УДК 658.012.45
JEL: G21, L26, L86
DOI: 10.18524/2413-9998.2025.3(61).345834

O. O. Shkeda
PhD (Economics), Senior Lecturer
E-mail: alexshkeda@icloud.com
ORCID: 0000-0003-3161-5983

A. O. Khomenko
Student
E-mail: krossovki110@gmail.com
ORCID: 0009-0008-0644-4717

State University of Intelligent Technologies and Telecommunications
1 Kuznechna Str., Odesa, 65023, Ukraine

SDK ARCHITECTURE MANAGEMENT FOR MOBILE APPLICATION DEVELOPMENT IN THE CONTEXT OF MARKETING DATA PRIVACY

Mobile applications increasingly rely on embedded analytics software development kits (SDKs) to monitor usage, optimize user experience and support data-driven business decisions. At the same time, empirical studies show that many third-party SDKs exfiltrate privacy-sensitive data, misrepresent their practices in privacy policies and expose applications to supply-chain security risks. These findings create a tension between the need for fine-grained behavioral data and strict requirements of regulations such as the GDPR and CCPA. The paper proposes a privacy-aware reference architecture for a mobile analytics SDK intended to be integrated into native applications as a first-party or carefully governed third-party component. The architecture decomposes the SDK into five layers: Core Module, Data Collection & Processing, Storage & Queuing, Networking & Communication, and Integration. For each layer responsibilities, data flows, and embedded privacy controls, including consent-driven initialization, configurable event schemas, local pre-processing, encrypted transport, throttling, and auditable public APIs were specified. The research design combines analysis of recent empirical studies on SDK privacy risks and platform policies, conceptual modeling, and inspection of configuration options in contemporary analytics SDKs. The proposed model is then discussed against existing evidence about default SDK behavior, fingerprinting practices, and developer constraints. The contribution of the paper is twofold: it systematizes scattered knowledge about how mobile analytics SDKs operate, and it offers a practical blueprint that helps developers align client-side analytics with modern privacy and security expectations while preserving the analytical value of event data.

Keywords: digital marketing, app-marketing, development management, product management, data protection and privacy, marketing analytics, SDK.

Introduction. Mobile apps have become one of the primary channels through which organizations interact with customers. To understand how users navigate interfaces, where they abandon funnels and which features drive value, developers embed analytics frameworks provided as mobile software development kits (SDK). Recent work shows that such analytics frameworks are now a standard part of the Android and iOS ecosystems and are integrated into a large share of popular applications [1]. At the same time, industry reports emphasise that mobile application analytics is a key enabler of product management, UX-optimisation and marketing decision-making [2].

However, these benefits are accompanied by significant privacy and security concerns. Large-scale static and dynamic analyses of Android SDK reveal extensive exfiltration of privacy-sensitive data, lack of transparency in privacy policies and long-lasting non-compliance with regulatory expectations [3]. Default settings in advertising and analytics SDKs favour extensive tracking, leaving developers with limited effective control over what data is collected in their apps [4, 5].

From the end-user perspective, mobile apps frequently request unnecessary permissions, and privacy notices are often ignored or misunderstood, especially in sensitive domains such as mobile health [6]. Platform vendors respond by tightening policies and issuing detailed guidelines for both SDK providers and app developers on how to handle user data, consent and policy disclosures [7].

Against this background, there is a practical need for architectural guidance on how a mobile analytics SDK should be structured so that it remains useful for behaviour analysis while supporting privacy-by-design principles. This paper addresses that need by proposing and discussing a layered, privacy-aware architecture of a mobile analytics SDK intended for integration into native mobile applications.

Literature Review. Research on SDK in mobile ecosystems has intensified in recent years. Fedynyshyn and Partyka analyse over six thousand Android application packages and identify the most widely used analytics frameworks, highlighting typical vulnerabilities such as insecure transmission, outdated dependencies and insufficient access control [1]. Their work underlines that analytics SDKs have become critical but fragile infrastructure in the mobile software supply chain.

Meng et al. conduct a targeted analysis of 158 popular Android SDKs and demonstrate that many of them engage in privacy-sensitive data collection

and sharing practices that are inconsistent with their declared privacy policies [3]. They show that more than one-third of examined SDKs over-collect data and that false or incomplete disclosures are commonplace.

Koch et al. focus on «default» mobile SDK usage and show that simply embedding contemporary analytics and advertising SDKs, without enabling any advanced features, already results in extensive user tracking and data transfers [4]. Developers have only limited options to restrict this behaviour, and these options are often poorly documented.

Rodriguez et al. study Facebook Android SDKs and find that most developers keep privacy-related configuration parameters at defaults and rarely use the available mechanisms to improve privacy, even when such options exist [5].

Specter et al. reveal large-scale device fingerprinting practices implemented via SDKs and emphasise that users often remain unaware of such tracking because it takes place inside third-party code bundled with legitimate apps [8].

Industry and platform actors also contribute to this discussion. Google's official guidance on SDK best practices stresses that app developers remain responsible for all user-data handling performed by integrated SDKs and recommends explicit consent management, minimisation of collected data and constant monitoring of SDK behaviour [7]. Practitioner-oriented literature on mobile app analytics highlights the importance of tracking plans, carefully selected key performance indicators and privacy-friendly analytics solutions hosted in appropriate jurisdictions [2].

At the same time, economic research suggests that third-party SDKs, particularly "tool-type" SDKs such as analytics and payments, can positively affect app performance and daily active users when used appropriately [9]. This confirms that analytics SDKs are not easily replaceable: they provide real business value but need to be governed carefully.

The reviewed works paint a consistent picture: analytics SDKs are indispensable for modern mobile development, yet their current implementations frequently conflict with privacy regulations and best practices. Most studies focus either on measurement of privacy violations or on governance and policy aspects. Less attention has been given to reference architectures that translate privacy requirements into concrete design decisions inside the SDK itself. This gap motivates the present paper.

Aim and scope. The purpose of this research is to develop and justify a

privacy-aware reference architecture for a mobile analytics SDK that enables fine-grained event tracking while supporting regulatory compliance and privacy-by-design principles.

To achieve this purpose, the following research tasks were defined:

- Analyse recent empirical and conceptual studies on mobile analytics frameworks, third-party SDK behaviour and platform policies in order to derive architectural requirements for a privacy-aware SDK.
- Conceptually decompose a mobile analytics SDK into functional layers and modules, explicitly identifying data flows and decision points where privacy controls can be embedded.
- Propose a model of the SDK architecture that integrates technical mechanisms such as configurable event schemas, local pre-processing, encrypted communication and consent-driven initialisation.
- Compare the proposed architecture with documented problems of existing SDKs and discuss how it can mitigate privacy and security risks highlighted in the literature.

Methods. A targeted literature review was performed focusing on works published between 2020 and 2025 that examine third-party SDKs, analytics frameworks and mobile privacy. Peer-reviewed articles and conference papers were prioritised, complemented by platform documentation and practitioner reports where necessary [1–9]. Architectural modelling techniques were applied. Based on typical responsibilities of mobile analytics SDKs described in technical documentation [10] and industry best practices [2, 7], the SDK was decomposed into layers. For each layer, responsibilities, data types and interfaces were specified. Special attention was paid to where personal data is accessed, transformed, stored and transmitted.

The resulting model was iteratively refined by mapping it against known classes of privacy and security issues identified in empirical studies, such as uncontrolled data exfiltration, opaque default settings, fingerprinting behaviour and lack of consent enforcement [3–5]. The focus was not on implementing or benchmarking a concrete SDK but on synthesising an architecture that could serve as a blueprint for privacy-conscious implementations.

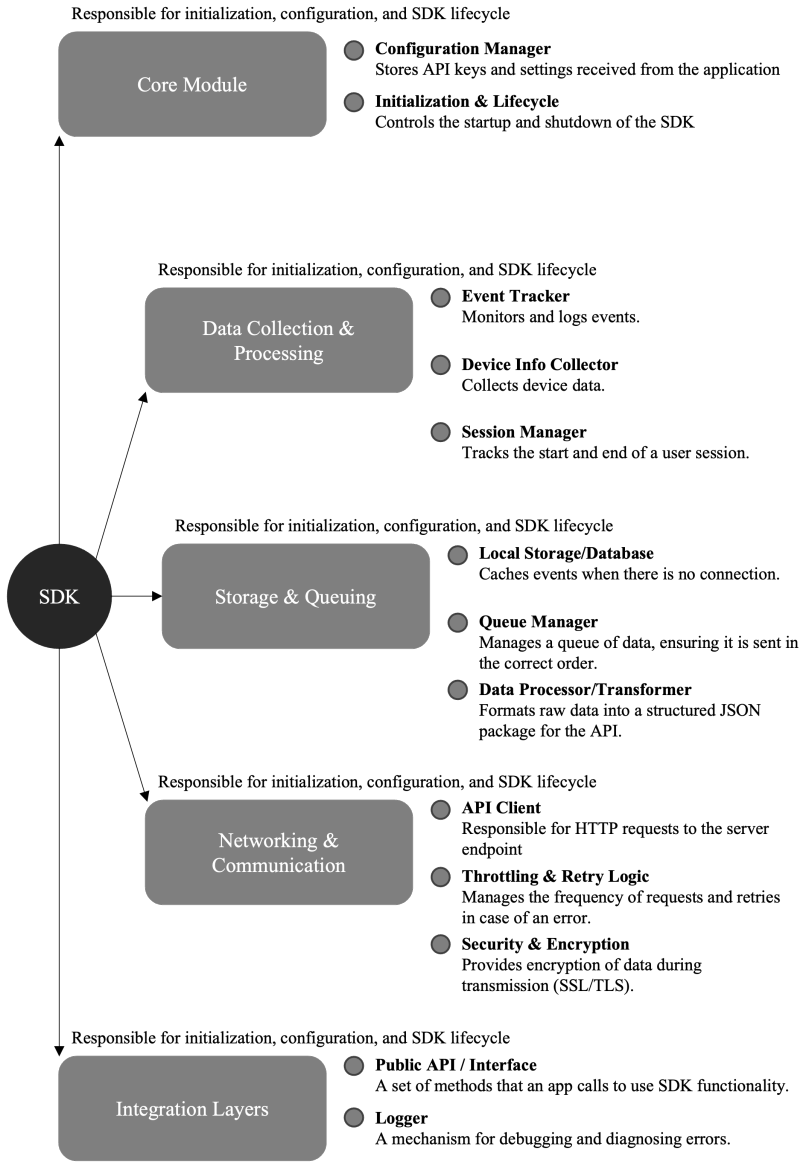
Results and discussion. The proposed mobile analytics SDK is organised into five interconnected layers: Core Module, Data Collection & Processing, Storage & Queuing, Networking & Communication, and Integration Layers. Each layer encapsulates specific responsibilities and exposes well-defined interfaces to the surrounding application and backend

services. The structure reflects typical designs of contemporary analytics SDKs while making privacy and security controls explicit [2, 7, 10]. The overall structure of the proposed mobile analytics SDK is summarised in Figure 1. The architecture decomposes the SDK into five layers: Core Module, Data Collection & Processing, Storage & Queuing, Networking & Communication, and Integration.

The Core Module coordinates the internal life cycle of the SDK and acts as the main control point through which the host application can influence its behaviour. It is responsible for reading configuration parameters, initialising internal services and exposing a small, clearly documented public API. From a privacy-by-design perspective, this module must be explicitly coupled with the app's consent management logic. Platform guidelines emphasise that app publishers remain accountable for all user data processed by integrated SDKs and that consent must be enforced consistently across components [7].

In practice, this means that the SDK should never start collecting or transmitting data automatically upon app launch. Instead, the Core Module exposes an initialisation function that accepts the current consent state and configuration profile as arguments. If the user has not yet provided consent, the module initialises only minimal, non-tracking functionality such as offline configuration loading or logging of purely technical errors. Once the user grants consent, the application can re-initialise the SDK with an extended configuration that enables event tracking, device identifiers or additional modules. This design ensures that changes in consent state are reflected in SDK behaviour without requiring a full app restart.

The configuration subsystem within the Core Module defines which data categories and destinations are active. For example, one configuration profile may allow only anonymous usage statistics directed to an EU-based endpoint, while another profile may additionally permit user identifiers or custom attributes for internal analytics. Contemporary analytics SDKs already use configuration flags to toggle automatic events and server regions [10]; the proposed architecture extends this approach by treating configuration as a formal contract that can be inspected, versioned and audited. Such explicitness reduces the risk that developers unknowingly rely on problematic defaults, a pattern repeatedly observed in empirical studies of analytics SDKs [4, 5].



Fugure 1. Mobile SDK Architecture
Source: created by the authors

The Data Collection & Processing layer is responsible for turning raw interactions into normalised analytics events that can be interpreted consistently on the backend. It aggregates several components: an event tracker, a device and context information collector, and a session manager.

The event tracker offers public methods such as trackEvent, trackScreen or trackPurchase that are called by the host application at semantically meaningful points. To avoid ad hoc, inconsistent instrumentation, events should follow a predefined tracking plan that specifies names, required properties and allowed value ranges [2]. The SDK can embed basic validation of this plan, rejecting malformed events locally instead of silently sending corrupted data to the server.

The device and context collector gathers information about the execution environment, such as operating system version, app version, locale and network type. Prior analyses show that many third-party SDKs go significantly further and collect stable device identifiers that can be used for cross-app tracking and fingerprinting [3,8]. In the proposed architecture, this collector is strictly governed by configuration: higher-risk identifiers (for example, advertising IDs) are disabled by default and can only be enabled through explicit opt-in in the configuration profile that is consistent with the legal basis chosen by the controller. This mirrors the "data minimisation" principle and helps developers align SDK usage with their declared privacy policies.

The session manager groups events into time-bounded interaction windows. It defines when a new session starts, how long it remains active and under which conditions it is reset (for example, after a period of inactivity or logout). Instead of relying on long-lived opaque identifiers, the architecture supports short random session IDs that are rotated regularly. This design still allows analysts to compute session-based metrics such as engagement or retention, while reducing the likelihood that the same identifiers are reused across different contexts. Furthermore, on-device pre-processing can perform lightweight aggregation (e.g. collapsing repeated low-value events) and redaction (e.g. truncating or hashing free-text fields) before events enter the queue. Empirical work shows that over-collection often arises unintentionally from default SDK behaviour [3–5]; therefore, implementing cautious pre-processing at this layer directly mitigates a major class of observed problems.

The Storage & Queuing layer ensures that events are preserved temporarily when the device is offline or connectivity is poor, and that they

are transmitted in a controlled, resource-efficient manner. It consists of a local storage engine, a queue manager and, optionally, a batch processor. The storage engine uses platform-provided secure storage primitives or encrypted files instead of plain-text logs. Security assessments of mobile apps and SDKs frequently report sensitive data left unencrypted on the device [1], which can be accessed by other apps on rooted devices or during forensic analysis. By design, the proposed architecture encrypts every persisted batch of events with a key bound to the application context, limiting the damage even if local storage is accessed.

The queue manager orchestrates how events move from local storage to the networking layer. It controls batch sizes, flush intervals and triggers (for example, app moving to background, device becoming idle, or network type changing from cellular to Wi-Fi). Reasonable defaults favour smaller, less frequent batches to reduce network overhead and exposure, while configuration parameters allow performance tuning for high-volume scenarios. The manager also enforces retention limits: events older than a configured threshold are automatically dropped, preventing silent accumulation of stale data on the device.

A batch processor can apply final transformations before transmission. For instance, it can add metadata such as the SDK version, current configuration profile identifier or aggregated counters. These metadata fields are valuable for backend monitoring and for auditing whether events observed on the server side are consistent with the configuration that should have been active on the client. By isolating storage and queuing concerns in this layer, the architecture makes it easier to reason about where data resides at each moment and to demonstrate compliance with retention and security requirements during assessments or audits.

The Networking & Communication layer is responsible for reliably delivering event batches to backend endpoints while enforcing transport-level security and rate controls. It encapsulates an HTTP client, retry and back-off policies, endpoint selection and error handling. All outbound connections are initiated only after the Core Module confirms that the current consent state permits analytics transmission. Furthermore, the network client is configured to use modern TLS versions with strict certificate validation and to reject insecure protocols. Numerous empirical studies have shown that misconfigured TLS or custom networking code are common sources of vulnerabilities in mobile apps [1]; by centralising

networking logic inside the SDK, providers can enforce secure defaults across all integrating applications.

Retry and back-off strategies are particularly important on mobile devices with intermittent connectivity. Instead of naïvely retrying failed requests in tight loops, the SDK should implement exponential back-off with random jitter and global rate limits. This prevents unnecessary battery drain and avoids generating excessive traffic that could look suspicious from the server's perspective. Configuration parameters allow the app developer to set upper bounds on daily event volume or bandwidth usage, which is especially relevant in regions where mobile data is expensive.

Endpoint selection in this architecture is also privacy-aware. Based on configuration and possibly the user's region, the SDK chooses between different data centres or logical endpoints to respect data residency requirements [2]. For example, events from users in the European Economic Area may always be routed to EU-based servers. The networking layer attaches only the minimal set of headers and identifiers required for authentication and routing; all superfluous identifiers that could facilitate cross-service correlation are removed. Combined with the data minimisation mechanisms in the Data Collection & Processing layer, this design reduces the attack surface and the potential for misuse of analytics traffic, while still supporting robust and timely ingestion of behaviour data on the backend.

The Integration layer connects the SDK to the host application through a public API and optional diagnostic tools. The public API should remain intentionally small and stable over time, ideally consisting of a handful of well-named methods for initialisation, event tracking and configuration updates. Prior research indicates that developers often misunderstand or overlook privacy-relevant configuration options in third-party libraries [5]. A concise, well-documented interface with explicit arguments for consent state, data categories and endpoints reduces the likelihood of such misunderstandings. Clear parameter names and type-safe interfaces help signal which calls may initiate data collection or transmission.

Diagnostic tooling is essential for successful integration, but it can itself become a source of leakage if implemented carelessly. The architecture therefore distinguishes between development-time debugging and production-time operational monitoring. In debug builds, the SDK may expose verbose logs, on-screen overlays or inspection consoles that show which events are being generated and queued. In production, however, these

tools should be disabled or heavily restricted, and any remaining logs must exclude personal data. Instead, the SDK can emit anonymised counters (for example, number of events dropped due to missing consent, or number of failed network attempts) that help teams monitor health without exposing individual user traces.

Finally, the Integration Layers provide hooks that allow the host application to react to privacy-relevant events. For example, when a user withdraws consent in the app's settings screen, the app can call a dedicated method such as updateConsent(false), after which the SDK flushes pending events according to policy, stops collectors and prevents future transmissions. Such explicit hooks create a direct mapping between user-facing controls and SDK behaviour, making it easier to demonstrate compliance to regulators or auditors. By treating integration points as part of the overall privacy architecture rather than a mere technical detail, the proposed design encourages closer collaboration between developers, legal teams and product managers when defining how analytics is used in the application.

The proposed architecture addresses several problems identified in recent empirical research. First, by making consent a mandatory parameter of SDK initialisation and tying configuration to explicit data categories, it counteracts default data collection that developers may not fully understand [4, 5]. Second, by emphasising configurable device identifiers and local pre-processing, it reduces the risk of fingerprinting practices uncovered in large-scale analyses [3, 8]. Third, encrypted storage, strict TLS usage and bounded retries respond to security weaknesses and leakage channels observed in existing analytics frameworks [1].

At the same time, the architecture preserves the analytical value of collected data. Event tracking, session management and queueing are standard features of successful analytics SDKs and are known to correlate with improved app performance when used responsibly [2, 9, 10]. The proposed model therefore aims not to eliminate analytics, but to embed it into an explicit governance structure that clarifies responsibilities between SDK provider, app developer and backend operator.

The main limitation of the present work is that the architecture has been evaluated only conceptually. Future research should implement at least a prototype SDK following this design and compare its behaviour empirically with widely used frameworks, both in terms of privacy guarantees and the quality of analytical insights.

Conclusions. This paper has proposed a privacy-aware reference architecture for a mobile analytics SDK intended for integration into native mobile applications. Based on recent empirical evidence about privacy and security risks associated with third-party SDKs and platform guidance on responsible data handling, the SDK was decomposed into five layers: Core Module, Data Collection & Processing, Storage & Queuing, Networking & Communication, and Integration Layers.

For each layer it was identified responsibilities, data flows and embedded control points where privacy-by-design mechanisms could be implemented. Key features of the architecture include consent-driven initialisation, configurable event schemas and device identifiers, encrypted local storage, secure and region-aware network communication, and transparent public APIs with privacy-aligned hooks.

Compared with typical "black box" analytics SDKs, the proposed model makes data handling steps explicit and provides developers with practical levers to align analytics with regulatory and organisational requirements. The architecture thus contributes to bridging the gap between high-level privacy principles and concrete implementation choices in mobile analytics. Future work should validate the model through prototype implementations, performance measurements and user studies on developers' understanding of SDK behaviour.

References

1. Fedynyshyn, T., & Partyka, O. (2025). Data privacy and security challenges in mobile application analytics frameworks. *Cybersecurity Providing in Information and Telecommunication Systems (CPITS 2025): Proceedings of the workshop* (pp. 233–240). CEUR-WS. https://ceur-ws.org/Vol-3991/paper17.pdf
2. Chronowska, N., & Lubowicka, K. (2022, May 31). *Mobile app analytics: Best practices that help you do it right.* Piwik PRO Blog. https://piwik.pro/blog/mobile-app-analytics-best-practices/
3. Meng, M. H., Yan, C., Hao, Y., Zhang, Q., Wang, Z., Wang, K., & Dong, J. S. (2024). A large-scale privacy assessment of Android third-party SDKs (arXiv:2409.10411). *arXiv*. https://arxiv.org/abs/2409.10411
4. Koch, S., Karl, M., Kirchner, R., Wessels, M., Paschke, A., & Johns, M. (2025). The impact of default mobile SDK usage on privacy and data protection. *Proceedings on Privacy Enhancing Technologies, 2025*(1), 808–810. https://petsymposium.org/popets/2025/popets-2025-0042.pdf
5. Rodriguez, D., Calandrino, J. A., del Álamo, J. M., & Sadeh, N. (2025). Privacy settings of third-party libraries in Android apps: A study of Facebook SDKs. *Proceedings on Privacy Enhancing Technologies*, *2025*(2), 175–185.

https://petsymposium.org/popets/2025/popets-2025-0056.pdf
6. Aljedaani, B., Ahmad, A., Zahedi, M., & Babar, M. A. (2023). An empirical study on secure usage of mobile health apps: The attack simulation approach. *Information and Software Technology*, *163*, Article 107285. https://www.sciencedirect.com/science/article/pii/S0950584923001398
7. Google. (2025, June 17). *About user safety and SDKs*. Android Developers. https://developer.android.com/guide/practices/sdk-best-practices
8. Specter, M. A., Christodorescu, M., Farr, A., Ma, B., Lassonde, R., Xu, X., & Kleidermacher, D. (2025). Fingerprinting SDKs for mobile apps and where to find them: Understanding the market for device fingerprinting (arXiv:2506.22639). *arXiv*. https://arxiv.org/pdf/2506.22639
9. Xia, Y., Chen, H., & Fang, Y. (2025). *Third-party SDK utilization and mobile app market performance: An empirical study from the boundary-spanning perspective*. Information Systems Research (forthcoming). SSRN. https://ssrn.com/abstract=5335497
10. Amplitude. (n.d.). *Android-Kotlin SDK documentation*. https://amplitude.com/docs/sdks/analytics/android/android-kotlin-sdk

Received 07.11.2025

О. О. Шкеда
доктор філософії, старший викладач
E-mail: alexshkeda@icloud.com
ORCID: 0000-0003-3161-5983

А. О. Хоменко
студентка
E-mail: krossovki110@gmail.com
ORCID: 0009-0008-0644-4717

Державний університет інтелектуальних технологій і зв'язку
вул. Кузнечна, 1, м. Одеса, 65023, Україна

УПРАВЛІННЯ АРХІТЕКТУРОЮ SDK ДЛЯ РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ У КОНТЕКСТІ КОНФІДЕНЦІЙНОСТІ МАРКЕТИНГОВИХ ДАНИХ

У статті запропоновано еталонну архітектуру мобільного аналітичного SDK, орієнтовану на принципи приватності даних. На основі аналізу сучасних досліджень щодо ризиків рекомендацій платформ SDK декомпоновано на п'ять шарів: ядро, модуль збору та обробки даних, шар тимчасового зберігання і черг, мережеві комунікації та інтеграційні шари. Для кожного шару визначено відповідальність, потоки даних та точки вбудовування контролів конфіденційності: ініціалізація, що залежить від згоди користувача, конфігуровані схеми подій та ідентифікаторів пристрою, локальне попереднє опрацювання, шифроване зберігання й передавання, механізми обмеження частоти запитів та прозорий публічний API. Показано, як запропонована модель пом'якшує проблеми, виявлені в емпіричних роботах щодо

аналітичних та рекламних SDK, і водночас зберігає аналітичну цінність даних для продуктового та маркетингового аналізу.

Ключові слова: диджитал-маркетинг, маркетинг мобільних застосунків, управління розробкою, продакт менеджмент, захист даних та конфіденційність, маркетингова аналітика, SDK.

Список використаної літератури

1. Fedynyshyn T., Partyka O. Data privacy and security challenges in mobile application analytics frameworks. *Cybersecurity Providing in Information and Telecommunication Systems (CPITS 2025): Proceedings of the Workshop. CEUR Workshop Proceedings*. 2025. Vol. 3991. P. 233–240. URL: https://ceur-ws.org/Vol-3991/paper17.pdf (дата звернення: 05.11.2025).
2. Chronowska N., Lubowicka K. *Mobile app analytics: Best practices that help you do it right*. Piwik PRO Blog. 31.05.2022. URL: https://piwik.pro/blog/mobile-app-analytics-best-practices/ (дата звернення: 05.11.2025).
3. Meng M. H., Yan C., Hao Y. et al. A large-scale privacy assessment of Android third-party SDKs. *arXiv preprint*. 2024. URL: https://arxiv.org/abs/2409.10411 (дата звернення: 05.11.2025).
4. Koch S., Karl M., Kirchner R., Wessels M., Paschke A., Johns M. The impact of default mobile SDK usage on privacy and data protection. *Proceedings on Privacy Enhancing Technologies*. 2025. Vol. 2025, No. 1. P. 808–810. URL: https://petsymposium.org/popets/2025/popets-2025-0042.pdf (дата звернення: 05.11.2025).
5. Rodriguez D., Calandrino J. A., del Álamo J. M., Sadeh N. Privacy settings of third-party libraries in Android apps: A study of Facebook SDKs. *Proceedings on Privacy Enhancing Technologies*. 2025. Vol. 2025, No. 2. P. 175–185. URL: https://petsymposium.org/popets/2025/popets-2025-0056.pdf (дата звернення: 05.11.2025).
6. Aljedaani B., Ahmad A., Zahedi M., Babar M. A. An empirical study on secure usage of mobile health apps: The attack simulation approach. *Information and Software Technology*. 2023. Vol. 163. Article 107285. URL: https://www.sciencedirect.com/science/article/pii/S0950584923001398 (дата звернення: 05.11.2025).
7. Google. About user safety and SDKs. Android Developers. 17.06.2025. URL: https://developer.android.com/guide/practices/sdk-best-practices (дата звернення: 05.11.2025).
8. Fingerprinting SDKs for mobile apps and where to find them: Understanding the market for device fingerprinting / M. A. Specter, M. Christodorescu, A. Farr et al. *arXiv preprint*. 2025. URL: https://arxiv.org/pdf/2506.22639 (дата звернення: 05.11.2025).
9. Xia Y., Chen H., Fang Y. Third-party SDK utilization and mobile app market performance: An empirical study from the boundary-spanning perspective. *Information Systems Research* (forthcoming). 2025. URL: https://ssrn.com/abstract=5335497 (дата звернення: 05.11.2025).
10. Amplitude. Android-Kotlin SDK documentation. URL: https://amplitude.com/docs/sdks/analytics/android/android-kotlin-sdk (дата звернення: 05.11.2025).