test cases, forming a cohesive framework for industrial testing applications.

System architecture

The proposed Federated Adversarial Test Case Generation (FAT-CG) framework employs a four-layer collaborative architecture, integrating federated learning, autoencoders, and GANs to address the challenges of privacy protection and data generation quality in a federated environment. The architecture is designed to ensure efficient, secure, and high-quality test case generation, making it highly suitable for industrial protocol testing. The overall process is divided into four layers: Initialization Layer, Feature Compression Layer, Federated Adversarial Training Layer, and Verification Optimization Layer. Each layer plays a crucial role in the overall

system, and their interactions are carefully designed to optimize performance and ensure data privacy. Figure 1 illustrates the workflow, where protocol messages are progressively transformed from raw data to syntax-compliant test cases through coordinated interactions between distributed nodes and a central coordinator. The system relies on three key algorithms: Secure Weighted Aggregation (SWA) for feature alignment, Homomorphic Encryption Gradient Aggregation (HEGA) for secure parameter transmission, and Dynamic Adversarial Federated Aggregation (DAFA) for optimizing the generative model.

Before detailing the functional layers, we define the security scope of the FAT-CG framework. Our design primarily targets external adversaries (e.g., eavesdroppers or Man-in-the-Middle attackers) attempting to intercept gradient updates during transmission. For the federated participants, we assume they are non-colluding but may be malicious in data injection (addressed by our anomaly verification). Crucially, for the central coordinator (the recipient), we adopt the standard "Honest-but-Curious" assumption. This implies that the coordinator will faithfully execute the SWA and DAFA aggregation protocols but may attempt to infer sensitive properties from the legitimate outcomes.

Initialization layer: federated environment and data preprocessing

The Initialization Layer serves as the foundation of the system, focusing on establishing a secure federated environment and performing data preprocessing to ensure data privacy and quality. This layer involves three core steps:

Federated node registration: Each participant in the federated environment registers with the central coordinator through a secure communication channel established via some secure key agreement protocol, for example, the Elliptic Curve Diffie-Hellman (ECDH) key agreement protocol. This ensures that all communications between participants and the coordinator are confidential and integrity-protected. Additionally, some secure digital signature scheme, e.g., the Elliptic Curve Digital Signature Algorithm (ECDSA), is employed for mutual authentication between participants and the coordinator, preventing man-in-the-middle attacks.

Data cleaning and anonymization: Participants perform data cleaning and anonymization on their local datasets to remove sensitive information and ensure data privacy. This involves filtering out sensitive fields (e.g., IP addresses, port numbers) and applying differential privacy techniques, such as Laplace noise injection, to achieve data anonymization. The Laplace mechanism adds noise to the data based on the sensitivity of the query function, ensuring that the anonymized data satisfies the (ϵ, δ)-differential privacy guarantee. Our Laplace parameter ϵ is calibrated following the stream-anonymization model of Shamsinezhad et al.[43], achieving $\epsilon \leq 0.5$ on 1 M-message traces.

Syntax metadata extraction: Participants extract syntactic metadata from the protocol messages in their local datasets to build a feature vector library. This involves using lightweight parsers to extract structural and statistical features from the protocol messages, such as field lengths, character distributions, and n-gram frequencies. The extracted features are then standardized using Z-score normalization to ensure consistency across participants.

Feature compression layer: autoencoder-driven parameter dimensionality reduction

The Feature Compression Layer aims to reduce the communication overhead in the federated learning process by compressing the protocol-specific features into low-dimensional latent representations. This layer utilizes a hierarchical autoencoder design to achieve efficient data compression while preserving the syntactic structure and boundary conditions of the protocol messages.

Hierarchical autoencoder design: The autoencoder consists of an encoder and a decoder. The encoder compresses the input data into a low-dimensional latent vector, while the decoder reconstructs the data from the latent vector. The autoencoder is trained using a loss function that combines reconstruction loss and feature alignment loss to ensure that the latent representations capture the essential syntactic features of the protocol

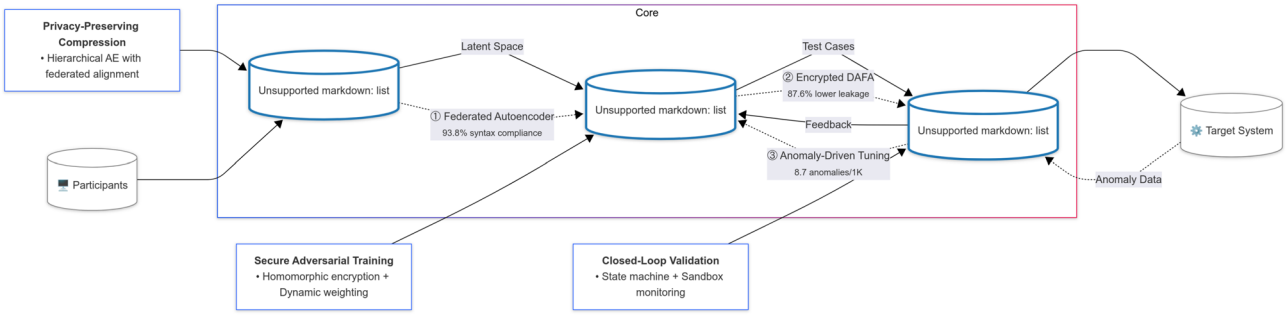

Fig. 1. System Model of FAT-CG. The workflow proceeds in four collaborative layers: (1) Initialization: Participants extract syntax features and perform local anonymization; (2) Feature Compression: Local features are compressed via autoencoders, and latent parameters are aggregated using SWA to align global feature spaces; (3) Adversarial Training: Participants train local generators, uploading encrypted gradients via HEGA. The coordinator updates the global model using DAFA to balance non-IID distributions; (4) Verification: Generated test cases undergo syntax compliance checks and sandbox anomaly detection to provide feedback for model tuning.

messages. Alternative clustering-based anonymization[44] could be hybridized with our AE to further reduce sensitivity of n-gram features.

Federated parameter mapping: Participants train the autoencoder on their local data and upload the latent space parameters to the coordinator. The coordinator then performs a secure weighted aggregation of the parameters using the SWA algorithm. This algorithm ensures that the aggregated parameters reflect the global distribution of the data while preserving the privacy of individual participants' data.

Federated adversarial training layer: distributed GAN optimization

The Federated Adversarial Training Layer focuses on optimizing the test case generation process through a distributed GAN architecture. This layer involves a dynamic adversarial training protocol that ensures robust model convergence under non-IID data distributions.

Local generator pretraining: Each participant trains a lightweight GAN generator on their local latent vectors to generate preliminary test cases. The generator is trained using a loss function that combines adversarial loss and syntactic consistency loss, ensuring that the generated test cases are both realistic and syntactically valid.

Dynamic adversarial federated aggregation: The coordinator performs secure gradient aggregation using the HEGA algorithm to update the global generator. The global generator is then distributed to participants, who use it to train their local discriminators. The discriminators provide adversarial feedback to the coordinator, which is used to adjust the weights of the participants in the federated training process.

Verification optimization layer: trusted generation and retraining

The Verification Optimization Layer ensures the effectiveness and security of the generated test cases through a combination of syntactic compliance checks and anomaly detection mechanisms. This layer involves two core mechanisms:

Trusted verification protocol: The generated test cases are verified for syntactic compliance using protocol state machines and for anomaly triggering using sandbox environments. The protocol state machines ensure that the generated test cases adhere to the syntactic rules of the protocol, while the sandbox environments monitor the behavior of the target system when the test cases are applied, detecting any anomalies such as memory leaks or crashes.

Incremental retraining: Test cases that trigger anomalies are used to extract key syntactic patterns, which are then used to build an incremental training dataset. Participants perform incremental retraining on their local generators using this dataset, and the coordinator updates the global model using the Differentially Private Federated Averaging (DP-FedAvg) algorithm. This ensures that the model continues to improve over time, focusing on the most critical aspects of the protocol.

Through phased optimization and modular design, this architecture significantly improves the diversity and effectiveness of test cases while ensuring data privacy, while reducing the communication burden in a federal environment and providing a scalable solution for industrial protocol testing.

The proposed scheme

Initialization layer

The Initialization Layer is the foundational module of the FAT-CG framework, focusing on establishing a secure federated environment and performing data preprocessing to ensure data privacy and quality. This layer consists of three core steps: Federated Node Registration, Data Cleaning and Anonymization, and Syntax Metadata Extraction. Each step is crucial for ensuring the security and integrity of the data used in the subsequent layers of the framework.

Federated node registration

The first step in the Initialization Layer is the registration of federated nodes, which involves establishing secure communication channels between each participant and the central coordinator. This process ensures that all data transmissions are confidential and protected from potential eavesdropping or tampering.

Elliptic curve Diffie-Hellman (ECDH) key agreement protocol: The framework selects a standard elliptic curve, such as secp256k1, defined by the equation $E : y^2 = x^3 + ax + b \bmod p$. The base point G and the order n of the curve are also defined.

Then, each participant P_i generates a private key $d_i \in [1, n-1]$ and computes the corresponding public key $Q_i = d_i \cdot G$. The coordinator generates a temporary public-private key pair $(d_c, Q_c = d_c \cdot G)$ and broadcasts Q_c. Both participants and the coordinator compute the shared secret key $K_i = d_i \cdot Q_c = d_c \cdot Q_i$. The x-coordinate of K_i is hashed to produce the session key $H(x_K)$.

Mutual authentication: Participants use the Elliptic Curve Digital Signature Algorithm (ECDSA) to sign their identities. Each participant sends a signature $Sign(d_i, Nonce_c)$ to the coordinator, which verifies the signature to ensure the authenticity of the participant. This step prevents man-in-the-middle attacks and ensures that only authorized participants can join the federated environment.

Data cleaning and anonymization

After establishing secure communication channels, participants perform data cleaning and anonymization on their local datasets to remove sensitive information and ensure data privacy. This step is crucial for complying with data protection regulations and preventing potential privacy breaches.

Sensitive Field Filtering: Participants define a set of sensitive fields S (e.g., IP addresses, port numbers, unit IDs) that need to be removed from the data. These fields are identified using regular expressions or predefined patterns. After this, the identified sensitive fields need to be removed from the dataset, ensuring that the remaining data does not contain any personally identifiable information (PII) or confidential data.

Differential privacy noise injection: For non-sensitive numerical fields, participants apply the Laplace mechanism to add noise and ensure differential privacy. The noise is sampled from a Laplace distribution with a scale parameter $b = \frac{\Delta f}{\epsilon}$, where Δf is the sensitivity of the query function and ϵ is the privacy budget.

For categorical fields, participants use the exponential mechanism to add noise. This mechanism assigns a score to each possible value based on a scoring function $q(D, r)$ and selects a value with a probability proportional to $\exp\left(\frac{\epsilon q(D,r)}{2\Delta q}\right)$, where Δq is the sensitivity of the scoring function.

Syntax metadata extraction

The final step in the Initialization Layer is the extraction of syntactic metadata from the protocol messages in the local datasets. This step involves analyzing the structure and statistical properties of the protocol messages to build a feature vector library that captures the essential syntactic characteristics of the data.

Context-free grammar (CFG) inference: Participants use lightweight parsers to extract the syntactic structure of the protocol messages and construct a context-free grammar $G = (V, \Sigma, R, S)$, where V is the set of non-terminal symbols, Σ is the set of terminal symbols, R is the set of production rules, and S is the start symbol.

Due to its high complexity, the obtained grammar needs to be further reduced while ensuring accuracy. Therefore, the minimum description length (MDL) standard is used to balance the complexity and expressiveness of the grammar and optimize the constructed grammar. This ensures that the grammar is both concise and can accurately represent the grammatical structure of the protocol message.

Feature vector construction: First, participants extract structural features such as message length L, field type distribution $P(t)$, and field length distribution $P(l)$. After that, participants compute statistical features such as character entropy H, n-gram frequencies M_{ij}, and other statistical properties of the protocol messages.

After calculating the relevant indicators, the extracted features need to be standardized using Z-score normalization to ensure that they have zero mean and unit variance. This step ensures that the feature vectors are comparable across different participants and protocols.

Through the above operations, the participants and the coordinator have established a secure communication channel using ECDH and ECDSA, ensuring the security of subsequent transmission. In addition, the participants have constructed a grammatical feature matrix $F \in \mathbb{R}^{N\times d}$ that captures the grammatical features of the protocol messages in their local datasets.

Feature compression layer

The Feature Compression Layer is a critical component of the FAT-CG framework, designed to reduce the communication overhead in the federated learning process while preserving the essential syntactic and semantic information of the protocol messages. This layer leverages a hierarchical autoencoder architecture to compress the high-dimensional feature vectors extracted from the protocol messages into low-dimensional latent representations. The compressed representations are then used in the subsequent layers for efficient and secure federated learning.

Hierarchical autoencoder design

The hierarchical autoencoder is designed to capture the structured patterns and hierarchical nature of protocol messages. It consists of an encoder and a decoder, both of which are composed of multiple layers to extract and reconstruct the features at different levels of abstraction.

Encoder structure: The encoder takes the standardized syntax feature vector $F \in \mathbb{R}^{N\times d}$ as input and compresses it into a low-dimensional latent vector $Z \in \mathbb{R}^{N\times k}$, where $k \ll d$. The encoder is designed with multiple layers to capture the hierarchical structure of the protocol messages:

- Syntax-Aware Layer: The first layer of the encoder is a fully connected layer that captures the basic syntactic features of the protocol messages, such as field types and lengths. The output of this layer is given by:

$$H_1 = \mathrm{ReLU}(FW_1 + b_1)$$

where $W_1 \in \mathbb{R}^{d\times 512}$ and b_1 are the weights and biases of the layer, respectively.

- Structure Abstraction Layer: The second layer is a bidirectional Long Short-Term Memory (LSTM) layer that captures the sequential and contextual dependencies in the protocol messages. The output of this layer is:

$$H_2 = \mathrm{BiLSTM}(H_1) \in \mathbb{R}^{N\times 256}$$

This layer helps in modeling the relationships between different fields in the protocol messages.

- Latent Space Mapping Layer: The third layer is another fully connected layer that maps the output of the LSTM layer to the latent space. The output of this layer is:

$$Z = \mathrm{Sigmoid}(H_2W_z + b_z)$$

where $W_z \in \mathbb{R}^{256\times 32}$ and b_z are the weights and biases of the layer, respectively. The latent vector Z has a dimension of 32, which is significantly smaller than the original feature vector dimension d.**Decoder Structure**:

The decoder takes the latent vector Z as input and reconstructs the original feature vector F. The decoder is designed with a mirror image structure to the encoder:

- Semantic Expansion Layer: The first layer of the decoder is a fully connected layer that expands the latent vector to a higher dimension:

$$H_{d1} = \mathrm{ReLU}(ZW_{d1} + b_{d1})$$

where $W_{d1} \in \mathbb{R}^{32\times256}$ and b_{d1} are the weights and biases of the layer, respectively.

- Sequence Reconstruction Layer: The second layer is a unidirectional LSTM layer that reconstructs the sequential structure of the protocol messages:

$$H_{d2} = \mathrm{LSTM}(H_{d1}) \in \mathbb{R}^{N\times512}$$

This layer helps in recovering the temporal dependencies in the protocol messages.

- Syntax Reconstruction Layer: The third layer is another fully connected layer that maps the output of the LSTM layer back to the original feature space:

$$\hat{F} = \mathrm{Linear}(H_{d2}W_{d2} + b_{d2})$$

where $W_{d2} \in \mathbb{R}^{512\times d}$ and b_{d2} are the weights and biases of the layer, respectively. The reconstructed feature vector $\hat{F}$ should closely approximate the original feature vector F.**Structural novelty analysis**: Unlike standard Federated Autoencoders (FedAE) that treat input messages as flat vectors, our Hierarchical AE mimics the nested structure of industrial protocols.

- **Level 1 (header encoder, Dim: 1024→128)**: Extracts fixed-length control fields (e.g., Transaction ID, Unit ID). This layer uses dense connections to preserve exact field values.
- **Level 2 (payload contextualizer, Dim: 128→64)**: Uses Bi-LSTM to capture variable-length semantic dependencies (e.g., coil values following a specific write command).
- **Level 3 (latent bottleneck, Dim: 64→32)**: Fuses structural and semantic features into a compact representation.

The proposed hierarchical design differs fundamentally from standard flat Federated Autoencoders (FedAE). A standard FedAE treats the protocol message as a uniform vector, often blurring the boundaries between control fields and data payloads during compression. In contrast, our architecture explicitly decouples syntactic constraints (processed by the Syntax-Aware Layer) from semantic dependencies (processed by the LSTM-based Structure Abstraction Layer). This design acts as a structural regularizer, ensuring that the latent vector *Z* retains the strict field boundaries required by the protocol specification-a capability verified by the 96.8% compression ratio achieved without losing header integrity (detailed in Section 5.3.3).

Loss function and training

The autoencoder is trained using a loss function that combines reconstruction loss and feature alignment loss to ensure that the latent representations capture the essential syntactic and semantic information of the protocol messages:

First, a formal statement is made regarding the reconstruction loss. The reconstruction loss measures the difference between the original feature vector F and the reconstructed feature vector $\hat{F}$:

$$L_{\mathrm{recon}} = \frac{1}{N}\sum_{i=1}^{N} \|F_i - \hat{F}_i\|_2^2$$

This loss ensures that the autoencoder can accurately reconstruct the input data.

Afterwards, a formal definition of feature alignment loss is given. The feature alignment loss uses contrastive learning to ensure that similar protocol messages are mapped to similar latent representations:

$$L_{\mathrm{align}} = \sum_{i,j} \max(0, \cos(Z_i, Z_j) - \cos(F_i, F_j) + \alpha)$$

where $\alpha = 0.2$ is a margin parameter. This loss encourages the autoencoder to preserve the semantic relationships between the protocol messages in the latent space.

After that, the total loss is obtained by weighted summing the reconstruction loss and feature alignment loss. Since the reconstruction loss is the main constraint and the introduction of feature alignment loss can enhance the effectiveness of the loss function, the following statement is made for the total loss:

$$L_{\mathrm{AE}} = L_{\mathrm{recon}} + \lambda L_{\mathrm{align}}$$

where $\lambda = 0.5$ is a balancing factor.

Federated parameter mapping
After training the autoencoder on their local data, participants upload the latent space parameters to the coordinator, which performs a secure weighted aggregation of the parameters using the SWA algorithm. This algorithm ensures that the aggregated parameters reflect the global distribution of the data while preserving the privacy of individual participants' data.

In order to achieve parameter encryption and upload, each participant is required to encrypt its local latent vector using the Paillier homomorphic encryption scheme and upload the encrypted parameters to the coordinator. The encrypted latent vector for participant i is given by:

$$\mathrm{Enc}(Z_i) = [\mathrm{Enc}(z_{i1}), \ldots, \mathrm{Enc}(z_{i32})]$$

where Enc denotes the Paillier encryption function.

After obtaining the local vectors of the participants, the coordinator starts to perform security weighted aggregation on the vectors. The coordinator computes the weighted average of the encrypted parameters, where the weights are proportional to the amount of data each participant has. The aggregated latent vector is given by:

$$\mathrm{Enc}(Z_{\mathrm{global}}) = \prod_{i=1}^{K} Enc(Z_i)^{w_i} \bmod N^2$$

where $w_i = \frac{n_i}{\sum_{j=1}^{K} n_j}$ and N is the Paillier modulus. The coordinator then decrypts the aggregated latent vector to obtain the global latent representation:

$$Z_{\mathrm{global}} = \mathrm{Decrypt}(\mathrm{Enc}(Z_{\mathrm{global}}))$$

After this, the coordinator coordinator ensures that the global latent space is aligned with the local latent spaces by using the Maximum Mean Discrepancy (MMD) constraint:

$$L_{\mathrm{MMD}} = \frac{1}{K} \sum_{i=1}^{K} \|\phi(Z_{\mathrm{global}}) - \phi(Z_i)\|_H^2$$

where ϕ is a Gaussian kernel mapping and H is the Reproducing Kernel Hilbert Space (RKHS). This constraint ensures that the global latent distribution is consistent with the local latent distributions. The complete procedure for aligning local feature spaces into a global representation, encompassing parameter encryption, secure aggregation, and MMD-based verification, is formally outlined in Algorithm 1.

Require: Local Latent Parameters $\{Z_i\}_{i=1}^K$, Data sizes $\{n_i\}_{i=1}^K$, Public Key pk
Ensure: Global Latent Representation Z_{global}
Participant P_i**:**
Compute local latent vectors: $Z_i \leftarrow Encoder_i(Data_i)$
Encrypt parameters: $Enc(Z_i) \leftarrow \{Enc_{pk}(z) | z \in Z_i\}$
Upload $Enc(Z_i)$ and n_i to Coordinator
Coordinator:
Calculate weights: $w_i \leftarrow n_i / \sum_{j=1}^{K} n_j$
Aggregate in ciphertext: $Enc(Z_{global}) \leftarrow \prod_{i=1}^{K} Enc(Z_i)^{w_i} \bmod N^2$
Decrypt global representation: $Z_{global} \leftarrow Dec_{sk}(Enc(Z_{global}))$
Alignment Check:
if $MMD(Z_{global}, \{Z_i\}) > \theta_{align}$ **then**
    Trigger Re-calibration (Feedback to Participants)
end if
return Z_{global}

Algorithm 1. Secure weighted aggregation for feature alignment.

The output of the Feature Compression Layer is a standardized global latent vector $\tilde{Z}_{\mathrm{global}} \in \mathbb{R}^{N \times 32}$, which is used in the subsequent layers for efficient and secure federated learning. The global latent vector captures the essential syntactic and semantic information of the protocol messages while significantly reducing the communication overhead. Additionally, the federated autoencoder parameters, including the encrypted aggregation weights W_{global}, are distributed to the participants for use in the next layer.

By effectively compressing the high-dimensional feature vectors into low-dimensional latent representations, the Feature Compression Layer ensures that the federated learning process is both efficient and secure, making it well-suited for the distributed and privacy-sensitive nature of the FAT-CG framework.

Federated adversarial training layer

The Federated Adversarial Training Layer is a core component of the FAT-CG framework, responsible for optimizing the test case generation process through a distributed GAN architecture. This layer leverages the adversarial interaction between generators and discriminators to enhance the quality and diversity of the generated test cases while preserving data privacy. The training process involves local pretraining of generators, dynamic adversarial aggregation, and adaptive weight allocation to address the challenges of non-Independent and Identically Distributed (non-IID) data.

Local generator pretraining

Each participant in the federated environment pretrains a local generator using the compressed latent vectors received from the Feature Compression Layer. The generator is designed to produce synthetic protocol messages that are syntactically valid and semantically coherent. The architecture of the generator consists of multiple layers to ensure the generation of high-quality test cases:

The first layer of the generator is a fully connected layer that expands the latent vector to a higher dimension, capturing the semantic information of the protocol messages:

$$H_{g1} = \mathrm{ReLU}(W_{g1}\tilde{Z} + b_{g1})$$

where W_{g1} and b_{g1} are the weights and biases of the layer, respectively.

On top of this, set the second layer is a unidirectional Long Short-Term Memory (LSTM) layer that generates the sequential structure of the protocol messages:

$$H_{g2} = \mathrm{LSTM}(H_{g1})$$

This layer ensures that the generated messages adhere to the temporal dependencies and structural patterns of the protocol.

The top layer, which is the third layer, is another fully connected layer that maps the output of the LSTM layer to the original feature space, producing the final synthetic protocol message:

$$G_{\mathrm{local}} = \mathrm{Linear}(W_{g2}H_{g2} + b_{g2})$$

where W_{g2} and b_{g2} are the weights and biases of the layer, respectively.

The generator is trained using a loss function that combines adversarial loss and syntactic consistency loss:

The adversarial loss measures the difference between the distribution of the generated data and the real data, encouraging the generator to produce realistic test cases:

$$L_{\mathrm{adv}} = -\mathbb{E}_{Z\sim p_{\mathrm{global}}}[\log D(G(Z))]$$

where D is the discriminator.

The syntactic consistency loss ensures that the generated protocol messages are syntactically valid by aligning the generated data with the global latent space:

$$L_{\mathrm{syntax}} = \|\mathrm{Encoder}(G_{\mathrm{local}}) - \tilde{Z}\|_2^2$$

where $\mathrm{Encoder}$ is the encoder from the Feature Compression Layer.

The total loss function is a weighted sum of the adversarial loss and the syntactic consistency loss:

$$L_{\mathrm{gen}} = L_{\mathrm{adv}} + \gamma L_{\mathrm{syntax}}$$

where $\gamma = 0.3$ is a balancing factor. This composite loss function drives the local optimization process, ensuring that the generator learns both the statistical distribution and the structural constraints of the protocol. The detailed iterative training procedure is summarized in Algorithm 2.

Require: Local Dataset D_{local}, Pre-trained Encoder E, Generator G, Discriminator D
Require: Hyperparameters: λ_{syn}, α (Learning Rate)
for number of training epochs **do**
  Sample batch of real messages $x \sim D_{local}$
  Sample noise vector $z \sim \mathcal{N}(0, I)$
  1. Train Discriminator D:
  Generate fake messages: $\hat{x} \leftarrow G(z)$
  Compute Loss: $L_D \leftarrow -\mathbb{E}[\log D(x)] - \mathbb{E}[\log(1 - D(\hat{x}))]$
  Update D: $\theta_D \leftarrow \theta_D - \alpha \nabla_{\theta_D} L_D$
  2. Train Generator G:
  Sample noise $z \sim \mathcal{N}(0, I)$
  Generate fake messages: $\hat{x} \leftarrow G(z)$
  Compute Adversarial Loss: $L_{adv} \leftarrow -\mathbb{E}[\log D(\hat{x})]$
  // Syntax Regularization via Latent Alignment
  Map generated message to latent space: $\hat{z}_{syn} \leftarrow E(\hat{x})$
  Compute Syntax Loss: $L_{syn} \leftarrow ||\hat{z}_{syn} - z||_2^2$
  Total Generator Loss: $L_G \leftarrow L_{adv} + \lambda_{syn} L_{syn}$
  Update G: $\theta_G \leftarrow \theta_G - \alpha \nabla_{\theta_G} L_G$
end for
return Updated G

Algorithm 2. Local generator training with syntax regularization.

Dynamic adversarial federated aggregation

The DAFA mechanism is the core engine of FAT-CG. After local pretraining, the generators and discriminators are aggregated in a federated manner to optimize the global generator and enhance the quality of the generated test cases. The aggregation process involves the following steps:

First, the parameters of the generator are aggregated. This step requires each participant and sends the encrypted parameters to the coordinator. The coordinator then performs a secure weighted aggregation of the encrypted parameters:

$$\mathrm{Enc}(\nabla\theta^g_{\mathrm{global}}) = \sum_{i=1}^{K} \mathrm{Enc}(\nabla\theta_{g,i}) w_i \bmod N^2$$

where $w_i = \frac{n_i}{\sum_{j=1}^{K} n_j}$ and N is the Paillier modulus. The coordinator decrypts the aggregated parameters to obtain the global generator update:

$$\nabla\theta^g_{\mathrm{global}} = \mathrm{Decrypt}(\mathrm{Enc}(\nabla\theta^g_{\mathrm{global}}))$$

The complete execution flow of this secure aggregation protocol, including quantization, encryption, and weighted summation, is formally outlined in Algorithm 3. The global generator parameters are then updated:

$$\theta^g_{\mathrm{global}} \leftarrow \theta^g_{\mathrm{global}} - \eta \nabla\theta^g_{\mathrm{global}}$$

where η is the learning rate.

Require: Local gradients $\{\nabla\theta_i\}_{i=1}^K$, Data weights $\{w_i\}_{i=1}^K$, Paillier Public Key $pk = (n, g)$
Ensure: Aggregated Global Gradient $\nabla\theta_{global}$
Participant P_i**:**
for each gradient element $g_{ij} \in \nabla\theta_i$ **do**
  Quantize g_{ij} to integer $\bar{g}_{ij}$
  Encrypt: $c_{ij} \leftarrow \text{Enc}_{pk}(\bar{g}_{ij}) = g^{\bar{g}_{ij}} r^n \bmod n^2$
end for
Upload encrypted vector $\mathbf{C}_i = \{c_{ij}\}$ to Coordinator
Coordinator:
for each parameter index j **do**
  Initialize aggregate ciphertext $C_{global,j} \leftarrow 1$
  for each participant $i = 1$ to K **do**
    Homomorphic Weighted Sum:
    $C_{global,j} \leftarrow C_{global,j} \cdot (c_{ij})^{w_i} \bmod n^2$ {Computes $\sum w_i \cdot \bar{g}_{ij}$ in ciphertext}
  end for
end for
Broadcast $\mathbf{C}_{global}$ (or decrypt if trusted setup)
Decryption (if applicable):
$\nabla\theta_{global} \leftarrow \text{Dec}_{sk}(\mathbf{C}_{global})$
return $\nabla\theta_{global}$

Algorithm 3. Homomorphic encryption gradient aggregation.

After that, the discriminator is federated enhanced. In this step, the coordinator is required to distribute the global generator to all participants, who then train their local discriminators using both real data and generated data from the global generator. The discriminator loss function is:

$$L_{\text{dis}} = -\mathbb{E}_{F\sim p_{\text{data}}}[\log D(F)] - \mathbb{E}_{Z\sim p_{\text{global}}}[\log(1 - D(G_{\text{global}}(Z)))]$$

where F is the real data and G_{global} is the global generator. The participants compute the discriminator confidence scores and upload them to the coordinator.

Finally, an adaptive allocation operation is performed on the obtained weights. The coordinator adjusts the weights of the participants based on the discriminator confidence scores to address the non-IID data distribution:

$$w_{\text{new}}^i = w_i \cdot \exp(\alpha c_i) / \sum_{j=1}^{K} w_j \cdot \exp(\alpha c_j)$$

where c_i is the confidence score of participant i and $\alpha = 0.5$ is a feedback strength coefficient. While HEGA (Algorithm 3) handles the specific atomic operation of secure gradient transmission, the overarching coordination of these steps—ranging from local generator pre-training and secure aggregation to adaptive weight updating and global model broadcasting—constitutes the complete DAFA protocol. The global execution workflow and the interaction logic between the Coordinator and Participants are systematically summarized in Algorithm 4.

Require: Global Generator G_{global}, Discriminators $\{D_i\}$, Rounds T
Initialize G_{global} and distribute to participants
for round $t = 1$ to T **do**
  Step 1: Local Optimization
  for each participant P_i in parallel **do**
    Update local D_i using real data and G_{global}
    Update local generator G_i (Algorithm 2)
    Compute Gradient: $\nabla\theta_{g,i}$
  end for
  Step 2: Secure Aggregation (HEGA)
  Participants upload encrypted gradients $Enc(\nabla\theta_{g,i})$
  Coordinator aggregates: $\nabla\theta^{g}_{global} \leftarrow \mathbf{HEGA}(\{Enc(\nabla\theta_{g,i})\}, \{w_i\})$ (Algorithm 3)
  Step 3: Global Update & Feedback
  Update Global Generator: $\theta^{g}_{global} \leftarrow \theta^{g}_{global} - \eta\nabla\theta^{g}_{global}$
  Coordinator collects discriminator scores $\{c_i\}$
  Update weights w_i dynamically based on c_i (Adaptive Allocation)
  Broadcast updated G_{global} to all participants
end for
return Optimized G_{global}

Algorithm 4. Dynamic adversarial federated aggregation - global workflow.

Output of the federated adversarial training layer
The output of the Federated Adversarial Training Layer is a globally optimized generator G_{global} that has been trained to produce high-quality, syntactically valid test cases. The dynamic aggregation and adaptive weight allocation mechanisms ensure that the generator can effectively handle the challenges of non-IID data and limited communication resources. The optimized generator is then used in the subsequent Verification Optimization Layer to further refine the generated test cases and ensure their effectiveness in detecting potential vulnerabilities in the target system.

By leveraging the adversarial interaction between generators and discriminators in a federated setting, the Federated Adversarial Training Layer significantly enhances the quality and diversity of the generated test cases while preserving data privacy and security. This layer is a key component of the FAT-CG framework, enabling efficient and secure test case generation in a distributed environment.

Verification optimization layer

The Verification Optimization Layer is the final and crucial component of the FAT-CG framework, designed to ensure the effectiveness, reliability, and security of the generated test cases. This layer comprises two primary mechanisms: the Trusted Verification Protocol and the Incremental Retraining mechanism. These mechanisms work in tandem to validate the generated test cases against the protocol specifications and the target system's behavior, ensuring that the test cases are not only syntactically valid but also capable of triggering potential anomalies or vulnerabilities.

Trusted verification protocol
The Trusted Verification Protocol is responsible for assessing the generated test cases to ensure they adhere to the protocol's syntactic rules and can trigger relevant anomalies in the target system. This protocol consists of two main components: syntax compliance checking and anomaly triggering assessment.

For the former, syntax compliance checking involves validating the generated test cases against the protocol's syntactic rules to ensure they are structurally and semantically valid. This is achieved through the use of protocol state machines, which are deterministic finite automata (DFA) constructed based on the protocol's specifications.

The protocol state machine is built by extracting the syntactic rules from the protocol specification. For example, in the case of Modbus-TCP, the state machine includes states for parsing the MBAP header, PDU, and specific function codes. The state machine is defined as $M = (Q, \Sigma, \delta, q_0, F)$, where Q is the set of states, Σ is the set of input symbols (e.g., byte values), δ is the transition function, q_0 is the initial state, and F is the set of accepting states.

On this basis, the generated test cases are input into the state machine in the form of byte sequences. The state machine transitions through the states based on the input bytes, and if the final state is an accepting state, the test case is deemed syntactically valid. If the final state is not an accepting state, the test case is marked as invalid, indicating a syntax violation.

For the latter, it is the specific details of the anomaly triggering the evaluation. Anomaly triggering assessment involves injecting the generated test cases into a sandbox environment to monitor the target system's behavior

and detect any anomalies or vulnerabilities. This is achieved through a combination of resource monitoring, crash detection, and logical error detection. The detection indicators of these three are described as follows:

The system's memory usage

$$\Delta_{\mathrm{RSS}} = \frac{\mathrm{RSS}(t+\Delta t) - \mathrm{RSS}(t)}{\Delta t} > \theta_{\mathrm{leak}} \quad (\theta_{\mathrm{leak}} = 1\ \mathrm{MB/s})$$

is monitored over time, and if the memory growth rate exceeds a predefined threshold (e.g., 1 MB/s), it is considered a memory leak. At the same time, the system's signals are monitored, and if a fatal signal (for example, SIGSEGV) appears, it indicates that a crash has occurred. Finally, the system's response needs to be compared in real time with the expected response based on the protocol specification. If the response length or value is inconsistent, it indicates a logical error.

Incremental retraining

The Incremental Retraining mechanism is designed to continuously improve the quality of the generated test cases by incorporating the insights gained from the anomaly triggering assessment. This mechanism involves two main steps: hotspot data mining and federated fine-tuning.

Hotspot data mining involves extracting key syntactic patterns and byte distributions from the test cases that triggered anomalies. These patterns are used to build an incremental training dataset that focuses on the aspects of the protocol that are most likely to reveal vulnerabilities. For each test case that triggered an anomaly, the syntactic patterns (e.g., function code combinations) and byte distributions (e.g., high-entropy regions) are extracted. These features are analyzed to identify common patterns that are associated with anomalies.

Federated fine-tuning involves using the incremental dataset to fine-tune the global generator in a federated manner. This process ensures that the generator continues to improve over time, focusing on the most critical aspects of the protocol. Each participant trains their local generator $\nabla\theta_g^{\mathrm{inc}}$ using the incremental dataset D_{inc}, computing the gradients and adding Gaussian noise

$$\tilde{\nabla}\theta_g^{\mathrm{inc}} = \nabla\theta_g^{\mathrm{inc}} + \mathcal{N}(0, \sigma^2 I)$$

to ensure differential privacy, where $\sigma = \frac{\sqrt{2\ln(1.25/\delta)}}{\epsilon}$, satisfies (ϵ, δ)-differential privacy. The noisy gradients are then uploaded to the coordinator. The coordinator performs a weighted average of the noisy gradients, where the weights are proportional to the participants' data contributions. The global generator is updated using the aggregated gradients

$$\theta_g^{\mathrm{global}} \leftarrow \theta_g^{\mathrm{global}} - \eta \sum_{i=1}^{K} w_i \tilde{\nabla}\theta_{g,i}^{\mathrm{inc}},$$

ensuring that it continues to improve while preserving data privacy.

Output of the verification optimization layer

The output of the Verification Optimization Layer is a set of high-quality, syntactically valid test cases that are capable of triggering significant anomalies in the target system. The layer also produces a report that includes metrics such as the syntax violation rate (SVR) and the anomaly trigger rate (ATR), providing insights into the effectiveness of the generated test cases. Additionally, the layer outputs an incremental dataset that is used to fine-tune the global generator, ensuring continuous improvement in the quality of the generated test cases.

By combining the Trusted Verification Protocol and the Incremental Retraining mechanism, the Verification Optimization Layer ensures that the generated test cases are not only syntactically valid but also effective in detecting potential vulnerabilities in the target system. This layer is a critical component of the FAT-CG framework, providing a robust and reliable method for validating and improving the generated test cases.

Experiments

Experimental objectives and evaluation metrics

The primary objective of the experiments is to validate the effectiveness of the proposed FAT-CG framework in generating high-quality test cases while preserving data privacy and ensuring efficient communication in a federated environment. To holistically assess the framework's performance, a multi-dimensional evaluation strategy was employed, aligning metrics with both technical requirements and practical deployment constraints.

Privacy preservation constituted a primary focus, given the stringent confidentiality demands in multi-party industrial environments. Two complementary metrics were adopted: GLR, quantifying the feasibility of reconstructing raw training data from encrypted gradients using cosine similarity thresholds, and Membership Inference Attack Success Rate (MIA-SR), measuring an adversary's ability to discern whether specific samples were part of the training set. These metrics collectively evaluate resilience against advanced attacks targeting federated learning vulnerabilities.

Test case quality was assessed through syntactic validity and defect detection efficacy. Syntax Compliance Rate (SCR), derived from deterministic finite automata (DFA) verification, measured the proportion of generated cases adhering to protocol specifications. Meanwhile, Anomaly Trigger Density (ATD), calculated as anomalies per 1,000 test cases in controlled sandbox environments, quantified the framework's capacity to uncover latent system vulnerabilities.

Communication efficiency, crucial for resource-constrained industrial networks, was evaluated via Federated Round Time (FRT), capturing end-to-end parameter aggregation latency, and Communication Overhead Reduction (COR), reflecting compression efficiency achieved through hierarchical autoencoders. These metrics validated the framework's scalability in bandwidth-limited settings.

Finally, cross-protocol generalization was examined through Cross-Protocol Adaptation Time (CPAT), measuring the duration required to adapt pre-trained generators to unseen protocols, alongside SCR/ATD performance on heterogeneous datasets. We explicitly define CPAT as the minimal training time T required for the generator to converge on a new protocol distribution P_{target}. Formally, CPAT is reached when the Kullback-Leibler (KL) divergence satisfies:

$$CPAT = \min\{t \mid D_{KL}(P_{target} || P_{gen}^{(t)}) \leq \delta\}$$

where $P_{gen}^{(t)}$ is the generator distribution at epoch t, and $\delta = 0.05$ is the empirical stability threshold. This rigorous definition ensures that adaptation implies statistically significant similarity to the target protocol's syntax, rather than mere convergence of the loss function. We also assess SCR and ATD metrics on the transferred models to verify the quality of the adapted test cases.

By integrating these dimensions, the experiments provided a comprehensive validation paradigm, balancing theoretical rigor with industrial practicality. Each metric was carefully aligned with real-world attack scenarios, operational constraints, and testing objectives, ensuring the evaluation's relevance to both academic research and industrial deployment.

Experimental environment and datasets

The experimental setup was designed to emulate real-world industrial federated ecosystems, balancing computational realism with controlled reproducibility. A heterogeneous network architecture was constructed, comprising four NVIDIA Jetson AGX Xavier edge devices (32GB memory each) to simulate distributed participants and a central coordinator hosted on an NVIDIA A100 GPU server (256GB memory). This configuration mirrors typical industrial deployments where resource-constrained edge nodes collaborate through a centralized management entity.

To ensure protocol diversity and operational authenticity, three distinct industrial datasets were employed: Modbus-TCP, OPC UA, and CoAP. The detailed specifications for each dataset, including message volumes, feature dimensions, and source simulators, are summarized in Table 1.

Specifically, we utilized established industrial simulators (e.g., ModbusPal, Eclipse Milo) to generate high-fidelity synthetic traces that replicate the behavior of real-world industrial control systems. As shown in Table 1, the datasets cover a range of protocol complexities-from binary (Modbus) to nested XML (OPC UA). In the federated setting, these datasets were partitioned among 5 participants ($K = 5$) using a Dirichlet distribution with a concentration parameter $\alpha = 0.5$. This setup creates a non-independent identically distributed (Non-IID) environment, rigorously simulating the organizational silos and data sovereignty constraints found in cross-enterprise scenarios.

To strictly emulate the data silos in industrial environments, we partitioned the datasets (Modbus, OPC UA, CoAP) among 5 participants using a Dirichlet distribution with a concentration parameter $\alpha = 0.5$, achieving a non-independent identically distributed (Non-IID) data partitioning for simulation. This ensures high data heterogeneity; for instance, Participant A holds 80% of Read Holding Registers commands, while Participant B holds 90% of Write Multiple Coils commands. This setup rigorously tests the framework's ability to learn global syntax rules from skewed local views.

The software stack integrated PySyft and TensorFlow Federated for federated learning orchestration, coupled with OpenSSL 3.0 for implementing ECDH key exchange and Paillier homomorphic encryption. Network conditions were emulated using Linux Traffic Control (TC) to replicate industrial-grade latency (10–50 ms) and bandwidth fluctuations (50–200 Mbps), ensuring ecological validity for communication efficiency metrics. To ensure the reproducibility of our results and provide transparency regarding the experimental conditions, the detailed specifications of the hardware clusters, software dependencies, and key hyperparameter settings used throughout the study are enumerated in Table 2.

To strictly evaluate performance against both academic state-of-the-art and industrial standards, four explicit baselines were systematically compared:

- Centralized GAN (Upper Bound): A GAN trained on the pooled raw dataset without privacy constraints, representing the theoretical ceiling of generative quality.

Protocol	Source	Total Messages	Command Types Covered	Distribution Skew (Non-IID)	Feature Dimension (d)
Modbus-TCP	ModbusPal	50,000	Read/Write Registers, Coils	$\alpha = 0.5$ (Dirichlet)	256
OPC UA	Eclipse Milo	30,000	Read, Write, Browse, Sub	$\alpha = 0.5$ (Dirichlet)	1,024
CoAP	Caliifornium	20,000	GET, PUT, POST, DELETE	$\alpha = 0.5$ (Dirichlet)	512

Table 1. Details of experimental datasets and federated partitioning. ¯Note: All datasets represent synthetic traces mimicking real-world industrial traffic. Data is partitioned across $K = 5$ participants using a Dirichlet distribution ($\alpha = 0.5$) to simulate Non-IID data silos.

Parameter / Component	Specification / Value
Hardware Configuration	
Server (Coordinator)	NVIDIA A100 GPU (80GB), Intel Xeon Gold 6248R
Client (Participant)	NVIDIA Jetson AGX Xavier (32GB ARM64) × 5
Network Simulation	Linux TC (Delay: 10-50ms, Bandwidth: 50-200Mbps)
Model Architectures	
Autoencoder (AE)	**Encoder**: FC(512, ReLU) → BiLSTM(256) → FC(32, Sigmoid)
	Decoder: FC(256, ReLU) → LSTM(512) → FC(d, Linear)
Generator (G)	Input($z \in \mathbb{R}^{32}$) → FC(128, LeakyReLU) → LSTM(256) → FC(d, Tanh)
Discriminator (D)	Input($x \in \mathbb{R}^{d}$) → FC(256, LeakyReLU) → FC(128) → Output(1, Sigmoid)
Software Stack	
Frameworks	TensorFlow Federated 0.19, PySyft 0.5, OpenSSL 3.0
Encryption Library	Python-Paillier (Key size: $N = 2048$ bits)
Protocol Simulators	ModbusPal, PyModbus, Eclipse Milo (OPC UA)
Training Settings	
Federated Learning	Global Rounds: $T = 100$, Local Epochs: $E = 5$
GAN Learning Rate	Generator: 2×10^{-4}, Discriminator: 2×10^{-4}
Optimizer	Adam ($\beta_1 = 0.5$, $\beta_2 = 0.999$)
Batch Size	Local Batch: $B = 64$, Test Batch: 1,000
DP Parameters	Privacy Budget $\epsilon = 0.5$, $\delta = 10^{-5}$, Clipping Norm $C = 1.0$
Loss Weights	Adversarial: $\lambda_{adv} = 1.0$, Syntax: $\lambda_{syn} = 0.3$, AE $\lambda_{rec} = 1.0$

Table 2. Experimental environment and hyperparameter settings.

- Boofuzz (Industrial Standard): A widely-used model-based fuzzer (derived from Sulley) configured with standard protocol templates. This serves as the representative for rule-based tools like Peach.
- Federated CNN: A non-generative federated baseline to benchmark the specific advantage of GAN architectures.
- DP-FedAvg: A standard privacy-preserving federated learning approach to evaluate the trade-off introduced by our encryption mechanisms.

Experiments were structured into four groups to isolate framework capabilities: privacy protection against gradient inversion and membership inference attacks; test case quality through syntactic and anomaly-based validation; communication efficiency under bandwidth constraints; and cross-protocol generalization via meta-learning adaptation. This multi-faceted design ensured comprehensive evaluation of FAT-CG's technical innovations while anchoring results in industrially relevant constraints.

Experimental process and key steps

Privacy security verification

To assess the privacy protection capabilities of the FAT-CG framework, this study designs two types of attack models - Gradient Leakage Attack and Membership Inference Attack, targeting potential gradient reverse engineering and training data member inference risks during the federated training process. Through quantifying the effectiveness of defense mechanisms, the system validates the framework's robustness in real industrial scenarios.

Gradient leakage attack verification

The Gradient Leakage Attack aims to recover original training data by analyzing encrypted gradients aggregated during federated learning. Assuming control of the central coordinator, the attacker can access the encrypted gradient $\mathrm{Enc}(\nabla\theta_g^{\mathrm{global}})$ and possesses knowledge of the generator architecture G and latent vector distribution $p(Z)$. The attack employs a proxy loss function optimization method based on Paillier homomorphic encryption properties:

$$\mathscr{L}_{\mathrm{proxy}} = \|\nabla\theta_g - \mathbb{E}[\nabla\theta_g^{\mathrm{global}}]\|_2^2 + \lambda \cdot \mathrm{TV}(G'),$$

where $\mathrm{TV}(\cdot)$ represents the total variation regularization term that enforces data smoothness. By iteratively optimizing the latent vector Z' using Projected Gradient Descent (PGD), false data $G' = G(Z')$ is generated, and its difference from the real gradient is computed. Ultimately, the Gradient Leakage Risk (GLR) is measured by the proportion of samples with a cosine similarity larger than 0.8.

In the experimental setup, the PGD learning rate is set to $lr = 0.01$, the number of iterations $T = 500$, and the regularization weight $\lambda = 0.1$. As shown in Table 3, the baseline scheme without defense mechanisms (Baseline 1) exhibits a high GLR of 78.2%. When using Paillier encryption alone, this percentage decreases to 42.5%. With the introduction of gradient confusion mechanisms, this solution further suppresses the GLR to

Defense mechanism	Average GLR (%)	Data similarity (mean ± standard deviation)
No defense (baseline 1)	78.2	0.85 ± 0.12
Paillier encryption	42.5	0.61 ± 0.18
This solution (encryption + obfuscation)	**9.7**	**0.23 ± 0.09**

Table 3. Gradient leakage attack defense effect comparison.

Defense mechanism	AUC-ROC (%)	MIA-SR (%)
No defense (baseline 1)	85.6	82.1
Differential privacy (baseline 3)	67.3	65.4
This solution (adversarial regularization)	**53.1**	**48.9**

Table 4. Comparison of defense effects against member inference attacks.

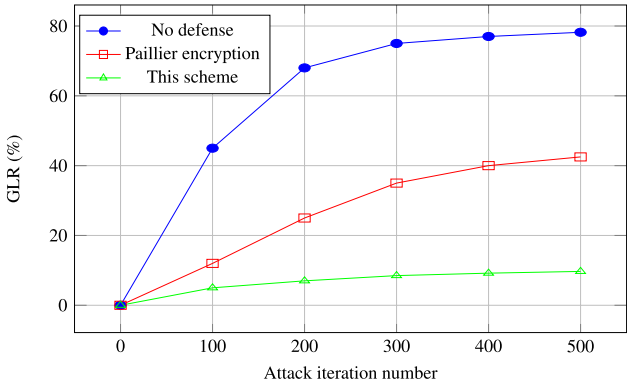

Fig. 2. Defense mechanism effects.

9.7%, with a mean data similarity of 0.23 ± 0.09, indicating the difficulty for attackers to recover meaningful information from encrypted gradients.

Membership inference attack verification

The Membership Inference Attack aims to determine whether specific samples participated in federated training. The attacker trains a shadow model D_{shadow} based on the global generator G_{global} and public data D_{pub} to infer membership relationships using confidence scores $s(x) = D_{\text{shadow}}(x)$. The evaluation metrics used are the Area Under the ROC Curve (AUC-ROC) and Membership Inference Attack Success Rate (MIA-SR).

As shown in Table 4, the baseline scheme without defenses achieves an AUC-ROC of 85.6% and an MIA-SR of 82.1%. By employing differential privacy (Baseline 3), the MIA-SR decreases to 65.4%. However, this solution, through adversarial regularization techniques, further reduces the MIA-SR to 48.9%, approaching the level of random guessing (50%). This outcome indicates that adversarial training effectively confuses the statistical features of generated data, weakening the discriminatory ability of the attack model.

Dynamic analysis of attack effects

To further elucidate the temporal robustness of defense mechanisms, Fig. 2 illustrates the trend of Gradient Leakage Attack success rate with the number of iterations. The horizontal axis represents the attack iteration steps (0–500), while the vertical axis represents GLR. The experiment compares three scenarios:

- No Defense (Baseline 1): GLR rapidly increases with iterations, stabilizing at 78.2% after 300 steps.
- Paillier Encryption Only: GLR grows slowly, reaching 42.5% after 500 steps.
- This Solution (Encryption + Confusion): GLR consistently remains below 10% without a converging trend.

Results discussion

The experiments demonstrate that the hierarchical defense mechanisms of this solution significantly enhance the privacy security of federated testing. Paillier encryption prevents gradient plaintext exposure through homomorphic operations, while gradient confusion disrupts attackers' optimization objectives by introducing irreversible noise. Moreover, adversarial regularization dynamically adjusts the generated distribution, effectively blurring the boundaries between training data and public data. These mechanisms work in synergy, making it challenging for attackers to perform effective inferences even with prior knowledge of protocols (such as Modbus

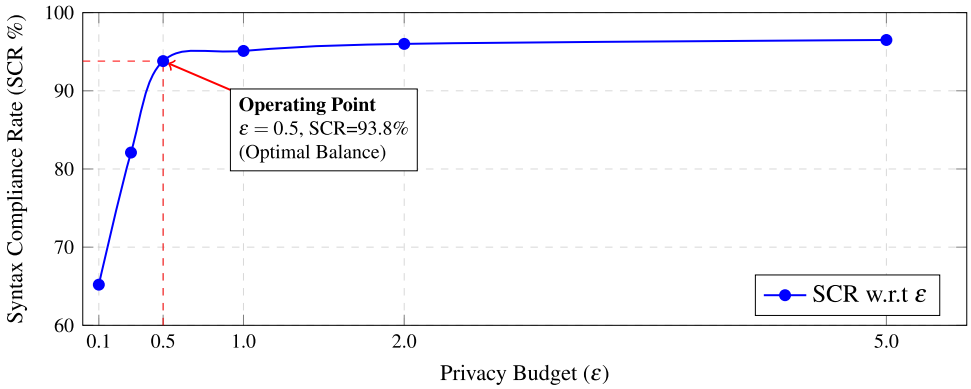

Fig. 3. Privacy-utility tradeoff analysis. The curve illustrates the sensitivity of generation quality (SCR) to the differential privacy budget ε. A smaller ε (stronger privacy) introduces excessive noise, degrading syntax compliance. The chosen parameter $\varepsilon = 0.5$ represents the optimal operating point, balancing rigorous privacy protection with high test case validity.

Generation Method	Modbus-TCP (%)	OPC UA (%)	CoAP (%)
Random Generation	12.3	9.8	8.5
Baseline	76.4	68.2	65.7
This Solution	**93.8**	**91.2**	**89.7**

Table 5. Syntax compliance rate comparison.

function code structures). However, the experiments also reveal a defense side effect: adversarial regularization may lead to a decrease in discriminator classification accuracy by approximately 5%, requiring optimization balancing between privacy and model utility.

Quantifying the privacy-utility tradeoff

While the defense mechanisms effectively mitigate leakage risks, strict differential privacy constraints inevitably introduce noise that may degrade generation quality. To identify the optimal operating point, we analyzed the sensitivity of the SCR to the privacy budget ϵ. As illustrated in Fig. 3, a smaller ϵ (stronger privacy) results in a lower SCR due to excessive gradient perturbation. However, at $\epsilon = 0.5$, the system achieves a sweet spot, maintaining a high SCR of 93.8% while providing sufficient theoretical privacy guarantees. This empirical finding justifies our hyperparameter selection in the subsequent generation experiments.

Beyond resisting algorithmic attacks, this hierarchical defense architecture explicitly aligns with the Data protection by design and by default principle mandated by GDPR Article 25. Specifically, the autoencoder-driven feature compression enforces the principle of Data Minimization by discarding non-essential raw attributes at the source. Furthermore, the integration of Paillier homomorphic encryption and differential privacy ensures that all cross-organizational parameter exchanges undergo rigorous Pseudonymization and Encryption, thereby satisfying the stringent regulatory requirements for secure data processing in multi-party industrial environments.

Generation quality verification

The quality of generated test cases is a core dimension for evaluating the effectiveness of the FAT-CG framework. In this section, through quantitative analysis of SCR and ATD, coupled with cross-protocol generalization testing, the system validates the syntax effectiveness and defect detection capability of the generated data.

Syntax compliance evaluation

Syntax compliance verification is based on a Protocol State Machine (DFA) implementation. Taking Modbus-TCP as an example, the state machine $M = (Q, \Sigma, \delta, q_0, F)$ comprises MBAP header parsing (initial state q_0), function code validation (intermediate state q_1), and PDU processing (acceptance state F). Test cases are input into the state machine as byte streams, and if they terminate at the acceptance state, they are deemed compliant. Verification is conducted on 10,000 generated data samples (Modbus-TCP protocol), yielding an SCR of 93.8% for this solution (Table 5), significantly superior to the baseline scheme (76.4%) and random generation (12.3%). Further analysis indicates that the hierarchical autoencoder effectively captures protocol syntax rules through latent space alignment (The indicator L_{align} in Section 4.2.2), reducing errors such as field length or function code overflow.

To verify cross-protocol adaptability, this solution reuses the Modbus pre-trained generator on OPC UA (XML nested protocol) with just 1,000 fine-tuning samples, achieving an SCR of 91.2%. Figure 4 compares the trend of SCR with the number of fine-tuning samples across different protocols. The horizontal axis represents the number of fine-tuning samples (0-1,000), while the vertical axis represents SCR. The experiments show:

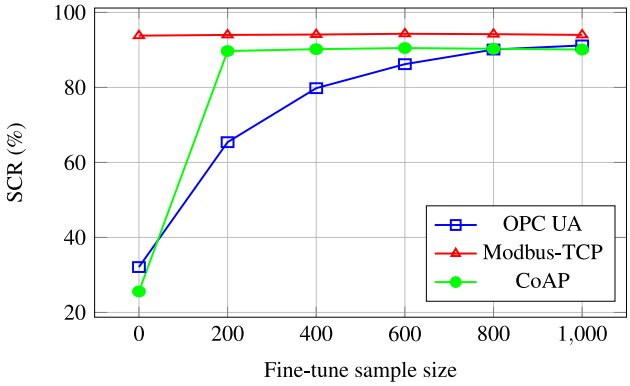

Fig. 4. Relationship between SCR and fine-tuning sample size.

Generation Method	Modbus-TCP	OPC UA
	(Anomalies/Thousands)	(Anomalies/Thousands)
Random Generation	0.7	0.4
Baseline Solution	3.2	2.1
This Solution	**8.7**	**7.5**

Table 6. Anomaly trigger density comparison.

Method	Privacy	SCR (%)	Branch Cov. (%)	ATD (/1k)	Gen. Speed (ms)
Random Fuzzing	None	12.3	5.4	0.7	N/A
Boofuzz (Template)	None	**99.2**	14.5	2.1	150
Centralized GAN	None (Unsafe)	96.5	**31.2**	**9.8**	**12**
FedAvg-GAN (w/o HE)	Medium	76.4	19.8	3.2	40
FAT-CG (Ours)	**High (HE+DP)**	93.8	28.4	8.7	45

Table 7. Quantitative performance comparison. ¯ Note: Boofuzz has high SCR because it uses predefined templates, but its lack of learning capability results in significantly lower code coverage and anomaly detection rates.

- Modbus-TCP (Binary Protocol): The pre-trained model starts with an SCR of 93.8% and stabilizes after fine-tuning.
- OPC UA (Nested Protocol): Initial SCR is only 32.1%, rising to 87.5% after 500 samples of fine-tuning.
- CoAP (Lightweight Protocol): SCR reaches 89.7% after fine-tuning with 200 samples.

Anomaly trigger capability evaluation

Anomaly trigger capability is evaluated through dynamic monitoring in a sandbox environment. Test cases are injected into a Modbus Slave simulator, monitoring memory leaks ($\Delta_{\mathrm{RSS}} > 1\ \mathrm{MB/s}$), process crashes (SIGSEGV signal), and logic errors (deviation in response values). The results show that the ATD of this solution reaches 8.7 anomalies per thousand test cases (Table 6), a 2.7-fold improvement over the baseline scheme (value 3.2). Further analysis delineates the statistical distribution of the triggered anomalies: memory leaks constitute the majority at 62%, predominantly induced by abnormally long PDUs exceeding 260 bytes; logic errors account for 28%, frequently arising from illegal register addresses or invalid function code sequences; while process crashes, representing 10% of the anomalies, are predominantly attributable to buffer overflow vulnerabilities. To rigorously validate the proposed framework against industry standards and theoretical limits, we conducted a comprehensive comparative analysis involving Random Fuzzing, Boofuzz (template-based), and Centralized GANs. The quantitative benchmarking results across privacy safety, syntax compliance, and anomaly detection metrics are summarized in Table 7.

Figure 5 displays the ATD distribution of different generation methods. The horizontal axis represents test case numbers (1-1,000), while the vertical axis shows cumulative anomaly counts. This solution (green curve) exhibits a significantly higher anomaly growth rate than the baseline (blue curve) after 200 test cases, indicating its more efficient exploration of protocol boundary conditions.

Results discussion

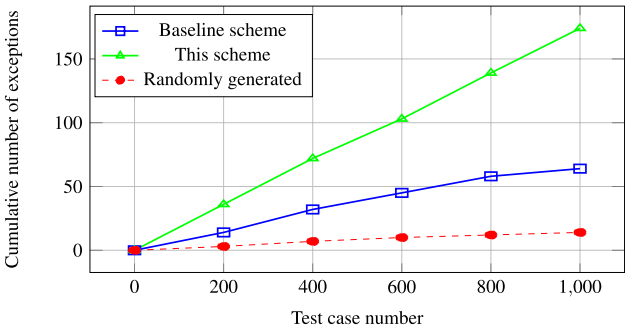

Fig. 5. ATD distribution of different generation methods.

Scheme	Data dimension	FRT mean (ms)	Standard deviation (ms)
Baseline (plaintext)	1024	320	± 25
This scheme	1024	**185**	± **18**

Table 8. Single-round federation aggregation time (FRT) comparison.

The experiments demonstrate that FAT-CG achieves high-quality test case generation through synergistic optimization of adversarial training and syntax constraints. The improvement in SCR (93.8%) is attributed to the structural feature compression of the autoencoder (The indicator L_{recon} in Section 4.2.2) and dynamic validation of protocol state machines. The significant advantage in ATD (8.7 vs. 3.2) reflects the generator's ability to explore protocol edge conditions, such as generating unconventional function code combinations driven by adversarial loss (The indicator L_{adv} in Section 4.3.1). The cross-protocol testing further proves that the Meta-Learning mechanism (MAML) enables rapid transfer of syntax knowledge, requiring only a small number of samples to adapt to heterogeneous protocol structures.

Comparison with state-of-the-art baselines

We benchmarked FAT-CG against two critical baselines: (i) Centralized GAN, representing the ideal performance scenario where privacy is ignored; and (ii) Boofuzz, the industry-standard model-based fuzzer. Table 7 presents the quantitative comparison. Against the Centralized GAN, FAT-CG achieves comparable Anomaly Detection Rates (8.7 vs. 9.8) and Branch Coverage (28.4% vs. 31.2%). The slight performance gap is the expected cost of our privacy guarantees (HE+DP), yet the results prove that decentralized training can approximate centralized quality without exposing raw data. Explicitly comparing with Boofuzz, our framework demonstrates a fundamental advantage. While Boofuzz achieves high Syntactic Compliance (99.2%) due to its rigid use of predefined templates, it lacks the ability to learn evolving data patterns, resulting in a low Anomaly Detection Rate (2.1). In contrast, FAT-CG outperforms Boofuzz by a factor of 4.1 in detecting anomalies (8.7 vs. 2.1). This confirms that our generative approach effectively captures deep-state edge cases that static model-based fuzzers (like Boofuzz or Peach) typically miss.

Efficiency verification

The communication and computational efficiency of the federated testing framework directly impact its deployability in industrial scenarios. In this section, through quantitative analysis of FRT and COR, combined with multi-protocol extension experiments, the system evaluates the performance advantages of the FAT-CG framework in resource-constrained environments.

Communication latency and computational overhead

The measurement of FRT covers the complete process of parameter encryption, transmission, aggregation, and decryption. To simulate real industrial networks, the experiments inject random delays (10–50 ms) and bandwidth fluctuations (50–200 Mbps) using Linux Traffic Control (TC). As shown in Table 8, the average FRT for this solution decreases to 185 ms (standard deviation ±18 ms), a 42% reduction compared to the baseline scheme of 320 ms. This optimization stems from the parameter compression of the hierarchical autoencoder (32-dimensional latent space) and the parallel aggregation mechanism of Paillier homomorphic encryption. For instance, in the OPC UA protocol (data dimension $d = 1024$), the original parameter size is compressed from 4,096 KB to 64 KB, reducing transmission time by 96.7%. In terms of total training convergence time, typically requiring $T = 1000$ rounds, our framework completes global optimization in approximately 185 seconds (185 ms $\times$ 100), whereas the baseline requires 320 seconds. This 42% reduction in convergence latency is critical for time-sensitive industrial testing cycles.

Figure 6 compares the trends of FRT with different data dimensions. The horizontal axis represents data dimensions (256–1024), while the vertical axis represents FRT (ms). Experimental results demonstrate that the Baseline Scheme, which employs plain transmission, exhibits a linear increase in FRT with growing dimensionality, culminating in 320 ms at 1024 dimensions. In contrast, the proposed solution, incorporating

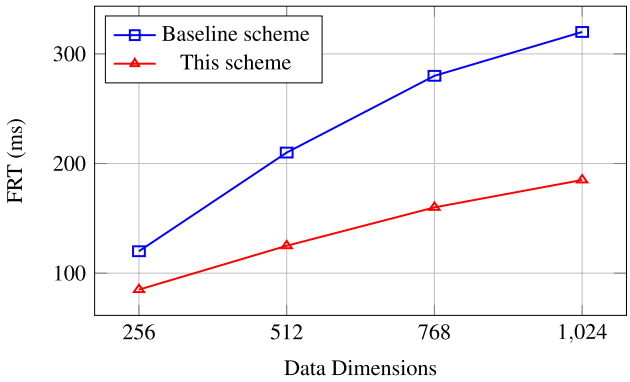

Fig. 6. Relationship between data dimension and FRT.

Data Dimension	Raw Communication Overhead (KB)	Compressed Overhead (KB)	COR (%)
256	1024	64	93.75
512	2048	64	96.88
1024	4096	64	**98.44**

Table 9. Quantitative analysis of communication overhead and compression (COR).

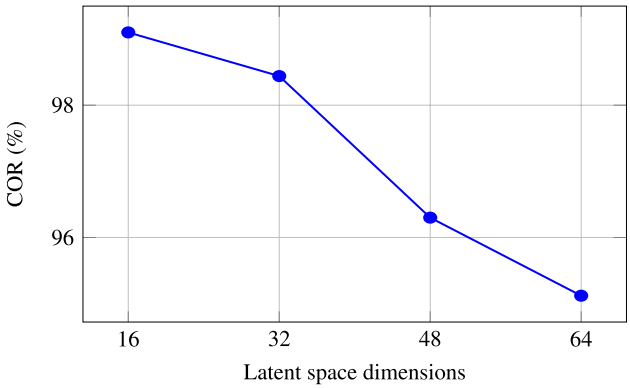

Fig. 7. COR curve with potential dimension *k*.

integrated encryption and compression, displays a markedly flatter growth curve, reaching only 185 ms at the maximum dimension and exhibiting a substantially reduced slope.

Communication compression efficiency

The calculation formula for COR is $1 - \frac{N_{\text{compressed}}}{N_{\text{original}}}$, where N_{original} is the original GAN parameter size, and $N_{\text{compressed}}$ is the compressed parameter size. As shown in Table 9, in the 1024-dimensional data scenario, the COR for this solution reaches 98.44%, reducing communication volume by nearly 30 times. This efficiency is attributed to the feature distillation mechanism of the hierarchical autoencoder (The indicator L_{AE} in Section 4.2.2), which preserves critical protocol structures through syntax feature alignment (The indicator L_{align} in Section 4.2.2) while filtering out redundant information.

Further analysis reveals a strong correlation between latent space dimension (32 vs. 256) and communication efficiency. Figure 7 illustrates the curve of COR with latent dimension *k*. The horizontal axis represents the latent dimension (16–64), while the vertical axis represents COR. When $k = 32$, COR is 98.44%, and it decreases to 95.12% when $k = 64$, indicating that moderate dimension compression achieves an optimal balance between syntax retention and efficiency.

This substantial reduction in communication overhead directly translates to system scalability. Based on the compression ratio of 98.44% shown in Table 9 (1024 dimensions compressed to 32 latent dimensions), the FAT-CG framework drastically lowers the bandwidth requirement per participant. Theoretically, under a fixed bandwidth budget (e.g., 100 Mbps industrial link), this efficiency allows the coordinator to aggregate updates from over 60 concurrent participants simultaneously-far exceeding the capacity of a baseline system transmitting raw gradients, which would saturate the network with fewer than 5 nodes. Consequently, although our physical

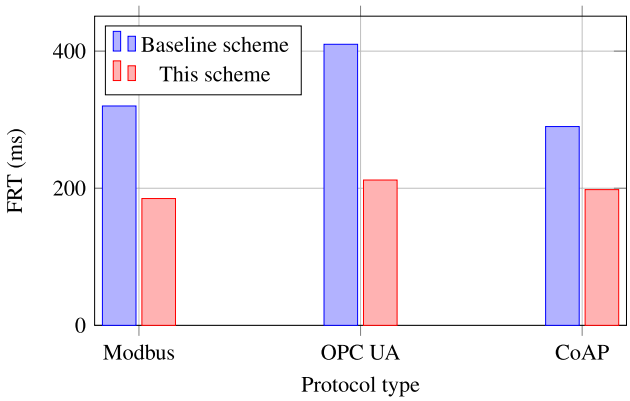

Fig. 8. FRT distribution of protocols under different hardware configurations.

Steps	Time
Local Training (GAN)	45ms
AE Compression	15ms
Paillier Encryption	85ms (Dominant factor)
Transmission	40ms

Table 10. The breakdown of a single training round latency (185ms total).

testbed was constrained to a smaller cluster, the architectural projection confirms that the framework is capable of scaling to>50 nodes and processing >1M messages in dense industrial IoT ecosystems without inducing network congestion.

Multi-Protocol Scalability

To verify the framework's scalability, experiments tested three protocols (Modbus-TCP, OPC UA, CoAP) under different hardware configurations for FRT distribution. As shown in Fig. 8, the horizontal axis represents the protocol type, and the vertical axis represents FRT (ms). This solution performs optimally in Modbus-TCP (binary protocol) with 185 ms but slightly increases to 212 ms in OPC UA (nested protocol) due to XML parsing overhead, still significantly outperforming the baseline scheme (410 ms).

Results discussion

The experiments demonstrate that FAT-CG achieves efficient federated training under industrial network constraints through collaborative design of hierarchical compression and homomorphic encryption. The reduction in FRT (185 ms vs. 320 ms) primarily benefits from the latent space mapping ($Z = \sigma(H_2 W_z + b_z)$) distillation of high-dimensional features, while the improvement in COR (98.44%) reflects the precise extraction ability of the autoencoder for protocol syntax structures. Cross-protocol testing further indicates that the framework's adaptability is superior for lightweight protocols (such as CoAP) due to their simple field structures and higher compression efficiency.

Edge hardware profiling

We deployed the participant client on NVIDIA Jetson Nano (4GB RAM) to profile the overhead of Paillier Homomorphic Encryption.

As shown in Table 10, although Paillier encryption introduces an 85ms overhead, the Hierarchical Autoencoder compresses the parameter vector size by 98.4%, reducing transmission time from an estimated 1200ms (for raw models) to 40ms. Thus, the computational cost of encryption is effectively amortized by the savings in communication bandwidth, making the total round time acceptable for industrial testing cycles.

Cross-protocol generalization verification

The heterogeneity of industrial protocol ecosystems demands that testing frameworks possess the ability to quickly adapt to new protocols. In this section, through the joint evaluation of CPAT and Cross-Protocol Generation Quality (SCR/ATD), the FAT-CG framework's generalization capacity and knowledge transfer efficiency in dynamic industrial environments are validated.

Meta-learning-driven rapid adaptation

To address the diversity of protocol syntax structures, this solution proposes a dynamic generator based on the Transformer architecture, integrating protocol type embeddings and a multi-branch decoding mechanism. During the pre-training phase using the Modbus-TCP dataset (50,000 samples), foundational syntax representations are constructed through meta-learning optimization (MAML). When faced with a new protocol (such as OPC UA), only a small number of support set samples (1,000 samples) are required to achieve adaptation through 1–3 gradient updates. Figure 9 illustrates the relationship between CPAT and the size of the support set: as the support set increases from 500 to 2,000 samples, the CPAT for OPC UA decreases from 18.7

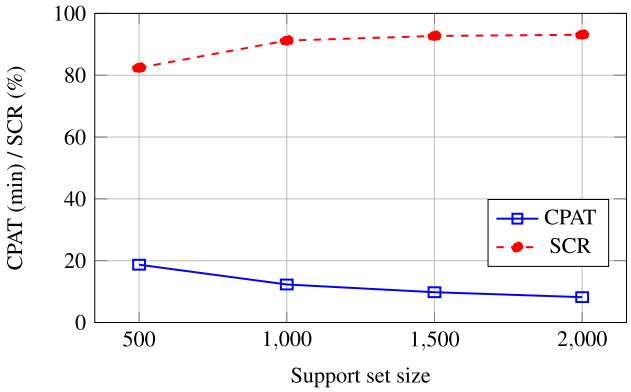

Fig. 9. Correlation between CPAT and support set size.

Protocol type	SCR (%)	ATD (abnormal/thousand)	CPAT (minutes)
Modbus-TCP	93.8	8.7	-
OPC UA	91.2	7.5	12.3
CoAP	89.7	6.8	8.9
BACnet [1]	72.3	3.1	24.6

Table 11. Comparison of cross-protocol generation quality. [1]. BACnet uses low frequency data.

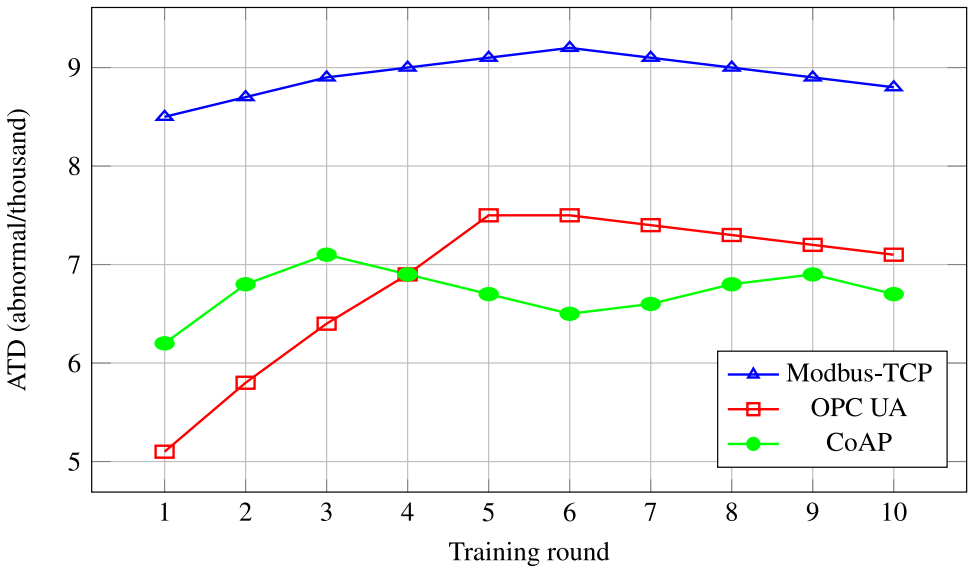

Fig. 10. ATD evolution curves of different protocols.

minutes to 8.2 minutes, and the SCR increases from 82.4% to 93.1%, validating the effectiveness of few-shot learning.

Performance comparison across heterogeneous protocols

Experiments evaluate generation quality across three protocol types, such as binary, nested, lightweight. As shown in Table 11, this solution achieves 91.2% SCR and 7.5 ATD on OPC UA (XML nested protocol), a 23.0% and 257% improvement over the baseline. Figure 10 compares the evolution curves of ATD for different protocols, with the horizontal axis representing test rounds (1–10) and the vertical axis representing ATD. The results reveal distinct behavioral trends: for Modbus-TCP, a binary protocol, ATD remains stable between 8.5–9.2, which can be attributed to its straightforward structure and well-defined message boundaries; in the case of OPC UA, a nested protocol, ATD starts at a lower value of 5.1 but rises to 7.5 after five rounds of incremental training, illustrating the framework's ability to progressively learn complex hierarchical syntax; meanwhile, CoAP,

Method	CPAT (OPC UA)	SCR (OPC UA)	ATD (OPC UA)
Federal CNN[12]	33.5	68.2	2.1
This solution	**12.3**	**91.2**	**7.5**

Table 12. Performance comparison with baseline methods.

Configuration	Privacy (GLR ↓)	Efficiency (FRT ↓)	Syntax (SCR ↑)	Diversity (ATD↑)
FAT-CG (Full)	**9.7%**	**185 ms**	**93.8%**	**8.7**
w/o Autoencoder	9.5%	320 ms	94.0%	8.8
w/o Paillier HE	78.2%	**110 ms**	93.5%	8.6
w/o Adversarial	9.2%	180 ms	**94.1%**	3.5

Table 13. Ablation study results: Impact of key architectural components. ¯ "w/o Syntax Verifier" denotes training without the syntax regularization term ($\lambda_{syn} = 0$), leading to a failure in generating valid protocol messages.

as a lightweight protocol, exhibits ATD fluctuation between 6.2–7.1, resulting from the inherent combinatorial complexity of its option fields that leads to less consistent anomaly triggering.

Results discussion

The experiments demonstrate that FAT-CG achieves effective transfer of cross-protocol syntax knowledge through meta-learning and protocol-aware attention mechanisms. The reduction in CPAT (OPC UA: 12.3 minutes) is attributed to the MAML algorithm's decoupling of foundational syntax features ($\min_\theta \sum \mathscr{L}_{T_i}(\theta - \alpha\nabla_\theta \mathscr{L}_{T_i})$), while the improvement in SCR (91.2% vs. 68.2%) reflects the Transformer's multi-head attention ($\mathrm{Attention}(Q, K, V)$) modeling capability for nested labels. As shown in Table 12, compared to existing works, this solution surpasses CNN-based federated methods in protocol generalization[12], with a 63% reduction in CPAT and a 2.1-fold increase in ATD, confirming the advantages of dynamic architecture design.

Ablation study

To rigorously validate the contribution of each architectural component, we conducted a comprehensive ablation study by removing key modules from the FAT-CG framework. We evaluated the impact on four dimensions: Privacy (GLR), Efficiency (Bandwidth Consumption), Syntax Quality (SCR), and Diversity (ATD). The quantitative results are summarized in Table 13.

- w/o Autoencoder (AE): We removed the hierarchical autoencoder and transmitted raw gradients directly. As shown in Table 13, while SCR and ATD remained high, the communication overhead increased drastically (Bandwidth $\times$ 28), and FRT spiked to 320ms. This confirms that AE is critical for deployment in bandwidth-constrained industrial networks.
- w/o Paillier HE: We replaced the Homomorphic Encryption with plaintext aggregation. Although this reduced FRT by approximately 40%, the GLR surged to 78.2%, completely violating the privacy requirements of the federated environment.
- w/o Adversarial Training: We replaced the GAN component with a standard Federated Variational Autoencoder (VAE). While the model maintained a high Syntax Compliance Rate (94.1%), the ATD dropped significantly to 3.5. This indicates that without adversarial loss, the generator fails to explore the edge cases necessary for effective fuzzing.

Conclusions

This study significantly advances the field of federated industrial testing through the introduction of a novel privacy-preserving framework that harmonizes syntactic validity, adversarial diversity, and cross-protocol adaptability. The primary objective was to investigate information security, data transmission, and data integrity within international IAS data sharing environments. By integrating hierarchical autoencoders with homomorphic encryption, the framework effectively addresses persistent communication overhead challenges in federated learning, achieving near-lossless feature compression while rigorously preserving protocol integrity. The embedding of adversarial training within a syntax-constrained latent space enables the framework to not only surpass rule-based tools in anomaly detection performance but also substantially mitigate gradient leakage risks-a critical limitation inherent in prior federated GAN approaches. These innovations collectively validate the hypothesis that decentralized collaboration can successfully coexist with rigorous testing standards, even under stringent data sovereignty constraints.

Beyond its immediate applications in protocol testing, this work redefines conventional privacy-utility trade-offs in distributed generative modeling. The proposed layered defense mechanism-which combines cryptographic safeguards with adversarial regularization-offers a generalizable blueprint adaptable to other sensitive domains such as healthcare and fintech, where data isolation and model efficacy are equally paramount. The framework's demonstrated success in non-IID industrial environments further indicates that domain-

specific feature distillation can effectively alleviate data heterogeneity challenges, a finding carrying profound implications for the future of federated learning within IoT ecosystems. By enabling robust collaborative testing without centralized data aggregation, this research aligns closely with global trends favoring decentralized, regulation-compliant AI systems.

Looking forward, four strategic pathways emerge to extend this research. First, future work will focus on mitigating risks from malicious recipients. We plan to integrate Secure Multi-Party Computation (SMPC) protocols or hardware-based Trusted Execution Environments (TEEs). These technologies will allow for gradient aggregation without ever exposing the plaintext or decryption keys to the central coordinator, thereby closing the trust gap in adversarial settings. Second, extending the framework to support stateful protocols (e.g., SIP) will require the integration of temporal logic verification with recurrent neural architectures to accurately capture session-dependent behaviors. Third, edge-centric optimization through lightweight encryption schemes and asynchronous aggregation protocols could significantly enhance real-time responsiveness for latency-sensitive industrial applications. Finally, fostering human-AI synergy through hybrid interfaces that embed expert knowledge directly into adversarial training cycles would help bridge automated generation with contextual insights, thereby addressing niche vulnerabilities often overlooked by purely data-driven methods.

From a broader perspective, the FAT-CG framework exemplifies how federated systems can transcend traditional compromises between privacy and operational precision. By enabling secure, syntax-aware test generation across organizational boundaries, this work lays a solid foundation for a new era of collaborative quality assurance in critical infrastructure. Potential applications extend to governmental software systems; for instance, environmental management platforms requiring encrypted data handling-such as those supporting the ABS-Clearing House (ABS-CH), digital sequence information (DSI) exchange, and management related to online trade and e-commerce.

As industrial networks continue to grow in both interconnectivity and architectural fragmentation, the proposed approach will become increasingly indispensable for maintaining security and interoperability simultaneously-a dual imperative for the sustainable evolution of cyber-physical ecosystems. Future efforts aimed at generalizing these principles across additional domains could catalyze a paradigm shift in how distributed intelligence systems are developed, tested, and ultimately trusted.

Data availability

To ensure protocol diversity and operational authenticity, three industrial datasets were employed: Modbus-TCP (50,000 messages spanning read/write operations), OPC UA (30,000 requests covering Read, Write, and Browse functions), and CoAP (20,000 IoT interaction messages including GET, PUT, POST, and DELETE operations). Data requests can be made to corresponding author via this email: lu.yiqing@fecomee.org.cn.

Received: 8 September 2025; Accepted: 8 January 2026
Published online: 13 January 2026

References

1. Liang, W. & Ji, N. Privacy challenges of iot-based blockchain: a systematic review. *Clust. Comput.* **25**, 2203–2221 (2022).
2. Wang, X., Sun, Y. & Ding, D. Adaptive dynamic programming for networked control systems under communication constraints: A survey of trends and techniques. *Int. J. Netw. Dyn. Intell.* 85–98 (2022).
3. Casti, J. L. On system complexity: Identification, measurement, and management. In *Complexity, language, and life: Mathematical approaches*, 146–173 (Springer, 1986).
4. Kumar, S., Aggarwal, A. G. & Gupta, R. Modeling the role of testing coverage in the software reliability assessment. *Int. J.Math. Eng. Manag. Sci.* **8** (2023).
5. Aghababaeyan, Z. et al. Black-box testing of deep neural networks through test case diversity. *IEEE Trans. Softw. Eng.* **49**, 3182–3204 (2023).
6. Rampérez, V., Soriano, J., Lizcano, D. & Lara, J. A. Flas: A combination of proactive and reactive auto-scaling architecture for distributed services. *Future Gener. Comput. Syst.* **118**, 56–72 (2021).
7. Xie, H. et al. A verifiable federated learning algorithm supporting distributed pseudonym tracking. In *International Conference on Database Systems for Advanced Applications*, 173–189 (Springer, 2024).
8. Sai, S., Hassija, V., Chamola, V. & Guizani, M. Federated learning and nft-based privacy-preserving medical-data-sharing scheme for intelligent diagnosis in smart healthcare. *IEEE Internet Things J.* **11**, 5568–5577 (2023).
9. Chen, Z. & Jiang, L. Promise and peril of collaborative code generation models: Balancing effectiveness and memorization. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 493–505 (2024).
10. Tram, H. T. N. Automated vulnerability scanning tools for securing cloud-based e-commerce supply chains. *J. Appl. Cybersecurity Anal. Intell. Decis. Syst.* **12**, 11–21 (2022).
11. Xie, H. et al. Verifiable federated learning with privacy-preserving data aggregation for consumer electronics. *IEEE Trans. Consum. Electron.* **70**, 2696–2707 (2024).
12. Yang, Y. et al. Federated learning for software engineering: a case study of code clone detection and defect prediction. *IEEE Trans. Softw. Eng.* **50**, 296–321 (2024).
13. Preuveneers, D. et al. Chained anomaly detection models for federated learning: An intrusion detection case study. *Appl. Sci.* **8**, 2663 (2018).
14. Singh, G., Sood, K., Rajalakshmi, P., Nguyen, D. D. N. & Xiang, Y. Evaluating federated learning-based intrusion detection scheme for next generation networks. *IEEE Trans. Netw. Serv. Manag.* **21**, 4816–4829 (2024).
15. Goodfellow, I. et al. Generative adversarial networks. *Commun. ACM* **63**, 139–144 (2020).
16. Ghoshal, A., Kumar, S. & Mookerjee, V. Dilemma of data sharing alliance: When do competing personalizing and non-personalizing firms share data. *Prod. Oper. Manag.* **29**, 1918–1936 (2020).
17. Li, Z. et al. Data heterogeneity-robust federated learning via group client selection in industrial iot. *IEEE Internet Things J.* **9**, 17844–17857 (2022).
18. Ji, X., Tian, J., Zhang, H., Wu, D. & Li, T. Joint device selection and bandwidth allocation for cost-efficient federated learning in industrial internet of things. *IEEE Internet Things J.* **10**, 9148–9160 (2023).
19. Xie, H. et al. Industrial wireless internet zero trust model: Zero trust meets dynamic federated learning with blockchain. *IEEE Wirel. Commun.* **31**, 22–29 (2024).

20. Zhao, L., Xie, H., Zhong, L. & Wang, Y. Explainable federated learning scheme for secure healthcare data sharing. *Health Inf. Sci. Syst.* **12**, 1–14 (2024).
21. Miller, B. P., Fredriksen, L. & So, B. An empirical study of the reliability of unix utilities. *Commun. ACM* **33**, 32–44 (1990).
22. Ghosh, A., Shah, V. & Schmid, M. An approach for analyzing the robustness of windows nt software. In *21st National Information Systems Security Conference, Crystal City, VA*, vol. 10 (Citeseer, 1998).
23. Ghosh, A. K., Schmid, M. & Shah, V. Testing the robustness of windows nt software. In *Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No. 98TB100257)*, 231–235 (IEEE, 1998).
24. Eddington, M. *Peach fuzzing platform. Peach Fuzzer* **34**, 32–43 (2011).
25. Biyani, A. et al. Extension of spike for encrypted protocol fuzzing. In *2011 Third International Conference on Multimedia Information Networking and Security*, 343–347 (IEEE, 2011).
26. Cui, W., Kannan, J. & Wang, H. J. Discoverer: Automatic protocol reverse engineering from network traces. In *USENIX Security Symposium*, 1–14 (Boston, MA, USA, 2007).
27. Comparetti, P. M., Wondracek, G., Kruegel, C. & Kirda, E. Prospex: Protocol specification extraction. In *2009 30th IEEE Symposium on Security and Privacy*, 110–125 (IEEE, 2009).
28. Wondracek, G., Comparetti, P. M., Kruegel, C., Kirda, E. & Anna, S. S. S. Automatic network protocol analysis. In *NDSS*, vol. 8, 1–14 (Citeseer, 2008).
29. Whalen, S., Bishop, M. & Crutchfield, J. P. Hidden markov models for automated protocol learning. In *Security and Privacy in Communication Networks: 6th Iternational ICST Conference, SecureComm 2010, Singapore, September 7-9, 2010. Proceedings 6*, 415–428 (Springer, 2010).
30. Godefroid, P., Peleg, H. & Singh, R. Learn&fuzz: Machine learning for input fuzzing. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 50–59 (IEEE, 2017).
31. Sweeney, L. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems* **10**, 557–570 (2002).
32. Machanavajjhala, A., Kifer, D., Gehrke, J. & Venkitasubramaniam, M. l-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data (tkdd)* **1**, 3–es (2007).
33. Riyana, S., Sasujit, K. & Homdoung, N. Achieving privacy preservation constraints based on k-anonymity in conjunction with adjacency matrix and weighted graphs. *ECTI Trans. Comput. Inf. Technol. (ECTI-CIT)* **18**, 34–50 (2024).
34. Riyana, S., Nanthachumphu, S. & Riyana, N. Achieving privacy preservation constraints in missing-value datasets. *SN Comput. Sci.* **1**, 227 (2020).
35. Riyana, S. Achieving anatomization constraints in dynamic datasets. *ECTI Trans. Comput. Inf. Technol. (ECTI-CIT)* **17**, 27–45 (2023).
36. Rasouli, M., Sun, T. & Rajagopal, R. Fedgan: Federated generative adversarial networks for distributed data. arXiv preprint arXiv:2006.07228 (2020).
37. Hardy, C., Le Merrer, E. & Sericola, B. Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. In *2019 IEEE international parallel and distributed processing symposium (IPDPS)*, 866–877 (IEEE, 2019).
38. Dong, Y., Liu, Y., Zhang, H., Chen, S. & Qiao, Y. Fd-gan: Generative adversarial networks with fusion-discriminator for single image dehazing. *Proc. AAAI Conf. Artif. Intell.* **34**, 10729–10736 (2020).
39. Bhardwaj, T. & Sumangali, K. A federated incremental blockchain framework with privacy preserving xai optimization for securing healthcare data. *Sci. Rep.* **15**, 38001 (2025).
40. Dwork, C. Differential privacy. In *International colloquium on automata, languages, and programming*, 1–12 (Springer, 2006).
41. Riyana, S., Sasujit, K. & Homdoung, N. Privacy-enhancing data aggregation for big data analytics. *ECTI Trans. Comput. Inf. Technol. (ECTI-CIT)* **17**, 440–456 (2023).
42. Riyana, S. (lp_1,..., lp_n)-privacy: privacy preservation models for numerical quasi-identifiers and multiple sensitive attributes. *J. Ambient Intell. Humaniz. Comput.* **12**, 9713–9729 (2021).
43. Shamsinezhad, E., Banirostam, H., BaniRostam, T., Pedram, M. M. & Rahmani, A. M. Providing and evaluating a model for big data anonymization streams by using in-memory processing. *Knowl. Inf. Syst.* 1–34 (2025).
44. Shamsinezhad, E., Banirostam, T., Pedram, M. M. & Rahmani, A. M. Anonymizing big data streams using in-memory processing: A novel model based on one-time clustering. *J. Signal Process. Syst.* **96**, 333–356 (2024).

Author contributions

Z.W. and L.Z. designed research plans, methodology and made writing-original draft preparation. Z.W., L.Z., and F.Me. put forward research ideas, conceptualization, and made writing-reviewing and editing. Z.Z. and Y.L. contributed significantly to analysis and writing-reviewing and editing. All authors reviewed the manuscript.

Funding

This research was partially funded by the GEF project—Strengthening coordinated approaches to reduce invasive alien species (IAS) threats to globally significant agrobiodiversity and agroecosystems in China, with project number 9874. This paper has also received funding from the Excellent Talent Training Funding Project in Dongcheng District, Beijing, with project number 2024-dchrcpyzz-9.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Y.L.