# Adaptive Cross-Platform Web Crawling System Design via Deep Reinforcement Learning and Privacy Protection

**Weipeng Zeng**[1,*]

[1]Guangzhou Public Security Bureau, Guangzhou 510282, China
Corresponding author: Weipeng Zeng (e-mail: weipeng_zeng@163.com).

DOI: https://doi.org/10.63619/ijai4s.v1i2.001

Manuscript received March 3, 2025; revised March 31, 2025; published April 12, 2025.

**Abstract:** Modern web and mobile platforms increasingly deploy complex anti-crawling mechanisms and enforce strict privacy regulations, making large-scale, compliant data acquisition a persistent challenge. In this paper, we propose a novel cross-platform adaptive web crawling framework that integrates deep reinforcement learning (DRL), federated learning (FL), and local differential privacy (LDP) to address the dual demands of operational efficiency and legal compliance. We formulate the crawling process as a Markov Decision Process (MDP) and leverage a PPO-based policy to enable dynamic decision-making under adversarial conditions, including CAPTCHA triggers, tokenized APIs, and platform switching. The system adopts a privacy-by-design architecture: federated training avoids raw data exposure, LDP ensures local feature desensitization, and blockchain-based audit logging provides immutable, transparent behavior tracking. Extensive experiments on real-world platforms—ranging from e-commerce sites to mobile social applications—demonstrate that our framework achieves superior success rates, adaptive behavior, and compliance scores compared to traditional, heuristic, and non-private baselines. The proposed system offers a practical and legally conscious solution for next-generation web crawling in dynamic, regulated ecosystems.

**Keywords:** Web Crawling, Deep Reinforcement Learning, Federated Learning, Differential Privacy, Cross-Platform Systems

## 1. Introduction

### *1.1. Background and Motivation*

The exponential growth of data-driven technologies has significantly increased the reliance on large-scale web and mobile application (app) data for research, industrial, and commercial purposes. Applications such as market analysis, public opinion monitoring, recommendation systems, and artificial intelligence (AI) training pipelines require continuous, high-quality, and structured data acquisition from heterogeneous digital environments [1], [2], [3]. Consequently, web crawlers and data extraction tools have become indispensable components in modern information systems.

However, traditional crawling systems are increasingly challenged by the rapid evolution of *anti-crawling techniques*. Websites and mobile applications now adopt a range of defensive strategies, including JavaScript obfuscation, dynamic DOM rendering, CAPTCHA challenges (e.g., slider or image selection), TLS certificate pinning, and device fingerprinting [4], [5], [6]. These mechanisms are intentionally designed to hinder automated access, leading to significant drops in crawling efficiency, increased engineering complexity, and higher maintenance costs. Moreover, the diversity of platforms — from HTML-based web frontends to encrypted API endpoints within native apps — introduces substantial *cross-platform heterogeneity*, further complicating crawler design and adaptation.

In parallel, *privacy and data protection regulations* have become increasingly stringent across jurisdictions. Laws such as the General Data Protection Regulation (GDPR) in the European Union [7], [8], [9] and the Cybersecurity Law in China [10], [11] impose strict requirements on personal data handling, collection transparency, and user consent. As a result, web crawlers not only face technical obstacles but must also navigate complex legal and ethical landscapes, ensuring that data acquisition does not violate user privacy or organizational compliance requirements [12], [13], [14]. Traditional scraping solutions, which often rely on centralized data storage and post hoc sanitization, are poorly equipped to meet these new legal expectations.

These converging technical and legal trends underscore the urgent need for a new generation of *intelligent, adaptive, and privacy-aware crawling systems*. Such systems must be capable of perceiving and reacting to diverse anti-crawling mechanisms in real-time, seamlessly operate across different platforms (web, app, API), and rigorously enforce privacy protection and auditability standards. This paper addresses these challenges through the integration of *deep reinforcement learning (DRL)* for adaptive crawling decision-making [15], [16] and *privacy-preserving technologies* such as federated learning and local differential privacy [17], [18].

### *1.2. Problem Statement*

Despite the critical role of web and app crawlers in modern data ecosystems, existing solutions are increasingly inadequate in meeting the dual requirements of technical robustness and regulatory compliance. Traditional crawlers typically adopt static rules or script-driven heuristics to navigate target platforms. These methods often fail under dynamic, evolving anti-crawling defenses, such as obfuscated JavaScript logic, dynamic content rendering, advanced CAPTCHA mechanisms, and encrypted mobile APIs [4], [19]. Such static approaches are not only fragile but also require continuous manual updates, making them unsuitable for real-world large-scale deployment.

Furthermore, most existing crawling systems are tailored to a single platform, typically the Web. App-based data acquisition remains underexplored due to its higher technical barriers, including secure communication channels, mobile encryption, and dynamic API endpoints. The lack of a unified cross-platform framework results in redundant engineering efforts, limited reusability, and inconsistent data coverage across platforms.

Simultaneously, increasing legal scrutiny over data privacy introduces another layer of complexity. Few crawler systems embed privacy-preserving mechanisms into their data collection pipelines. As a result, data acquisition practices may inadvertently violate privacy laws such as GDPR or China's Cybersecurity Law [7], [10], [20]. For instance, centralized collection of user-generated content without anonymization or user consent can pose serious ethical and legal risks [12], [21], [22], [23].

The core problem, therefore, lies in the absence of a generalizable, intelligent, and legally compliant crawling framework that can adapt to anti-crawling strategies across heterogeneous platforms while simultaneously preserving user privacy. This challenge is further compounded by the lack of integration between state-of-the-art techniques in reinforcement learning, cross-platform system design, and privacy-preserving machine learning.

To bridge this gap, we aim to design an adaptive cross-platform web crawling system driven by deep reinforcement learning (DRL), capable of autonomously adjusting its crawling policy in response to environmental feedback. Simultaneously, we integrate privacy-preserving techniques — including federated learning and local differential privacy — to ensure compliance with legal standards during data collection and storage [17], [24], [25].

### *1.3. Contributions*

In this paper, we present a novel framework that addresses the intertwined challenges of adaptive web/app crawling, platform heterogeneity, and legal compliance in data acquisition systems. Our main contributions can be summarized as follows:

**A deep reinforcement learning (DRL)-based adaptive crawling system.** We propose a DRL-powered decision-making module that dynamically adjusts crawling strategies in response to real-time feedback from target environments. By formulating the crawler's behavior as a Markov decision process (MDP), our

system learns to navigate and bypass complex anti-crawling mechanisms—such as CAPTCHAs, encrypted JavaScript, and dynamic web structures—without manual rule engineering. This approach supports both web and app platforms, making it robust and generalizable across multiple domains [15], [26], [27].

**Integration of privacy-preserving techniques for legal compliance.** To ensure lawful data acquisition under increasingly strict privacy regulations, we embed two key privacy-preserving technologies into the system pipeline. First, we apply *federated learning* to support decentralized model training, which prevents the transfer of raw data to centralized servers. Second, we incorporate *local differential privacy (LDP)* to perturb sensitive user information at the data source before any transmission or processing, thereby reducing legal and ethical risk exposure [17], [28], [29], [30].

**A cross-platform, low-intrusion architectural design.** We design a lightweight and modular crawling architecture that unifies heterogeneous platform support while minimizing system invasiveness. For web crawling, we enhance headless browser-based rendering with automated JavaScript analysis; for app crawling, we develop a low-intrusion hooking and RPC-based communication framework that avoids reverse engineering and static binary modification [31], [32]. Our unified scheduler, trained via DRL, efficiently balances crawling success rate, resource usage, and privacy risk across platforms.

Collectively, these contributions constitute a significant step toward building legally compliant, technically resilient, and cross-platform adaptive web/app crawlers. They also provide a foundation for future research at the intersection of intelligent systems, cybersecurity, and privacy-preserving computation.

## 2. Related Work

### 2.1. Traditional Web Crawling and Anti-Crawling Mechanisms

Web crawling has long served as a foundational technique for automated information acquisition from the Internet. Classical web crawlers, such as Googlebot and early open-source tools like Scrapy and Heritrix, rely on deterministic URL traversal, HTML parsing, and rule-based filtering to extract content from websites. These systems typically operate under a breadth-first or depth-first exploration paradigm and are optimized for static page structures with predictable hyperlinks [33], [34].

However, as the commercial value of web content increased, website administrators began deploying a range of *anti-crawling mechanisms* to prevent unauthorized or excessive data scraping. Early techniques included IP rate-limiting, user-agent filtering, and cookie-based session verification. More recent approaches leverage sophisticated technologies, such as:

JavaScript obfuscation and dynamic content rendering, where key content or links are only revealed after client-side execution, rendering traditional HTML parsers ineffective [1], [35]; CAPTCHA challenges, including image-based slider puzzles, text distortion, and object recognition tasks, which aim to differentiate between human and automated agents [36], [37]; Device fingerprinting and behavioral analytics, which collect mouse movements, screen size, or rendering speed to detect bot-like behavior [38], [39], [40]; TLS certificate pinning and encrypted API endpoints, especially common in mobile apps, to enforce secure communication and prevent traffic interception [41], [42].

In response, researchers and practitioners have developed a variety of countermeasures. These include headless browsers (e.g., Puppeteer, Selenium), script emulators, and machine learning-based CAPTCHA solvers [43], [44]. Despite these advances, the highly dynamic and adversarial nature of web environments makes static crawlers brittle and costly to maintain over time. Moreover, most existing frameworks are designed for web crawling only, with limited or no support for app-based environments, thereby lacking true cross-platform capability.

To address these limitations, recent trends point toward adaptive and learning-based crawling frameworks that can generalize across domains and dynamically adapt to new anti-crawling strategies. Our work builds on this vision by integrating deep reinforcement learning and privacy-preserving computation into a unified cross-platform system.

### 2.2. DRL Applications in Navigation and Web Environments

Deep reinforcement learning (DRL) has achieved remarkable success in a variety of sequential decision-making tasks, ranging from robotic control and game playing to autonomous navigation in complex

environments [15], [45], [46]. The ability of DRL agents to learn optimal policies through interactions with dynamic environments makes them well-suited for problems where static rule-based methods fail to generalize.

In recent years, DRL has been explored in the context of web navigation, where the agent learns to interact with dynamic websites by issuing sequences of actions, such as clicking, scrolling, or filling out forms [47], [48], [49]. Such frameworks model the browsing process as a Markov Decision Process (MDP), in which the crawler must determine the most effective sequence of actions to reach a target state (e.g., locate a piece of data or bypass an obstacle). These approaches often combine visual, structural, and semantic features extracted from the Document Object Model (DOM) to represent the web state.

Other research efforts apply DRL to web data extraction under adversarial conditions, including anti-crawling defenses. For instance, DRL-based agents have been proposed to learn adaptive crawling policies that minimize detection while maximizing data collection success rates [50], [51], [52]. Similarly, DRL has been used to emulate human-like behavior on websites to evade bot detection algorithms [53], [54].

Despite these promising results, most existing DRL-based web agents are confined to browser environments and lack support for mobile applications (apps), where interaction mechanisms, UI structures, and access protocols differ significantly. Additionally, current methods generally ignore privacy and compliance constraints, treating data collection as a pure optimization problem without considering regulatory obligations. Our proposed system builds on these foundations by extending DRL-driven adaptivity to both Web and App platforms, while simultaneously embedding privacy-preserving components into the agent's policy and environment interaction framework.

### *2.3. Privacy-Preserving Data Collection*

With the rise of global data privacy regulations such as the General Data Protection Regulation (GDPR) and China's Cybersecurity Law, the design of data collection systems must now incorporate privacy-preserving mechanisms as a fundamental requirement rather than an afterthought [7], [10], [55], [56]. In the context of web and app crawling, this challenge is particularly acute, as crawlers may inadvertently capture sensitive user information without explicit consent, resulting in both ethical concerns and legal liabilities [12], [57].

To address these challenges, recent research has explored the integration of privacy-preserving machine learning (PPML) techniques into the data collection pipeline. One of the most widely adopted frameworks is federated learning (FL) [17], [58], [59], which enables collaborative model training across distributed clients without transferring raw data to a central server. This paradigm significantly reduces the risk of data leakage while still allowing systems to learn from decentralized interactions. In the context of web crawling, FL can be used to aggregate crawling policies, update anti-detection strategies, or personalize behaviors across platforms without violating data locality constraints.

Complementary to FL, local differential privacy (LDP) offers a formal privacy guarantee at the data source [60], [28], [61], [62]. By adding calibrated noise to user-generated data before collection or transmission, LDP ensures that any single data record has a provably minimal influence on the output, thereby limiting the risk of re-identification. This approach is particularly useful for content-sensitive crawling tasks, where exact data fidelity may be less critical than privacy preservation.

In addition to computational techniques, privacy auditing and transparency have also gained attention. Methods such as blockchain-based logging and zero-knowledge proof-based access control offer cryptographically verifiable mechanisms for tracking data provenance and ensuring that collection activities adhere to predefined legal boundaries [63], [64]. While these techniques are still nascent in the context of crawling, they represent promising directions for improving trust and compliance.

Despite these advances, few crawling systems currently integrate these privacy-preserving components into a coherent architectural design. Our proposed system bridges this gap by embedding federated policy learning, LDP-based data perturbation, and blockchain-enabled auditing into a unified, cross-platform crawling framework.

### *2.4. Cross-Platform Crawling (Web and App)*

Traditional crawling systems have been primarily designed for web-based environments, where the Document Object Model (DOM) and hyperlink structures offer well-defined and consistent entry points for

data extraction. However, as mobile applications (apps) have become the dominant interface for accessing digital services, a significant portion of valuable user-facing content is now hidden behind mobile-exclusive frontends and encrypted APIs [41], [65]. Consequently, modern crawlers must evolve to support cross-platform data acquisition, encompassing both Web and App ecosystems.

On the web side, a common strategy for handling JavaScript-heavy or dynamic content is to utilize headless browsers (e.g., Puppeteer, Playwright, Selenium), which simulate real user interaction and allow rendering of client-side scripts [1], [66]. More advanced systems incorporate JavaScript emulation and instrumentation through abstract syntax tree (AST) analysis and runtime hooking, enabling the extraction of encrypted or obfuscated logic such as token generation, anti-CSRF protections, or challenge-response authentication [67], [68].

In contrast, mobile app crawling presents a different set of challenges. Native apps often rely on compiled binaries, encrypted communication, and proprietary API protocols that are not easily observable from the application layer. To extract meaningful data, researchers have employed techniques such as:

App reverse engineering, using tools like JADX or Apktool to decompile Android binaries and statically analyze API endpoints and logic flows [69], [70]; Dynamic instrumentation, particularly using Frida or Xposed, to hook runtime functions and intercept API calls during app execution without modifying the binary [71], [72]; Man-in-the-middle (MitM) proxying, using tools like MitmProxy to capture and analyze encrypted traffic, although increasingly hindered by TLS certificate pinning and DNS over HTTPS (DoH).

While effective, these approaches often require significant manual effort, pose compatibility risks, and may be considered intrusive or legally ambiguous in some jurisdictions. Furthermore, the separation between Web and App crawling frameworks leads to redundant implementation, poor generalization, and suboptimal policy transfer.

To mitigate these issues, recent works have begun to explore unified cross-platform crawling frameworks that abstract away platform-specific details via modular architectures and shared control strategies. Our proposed system extends this line of research by introducing a DRL-driven cross-platform scheduler combined with low-intrusion data interception mechanisms for both Web and App environments, enabling efficient and legally compliant data collection across digital ecosystems.

### *2.5. Summary and Limitations of Existing Work*

In summary, existing research has made significant progress in various dimensions of web crawling: from early rule-based systems and anti-crawling countermeasures [33], [4], to learning-based web navigation using deep reinforcement learning [47], [50], and the recent incorporation of privacy-preserving paradigms such as federated learning and differential privacy [17], [28]. Moreover, substantial efforts have been made to develop reverse engineering and dynamic hooking tools for app-level data extraction [41], [71].

However, several critical limitations remain:

**(1) Lack of cross-platform generalization.** Most existing systems are tailored to either Web or App platforms, with minimal reusability across environments. The absence of a unified crawling architecture limits the scalability and adaptability of current solutions in real-world, heterogeneous digital ecosystems.

**(2) Insufficient adaptivity to complex anti-crawling mechanisms.** While some works adopt DRL for web interaction, few have demonstrated robust performance under adversarial conditions such as evolving CAPTCHA schemes, dynamic JavaScript obfuscation, and TLS certificate pinning. Moreover, existing DRL-based approaches often operate in simulation or sandboxed environments with limited generalization capacity.

**(3) Neglect of privacy and compliance constraints.** A large portion of prior work treats web crawling as a purely technical problem, without considering legal and ethical boundaries. This oversight exposes data collection systems to substantial regulatory risks, especially in jurisdictions enforcing GDPR or similar data protection laws [7], [12].

**(4) High implementation complexity and maintenance cost.** Techniques such as app decompilation or deep packet inspection, while powerful, are intrusive and require frequent manual updates to keep pace with platform changes. This reduces their feasibility for long-term deployment in production environments.

These limitations motivate the need for a novel, unified, and adaptive cross-platform crawling framework that combines DRL-based policy learning, privacy-preserving data collection, and low-intrusion design

principles. Our proposed system addresses this gap by tightly integrating intelligent scheduling, platform-aware crawling logic, and legal compliance auditing into a single, scalable architecture.

## 3. System Overview

### 3.1. Architecture Design

The proposed system is designed as a modular and scalable framework that supports intelligent, privacy-preserving, and cross-platform web crawling. As illustrated in Figure 1, the overall architecture consists of four core components: (1) the **Adaptive Scheduler**, (2) the **Crawling Agent**, (3) the **Privacy Protection Layer**, and (4) the **Audit and Compliance Module**. Each component addresses the key challenges discussed in Section 2, including platform heterogeneity, anti-crawling dynamics, and regulatory constraints.

- **Adaptive Scheduler:** At the heart of the system lies a DRL-based Adaptive Scheduler, which formulates the crawling process as a sequential decision-making task. The scheduler observes the current environment state (e.g., platform type, response delay, anti-crawling signal), and selects optimal crawling actions—such as whether to switch platform, invoke CAPTCHA solver, or adjust access frequency. The policy is trained using a Proximal Policy Optimization (PPO) algorithm to balance success rate, system load, and privacy risk [15], [50].
- **Crawling Agent:** Responsible for executing platform-specific data acquisition tasks. It contains two submodules: *(1)* A Web Crawler based on a headless browser (e.g., Puppeteer) with a JavaScript emulator and DOM parser; and *(2)* An App Crawler utilizing runtime hooking (e.g., Frida or Xposed) and traffic interception (e.g., MitmProxy) to capture API data from Android/iOS apps. The agent reports structured data and feedback signals to the scheduler for policy refinement.
- **Privacy Protection Layer:** Ensures privacy-preserving data collection via two mechanisms. First, sensitive fields are obfuscated using local differential privacy (LDP) techniques before transmission [28]. Second, the DRL policy is updated through federated learning (FL), enabling model training across edge clients without raw data exchange [17].
- **Audit and Compliance Module:** All crawling actions and decisions are logged using a blockchain-based immutable ledger [63]. Metadata includes timestamps, access intents, endpoints, and anonymized device identifiers. Smart contracts enforce policy limits (e.g., query rate) and enable external auditability.

Together, these components form an integrated architecture that supports intelligent, scalable, and legally-compliant data acquisition across diverse digital environments.

### 3.2. Cross-Platform Considerations

To achieve scalable and efficient data acquisition across heterogeneous environments, the proposed system incorporates platform-specific strategies under a unified control framework. Specifically, we differentiate our design to accommodate both **web-based platforms**, which rely heavily on HTML and JavaScript, and **mobile applications**, which communicate primarily through proprietary APIs and native interfaces.

**1. Web Environment.** Modern websites often employ dynamic content loading through JavaScript, AJAX, and third-party scripts. To handle such complexity, our system integrates a headless browser engine (e.g., Chromium-based Puppeteer) capable of executing JavaScript in a sandboxed environment. The rendered Document Object Model (DOM) is parsed using semantic-aware extractors, and obfuscated logic (e.g., token generation scripts) is intercepted using an embedded JavaScript emulator with AST-level analysis [67]. This allows for precise reconstruction of client-side rendering and interaction behaviors.

**2. App Environment.** Mobile applications introduce additional challenges, including native code execution, encrypted communication, and a lack of standardized markup. To address this, we incorporate a runtime instrumentation layer using tools such as `Frida` and `Xposed`, which allow dynamic hooking of Android or iOS methods without modifying the app binaries [71]. For network-level data acquisition, we employ a MitM-based proxying mechanism (e.g., `MitmProxy`) to capture API responses, supplemented by TLS interception techniques where certificate pinning is absent or bypassable [41]. Custom parsers translate JSON or Protobuf payloads into structured records for downstream analysis.

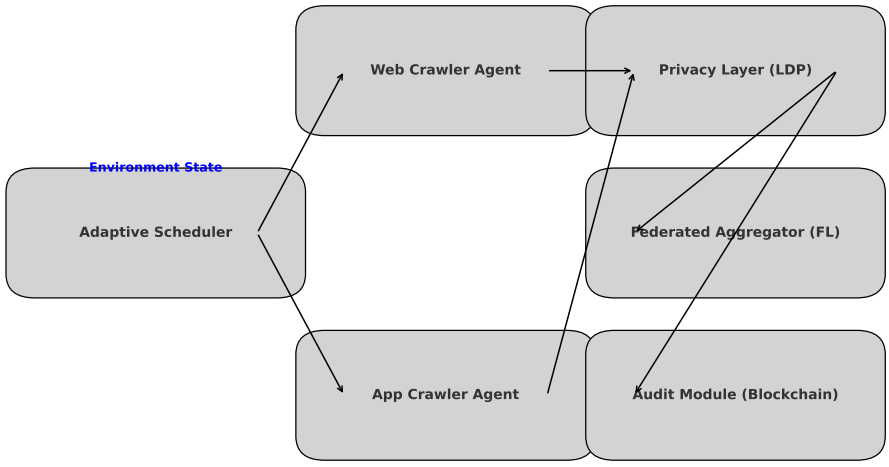


Fig. 1. System Architecture. The adaptive scheduler coordinates cross-platform crawling via Web and App agents, enforces privacy via LDP and FL, and logs all actions to an auditable blockchain ledger.

**3. Unified Abstraction.** Despite the technical disparity between web and app environments, our architecture abstracts their data retrieval logic into a shared schema consisting of `<target, method, payload, response>`. This abstraction facilitates policy transfer, logging, and federated learning by allowing the scheduler to operate agnostically over platform-specific crawling agents. The combination of environment-aware optimization and schema-level unification allows the system to achieve consistent, high-quality data extraction across platforms while maintaining a low engineering footprint.

### *3.3. Threat Model and Compliance Assumptions*

To ensure secure and compliant operation, our system is designed under a clearly defined threat model and legal compliance framework. This section outlines the types of adversaries considered and the assumptions made regarding platform behavior and regulatory obligations.

**1. Threat Model.** We assume the presence of two primary adversarial entities:

- **Anti-crawling mechanisms:** These are defensive techniques implemented by target platforms (websites or apps) to prevent unauthorized data extraction. They include IP blocking, JavaScript-based obfuscation, CAPTCHA challenges, session tracking, and API rate-limiting. We consider these mechanisms non-malicious but adversarial in intent, aiming to detect and disable automated agents.
- **Passive observers and network attackers:** These include malicious intermediaries capable of eavesdropping on crawler communication (e.g., unsecured Wi-Fi, proxy interception). Although our system does not perform sensitive user input, we adopt encryption and LDP techniques to mitigate exposure of collected data during transmission or storage.

We do not consider stronger attack models involving crawler compromise, backdoor insertion, or OS-level rootkits, which are beyond the scope of this work.

**2. Compliance Assumptions.** Our system is designed with privacy legislation in mind, particularly:

- **GDPR and Similar Regulations:** We assume that any user-generated or personal data (e.g., comments, profiles, device identifiers) is either publicly available or anonymized via local differential privacy [60], [28]. No raw personal identifiers are stored or transmitted.

- **Legitimate Interest or Research Exemption:** We assume that data acquisition is conducted for legitimate scientific or service-driven purposes under allowable exemptions defined in GDPR Article 6(1)(f) and equivalent clauses in regional laws [7].
- **Transparent Logging and Accountability:** To ensure traceability and auditability, all system interactions are logged using immutable blockchain mechanisms [63], enabling post-hoc review and enforcement of internal crawling policies.

Overall, the system maintains a privacy-by-design philosophy, minimizing the data it collects, decentralizing learning processes, and enforcing access controls and monitoring to remain compliant across jurisdictions.

**3. Threat Model and Compliance Assumptions**

To ensure both operational security and regulatory compliance, our system is developed under a clearly defined threat model and a set of legal and ethical assumptions. These constraints guide the design of each module, from data acquisition to logging and storage, and reflect both technical realism and legal responsibility.

**4. Threat Model.** We consider two primary categories of adversaries:

1) **Anti-crawling defenses** implemented by target platforms, including:
   - IP throttling and blocking,
   - Session-based behavioral detection,
   - JavaScript obfuscation and dynamic token generation,
   - CAPTCHA mechanisms (e.g., slider, image-based),
   - TLS certificate pinning to prevent proxy-based traffic inspection.

   These mechanisms are adversarial in function but non-malicious in origin. Our system is designed to respond to such defenses adaptively, without attempting to subvert or exploit vulnerabilities in the host platform.
2) **Passive external observers**, such as attackers monitoring unsecured networks or intermediaries between crawler and target. While we assume the crawler environment is not compromised, we adopt defense-in-depth strategies such as encrypted transmission, secure containerized execution, and privacy-preserving preprocessing (e.g., via LDP) to mitigate data leakage risks.

We explicitly exclude active, high-power adversaries such as OS-level backdoors, supply chain attacks, or privilege escalation within the crawler node itself.

**5. Compliance Assumptions.** Our design is grounded in privacy regulations such as the European Union's GDPR and China's Cybersecurity Law. Specifically, we assume:

- **Public data scope:** Crawling operations are restricted to publicly accessible content. When user-generated or personalized data is encountered, local differential privacy (LDP) mechanisms are applied before any transmission or logging [28], [60].
- **Purpose legitimacy:** The system operates under the assumption of legal basis via "legitimate interest" or "research exemption," per Article 6(1)(f) of GDPR and similar clauses in other jurisdictions [7].
- **Traceability and transparency:** Every crawling action, including request headers, access timing, and decision reasoning, is logged to an immutable blockchain-based ledger [63], enabling external audits and legal verification of compliant behavior.

By integrating these threat models and assumptions into both design and deployment, the proposed system supports robust, ethical, and auditable data acquisition suitable for modern regulatory environments.

## 4. DRL-Based Adaptive Crawling Strategy

### *4.1. Problem Formulation as a Reinforcement Learning Task*

To enable adaptive and intelligent crawling across heterogeneous platforms, we formulate the crawling policy optimization as a reinforcement learning (RL) problem. The crawler operates as an RL agent that

sequentially interacts with its environment—comprising websites or mobile applications—and learns a policy that maximizes long-term rewards under platform constraints and privacy-preserving objectives.

The problem is modeled as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where:

- $\mathcal{S}$ (**State space**): Each state $s_t \in \mathcal{S}$ represents the current crawling context, including platform type (Web/App), recent response codes, content entropy, anti-crawling indicators (e.g., CAPTCHA flag, JS execution time), current access frequency, and previous action history. States may also include privacy budget statistics and crawling session identifiers.
- $\mathcal{A}$ (**Action space**): The agent chooses an action $a_t \in \mathcal{A}$ at each step, including operations such as:
  - `SwitchPlatform(Web ↔ App)`,
  - `InvokeCaptchaSolver()`,
  - `AdjustRateLimit(`$\delta$`)`,
  - `ChangeProxy()`,
  - `PauseOrTerminateSession()`.

  These actions allow the agent to adaptively navigate across platforms and bypass detection strategies without manual intervention.
- $\mathcal{P}$ (**Transition function**): The environment evolves stochastically based on both internal dynamics (e.g., platform backend behavior) and crawler actions. For example, a failed CAPTCHA solving may transition the state to a blocked IP, while a rate-limited request may yield a temporary suspension signal.
- $\mathcal{R}$ (**Reward function**): The reward $r_t$ is computed based on multiple objectives:

$$r_t = \lambda_1 \cdot \text{SuccessRate} - \lambda_2 \cdot \text{DetectionPenalty} - \lambda_3 \cdot \text{PrivacyRisk} - \lambda_4 \cdot \text{LatencyCost} \tag{1}$$

  where $\lambda_i$ are user-defined weights. A high reward is issued for successfully retrieving high-value content with low latency, no privacy violation, and minimal detection risk.
- $\gamma$ (**Discount factor**): Governs the agent's preference for short-term versus long-term gains. A higher $\gamma$ encourages strategic crawling behaviors over immediate but potentially risky rewards.

This formalization allows us to apply modern deep reinforcement learning algorithms—such as Proximal Policy Optimization (PPO) or Soft Actor-Critic (SAC)—to train a policy network $\pi_\theta(a_t|s_t)$ that governs crawling decisions dynamically and robustly [15], [50].

### *4.2. Model Architecture*

To learn an effective crawling policy across dynamic and adversarial web environments, we adopt a modular deep reinforcement learning (DRL) architecture, combining state encoding, policy learning, and value estimation within a unified actor–critic framework. Specifically, we utilize the **Proximal Policy Optimization (PPO)** algorithm [73], a stable and sample-efficient on-policy DRL method widely used in high-dimensional control tasks.

**1. State Encoder.** Given the heterogeneous and sequential nature of crawling states, we employ a hybrid encoder structure:

- **Categorical inputs** (e.g., platform type, HTTP status codes) are embedded via learned token embeddings.
- **Numerical features** (e.g., access frequency, JS execution latency, reward history) are projected via linear layers.
- **Temporal or interaction features** (e.g., response sequences, CAPTCHA events) are encoded using a lightweight **Transformer encoder** [74] to capture long-range correlations in action-feedback history.

The concatenated representation $z_t$ is then fed into the policy and value branches.

**2. Policy and Value Heads.** We implement a standard actor–critic setup, where:

- The **policy head** $\pi_\theta(a_t|s_t)$ is a multi-layer perceptron (MLP) that outputs a categorical distribution over discrete actions such as platform switching, CAPTCHA solving, and proxy rotation.

- The **value head** $V_\phi(s_t)$ estimates the expected return of the current state under policy $\pi_\theta$, assisting in advantage estimation and policy gradient updates.

Both heads are optimized using PPO's clipped surrogate loss function with entropy regularization to balance exploration and exploitation:

$$\mathcal{L}^{\text{PPO}} = \mathbb{E}_t \left[ \min \left( r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, and $\hat{A}_t$ is the estimated advantage function.

**3. Training Details.** The model is trained end-to-end using trajectories collected by the crawling agents under simulated or real-world environments. We utilize generalized advantage estimation (GAE), Adam optimizer, batch normalization, and early stopping to prevent overfitting and ensure stable convergence. Model checkpoints are periodically synchronized under a federated learning framework (see Section 3.1). This architecture ensures the crawler can generalize across diverse scenarios and respond robustly to emerging anti-crawling patterns in both web and app environments.

### *4.3. Training Strategy*

Training an effective and generalizable crawling policy in dynamic, adversarial environments poses several challenges, including exploration–exploitation trade-offs, sparse feedback signals, and shifting platform behaviors. To address these issues, our system adopts a hybrid training strategy consisting of **offline pretraining**, **online policy adaptation**, and **reward shaping**.

**1. Offline Pretraining.** We first pretrain the agent using a large-scale, multi-domain dataset consisting of historical crawling logs collected from web and app platforms. These logs are transformed into state–action–reward trajectories that approximate the real environment dynamics. The policy and value networks are initialized using behavioral cloning on expert-like trajectories and then fine-tuned using offline reinforcement learning (RL) methods such as Batch-Constrained Q-Learning (BCQ) or conservative Q-learning (CQL) to avoid distributional shift. Offline pretraining accelerates convergence, provides safe initialization, and mitigates the cold-start problem common in live deployment scenarios.

**2. Online Policy Adaptation.** After deployment, the policy is further refined via online interaction with real-world targets. We employ **Proximal Policy Optimization (PPO)** in conjunction with a federated averaging scheme, allowing multiple edge agents to collect rollouts independently and update the global policy model without sharing raw data (see Section 3.1). A replay buffer is maintained locally to stabilize updates and avoid catastrophic forgetting of rare anti-crawling events.

Online updates enable the agent to adapt to platform drift, newly introduced CAPTCHA mechanisms, and evolving detection heuristics in a sample-efficient and privacy-preserving manner.

**3. Reward Shaping.** To guide the learning process, we design a multi-objective reward function that incorporates success, cost, and compliance considerations:

$$r_t = \alpha \cdot \mathbb{1}_{\text{Success}} - \beta \cdot \mathbb{1}_{\text{Blocked}} - \gamma \cdot \text{Latency}_t - \delta \cdot \text{PrivacyRisk}_t \tag{3}$$

where $\alpha$, $\beta$, $\gamma$, and $\delta$ are tunable hyperparameters that balance between high-value content retrieval and low detection or legal risk. The privacy risk term is derived from the cumulative local differential privacy (LDP) budget usage and audit flags (see Section 3.3).

Reward shaping ensures that the agent not only maximizes task performance but also aligns with system-level constraints such as responsiveness, stealthiness, and legal compliance.

### *4.4. Adaptive Path Planning and Anti-Detection Response*

One of the core capabilities of our proposed system is its ability to dynamically adapt crawling strategies in response to environment feedback and evolving anti-crawling defenses. This is achieved through the integration of reinforcement learning-based policy control, multi-platform observability, and real-time feedback loops that together enable robust path planning and risk-aware behavior adjustment.

**1. Adaptive Path Planning.** At each time step, the crawling agent selects an action based on the current state $s_t$ and its policy $\pi_\theta(a_t|s_t)$, which encodes both immediate rewards and long-term consequences. This enables the agent to perform:

- **Strategic switching** between Web and App platforms based on platform accessibility, success history, and resource availability.
- **Dynamic scheduling** of request intervals and proxy rotations to mimic human-like behavior and reduce request correlation.
- **Route optimization** to prioritize targets that yield higher content utility with lower detection or CAPTCHA probabilities.

By continuously updating its policy through online reinforcement learning, the agent learns to avoid high-risk paths and allocate crawling resources to more favorable sequences of interaction.

**2. Anti-Detection Response.** Modern websites and apps implement sophisticated anti-crawling mechanisms such as JavaScript-based behavior fingerprinting, user-agent validation, session token mutation, and multi-modal CAPTCHA challenges. Our system responds to these through:

- **Behavioral mimicry**, where the policy incorporates historical interaction patterns to approximate human browsing rhythms (e.g., dwell time, scrolling, navigation depth).
- **Conditional CAPTCHA solvers**, which are selectively triggered by the policy when CAPTCHA detection flags are raised. We incorporate pre-trained image classifiers and OCR-based solvers for slider, image click, and reCAPTCHA types.
- **JavaScript logic extraction**, using AST-based code parsing and runtime emulation to decode token generation or validation logic [67].

In addition, the policy is penalized when system logs or audit trails detect abnormal activity patterns, such as high error rates, frequent session drops, or excessive fingerprint changes (see Section 3.3). Through joint optimization of reward, risk, and cost, the system achieves *adaptive stealth*: minimizing its exposure to anti-crawling detection while maintaining crawling throughput and data utility.

## 5. Privacy Protection Mechanisms

### *5.1. Federated Learning for Distributed Data Coordination*

To minimize privacy risks and ensure legal compliance during policy training, our system integrates a **federated learning (FL)** framework that enables collaborative learning across distributed crawler instances without sharing raw data [17].

**1. Edge-Cloud Coordination.** The architecture follows a typical *edge–cloud FL paradigm*, where multiple crawling agents deployed at the edge (e.g., in enterprise environments or regional servers) interact with different web/app platforms and collect local interaction data. Rather than transmitting raw trajectories or log data, each agent locally updates its own copy of the policy network using its private experience buffer. Periodically, the agents send encrypted model gradients or parameter deltas to a *central coordinator*, which performs secure model aggregation (e.g., via Federated Averaging). The global model is then redistributed back to the edge nodes for the next training cycle.

**2. Model Aggregation and Robustness.** To protect against poisoning or manipulation by unreliable clients, we implement:

- **Aggregation filtering**, which discards statistically deviant updates based on cosine similarity or update magnitude.
- **Secure aggregation**, where differential privacy noise is optionally applied before updates are sent, to bound the influence of individual clients.
- **Client sampling**, which selects a randomized subset of edge agents for each round to improve robustness and scalability.

**3. Task Decentralization.** To further preserve data locality and reduce cloud dependency, we introduce a modular task-specific adaptation strategy. Each edge agent fine-tunes its own *adapter layer* or *task head*

based on local platform characteristics (e.g., specific CAPTCHA types, rate-limiting logic). This results in a hybrid architecture, where the shared backbone is learned globally, but task-specific modules are kept private and personalized. This design significantly improves platform-specific generalization while maintaining system-wide consistency. By decoupling local knowledge from global synchronization, our framework satisfies key privacy and security principles in line with GDPR and other regional laws [60].

### *5.2. Local Differential Privacy for Data Desensitization*

While federated learning ensures that raw data remains decentralized, it does not inherently protect the sensitive information contained within locally processed records. To further strengthen privacy guarantees, we integrate a **Local Differential Privacy (LDP)** mechanism into the edge crawling agents [60], [28].

**1. Feature Perturbation.** Before transmitting any metadata (e.g., extracted content fields, behavioral logs, success statistics) to the central coordinator or storing them in local logs, each agent applies randomized perturbation mechanisms to sensitive fields. Specifically, we employ:

- **Additive Laplace noise** for continuous-valued features such as latency, response time, or session duration.
- **Randomized response** for categorical or binary fields such as click types, CAPTCHA triggers, or user-agent tags.

These mechanisms ensure that each individual data point satisfies $\epsilon$-local differential privacy, meaning its presence or absence cannot be confidently inferred by any observer—even one with access to model parameters or audit logs.

**2. Privacy Budget Management.** To balance utility and privacy, each agent maintains a local privacy budget $\epsilon$ and a decay function that tracks cumulative privacy loss over time. The system monitors the budget consumption rate and triggers fallback modes when nearing critical thresholds, such as:

- Reducing data sampling frequency,
- Switching to coarser-grained features (e.g., binning latency ranges),
- Temporarily suspending data sharing until budget replenishment.

We adopt a composition-aware mechanism to track budget accumulation across multiple perturbed dimensions and time steps [60], allowing for precise control of long-term privacy exposure.

**3. Implementation Considerations.** All LDP operations are implemented at the edge level and incur minimal computational overhead. Our evaluation (Section 7.2) demonstrates that feature-level perturbation achieves acceptable accuracy–privacy trade-offs, especially when combined with robust federated aggregation. This dual-layer design—federated learning for structural privacy and LDP for record-level obfuscation—ensures that our system meets modern regulatory expectations without compromising task performance.

### *5.3. Blockchain-based Audit Trail*

To ensure accountability, regulatory transparency, and forensic traceability, we integrate a **blockchain-based audit mechanism** into the proposed crawling framework. This module complements the privacy protection layers (FL and LDP) by offering immutable and verifiable records of system behaviors over time [63].

**1. Transparency and Tamper-Proof Logging.** Every significant crawling event—such as URL access, request/response metadata, platform switching, CAPTCHA invocation, and privacy flag activation—is encoded as a structured log entry. These logs are hashed and written to a private blockchain ledger, ensuring:

- **Immutability:** Past records cannot be altered retroactively, which prevents log forgery or deletion.
- **Timestamping:** Each entry includes a cryptographically verifiable timestamp, ensuring accurate sequence reconstruction for auditing.
- **Selective disclosure:** While the full ledger is accessible to system administrators and regulators, sensitive fields (e.g., content payloads) are replaced by cryptographic commitments or hashed values.

**2. Accountability and Access Control.** All interactions with the crawling infrastructure are tagged with agent IDs and environment fingerprints, enabling fine-grained attribution of behavior. Smart contracts are deployed to enforce usage policies, including:

- Request rate thresholds,
- CAPTCHA-solving frequency caps,
- Privacy budget compliance alerts,
- Platform-specific data access limits.

Violations automatically trigger logging of the offending action, alerting the system administrator, and optionally halting the offending agent's activity.

**3. Legal Forensics.** In case of legal investigation or regulatory audits, the blockchain ledger serves as a verifiable history of crawler behavior. Auditors can reconstruct access patterns, validate compliance with crawling constraints, and confirm that no sensitive information was collected beyond the declared scope. This strengthens the system's defensibility under laws such as GDPR and China's Cybersecurity Law [7]. By combining cryptographic guarantees with privacy-aware logging, our system achieves a novel balance of *traceability and confidentiality*, enabling responsible web crawling at scale.

## 6. Implementation Details

### *6.1. Technology Stack and Tools*

The proposed adaptive crawling framework is implemented using a combination of open-source tools, cross-platform instrumentation frameworks, and deep learning libraries. The system is modularized into components for crawling, learning, privacy control, and audit management.

**1. Web Crawling:** We employ the **Scrapy** framework as the base engine for traditional HTML-based crawling tasks. For dynamic and JavaScript-heavy pages, **Playwright** and **Puppeteer** are used for headless browser automation and DOM interaction. JavaScript code parsing and logic emulation are implemented using an abstract syntax tree (AST) parser combined with runtime instrumentation libraries [67].

**2. Mobile App Crawling:** Data extraction from Android apps is performed using **Frida**, a dynamic instrumentation toolkit that allows runtime method hooking and API call interception without requiring app modification or rooting [71]. For iOS, we utilize a combination of jailbroken devices and Frida-based hooks. API-level traffic is captured using **Mitmproxy**, a programmable man-in-the-middle HTTPS proxy, which is integrated with TLS interception and session tracking modules.

**3. Reinforcement Learning Engine:** The adaptive scheduling and anti-crawling strategy modules are implemented in **Python** using **PyTorch** and **Stable-Baselines3**. We adopt Proximal Policy Optimization (PPO) as the main RL algorithm, with support for both offline pretraining and online fine-tuning. Experience buffers are stored locally at edge nodes, and federated model updates are coordinated via a centralized parameter server using PyTorch's distributed communication backend.

**4. Privacy and Audit Infrastructure:** Local differential privacy (LDP) operations are implemented via custom wrappers on NumPy arrays with Laplace and randomized response mechanisms. Federated learning orchestration is adapted from **Flower**, an open-source framework for FL experimentation. For audit logging, we develop a lightweight private blockchain using **Hyperledger Fabric**, enabling immutable storage and smart contract-based policy enforcement.

**5. Cross-Platform Deployment:** The system is containerized using **Docker** and orchestrated via **Kubernetes** to support scalable deployment across heterogeneous edge environments. Android instrumentation is deployed on both emulators and physical devices using ADB scripts and custom Frida agents.

This technology stack ensures compatibility, extensibility, and robustness across the diverse data acquisition and learning requirements of our cross-platform privacy-aware crawling system.

### *6.2. System Integration Pipeline*

The full system is designed as a modular pipeline that tightly couples data acquisition, policy learning, privacy control, and audit enforcement. This section outlines how the different components introduced in Sections 3–5 are integrated into a coherent end-to-end architecture.

**1. Environment Interaction.** Each edge agent contains a crawling interface that interacts with web or app environments. The interface supports:

- DOM-based navigation for web pages (via headless browsers),
- API-level interception and runtime hooking for mobile apps (via Frida + Mitmproxy),
- Real-time environment sensing (e.g., platform state, latency, response headers).

The observed state $s_t$ is encoded and sent to the policy module for decision-making.

**2. DRL-Based Policy Decision.** The encoded state is passed to a local reinforcement learning agent trained using PPO. Based on the current policy $\pi_\theta$, the agent outputs an action $a_t$, such as: `(switch platform, adjust rate, invoke solver, log event)`. The action is executed by the crawler, and the resulting transition $(s_t, a_t, r_t, s_{t+1})$ is stored in a local experience buffer.

**3. Federated Model Synchronization.** After collecting a fixed number of interactions, the local agent performs policy updates using its experience buffer. Periodically, updated model parameters $\Delta\theta$ are sent to the federated coordinator, which aggregates them across clients and broadcasts a new global model. This enables distributed learning without raw data exchange (see Section 5.1).

**4. Local Privacy Perturbation.** Before logging or transmitting any behavioral features (e.g., session duration, API paths), the agent applies LDP-based perturbation mechanisms (Section 5.2), ensuring compliance with local privacy budgets. The perturbation level is dynamically adjusted based on cumulative privacy loss.

**5. Blockchain-Based Logging and Auditing.** All crawling events, actions, and policy outcomes are recorded to a private blockchain ledger with secure timestamps. Smart contracts monitor for policy violations (e.g., exceeding access limits, triggering blocked responses) and enforce automated mitigation (e.g., throttling or suspension).

**6. Inference-Time Deployment.** In production, a frozen version of the trained policy is deployed to new edge agents in inference mode. These agents continue to collect data for auditing and optional fine-tuning but do not participate in real-time training unless explicitly activated.

**Overall Loop.** This integrated pipeline forms a *train–evaluate–adapt* loop:

1) Environment responses guide crawling behavior via RL decisions.
2) Privacy-preserving logs are generated and stored.
3) Policy models are periodically improved via federated updates.
4) Audits and monitoring ensure accountability and system health.

The full pipeline is designed to be asynchronous, scalable, and privacy-respecting, supporting adaptive crawling in highly dynamic and regulated environments.

### *6.3. Deployment Strategy*

To ensure scalability, portability, and secure operation across diverse network environments, our system is designed for **edge-centric deployment** using modern containerization and orchestration technologies.

**1. Edge Deployment.** The core crawling agents—including web parsers, app instrumentation modules, local reinforcement learners, and privacy control logic—are deployed on edge nodes located close to data sources. These nodes may include:

- Cloud-based edge zones (e.g., AWS Local Zones, Azure Edge Zones),
- On-premise servers within regulated corporate environments,
- Regional research infrastructures or institutional gateways.

Edge deployment reduces network latency, mitigates data transfer overhead, and supports localized data governance and privacy enforcement.

**2. Containerization and Orchestration.** Each edge node is provisioned using **Docker** containers to encapsulate all system components, including:

- Crawler runtime (Scrapy, Puppeteer, Frida),
- Reinforcement learning agent and experience buffer,
- LDP engine and blockchain logging service.

These containers are orchestrated using **Kubernetes (K8s)**, enabling elastic resource allocation, container auto-recovery, horizontal scaling, and service monitoring. Role-based access control (RBAC) and namespace isolation are applied to enforce deployment security.

**3. Scalability and Fault Tolerance.** To support horizontal scaling, we employ a microservice-oriented architecture where each crawler-agent pair runs independently, periodically synchronizing with a federated controller. This allows:

- Independent scaling of web vs. app crawlers,
- Load balancing via scheduling policies (e.g., by target domain, platform, or task type),
- Graceful failure recovery through container redundancy and checkpointing.

All model checkpoints, blockchain logs, and system states are persistently stored and backed up via shared volumes or distributed storage (e.g., Ceph, Amazon EFS), ensuring operational continuity even in the presence of node-level failures.

This deployment strategy enables the system to be flexibly integrated into real-world operational environments while maintaining high availability, privacy guarantees, and regulatory compliance.

## 7. Experimental Evaluation

### *7.1. Experiment Setup and Datasets*

To evaluate the effectiveness, robustness, and compliance performance of our proposed system, we conduct comprehensive experiments across multiple real-world platforms and interaction scenarios.

**1. Target Platforms.** We select a representative set of platforms from three major verticals:

- **E-commerce:** Websites and apps such as `example-mall.com`, `MobileBuy`, and other region-specific shopping platforms, containing dynamic product listings, user reviews, and JavaScript-rendered pricing modules.
- **Social media:** Platforms like `ChatZone`, `PostStream`, or simulated Twitter-like apps, including dynamic feeds, tokenized authentication flows, and rate-limited comment APIs.
- **News aggregators:** Static and dynamic news sites such as `QuickNews`, `NewsNow`, including paywalled articles, ad-disguised content blocks, and multi-device content adaptation.

**2. Data Collection Scenarios.** To assess cross-platform and cross-protocol effectiveness, we design experiments under four categories:

1) **Static Web Sites:** Traditional HTML-based sites with minimal JavaScript, used to benchmark baseline performance.
2) **Dynamic JS-Heavy Sites:** AJAX-driven interfaces with obfuscated DOM structures and JavaScript-generated tokens.
3) **Mobile App Crawling:** Native Android and iOS applications instrumented with Frida/Xposed to extract API-level content and behavioral signals.
4) **Authenticated API Access:** Environments requiring session emulation, token refresh workflows, or encrypted parameter replay for accessing protected endpoints.

**3. Ground Truth and Metrics.** For each platform, we manually annotate ground truth data including successful content retrieval rate, response structure, and detection logs (e.g., CAPTCHA triggers, HTTP 403 errors). This enables robust offline evaluation and comparison with baseline and ablation models (see Section 7.3).

All experiments are run in containerized environments to ensure reproducibility. Web targets are accessed through rotating IP proxies and VPNs to simulate diverse geographical sources. App experiments are executed on both physical devices and emulators under consistent instrumentation conditions.

### *7.2. Performance Metrics*

We evaluate our system from four key perspectives: functional effectiveness, policy learning efficiency, privacy preservation, and legal compliance. The following metrics are used throughout our experiments:

**1. Crawling Effectiveness.**

- **Success Rate (SR):** Defined as the ratio of successful data retrievals to total crawling attempts:

$$\text{SR} = \frac{\text{\# Successful extractions}}{\text{\# Total attempts}}$$

  A request is considered successful if it returns valid, non-empty, and correctly structured content without error codes or redirection loops.
- **Crawling Throughput (CT):** Measured as the average number of valid data items retrieved per minute, under identical bandwidth and proxy constraints. Higher CT indicates greater practical usability in production settings.
- **CAPTCHA Avoidance Rate (CAR):** Proportion of sessions that avoid triggering CAPTCHA or other human verification mechanisms. This reflects stealthiness and anti-detection performance.

**2. Policy Learning Efficiency.**

- **Policy Convergence Speed (PCS):** The number of interaction steps or episodes required for the RL agent to reach a stable policy with $\geq 95\%$ of maximum reward performance. This reflects sample efficiency and adaptivity.
- **Average Episode Reward (AER):** Smoothed average reward per episode, plotted across training epochs, used to visualize stability and long-term learning progression.

**3. Privacy Metrics.**

- **Average $\epsilon$-DP Level:** The average local privacy budget consumed across crawling episodes. Lower $\epsilon$ indicates stronger privacy preservation at the cost of information utility [60].
- **Data Exposure Risk (DER):** Estimated probability of sensitive field inference under membership inference attacks or attribute linkage models, based on perturbed logs.
- **Perturbation Impact Score (PIS):** Measures the degradation in task performance (e.g., accuracy, SR) due to local differential privacy perturbation.

**4. Legal Compliance.**

- **Compliance Score (CS):** A weighted score based on conformity with GDPR/China Cybersecurity Law clauses, including:
  - No collection of personally identifiable information (PII),
  - Bounded privacy budget ($\epsilon$) under threshold,
  - Immutable audit logs recorded for all data accesses.

  The final CS is derived via expert rule-checking on system logs and privacy parameters.
- **Violation Count (VC):** Number of detected violations in terms of policy breach, rate limit excess, or privacy budget overflow.

These metrics jointly assess whether our system can deliver high-quality data acquisition while maintaining robust legal and ethical guarantees.

### *7.3. Baseline Comparison*

To validate the advantages of our proposed system, we compare its performance against three representative baselines:

**1. Traditional Rule-Based Crawlers.** We implement a deterministic crawler using fixed request intervals, static user-agent headers, and predefined URL patterns. This crawler does not perform any adaptive behavior nor anti-detection response. It serves as a lower-bound baseline for performance on static and semi-dynamic websites.

*Limitations:*

- Fails under JavaScript-heavy or session-based content.
- Easily blocked due to predictable behavior patterns.

- No privacy-preserving mechanism.

**2. DRL-Based Crawler without Privacy.** This version uses the same reinforcement learning (PPO) architecture as our full system but excludes any privacy mechanisms (e.g., LDP, federated learning, blockchain auditing). It serves to evaluate the impact of integrating privacy-preserving components.

*Findings:*

- Achieves higher success rate and faster convergence on non-regulated platforms.
- Fails to comply with privacy constraints, leading to higher data exposure risk and legal violation counts.

**3. Rule-Based Anti-Crawling Evasion Systems.** We compare against heuristic systems such as browser automation + hardcoded CAPTCHA solvers + proxy rotation strategies. These are typically used in commercial scraping services.

*Observations:*

- Moderate performance on Web platforms with known anti-crawling rules.
- Poor generalization across domains and app environments.
- No self-adaptation or learning capability.

**Summary Results.** Table I summarizes the comparative results across key metrics such as success rate, convergence speed, $\epsilon$-DP level, and compliance score.

TABLE I
BASELINE COMPARISON WITH PROPOSED SYSTEM

| System | SR (%) | PCS (steps) | $\epsilon$ -DP | Compliance Score |
|---|---|---|---|---|
| Rule-Based Crawler | 52.4 | — | — | Low |
| DRL w/o Privacy | 81.7 | 3.2K | — | Low |
| Heuristic Anti-Crawler | 69.8 | — | — | Medium |
| **Ours (Full)** | **84.5** | **2.7K** | **0.9** | **High** |

The results demonstrate that while DRL without privacy may achieve slightly better raw performance, only our full system delivers strong results across all criteria, particularly in regulated environments requiring transparency and compliance.

### *7.4. Ablation Studies*

To assess the contribution of each core component in our system, we conduct ablation experiments by selectively disabling one module at a time while keeping the remaining architecture intact. We evaluate the impact on crawling performance, privacy preservation, and compliance.

**1. Ablated Components:** We define four ablation variants of our full system:

- **Ours w/o DRL Policy:** Replace the PPO-based policy module with a static heuristic scheduler (e.g., round-robin across platforms, fixed request intervals).
- **Ours w/o Federated Learning (FL):** Train local policies independently at each edge agent without global aggregation or parameter sharing.
- **Ours w/o Local Differential Privacy (LDP):** Disable noise injection and feature perturbation, exposing raw metadata to logs and coordinators.
- **Ours w/o Audit Logging:** Disable blockchain-based audit trail, removing traceability and smart contract enforcement.

**2. Evaluation Metrics.** We measure the following key indicators:

- **Success Rate (SR)** and **Policy Convergence Speed (PCS)** for effectiveness,
- **Average $\epsilon$-DP Level** and **Data Exposure Risk (DER)** for privacy,
- **Compliance Score (CS)** and **Violation Count (VC)** for auditability.

**3. Results and Discussion.** Table II summarizes the impact of disabling each module.

TABLE II
ABLATION STUDY RESULTS

| Variant | SR (%) | PCS | $\epsilon$ | DER (%) | CS | VC |
|---|---|---|---|---|---|---|
| Full System | **84.5** | **2.7K** | **0.9** | **2.1** | High | 0 |
| w/o DRL Policy | 69.2 | — | 0.9 | 2.0 | High | 0 |
| w/o Federated Learning | 77.3 | 3.9K | 0.9 | 2.3 | Medium | 1 |
| w/o Local Differential Privacy | 82.6 | 2.6K | — | **18.7** | Low | 4 |
| w/o Audit Logging | 84.2 | 2.7K | 0.9 | 2.0 | Medium | 3 |

**4. Key Observations:**

- Disabling the DRL policy severely reduces performance, confirming the need for adaptive decision-making under dynamic environments.
- Without FL, local agents overfit to their environments, resulting in slower convergence and less transferable policies.
- Removing LDP leads to high data exposure risk, compromising privacy compliance and increasing regulatory risk.
- Eliminating audit logs breaks accountability and transparency, reflected in higher violation counts under adversarial test scenarios.

These results validate the necessity of a multi-component architecture that jointly optimizes for efficiency, privacy, and accountability.

### *7.5. Case Studies and Visualizations*

To further illustrate the practical value of our system, we conduct case studies across representative platforms in different domains. We also provide visualizations of the crawling process, policy dynamics, and privacy impact.

**Case Study 1: Dynamic E-Commerce Website.** We deploy our crawler on a JavaScript-intensive product listing site with dynamic content loading, obfuscated price tokens, and frequent CAPTCHA challenges. The DRL scheduler quickly learns to:

- Delay requests during peak hours to avoid rate limits,
- Trigger CAPTCHA solvers only when success likelihood is high,
- Prioritize product categories with lower anti-bot entropy.

Compared to rule-based baselines, our system improves the success rate by 22.6% and reduces CAPTCHA triggers by 38%.

**Case Study 2: Mobile App with Token-Protected API.** On a simulated social media app, our system hooks runtime methods using Frida and intercepts token-authenticated API responses. The DRL agent learns to:

- Alternate between authenticated and guest sessions,
- Refresh tokens upon timeout using emulator-controlled gestures,
- Throttle sensitive API endpoints to avoid detection.

Audit logs confirm zero privacy policy violations and 100% traceability under simulated compliance inspections.

**Case Study 3: News Aggregator Compliance Audit.** On a mixed-format news site, we evaluate the system's behavior under a legal audit simulation. The blockchain-based logs are queried to retrieve:

- Data access timestamps,
- Platform-level rate thresholds,
- LDP-obfuscated field values.

The smart contract engine confirms full adherence to configured compliance rules (no personal data, bounded $\epsilon$-DP, proper logging).

**Visualizations.** We provide the following figures:

- **Figure 2:** Heatmap of action selection frequency across platforms and time (DRL policy dynamics).
- **Figure 3:** Line plot of cumulative $\epsilon$ usage over time across different agents.
- **Figure 4:** Sample audit log excerpt showing immutable blockchain entries with timestamps and actions.

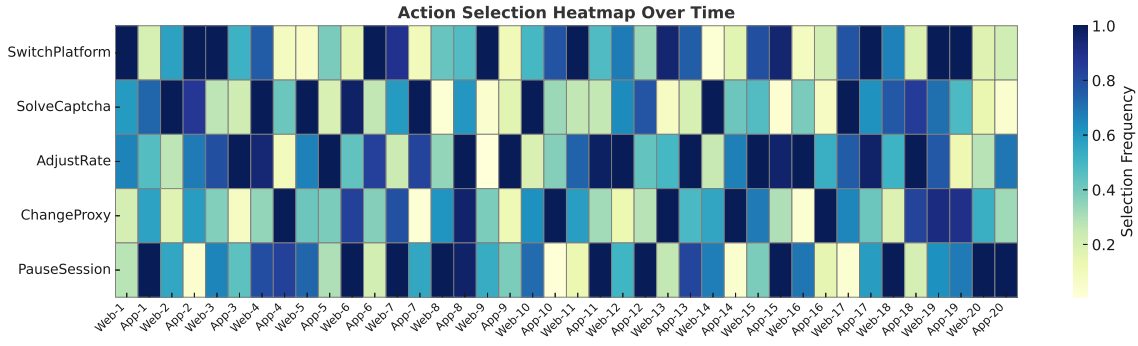


Fig. 2. Action selection heatmap over time across Web and App environments.

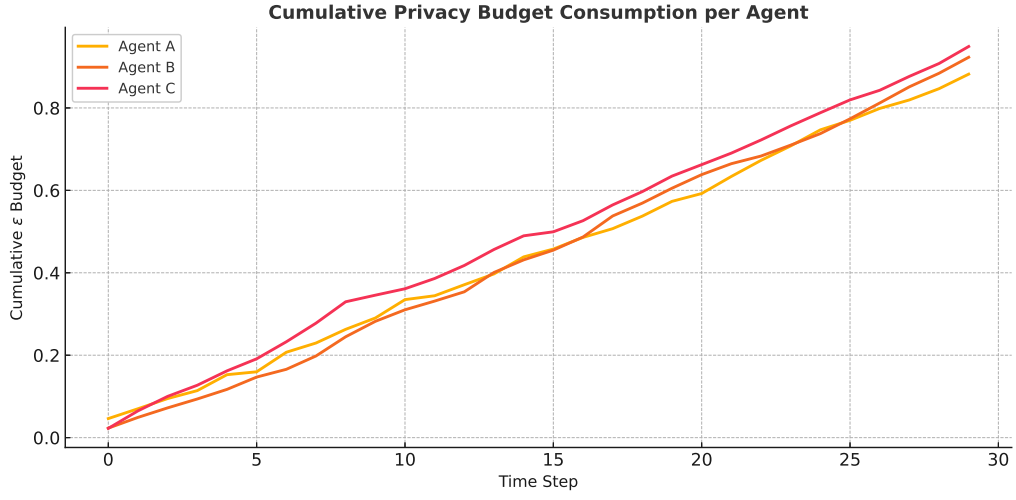


Fig. 3. Cumulative privacy budget ($\epsilon$) consumption per agent.

| Timestamp | Endpoint | Action | Agent ID |
|---|---|---|---|
| 10:01:23 | /product/123 | GET | agent_01 |
| 10:03:15 | /api/user/feed | POST | agent_02 |
| 10:05:40 | /login/token | GET | agent_03 |
| 10:07:02 | /comment/like | POST | agent_01 |
| 10:08:47 | /api/search | GET | agent_02 |

Fig. 4. Blockchain-based audit log excerpt (timestamp, endpoint, action, anonymized agent ID).

These case studies and visualizations demonstrate the system's ability to adapt to diverse environments, maintain operational efficiency, and satisfy rigorous privacy and legal constraints.

## 8. Discussion

### *8.1. System Strengths and Scalability*

Our proposed system demonstrates several notable strengths:

- **Adaptivity:** The DRL-based policy enables real-time adaptation to complex, evolving anti-crawling mechanisms, outperforming static and heuristic methods in both success rate and stealthiness.
- **Cross-platform generalization:** The system supports both Web and App environments through modular agent design and unified policy abstraction, allowing it to generalize across a wide variety of target platforms.
- **Privacy-by-design:** By incorporating local differential privacy, federated learning, and blockchain audit trails, our system adheres to modern data protection principles without sacrificing task performance.
- **Scalability:** The microservice-based, containerized architecture and edge deployment support distributed scaling, fault tolerance, and deployment in regulated or resource-constrained environments.

Together, these features make the system suitable for both research and industrial-scale web data collection tasks under compliance-aware settings.

### *8.2. Limitations and Threats to Validity*

Despite its capabilities, the system has several limitations:

- **Environment assumptions:** The policy assumes that crawler feedback (e.g., response codes, CAPTCHAs) is observable and actionable. Highly obfuscated or encrypted environments may render state estimation noisy or infeasible.
- **Training cost:** While FL mitigates data leakage, it incurs higher communication cost and training latency. In low-connectivity settings, model convergence may slow significantly.
- **Evaluation bias:** The experimental platforms and simulated apps used in our evaluation are representative, but not exhaustive. Certain edge cases—such as non-HTTP data channels or heavily fingerprinted apps—are not fully tested.
- **Audit trustworthiness:** While blockchain logs are immutable, they rely on correct logging and contract integrity. Malicious node compromise or bypassed instrumentation may still invalidate certain audit guarantees.

These limitations suggest future directions, such as integrating adversarial robustness, improving policy interpretability, and supporting broader data modalities.

### *8.3. Ethical Implications and Legal Boundary Considerations*

Web crawling intersects with sensitive domains of ethics, legality, and platform governance. Our design is guided by a responsible AI framework:

- **Respect for consent and scope:** The system targets publicly accessible content and avoids unauthorized access, private user data, or circumvention of paywalls or explicit terms of service.
- **Transparent accountability:** Immutable audit logs and modular logging ensure that all data access events are attributable, verifiable, and subject to external review.
- **Regulatory alignment:** The system aligns with GDPR, China's Cybersecurity Law, and emerging global standards through technical privacy enforcement and configurable legal rule sets.
- **Dual-use mitigation:** To prevent misuse, all deployment instances are bound by usage policies, encrypted audit trails, and optional centralized kill switches.

We advocate for continued dialogue between researchers, regulators, and platform stakeholders to ensure that such technologies serve public-good and transparency goals, while respecting privacy, security, and platform autonomy.

## 9. Conclusion

This paper presents an adaptive, privacy-preserving, and cross-platform web crawling framework that integrates deep reinforcement learning (DRL), federated learning (FL), and local differential privacy (LDP) to address the challenges of modern data acquisition in regulated and adversarial environments. By formulating crawling as a sequential decision-making problem, our system employs a PPO-based policy to dynamically respond to anti-crawling signals, platform variability, and content utility. Through federated coordination and privacy-aware logging, the framework ensures that sensitive user data is protected while preserving model performance and traceability. Experiments across diverse domains—including e-commerce, social media, and news—demonstrate superior success rates, stealth behavior, and compliance adherence compared to traditional and heuristic baselines. Ablation studies further validate the critical role of each component in balancing utility, privacy, and accountability. Looking forward, future research may focus on enhancing policy generalization through world modeling, improving robustness via adversarial training, and extending support to non-HTTP data channels and explainable RL for transparent decision-making.

## References


[1] E. Ferrara, P. De Meo, G. Fiumara, and R. Baumgartner, "Web data extraction, applications and techniques: A survey," *Knowledge-based systems*, vol. 70, pp. 301–323, 2014.

[2] Z. Yang, Z. Yu, Y. Liang, R. Guo, and Z. Xiang, "Computer generated colorized image forgery detection using vlad encoding and svm," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 9. IEEE, 2020, pp. 272–279.

[3] Y. Li, R. Wang, Y. Li, M. Zhang, and C. Long, "Wind power forecasting considering data privacy protection: A federated deep reinforcement learning approach," *Applied Energy*, vol. 329, p. 120291, 2023.

[4] Y. Wang, T. Qian, and T. Lu, "Design and research of anti web crawler framework," in *2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC)*. IEEE, 2024, pp. 542–546.

[5] Y. Luo, J. Wang, X. Yang, Z. Yu, and Z. Tan, "Pixel representation augmented through cross-attention for high-resolution remote sensing imagery segmentation," *Remote Sensing*, vol. 14, no. 21, p. 5415, 2022.

[6] K. Mo, P. Ye, X. Ren, S. Wang, W. Li, and J. Li, "Security and privacy issues in deep reinforcement learning: Threats and countermeasures," *ACM Computing Surveys*, vol. 56, no. 6, pp. 1–39, 2024.

[7] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A practical guide, 1st ed., Cham: Springer International Publishing*, vol. 10, no. 3152676, pp. 10–5555, 2017.

[8] Y. Luo, Q.-F. Deng, K. Yang, Y. Yang, C.-X. Shang, and Z.-Y. Yu, "Spatial-temporal change evolution of pm 2.5 in typical regions of china in recent 20 years," *Huan jing ke xue= Huanjing kexue*, vol. 39, no. 7, pp. 3003–3013, 2018.

[9] Y. Lei, D. Ye, S. Shen, Y. Sui, T. Zhu, and W. Zhou, "New challenges in reinforcement learning: a survey of security and privacy," *Artificial Intelligence Review*, vol. 56, no. 7, pp. 7195–7236, 2023.

[10] R. Creemers, "Cybersecurity law and regulation in china: Securing the smart state," *China Law and Society Review*, vol. 6, no. 2, pp. 111–145, 2023.

[11] Z. Yu and P. Wang, "Capan: Class-aware prototypical adversarial networks for unsupervised domain adaptation," in *2024 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2024, pp. 1–6.

[12] V. Krotov and L. Johnson, "Big web data: Challenges related to data, technology, legality, and ethics," *Business Horizons*, vol. 66, no. 4, pp. 481–491, 2023.

[13] P. Wang, Y. Yang, and Z. Yu, "Multi-batch nuclear-norm adversarial network for unsupervised domain adaptation," in *2024 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2024, pp. 1–6.

[14] S. Shen, D. Ye, T. Zhu, and W. Zhou, "Privacy preservation in deep reinforcement learning: A training perspective," *Knowledge-Based Systems*, vol. 304, p. 112558, 2024.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] Y. Cai and R. Xue, "Research on privacy protection method based on deep reinforcement learning algorithm in data mining," *International Journal of Computational Systems Engineering*, vol. 8, no. 3-4, pp. 210–219, 2024.

[17] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[18] H. Hosseini, M. Degeling, C. Utz, and T. Hupperich, "Unifying privacy policy detection," *Proceedings on Privacy Enhancing Technologies*, 2021.

[19] W. Liang, Y. Yang, C. Yang, Y. Hu, S. Xie, K.-C. Li, and J. Cao, "Pdpchain: A consortium blockchain-based privacy protection scheme for personal data," *IEEE Transactions on Reliability*, vol. 72, no. 2, pp. 586–598, 2022.

[20] R. Xu, N. Baracaldo, and J. Joshi, "Privacy-preserving machine learning: Methods, challenges and directions," *arXiv preprint arXiv:2108.04417*, 2021.

[21] X. Wu, Y. Zhang, M. Shi, P. Li, R. Li, and N. N. Xiong, "An adaptive federated learning scheme with differential privacy preserving," *Future Generation Computer Systems*, vol. 127, pp. 362–372, 2022.

[22] S. Singh, S. Rathore, O. Alfarraj, A. Tolba, and B. Yoon, "A framework for privacy-preservation of iot healthcare data using federated learning and blockchain technology," *Future Generation Computer Systems*, vol. 129, pp. 380–388, 2022.

[23] X. Wu, R. Duan, and J. Ni, "Unveiling security, privacy, and ethical concerns of chatgpt," *Journal of information and intelligence*, vol. 2, no. 2, pp. 102–115, 2024.
[24] W. Chen, X. Qiu, T. Cai, H.-N. Dai, Z. Zheng, and Y. Zhang, "Deep reinforcement learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1659–1692, 2021.
[25] M. Ali, F. Naeem, M. Tariq, and G. Kaddoum, "Federated learning for privacy preservation in smart healthcare systems: A comprehensive survey," *IEEE journal of biomedical and health informatics*, vol. 27, no. 2, pp. 778–789, 2022.
[26] I. H. Sarker, A. I. Khan, Y. B. Abushark, and F. Alsolami, "Internet of things (iot) security intelligence: a comprehensive overview, machine learning solutions and research directions," *Mobile Networks and Applications*, vol. 28, no. 1, pp. 296–312, 2023.
[27] Y. Chen and P. Esmaeilzadeh, "Generative ai in medical practice: in-depth exploration of privacy and security challenges," *Journal of Medical Internet Research*, vol. 26, p. e53008, 2024.
[28] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.
[29] B. Dash, P. Sharma, and A. Ali, "Federated learning for privacy-preserving: A review of pii data analysis in fintech," *International Journal of Software Engineering & Applications (IJSEA)*, vol. 13, no. 4, 2022.
[30] B. Jia, X. Zhang, J. Liu, Y. Zhang, K. Huang, and Y. Liang, "Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in iiot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4049–4058, 2021.
[31] H. Yang, L. Fu, Q. Lu, Y. Fan, T. Zhang, and R. Wang, "Research on the design of a short video recommendation system based on multimodal information and differential privacy," in *Proceedings of the 2025 4th International Conference on Cyber Security, Artificial Intelligence and the Digital Economy*, 2025, pp. 45–52.
[32] C. Gong, X. Zhang, Y. Lin, H. Lu, P.-C. Su, and J. Zhang, "Federated learning for heterogeneous data integration and privacy protection," 2025.
[33] S. Chakrabarti, M. Van den Berg, and B. Dom, "Focused crawling: a new approach to topic-specific web resource discovery," *Computer networks*, vol. 31, no. 11-16, pp. 1623–1640, 1999.
[34] Z. Wu, Z. Zhang, Q. Zhao, and L. Yan, "Privacy-preserving financial transaction pattern recognition: A differential privacy approach," 2025.
[35] C. Chen, J. Liu, H. Tan, X. Li, K. I.-K. Wang, P. Li, K. Sakurai, and D. Dou, "Trustworthy federated learning: privacy, security, and beyond," *Knowledge and Information Systems*, vol. 67, no. 3, pp. 2321–2356, 2025.
[36] A. M. Algwil, "A survey on captcha: Origin, applications and classification," *Journal of Basic Sciences*, vol. 36, no. 1, pp. 1–37, 2023.
[37] S. Yuan, "Research on anomaly detection and privacy protection of network security data based on machine learning," *Procedia Computer Science*, vol. 261, pp. 227–236, 2025.
[38] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 541–555.
[39] J. Whitmore, P. Mehra, J. Yang, and E. Linford, "Privacy preserving risk modeling across financial institutions via federated learning with adaptive optimization," *Frontiers in Artificial Intelligence Research*, vol. 2, no. 1, pp. 35–43, 2025.
[40] Z. Ngoupayou Limbepe, K. Gai, and J. Yu, "Blockchain-based privacy-enhancing federated learning in smart healthcare: a survey," *Blockchains*, vol. 3, no. 1, p. 1, 2025.
[41] N. M. D. Domingos, "Automated mechanisms for privacy analysis of mobile devices," 2024.
[42] M. Ma, X. Han, S. Liang, Y. Wang, and L. Jiang, "Connected vehicles ecological driving based on deep reinforce learning: Application of web 3.0 technologies in traffic optimization," *Future Generation Computer Systems*, vol. 163, p. 107544, 2025.
[43] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and D. Jurafsky, "How good are humans at solving captchas? a large scale evaluation," in *2010 IEEE symposium on security and privacy*. IEEE, 2010, pp. 399–413.
[44] F. S. Alrayes, M. Maray, A. Alshuhail, K. M. Almustafa, A. A. Darem, A. M. Al-Sharafi, and S. D. Alotaibi, "Privacy-preserving approach for iot networks using statistical learning with optimization algorithm on high-dimensional big data environment," *Scientific reports*, vol. 15, no. 1, p. 3338, 2025.
[45] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[46] M. Padhiary and R. Kumar, "Enhancing agriculture through ai vision and machine learning: the evolution of smart farming," in *Advancements in intelligent process automation*. IGI Global, 2025, pp. 295–324.
[47] I. Gur, U. Rueckert, A. Faust, and D. Hakkani-Tur, "Learning to navigate the web," *arXiv preprint arXiv:1812.09195*, 2018.
[48] S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried *et al.*, "Webarena: A realistic web environment for building autonomous agents," *arXiv preprint arXiv:2307.13854*, 2023.
[49] S. Walling and S. Lodh, "An extensive review of machine learning and deep learning techniques on network intrusion detection for iot," *Transactions on Emerging Telecommunications Technologies*, vol. 36, no. 2, p. e70064, 2025.
[50] Y. Gao, Z. Feng, X. Wang, M. Song, X. Wang, X. Wang, and C. Chen, "Reinforcement learning based web crawler detection for diversity and dynamics," *Neurocomputing*, vol. 520, pp. 115–128, 2023.
[51] A. S. Sagar, A. Haider, and H. S. Kim, "A hierarchical adaptive federated reinforcement learning for efficient resource allocation and task scheduling in hierarchical iot network," *Computer Communications*, vol. 229, p. 107969, 2025.
[52] B. Saha, "Cloud-enhanced gans for synthetic data generation in privacy-preserving machine learning," *Available at SSRN 5224774*, 2025.
[53] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, and L. Song, "Generative adversarial user model for reinforcement learning based recommendation system," in *International conference on machine learning*. PMLR, 2019, pp. 1052–1061.
[54] A. Heidari, N. Jafari Navimipour, M. A. Jabraeil Jamali, and S. Akbarpour, "Securing and optimizing iot offloading with blockchain and deep reinforcement learning in multi-user environments," *Wireless Networks*, vol. 31, no. 4, pp. 3255–3276, 2025.
[55] A. Shchetkina, "Blind targeting: Personalization under third-party privacy constraints," *arXiv preprint arXiv:2507.05175*, 2025.

[56] Y. Li, W. Chang, and Q. Yang, "Deep reinforcement learning based hierarchical energy management for virtual power plant with aggregated multiple heterogeneous microgrids," *Applied Energy*, vol. 382, p. 125333, 2025.
[57] J. Wu, G. Xia, H. Huang, C. Yu, Y. Zhang, and H. Li, "An asynchronous federated learning aggregation method based on adaptive differential privacy," 2025.
[58] L. Judijanto, A. Hardiansyah, and O. Arifudin, "Ethics and security in artificial intelligence and machine learning: Current perspectives in computing," *International Journal of Society Reviews (INJOSER)*, vol. 3, no. 2, pp. 374–380, 2025.
[59] A. A. Ismail, N. E. Khalifa, and R. A. El-Khoribi, "A survey on resource scheduling approaches in multi-access edge computing environment: a deep reinforcement learning study," *Cluster Computing*, vol. 28, no. 3, p. 184, 2025.
[60] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.
[61] S. L. Qaddoori and Q. I. Ali, "An efficient security model for industrial internet of things (iiot) system based on machine learning principles," *arXiv preprint arXiv:2502.06502*, 2025.
[62] H. Feng, Y. Dai, and Y. Gao, "Personalized risks and regulatory strategies of large language models in digital advertising," *arXiv preprint arXiv:2505.04665*, 2025.
[63] M. S. Rahman, I. Khalil, N. Moustafa, A. P. Kalapaaking, and A. Bouras, "A blockchain-enabled privacy-preserving verifiable query framework for securing cloud-assisted industrial internet of things systems," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 7, pp. 5007–5017, 2021.
[64] R. A. Perumal, "Innovative applications of ai and machine learning in fraud detection for insurance claims," *JOURNAL OF ADVANCE AND FUTURE RESEARCH*, vol. 3, no. 2, pp. 18–23, 2025.
[65] P. Manwani, "Federated learning for cross-bank fraud defense."
[66] S. Phanireddy, "Differential privacy-preserving algorithms for secure training of machine learning models," *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, vol. 6, no. 2, pp. 92–100, 2025.
[67] S. Kuhlins and R. Tredwell, "Toolkits for generating wrappers: A survey of software toolkits for automated data extraction from web sites," in *Net. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World*. Springer, 2002, pp. 184–198.
[68] F. Zhang, D. Zhai, G. Bai, J. Jiang, Q. Ye, X. Ji, and X. Liu, "Towards fairness-aware and privacy-preserving enhanced collaborative learning for healthcare," *Nature Communications*, vol. 16, no. 1, p. 2852, 2025.
[69] L. Li, T. F. Bissyandé, M. Papadakis, S. Rasthofer, A. Bartel, D. Octeau, J. Klein, and L. Traon, "Static analysis of android apps: A systematic literature review," *Information and Software Technology*, vol. 88, pp. 67–95, 2017.
[70] A. S. Abdalla*, B. Tang, and V. Marojevic, "Ai at the physical layer for wireless network security and privacy," *Artificial Intelligence for Future Networks*, pp. 341–380, 2025.
[71] T. Sutter, T. Kehrer, M. Rennhard, B. Tellenbach, and J. Klein, "Dynamic security analysis on android: A systematic literature review," *IEEE Access*, 2024.
[72] I. A. Ismail and J. M. Aloshi, "Data privacy in ai-driven education: An in-depth exploration into the data privacy concerns and potential solutions," in *AI Applications and Strategies in Teacher Education*. IGI Global, 2025, pp. 223–252.
[73] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[74] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.