Bringing Your Own Privacy: Towards Measurable Privacy in Mobile Device Management and Security

Shuwen Liu
Worcester Polytechnic Institute
Worcester, MA, USA
sliu9@wpi.edu

Craig A. Shue
Worcester Polytechnic Institute
Worcester, MA, USA
cshue@wpi.edu

***Abstract*—In the "bring your own device" (BYOD) paradigm, people use their personally-owned devices in the workplace. This creates challenges for maximizing the security of organizational assets while ensuring user privacy and autonomy. Traditional approaches either a) favor organizations with tight control and feature-rich security management at the cost of device owner control and privacy or b) monitor only specific device applications, granting users more privacy and control at the cost of situational understanding for the organization.**

In this work, we obtain and modify an existing mobile device management tool to enhance end-user privacy. We create a module that applies policy locally and communicates policy outcomes in real-time, providing organizational situational awareness. At the same time, we use a third-party escrow service to store the sensitive details and context surrounding those actions. Our system allows the security staff at the organization to query the escrow service for greater insight into traffic with a mechanism that allows the escrow service to score and report the privacy impacts of queries to end users. We create a mechanism and configurable model that provides scoring of the privacy impacts of organizations' queries while allowing organizations to respond rapidly to incidents. It protects users' privacy by default while allowing end-users to understand the privacy implications of the organization's actions.

***Index Terms*—Mobile Device, Measurable Privacy, Security**

I. Introduction

When considering the use of a personally-owned device for work, end-users must make complex decisions regarding the rights they have on their device and what rights they must cede to their employers. With the rapid growth of "bring your own device" (BYOD) usage since 2012 [1], more users are being confronted with these choices. The BYOD market was valued at $114 billion in 2024 and has a steady anticipated growth rate of over 15% through 2029 [2].

Solutions could place different weights on the importance of security and privacy. By weighing one factor more than the other, it could result in benefits that skew towards the device owners or towards organizations. The mobile device management (MDM) approach is most strongly weighted towards organizational security goals: it provides capabilities for organizations to secure, monitor, and manage BYOD systems [3]. This approach also allows organizations to track employees' locations at all times, view installed applications, and access personal information such as private emails and photos; this naturally affects end-user privacy and has raised privacy concerns for device owners. With such devices being personally owned, some employees simply refuse to install MDM on their devices and forgo mobile access to company systems [4], resulting in decreased productivity and employee morale. Ayedh et al. [5] argue that traditional security policies, such as network access control and MDM policies, are unable to adequately address privacy requirements that are crucial for BYOD systems. While Herrera et al. [6] propose a privacy-preserving access control system for mobile devices, this system lacks the user transparency of data privacy and still has privacy risks from the organization servers.

At the other end of the spectrum, mobile application management (MAM) is designed to secure individual applications on a BYOD system without having any control or insight into any other applications without MAM implementations [7]. Unfortunately, Khellaf et al. [8] note that MAM falls short in fulfilling key organizational security requirements.

Our goal in this work is to both support the security goals of organizations while providing concrete privacy metrics for device owners to understand and respond to organizations' use of the device's data. We combine theoretical models of privacy with an existing mobile device endpoint security tool to allow privacy-preserving situational awareness for organizations. We allow organizations to have a real-time view of policy decisions on BYOD devices associated with the organization. Rather than directly sharing the details and context for those policy decisions with the organization, we use encrypted record storage with an escrow service that brokers access and shares usage information. That escrow service provides the organization with an interface that employs differential privacy, masking, and encryption. The organization can make queries to the escrow service to obtain privacy-preserving responses, and where needed for security goals, to reveal encrypted or masked context and to remove differential privacy protections. In exchange for accessing this private data, the escrow records statistics about the organization's queries and the resulting privacy risk.

The escrow service shares these statistics and privacy score information with device owners and the organization. This effectively allows end-users to understand how the organization uses private data and whether it routinely accesses private data or if it accesses private data sparingly or only when needed. Organizations can predict the privacy score implications of queries they may make during an incident and determine if

the context is worth the privacy score penalties, while end-users can configure their devices to stop participating (and accessing organizational resources) if the privacy score crosses user-defined thresholds.

We explore three research questions: *To what extent can the privacy-preserving mechanisms inform mobile device users about the use of their private data? Can such a privacy-preserving access control system fulfill the security requirements of mobile devices? To what extent would such a privacy-preserving module influence the performance of Android devices?* We make the following contributions:

Creation of a Local Controller and Privacy Module for Android Devices: We extend Liu et al.'s [9] APPJUDICATOR, which is the first access control system for mobile devices that leverages the information from user interfaces (UIs) in the security policy. Since the APPJUDICATOR tool has the potential to introduce particularly acute privacy risks, we use it as an exemplar to study the introduction of privacy-preserving mechanisms with the understanding that other management tools with lesser privacy risks could proceed similarly. We modify APPJUDICATOR to add an on-device controller that determines policy rules, enacts them locally, and reports to the organization controller only information about the policy rules applied, verdicts, and timing. Our module provides detailed context in an encrypted fashion to an escrow service.

Privacy Metrics that Quantify Privacy Risks in BOYD Devices: Using the framework proposed by Giomi et al. [10], we quantify the privacy risks in synthetic data constructed from the fusion of real organizational packet captures and APPJUDICATOR logs[1]. We propose privacy metrics and scorecards for users to understand the effective privacy for BYOD in an organizational scenario.

Analysis of Security Impacts: We compare the access control and policy capabilities of the original APPJUDICATOR and our privacy-enhanced version. We examine the security implications of storing policy locally on the endpoint.

Performance Impact Analysis of Privacy Measures: We measure the network performance of Android devices running the original APPJUDICATOR with those running the privacy-preserving modules as to end-to-end round-trip times.

II. BACKGROUND AND RELATED WORK

A. BYOD Mobile Device Security

There has been steady market growth in BYOD use globally [2]. Incorporating personally-owned devices introduces security risks for companies, including the loss of devices containing sensitive corporate data, exposure to malware leading to data corruption, data theft, and reduced control over enterprise networks [11]. For example, the "Dresscode" malware was designed to infiltrate corporate networks [12] by appearing as a legitimate application in order to steal data and add infected devices to a botnet [13]. The malware attempts to propagate once an infected device connects to a network.

Prior work has explored network pattern analysis to enhance mobile device security. Burnett et al. [14] evaluate endpoint network logging by examining end-to-end network paths with browser-generated data to differentiate between on-path network failures and malicious attacks. Netsight [15] employs network profiling techniques to pinpoint the root causes of network issues, while ProfileDroid [16] utilizes traffic profiling to identify advertisement-related traffic.

Network logging has also been effectively applied to mobile network analysis [17]. Sipola et al. [18] introduce an approach utilizing deep learning algorithms to detect anomalous behaviors based on network log analysis; however, these techniques are constrained by limited logging sources. In parallel, profiling local system activities on Android has been explored. Lanzi et al. [19] reconstruct user actions through Android system-call logs, while AppContext [20] analyzes system logs to identify patterns characteristic of malicious apps. Both approaches require root or debugger access, limiting their potential for widespread adoption.

To address security issues introduced by BYOD, companies may implement MDM solutions to secure, monitor, and manage mobile devices [3]. MDM solutions [21], [22] offer mobile device users security functions including data protection, conditional access, and device configuration. Compared to MAM solutions, MDM provides device-wide monitoring and management. It allows organization IT teams to configure a set of security policies for various mobile devices in a centralized platform. After enforcing these policies, the platform collects security reports from mobile devices and keeps these logs for a specified retention time.

Degirmenci et al. [4] indicate that MDM solutions may evoke employees' concerns about their privacy protection. They note that some MDM solutions allow companies to track employees' locations, installed applications, and private data. APPJUDICATOR [9] is an MDM tool enabling network access control in Android devices. It extracts the UI information as an extra attribute used in the security policy. In experiments on Android emulators, the APPJUDICATOR authors show the UI-enhanced security policies outperform traditional policies that rely on information of network packets. However, the tool's use of end-user information may raise privacy concerns.

B. Privacy of User Data

When considering privacy, any "possible loss of privacy as a result of information disclosure" can be considered a privacy concern [23]. El Emam et al. [24] explore such disclosures in a set of "re-identification" scenarios in which an adversary gains knowledge about a single individual.

In a survey of 150 students who were asked questions about bringing their own devices to their classes [25], the authors found 70.2% of participants believed that the privacy of students' personal data may become vulnerable to computer espionage by college administrators or faculty.

[1]Throughout this work, we use "logs" to indicate the security data captured by solutions like APPJUDICATOR.

Birrell et al. [26] analyzed a set of 24 privacy laws and data protection regulations drawn from around the world and developed a taxonomy of rights granted and obligations imposed by these laws. Significant work focused on General Data Protection Regulation (GDPR) definitions of data anonymization [27]: anonymization of personal data is the process of encrypting or removing personally identifiable data from data sets so that the person can no longer be identified. Werthmann et al. [28] propose iOS policy enforcement to prevent privacy leaks.

K-anonymity is often used for data anonymization [24]. A k-anonymized data set has the property that each record is similar to at least $k-1$ other records for any potentially identifying variables. Therefore, attempts to re-identify any record in a k-anonymized data set have a maximum probability of $\frac{1}{k}$ of being re-identified. However, Machanavajjhala et al. [29] found that k-anonymity may leak information when there is a lack of diversity in the associated sensitive attributes: even if an adversary can only re-identify a target's record with probability $\frac{1}{k}$, if all k records have the same value for a sensitive attribute, an adversary can infer the target has that value for the attribute. They propose l-diversity to ensure l well-represented values for the sensitive attribute occur in every block. Li et al. [30] argue l-diversity is insufficient at protecting sensitive attributes when those attributes are distinct but semantically similar.

Dwork [31] proposes a Differential Privacy (DP) mechanism that adds crafted noise to create synthetic records in data sets. Ashena et al. [32] indicate that DP stands out among the technical solutions for privacy protection since it offers a mathematically-provable privacy guarantee. Küchler et al. [33] state that DP is recognized as superior for protecting personal information and provides a mathematically rigorous definition of privacy. Accordingly, we use DP noise injection [34] in our escrow service's responses.

Abd Razak et al. [35] use pseudonyms to ensure that unauthorized parties cannot predict the data's owner. Liu et al. [36] propose a fully uncoordinated approach where individual nodes can independently change their pseudonyms. Hlaing et al. [37] introduce a privacy-preserving system ensuring that patients' direct identifiers are not revealed to any adversary by masking them with a pseudonymized token value using SHA-256 hashing. We use pseudonyms to eliminate correlations between devices' identifiers and users' private information.

Giomi et al. [10] propose a unified framework using the success rate of concrete privacy attacks to quantify privacy risk in synthetic data. They demonstrate the framework on the three types of privacy attacks that GDPR requires anonymization techniques to defeat: singling out, linkability, and inference. Since BYOD users want to protect the privacy of their information, we use the Giomi framework in this work. Effectively communicating these privacy risks to end-users is also a challenge. Barth et al. [38] introduce a privacy rating tool for mapping and visualizing the privacy of online services using penalty points and privacy scores. We leverage these insights to evaluate and describe privacy risks.

Herrera et al. [6] introduce PADEC, which is a privacy-preserving access control system for mobile devices. PADEC allows users to define context-sensitive and privacy-preserving access control rules over users' resources. PADEC also keeps data sharing local to empower individual users to control what and how information is shared. However, PADEC has limitations in user transparency and organization visibility. PADEC focuses on privacy in the policy logic and ignores the user awareness of the privacy protection. In this work, we aim to build a user-visible scoring of privacy impact for each organization query on users' data. PADEC enables the organization access control server to directly receive user context data after privacy-preserving transformation. It is still possible for the organization server to conduct privacy attacks on users' data. In this work, we introduce an escrow service to store all user data. We consider to leverage this third-party service to ensure every organization query action is recorded and reflected on the privacy rating system.

III. Design: Goals and Quantified Privacy

Organizations need to enforce policy for BYOD devices to meet their security goals and they need situational understanding of what is happening on those devices. At the same time, device owners and users need autonomy over their devices [39] and to have privacy protections. To support both stakeholder groups, we create local policy enforcement and logging infrastructure that provides a real-time summary of activity to organizations while logging the detailed context with an escrow service for follow-up. In doing so, we have five key design goals:

Goal 1: Real-Time Situational Awareness: When a personally-owned device interacts with an organization's resources, the organization should have real-time insight into those interactions and their policy compliance status.

Goal 2: Private by Default: The detailed context and actions of an end-user should be kept private unless there is a specific need to reduce privacy protection.

Goal 3: Immediately Queryable Context: If the organization has a need to investigate the details of a situation further (e.g., in response to an incident or to investigate a potential incident), they must be able to obtain this potentially-private context immediately. Since threat detection and response is often time-sensitive, investigators should not have to wait for the device owner to authorize it or for the device to be online. Since such logging is a condition for organizations to grant a device access to resources, users cannot later deny access to the logs.

Goal 4: Quantifiable Privacy: Organizations and device owners should have trustworthy, quantifiable data, in the form of privacy metrics or scores, about how the organization treats private end-user data. Organizations must know the implications on privacy metrics/scores of every query they would make that would reveal private information. Device owners must be able to obtain these metrics and scores; organizations must not be able to illegitimately alter or deny access to the scores. Finally, the privacy metrics and scores must be

presented in the aggregate to avoid potentially compromising an organization's investigation.

Goal 5: Preserve Device Owner Control: The owners of a resource must have autonomy and control over that resource. For personally-owned devices, the owner (often the device's primary end-user) must retain authority over the device's settings and actions. Where an organization sets requirements for access to the organization's resources, 1) those requirements and implications must be obvious to the device owner, 2) the device owner must have to grant consent, 3) the device owner should know how that device's data is being used, and 4) the device owner should be able to revoke forward consent.

These goals ensure device owners and their associated organizations gain mutual accountability. We can model this as mutually distrusting parties (e.g., device owners desire protection from privacy-invading organizations, whereas organizations desire robust, auditable logs of policy violations).

Escrow arrangements are commonly used to resolve scenarios in which mutually-distrusting entities must cooperate to achieve their goals [40]. The parties designate in advance an escrow protocol for a neutral third-party, the escrow service, to enact for those parties. The escrow service needs to be available to both parties, so the escrow service operators would need to use standard best practices for resilience and redundancy to protect against availability failures. In our design, we will use such an escrow service for detailed log storage and to report query behavior against those logs. We use a Trusted Platform Module (TPM) and message signing to provide evidence of the escrow's actions, allowing organizations and users to identify if an escrow service ever operated maliciously (see Section IV-C).

IV. Approach: Local Policy and Escrow

To achieve our design goals, we modify an existing access control system, the AppJudicator system created by Liu et al. [9], to integrate privacy-preserving measures. Our modifications obtain policy from the organization and enforce that policy locally when accessing the organization's resources. The modified system constructs both a detailed audit log with in-depth context, which is encrypted and provided to a neutral third-party escrow service, and a less-detailed summary of policies enforced, which it sends in real-time to a control system at the organization. We now describe the existing AppJudicator system, the modifications we make to it, and the escrow and organization servers.

A. Overview and the AppJudicator System

In Figure 1, we provide an overview of the overall privacy-preserving BYOD access control system. We use color-coding to distinguish existing Android and AppJudicator components from the contributions we make in this work. The system components are: 1) the standard Android operating system and applications, 2) the AppJudicator endpoint access control system, 3) the privacy modules we add to AppJudicator, 4) the AppJudicator controller, 5) the modifications we add to the controller, and 6) the escrow service.

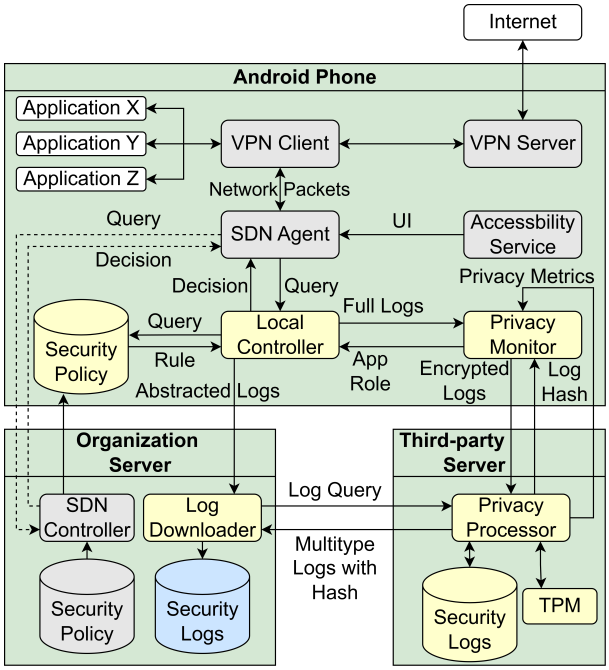

Fig. 1. The unshaded shapes show standard Android components. The original AppJudicator functionality is shown in gray shapes. The yellow shapes depict the privacy-preserving components we add in this work, including a local controller, a privacy monitor, an escrow service, and a logging system. The blue cylinder is an original AppJudicator component that we modify to enhance privacy.

The first component is the regular Android system and its software which may already be familiar to readers. We thus begin by briefly describing the AppJudicator system that we extend. Liu et al. [9] describe the details of the AppJudicator system they created in their publication. The AppJudicator tool essentially acts as an endpoint-based network access control system that obtains policy decisions from a controlling server run by an organization. Using a software-defined networking (SDN) protocol, it provides detailed context about the device's operating state to the controller. The AppJudicator system uses Android's `VpnService` API to intercept network traffic from all Android applications and collects UI events through Android's `AccessibilityService` API. It implements a local SDN agent to communicate with an SDN controller to enforce access control decisions. The AppJudicator developers gave us the system's source code and permission to modify both the endpoint and controller software to add the privacy functionality. This allows us to perform direct comparisons with the AppJudicator work in terms of performance overheads.

B. Endpoint Privacy Modifications

AppJudicator originally used a remote server, an SDN controller running at the organization, to make an access control decision for each new connection request. These decisions required the SDN agent to transmit all context surrounding the request to the SDN controller to determine which policy rules may be relevant and to render a verdict. While the

transmissions were encrypted, the system provides no insight to device owners about what the organization's SDN controller does with the data. This issue is common to MDM approaches and is the source of significant privacy concerns.

We first modified the APPJUDICATOR system to create a local SDN controller on the mobile device itself. As shown in Figure 1, we store a copy of the organization's security policy on the device for the local controller to query to make verdicts locally. This enables policy enforcement while the device cannot reach the organization server (e.g., behind a captive portal) and may decrease decision latency. We use Android's `LocalServerSocket` API for these queries.

The Local Controller approach enables access control while eliminating the privacy risk of providing the detailed context to the organization, achieving design goals #2 and #5 (while rendering goal #4 moot). However, it introduces new threat surfaces where could be infected by malware with root privileges and allow attacker to manipulate the security logs. It also fails to meet design goals #1 and #3 and the associated security goals for organizations.

To address these issues, we add functionality in the Local Controller to immediately transmit a summary of the activity. It transmits a record containing the device ID, the timestamp of the event, the identifier of the matching policy rule, and the verdict applied based on that policy rule. This meets design goal #1 since the organization immediately learns about the device's interactions with the organization's resources and whether they are allowed or denied by policy.

To achieve design goal #3 (immediately queryable context), we create a Privacy Monitor module. The Local Controller provides the Privacy Monitor with full contextual details about the requested connection, the associated policy rules, and decisions, including potentially-private information (e.g., the user's identity, packet headers and payload, and UI interactions).

The Privacy Monitor provides an interface for the end-user to control the APPJUDICATOR system (for design goal #5: preserve device owner control) and to learn about the organization's overall use of private data across devices (for design goal #4: quantifiable privacy). For applications marked as "private" by the user, the Local Controller denies access to any corporate resources listed by the organization's controller. The Local Controller will also not apply organizational policy or report logs for private applications.

For corporate or mixed applications, the Privacy Monitor will provide full raw log information and a "synthetic" data set to a third-party escrow server in an encrypted format. Consistent with Section V-B, by default, the Policy Monitor provides encrypted values for highly sensitive attributes (e.g., UI data associated with connection requests). For other information that could explicitly identify the endpoint device or user, the Privacy Monitor provides a hash function output that is consistent across records associated with a particular query transaction. This allows organizations to determine records associated with the same device or user without revealing the identity of that user or device. Finally, the Privacy Monitor employs a noise injection tool [41] to apply differential privacy techniques [42] on all attributes, including information of devices, applications, and policy enforcement.

The escrow service stores the data from the Privacy Monitor in a database. To prevent the escrow service from becoming a privacy risk, we use a Trusted Platform Module (TPM) on the escrow server. The Privacy Monitor encrypts all records with two symmetric keys: it first encrypts the data with a key shared with the organization and then with a generated symmetric key that is RSA-encrypted so it can only be extracted using the private RSA key associated with the TPM on the escrow server. The Privacy Monitor sends the encrypted symmetric key and encrypted full logs to the escrow service which stores them. Since the generated symmetric key is protected by the TPM, the escrow server must use the TPM to decrypt client data. Even then, the escrow cannot access the sensitive attributes since the requisite second round of decryption can only be done with the symmetric key held by the organization. Using the escrow API, the organization can query for more context via the escrow's TPM, supporting design goal #3.

C. Escrow Service and Organization Controller

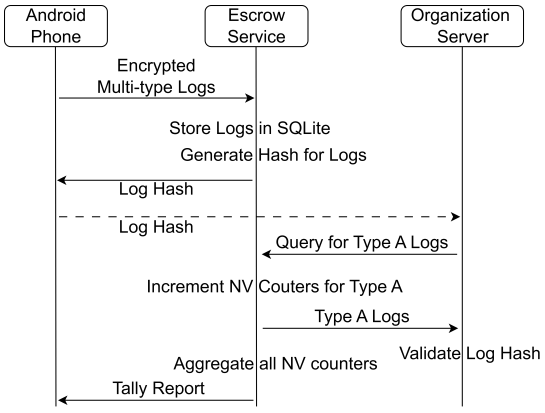

Fig. 2. This diagram illustrates how a user's device and the organization server interact with an escrow service. It uses a query for Type A logs as an example to show how different types of queries correspond to different levels of privacy exposure.

The Privacy Processor at the escrow server interacts with the encrypted security logs and queries from the organization. To detect any malicious escrow activity, we use a TPM at the escrow to create verifiable, tamper-resistant log records about the escrow's actions. The escrow uses the TPM to constructs Merkle tree–based logs to ensure that once the Processor logs a record or query interaction, that logged entry cannot be retroactively altered or deleted without detection. The Processor reports metrics for the privacy implications of the organization's queries to the organization and the endpoints, allowing them to audit the escrow service.

To implement the hardware-backed escrow model, we deploy the escrow service as a lightweight `Socket.IO` [43] server backed by `SQLite` [44] for persistent storage and `tpm2-tools` [45] for TPM management. The escrow uses endpoint-provided identifiers to determine the log entry type

and stores logs in separate `SQLite` tables based on the type. When servicing queries, the escrow can then assign penalties based on the requested log type. For each new log entry it receives, the escrow stores a record in an append-only Merkle tree. It sends the client a receipt containing the record hash and the Merkle root, allowing later auditing of the escrow service.

Upon receiving a query from the organization, the escrow server invokes `tpm2_rsadecrypt` to extract the symmetric key to decrypt the security logs. When a query is successful, the service appends an entry to the Merkle log and returns the decrypted data with a Merkle inclusion proof. If the query causes an error (e.g., no records found), it instead appends an entry with the error code and responds with a refusal code and proof. By comparing receipts and Merkle logs, the organization and client can detect any escrow data forgery.

When queried by an organization, the escrow increments the TPM's non-volatile (NV) counters (created using `tpm2_nvdefine`) for a global counter and type-specific counters for each returned query. The Processor also performs analysis on the privacy implications of the query's results using the same framework from Giomi et al. [10] (Section V-B). The Processor simulates an attacker and measures the attacker's rate of success at the singling out, linkability, and inference attacks based on the query results it provides to the organization. We use such success rates to quantify the privacy risks.

The organization must decide what queries to make to the escrow server. While the organization could simply query the Privacy Processor for the full raw data for the entire database, the resulting privacy metrics the escrow will publish will indicate a significant intrusion upon users' privacy. Instead, the organization can issue queries designed to explore an incident without revealing identities or the sensitive attribute data. The more privacy-preserving features the organization allows the Privacy Processor to apply when returning the results, the lower the penalties on privacy metrics. Once the organization has narrowed the scope of the needed data, it can query only for that unmasked data for a lower privacy penalty.

We can use these metrics to provide an overall privacy score. Barth et al. [38] proposed a privacy rating tool that categorizes online services into seven classes, from the level of lowest privacy risks to the level of highest privacy risks. We use a similar approach to evaluate the end-users' privacy level by using our BYOD privacy metric scenarios. The escrow provides a tally report at fixed intervals. Each tally report aggregates the TPM NV counters, the computed privacy penalty, and the current Merkle log root. We sign these reports with `tpm2_quote` using the TPM's attestation key (created via `tpm2_createak`). This signature and attestation allow recipients to verify that the report comes from the TPM and reflects the escrow server's state at the reported time. We describe our overall privacy score approach in Section VI.

V. System Trust Model and Privacy Risk Analysis

This section defines the adversarial models considered in our work. We separate threats targeting the trusted computing base from privacy risks arising through organization queries.

A. Threat Model and Trusted Computing Base

In all endpoint security tools, including MDM and MAM, defenders have little choice but to include the Android OS and the security tool's application in their trusted computing base (TCB). An untrustworthy OS or security application could ignore security controls and policy. They likewise could fabricate or alter any logging data they produce. The AppJudicator system uses this same trust model. In our case, we extend the AppJudicator application to include the local Security Policy and the Local Controller.

The TCBs often process detailed and sensitive organizational policy settings. For MDM configurations, such a policy may be transferred proactively. For SDNs, the policy may be obtained reactively. Our modifications require a proactive policy transfer to the device, unlike AppJudicator. If the TCB remains trustworthy, this change has no impact on the effective security of the system. However, if the TCB becomes compromised, a proactive transfer would allow an adversary to gather a comprehensive understanding of the organization's policy rather than the incremental policy obtained reactively.

Some organizations do not consider the confidentiality of their policies to be a security goal. Halim et al. [46] note that some organizations choose to disclose their BYOD security policies to device owners in online documents. Organizations that do want to preserve policy confidentiality may choose to implement redundant controls by deploying lower-risk policy controls to mobile endpoints and other confidentiality-sensitive policies on server endpoints. This division can limit the impact of policy disclosure in the mobile device's compromised TCB.

B. Quantifying Privacy Threats and Risks

Under our threat model, we explore the research question: *Can the privacy-preserving access control system fulfill the security requirements of mobile devices?* By building on the original AppJudicator design [9], we retain the same allow-list policy enforcement when shifting to an on-device controller. Accordingly, we focus on the privacy risks that may arise during user logs disclosure to the organization.

Giomi et al. [10] propose the Anonymeter framework that quantifies privacy risks in synthetic data by evaluating the success rates of privacy attacks under the GDPR's [27] three key indicators of anonymization: risks for singling out, linkability, and inference. As Anonymeter provides measurable privacy risks, which is aligned with our goal #4, we leverage their framework and apply it to BYOD systems to provide metrics for organizations and device users.

Unlike the previous subsection, we assume correct system operation and focus on user information leakage via potential privacy attacks from the organization. Since such privacy risk evaluations are attack-based, the measured risk should be interpreted as a lower bound on privacy risk [10], while DP [31] provides complementary worst-case upper bounds. Attack-based approaches have become state-of-the-art for establishing lower bounds of privacy leakage in several domains, such as machine learning [47]. To the best of our knowledge, our work

is the first to adopt such an attack-based framework to quantify privacy risks in user data of MDM solutions.

We describe the framework's principles and how we apply it to BYOD devices. The endpoint device provides a comprehensive set of the real data to a third-party escrow service in an appropriately-encrypted format. The escrow service provides a view of this data to the organization via query responses. That view of data is called a "synthetic data set" since some privacy features may mask or encrypt records or provide false additional records to achieve privacy goals. The organization may use a query type that will provide accurate access to the underlying real data set, albeit at a heavy privacy penalty cost.

1) Singling Out Attack: This attack happens when there is a single data record, within the original dataset, that can be deduced with a unique combination of one or more given attributes [48]. The mobile device transmits its full-context, original security logs, denoted D_{orig}, to the third-party escrow service. From this, the escrow service derives a synthetic data set, D_{syn}, that the organization can access. Using the ANONYMETER framework for the singling out attack, the attacker's goal is to formulate predicates that can isolate a single record in D_{orig} using only observations from D_{syn}.

Using the ANONYMETER framework, our attacker's approach starts by randomly sampling a record, x, from D_{syn}. The attacker selects a subset of categorical attributes, denoted $\{a_1, ..., a_m\} \in A$. The attacker then constructs a logical AND condition combining multiple categorical attributes:

$$P_x \leftarrow (a_1 == x[a_1]) \wedge (a_2 == x[a_2]) \wedge ... \wedge (a_m == x[a_m]) \tag{1}$$

where P_x represents the predicate for the random record x, a_1 represents the value of the first categorical attribute selected by the attacker and $x[a_1]$ represents the value of the first attribute of record x. The attacker then evaluates each predicate and records those that yield a single match in D_{syn}.

We evaluate this attack by applying each valid predicate to the D_{orig} data set. We then declare an attack successful if a predicate isolates a unique record in D_{orig}. We measure the overall success rate as:

$$P^{singling} = \frac{|\{P_x \text{ isolates a unique record in } D_{orig}\}|}{|P|} \tag{2}$$

where $|P|$ is the number of attack predicates generated.

Given the above, we can determine if a synthetic view, D_{syn}, has any records that are vulnerable to a singling out attack. In practice, we can test singling out risks in the synthetic view, D_{syn}, via sampling. For example, we can randomly sample 1,000 target records from D_{syn}, perform the singling out attack, and determine the number that can effectively be singled out as the success rate.

2) Inference Attack: This attack happens when an attacker can confidently infer the value of an unknown attribute of the original data record [48]. The original APPJUDICATOR system collects detailed data about the device's UI and the user's inputs. We treat that UI context as a secret attribute. For the inference attack, we define the attacker's goal to be the inference of the secret attribute S_{x_t} of a targeted record, x_t, by searching for its closest match in D_{syn} using a nearest neighbor approach.

We begin with a synthetic dataset D_{syn} generated from an original dataset D_{orig}. The targeted record, x_t, which is selected from a set of targeted records, X, is composed of auxiliary attributes, $\{a_1, a_2, \ldots, a_n\} \in A$, combined with the UI context being secret attribute S_{x_t}, i.e., $x_t = (a_1, a_2, \ldots, a_n, S_{x_t})$. The adversary has the ability to access everything in the synthetic records except for the secret attribute of the target record. In other words, the adversary has knowledge of auxiliary attributes $\{a_1, a_2, \ldots, a_n\}$ for x_t and all attributes including the secret attribute $\{a_1, a_2, \ldots, a_n, S_{x_u}\}$ for $x_u \forall u \in D_{syn}$ such that $u \neq t$.

With this setting, the attacker can then use a methodology based on the nearest neighbor search algorithm. Since the quasi-identifier subspace may contain heterogeneous attribute types (e.g., numerical and categorical features) and the Gower coefficient [49] normalizes numerical attributes and handles categorical mismatches in a bounded manner, the attacker uses it to measure the distance $d(x_i, x_j)$ over the subspace between two records, i and j. For each target x_t, the attacker finds the target's closest synthetic record $x_{closest}$ such that $x_t \neq x_{closest}$:

$$x_{closest} = \arg \min_{x_i \in D_{syn}} d(x_t, x_i) \tag{3}$$

The attacker then extracts the secret attribute $S_{x_{closest}}$ from $x_{closest}$.

We evaluate the success rate of the attack by modeling an attacker that guesses $\hat{S_{x_t}} = S_{x_{closest}}$. In other words, the attacker guesses that the value of the secret attribute for x_t is the same as that attribute's value in the closest record, $x_{closest}$. The attacker's inference is successful if the attacker's guess exactly matches the true value of the secret attribute for record x_t. The success rate is measured over the target records in X:

$$P^{inference} = \frac{|\{x_t \in X \mid S_{x_t} = \hat{S_{x_t}}\}|}{|X|} \tag{4}$$

As with the prior attack, we test inference risks in the synthetic view, D_{syn} via sampling. For example, we can randomly sample 1,000 target records from D_{syn}, perform the inference attacks, and compute a success rate as the ratio of correct guesses to total number of guesses. This is determines the attacker's success with their top guess, but the approach can generalize to allow for top n matches.

We consider the UI contextual data to be our sensitive attribute, but other information could likewise be designated as sensitive in the inference analysis.

3) Linkability Attack: This attack happens when it is possible to link together two records in different data sets belonging to the same individual [48]. With our modifications to the APPJUDICATOR system, it will provide real-time logs that abstract details of the application context while also providing detailed context to the escrow service. We denote these tables E and F, respectively. In this work, we define the attacker's linkability attack goal to be to correctly match any record in

table E with its corresponding record in table F using the synthetic dataset, D_{syn}.

Using the ANONYMETER framework, we formulate the setting as a table E that contains attributes A of BYOD systems from an original dataset D_{orig}. We denote a second table, F, as having a set of attributes B of the same BYOD systems from D_{orig}, but without explicit identifiers. We construct a synthetic dataset D_{syn} from the original dataset D_{orig}. This data set contains attributes from both E and F. Finally, an adversary with access to table E wants to identify corresponding records in table F using D_{syn} as a bridge.

Neighbor searching is key to the linkability attack. The attacker selects a set of target records X from table E. For each record, $x_t \in X$, the attacker finds the closest synthetic record $x_{closest}$ in D_{syn} based on the known attributes A:

$$x_{closest} = \arg \min_{x_i \in D_{syn}} d(x_t, x_i) \tag{5}$$

where $d(x_t, x_i)$ uses the Gower coefficient [49], which fits well with heterogeneous attribute types, to determine similarity between the target record and synthetic records. After identifying $x_{closest}$, the attacker retrieves its corresponding table F attributes from $D_{syn}[:, B]$, where $D_{syn}[:, B]$ represents the selection of only attributes B from D_{syn}. $O(x_t, x_{closest})$ represents a linkability outcome between x_t and $x_{closest}$ and is determined as:

$$O(x_t, x_{closest}) = \begin{cases} 1, & \text{if } x_{closest} \text{ contains correct attributes for} \\ & x_t \text{ from table } F \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

The attacker's success rate, P^{link}, is measured as:

$$P^{link} = \frac{|\{O(x_t, x_{closest}) = 1\}|}{|X|} \tag{7}$$

As before, we can determine the risk of linkability in D_{syn} using sampling. For example, upon randomly sampling 1,000 records in table E, we can determine the attacker's success rate of linking record attributes in table F.

VI. PRIVACY EVALUATION

We aim to explore the research question *To what extent can the privacy-preserving mechanisms inform mobile device users about the use of their private data?* by quantifying the privacy risks in mobile devices' access control security logs. An ideal exploration would collect such data for a real-world deployment across multiple organizations. However, there are significant obstacles for arranging such a study. To validate any findings, we would need the ground truth data logs associated with the escrow and organization servers, raising the same privacy issues that our approach aims to combat. There are further scaling challenges in recruiting organizations and users (e.g., obtaining data sharing agreements).

Given that, we explore an approach where we fuse two real-world data sets into a synthetic data set that allows us to roughly gauge the implications for real usage. We fuse existing public data sets on network traffic from two networks with the APPJUDICATOR logs we obtain from a set of Android emulator device instances running on the Internet. This fusion allows us to explore the real traffic patterns and device activity from those networks while obtaining the real APPJUDICATOR context that would be associated with application traffic from Android devices. At the same time, there is a loss of fidelity in that we do not know what applications or operating systems the production network devices were running, and thus the application context is unlikely to match. Since this study is to explore the extent to which the queries can affect the privacy of the devices, this mismatch may not be significant.

For our network data sets, we use two public network packet captures. The first is a packet capture from the 4SICS Geek Lounge [50], which captures the industrial network traffic at the ICS Lab during a conference. From that data set, we use the records of network traffic from a weekday that contains 65 IP addresses and 2,253,164 IP packets. The second data set tracks the first four weeks of corporate history of the M57 Patents company [51]. We use the records for a weekday that contains 559 IP addresses and 571,505 IP packets.

To collect real data from the APPJUDICATOR system with our privacy modules, we run a suite of Android Studio emulator instances, each emulating a Pixel 4 with version 31 of the Android API. On each, we install APPJUDICATOR with our Local Controller and our Privacy Monitor. To generate traffic and UI context, we simulate end-users' operations using `Appium` [52], an automated testing tool. We execute a set of typical workflows for a set of popular applications that may have mixed personal and corporate usage. The tested applications were the Chrome web browser, the YouTube video application, the LinkedIn social networking application, and the Terminus SSH client. We use these workflows to search and access the top 10,000 worldwide domains, as determined by the Cloudflare site [53]. The APPJUDICATOR system collects 139,912 records in its security logs, including information about the devices, users, and applications.

We create the 4SICS experimental data set by merging that organization's network data with the APPJUDICATOR data. We construct the M57 data set by merging its network data with the APPJUDICATOR data. For each, we extract the network packet information, including the network header, transport header, and transmission timestamp, from the associated organizations' packet captures and store them in a table. We then collect all the security logs generated from APPJUDICATOR and store them in a second table. We then proceed to attach each row of the network packet table with an entry in the security log table in a sequence, concatenating the network data and the APPJUDICATOR context into a new record. We use affinity in our merging so that a given source IP address in the data set is linked consistently with the same APPJUDICATOR table interactions to emulate consistent behavior for each machine. This merged data set becomes our original data set, D_{orig}, in our subsequent analysis.

We transform the D_{orig} data set into a queryable synthetic data set, D_{syn}, using the techniques from Section IV-B:

- **Type A (encrypted UI):** We use AES-128-GCM to

encrypt the UI context, which is considered sensitive.

- **Type B (masking and encrypted UI):** We extend the Type A protections, mask the device identifier with hashing, and coarsen the timestamp to a 24-hour period.
- **Type C (adding differential privacy):** We extend the Type B protections and include records from a noise injection tool [34] to enable central differential privacy. That tool uses the Differentially Private Conditional Tabular Generative Adversarial Networks (DPCTGAN) [54] approach that was proposed for medical records. A lower value (e.g., 0.1 or 0.5) for the amount of noise injected, ε, produces stronger privacy protections but noisier data sets. Conversely, a higher ε value (e.g., 5 or 10) weakens privacy but provides cleaner data. We use $\varepsilon = 2$ as a representative configuration point.

In Table I, we show the privacy protections by D_{syn} log type from the APPJUDICATOR system. Even for the noise records in Type C logs, we apply masking and encryption to avoid creating a signal for an adversary to detect noise records.

TABLE I
SYNTHETIC D_{syn} LOG TYPES AND ASSOCIATED PRIVACY

	Type of Synthetic Logs		
Attributes	Type A	Type B	Type C
Device ID	Original	Masked	Masked
Policy ID	Original	Original	Noise Added
Timestamp	Original	Masked	Masked
Judgment Verdict	Original	Original	Noise Added
Host Name	Original	Original	Noise Added
Transport Port	Original	Original	Noise Added
UI Context	Encrypted	Encrypted	Encrypted
Application Name	Original	Original	Noise Added

We launch the privacy attacks described in Section V-B on each of these synthetic security logs for each of the two real-world data sets to determine their privacy impacts. For both the inference attack and linkability attacks, we randomly choose 1,000 entries as a set of target records. For the singling-out attack, we try to maximize the number of predicates. In Table II, we show the success rates of these attacks. With the weakest form of privacy protection, Type A, all the privacy attacks achieve high success rates. This shows that simply encrypting the sensitive attribute, in this case the UI context, is insufficient to provide robust privacy. When adding the Type B protections, namely device ID masking and coarse-grain timestamps, the success rates for the inference attack and linkability attacks drop significantly. The intuition behind this outcome is that the masking approach essentially hides the distance between values, eliminating the effectiveness of the attacker's strategy of finding the closest neighboring value. Unfortunately, for the singling out attack, the adversary can use the record attributes that are not masked to isolate individual records. The differential privacy approach for Type C records significantly reduces the attack success rates. The injected noise records essentially eliminate the attacker's ability to generate a unique combination or find authentic records with highly-similar values. These results show the differential privacy approach contributes most to the privacy preservation. While prior research [55] shown that using differential privacy would affect the utility of data and the accuracy of models, our method still supports the organization to query for raw logs with highest privacy penalty.

TABLE II
THE INTRODUCTION OF ADDITIONAL PRIVACY-PRESERVATION MECHANISMS GENERALLY DECREASES THE ATTACKER'S SUCCESS AT PRIVACY ATTACKS ON THE SYNTHETIC SECURITY LOGS.

	Success Rates of Attacks					
	Singling-out		Inference		Linkability	
Log Type	4SICS	M57	4SICS	M57	4SICS	M57
Type A	83.2%	69.9%	57.0%	54.6%	68.4%	32.1%
Type B	54.4%	69.1%	16.6%	28.4%	4.7%	3.6%
Type C	0.1%	0.2%	0.0%	0.0%	0.0%	0.0%

While the attack success rates can characterize the privacy risks when an organization queries against different types of synthetic records, it can be difficult for organizations and users to interpret such data. We aim to distill these interactions into a summary. Concretely, we create a set of privacy metrics that define penalty points for each type of access request the organization makes to the synthetic security logs. We generate the penalty points using the average of success rates of privacy attacks on each type of log. We assign the highest penalty for requests for raw logs since they lack any privacy protections. We further set a minimum penalty point for accessing even Type C logs, despite their relatively low success rates in privacy attacks, since differential privacy's benefits can be weakened by carefully crafted query flooding attacks [56].

Our goal is to take the history and request volume for each of the different types of log queries into account, weighting each by the privacy risks associated with each type of log. We use a time windowing approach using a 24-hour granularity and an aging scheme to derive a value representing the history of requests. We weight the most recent requests more heavily than requests from older requests. Given a time series of requests $\{R\}$, we calculate a set of aged values $\{V\}$ iteratively at each timestep. We use the equation $V_t = \alpha R_t + (1-\alpha)V_{t-1}$, where V_t is the aged value at time t, R_t is the actual number of requests at time t, and α is a smoothing factor (we use $\alpha = 0.6$). We set the value of the initial condition to be $V_0 = 0$. We calculate a different set of aged values, V, for each of the different types of logs, including the unprotected logs. We denoted these sets as V^A, V^B, V^C, and V^R for Types A, B, C, and raw logs, respectively.

We calculate the total penalty points at time window t using a weighting factor on the aged value associated with that time window. For Type A logs, we multiply that data set's aged value at time t (i.e., V_t^A) by the weight of 60.8. For Type B logs, we use a factor of 29.5. For Type C, we use a factor of 0.1. We determine these weights based on the average of success rates of three privacy attacks on each type of synthetic logs, as shown in Table II. Since all the privacy attacks will succeed when accessing the unprotected logs, we set a factor of 100.0 for those raw logs. Accordingly, the total point penalty at time t, $P_t^{penalty}$ is thus $P_t^{penalty} = (60.8 \times V_t^A) + (29.5 \times V_t^B) + (0.1 \times V_t^C) + (100.0 \times V_t^R)$.

Our Privacy Processor responds to requests from each endpoint's Privacy Monitor for information about the organization's privacy impact. The Privacy Monitor can generate a privacy report to show end users the privacy level at different points in time. Using the same schema as Barth et al. [38] for their privacy rating tool, we can characterize privacy risks across seven classes. Using non-inclusive upper bounds, we assign a low-risk 'A' label to penalty scores under 10, a 'B' to scores of 10-30, a 'C' to 30-60, a 'D' to 60-100, an 'E' to 100-150, an 'F' to 150-200, and a highest-risk 'G' to scores 200 and above. The Privacy Monitor can share these labels with users to effectively summarize the privacy risks. Barth et al. [38] ran a user study that found that such labels of privacy risks appeared to be understood and useful for lay users.

VII. Performance Evaluation

We explore the research question: *To what extent would such a privacy-preserving module influence the performance of Android devices?* We compare the performance of Android devices installed with the original AppJudicator and the privacy-preserving AppJudicator in terms of network delays and computational resource usage.

A. Experiment Setup

The experiment setup includes a simulated Google Pixel 4 device running Android API level 30 with 4 CPU cores and 2 GB of RAM, a remote Ubuntu 20.04 server hosted on our organization public network, and a laptop with a embedded TPM 2.0 security chip, eight 1.8 GHz cores, and 16 GB of RAM. The Android emulator and the laptop are connected with a local router with Internet access via Wi-Fi. We implement the AppJudicator on the emulator, the organization controller on the Ubuntu server, and the escrow service on the laptop.

B. Network Delay

We first analyze the end-to-end round trip time (RTT) for packets exchanged between an application in the Android emulator and the remote Ubuntu server. This measures the per-packet overheads for established connections. The remote server acts as a testbed for generating and receiving network traffic. The server periodically transmits packets with specific payloads to the emulator application, which responds by echoing the packets back. We record the total RTT in three scenarios: without any AppJudicator running, with the original AppJudicator running, and with our privacy-preserving AppJudicator running. To measure the delays on the device side, we use a timer on the remote server to measure the RTT between sending a request and receiving the echoed response. We send packets at a rate of 50 requests per second and collect 1,000 trials.

We show the results in Table III. When AppJudicator is disabled, the median RTT is 15.8 ms with a standard deviation of 31.3 ms. These values reflect typical network delays between the emulator's local network and the remote server. In contrast, the original AppJudicator results in a median of 19.7 ms, and the privacy-preserving AppJudicator results in a median of 21.5 ms. The privacy-preserving features add around 1.8 ms to the median RTT of the AppJudicator.

TABLE III
End-to-end RTT for normal Android, the original AppJudicator, and the privacy-preserving AppJudicator

	RTT (milliseconds)			
Scenario	10th	50th	90th	std. dev.
No AppJudicator Running	15.3	15.8	19.6	31.3
Original AppJudicator	17.7	19.7	23.3	32.9
Privacy-Preserving AppJudicator	19.4	21.5	25.6	30.5

We next explore the impact of the AppJudicator modifications on new network connection requests. Specifically, the local SDN agent is designed to send packet query requests to the SDN controller for new network flows and to install processing rules based on the controller's response. We conduct experiments to compare the time required to query the remote organization SDN controller with the time for the on-device SDN controller. We host the organization controller on the same Ubuntu server from the previous experiment. We conduct 1,000 trials with an `Appium` script making requests to randomly-selected domains.

In Table IV, we see the impact the on-device controller has on query time. Unsurprisingly, the on-device controller responds faster since it does not require the network propagation time to the remote controller, leading to 90% of requests being processed within 2.7 ms. The on-device controller can asynchronously log the events with the organization.

TABLE IV
Query response time by percentile for different controllers

	Response Time (milliseconds)			
Controller	10th	50th	90th	std. dev.
Organization Controller	19.5	26.0	40.7	11.5
On-device Controller	0.9	1.3	2.7	0.9

We further evaluate the performance of the escrow service. We implement the escrow service in the TPM-backed laptop. The escrow service is designed to send a privacy report to the emulator and the controller periodically. We consider to measure the response time of the privacy report by recording the time difference between the escrow's sending the report and receiving the acknowledgment response. We let the escrow service send 1,000 privacy reports in the rate of one report per second to the emulator and controller respectively. The results of response time are shown in the Table V. The response time of privacy reports are less than 45 ms for 90% trials. The response time for the Android emulator is less than the organization controller as the emulator is in the same local network with the escrow service, while the emulator's response time is less stable than the controller.

C. Device and Escrow Computational Resource Usage

With limited computing resources on mobile devices, it is important to assess the usage of such resources. To evaluate the performance of the privacy-preserving AppJudicator,

TABLE V
RESPONSE TIME OF PRIVACY REPORT FROM THE ESCROW SERVICE TO THE ORGANIZATION CONTROLLER AND THE ANDROID EMULATOR

	Response Time (milliseconds)			
Escrow Service to	10th	50th	90th	std. dev.
Organization Controller	35.9	39.5	44.6	2.1
Android Emulator	7.6	8.7	20.1	8.6

we analyze its battery usage, memory consumption, and CPU utilization using an Android emulator while streaming data.

For data collection, we employ Android Studio's `Profiler` [57] and the `top` tool. The experiment is designed to mimic a common streaming use case. We first launch APPJUDICATOR on the emulator and then play music in the YouTube Music application to stream content from the Internet. This setup require continual network activity, reflecting real-world conditions. We record CPU usage data for both APPJUDICATOR and YouTube Music every second using the `top` tool, spanning a test duration of 1,000 seconds. We repeat this process for both the original APPJUDICATOR and the privacy-preserving APPJUDICATOR. We connect the original APPJUDICATOR with a remote SDN controller and the privacy-preserving APPJUDICATOR uses a local controller and connects with the escrow and organization.

We show the results in Table VI. The privacy-preserving APPJUDICATOR demonstrates stable memory usage, with a median consumption of 70.8 MBytes, significantly lower than YouTube Music's median of 289.3 MBytes. It also uses less memory than the original APPJUDICATOR with a median comsumption of 132.7 Mbytes; this is likely due to our privacy-preserving implementation replacing Java code with a more memory-efficient Kotlin implementation.

For CPU usage, the privacy-preserving APPJUDICATOR consumes a median of 48.0% of one of the four cores, compared to the original APPJUDICATOR's 12.0% and YouTube Music's 29.0%. The privacy-preserving module enforces security policy locally and has extra privacy-preserving steps prior to communicating with both the organization server and the third-party server, which incurs more CPU utilization. With the emulated device having four CPU cores, provide a combined capacity of 400%, the privacy-preserving APPJUDICATOR's CPU usage even in streaming usage leaves ample resources available. The Android Studio `Profiler` classifies the privacy-preserving APPJUDICATOR's energy consumption as "light." While the privacy-preserving APPJUDICATOR implementation may have optimization opportunities, its resource usage is compatible with popular Android workflows.

To evaluate the runtime overhead of the escrow service, we measure its computing utilization using stress tests. The service is deployed on the same laptop using `uvicorn` [58] and monitored at one-second intervals using `pidstat` to record per-process CPU and memory usage. We generate concurrent client workloads using an asynchronous `Socket.IO` test with three volume configurations: low volume (10 workers, 100 operations each), moderate volume (50 workers, 200 operations each), and high volume (100 workers, 500 operations each). Each worker issues requests continuously and each test completes within approximately 10 seconds, ensuring comparable observation windows across volume levels. The workload consists of a mixture of `escrow_store` requests (encrypted record insertion and audit log updates), `escrow_decrypt` requests (TPM-based key unwrapping, decryption, and counter updates), and periodic `tally_report` requests (reading TPM-protected counters and generating signed reports).

TABLE VI
COMPARISON OF CPU AND RAM USAGE IN THE ORIGINAL APPJUDICATOR, THE PRIVACY-PRESERVING APPJUDICATOR, AND THE YOUTUBE MUSIC APPLICATION. THE DEVICE HAD FOUR CORES.

		Percentile of Trials		
Application	Resource	10th	50th	90th
APPJUDICATOR (original)	Single CPU core use (%)	11.0	12.0	14.0
	RAM usage (MBytes)	130.7	132.7	132.7
APPJUDICATOR (privacy-enhanced)	Single CPU core use (%)	43.0	48.0	54.0
	RAM usage (MBytes)	68.9	70.8	70.8
YouTube Music	Single CPU core use (%)	24.0	29.0	39.1
	RAM usage (MBytes)	281.4	289.3	295.2

As shown in Table VII, across all request volumes, the CPU utilization remains modest. Under low volume, the average CPU utilization is 2.8% with a peak of 7%. Under moderate volume, the average utilization increases to 5.0% with a peak of 17%, while under high volume it reaches an average of 8.3% with a peak of 24%. Since utilization is reported relative to a single core, even the highest load consumes less than one quarter of a single CPU core on the eight-core system. Memory usage increases gradually with load but remains stable throughout execution, with an average of 66.3 MB under low volume and 94.4 MB under high volume. These results indicate that even with high query volume, the escrow service computational overheads are unlikely to be a bottleneck.

TABLE VII
COMPARISON OF CPU AND RAM USAGE OF THE ESCROW SERVICE WITH VARYING TRAFFIC VOLUME. THE DEVICE HAD EIGHT CORES.

Request Volume	Resource	Average	Peak
Low	Single CPU core use (%)	2.8	7.0
	RAM usage (MBytes)	66.3	66.8
Moderate	Single CPU core use (%)	5.0	17.0
	RAM usage (MBytes)	80.9	81.2
High	Single CPU core use (%)	8.3	24.0
	RAM usage (MBytes)	94.4	98.6

VIII. CASE STUDIES VIA APPLICATION WORKFLOWS

The human-computer interaction community has leveraged user scenarios for decades to explore the impacts of a system on users without heavier-weight user studies [59], [60]. With the inherent challenges of deploying the technique at scale, we evaluate the proposed privacy-enhancing method via common application user scenarios. We install the privacy-preserving APPJUDICATOR on a physical Moto G Power Android phone with eight 2300 MHz cores and 4 GB of RAM. We use the phone in a residential apartment setting. We run the privacy-preserving APPJUDICATOR with a local controller in the phone's background. We configure the phone to use an escrow

service hosted on an external server. We install four popular applications: Chrome, YouTube, LinkedIn, and Termius.

In our experiments, we manually navigate applications on the Moto G Power phone according to a defined use case. The operator interacts with each of the four described applications for 15 minutes. During the interactions, the operator randomly chooses a series of websites from the top 100 worldwide domains [53] and tries to search and access the related information. The APPJUDICATOR system collects and transmits its data to the escrow and to the organization. We collect the data sent to the escrow service and the organization's controller and apply the three privacy attacks from Section V-B. We first incorporate the detailed data generated by the physical phone into the 4SICS experimental data set described in Section VI. This merged data set becomes our original dataset, D_{orig}, in our subsequent analysis. We apply the Section IV-B techniques to transform the D_{orig} data into a synthetic data set, D_{syn}.

A. Case Study I: Privacy Attacks on Multi-types User Data

To explore the privacy risks in the usage scenario, we apply three privacy attacks on the synthetic data set, D_{syn}, using the Type A, B, and C queries at the escrow. In the singling out attack, the attacker tries to formulate predicates of queries that can isolate a single record in D_{orig} using only observations from D_{syn}. The goal of inference attacks is to infer a targeted record's secret attribute, defined as the users' UI context, by querying for its closest match in D_{syn} using a nearest neighbor approach. The attacker's linkability attack goal is to correctly match any query record in the table of abstracted data with its corresponding record in the table of detailed data using the synthetic dataset. The organization's synthetic data set only includes abstracted data, so its log cannot be used to uncover sensitive attributes, making the inference and linkability attacks not applicable (N/A).

Table VIII shows the privacy-preservation mechanisms are effective. The encryption technique in Type A logs prevents attackers from simply inferring users' UI context. The masking of the device ID and timestamp in Type B logs avoids the linkability issues between two separated data tables. The differential privacy in Type C logs protects against all three privacy attacks. The abstracted organization logs lack predicates that could effectively isolate a single record. This use scenario confirms the utility of the privacy-preserving measures.

TABLE VIII
THE SUCCESS RATES OF PRIVACY ATTACKS ON MULTI-TYPE LOGS

Receiver	Log Type	Success Rates of Attacks		
		Singling-out	Inference	Linkability
Escrow	Type A	87.3%	57.8%	67.4%
	Type B	61.5%	31.5%	4.2%
	Type C	5.6%	1.7%	0.0%
Organization	Abstracted	0.0%	N/A	N/A

B. Case Study II: Privacy Impacts of Organization Queries

To demonstrate the practical utility of our privacy-impact awareness method, we conduct a case study in which an organization decides to ban access to a specific network destination (e.g., an external website or service). We evaluate the privacy impact of performing the necessary investigations to support the ban decision on the synthetic data set, D_{syn}.

When the organization initiates a ban on a sample shopping website, which we anonymize as `example.com`, its IT operators would issue a series of queries to determine how frequently the destination has been accessed, by which applications, and by which devices. To first determine the frequency, the IT operators could use a query like `SELECT COUNT() FROM Type_C_Table WHERE Domain = 'example.com'` to get the number of specific logs. That query would return 9 results from the 39,650 rows in the Type C log table, with a frequency of 0.0227%. Since the Type C logs include noise for differential privacy, we determined the ground truth, which was 6 matches with an actual frequency of 0.0151%. From the IT operator's perspective, the difference between the actual frequency and the Type C result has minimal influence on the data's utility.

To further determine the application accessing that domain, the IT operators need to make a query to the Type B logs to avoid noise records. The Type B logs would display the original application name, as shown in Table I. The operators could execute the query `SELECT Application_Name FROM Type_B_Table WHERE Domain = 'example.com'`. This would give the IT operators the actual application name of `com.android.chrome`. If IT operators needed the specific devices accessing the domain, the IT operators would need to make a query to the Type A logs, since the Type B logs mask the device ID field. With a query like `SELECT Device_ID FROM Type_A_Table WHERE Domain = 'example.com'`, they would receive the actual device ID.

In our case study run, the total penalty score during the full investigation period peaked at $P_t^{penalty} = (60.8 \times 1) + (29.5 \times 1) + (0.1 \times 1) = 90.4$, which corresponds to a moderate-risk 'D' label in privacy rating. In making the queries, the IT operators could determine whether the Type A query was vital to their purposes. If they decided it was unnecessary, the penalty for the Type B and C queries would have been only 29.6, earning a lower-risk 'B' privacy rating. This case study shows our method can quantify the privacy impact and map it to a human-interpretable privacy risk class. It shows how the privacy-preserving APPJUDICATOR can enable operational security teams to balance the privacy costs of different queries, leading to end-user transparency about privacy practices.

IX. CONCLUSION

By integrating a local SDN controller and a privacy monitor, the privacy-preserving APPJUDICATOR addresses critical privacy concerns while maintaining robust security. The local controller enforces policies without continuous connectivity. The privacy monitor quantifies privacy via metrics from BYOD logs. Evaluation results demonstrate that the enhanced APPJUDICATOR effectively balances security and privacy, with minimal impact on Android device performance.

References

[1] C. Sørensen and J. S. Landau, "Academic agility in digital innovation research: The case of mobile ict publications within information systems 2000–2014," *The Journal of Strategic Information Systems*, vol. 24, no. 3, pp. 158–170, 2015.

[2] U. Goyal and G. Sachdeva, "BYOD and the evolving workplace," in *Integrating Cutting-Edge Technology Into the Classroom*. IGI Global, 2024, pp. 15–38.

[3] J. Lee Jr, M. Warkentin, R. E. Crossler, and R. F. Otondo, "Implications of monitoring mechanisms on bring your own device adoption," *Journal of Computer Information Systems*, vol. 57, no. 4, pp. 309–318, 2017.

[4] K. Degirmenci, M. H. Breitner, F. Nolte, and J. Passlick, "Legal and privacy concerns of byod adoption," *Journal of Computer Information Systems*, pp. 1–12, 2023.

[5] A. T. Ayedh M, A. W. A. Wahab, and M. Y. I. Idris, "Systematic literature review on security access control policies and techniques based on privacy requirements in a byod environment: State of the art and future directions," *Applied Sciences*, vol. 13, no. 14, p. 8048, 2023.

[6] J. L. Herrera, H.-Y. Chen, J. Berrocal, J. M. Murillo, and C. Julien, "Context-aware privacy-preserving access control for mobile computing," *Pervasive and Mobile Computing*, vol. 87, p. 101725, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574119222001389

[7] G. A. Safdar and A. Mansour, "Security and trust issues in BYOD networks," *IT Professional*, vol. 25, no. 4, pp. 45–51, 2023.

[8] R. Khellaf, S. Boudouda, and S. Hacini, "MAM security enhancement: Proposed control mechanism." in *Proceedings of The 11th Seminary of Computer Science Research at Feminine (RIF)*, 2022, pp. 88–100.

[9] S. Liu, J. P. Petitti, Y. Lei, Y. Liu, and C. A. Shue, "By your command: Extracting the user actions that create network flows in Android," in *International Conference on Network of the Future (NoF)*. IEEE, 2023, pp. 118–122.

[10] M. Giomi, F. Boenisch, C. Wehmeyer, and B. Tasn'adi, "A unified framework for quantifying privacy risk in synthetic data," *Proceedings on Privacy Enhancing Technologies*, vol. 2023, pp. 312–328, 2023.

[11] R. Palanisamy, A. A. Norman, and M. L. Mat Kiah, "BYOD policy compliance: Risks and strategies in organizations," *Journal of Computer Information Systems*, vol. 62, no. 1, pp. 61–72, 2022.

[12] M. Kan, "Android malware that can infiltrate corporate networks is spreading," 2016. [Online]. Available: https://www.computerworld.com/article/3126390/android-malware-that-can-infiltrate-corporate-networks-is-spreading.html

[13] D. Palmer, "Over 400 instances of dresscode malware found on Google Play Store, say researchers," https://www.zdnet.com/article/over-400-instances-of-dresscode-malware-found-on-google-play-store-say-researchers/, Oct. 2016.

[14] S. Burnett, L. Chen, D. A. Creager, M. Efimov, I. Grigorik, B. Jones, H. V. Madhyastha, P. Papageorge, B. Rogan, C. Stahl *et al.*, "Network error logging: Client-side measurement of end-to-end web service reliability," in *USENIX Symposium on Networked Systems Design and Implementation*, 2020, pp. 985–998.

[15] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "I know what your packet did last hop: Using packet histories to troubleshoot networks," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, 2014, pp. 71–85.

[16] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Profiledroid: Multi-layer profiling of android applications," in *International Conference on Mobile Computing and Networking*, 2012.

[17] A. Le, J. Varmarken, S. Langhoff, A. Shuba, M. Gjoka, and A. Markopoulou, "Antmonitor: A system for monitoring from mobile devices," in *Proceedings of ACM SIGCOMM Workshop on Crowdsourcing and Crowdsharing of Big (Internet) Data*, 2015, pp. 15–20.

[18] T. Sipola, A. Juvonen, and J. Lehtonen, "Anomaly detection from network logs using diffusion maps," in *Engineering Applications of Neural Networks*. Springer, 2011, pp. 172–181.

[19] A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "Accessminer: using system-centric models for malware protection," in *Proceedings of ACM Conference on Computer and Communications Security*, 2010, pp. 399–412.

[20] W. Yang, X. Xiao, B. Andow, S. Li, T. Xie, and W. Enck, "Appcontext: Differentiating malicious and benign mobile app behaviors using context," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 303–313.

[21] Microsoft Intune, "Microsoft intune securely manages identities, manages apps, and manages devices," 2021. [Online]. Available: https://learn.microsoft.com/en-us/mem/intune/fundamentals/what-is-intune

[22] IBM, "Mobile device management (MDM) solutions," 2024. [Online]. Available: https://www.ibm.com/products/maas360/mobile-device-management

[23] H. Xu, T. Dinev, H. J. Smith, and P. Hart, "Examining the formation of individual's privacy concerns: Toward an integrative view," 2008.

[24] K. El Emam and F. K. Dankar, "Protecting privacy using k-anonymity," *Journal of the American Medical Informatics Association*, vol. 15, no. 5, pp. 627–637, 2008.

[25] P. Saa, O. Moscoso-Zea, and S. Lujan-Mora, "Bring your own device (BYOD): Students perception — privacy issues: A new trend in education?" in *2017 16th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2017, pp. 1–5.

[26] E. Birrell, J. Rodolitz, A. Ding, J. Lee, E. McReynolds, J. Hutson, and A. Lerner, "Sok: Technical implementation and human impact of internet privacy regulations," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024, pp. 673–696.

[27] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 (General Data Protection Regulation)," *Official Journal of the European Union*, vol. L119, pp. 1–88, May 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[28] T. Werthmann, R. Hund, L. Davi, A.-R. Sadeghi, and T. Holz, "Psios: bring your own privacy & security to IOS devices," in *Proceedings of ACM SIGSAC Symposium on Information, Computer and Communications Security*, 2013, pp. 13–24.

[29] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 3–es, 2007.

[30] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *International Conference on Data Engineering*. IEEE, 2007, pp. 106–115.

[31] C. Dwork, "Differential privacy," in *International colloquium on automata, languages, and programming*. Springer, 2006, pp. 1–12.

[32] N. Ashena, O. Inel, B. L. Persaud, and A. Bernstein, "Casual users and rational choices within differential privacy," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024, pp. 932–950.

[33] N. Küchler, E. Opel, H. Lycklama, A. Viand, and A. Hithnawi, "Cohere: Managing differential privacy in large scale systems," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024, pp. 991–1008.

[34] Open DP, "Use Data. Preserve Privacy. A differential privacy toolkit for analytics and machine learning," 2020. [Online]. Available: https://smartnoise.org/

[35] S. Abd Razak, N. H. M. Nazari, and A. Al-Dhaqm, "Data anonymization using pseudonym system to preserve data privacy," *IEEE Access*, vol. 8, pp. 43 256–43 264, 2020.

[36] Z. Liu, L. Zhang, W. Ni, and I. B. Collings, "Uncoordinated pseudonym changes for privacy preserving in distributed networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1465–1477, 2019.

[37] H. H. Hlaing and H. Asaeda, "Privoff: Secure and privacy-preserving data management for distributed off-chain networks," in *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2023, pp. 326–333.

[38] S. Barth, D. Ionita, M. D. De Jong, P. H. Hartel, and M. Junger, "Privacy rating: A user-centered approach for visualizing data handling practices of online services," *IEEE Transactions on Professional Communication*, vol. 64, no. 4, pp. 354–373, 2021.

[39] Y. Chen, C. G. Cassandras, and K. Xu, "Optimal sequencing and motion control in a roundabout with safety guarantees," in *IEEE Conference on Decision and Control (CDC)*, 2024, pp. 5699–5704.

[40] S. Xia, Z. Zhu, C. Zhu, J. Zhao, K. Chard, A. J. Elmore, I. Foster, M. Franklin, S. Krishnan, and R. C. Fernandez, "Data station: delegated, trustworthy, and auditable computation to enable data-sharing consortia with a data escrow," *Proc. VLDB Endow.*, vol. 15, no. 11, p. 3172–3185, Jul. 2022. [Online]. Available: https://doi.org/10.14778/3551793.3551861

[41] Google, "Google's differential privacy libraries," https://github.com/google/differential-privacy, 2025.

[42] J. Jordon, J. Yoon, and M. Van Der Schaar, "Pate-gan: Generating synthetic data with differential privacy guarantees," in *International Conference on Learning Representations*, 2018.

[43] Socket.IO, "Socket.IO: Bidirectional and low-latency communication for every platform," https://socket.io/, 2025.

[44] SQLite Consortium , "What is SQLite?" https://sqlite.org/, 2025.
[45] tpm2-software community, "This site contains the code for the TPM (Trusted Platform Module) 2.0 tools based on tpm2-tss," https://tpm2-tools.readthedocs.io/en/latest/, 2025.
[46] I. I. A. Halim, A. G. Buja, M. S. S. Idris, and N. J. Mahat, "Implementation of BYOD security policy in malaysia institutions of higher learning (MIHL): an overview," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 33, pp. 1–14, 2023.
[47] M. Nasr, S. Songi, A. Thakurta, N. Papernot, and N. Carlin, "Adversary instantiation: Lower bounds for differentially private machine learning," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 866–882.
[48] Article 29 Data Protection Working Party, "Opinion 05/2014 on anonymisation techniques," 2014. [Online]. Available: https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf
[49] J. C. Gower, "A general coefficient of similarity and some of its properties," *Biometrics*, pp. 857–871, 1971.
[50] ICS Lab, "Capture files from 4SICS geek lounge," https://www.netresec.com/?page=PCAP4SICS, 2015.
[51] Digital Corpora, "2009 M57-patents scenario," 2025. [Online]. Available: https://digitalcorpora.org/corpora/scenarios/m57-patents-scenario/
[52] Appium Developers, "Introduction to Appium," https://appium.io/docs/en/latest/, 2025.
[53] Cloudflare Radar, "Domain rankings worldwide," 2025. [Online]. Available: https://radar.cloudflare.com/domains
[54] M. L. Fang, D. S. Dhami, and K. Kersting, "DP-CTGAN: differentially private medical data generation using CTGANs." Berlin, Heidelberg: Springer-Verlag, 2022, p. 178–188. [Online]. Available: https://doi.org/10.1007/978-3-031-09342-5_17
[55] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1895–1912. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/jayaraman
[56] H. Yan, X. Li, H. Li, J. Li, W. Sun, and F. Li, "Monitoring-based differential privacy mechanism against query flooding-based model extraction attack," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2680–2694, 2022.
[57] Android Stuido, "Measure app performance with Android Profiler," Oct. 2020. [Online]. Available: https://developer.android.com/studio/profile/android-profiler
[58] M. Trylesinski, "uvicorn: An ASGI web server, for python," https://uvicorn.dev/, 2026.
[59] R. M. Young and P. Barnard, "The use of scenarios in human-computer interaction research: turbocharging the tortoise of cumulative science," in *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*, ser. CHI '87. New York, NY, USA: Association for Computing Machinery, 1986, p. 291–296. [Online]. Available: https://doi.org/10.1145/29933.275645
[60] J. Lazar, J. H. Feng, and H. Hochheiser, *Research methods in human-computer interaction*. Morgan Kaufmann, 2017.