electronics

MDPI

Article

A Differentially Private Framework for the Dynamic Heterogeneous Redundant Architecture System in Cyberspace

Yilin Kang [1,†], Qiao Zhang [1,†], Bingbing Jiang [1,*] and Youjun Bu [2]

[1] Purple Mountain Laboratories, Nanjing 211111, China; kangyilin@pmlabs.com.cn (Y.K.); zhangqiao@pmlabs.com.cn (Q.Z.)
[2] Information Technology Institute, Information Engineering University, Zhengzhou 450002, China; buyoujun@pmlabs.com.cn
* Correspondence: njubing.jiang@gmail.com
† These authors contributed equally to this work.

Abstract: With the development of information technology, tremendous vulnerabilities and backdoors have evolved, causing inevitable and severe security problems in cyberspace. To fix them, the endogenous safety and security (ESS) theory and one of its practices, the Dynamic Heterogeneous Redundant (DHR) architecture, are proposed. In the DHR architecture, as an instance of the multi-heterogeneous system, a decision module is designed to obtain intermediate results from heterogeneous equivalent functional executors. However, privacy-preserving is not paid attention to in the architecture, which may cause privacy breaches without compromising the ESS theory. In this paper, based on differential privacy (DP), a theoretically rigorous privacy tool, we propose a privacy-preserving DHR framework called DP-DHR. Gaussian random noise is injected into each (online) executor output in DP-DHR to guarantee DP, but it also makes the decision module unable to choose the final result because each executor output is potentially correct even if it is compromised by adversaries. To weaken this disadvantage, we propose the advanced decision strategy and the hypersphere clustering algorithm to classify the perturbed intermediate results into two categories, candidates and outliers, where the former is closer to the correct value than the latter. Finally, the DP-DHR is proven to guarantee DP, and the experimental results also show that the utility is not sacrificed for the enhancement of privacy by much (a ratio of 4–7% on average), even in the condition of some executors (less than one-half) being controlled by adversaries.

Keywords: differential privacy; endogenous safety and security; dynamic heterogeneous redundant architecture; decision strategy; hypersphere clustering

check for updates

Citation: Kang, Y.; Zhang, Q.; Jiang, B.; Bu, Y. A Differentially Private Framework for the Dynamic Heterogeneous Redundant Architecture System in Cyberspace. *Electronics* **2024**, *13*, 1805. https://doi.org/10.3390/electronics13101805

Academic Editor: Aryya Gangopadhyay

Received: 10 April 2024
Revised: 5 May 2024
Accepted: 6 May 2024
Published: 7 May 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Recent decades have witnessed the development of information technology; thousands of complicated information systems not only bring convenience but also lead to a huge amount of cyberspace security and safety problems [1–5]. Traditional methods always protect the information system (or cyberspace) by adding 'fencing' around the system, such as firewalls [6,7], an Intrusion Detection System (IDS) [8,9], etc. These conventional protection methods attempt to eliminate cyberspace safety and security problems through various approaches.

However, *absolute* safety and security are impossible [10] since the vulnerabilities and backdoors are inevitable in the present information systems which have millions of lines of code, and adversaries can either exploit preexisting vulnerabilities and backdoors or discover them later, then launch cyber attacks. To address this issue, the ESS theory [11,12] and its practice, the DHR architecture, are proposed, where the attacks can be thwarted through a meticulously designed system architecture, without any prior knowledge of adversaries [13]. Specifically, in the DHR architecture, upon receiving an input, equivalent but heterogeneous executors process the same input and output intermediate results, then

a decision module processes these intermediate results using a decision-making strategy (e.g., the majority principle) and outputs the final result. From this perspective, the DHR architecture system is a specified multi-heterogeneous system. Due to the superiority, the ESS theory along with the DHR architecture have been introduced to the fields of wireless communications systems [14], sixth-generation mobile communication (6G) networks [15], embedded systems [16], industrial control systems [17], zero trust systems [18], etc.

While the ESS theory offers a groundbreaking perspective on safeguarding cyberspace, the privacy discourse surrounding the DHR architecture remains unexplored. Nowadays, vast amounts of data are being collected and analyzed, posing privacy challenges for all information systems, particularly in sensitive domains such as industrial control, wireless communications, zero trust (including healthcare and finance), etc. Therefore, it is imperative to integrate this new breed of cybersecurity technology with privacy-preserving techniques, in order to provide comprehensive security measures for these sensitive domains.

Differential privacy (DP) [19,20], a rigorously established tool, has emerged as the de facto definition of privacy and is being widely applied in various scenarios due to its simplicity, soundness and completeness, including the Internet of Things (IoT) [21], healthcare [22], data management [23–25], machine learning [26–28], and quantum computations [29,30]. Its purpose is to protect the sensitive data of individuals who interact with information systems. Specifically, DP introduces random noise into the information system, preventing adversaries from inferring any specific information about individuals participating in data collection. When compared to most alternative security mechanisms, such as Homomorphic Encryption (HE) and Trusted Execution Environments (TEE), DP exhibits superior complexity and scalability, while maintaining comparable performance as shown in Table 1. Researchers have proved that it guarantees privacy mathematically. And in practice, researches have demonstrated that DP can effectively mitigate various types of attacks, such as data inference attacks, attribute inference attacks, membership inference attacks, and memorization attacks [31–34].

Table 1. Comparisons of HE, TEE, and DP.

	Methods	**Computational Complexity**	**Scalability**	**Real-Time**	**Privacy Confidence** *
HE	Encryption	High	Medium	Medium	High
TEE	Hardware Isolation	Medium	Low	High	Low
DP	Noise Perturbation	Low	High	High	Medium

* For the term 'Privacy Confidence', DP has theoretical guarantees as shown in Definition 1. However, TEE lacks corresponding theoretical foundations, leading to its classification as 'Low', while DP is classified as 'Medium'.

In this paper, one of the limitations of the DHR architecture, namely, data privacy, is addressed, by introducing DP to the ESS theory. We propose a DP-DHR framework that aims to protect sensitive information exposed to the DHR architecture information system. However, it is not a straightforward task to apply DP to the traditional DHR architecture due to the introduction of random noise. This noise can potentially confuse the decision module when judging perturbed intermediate results. To fix it, we propose the *advanced decision strategy* and *hypersphere clustering method* to filter out values that are far from the true value, including the normal results but adding larger noises with very small probabilities or the influenced results controlled by adversaries. The contributions of this paper can be summarized as follows:

1. We propose the DP-DHR framework integrating DP into the DHR architecture. To the best of our knowledge, this is the first work to incorporate data privacy considerations into the DHR architecture.
2. We conduct an analysis of the privacy property of DP-DHR using the Gaussian mechanism and the post-processing lemma of DP. The theoretical results demonstrate that the guarantee of (ϵ, δ)-DP is achieved.

3. We perform experiments on a simulated DHR architecture system to verify the utility of our proposed DP-DHR framework. The experimental results show that the utility is similar to the traditional DHR architecture, indicating that enhancing privacy does not sacrifice the system's utility. On average, the difference in utility is around 4–7%. Additionally, the experiments demonstrate that even if an adversary has control over some of the executors (less than one-half), they cannot manipulate the output of the DP-DHR system without prior knowledge of the zero-noise result.

The rest of the paper is organized as follows. The related work is discussed in Section 2. Preliminaries are introduced in Section 3. We propose a DP-DHR framework and give the privacy guarantees in Section 4. Experimental results and discussions are given in Section 5. Finally, we conclude the paper in Section 6.

2. Related Work

The ESS theory [11] proposes that vulnerabilities and backdoors, whether visible or invisible, are inherent in information systems in practice, and claims the inevitability of endogenous security problems in information systems. These problems cannot be completely eliminated; instead, they tend to evolve into more complex threats [35] alongside traditional protective measures such as digging holes, patching, sealing doors, checking poisons, and killing Trojan horses. Therefore, inspired by the consensus mechanism [36] and the dissimilar redundant structure (DRS) [37], the DHR architecture is designed to address endogenous safety and security problems from the perspective of system architecture, which has been proved according to the Shannon channel coding theorem [38]. Additionally, there are various types of mimic devices that utilize the DHR architecture in practice, including routes, clouds, and web servers [12]. 'Mimic devices' implement a deceptive mimicry, camouflage, or invisibility mechanism that creates a physical or logical scenario akin to the uncertainty principle of quantum mechanics. This structure and mechanism can be utilized to enhance the 'trinity' functionality: application service provision, reliability assurance, and secure and trusted defense within the target entity.

The DHR architecture finds application across various critical domains: Wireless ESS (WESS) is introduced in [14], analyzing ESS attributes of the wireless channel. Moreover, the DHR architecture is extended to WESS to address ESS challenges in wireless communication systems, demonstrating improvements in signal-to-noise ratio and anti-interference capabilities. In [15], to tackle complex ESS issues in 6G networks, the DHR architecture 6G polymorphic network is devised. Here, both the control plane and user plane of the 6G core network are based on the DHR architecture. The DHR architecture is leveraged in embedded systems in [16]. Theoretical and experimental results across diverse operating conditions under various strategies demonstrate the effectiveness and correctness of the DHR architecture. Industrial control systems adopt a similar concept: parallel redundant mimicry (this is similar to the DHR architecture). The system resilience is enhanced, introducing four protection dimensions, Technology, Emergency, Management, and Time (TEMt) in [17]. The internal structure of the basic framework of the zero-trust system is restructured in [18] through the lens of the DHR architecture, termed ZETSA. This restructuring improves network services, reliability, and security defenses with broad applicability.

DP [19,20] is proposed to formally define the concept of data privacy and offers solutions to privacy concerns in diverse domains, including the Internet of Things (IoT) [21], healthcare [22], data management [23–25], machine learning [26–28], and quantum computations [29,30]. Extensive research studies [31–34] have illustrated that DP can block some attacks such as data inference attacks, attribute inference attacks, membership inference attacks, and memorization attacks.

Although the ESS theory presents a new paradigm for addressing cyberspace challenges and the DHR architecture offers an implementation approach, the privacy is neither considered theoretically nor practically. DP provides a method for privacy protection with theoretical guarantees, but there is currently limited research on the integration of these two technologies. To address these issues, we propose the DP-DHR framework in this

paper, which integrates DP into the DHR architecture to resolve privacy concerns. We also mathematically prove that the DP-DHR framework satisfies the DP property. Moreover, directly incorporating DP into the DHR architecture can pose significant challenges to the decision-making module, potentially enabling adversaries to manipulate system outputs, and undermining the ESS theory. To overcome these challenges, we design the *hypersphere clustering method* to construct the *advanced decision strategy*. This method classifies the intermediate results perturbed by random noise (to guarantee DP) into two sets: the candidates set and the outliers set. The candidates set is then utilized to make the final decision. This approach reduces the confusion of the decision module caused by different perturbed intermediate results.

3. Preliminaries

In this section, we introduce basic definitions and notations of the ESS theory, the DHR architecture, and DP.

3.1. ESS and DHR

Simply put, the ESS theory states that vulnerabilities, backdoors and random faults during system running are inevitable in an information system, but even in such circumstances where adversaries can exploit these weaknesses to launch cyber attacks, the probability of the system producing incorrect outputs is negligible.

Motivated by the reliability theory and methods, one of implementations of the ESS theory is proposed, namely, the DHR architecture as shown in Figure 1, in which there are m functionally equivalent but heterogeneously implemented executors in total, and k executors among them are chosen to work at the same time (called *online executors*; this will lead to $m - k$ free executors, called *offline executors*). As shown in Figure 1, a traditional system can be considered a *single executor* system and the DHR architecture system is formed by its heterogeneous extension k times when working.

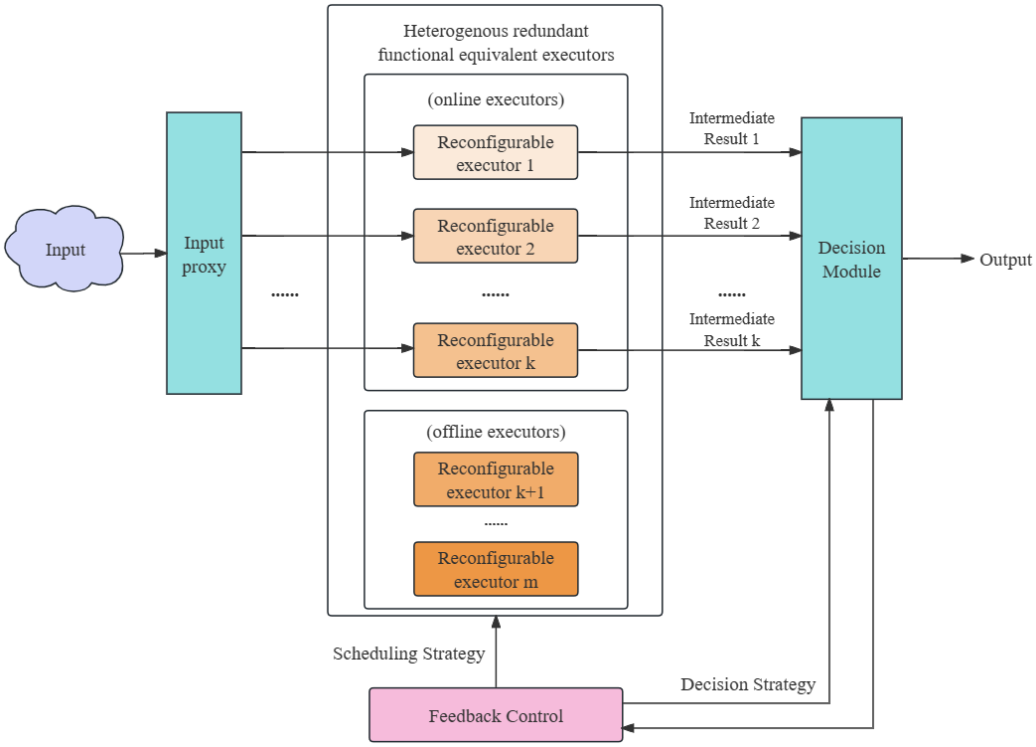


Figure 1. The DHR architecture.

The goal of the DHR architecture is **to enhance the reliability of information transmission** and the workflow can be summarized as follows:

1. The *input proxy* receives the input of the system and sends it to all the *k online executors*.
2. Each online executor processes the input and forwards the resulting output (called the *intermediate result*) to the *decision module*.
3. The decision module renders a verdict based on all k intermediate results according to the *decision strategy* (in this paper, we take the majority rule as the decision strategy) and choose (or generate) the final result.
4. The system outputs the final result.

To further demonstrate the superiority of the DHR architecture system against the traditional information system, we consider the following example: If an attacker controls an executor, then the whole traditional system is controlled since it can be considered a single executor system, and the attacker can output whatever he/she would like. In contrast, with the DHR architecture system under the same case that is not controlled by the attacker, if $k > 2$, the decision module can obtain correct intermediate results from other executors since different executors are heterogeneously implemented and it is hard for the attacker to control most of them. As a result, the system output cannot be tampered with since the DHR architecture enhances the reliability of the system.

In brief, due to the heterogeneous implementations of these executors, it is extremely difficult to identify common vulnerabilities or weaknesses among them, e.g., exploiting shared vulnerabilities between Windows, Linux and macOS systems is nearly impossible; thus, the success probability of attacks is relatively low. Via the discussion mentioned above, one may easy to observe that the DHR architecture system is one of the practices of the multi-heterogeneous systems.

While it may be challenging to successfully attack all of the online executors, gaining control over one or a few of them is comparatively straightforward. Considering the majority rule as an illustrative decision strategy, the system's output cannot be altered unless a minimum of $\lceil \frac{k}{2} \rceil$ online executors are compromised and manipulated to produce the same intermediate result. The scenario where more than $\lceil \frac{k}{2} \rceil$ online executors are compromised is called a 'common modulus attack'. However, for the DHR architecture, achieving a 'common modulus attack' is theoretically difficult. This is because of the heterogeneously redundant implementations, which undergo scheduling and cleaning operations periodically or when some executors are suspected of being compromised. Specifically, if some executors attempt to deviate from the original function of the operation, called 'bad executors', the decision module will detect this discrepancy compared to all intermediate results from other executors. Consequently, the system will cease their operations and assign new executors from the offline pool based on the scheduling strategy. As for the compromised executor, it will be reconfigured to eliminate the attacked parts. The system will replace online executors even in non-attack scenarios, following the scheduling strategy, in order to reduce the attack surface. Thus, the output of the system remains unaffected by the 'out of control' situation of a few executors, as the decision and scheduling strategies prevent any fluctuations in the intermediate results from impacting the final output.

Figure 1 shows the fundamental principles and distinctive features of the DHR architecture: dynamism, heterogeneity, and redundancy. The scheduling strategy, encompassing both planned and triggered replacement between online and offline executors, embodies the dynamism of the DHR architecture. This strategy effectively removes any attacked executors, resulting in a reduced attack surface. The presence of m different but equivalent executors showcases the heterogeneity of the architecture, ensuring that no single attack can compromise the entire system. The combination of k online executors and $m - k$ offline executors represents the redundancy, which forms the basis for the aforementioned dynamic and heterogeneous features.

The theory of coding channel [39] is introduced to illustrate the efficacy of the DHR architecture, drawing inspiration from the Shannon channel coding theorem [38]. It indicates that the DHR architecture enhances cyberspace endogenous safety and security, particularly in terms of reliable transmission.

In the DHR architecture, we denote the executors by $\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_k, \mathcal{E}_{k+1}, \cdots, \mathcal{E}_m$, where the first k executors are online executors and the left are offline ones (due to the dynamic of the DHR architecture, the online and offline executors are not fixed, and here, we use this notation for simplicity). For a fixed task, the system can be formulated as function f, and given the input I, the expected (correct) output of the system is denoted by $f(I)$. Similarly, the executors $\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_m$ can also be formulated as functions, denoted by $f_1, f_2, \cdots, f_m$, respectively, and the intermediate result of the corresponding executor is represented by $f_i(I)$. In the DHR architecture, the executors are heterogeneous but equivalent, so the expected (correct) functions of all the heterogeneous executors are the same as the system, i.e., $f_1 = f_2 = \cdots = f_m = f$. Additionally, given an input I, we denote the intermediate results and the final output of the system *in practice* by $o_1(I), o_2(I), \cdots, o_m(I)$ and $o(I)$, respectively.

3.2. Differential Privacy

Let $\mathcal{P}$ be a probability measure defined on the data space $\mathcal{Z}$ and let database $S = \{z_1, \cdots, z_n\}$ be independently drawn from $\mathcal{P}$. Databases S, S' differing by at most one data instance are denoted by $S \sim S'$, called adjacent databases.

Definition 1 (Differential Privacy [20]). *Algorithm $\mathcal{A} : \mathcal{Z}^n \to \mathbb{R}^p$ is (ϵ, δ)-differential privacy (DP) if for all $S \sim S'$ and event $O \in range(\mathcal{A})$,*

$$\mathbb{P}[\mathcal{A}(S) \in O] \leq e^{\epsilon}\mathbb{P}\left[\mathcal{A}(S') \in O\right] + \delta, \tag{1}$$

where ϵ, δ are the privacy budgets.

Differential privacy requires essentially the same distributions to be drawn over any adjacent databases so that the adversaries cannot infer whether an individual participates on the data collection. Some kinds of attacks, such as data inference attack [31], attribute inference attack [32], membership inference attack [34], and memorization attack [33], can be thwarted by DP.

The Gaussian mechanism is proposed to guarantee DP, shown in Lemma 1.

Lemma 1 (Gaussian Mechanism [20]). *Let $\epsilon \in (0, 1)$ be arbitrary, and $f : \mathcal{Z}^n \to \mathbb{R}^p$ is a p-dimensional function, then for a database $S \in \mathcal{Z}^n$, if function f returns:*

$$f_\epsilon(S) = f(S) + b, \tag{2}$$

where $\mathbb{R}$ is the real number set, $b \sim \mathcal{N}(0, \sigma^2 I_p)$ is the p-dimensional Gaussian random noise and

$$\sigma > \frac{\Delta_2(f)\sqrt{2\log(1.25/\delta)}}{\epsilon}, \tag{3}$$

then $f_\epsilon(S)$ satisfies (ϵ, δ)-DP.

In the inequality (3) of Lemma 1, $\Delta_2(f)$ represents the ℓ_2-sensitivity of the function f, defined as the following.

Definition 2 (ℓ_2-sensitivity [20]). *For any adjacent databases $S \sim S'$, the ℓ_2-sensitivity of the function $f : \mathcal{Z}^n \to \mathbb{R}^p$ is:*

$$\Delta_2(f) = \max_{S, S' \in \mathcal{Z}^n} \left\| f(S) - f(S') \right\|_2, \tag{4}$$

where $\|\mathbf{x}\|_2$ represents the Euclidean norm (ℓ_2 norm) of vector $\mathbf{x}$. Suppose that vector $\mathbf{x} = [x_1, x_2, \cdots, x_p]$ is p-dimensional, and its Euclidean norm is $\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{p} |x_i|^2\right)^{1/2}$.

Remark 1. *Note that the ℓ_2-sensitivity $\Delta_2(f)$ is a quantity inherent in function f: it only depends on f, cannot be chosen by policy, and is independent of the actual database [19].*

For DP, an important property is the post-processing property, which claims the fact that the composition of a data-independent mapping g with an (ϵ, δ)-DP algorithm f is also (ϵ, δ)-DP, formally described as the following.

Lemma 2 (Post-processing [20]). *Let $f : \mathcal{Z}^n \to \mathbb{R}^p$ be a randomized algorithm that is (ϵ, δ)-DP, let $g : \mathbb{R}^p \to \mathbb{R}^p$ be an arbitrary randomized mapping, then $g \circ f : \mathcal{Z}^n \to \mathbb{R}^p$ is (ϵ, δ)-DP, where symbol $\circ$ represents the composition of functions.*

4. Materials and Methods

In this section, we propose the DP-DHR framework (Figure 2), which firstly guarantees differential privacy of the DHR architecture (Figure 1). All the notations are given in Section 3.

In Figure 1, it can be observed that the input I is transmitted to all k online executors, making it possible for any of them to disclose sensitive information. Therefore, to achieve DP, random noise should be injected into the k intermediate results obtained from the online executors $\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_k$.

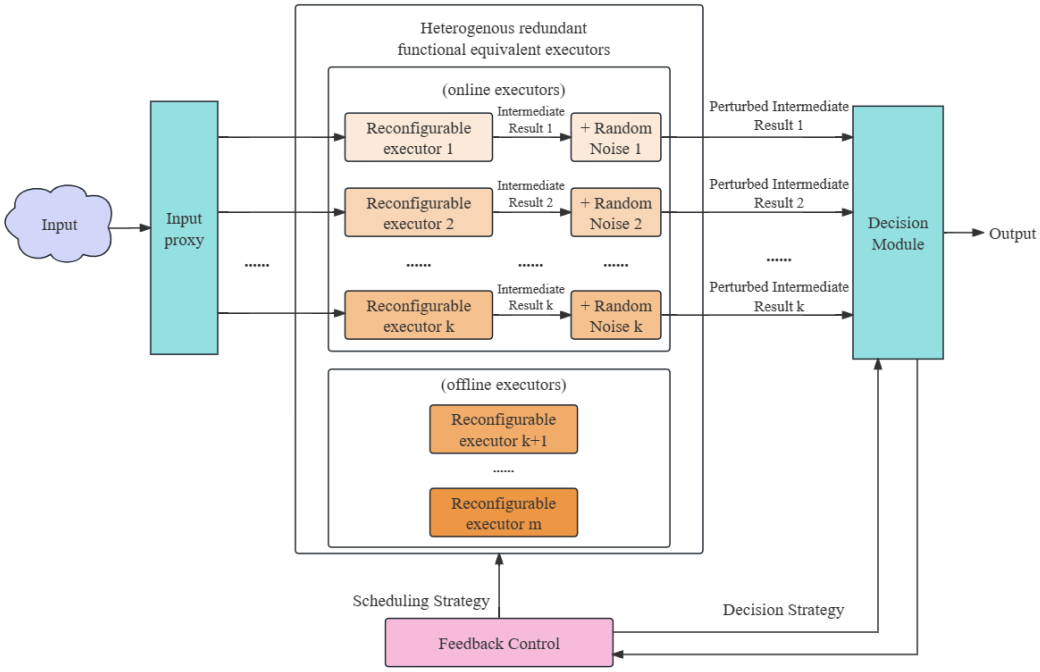


Figure 2. The DP-DHR framework.

In Figure 2, the random noises that are independently sampled by $\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_k$ are denoted by $b_1, b_2, \cdots, b_k$, respectively. The perturbed intermediate result of each executor $\mathcal{E}_i$ on the input I is denoted by $o_i(I) + b_i$. In contrast to the traditional DHR architecture, the DP-DHR framework has differences in two coins. One is that each online executor $\mathcal{E}_i$ samples the noise b_i and adds it to the intermediate result $o_i(I)$ before sending it to the decision module. Another is that the decision module makes a verdict upon k *perturbed intermediate results* according to the advanced decision strategy which requires invoking the machine learning algorithm.

Remark 2. *A key challenge of applying DP to the DHR architecture is making a decision upon perturbed intermediate results. Specifically, for the traditional DHR architecture (Figure 1), under normal system operating conditions (no attacks or failures happen), each online executor $\mathcal{E}_i \in \{\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_k\}$ on a given input I outputs $o_i(I)$ such that $o_i(I) = f_i(I) = f(I)$. In other words, the expected output of each online executor is consistent with the actual result in the normal system running. In this scenario, the decision module can easily make a verdict using the majority rule, and the expected system output also aligns with the actual one. On the other hand, if attacks or*

failures occur, the executors' outputs and even the system output may be influenced. Concretely, the online executors are divided into two categories, the 'attacked executors' set A and the 'normal executors' set U:

$$\begin{aligned} A &= \{i|\mathcal{E}_i \quad is \quad attacked \quad and \quad controlled \quad by \quad the \quad adversary\}, \\ U &= \{j|\mathcal{E}_j \quad is \quad not \quad controlled \quad by \quad the \quad adversary\}, \end{aligned} \tag{5}$$

the practical intermediate results are

$$\begin{aligned} o_i(I) \neq f_i(I) = f(I), \quad i \in A, \\ o_j(I) = f_j(I) = f(I), \quad j \in U. \end{aligned} \tag{6}$$

If the size of set A satisfies $|A| < \lceil \frac{k}{2} \rceil$, which means the common modulus attack does not appear (for the DHR architecture system, a successful common modulus attack means 'zero utility'; in this case, data privacy without utility is meaningless, so we do not discuss the common modulus attack condition in this paper), the decision module can easily reach a verdict by the majority rule, as all 'normal executors' generate intermediate results identical to $f(I)$.

However, in the DP-DHR framework, the above decision process may be prone to errors because of introducing random noise into the intermediate results. It is extremely hard to obtain $b_1 = b_2 = \cdots = b_k$ due to the random noises independently sampled by online executors. As a result, k perturbed intermediate results collected by the decision module are highly likely to be different even under the non-attack condition, and it will be more complicated under attacks. Thus, the decision strategy of the majority rule is not directly performed in the DP-DHR framework, and we need to redesign an advanced decision strategy in this paper.

The advanced decision strategy is illustrated in Figure 3, where the decision module is highlighted, and the details of online executors are omitted.

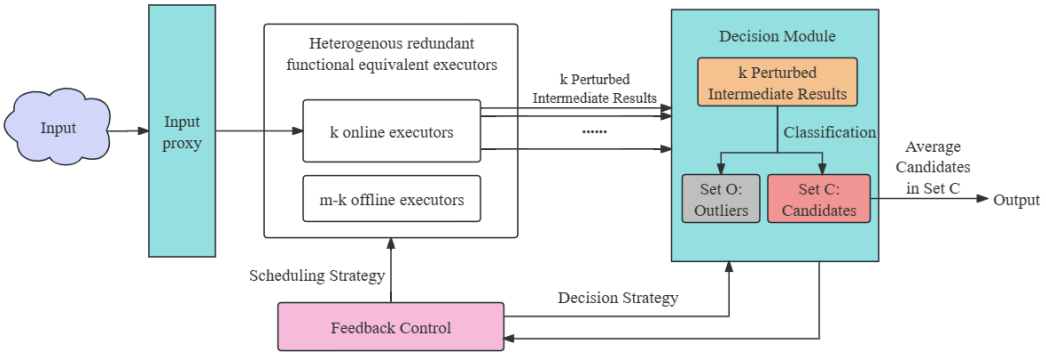


Figure 3. The advanced decision strategy of the DP-DHR framework.

Specifically, the decision module receives k perturbed intermediate results and divides them into two categories, the candidates C and the outliers O, which is the key of the advanced decision strategy. Due to the majority rule, the number of the candidates C is larger than that of the outliers O, and the perturbed intermediate results in the candidates C can be applied to generate the system output.

Details of the DP-DHR framework with the advanced decision strategy are formally described in Algorithm 1.

The differences between the DP-DHR framework and traditional DHR architecture can be described from three perspectives:

1. In lines 3 and 4 of Algorithm 1, each online executor $\mathcal{E}_i$ injects independently sampled Gaussian random noise into the intermediate results to guarantee the DP property.
2. The introduction of random noise to the intermediate results poses challenges for the decision module in generating the final output (discussed in Remark 2). To address

this problem, the decision module applies a classification method (the Algorithm 1) to make a decision upon the perturbed intermediate results.

3. After classification, the DP-DHR framework outputs the average of the candidates C. It is worth noting that the random noise added to the intermediate results is zero-mean, and the averaging process can somewhat mitigate the impact of the noise.

In Algorithm 1, the classification method is utilized to separate k perturbed intermediate results into the candidates C and the outliers O. To ensure the utility of the DP-DHR framework, it is preferable to minimize the gaps between the expected output (correct, without random noise) and the candidates C. Conversely, the gap between the expected output and the outliers O should be relatively large to guarantee DP. Therefore, before classifying the perturbed intermediate results, the expected output should be first estimated (since the decision module cannot obtain the exact expected value from the executors due to the perturbations).

Algorithm 1 Differentially private algorithm for the DHR architecture.

Require: Privacy budgets $\epsilon, \delta > 0$, input I, the executor functionality f
Ensure: A system output $o_b(I)$ guaranteeing (ϵ, δ)-DP

1: **for** $i = 1$ to k **do**
2: $\mathcal{E}_i$ gives the p-dimensional intermediate result $o_i(I)$ upon the input I;
3: $\mathcal{E}_i$ samples the Gaussian random noise $b_i \sim \mathcal{N}(0, \sigma^2 I_p)$ independently;
4: $\mathcal{E}_i$ sends the perturbed intermediate result $o_i(I) + b_i$ to the decision module;
5: **end for**
6: The decision module collects k perturbed intermediate results, invokes the classification method of the *hypersphere clustering method* (the Algorithm 2) and obtains two categories, the candidates C and the outliers O:

$$\begin{aligned} C &= \{v_{11}, v_{12}, \cdots, v_{1k_1}\}, \\ O &= \{v_{21}, v_{22}, \cdots, v_{2k_2}\}, \end{aligned} \tag{7}$$

whose sizes are $|C| = k_1, |O| = k_2$ s.t. $k_1 > k_2, k_1 + k_2 = k$;
7: $o_b(I) = \frac{1}{k_1} \sum_{i=1}^{k_1} v_{1i}$
8: Return $o_b(I)$

In line 3 of Algorithm 1, the Gaussian random noise added to the original expected intermediate results from online executors has a zero-mean, implying that the expected result (without random noise) and its corresponding perturbed intermediate result (with random noise) are statistically unbiased. Based on this observation, a straightforward approach to approximate the expected output for the decision module is to compute the average of all the perturbed intermediate results from online executors and treat this average as the estimated output. Furthermore, the accuracy of this approximation improves as the number of online executors k increases. Nonetheless, this approximation approach is susceptible to attacks because the influence of a few controlled executors can manipulate the estimated value (the average) and deviate it from the expected result. The accuracy of the approximation decreases as adversaries gain control over more executors.

On the other hand, the candidates C should be close to the estimated output, and the outliers O should be far from the estimated output. For the p-dimensional output, the distribution of the candidates is a p-dimensional hypersphere around the evaluated expected output, and the distribution of the outliers is a p-dimensional hyperspherical shell outside the hypersphere. However, this classification requirement cannot be easily achieved by traditional clustering methods such as k-means [40] and its variants [41,42], spectral clustering [43], density-based clustering [44], (variational) auto-encoder (AE or VAE)-based clustering [45,46], graph-based clustering [47], etc.

In brief, in Algorithm 1, the intermediate results should be classified into two categories, the candidates set C and the outliers set O, according to the invisible expected correct

result $o(I)$. Considering that the DHR architecture aims to ensure reliable transmission, i.e., the correct output of the system, certain intermediate results are classified into the outliers set O due to the following reasons:

(1) They originate from executors controlled by adversaries attempting to manipulate the output.
(2) The sampled Gaussian random noise (to guarantee DP) is too large.

In Algorithm 1, due to the property of Gaussian random noise injected into the intermediate results, the perturbed intermediate results are near $o(I)$ with high probability. These factors result in the following observation: the candidates set C comprises intermediate results close to $o(I)$, while elements in the outliers set O are distant from $o(I)$. In a p-dimensional Euclidean space, this observation renders the candidates set C a p-dimensional hypersphere around $o(I)$, with the outliers set O forming a p-dimensional hyperspherical shell outside the hypersphere, further away from $o(I)$. This distinction cannot be achieved through traditional clustering methods.

To overcome this problem, we propose the *hypersphere clustering method* to generate hypersphere-type clusters with robustness against attacks; details can be found in Algorithm 2, in which $|S|$ denotes the size of set S.

Algorithm 2 Hypersphere clustering.

Require: A p-dimensional dataset with k elements $V = \{v_1, v_2, \cdots, v_k\}$, a candidate percentage parameter ρ,a step parameter τ
Ensure: The candidates C and the outliers O

```
for i = 1 to k do
    Initialize r_i = 0, C_i = {v_i}
    while |C_i| ≤ ρk do
        r_i = r_i + τ
        With the center v_i and the radius r_i, construct hypersphere H_i
        For any element v_j ∈ V, if v_j ∈ H_i, C_i = C_i ∪ v_j
    end while
end for
Get the variance of elements in each set C_i, denoting by var_i
C = arg min_{C_i} var_i, O = V \ C
Return the candidates set C and the outliers set O
```

In Algorithm 2, the hypersphere clustering method classifies all the perturbed intermediate results to two categories: the candidates and the outliers. To elaborate further, each perturbed intermediate result is considered the center of a hypersphere in p-dimensions, with ρk other perturbed intermediate results enclosed within the hypersphere. Subsequently, the hypersphere clustering method selects the elements within the hypersphere that have the least variance as the candidates, while the remaining elements are categorized as outliers. This approach ensures that the candidates are more uniformly distributed and physically more compact, which aligns well with the decision strategy of the DHR architecture. Furthermore, this clustering method enhances the robustness of Algorithm 2 against attacks mentioned earlier to some extent as discussed in Section 5.

Here, we theoretically analyze the privacy of our proposed DP-DHR framework.

Theorem 1. *Given the parameters $\epsilon, \delta \in (0,1)$, if the variance of the Gaussian random noise σ^2 satisfies:*

$$\sigma^2 \geq \frac{2\Delta_2(f)^2 \log(1.25/\delta)}{\epsilon^2}, \tag{8}$$

the output $o_b(I)$ of Algorithm 1 guarantees (ϵ, δ)-DP.

The following is a sketch of the proof.

Proof. According to the Gaussian mechanism given in Lemma 1, it is obvious to obtain that each perturbed intermediate result $o_i(I) + b_i$ satisfies (ϵ, δ)-DP sampled by (8).The generated process of the perturbed intermediate results is denoted as the mechanism $g_1(I)$.

The adjudication process of the decision module, including the hypersphere clustering method given in Algorithm 2, is independent of the above mechanism. We denote this adjudication process as the mechanism g_2.

Thus, based on the post-processing property given in Lemma 2, we have that:

$$o_b(I) = g_2 \circ g_1(I) \tag{9}$$

is (ϵ, δ)-DP.

The proof is concluded. □

In Theorem 1, the ℓ_2-sensitivity $\Delta_2(f)$ is determined by the specific system functionality f and can vary across different systems. And it is expected that when the privacy budget becomes smaller (representing stricter privacy requirements), the variance of the random noise also increases. This aligns with our intuition, as smaller privacy budgets necessitate stronger perturbation to protect sensitive information.

Remark 3. *As shown in Definition 1, smaller privacy budgets ϵ, δ indicate that the probability distributions over adjacent databases are more similar, reflecting higher privacy requirements. Consequently, in real-world scenarios, stricter privacy requirements, such as those in financial or healthcare systems, lead to smaller privacy budgets, and vice versa. Furthermore, to guarantee meaningful DP, ϵ is consistently set smaller than 1.0 and δ is always set smaller than $1/n$, where n denotes the size of the database [20]. Moreover, smaller privacy budgets decrease the utility of the system, which is in line with the common sense and leads to a tradeoff between the utility and the privacy; corresponding discussions are studied in the field of machine learning [48], data-processing systems [49], open data sharing [50], etc. The tradeoff is also shown by the experimental results given in Section 5.*

Remark 4. *From the perspective of generalized robust control, the DP-DHR framework strengthens the cyberspace endogenous safety and security. In the cyberspace endogenous safety and security theory, attacks can be considered perturbations that affect information systems. As a result, the defense mechanisms can be formulated as measures to enhance the robustness of the systems. Within the DP-DHR framework, the random noise introduced into the system serves as the perturbation, and the advanced decision strategy implemented is specifically designed to be resilient against the perturbations induced by this random noise. Consequently, the overall robustness of the DHR architecture is significantly enhanced.*

5. Results and Discussion

In this section, we conduct simulated experiments to evaluate the utility and the robustness of our proposed DP-DHR framework.

Before presenting the detailed results, we first give the information of the simulation environment as shown in Table 2. The original data (the 100×30 matrix mentioned below) follow the uniform distribution, which is generated by the Java function random.NextInt(), with random seed 1. The variance of the sampled (zero-mean) Gaussian random noise follows Theorem 1.

Table 2. The simulation environment information.

Term	Information
OS	Windows10
CPU	Intel(R) Core(TM) i7-10700
Memory	64 GB
Programming	Java 20.0.1

In certain fields where sensitive information is collected, such as finance and healthcare, ensuring cyberspace security and data privacy are of the utmost importance. In these contexts, the *frequency* feature of private data is a critical factor to consider. For example, in the field of healthcare, the incidence rate of a specific disease, and in the field of finance, the high or low income rates for a particular population, are crucial statistical measures in the real world. Therefore, in this section, we consider the 'frequency statistics' as the executor functionality of the DP-DHR framework and analyze its utility.

To simulate our experiment, we generate a dataset consisting of 100 data points, each with 30 features. This dataset can be represented as a 100×30 matrix, where each element at the i-th row and j-th column is denoted as e_{ij} and $e_{ij} \in \{0, 1\}$. This representation is frequently utilized in real-life situations, like medical databases that capture the occurrence or absence of diseases within a population, or financial databases that depict salary ranges for various individuals.

Two experiments were performed on this dataset:

- 1-Dimensional Outputs: The outputs of executors are scalar values (1-dimension) with $p = 1$ as specified in Algorithm 1. We performed the experiments using different privacy budgets ϵ and varying numbers of executors k to demonstrate that the differences between the outputs of the DP-DHR framework and the traditional DHR architecture are minimal.
- 2-Dimensional Outputs: The outputs of the executors are 2-dimensional vectors with $p = 2$ as specified in Algorithm 1. We conducted the experiments under the 'normal' (non-attack) and 'attacked' conditions to show that our proposed advanced decision strategy with the *hypersphere clustering method* is robust against attacks to some extent.

In all the experiments described below, we fix the candidate percentage parameter (introduced in Algorithm 2) as $\rho = 60\%$, indicating that 60% of the perturbed intermediate results are selected as candidates C.

5.1. 1-Dimensional Outputs

Given that outputs are 1-dimensional scalars, we define the expected output of the system as the frequency of data points where the value of the first dimension is 1. For example, if the database D consists of data points $[1, 0, 0]$, $[1, 0, 1]$ and $[0, 1, 1]$, denoted as:

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \tag{10}$$

then the expected output of the system is $2/3 = 0.67$. This notion of frequency is often used in scenarios such as determining the prevalence of a particular characteristic within a population.

We conducted experiments with varying numbers of executors, k = 10, 50, 100, 200, 300, 400, 500, and privacy budgets ϵ = 0.01, 0.05, 0.1, 0.3, 0.5, 1.0, while ensuring meaningful privacy $\delta = 10^{-6}$. The experimental results are presented in Table 3, where the absolute differences between the outputs of the traditional DHR system and the DP-DHR system are shown in the parenthesis. For our simulated database, the frequency output by the traditional DHR system is 0.51.

As shown in Table 3, despite some fluctuations, the difference between the outputs of the DP-DHR framework and the traditional DHR architecture increases as the privacy budget ϵ decreases, which means that higher privacy requirements (smaller privacy budget ϵ) derive lower utility of the system. This observation is consistent with the fact that higher privacy levels typically result in lower utility and this phenomenon demonstrates the tradeoff between the utility and the privacy as discussed in Remark 3. Notably, for $\epsilon > 0.1$, the outputs of both systems exhibit somewhat negligible distinction.

Table 3. Frequency output of the DP-DHR framework system.

		ϵ					
		0.01	**0.05**	**0.1**	**0.3**	**0.5**	**1.0**
k	10	1.205 (0.695)	0.317 (0.193)	0.515 (0.005)	0.540 (0.030)	0.521 (0.011)	0.522 (0.012)
	50	0.649 (0.139)	0.720 (0.210)	0.555 (0.045)	0.502 (0.008)	0.513 (0.003)	0.495 (0.015)
	100	0.037 (0.473)	0.604 (0.094)	0.547 (0.037)	0.524 (0.014)	0.508 (0.002)	0.511 (0.001)
	200	0.365 (0.145)	0.579 (0.069)	0.486 (0.024)	0.513 (0.003)	0.506 (0.004)	0.510 (0.000)
	300	0.540 (0.030)	0.533 (0.023)	0.501 (0.009)	0.514 (0.004)	0.510 (0.000)	0.508 (0.002)
	400	0.224 (0.286)	0.617 (0.107)	0.535 (0.025)	0.513 (0.003)	0.501 (0.009)	0.511 (0.001)

Additionally, in spite of the presence of fluctuations, it can be observed that the difference between the outputs of the DP-DHR framework and the traditional DHR architecture generally decreases with an increasing number of executors k. This can be attributed to the fact that the injected random noise has a zero-mean, resulting in an unbiased approximation, as the perturbed intermediate results are averaged over a larger number of executors.

As a result of the aforementioned phenomenon, it can be concluded that for $k \geq 200$ and $\epsilon \geq 0.3$, the outputs of the DP-DHR framework and the traditional DHR architecture are virtually identical.

5.2. Two-Dimensional Outputs

If we consider 2-dimensional outputs, then the frequency of data points with a value of 1 in the first dimension is denoted as d_1, and that in the second dimension is denoted as d_2. Therefore, the expected output of the system can be represented as the pairwise frequency (d_1, d_2). For instance, considering the database D given in (10), the expected output of the system would be $(2/3 = 0.67, 1/3 = 0.33)$. This type of frequency is often utilized when assessing the concordance property of disease prevalence.

In our simulated database, the traditional DHR architecture, without being subjected to any attacks, produces a pairwise frequency output of $(0.51, 0.58)$. This disparity between the outputs of the DP-DHR framework and the traditional DHR architecture (referred to as T-DHR) is illustrated in Figure 4.

Figure 4a depicts the outputs obtained at various privacy budgets ϵ, with a randomly chosen $k = 100$. The green point in the figure represents the output produced by the T-DHR architecture, while the red points correspond to the outputs generated by the DP-DHR framework for different values of ϵ, with the respective values of ϵ marked next to the points for reference.

Similarly, Figure 4b shows the outputs at different numbers of executors k, with a randomly chosen $\epsilon = 0.1$. The green point in the figure represents the output of the T-DHR architecture, whereas the red points denote the outputs generated by the DP-DHR framework. The numbers adjacent to the red points correspond to the respective values of k.

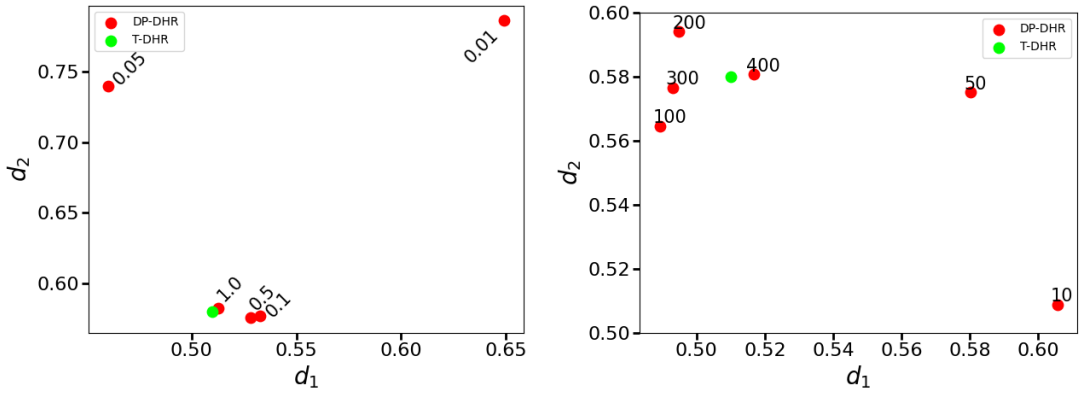
As shown in Figure 4, the difference between the outputs generated by the T-DHR architecture and the DP-DHR framework (measured by the Euclidean distance) diminishes as ϵ and k increase, respectively. This observation is similar to the phenomenon discussed in the experiments involving '1-dimensional outputs' in Section 5.1, in which the former (Figure 4a) demonstrates the tradeoff between the utility and the privacy as demonstrated in Remark 3.

As is evident, the hypersphere clustering method plays a crucial role in the advanced decision strategy outlined in Algorithm 1. Thus, our focus is on evaluating the utility of our proposed clustering method under both normal (unattacked) and attacked conditions.

To begin, we conduct experiments under normal conditions. In order to evaluate the performance of our proposed hypersphere clustering method, we randomly choose $\epsilon = 0.1$ and showcase the clustering results for different numbers of executors k = 10, 50, 100, 200, 300, 400.

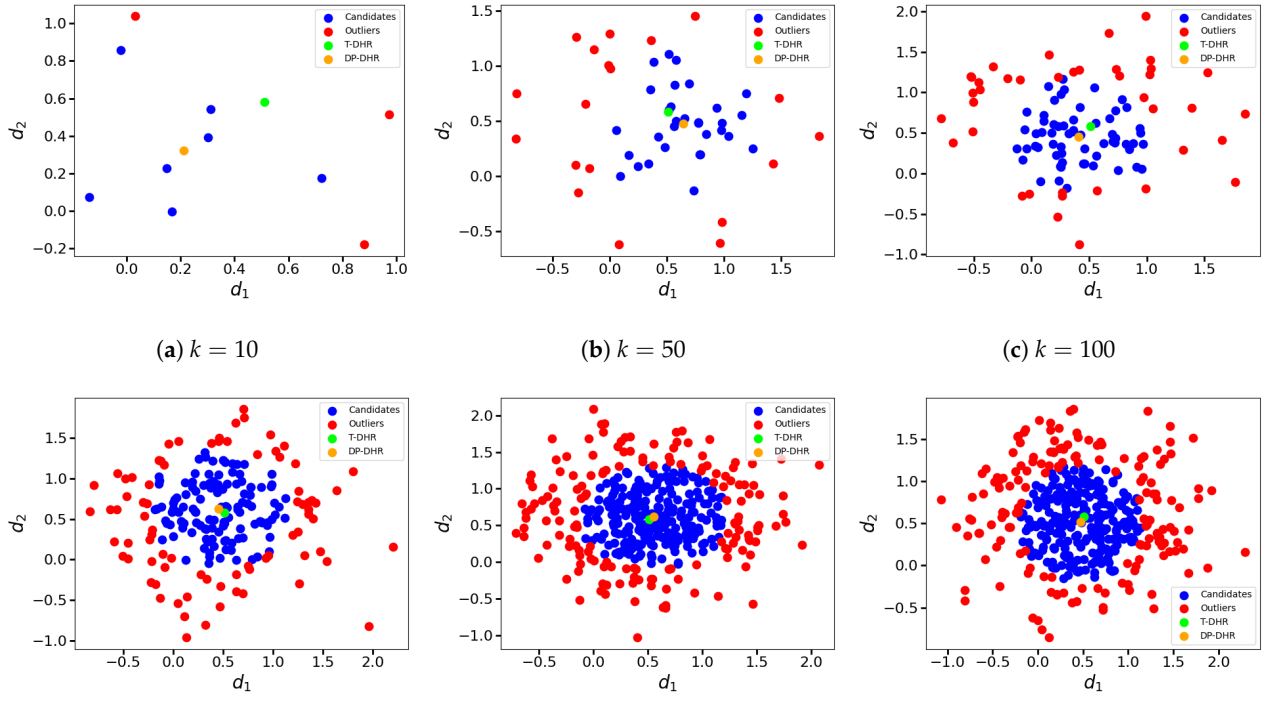
The clustering results are shown in Figure 5. The blue points are the perturbed intermediate results classified to the candidates C, while the red points represent the perturbed intermediate results classified to the outliers O. The green points represent the

outputs of the T-DHR architecture ($d_1 = 0.51$ and $d_2 = 0.58$, as mentioned before), and the brown points represent the outputs generated by the DP-DHR framework.



(**a**) Outputs at different ϵ ($k = 100$) (**b**) Outputs at different k ($\epsilon = 0.1$)

Figure 4. Outputs of DP-DHR framework and T-DHR architecture (unattacked).



(**a**) $k = 10$ (**b**) $k = 50$ (**c**) $k = 100$

(**d**) $k = 200$ (**e**) $k = 300$ (**f**) $k = 400$

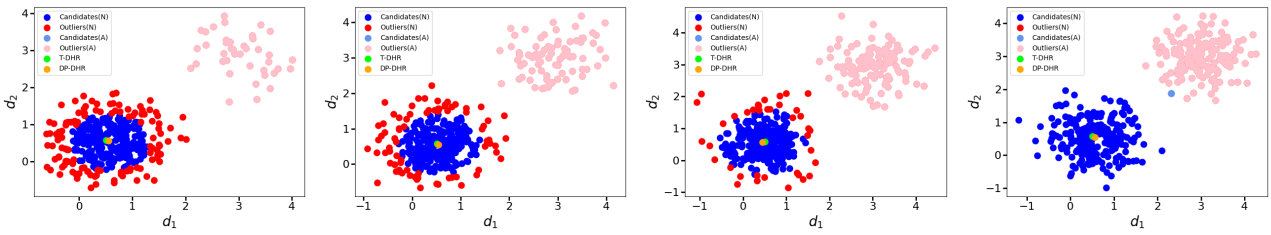
Figure 5. Hypersphere clustering results at different k ($\epsilon = 0.1$).

In Figure 5a–f, we consider a candidate percentage of $\rho = 60\%$ as mentioned above. Accordingly, the number of blue points is slightly larger than $0.6k$ as per Algorithm 2. From Figure 5, it can be seen that for the DP-DHR framework, the injected random noise causes the perturbed intermediate results (represented by the red and blue points) to deviate from the expected system output (the green points). The gaps are particularly large for some individual executors. However, since the Gaussian random noise added to the intermediate results has a zero-mean, the gaps brought by individuals are weakened by averaging all the candidates, as discussed in the analysis of Algorithm 1. As mentioned earlier, due to this property, the gap between the outputs of the DP-DHR framework and the T-DHR architecture decreases with the increase in the number of executor k. When $k \geq 50$,

the difference between them becomes relatively small, and with $k \geq 200$, the outputs of the DP-DHR framework and T-DHR architecture are almost identical.

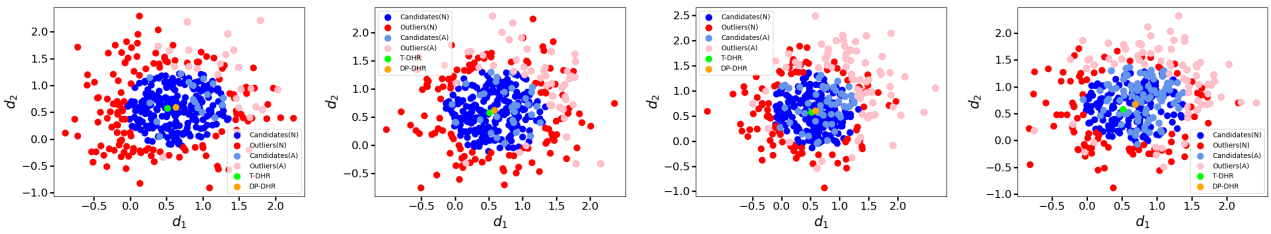
Then, we perform experiments under attacked conditions. In this scenario, the adversary gains control over a certain percentage of executors and aims to manipulate the system output. In our experiments, we consider different *attack rates* (AR), where 10%, 20%, 30%, 40% of the executors are assumed to be under the adversary's control. Due to the decision strategy of the majority rule, all the executors controlled by the adversary are assumed to output the same result. Furthermore, for the attacked condition, we consider two cases: (1) where the random noise injecting process is not controlled, and (2) where the random noise injecting process is controlled. In the latter case, the attack is considered to be more sophisticated. We start with the former case.

In the first case, where the random noise injecting process is not controlled, the outputs of the compromised executors are assumed to be fixed values, such as $o = (3,3)$ and $o = (1,1)$. Then, random noise is added to these outputs o to obtain the perturbed intermediate results. The clustering results are shown in Figures 6 and 7, where the green and brown points represent the same data points as in Figure 5. Additionally, in these figures, the light blue points represent the outputs given by the attacked executors and classified into the candidates C. The dark blue points represent the outputs given by the normal executors and also classified into the candidates C. The light red points correspond to the outputs given by the attacked executors and classified into the outliers O, while the dark red points correspond to the outputs given by the normal executors and classified into the outliers O.



(**a**) AR = 10% (**b**) AR = 20% (**c**) AR = 30% (**d**) AR = 40%

Figure 6. Hypersphere clustering at different AR ($k = 400, \epsilon = 0.1, o = (3.0, 3.0)$, random noise injected normally).



(**a**) AR = 10% (**b**) AR = 20% (**c**) AR = 30% (**d**) AR = 40%

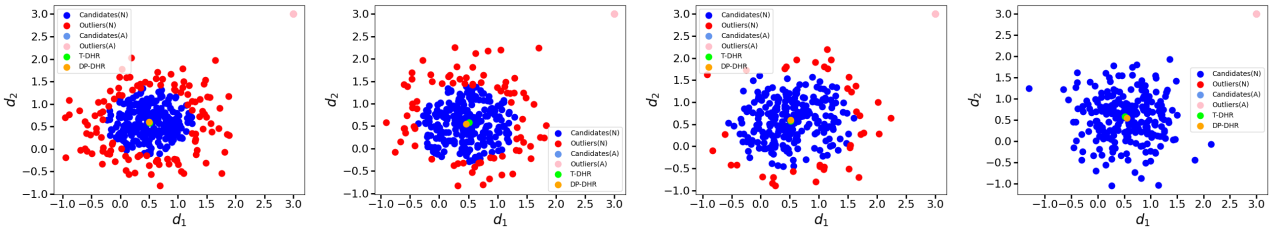
Figure 7. Hypersphere clustering at different AR ($k = 400, \epsilon = 0.1, o = (1.0, 1.0)$ random noise injected normally).

In the case where the adversary outputs $o = (3,3)$, we simulate the scenario where the adversary intends to mislead the system output, regardless of whether it is detected or not. Since the frequency values $d_1, d_2 \in [0,1]$ naturally, and $o = (3,3)$ is obviously far from the reasonable range, this simulation represents an attempt by the adversary to manipulate the system. In this case, Figure 6 illustrates that all the outputs generated by the attacked executors are classified to the outliers O, except for Figure 6d. The reason for this exception is that the candidate percentage parameter and the attack ratio are set to $\rho = 60\%$ and

AR = 40%, which means that at least one attacked perturbed intermediate result is selected as a candidate according to Algorithm 2. However, this does not affect the final system output. In general, unless the candidates percentage $\rho \geq 1-\text{AR}$, it is highly likely for the outputs given by the attacked executors to be classified as candidates when the adversary's output is significantly different from the expected value. This indicates that our proposed hypersphere clustering method is robust against this kind of attack.

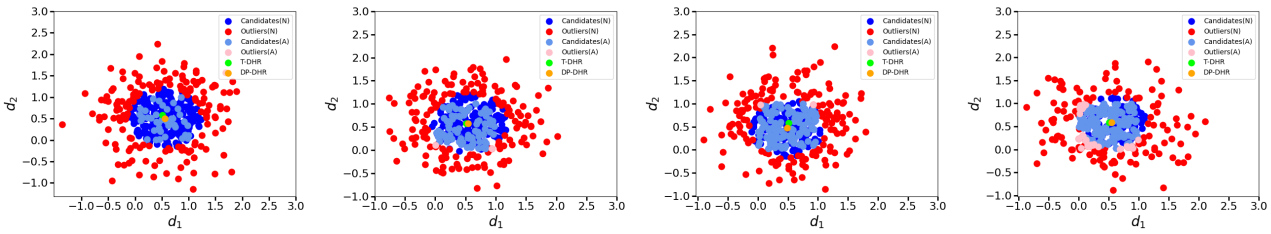
In the case where the adversary output $o = (1,1)$, we simulate the scenario where the adversary intends to mislead the system output but is not easily detected because of the frequency $d_1, d_2 \in [0,1]$ naturally. In this case, Figure 7 demonstrates that while some of the outputs generated by the attacked executors are classified as candidates, the impact of the adversary on the final output of the DP-DHR framework is minimal. This occurrence becomes more pronounced with an increase in AR, indicating the robustness of Algorithm 2 against this type of attack.

Now, let us consider a more complex scenario where the random noise injection process is controlled by the adversary. In this case, we assume that the outputs o of the executors controlled by the adversary are as follows: (1) $o = (3,3)$: Both dimensions of o are set to 3; (2) $o_1, o_2 \in [0,1]$: Two dimensions of o are randomly chosen within the range of $[0,1]$; (3) $o = (1,1)$: Both dimensions of o are set to 1. The clustering results for these scenarios are depicted in Figure 8, Figure 9, and Figure 10, respectively. The point colors in these figures correspond to those in Figures 6 and 7.



(**a**) AR = 10% (**b**) AR = 20% (**c**) AR = 30% (**d**) AR = 40%

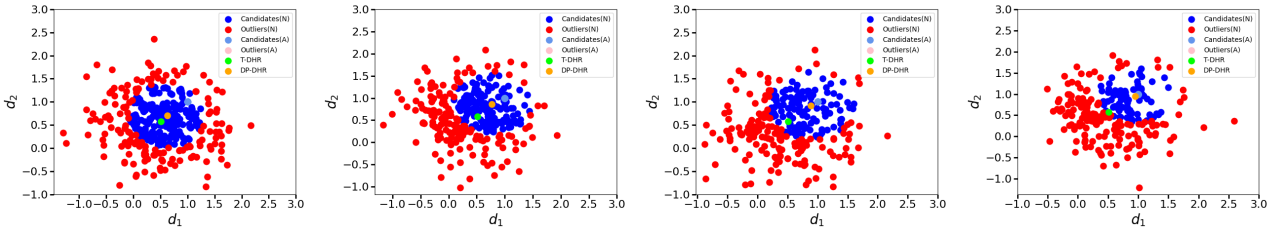
Figure 8. Hypersphere clustering at different AR ($k = 400, \epsilon = 0.1, o = (3.0, 3.0)$, random noise injected controlled).



(**a**) AR = 10% (**b**) AR = 20% (**c**) AR = 30% (**d**) AR = 40%

Figure 9. Hypersphere clustering at different AR ($k = 400, \epsilon = 0.1, o_1, o_2 \in [0.0, 1.0]$, random noise injected controlled).

Regarding the adversary's output $o = (3,3)$, similar to Figure 6, we simulate the scenario where the adversary intends to mislead the system output, whether it is detected or not. In Figure 8, the light red point located at the top right corner (coordinates $(3,3)$) represents all the intermediate results generated by the attacked executors, as all the controlled intermediate results are mapped to this single point. Specifically, in Figure 8a–d, $0.1k$ (40), $0.2k$ (80), $0.3k$ (120), and $0.4k$ (160) intermediate results are respectively mapped to this point.



(**a**) AR = 10% (**b**) AR = 20% (**c**) AR = 30% (**d**) AR = 40%

Figure 10. Hypersphere clustering at different AR ($k = 400, \epsilon = 0.1, o = (1.0, 1.0)$, random noise injected controlled).

The clustering results presented in Figure 8 demonstrate that the perturbed intermediate results classified as candidates C are almost exclusively generated by normal executors, and not by the attacked executors controlled by the adversary. Furthermore, the results obtained from the T-DHR architecture and the DP-DHR framework are similar, indicating that our proposed hypersphere clustering method is robust against this kind of attack.

In the case where the adversary's output $o = (o_1, o_2)$, we simulate that the adversary randomly generates two dimensions $o_1, o_2 \in [0, 1]$. The intention of the adversary is to mislead the system output while avoiding easy detection, as the frequency $d_1, d_2 \in [0, 1]$ naturally. Figure 9 displays the clustering results in this scenario.

In this case, the intermediate results classified as candidates C may contain results generated by the attacked executors. The ratio of light blue points (representing attacked candidates) to the total number of blue points (attacked candidates + (normal) candidates) increases with the growth in AR. However, the difference between the outputs of the T-DHR architecture and the DP-DHR framework is not significant, regardless of the scale of AR. One reason for this is that the expected output in the T-DHR architecture is close to $(0.5, 0.5)$. If the expected result was different (e.g., (0,0)), the gap between the outputs might be larger.

In the case where the adversary's output $o = (1.0, 1.0)$, we simulate the scenario where the adversary aims to mislead the system output without being easily detected. The clustering results are shown in Figure 10, in which all the attacked results are mapped to the single point $(1, 1)$, represented by the light blue point.

As mentioned above, 40% of the attacked results are stained by the light blue color, indicating that all the attacked results are classified as candidates. This represents that all the intermediate results generated by the attacked executors are misclassified and the system is successfully attacked.

As shown in Figure 10, the gap between the outputs given by the DP-DHR framework and the T-DHR architecture is larger compared to previous conditions. When comparing Figure 10a–d to Figure 9 (or Figure 8a–d), it is obvious that the gaps between the brown and green points are notably wider in Figure 10. Due to the relatively large distance between $(1, 1)$ and the expected output $(0.51, 0.58)$, the output of the DP-DHR framework (the brown point) is 'dragged' away from the expected output of the T-DHR architecture (the green point) by the attacked executors. When AR is small, the consequence is not very pronounced. For example, in Figure 10a, when AR $= 10\%$, although all the attacked results are classified as the candidates C, the gap between the DP-DHR output and the T-DHR output is not significant due to the small AR. On the contrary, when a large AR is employed as shown in Figure 10d, the attacked results would significantly mislead the output of the DP-DHR framework. In conclusion, it is evident that the gap between the DP-DHR framework and the T-DHR architecture becomes larger with the increase in AR as shown in Figure 10a–d. This observation aligns with common sense and the expectations.

In summary, the proposed DP-DHR framework demonstrates resilience against the majority of attack conditions, with one notable exception: when the adversary coerces the majority of attacked executors to produce the same value $o_{\mathcal{A}}$ within a reasonable range

(e.g., (0,1) in frequency), particularly when the disparity between $o_{\mathcal{A}}$ and the expected correct value $o_{\mathcal{E}}$ is substantial. This issue is exacerbated with higher levels of AR as shown in Figure 10.

Here, we discuss the effects of the Advanced Persistent Threat (APT) attack, using the coordinated multi-vector attack as an example. In the DHR architecture (or the proposed DP-DHR framework), a coordinated multi-vector attack occurs if $AR > 1/k$ (when more than one executor is controlled by the adversary) and we classify the attack into two categories:

1. $AR < 50\%$: Due to the majority principle decision strategy, in the case of the traditional DHR architecture, the system's output cannot be tampered with. This is because the intermediate results provided by the controlled executors are categorized as 'attacked', while the correct intermediate results are considered when making decisions. As a result, the integrity and the availability of traditional DHR do not decrease when $AR < 50\%$; however, the confidentiality (data privacy) is not protected. On the other hand, for the proposed DP-DHR framework, the theoretical and experimental results given above (Theorem 1 and Figures 6–10) enhance the confidentiality (data privacy) without much sacrifice of the integrity and the availability.
2. $AR \geq 50\%$: If more than half of the executors are controlled by an adversary, the adversary would definitely control the output of the traditional DHR architecture. This is because of the majority principle decision strategy, which allows the adversary to map the majority of the intermediate results to a single value. In this case, neither the traditional DHR architecture nor the proposed DP-DHR framework could ensure the integrity and the availability. However, the confidentiality (data privacy) is guaranteed in the DP-DHR framework rather than the traditional DHR architecture. It is worth noting that such a case is unlikely to occur in the DHR architecture (and the DP-DHR framework). This is because different executors are heterogeneously designed, reducing the probability of the majority of these heterogeneous executors being simultaneously controlled. Additionally, the dynamic property of the DHR architecture (and the DP-DHR framework) further decreases the probability of $AR \geq 50\%$.

Remark 5. *Through our analysis of the attacked results, we observed that the outputs generated by the controlled executors can alter the distributions of the injected Gaussian random noise, which in turn affects the privacy properties of the entire DP-DHR system. This phenomenon is even more pronounced in the DHR architecture, as it is designed to tolerate the compromise of several executors. This allows the adversary to construct certain probability distributions using a certain number of controlled executors to affect the privacy properties.*

For instance, to guarantee DP, online executors not under the adversary's control sample Gaussian random noise $b \sim \mathcal{N}(0, \sigma^2)$. However, if the adversary employs a different distribution, such as the uniform distribution with a sample range of (a, b), $b_{\mathcal{A}} \sim \mathcal{U}(a, b)$, and adds this noise to the intermediate results from controlled executors, the resulting noise distribution received by the decision module becomes unpredictable due to the aggregation, which alters the original Gaussian distribution $\mathcal{N}(0, \sigma^2)$.

Furthermore, due to the dynamic feature of the DHR architecture, the attacked online executor may be scheduled offline, while new executors may be scheduled online, making the situation more complicated. This issue is a subsequent consequence arising from our proposed framework, which aims to tackle privacy concerns in the DHR architecture. It stems from the specific noise distribution within the utilized DP tool and holds significant research value both theoretically and empirically. Therefore, we will conduct more detailed research on this matter in our future work.

Remark 6. *Here, we delve into the implementation and potential implications of the DP-DHR framework in practical scenarios. On one hand, network latency in real-world scenarios could potentially impact the efficiency of the DP-DHR framework. Since the framework lacks the ability to verify the correctness of individual intermediate results, the effectiveness of the hypersphere clustering method (Algorithm 2) relies on the majority principle. Consequently, the decision*

module must await the aggregation of most (if not all) of the k intermediate results to reach a verdict, which may lead to efficiency decreases in the presence of prolonged network latency. Moreover, the heterogeneous nature of the DP-DHR framework further complicates efficiency considerations. Heterogeneously designed executors exhibit varying processing times for identical inputs and experience divergent network latency within the same cyber environment. This phenomenon mirrors challenges encountered in traditional DHR architectures. On the other hand, due to the heterogeneously designed executors, the DHR architecture requires more hardware or software resources compared to the traditional architecture. On this basis, the time complexity of the DP-DHR framework is of the order $\mathcal{O}(k^2)$*, which is brought by the hypersphere clustering method. This might lead to more processing time. However, both the resource costs associated with the DHR architecture and the time complexity introduced by the DP-DHR framework stem from security and privacy concerns, which are deemed acceptable. Furthermore, the validation and corresponding experiments under realistic conditions (including live data stream) will be performed in the future work.*

Based on the comprehensive discussions provided above, we observed that the gap between the output of our proposed DP-DHR framework, which incorporates advanced decision strategies and hypersphere clustering methods, and the output of the T-DHR architecture is relatively small under most circumstances. Specifically, the DP-DHR framework demonstrates greater resilience against attacks designed to manipulate system outputs, regardless of whether they are detected or not. However, in the case of attacks that aim to mislead the system output without being easily detected, the robustness of the DP-DHR architecture diminishes, particularly when the attack rate is high. This behavior is similar to that observed in the T-DHR architecture.

6. Conclusions

In this paper, the focus is on the differential privacy of the DHR architecture, which is a practice of the multi-heterogeneous system designed to enhance cyberspace endogenous safety and security. We propose the DP-DHR framework, which involves adding Gaussian random noise to the DHR architecture, and address technical issues that arise from the injected noise with our advanced decision strategy through the use of hypersphere clustering methods within the decision module. Our theoretical analysis shows that the proposed framework guarantees (ϵ, δ)-differential privacy. Experimental results demonstrate that the utility of the framework is not sacrificed, while its privacy is significantly enhanced. Additionally, the DP-DHR framework, equipped with our proposed advanced decision strategy and hypersphere clustering method, exhibits robustness against some attacks. To the best of our knowledge, this is the first work to incorporate differential privacy with the ESS theory, providing a theoretical guarantee of privacy to the DHR architecture. In future work, we will attempt to design decision strategies that are robust against common modulus attacks and analyze the issue that the outputs generated by attacked executors can change the distribution of the injected Gaussian random noise, thereby affecting privacy properties (as explained in Remark 5).

Author Contributions: Conceptualization, B.J.; methodology, Y.K.; software, Q.Z.; validation, B.J., Y.K. and Y.B.; formal analysis, Y.K.; investigation, Q.Z.; resources, Q.Z.; data curation, Q.Z.; writing—original draft preparation, Y.K.; writing—review and editing, B.J.; visualization, Q.Z.; supervision, Y.B.; project administration, Y.B.; funding acquisition, Y.B. and B.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 62176264 and Shuangchuang Program of Jiangsu Province grant number JSSCBC20221657. The APC was funded by National Natural Science Foundation of China grant number 62176264.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

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

Abbreviations

The following abbreviations are used in this manuscript:

IDS	Intrusion Detection System
DP	Differential Privacy
DHR	Dynamic Heterogeneous Redundant
DRS	Dissimilar Redundant Structure
IoT	Internet of Things
ESS	Endogenous Safety and Security

References

1. Huang, J.; Luo, Y.; Fu, Q.; Chen, Y.; Wang, C.; Song, L. Generic attacks on small-state stream cipher constructions in the multi-user setting. *Cybersecurity* **2023**, *6*, 53. [CrossRef]
2. Ghiasi, M.; Niknam, T.; Wang, Z.; Mehrandezh, M.; Dehghani, M.; Ghadimi, N. A comprehensive review of cyber-attacks and defense mechanisms for improving security in smart grid energy systems: Past, present and future. *Electr. Power Syst. Res.* **2023**, *215* , 108975. [CrossRef]
3. Ahmetoglu, H.; Das, R. A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions. *Internet Things* **2022**, *20* , 100615. [CrossRef]
4. Duo, W.; Zhou, M.; Abusorrah, A. A Survey of Cyber Attacks on Cyber Physical Systems: Recent Advances and Challenges. *IEEE/CAA J. Autom. Sin.* **2022**, *9* , 784–800. [CrossRef]
5. Scala, N.M.; Reilly, A.C.; Goethals, P.L.; Cukier, M. Risk and the Five Hard Problems of Cybersecurity. *Risk Anal.* **2019**, *39*, 2119–2126. [CrossRef]
6. Heino, J.; Hakkala, A.; Virtanen, S. Study of methods for endpoint aware inspection in a next generation firewall. *Cybersecurity* **2022**, *5*, 25. [CrossRef]
7. Zalenski, R. Firewall technologies. *IEEE Potentials* **2002**, *21*, 24–29. [CrossRef]
8. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
9. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, 36, 16–24. [CrossRef]
10. Wu, J. Problems and solutions regarding generalized functional safety in cyberspace. *Secur. Saf.* **2022**, *1* , 2022001. [CrossRef]
11. Wu, J. *Introduction to Cyberspace Mimic Defense*; Science Press: Beijing, China, 2017.
12. Wu, J. The Principle of Cyberspace Mimic Defense. In *Cyberspace Mimic Defense: Generalized Robust Control and Endogenous Security*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 371–493.
13. Wu, J. Development paradigms of cyberspace endogenous safety and security. *Sci. China Inf. Sci.* **2022**, *65*, 156301. [CrossRef]
14. Jin, L.; Hu, X.; Lou, Y.; Zhong, Z.; Sun, X.; Wang, H.; Wu, J. Introduction to wireless endogenous security and safety: Problems, attributes, structures and functions. *China Commun.* **2021**, *18* , 88–99. [CrossRef]
15. Ji, X.; Wu, J.; Jin, L.; Huang, K.; Chen, Y.; Sun, X.; You, W.; Huo, S.; Yang, J. Discussion on a new paradigm of endogenous security towards 6G networks. *Front. Inf. Technol. Electron. Eng.* **2022**, *23* , 1421–1450. [CrossRef]
16. Zhiwen, J.; Tao, L.; Aiqun, H. Research on Endogenous Security Methods of Embedded System. In Proceedings of the IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020; pp. 1946–1950.
17. Xin, Y. Protection architecture of endogenous safety and security for industrial control systems. *Secur. Saf.* **2023**, *2*, 2023001. [CrossRef]
18. Guo, J.; Xu, M. ZTESA—A Zero-Trust Endogenous Safety Architecture: Gain the endogenous safety benefit, avoid insider threats. In Proceedings of the International Symposium on Computer Applications and Information Systems, Shenzhen, China, 25–27 February 2022; p. 122500S.
19. Dwork, C.; McSherry, F.; Nissim, K.; Smith, A.D. Calibrating Noise to Sensitivity in Private Data Analysis. In Proceedings of the Theory of Cryptography Conference, New York, NY, USA, 4–7 March 2006; pp. 265–284.
20. Dwork, C.; Roth, A. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* **2014**, *9*, 211–407. [CrossRef]
21. Zhang, K.; Tian, J.; Xiao, H.; Zhao, Y.; Zhao, W.; Chen, J. A Numerical Splitting and Adaptive Privacy Budget-Allocation-Based LDP Mechanism for Privacy Preservation in Blockchain-Powered IoT. *IEEE Internet Things J.* **2023**, *10* , 6733–6741. [CrossRef]
22. Ali, M.; Naeem, F.; Tariq, M.; Kaddoum, G. Federated Learning for Privacy Preservation in Smart Healthcare Systems: A Comprehensive Survey. *IEEE J. Biomed. Health Inform.* **2023**, *27* , 778–789. [CrossRef]
23. Zhao, Y.; Chen, J. A Survey on Differential Privacy for Unstructured Data Content. *ACM Comput. Surv.* **2022**, *54* , 5217–5233. [CrossRef]
24. Wang, Q.; Zhang, Y.; Lu, X.; Wang, Z.; Qin, Z.; Ren, K. Real-Time and Spatio-Temporal Crowd-Sourced Social Network Data Publishing with Differential Privacy. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15* , 591–606. [CrossRef]

25. Chen, R.; Mohammed, N.; Fung, B.C.M.; Desai, B.C.; Xiong, L. Publishing Set-Valued Data via Differential Privacy. *Proc. VLDB Endow.* **2011**, *4* , 1087–1098. [CrossRef]
26. Ren, C.; Yu, H.; Yan, R.; Li, Q.; Xu, Y.; Niyato, D.; Dong, Z.Y. SecFedSA: A Secure Differential Privacy-Based Federated Learning Approach for Smart Cyber-Physical Grid Stability Assessment. *IEEE Internet Things J.* **2023**, *11*, 5578–5588. [CrossRef]
27. Blanco-Justicia, A.; Sánchez, D.; Domingo-Ferrer, J.; Muralidhar, K. A Critical Review on the Use (and Misuse) of Differential Privacy in Machine Learning. *ACM Comput. Surv.* **2022**, *55* , 1–16. [CrossRef]
28. Denisov, S.; McMahan, H.B.; Rush, J.; Smith, A.; Guha Thakurta, A. Improved Differential Privacy for SGD via Optimal Private Linear Operators on Adaptive Streams. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; pp. 5910–5924.
29. Hirche, C.; Rouzé, C.; França, D.S. Quantum Differential Privacy: An Information Theory Perspective. *IEEE Trans. Inf. Theory* **2023**, *69*, 5771–5787. [CrossRef]
30. Du, Y.; Hsieh, M.H.; Liu, T.; You, S.; Tao, D. Quantum Differentially Private Sparse Regression Learning. *IEEE Trans. Inf. Theory* **2022**, *68* , 5217–5233. [CrossRef]
31. Ye, D.; Shen, S.; Zhu, T.; Liu, B.; Zhou, W. One Parameter Defense—Defending Against Data Inference Attacks via Differential Privacy. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17* , 1466–1480. [CrossRef]
32. Jayaraman, B.; Evans, D. Evaluating Differentially Private Machine Learning in Practice. In Proceedings of the 28th USENIX Security Symposium, Santa Clara, CA, USA, 14–16 August 2019; pp. 1895–1912.
33. Carlini, N.; Liu, C.; Erlingsson, Ú.; Kos, J.; Song, D. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In Proceedings of the 28th USENIX Security Symposium, Santa Clara, CA, USA, 14–16 August 2019; pp. 267–284.
34. Backes, M.; Berrang, P.; Humbert, M.; Manoharan, P. Membership Privacy in MicroRNA-based Studies. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 319–330.
35. Wu, J. Cyberspace Endogenous Safety and Security. *Engineering* **2022**, *15* , 179–185. [CrossRef]
36. Dwork, C.; Lynch, N.; Stockmeyer, L. Consensus in the Presence of Partial Synchrony. *J. ACM* **1988**, *35* , 288–323. [CrossRef]
37. Zhong, W.; Wu, W.; An, G.; Ren, J.; Yu, S. Dissimilar Redundancy Structure Design for Carrier Landing Guidance Computer and Reliability Analysis. In *Proceedings of the First Symposium on Aviation Maintenance and Management-Volume II*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 379–385.
38. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27* , 379–423. [CrossRef]
39. Wu, J. *Cyberspace Endogenous Safety and Security*; Science Press: Beijing, China, 2020.
40. MacQueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 21 June 1967; pp. 281–297.
41. Pelleg, D.; Moore, A.W. X-Means: Extending K-Means with Efficient Estimation of the Number of Clusters. In Proceedings of the Seventeenth International Conference on Machine Learning, San Francisco, CA, USA, 29 June–2 July 2000; pp. 727–734.
42. Arthur, D.; Vassilvitskii, S. K-Means++: The Advantages of Careful Seeding. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, USA, 7–9 January 2007; pp. 1027–1035.
43. Luxburg, U. A Tutorial on Spectral Clustering. *Stat. Comput.* **2007**, *17* , 395–416. [CrossRef]
44. Khan, K.; Rehman, S.U.; Aziz, K.; Fong, S.; Sarasvady, S. DBSCAN: Past, present and future. In Proceedings of the Fifth International Conference on the Applications of Digital Information and Web Technologies, Hanoi, Vietnam, 4–5 December 2014; pp. 232–238.
45. Xu, J.; Ren, Y.; Tang, H.; Pu, X.; Zhu, X.; Zeng, M.; He, L. Multi-VAE: Learning Disentangled View-Common and View-Peculiar Visual Representations for Multi-View Clustering. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021; pp. 9234–9243.
46. Caciularu, A.; Goldberger, J. An entangled mixture of variational autoencoders approach to deep clustering. *Neurocomputing* **2023**, *529*, 182–189. [CrossRef]
47. Tsitsulin, A.; Palowitch, J.; Perozzi, B.; Müller, E. Graph Clustering with Graph Neural Networks. *J. Mach. Learn. Res.* **2023**, *24*, 1–21.
48. Li, Y.; Liu, Y.; Li, B.; Wang, W.; Liu, N. Towards practical differential privacy in data analysis: Understanding the effect of epsilon on utility in private ERM. *Comput. Secur.* **2023**, *128*, 103147. [CrossRef]
49. Seeman, J.; Susser, D. Between Privacy and Utility: On Differential Privacy in Theory and Practice. *ACM J. Responsibale Comput.* **2024**, *1*, 1–18. [CrossRef]
50. Slavković, A.; Seeman, J. Statistical Data Privacy: A Song of Privacy and Utility. *Annu. Rev. Stat. Its Appl.* **2023**, *10*, 189–218. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.