Proceedings of CONF-SEML 2026 Symposium: Computational Analysis and Modeling in Complex Intelligent Systems
DOI: 10.54254/2755-2721/2026.GU34026

Secure Data Aggregation for Digital Cousin Systems with Homomorphic Encryption and Integrity Verification

Jiasheng Hu

School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China
18870019720@163.com

Abstract. We're seeing more and more heterogeneous IoT devices being deployed acrossdifferent networks these days, and they're generating massive amounts of distributed data,this creates new challenges for secure multi-source aggregation in edge intelligenceenvironments. Most conventional approaches built for Digital Twin (DT) add way too muchmodeling complexity. They may not keep up when IoT environments shift quickly andunpredictably. Also, many existing homomorphic encryption solutions focus at privacyprotection, but they don't handle integrity verification during data aggregation processproperly. So to fix these issues, we developed a secure data aggregation framework for DigitalCousin (DC) systems. We built it using Paillier homomorphic encryption to keep data private,added RSA-based signatures so we can verify each data piece's authenticity, and a ChineseRemainder Theorem (CRT)-driven aggregation mechanism makes the actual aggregationprocess work smoothly. In this framework, edge gateways take charge of aggregatingencrypted data along with their associated signatures. Original plaintext stays completelyinaccessible throughout aggregation stage, no part of it can be accessed while the aggregationis still going on. Once the aggregated ciphertext is delivered to the DC side, signaturevalidation and plaintext reconstruction are carried out to guarantee both data authenticity andaggregation correctness. Compared with conventional aggregation schemes, the proposedframework decreases the processing pressure on edge gateways and reduces communicationcosts in large-scale IoT data collection tasks. The security evaluation further demonstratesthat the scheme can simultaneously support data confidentiality, integrity protection, andreliable source authentication in multi-device aggregation environments. Security analysisshows that the framework can effectively achieve confidentiality, integrity verification, andsource authentication for multi-device data aggregation scenarios.

Keywords: Digital cousin, homomorphic encryption, data aggregation, integrityverification

1. Introduction

With the rapid development of IoT, edge computing, and intelligent manufacturing technologies, a large amount of heterogeneous sensing data is continuously generated by distributed devices. These data provide important support for predictive maintenance, task scheduling, and resource management. However, because IoT devices are usually deployed in distributed environments with strict delay requirements, efficient processing of massive data remains a difficult problem. Data

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.GU34026

aggregation is used really widely these days. What it does is take all raw sensing information, compress and integrate it into more concise, useful results, it may also cut down on unnecessary data transmission and makes processing work run much faster.

These years, DT—important tech for cyber–physical integration—has gotten a lot of attention from researchers [1]. says it can set up real-time interaction between physical devices and virtual models, which brings benefits for system monitoring, data fusion, and intelligent decision-making. But current DT frameworks have some practical limits, keeping their virtual models accurate requires physical devices to sync and update nonstop, which creates a lot of extra work for modeling and communication. In highly dynamic IoT scenarios, this strict synchronization mechanism also makes system less flexible, and it can't scale well when you add more devices. To ease these problems, DC has been introduced as lightweight alternative to DT. Instead of building a fully synchronized replica for each physical entity, DC focuses at capturing the main behaviors and functional characteristics of devices. This way, we can cut down dependence on high-frequency synchronization while maintaining acceptable service performance in heterogeneous IoT environments [2]. also shows that DC-based collaborative architectures may offer better flexibility in heterogeneous edge scenarios.

When we talk about real-world IoT applications, security and privacy aren't just afterthoughts, they're as big a concern as efficiency issues. Take smart factories or data centers, for example, devices there generate data that holds sensitive operational details. If this data gets exposed directly during transmission or aggregation, privacy leakage can happen. But a lot of current aggregation methods still require intermediate gateways to process plaintext data, this raises security risks, plus it adds extra communication and computation overhead. As more and more devices come online, gateways need to decrypt and verify uploaded data over and over, which makes their processing burden way heavier than before. Homomorphic encryption supports operations over ciphertexts, but most existing schemes focus at confidentiality protection mostly, and offer relatively limited support for aggregation integrity verification. Also, the centralized architectures discussed in [3-5] rely heavily on server-side processing, in large-scale IoT systems, this can easily cause bandwidth bottlenecks and single-point failures. So right now, figuring out how to keep privacy intact, making sure that aggregation results are always accurate, and can we do all this without sacrificing processing efficiency seems to be really challenging problem.

Most old ciphertext aggregation schemes for DC systems force gateways to decrypt uploaded data repeatedly, that puts a huge computational strain on them especially in large-scale IoT setups. So we came up with secure ciphertext aggregation scheme that combines Paillier homomorphic encryption with RSA-based signatures to fix these issues. Also, we use RSA-based signatures. They check data integrity and verify data sources, this stops malicious actors from creating fake aggregation results. First, each IoT device encrypts its local sensing data before sending it out, so gateways can aggregate all incoming ciphertexts directly without ever accessing plaintext information. Once aggregation finishes, the ciphertext results are sent to central server for final decryption and plaintext recovery. By shifting most decryption tasks to the central server instead of making gateways handle them, we seem to effectively reduce the computational pressure on edge gateways, may work even better as the IoT network scales up. We ran experiments and did security analysis, it shows our method achieves secure and efficient ciphertext aggregation while keeping data confidential and ensuring aggregation correctness.

DOI: 10.54254/2755-2721/2026.GU34026

2. Preliminaries

2.1. Digital cousins

DCs are mainly introduced to provide virtual training environments for mitigating the sim-to-real gap in robotic policy learning. Conventional DT frameworks generally pursue highly precise virtual replicas of physical entities, where detailed physical properties and environmental dynamics must be continuously maintained. Although such methods can achieve high modeling accuracy, they usually require considerable modeling effort, large amounts of data collection, and high deployment costs. Compared with DT, DC adopts a more lightweight modeling idea. Instead of strictly reconstructing every detail of the real environment, DC pays more attention to maintaining functional similarity between the physical scene and the generated virtual environment. During the construction process, task-related spatial structures and semantic interaction information are preserved, while redundant visual details are simplified as much as possible. Therefore, DC can generate multiple functionally comparable virtual environments with lower construction overhead and improved scalability. At present, most DC-related studies focus on automated virtual scene generation, among which ACDC (Automated Creation of DCs) is a representative framework proposed in [1]. When you capture a single RGB image from real scene, this framework can generate matching virtual training environments automatically, no need for manual annotation. Traditional reconstruction approaches mostly focus at texture recovery, but ACDC prioritizes information closely tied to robotic tasks: object distribution, spatial relationships, and what interactions robots can have with the space. The generation process has three core stages: real-scene feature extraction, DC asset matching, and virtual scene construction. I'll break each down to make it clear. We feed the input image into multiple models, they sift through every pixel to carry out real-scene feature extraction, pulling out key environmental features that robots need to complete their tasks. Next, we use a hierarchical retrieval strategy to handle DC asset matching, picking out virtual assets that fit the scene's layout and requirements perfectly. Finally, we move on to virtual scene construction, placing these selected assets in the simulation environment, adjusting their positions and properties to follow all physical constraints so the scene behaves just like the real one. GPU acceleration speeds up every part of this work, the framework can build dozens of diverse virtual scenes in a fraction of the time it would take without it. This means we can expose robotic policies to way more varied scenarios, so they may perform better when faced with unexpected real-world situations, helping boost their generalization capability.

2.2. Homomorphic encryption

Homomorphic encryption enables direct computations on encrypted data, with decrypted results matching those of plaintext operations. This property protects privacy in DTs and data aggregation by processing data without decryption, preventing leaks at intermediate nodes. It includes additive (e.g., Paillier) and multiplicative (e.g., RSA) types.

The Paillier cryptosystem works as follows:

Choose two large primes p and q, compute n=pq and λ=lcm(p−1,q−1). Let h=n+1. Define the function $L(x)=(x-1)/n$. Then compute

$$\mu = \left(\left(\frac{-1}{n} h^{\lambda} \bmod n^2 \right) \right)^{-1}. \tag{1}$$

DOI: 10.54254/2755-2721/2026.GU34026

Encryption

$$c = h^m r^n mod\, n^2, where\, r \in Z_{\mathrm{N}}. \tag{2}$$

Decryption

$$m = \left(\frac{-1}{n}\, c^{\lambda} \mathrm{mod}\ N^2\right) \cdot \mu\ mod\ n. \tag{3}$$

Additive homomorphism

$$D\left(E\left(m_1\right) \cdot E\left(m_2\right)\right) = m_1 + m_2. \tag{4}$$

RSA signatures are defined with n=pq,

$$\phi\left(\mathrm{n}\right) = \left(\mathrm{p}-1\right)\left(\mathrm{q}-1\right), \tag{5}$$

a public exponent e with

$$gcd\ \left(\mathrm{e}, \phi\left(\mathrm{n}\right)\right) = 1, \tag{6}$$

a nd private exponent

$$d = \mathrm{e}^{-1}\ mod\ \phi\left(\mathrm{n}\right). \tag{7}$$

Signature

$$\sigma = \mathrm{m}^d\ mod\ n. \tag{8}$$

Verification

$$m = \sigma^e\ mod\ n. \tag{9}$$

Multiplicative homomorphism

$$m_1 m_2 = \left(\sigma_1 \sigma_2\right)^e\ mod\ n. \tag{10}$$

DOI: 10.54254/2755-2721/2026.GU34026

However, single schemes have limitations: Paillier offers confidentiality without verification; RSA signatures need key sharing and are incompatible with Paillier. To overcome these, a verifiable homomorphic encryption scheme based on the CRT is proposed. A key center assigns keys to devices, gateways, and virtual spaces.

Devices encrypt data with Paillier:

$$c_i = h^{m_i} r_i^N \ mod\ N^2, \tag{11}$$

then sign ciphertext with CRT-based RSA:

$$\tau_i = c_i^{sk_{SM_i}^{A_i}} \ mod\ N. \tag{12}$$

The gateway aggregates ciphertexts:

$$\pi = \prod c_i \ mod\ N, \tag{13}$$

and signatures

$$\Omega = \left(c_i^{sk_{gw}^{B_i}} \right) \left(\prod \tau_i \right) \ mod\ N. \tag{14}$$

The DC verifies integrity by checking

$$\pi = \Omega^{pk} \ mod\ N, \tag{15}$$

and recovers aggregated plaintext

$$M = \sum m_i = \left(\frac{-1}{n} \pi^{\lambda} \bmod N^2 \right) \cdot \mu \ mod\ N, \tag{16}$$

ensuring both confidentiality and integrity.

The core value of homomorphic encryption is "data usable but invisible." Practical deployment must balance efficiency and security by reducing computation/communication overhead through aggregation, resisting forgery with hash functions and round identifiers, and optimizing modular exponentiation for real-time needs.

DOI: 10.54254/2755-2721/2026.GU34026

2.3. Chinese Remainder Theorem (CRT)

The CRT is a fundamental result in number theory. It states that for a set of pairwise coprime positive integers n_1 , n_2 ,…, n_k , the system of congruences.

$$\begin{cases} x \equiv a_1 \,(\mathrm{mod}\, n_1) \\ x \equiv a_2 \,(\mathrm{mod}\, n_2) \\ \quad\quad\vdots \\ x \equiv a_k (mod\, n_k) \end{cases}, \tag{17}$$

has a unique solution modulo.

$$n = n_1 n_2 \cdots n_k. \tag{18}$$

Construction of the solution:
Let

$$N = \prod_{i=1}^{k} n_i, \tag{19}$$

and define

$$N_i = \frac{N}{n_i}, \tag{20}$$

for each i . Since

$$\gcd(N_i, n_i) = 1, \tag{21}$$

there exists a multiplicative inverse M_i such that

$$N_i \cdot M_i \equiv 1 \,(\mathrm{mod}\, n_i). \tag{22}$$

Then the unique solution modulo N is given by

$$x \equiv \sum_{i=1}^{k} a_i \cdot N_i \cdot M_i \,(\mathrm{mod}\, n). \tag{23}$$

Application in cryptography:

DOI: 10.54254/2755-2721/2026.GU34026

CRT is widely used to accelerate RSA decryption and signing by decomposing large-modulus computations into smaller moduli. For example, with RSA modulus N=pq, a ciphertext cc can be decrypted as:

$$m_p = c^{d \bmod (p-1)} \ (\bmod p), \tag{24}$$

$$m_q = c^{d \bmod (q-1)} \ (\bmod q), \tag{25}$$

and then combined via CRT to obtain m(mod N). This reduces the computational complexity by about a factor of four compared to direct exponentiation.

In this paper, CRT also enables the aggregation of signatures generated under distinct private keys. Specifically, the Key Distribution Center (KDC) constructs a system-wide private key sk_s that simultaneously satisfies multiple congruences:

$$sk_s \equiv sk_{gw} B_i + sk_{\mathrm{SM}i} A_i \ (\mathrm{mod} \phi(N)), \forall i. \tag{26}$$

Where A_i , B_i are parameters derived from CRT. This allows the edge gateway to combine individual device signatures into a single aggregated signature, which can be verified using the single public key pk_s .

3. System model

We assign different functions to different entities so multi-source data can be aggregated securely, this can spread computational tasks around and cut down processing burden that individual nodes have to handle, to some extent. I designed this framework for DC-based IoT environment, where hundreds of physical devices placed in various locations continuously produce sensing information without pausing. Fig. 1 lays out every part of the proposed scheme's overall architecture, Table 1 lists all major notations I use from start to finish of this paper.

Physical devices first gather local sensing data from their surroundings, then they complete encryption and generate a proper signature before transmitting that data to external networks. This keeps sensitive information protected before it leaves device side. The KDC focuses at two main responsibilities, it handles system setup and key distribution, plus it initializes public parameters and provides cryptographic keys for authorized participants. Multiple devices upload encrypted data, edge gateways collect it and run aggregation operations directly over ciphertexts. They don't hold corresponding decryption keys, so during the aggregation process, they can't access original plaintext content. This setup may cut down privacy leakage risks at intermediate nodes. Also, when the number of connected devices becomes really large, it can avoid frequent decryption operations that would otherwise slow down the system's performance. DC entities have two key roles: they request data, they also receive the final aggregation results. Once they pull aggregated ciphertexts from gateway, they run signature verification first, then do result decryption to get plaintext information they need. This entire process may let them spot any forged or modified aggregation

DOI: 10.54254/2755-2721/2026.GU34026

data before they fully accept the final results, so they don't end up using inaccurate or unreliable data for their work.

During every step of aggregation process, plaintext information never gets exposed directly to intermediate entities, this improves data confidentiality. The proposed framework has four main procedures to make this work. First we can do system initialization, next is data encryption with signature generation, then we carry out ciphertext aggregation, and finally we handle result decryption together with integrity verification. It may also ensure aggregation results stay correct and reliable.

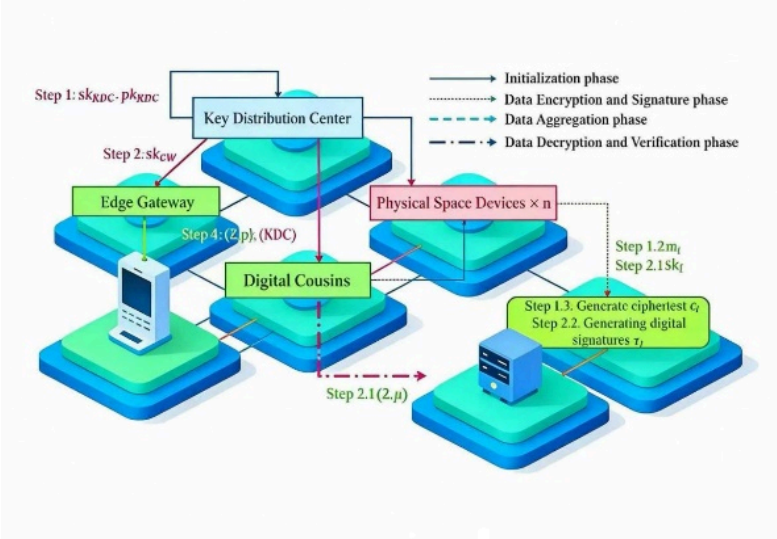

Figure 1. System model

Table 1. Common notations used

Notation	Description
q_1, q_2, N	Random prime numbers and their product
Φ(N)	Euler function
$a_1 < a_2 < \ldots < a_l$	Randomly generated relatively prime numbers
sk_s, pk_s	Public-private key pair for integrity protection
sk_{KDC}	The private key of the KDC
pk_{KDC}	The public key of the KDC
sk_{GW}	The private key of the gateway
sk_{gw}	The aggregation key of the gateway
sk_i	The private key of the i th PS device
L()	Function for Paillier cryptographic systems
(λ, μ)	The private key of the DC entity
(N, h)	The public key of the DC entity
sk_{DC}, pk_{DC}	The public key and private key of the DC
m_i	The plaintext produced by the i th PS device
c_i	The ciphertext calculated by the i th PS device

DOI: 10.54254/2755-2721/2026.GU34026

Table 1. (continued)

η	The set of ciphertexts calculated by PS devices
r_i	Randomly generated number
τ_i	Digital signature of the i th PS device
τ	The set of digital signature
π	The aggregated ciphertext
Ω	The aggregated signature of the π
M	The aggregated data extracted by the DC

3.1. System initialization phase

The system initialization phase is led by the KDC, which serves as the core trust anchor to build the global security foundation. First, the KDC generates global public cryptographic parameters such as elliptic curves and large prime numbers, and creates exclusive key pairs for physical space devices, edge gateways, and virtual spaces respectively.

Step 1: Private keys (e.g., sk_{KDC}, sk_{GW}, sk_I) are distributed to corresponding entities through secure channels, while the public keys of each entity are published to a globally accessible directory.

$\{ sk_{KDC}, pk_{KDC} \} \leftarrow$ Init(). This algorithm is executed by the KDC for generating its own public and private keys. The private key is sk_{KDC} and the public key is pk_{KDC}.

$\{ sk_{GW} \} \leftarrow$ gateAggK(sk_{KDC}, pk_{KDC}). This algorithm is executed by the KDC for generating the aggregating key sk_{GW} for the edge gateway. It takes as inputs its own private key (i.e., sk_{KDC}) and the corresponding public key (i.e., pk_{KDC}), and outputs the private key of the edge gateway (i.e., sk_{GW}).

Step 2: The KDC configures parameters for core security components including hash functions, homomorphic encryption algorithms, and signature algorithms, unifying the cryptographic specifications of all participating entities.

$\{ sk_i \} \leftarrow$ genPsK(sk_{KDC}, pk_{KDC}). This algorithm is executed by the KDC for generating the signing and encryption key for the ith PS device. It takes as inputs its own private key (i.e., sk_{KDC}) and the corresponding public key (i.e., pk_{KDC}), and outputs the private key of the i th PS device (i.e., sk_i).

$\{ sk_{DC}, pk_{DC} \} \leftarrow$ genDCK(sk_{KDC}, pk_{KDC}). This algorithm is executed by the KDC for generating public and private keys for the DC. It takes as inputs its own private key (i.e., sk_{KDC}) and the corresponding public key (i.e., pk_{KDC}), and outputs the private key of the DC (i.e., sk_{DC}) and the corresponding public key of the DC (i.e., pk_{DC}).

This ensures that subsequent data encryption, signature, aggregation, and verification operations can be executed under consistent security assumptions. After this phase, all entities obtain legitimate identity credentials and secure computing capabilities, laying a trust foundation for the entire data flow closed loop.

3.2. Data encryption and signature phase

The data encryption and signature phase is executed in parallel by each physical space device locally to achieve source-binding of data security attributes.

DOI: 10.54254/2755-2721/2026.GU34026

Step 1: The device uses the local private key sk_I obtained during the initialization phase to generate an individual digital signature τ_i for the collected raw data m_t

$\{ c_i \} \leftarrow$ Encrypt(m_i , pk_{DC} , pk_{KDC}). This algorithm is executed by the i th PS device for encrypting its data. It takes as inputs the public key of the DC (i.e., pk_{DC}), the public key of the KDC (i.e., pk_{KDC}) and the plaintext m_i , and outputs the ciphertext of m_i (i.e., c_i).

Step 2: The signature proves the generation source and integrity of the data, preventing data tampering or forgery. Then, the device encrypts the raw data m_t using the public key pk of the virtual space to generate ciphertext c_i , ensuring that the data can only be decrypted by the virtual space during transmission and aggregation.

$\{\tau_i\} \leftarrow$ Sign(c_i , sk_i , pk_{KDC}). This algorithm is executed by the ith PS device for signing the encrypted data. It takes as inputs the encrypted data (i.e., c_i), the private key of the i th PS device (i.e., sk_i) and the public key of the KDC (i.e., pk_{KDC}), and outputs a signature for c_i (i.e., τ_i).

3.3. Data aggregation phase

The data aggregation phase takes the edge gateway as the core hub to achieve efficient compression of multi-source data within the ciphertext domain. First, the gateway collects ciphertexts c_i and individual signatures τ_i from multiple physical space devices.

Based on the algebraic characteristics of homomorphic encryption algorithms, it aggregates multiple groups of independent ciphertexts into a single aggregated ciphertext (Σ, p) without decrypting any ciphertext, achieving significant reduction in data volume. Meanwhile, the gateway compresses multiple individual signatures τ_i into one aggregated signature (Σ, μ) using aggregate signature technology, retaining the integrity verification capability of all signatures.

The entire aggregation process is completed entirely in the ciphertext domain, and the edge gateway cannot access plaintext or private keys, which not only ensures data confidentiality but also significantly reduces subsequent computing and load overhead of the DCs.

$\{\pi\} \leftarrow$ Aggci(η). This algorithm is executed by the edge gateway for aggregating multiple ciphertexts into one. It takes as input the set of ciphertexts (i.e., η), and outputs one aggregated ciphertext (i.e., π).

$\{\Omega\} \leftarrow$ Aggsign(η, sk_{gw} ,τ). This algorithm is executed by the edge gateway for aggregating multiple signatures into one. It takes as inputs the set of signatures (i.e., τ), the edge gateway's private key (i.e., sk_{gw}) and the set of ciphertexts (i.e., η), and outputs one aggregated signature (i.e., Ω).

3.4. Data decryption and verification phase

The data decryption and verification phase completes the final data consumption and security verification in the DC.

Step 1: The DC obtains the aggregated ciphertext (Σ, p) and aggregated signature (Σ, μ) transmitted by the edge gateway, and performs batch verification on the aggregated signature using the public keys of all participating devices.

{True, False} $\leftarrow$ Auth(π, pk_{KDC} , Ω). This algorithm is executed by the DC for checking the integrity of the received data. It takes as inputs the aggregated ciphertext (i.e., π), the KDC's public key (i.e., pk_{KDC}) and the aggregated signature (i.e., Ω), and outputs True if the received data is not tampered by the adversary. Otherwise, it outputs False.

DOI: 10.54254/2755-2721/2026.GU34026

This step confirms that all data sources are legitimate physical devices and the aggregation process has not been tampered with, ensuring data integrity and source credibility. After successful verification, the DC decrypts the aggregated ciphertext (Σ, p) using its own private key to restore the final aggregated plaintext data.

Step 2: The decrypted plaintext can be directly used for status monitoring, trend analysis, and decision-making deduction of the DC, completing a closed loop from data collection and secure aggregation to business application.

{M} ← Decry(pk_{KDC} , π, sk_{DC} , pk_{DC}). This algorithm is executed by the DC for decrypting the plaintext from the aggregated ciphertext. It takes as inputs the public key of the KDC (i.e., pk_{KDC}), the aggregated ciphertext (i.e., π), the private key of the DC (i.e., sk_{DC}) and the public key of the DC (i.e., pk_{DC}), and outputs the plaintext (i.e., M).

4. Secure data sharing system construction

This section presents the construction of the secure data sharing system for DCs. The proposed framework integrates multi-participant data sources, homomorphic encryption, homomorphic signatures, and the CRT, orchestrated under a KDC. The system ensures end-to-end security, covering data encryption, digital signature generation, ciphertext/signature aggregation, integrity verification, and final decryption. The construction consists of ten probabilistic polynomial-time algorithms, involving participants including the KDC, t data owners, one aggregation node (edge gateway), and one DC.

1. System Initialization: { sk_{KDC} , pk_{KDC} } ← Init(). The KDC randomly generates two prime numbers q_1 and q_2, computes $N = q_1 q_2$, and calculates

$$\lambda = \mathrm{lcm}\,(\mathrm{q}_1 - 1, \cdot \mathrm{q}_2 - 1), \tag{27}$$

via the extended Euclidean algorithm. Euler's totient function is evaluated as

$$\phi\,(N) = (\mathrm{q}_1 - 1)\,(\mathrm{q}_2 - 1). \tag{28}$$

Next, the KDC generates l pairwise coprime numbers $a_1 < a_2 < \cdots < a_l$, computes

$$A_{\mathrm{m}} = \prod_{i=1}^{l} a_{\mathrm{i}}, \tag{29}$$

and

$$A = \prod_{i=1}^{l} a_{\mathrm{i}}, \tag{30}$$

randomly selects $sk_s \in (\mathrm{a_l}, \mathrm{A_m})$, and solves

$$pk_s = sk_s^{-1}\ mod\ \Phi\,(\mathrm{N}). \tag{31}$$

DOI: 10.54254/2755-2721/2026.GU34026

The resulting key sets are sk_{KDC} = {a_1, a_2, …, a_l, A_m, sk_s , q_1, q_2, λ} (private), pk_{KDC} = { pk_s , A, N} (public, published globally).

2. Aggregation Node Key Generation: { sk_{GW} } ← gateAggK(sk_{KDC} , pk_{KDC}).The KDC generates the aggregation private key for the edge gateway and computes

$$sk_{gw} = sk_s \ mod \ a_1. \tag{32}$$

For each i ($1 \leq i \leq$ t), it calculates

$$B_i = a_{i+1}\left({a_{i+1}}^{-1} \ mod \ a_1\right), \tag{33}$$

and outputs sk_{GW} = { sk_{gw} , B_i | $1 \leq i \leq$ t}.

3. Data Owner Key Generation: sk_i ← genPSK(sk_{KDC} , pk_{KDC}). The KDC generates a unique private key for each data owner. For the i th data owner, it computes

$$sk_{SMi} = sk_s \ mod \ a_{i+1}. \tag{34}$$

And

$$A_{\mathrm{i}} = a_1 \left({a_1}^{-1} \bmod a_{i+1}\right), \tag{35}$$

then outputs sk_i = { sk_{SMi} , A_i }.

4. Data User Key Generation: { sk_{KDC} , pk_{KDC} } ← genDCK(sk_{KDC} , pk_{KDC}). The KDC generates homomorphic encryption/decryption keys for the DCs by setting h = N+1, defining the Paillier function

$$L\left(\mathrm{x}\right) = \frac{\mathrm{x}-1}{\mathrm{N}}, \tag{36}$$

and computing

$$\mu = \left(L\left(\mathrm{h}^{\lambda} \bmod \mathrm{N}^2\right)^{-1}\right) \ mod \ N. \tag{37}$$

The resulting keys are sk_{DC} = {μ, λ}, pk_{DC} = {N, h}.

5. Data Encryption: { c_i } ← Encrypt(m_i , pk_{DC} , pk_{KDC}). Each PS device randomly generates $r_i \in Z_N$ and computes the ciphertext

DOI: 10.54254/2755-2721/2026.GU34026

$$c_i = \mathrm{h}^{m_i}\, {r_i}^{\mathrm{N}}\ mod\ N^2, \tag{38}$$

using Paillier encryption, then transmits c_i to the aggregation node along with the signature.

6. Ciphertext Signing: { τ_i } ← Sign(c_i , sk_i , pk_{KDC}). Each PS device generates a digital signature by extracting sk_{SMi} and A_i from sk_i , computing τ_i = (c_i)^{ sk_{SMi} A_i } mod N, and transmitting { c_i , τ_i } to the aggregation node.

7. Ciphertext Aggregation: {π} ← Aggci(η). The aggregation node collects ciphertexts η = { c_i | 1 ≤ i ≤ t} and computes the aggregated ciphertext

$$\pi = \prod_{i=1}^{t}\ c_i\ mod\ N^2, \tag{39}$$

using Paillier additive homomorphism.

8. Signature Aggregation: {Ω} ← Aggsign(η, sk_{gw} , τ). The aggregation node collects signatures τ = { τ_i | 1 ≤ i ≤ t}, extracts sk_{gw} and B_i from sk_{GW} , and computes

$$\Omega = \prod_{i=1}^{t} {c_i}^{\,sk_{gw}B_i} \cdot \prod_{i=1}^{t} \tau_i\ mod\ N, \tag{40}$$

then transmits {π, Ω} to the data user.

9. Integrity Verification: {True, False} ← Auth(π, pk_{KDC} , Ω). The data user verifies the integrity of the received ciphertext. By checking whether or not

$$\pi = \ \Omega^{\,pk_s}\ mod\ N, \tag{41}$$

the system outputs True if untampered or False otherwise (discarding invalid data).

10. Aggregated Ciphertext Decryption: .

$$\{M\} \leftarrow Decry\left(pk_{\mathrm{KDC}}, \pi, sk_{\mathrm{DC}}, pk_{\mathrm{DC}}\right). \tag{42}$$

The data user computes

$$M = \sum_{i=1}^{t} m_i = \frac{L\left(\pi^{\lambda} \bmod N^2\right)}{L\left(h^{\lambda} \bmod N^2\right)}\ modN = L\left(\pi^{\lambda} \bmod N^2\right) \cdot \mu\ mod\ N, \tag{43}$$

using Paillier decryption, recovering the final aggregated plaintext.

Derivation of the decryption formula:

Since

DOI: 10.54254/2755-2721/2026.GU34026

$$\pi = \prod_{i=1}^{t} c_i \ mod \ N^2, \tag{44}$$

and

$$c_i = h^{m_i} r_i^N \ mod \ N^2, \tag{45}$$

we have

$$\pi = h^{\sum m_i} \cdot \left(\prod r_i\right)^N \ mod \ N^2. \tag{46}$$

Raise both sides to the power λ:

$$\pi^\lambda = h^{\lambda \sum m_i} \cdot \left(\prod r_i\right)^{N\lambda} \ mod \ N^2, \tag{47}$$

by Carmichael's theorem, for any ri coprime to N,

$$\left(r_i\right)^{N\lambda} \equiv 1 \ \left(\mathrm{mod}\ N^2\right), \tag{48}$$

also, h=N+1 gives

$$h^{\lambda \sum m_i} = \left(1+N\right)^{\lambda \sum m_i} \equiv 1 + N\lambda \sum m_i \left(mod\ N^2\right). \tag{49}$$

Thus,

$$\pi^\lambda \equiv 1 + N\lambda \sum m_i \left(\mathrm{mod}\ N^2\right). \tag{50}$$

Applying

$$L\left(x\right) = \tfrac{x-1}{N}, \tag{51}$$

$$L\left(\pi^\lambda \mathrm{mod}\ N^2\right) = \lambda \sum m_i \ mod \ N. \tag{52}$$

Finally, because μ= λ^{-1} modN, we obtain

DOI: 10.54254/2755-2721/2026.GU34026

$$M = L\left(\pi^{\lambda}\mathrm{mod}\ N^2\right)\cdot\mu = \lambda\sum m_i\cdot\lambda^{-1} = \sum m_i\ mod\ N. \tag{53}$$

Notably, all participating entities—including PS devices, the edge gateway, and the DCs—perform only a limited number of modular exponentiation operations, substantially reducing computational overhead while maintaining end-to-end security.

5. Security analysis

This section presents a unified security analysis of the proposed scheme, jointly considering the correctness of aggregated data recovery and the validity of integrity verification. Specifically, we demonstrate that the virtual space (DC) can accurately recover the aggregated plaintext from the aggregated ciphertext, while ensuring that the integrity of the aggregated data can be verified through the designed verification equation.

Correctness of aggregated plaintext recovery.

In the Decry algorithm, the DC extracts the aggregated plaintext M from the aggregated ciphertext π as

$$M = \frac{L\left(\pi^{\lambda}\mathrm{mod}N^2\right)}{L\left(h^{\lambda}\mathrm{mod}N^2\right)}\ mod\ N = L\left(\pi^{\lambda}\mathrm{mod}N^2\right)\times\mu\ mod\ N. \tag{54}$$

We now prove that

$$M = \sum_{i=1}^{t} m_i. \tag{55}$$

Since h=N+1, the binomial theorem implies

$$h^{\lambda} = \left(1+N\right)^{\lambda} \equiv 1+\lambda N\ \left(mod\ N^2\right). \tag{56}$$

Hence,

$$L\left(h^{\lambda}\mathrm{mod}N^2\right) = \frac{(1+\lambda N)-1}{N}\ = \lambda\ mod\ N, \tag{57}$$

and consequently μ= λ^{-1} mod N.

The aggregated ciphertext from the Aggci algorithm is

$$\pi = \prod_{i=1}^{t} c_i\ mod\ N^2. \tag{58}$$

From the Encrypt algorithm,

DOI: 10.54254/2755-2721/2026.GU34026

$$c_i = h^{m_i} r_i^N \ mod\ N^2, \tag{59}$$

so

$$\pi = h^{\sum m_i} \cdot \left(\prod r_i\right)^N \ mod\ N^2, \tag{60}$$

raising both sides to the power λ:

$$\pi^\lambda = h^{\lambda \sum m_i} \cdot \left(\prod r_i\right)^{N\lambda} \ mod N^{\ 2}, \tag{61}$$

by Carmichael's theorem, for each r_i coprime to N

$$r_i^{N\lambda} \equiv 1 \left(\mathrm{mod}\ N^2\right). \tag{62}$$

Thus

$$\left(\prod r_i\right)^{N\lambda} \equiv 1 \left(\mathrm{mod}\ N^2\right). \tag{63}$$

Meanwhile,

$$h^{\lambda \sum m_i} = \left(1+N\right)^{\lambda \sum m_i} \equiv 1 + N\lambda \sum m_i \ \left(\mathrm{mod}\ N^2\right). \tag{64}$$

Therefore,

$$\pi^\lambda \equiv 1 + N\lambda \sum m_i \left(\mathrm{mod}\ N^2\right). \tag{65}$$

Applying the L function:

$$L\left(\pi^\lambda \mathrm{mod} N^2\right) = \frac{(1+N\lambda \sum m_i)-1}{N} \ = \lambda \sum m_i \ mod\ N. \tag{66}$$

Finally,

$$M = (\lambda \sum m_i) \cdot \mu = (\lambda \sum m_i) \cdot \lambda^{-1} = \sum m_i \ mod\ N, \tag{67}$$

DOI: 10.54254/2755-2721/2026.GU34026

which confirms that the DC can correctly recover the aggregated plaintext.

Meanwhile, the integrity of the aggregated data is ensured through the homomorphic signature aggregation mechanism. In the Auth algorithm, the DC verifies correctness via

$$\pi = \Omega^{pk_s} \ mod \ N, \tag{68}$$

where Ω denotes the aggregated signature and (sk_s , pk_s ,) is the system-wide RSA key pair generated by the KDC.

According to the Sign algorithm, each PS device generates

$$\tau_i = (c_i)^{sk_{SMi}A_i} \ mod \ N. \tag{69}$$

During aggregation, the edge gateway computes

$$\Omega = \prod_{i=1}^{t} \left(c_i\right)^{sk_{gw}B_i} \cdot \prod_{i=1}^{t} \tau_i = \prod_{i=1}^{t} \left(c_i\right)^{sk_{gw}B_i + sk_{\mathrm{SM}i}A_i} mod \ N. \tag{70}$$

The KDC constructs sk_s such that for all i,

$$sk_s \equiv sk_{gw}B_i + sk_{\mathrm{SM}i}A_i \ (\mathrm{mod}\phi\,(N)), \tag{71}$$

which exists and is unique modulo ϕ(N) by the CRT. Hence,

$$\Omega = \prod_{i=1}^{t} \left(c_i\right)^{sk_s} = \left(\prod_{i=1}^{t} c_i\right)^{sk_s} = \pi^{sk_s} mod \ N. \tag{72}$$

Then,

$$\Omega^{pk_s} = \left(\pi^{sk_s}\right)^{pk_s} = \pi^{sk_s \cdot k_s} \ mod \ N. \tag{73}$$

Since,

$$sk_s \cdot k_s \equiv 1 \ (\mathrm{mod}\phi\,(N)), \text{we have } \pi^{sk_s \cdot k_s} \equiv \pi \ (\mathrm{mod}\ N). \tag{74}$$

so the verification equation holds.

To further enhance robustness against replay attacks across multiple aggregation rounds, a round identifier ID_{round} can be incorporated into the signing process, as described in the main text.

DOI: 10.54254/2755-2721/2026.GU34026

6. Simulation experiment and analysis

We conduct experimental evaluation in Python with the deep learning library of PyTorch to simulate data sharing from physical devices. The experimental emulation platform is running on a Linux server with an NVIDIA A100 GPU (VRAM 16 G), an Intel Xeon processor and 128 G memory. Fairness is ensured by config- uring DT and DC environment according to parameter values reported in [6] and [7] and using the two synchronized datasets reported there for the evaluation.

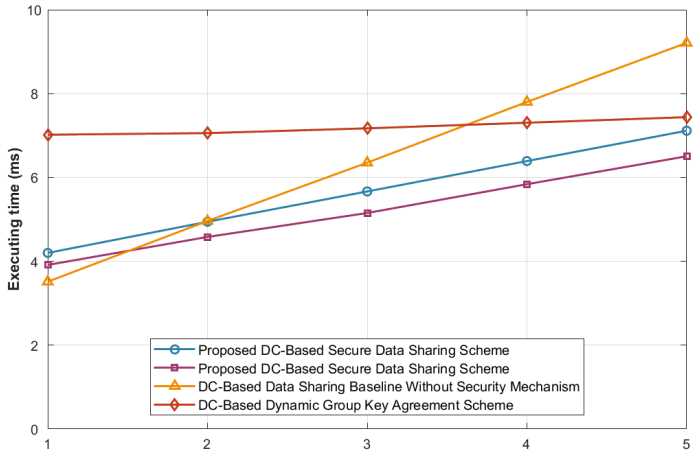

Figure 2. Number of physical devices participating in sharing

Fig. 2 gives the execution latencies of the four schemes as the number of participating devices increases from 1 to 5. In general, all schemes have longer execution latencies when more devices participate in the aggregation operation, but their increases differ. For the proposed DC-based secure sharing method, the total latency increases from roughly 4.2 ms to 6.8 ms. The average latency increment induced by each participating device is approximately 0.65 ms, which means the proposed method exhibits a fairly flat scalability over increasing workload. The non-secure baseline scheme has a slightly shorter latency with execution time of 3.9 ms–6.4 ms. But as the number of participating devices continues to grow, the performance gap between the two schemes becomes smaller. So the extra encryption and integrity verification procedures proposed in our framework lead to only slight increase of overhead in large-scale cases. For the DC-based dynamic group key agreement scheme proposed in [8], more cryptographic operations are involved in the sharing, so the execution latency becomes larger from 7.0 ms to 9.2 ms. In the whole experiment, its overall delay is clearly much higher than the proposed scheme. Another scheme reported in [8] keeps the execution time of about 2 ms as constant, because its own aggregation process does not include integrity verification operations. Such a scheme design enhances the efficiency; however, it provides no security assurance against forged or tampered aggregation result. The performance comparison between the above schemes is related to the aggregation mechanism of schemes. As to the proposed approach, multiple ciphertexts can be summed up directly using the homomorphic operation, no decryption and verification is needed at intermediary gateways. On the contrary, some of the existing typical methods process uploaded data separately for each device, the calculation overhead becomes quite large when the number of connected devices is huge. Overall, the experiment results show that the proposed scheme can achieve the balance of security and efficiency very well. The scheme not only can ensure confidentiality protection and integrity protection, but also can provide

DOI: 10.54254/2755-2721/2026.GU34026

relatively less execution overhead, which is applicable to large-scale DT/DC with increasing device numbers of IoT.

7. Conclusion

This paper studies the problem of secure data aggregation in DC-enabled large-scale IoT environments. To improve the security of data transmission and aggregation, a framework integrating Paillier homomorphic encryption, CRT-based RSA signatures, and edge-assisted aggregation is proposed. With the help of ciphertext aggregation, edge gateways are able to aggregate data uploaded from multiple devices directly in the encrypted domain without obtaining the original plaintext content. After receiving the aggregation result, DC entities then completely decrypt the result and verify the signature to ensure the correctness of aggregation result and aggregation result integrity. In comparison with other plaintext aggregation schemes, our approach decreases repeated decryption at intermediate gateways and communication burden for large-scale data collection. Experiment shows that our framework is able to maintain stable execution efficiency regardless of increasing number of devices participating in the aggregation. Meanwhile, our scheme still provides efficient concealment and integrity for the aggregated result with a reasonable computational overhead. Our proposed framework is applicable to real IoT applications with secure aggregation of data from multiple devices, such as intelligent manufacturing, distributed monitoring, dynamic industry data analysis, etc.

References

[1] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital twin in industry: State-of-the-art, " IEEE Transactions on industrial informatics, vol. 15, no. 4, pp. 2405–2415, 2018.
[2] Z. Zhijie, D. Rui, W. Yuhang, D. Yihe, Z. Fuhui, and W. Qihui, "A novel digital cousin based intelligent resource allocation scheme for secure low-altitude uav spectrum sharing in amn, " Chinese Journal ofAeronautics, p. 103983, 2025.
[3] P. Jesus, C. Baquero, and P. S. Almeida, "A survey of distributed data aggregation algorithms, " IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 381–404, 2014.
[4] R. Rajagopalan and P. K. Varshney, "Data aggregation techniques in sensor networks: A survey, " 2006.
[5] W. A. Clark and K. L. Avery, "The effects of data aggregation in statistical analysis, " Geographical Analysis, vol. 8, no. 4, pp. 428–438, 1976.
[6] T. Dai et al., "Automated Creation of Digital Cousins for Robust Policy Learning, " Oct. 2024. [Online]. Available: https: //arxiv.org/abs/2410.07408
[7] K. Wang et al., "Multi-Tier Distributed Computing Systems by Leveraging Digital Twin: Challenges, Techniques, and Research Directions, " IEEE Wireless Communications, vol. 32, no. 5, pp. 236–244, Oct.2025.
[8] Y. Tang and K. Wang, "Fppfl: Fedavg-based privacy-preserving federated learning, " in Proceedings of the 2023 15th International Conference on Computer Modeling and Simulation, ser. ICCMS '23. Dalian, China: Association for Computing Machinery, Oct. 2023, p. 51–56.