International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

Implementation of Zero-Trust Architecture in Mobile Applications of the Financial Sector

Pankiv Oleg*

Senior IOS Developer, Myseum.Inc, Bayonne, NJ, USA
Email: oleg.pankiv@gmail.com

Abstract

This article examines the rethinking of zero-trust architecture for mobile applications in the financial sector as a crystallization of the contemporary model of cyber-resilience amid explosive growth in mobile banking and the rise of specialized smartphone attacks. It is demonstrated that traditional perimeter-based security models do not provide the required level of protection in the context of a high density of vulnerabilities in client applications and the exponential increase in malicious activity, which predetermines the relevance of a transition to an approach in which every request and every action is treated as potentially untrusted. The purpose of the study is to conceptualize and specify the principles of zero-trust architecture at the mobile client level. The scientific novelty lies in shifting the center of gravity from an abstract trusted perimeter to a set of explicitly defined decision points across the interface, domain logic, and infrastructural library layers, as well as in introducing an application trust map, standardized interface components as carriers of security policy, and a unified software kit for authentication, session management, and authorization checks. The proposed roadmap, from constructing a trust map to the phased implementation of multi-step, risk-oriented authentication, enables transforming the zero-trust architecture from a one-time initiative into a continuous process embedded in the life cycle of mobile financial product development. The article is intended for researchers, architects, and practitioners engaged in the design and evolution of mobile solutions in the banking and fintech sectors.

Keywords: zero-trust architecture; mobile banking; fintech; mobile applications; cyber-resilience.

Received: 12/10/2025
Accepted: 2/10/2026
Published: 2/21/2026

---

* *Corresponding author.*

1. Introduction

Large-scale digitalization of financial services has led to mobile banking applications becoming the primary interface for client interaction with financial institutions. According to industry research estimates, by 2025, approximately two-thirds of the global population will have access to mobile banking. In several regions, the share of users exceeds 70%, making the mobile channel dominant not only for basic operations but also for lending, investment services, and the remote servicing of complex products [1]. At the same time, systematic security reviews of banking applications show that a substantial number of vulnerabilities accompany the widespread adoption of technologies. In one recent study, an analysis of 37 mobile banking applications identified 2,157 potential vulnerabilities of varying criticality [2].

Against this backdrop, zero-trust architecture, conceptualized in the guidance of the U.S. National Institute of Standards and Technology as a model in which no subject and no request is considered trusted by default and every access to a resource requires explicit verification and minimization of granted privileges, is becoming one of the key methodological reference points for the design of secure financial systems [3].

For the financial sector, the specific characteristics of threats make the shift to a zero-trust architecture not a matter of technological fashion, but a prerequisite for basic resilience. Banks and other financial institutions store critical economic data, manage highly liquid assets, and operate under strict regulatory constraints and constant pressure from cybercrime. Contemporary research demonstrates that traditional perimeter-based protection models and classical multilayer defense schemes do not provide sufficient adaptability to sophisticated attacks and insider threats, whereas a zero-trust architecture tailored to the financial industry offers significantly greater resilience against credential compromise, malicious activity within the infrastructure, and cross-border attacks [4].

This is also reflected in implementation practices: according to a global survey of cybersecurity professionals conducted in 2023, more than 60 % of organizations already have a formulated zero-trust strategy, and in the financial services segment, the share of such companies reaches approximately 70 %, making the sector one of the leaders in terms of institutionalization of this approach [5]. Thus, for financial institutions, zero-trust architecture is ceasing to be an abstract idea and becoming a practical framework that defines requirements for the design, operation, and evolution of information systems.

The characteristics of mobile banking and fintech applications amplify the need for such a framework while simultaneously complicating its realization. Unlike traditional channels, a mobile application operates on heterogeneous user devices, often with outdated operating system versions, in potentially insecure networks, and with a wide set of sensors and permissions that create an additional attack surface. Mobile banking brings not only advantages in terms of convenience and financial inclusion, but also specific risks of unauthorized account access, leakage of confidential information, and transaction tampering, all of which require dedicated management efforts from banks. At the same time, empirical threat data show that attackers are increasingly exploiting the mobile channel: in 2024, the number of banking-trojan-class malware attacks on smartphones almost tripled (from 420,000 to 1,242,000 incidents per year), and the total number of attacks on mobile devices exceeded 33 million, a substantial share of which targeted theft of payment and account data [6].

In combination with the high density of vulnerabilities in the applications themselves, this forms a situation in which reliance solely on perimeter protection of server infrastructure becomes manifestly insufficient; architectural solutions are required that can transpose the principles of zero trust directly into the logic of the mobile client, its component structure, and the scenarios of user interaction with financial services, which constitutes the subject of further analysis in this article.

2. Materials and Methodology

The empirical and theoretical basis of the study is a purposefully assembled corpus of 10 sources covering statistics on the spread of mobile banking, vulnerabilities of mobile financial applications, the evolution of threats in the fintech sector, and the normative and architectural foundations of zero trust. The macro-level materials include industry estimates of mobile banking penetration and the transformation of client interfaces [1], as well as a global survey of zero-trust strategy maturity that enables comparison of the financial sector with other industries in terms of the degree of institutionalization of this approach [5]. The technical and applied layer is formed by a systematic review of privacy and security in mobile banking applications [2], an analysis of vulnerabilities in digital banking clients [8], and aggregated data on the dynamics of specialized mobile attacks, primarily banking-trojan-class malicious software [6], which together define a quantitatively and qualitatively delineated risk landscape. Finally, the conceptual core of the materials is formed by the normative description of zero-trust architecture in the NIST guidance [3], proposals for its reinforcement in the financial industry through blockchain integration [4], a systematic review of cyberthreats in fintech [7], as well as practice-oriented OWASP guidelines on certificate pinning and on the construction of authentication and session management in mobile applications [9, 10].Methodologically, the work is based on a combination of systematic literature review, regulatory and legal analysis, and problem-oriented architectural synthesis. At the first stage, the context was reconstructed: statistics on the spread of mobile banking [1], empirical data on application vulnerabilities [2, 8], and the dynamics of specialized attacks on mobile devices [6, 7] were juxtaposed for the purpose of identifying stable threat classes and vulnerable points specific to financial mobile clients. At the second stage, a normative and architectural analysis was conducted of the NIST guidance on zero-trust architecture [3], an industry framework for finance using blockchain [4], and OWASP's practical recommendations [9, 10]. The results were interpreted as a set of invariant requirements for authentication, session management, channel protection, and privilege minimization in the mobile context. At the third stage, comparative and content-analytical approaches were applied: the identified requirements were correlated with the previously established characteristics of mobile threats and vulnerabilities [2], [6–8], which made it possible to derive a set of architectural patterns and principles for implementing zero trust directly within the structure of the mobile application and its associated software kits, serving as the basis for subsequent conceptual design.

3. Results and Discussion

In the context of the already-mentioned challenges in the financial sector, zero-trust architecture can be viewed as a formalized set of access-control principles in which the initial assumption is that any subject, device, or network is potentially malicious. In publications by the U.S. National Institute of Standards and Technology, zero trust is described as a model in which each resource request is accompanied by explicit authentication, authorization, and

context evaluation, and decisions are made based on policies constructed around protected resources rather than the network perimeter [3]. For mobile applications of financial organizations, this implies abandoning the concept of a single sign-in to a trusted zone: a user who has passed authentication does not automatically receive unrestricted access; each sensitive step, viewing transaction history, creating a payment order, changing limits, binding a new device, must be accompanied by a reassessment of the trust level, taking into account the behavioral profile, environmental parameters, and accumulated telemetry. Research on threats in the fintech sector shows that it is precisely insufficiently stringent control at the level of specific operations, rather than the absence of basic authentication, that leads to a significant share of incidents, including fraudulent transfers and unauthorized access to confidential customer data [7].

The first practical layer of zero trust in a mobile application is the principled distrust toward the user device, which is treated as potentially compromised. In banking and payment software, this is expressed first in the need to detect modified environments: the presence of superuser rights and non-standard firmware significantly increases the likelihood of bypassing isolation mechanisms and substituting the application interface, as confirmed by empirical studies of vulnerabilities in mobile banking clients [8].

Second, it is reflected in strict requirements for local storage of key material: account credentials, cryptographic keys, and access tokens must be stored solely in specialized, secure operating system storage facilities and, where possible, bound to hardware-based trusted execution modules, thereby reducing the probability of their extraction by malicious code. A comparable degree of skepticism is extended to the network environment: the mobile client must assume that any communication channel may be subject to a man-in-the-middle attack and therefore must implement rigorous verification of endpoint authenticity. Certificate pinning of trusted server credentials in the application code and refusal to accept unexpected certificates are considered basic measures against man-in-the-middle attacks, especially in financial transaction scenarios [9].

The next fundamental aspect of zero-trust architecture in a mobile financial client is the rejection of session-wide trust and the use of access tokens with limited validity periods, with strict lifecycle management. Mobile security testing guidelines emphasize that long-lived sessions without forced renewal, predictable identifiers, and the absence of binding of the session to the device and context significantly increase the risk of account hijacking and replay of stolen tokens [10].

Under zero-trust conditions, each session is treated as a temporary and conditional state that is preserved only while control policies are satisfied: upon changes in environmental parameters, an increase in risk profile, or attempts to perform a highly critical operation, the user may be required to undergo re-authentication or confirmation through an independent channel. This approach is closely connected with the principle of least privilege, according to which the mobile application and its individual functional modules must operate only with the set of rights and data strictly necessary to implement a specific scenario, and any expansion of access must be explicit, time-bound, and subject to audit. Systematic reviews of vulnerabilities in mobile banking applications show that excessive permissions and superfluous privileges of client components remain a persistent source of risk, and that the transition to an architecture that rigorously enforces privilege minimization at the levels of interface, logic, and network interaction is considered one of the key directions in the evolution of security

practices for this class of systems [2]. The main principles of implementing zero trust in mobile finance are shown in Fig. 1.

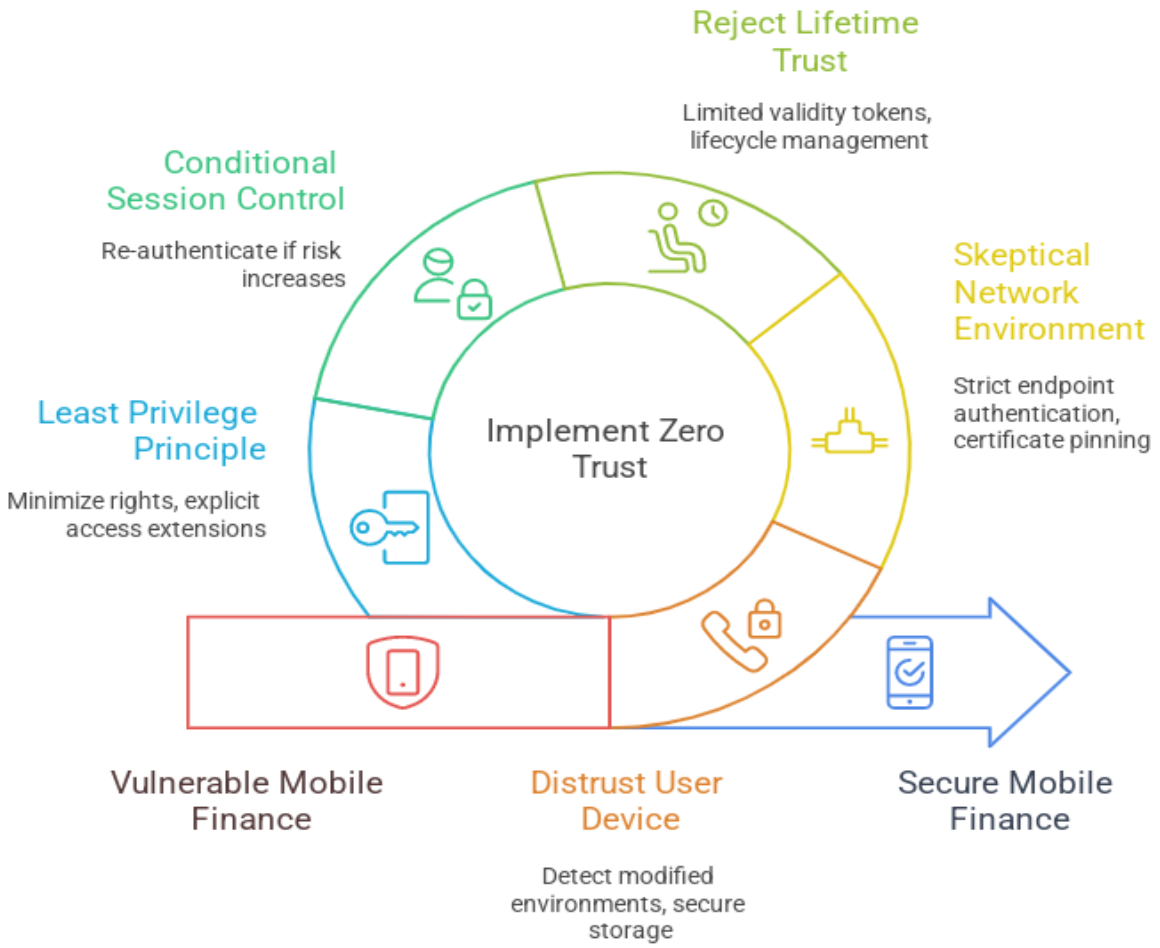

Figure 1: Implementing Zero Trust in Mobile Finance

Zero-trust architecture in a mobile financial application manifests primarily in how the system is internally decomposed into layers and how decision-making functions are distributed among them. Within this approach, the user interface layer is not reduced to screen rendering; instead, it becomes a controlled interaction space in which each visual element is associated with a specific security policy. Operation-confirming buttons and secret-data input fields share the same look and feel, as do screens displaying account balances and transaction histories; all checks for authority and context are handled consistently. An authorization-logic layer over the user-interface layer interprets user events as requests for business operations, which are authorized, verified, confirmed, or rejected. Even deeper are the network and cryptographic layers, often implemented as shared libraries, which provide encryption, key management, token management, and interaction with external services. Such separation of layers makes the trust model transparent: the interface initiates actions, the domain logic evaluates their admissibility, and the infrastructural layer ensures technical execution in accordance with cryptographic and protocol requirements.

Suppose the application is considered a system of data flows. In that case, it becomes apparent that a zero-trust

architecture requires not only protection of information storage and transmission, but also a clear definition of the points at which decisions to grant or deny access are made. Data about the user, device, environment, behavioral indicators, and preceding operations must not be diffused across interface code but instead be concentrated in dedicated decision-making components governed by policy and rules. An explicit context shift must accompany any movement of sensitive data between layers: the interface receives only the minimally necessary fragments, the domain logic operates with abstractions such as right to view or right to initiate a transfer, and the network layer does not see superfluous details of the business scenario at all, being responsible only for reliable delivery and verification of the authenticity of the remote side. Crucially, in such a mode,l trust does not arise from the mere fact of being inside the application: each new flow, each access to a critical resource is treated as an event requiring correlation with policy and the current risk level.

In interaction with the server side, the zero-trust architecture manifests itself in rejecting the legacy logic of a protected perimeter, in which a mobile client that has once passed authentication is treated as conditionally trusted. Instead, a continuous dialogue is established in which the client and server side exchange not only data but also signals regarding the state of trust. The mobile application does not store long-term, unrestricted sessions. Still, it uses short-lived tokens that confirm rights only for a specific set of actions and for a limited period of time, with each request accompanied by checks for the integrity, freshness, and binding of these tokens to the device and context. On the server side, the client is regarded not as a single subject but as an aggregate of operations, and each such operation passes through a chain of checks that includes assessment of behavioral anomalies, account status, and compliance with regulatory constraints. As a result, the mobile application is transformed not into a thin terminal for accessing a trusted server zone, but into an active participant in a distributed trust management system, in which decisions are made stepwise and are revised whenever the usage context changes. The proposed architecture is shown in Fig. 2.

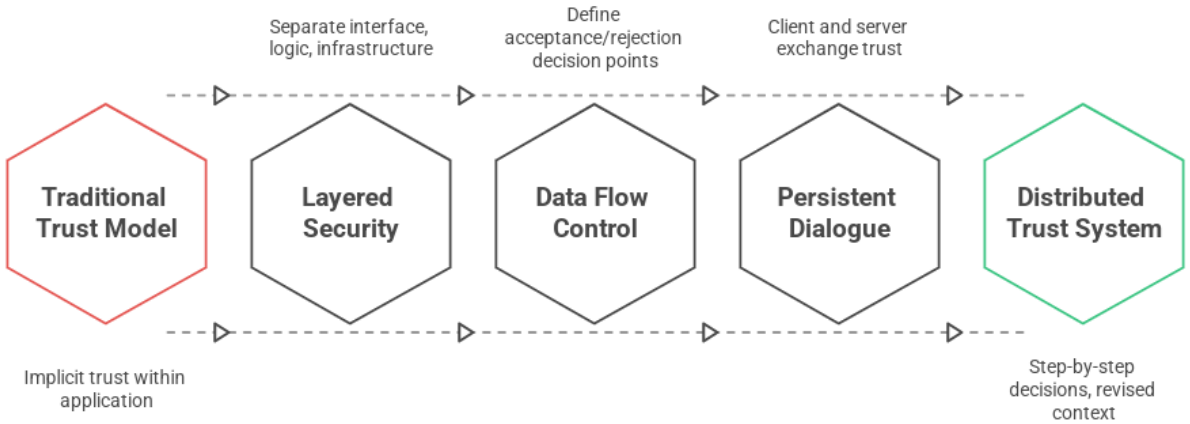

Figure 2: Proposed architecture of mobile application from the perspective of trust

The transfer of zero-trust principles to the mobile interface requires not only configuring network and cryptographic mechanisms, but also standardizing the interaction elements themselves. Reusable interface components become conduits for security policy: they define expected behavior, eliminate ad hoc improvisation by individual developers, and reduce the likelihood that a critical scenario will be implemented in a way that

bypasses standard rules. A navigation component acting as a shared container of screens can centrally control transitions, provide instant logout with session revocation for the entire application, intercept attempts to access screens with special trust-level requirements, and enforce additional checks. A unified action button, which varies by type based on the operation, allows strictly separating innocuous requests from financially significant steps, adds protection against repeated clicks, and embeds checks for session freshness and risk profile without duplicating this logic across screens. A component displaying a group of participants becomes not merely a decorative element but a bearer of status: it visually indicates whether a participant has passed verification, whether access has been blocked, and the participant's role in a joint financial operation. Thus, visual interface patterns are linked to an intuitive understanding of risk: the user begins to associate a particular appearance and behavior of elements with heightened caution requirements, and the system acquires a robust mechanism for applying uniform rules.

Beneath this surface layer, a software kit of components operates, serving as the core implementation of the zero-trust architecture across the entire product ecosystem. In contrast to fragmented security implementations across individual applications, a unified kit assumes responsibility for authentication, session management, authorization checks, encryption, signing, key material handling, and integration with risk analytics and anti-fraud subsystems. For payment and banking solutions, such kits, which support diverse payment scenarios, customer identification, and regulatory requirements, enable establishing a trust model once and then replicating it across different products without losing essential details. The architecture of the kit itself becomes critical: modularity empowers the evolution of individual mechanisms without disrupting the entire system, managed versioning and backward compatibility ensure smooth migration of legacy applications to new rules, and strict contracts with the server side define a predictable scheme of errors and responses to policy violations. As a result, interface components rely not on local heuristics but on precise decisions made by the shared security subsystem and act only as a facade for the more profound zero-trust logic.

The connection between the interface, the shared kit, and server logic becomes particularly evident in complex user scenarios where a single user action generates a long chain of operations with varying levels of risk. In implementing invitation mechanisms and establishing financial links between clients, the system must not only send and accept invitations but also confirm the identity of the addressee, verify ownership of the communication channel, grant only the minimally necessary rights, and provide continuous auditing of all subsequent actions. In this scenario, the navigation component manages transitions between stages, buttons trigger strictly typed requests to the shared kit, and the server side incrementally increases or restricts trust. A similar structure characterizes scenarios for granting access to accounts, documents, and payment details: the interface presents a transparent view to the user of what exactly is being shared and for how long, the shared kit issues temporary tokens with limited rights and handles repeated authentication checks, and the server subsystem stores and enforces access revocation rules. In cases involving cameras and media data for document scanning or video identification, the same linkage enables avoiding reliance on local storage, using secure containers, binding records to a specific session and context, and controlling the integrity and authenticity of the material. In this way, even the most complex and abuse-prone flows cease to be a set of disparate screens and become a sequence of steps in which zero-trust architecture is consistently realized at every level, from visual patterns to cryptographic checks on the server side.

Practical work on zero-trust architecture in a mobile application is reasonably initiated by constructing a so-called trust map, that is, an explicit description of where and how, in the current solution, assumptions about the trustworthiness of the user, device, environment, and server side arise. Such a map includes a list of screens and user scenarios, a classification of operations by criticality, a description of the data used and their movement, and a list of points where authentication, authorization checks, encryption, and logging occur. It is essential not to limit this to a formal inventory of functions but to attempt to trace the actual trajectories along which the user and the accompanying information move: how the user reaches money transfer operations, how the context changes when switching between accounts, and what happens upon returning to the application after prolonged inactivity. Such a map makes visible the gaps in which the interface trusts unverified data or passes superfluous data to lower layers, and it forms the basis for targeted rethinking of the architecture.

The next step involves extracting the stable elements from fragmented code that implements security policies and transforming them into reusable user interface components and shared library logic. Navigation containers, confirmation buttons, forms for entering sensitive data, and access-error handlers can and should be externalized into a unified set, with behavior predefined and aligned with zero-trust architecture requirements. In parallel, a shared security software kit is formed: a unified library for handling authentication, session lifecycles, authorization checks, encryption, and interaction with the risk management subsystem. The transition to such a kit rarely occurs instantaneously. Hence, a phased approach is appropriate: new scenarios are built from the outset on shared components and libraries, while legacy fragments are gradually wrapped in adapters that translate their interactions to new contracts without breaking existing behavior. It is vital to maintain transparency: development teams must clearly understand which paths to protected operations already traverse the shared security subsystem and which still rely on provisional solutions.

Special attention must be paid to implementing multi-step authentication and risk-oriented checks to avoid the crude dichotomy between constant input of confirmation codes and complete freedom of action after a one-time login. The logic for improving verification needs a sort of policy and risk levels. A device change, an unexpected deviation from regular behavior, or an attempt to perform unusual or high-risk actions should all trigger additional verification. Wherever possible, low-risk everyday actions should be done at a low level of trust. Additional features will enable policies to be easily turned on or off through feature flags, can be gradually rolled out to a subset of users, and can be easily rolled back when they degrade quality of service. Meanwhile, It is necessary to make some changes in-house and start documenting our decision processes, standardize on a vocabulary to describe levels of trust, and revise the trust map as new functions and dangers emerge. The proposed roadmap for mobile development teams is shown in Fig. 3.

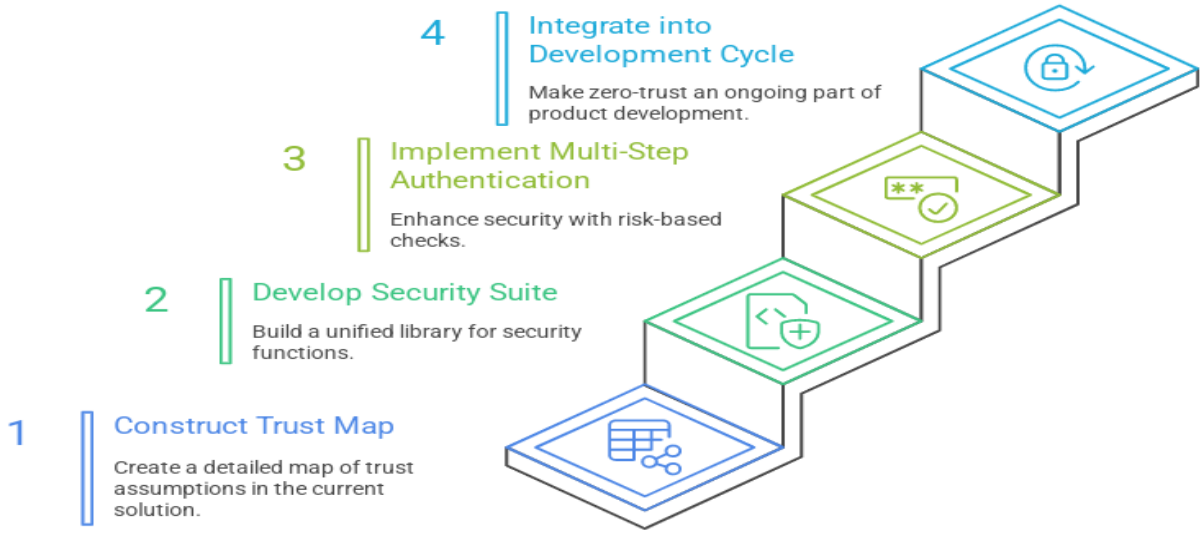

Figure 3: Roadmap for mobile development teams

Taken together, this transforms zero-trust architecture from a one-off project into a continuous process embedded in the standard development cycle of mobile products.

4. Conclusion

The leverage of high mobile phone penetration rates, the constant increase in vulnerabilities, and targeted attacks on mobile devices and applications imply that the perimeter approach is no longer feasible to maintain the required level of resilience of the financial infrastructure. For organizations dealing with highly liquid assets or highly sensitive personal information, the zero-trust model is not just a conceptual shift; it is a fundamental precondition for cyber-resilience. It systematically re-evaluates all initial assumptions about the levels of trust assigned to users, devices, networks, and the server environment; it enforces explicit controls for each request, continuously revokes unnecessary permissions, and continuously re-evaluates the trustworthiness of specific operations and their contexts.The interpretation of zero-trust architecture for mobile applications in the financial sector examined in the article demonstrates that its implementation is possible only with the simultaneous transformation of several system layers: from strict distrust toward the user device and network environment (detection of modified environments, use of secure storage and hardware modules, certificate pinning) and the treatment of sessions as short-lived, context-dependent entities to strict minimization of the privileges of functional modules and components. The key outcome is the shift of the center of gravity from an abstract trusted perimeter to a set of explicitly defined decision points distributed across the interface, domain logic, and infrastructural library layers, as well as the transformation of the mobile client into a fully fledged element of a distributed trust management system in which every financially significant operation passes through a sequential chain of checks.The practical value of the proposed approach is manifested in the described implementation roadmap: constructing a trust map of the existing application, identifying and standardizing interface and logical components through which security policies are implemented, forming a unified software kit for authentication, session management, authorization checks, and integration with risk-oriented subsystems, as well as phased introduction of multi-step authentication and contextual checks with controlled policy changes. Taken together, this shifts zero-trust architecture from a

one-time initiative to a continuous process embedded in the design and evolution cycles of mobile financial products, enabling financial organizations to systematically reduce the impact of mobile channel vulnerabilities and adapt to an increasingly complex threat landscape without sacrificing service convenience and inclusiveness.

References

[1] S. Burnett and K. Kinder, "Mobile Banking Statistics 2025: How Digital Finance is Redefining Banking," *CoinLaw*, 2025. https://coinlaw.io/mobile-banking-statistics/ (accessed Nov. 01, 2025).

[2] S. Schmelk *et al.*, "Privacy and Security of Mobile Banking: A PRISMA-Centric Review of Android Finance Applications," *Lecture notes in networks and systems*, vol. 1155, pp. 11–29, Jan. 2024, doi: https://doi.org/10.1007/978-3-031-73122-8_2.

[3] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," *NIST Special Publication 800-207*, Aug. 2020, doi: https://doi.org/10.6028/nist.sp.800-207.

[4] C. Daah, A. Qureshi, I. Awan, and S. Konur, "Enhancing Zero Trust Models in the Financial Industry through Blockchain Integration: A Proposed Framework," *Electronics*, vol. 13, no. 5, p. 865, Jan. 2024, doi: https://doi.org/10.3390/electronics13050865.

[5] Okta, "The State of Zero Trust Security 2023," Okta, 2023. Accessed: Nov. 04, 2025. [Online]. Available: https://www.okta.com/sites/default/files/2023-09/SOZT_Report.pdf

[6] Kaspersky, "Banking data theft attacks on smartphones triple in 2024," *Kaspersky*, Mar. 03, 2025. https://www.kaspersky.com/about/press-releases/banking-data-theft-attacks-on-smartphones-triple-in-2024-kaspersky-reports (accessed Nov. 05, 2025).

[7] D. Javaheri, M. Fahmideh, H. Chizari, P. Lalbakhsh, and J. Hur, "Cybersecurity threats in FinTech: A systematic review," *Expert Systems with Applications*, vol. 241, p. 122697, May 2024, doi: https://doi.org/10.1016/j.eswa.2023.122697.

[8] P. V. Falade and G. B. Ogundele, "Vulnerability Analysis of Digital Banks' Mobile Applications," *Arxiv*, Feb. 2023, doi: https://doi.org/10.48550/arxiv.2302.07586.

[9] Owasp, "MASTG-KNOW-0015: Certificate Pinning," *Owasp*. https://mas.owasp.org/MASTG/knowledge/android/MASVS-NETWORK/MASTG-KNOW-0015/ (accessed Nov. 08, 2025).

[10] Owasp, "Mobile App Authentication Architectures," *Owasp*. https://mas.owasp.org/MASTG/0x04e-Testing-Authentication-and-Session-Management/#stateless-authentication (accessed Nov. 09, 2025).