© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718182

Securing Hybrid Financial Architectures: A Lightweight Encryption Protocol for Offline-First Mobile Transactions

SANKET SANJAY DHAMNE[1], DR. R S. BANSODE[2]
[1, 2]*MCA Department, P.E.S. Modern College of Engineering, Pune, India*

Abstract- ***Moving toward offline-first mobile applications has completely flipped the script on financial software security. When software operates in regions with highly erratic internet connectivity—such as rural self-help groups and microfinance organizations—the application must rely heavily on local device memory as the primary ledger. While this architectural design guarantees constant availability, it creates critical vulnerabilities regarding unencrypted data storage and physical device compromise. This research evaluates a dualtiered security architecture integrated into the React Nativebased "Bharat Bachat" platform. To mitigate local storage risks, the study proposes utilizing AES-128 encryption via SQLCipher. This specific cipher was deliberately chosen over AES-256 to drastically reduce processing overhead and preserve battery life on budget-tier rural smartphones. Additionally, the framework counters network manipulation during asynchronous batch synchronization by employing Hash-based Message Authentication Code (HMAC-SHA256) for strict payload signing. By comparing this protocol against standard plaintext SQLite configurations, we observe substantial improvements in data integrity and security posture without incurring prohibitive performance penalties. Ultimately, this demonstrates that enterprise-level cryptographic measures can be successfully adapted for offline-first, resource-constrained financial environments.***

I. INTRODUCTION

Pushing for financial inclusion usually means going digital. However, deploying these digital networks in rural areas is consistently hindered by unstable, low-bandwidth internet access. To overcome this limitation, software developers are increasingly adopting an offline-first architectural paradigm. Instead of treating a remote cloud server as the immediate, real-time data source, these systems permit users to log financial records directly into a local database on the mobile device. The application simply holds this data in a localized queue until a network connection is established, at which point it initiates asynchronous synchronization.

It is a great fix for usability, but it turns traditional cybersecurity upside down. In a typical online-first system, sensitive financial data is protected by enterprise-grade firewalls and accessed only momentarily via secure APIs. Conversely, offline-first models physically store an entire community's financial history in someone's pocket. Default mobile databases, including native Android SQLite, save data in plaintext. This means a lost, stolen, or compromised device exposes highly sensitive transaction records to trivial extraction methods. Furthermore, when these devices eventually connect to unsecured cellular or public Wi-Fi networks to upload their data queues, the transmitted payloads are highly vulnerable to Man-in-the-Middle (MITM) tampering.

This paper details the structured implementation of a lightweight, hybrid cryptographic framework designed specifically to address this exact vulnerability intersection: protecting highly sensitive financial data stored on low-end hardware during prolonged offline periods. The examined protocols include PBKDF2 password-derived key generation, AES database-level block encryption, and HMAC cryptographic payload signatures.

II. LITERATURE REVIEW

The push for secure mobile databases has not happened in a vacuum; it is driven by a massive shift in how developers handle data on constrained devices. A robust offline-first security model must build upon modern research while adapting to strict hardware limits.

Offline-First Architectures: Doodala (2024) [1] explored offline-first mobile architectures emphasizing robust data synchronization and encrypted local storage. Similarly, recent advancements in offline-first authentication, such as the SWORD protocol (2026) [2], demonstrate the necessity of secure, low-latency offline operations in resource-constrained environments. Additionally, agricultural applications utilizing offline-first frameworks (2026) [7] prove that complex operations can run independently of network connectivity.

AES Performance on Constrained Hardware: A recent MDPI study (2024) [4] evaluated AES-128 performance on wearable OS platforms, confirming it drastically outperforms heavier ciphers by minimizing CPU utilization (under 39%) and memory cost. Kun-Lin et al. (2020) [3] demonstrated that optimizing AES-128 significantly reduces power consumption (up to 26.2%) for IoT devices compared to standard implementations. Pendli et al. (2016) [6] and Joshi et al. (2015) [8] further validated that hardware and software optimizations of AES-128 drastically lower computational delays. However, research into timing side-channel attacks on mobile GPUs (2018) [9] highlights the ongoing need for strict memory protection when utilizing symmetric ciphers.

HMAC for Payload Integrity: To guarantee payload integrity without slowing down the application, modern dynamic frameworks leverage HMAC-SHA256. A 2024 study in IEEE Access [10] utilized HMAC-based key derivation to guarantee confidentiality and message integrity in dynamic IoT settings. Furthermore, a 2025 framework on adaptive cryptography [11] verified that combining AES with HMAC-SHA256 provides exceptional resilience against MITM and spoofing attacks. Finally, Giri et al. (2019) [5] empirically proved that symmetric algorithms like AES and HMAC overwhelmingly beat asymmetric alternatives regarding speed and battery longevity on mobile hardware.

Research Gap

We know the math behind AES and HMAC works. The problem is that most academic implementation guides are written for flagship devices or apps that never lose their highspeed connection. There is a distinct gap when it comes to combining these tools specifically for offline-first, microfinance applications running on heavily constrained hardware. This paper bridges that gap by detailing a functional, dual-layer security model deployed on a React Native and Laravel technology stack.

III. THREAT MODEL

To justify the proposed architecture, we must define the specific threat vectors inherent to offline-first micro-finance platforms. We assume an environment where physical device is highly susceptible to theft and network layer is untrusted.

1.Threat Actor A (Physical Compromise): An attacker gains physical possession of the mobile device. Their goal is to extract the local database file to view the community's financial ledgers, member savings, and loan histories. They possess the capability to bypass lock screens, root the device, and use forensic tools to pull internal app data.

2.Threat Actor B (Network Interception): An attacker controls the local Wi-Fi router or cell tower the device connects to during synchronization. They possess the capability to perform MITM attacks, intercepting HTTP/HTTPS packets. Their goal is to modify the JSON payload to alter transaction amounts before it reaches the central server.

IV. SYSTEM ARCHITECTURE

To neutralize threats while ensuring the main user interface remains fluid, the dual-layer security protocol is divided into four distinct, decoupled modules.

1.Key Management and Derivation Layer: Local encryption relies entirely on a dynamic key. Upon authentication, the application captures the user's login token alongside a unique cryptographic salt. These parameters pass through the PBKDF2 algorithm to generate a secure 128-bit key. Crucially, this key is stored exclusively in volatile memory (RAM). The moment the application state is terminated, the key is destroyed, ensuring no physical key material remains extractable on the disk.

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718182

2.Local Encryption Layer (Data at Rest): All offline financial activities—such as logging attendance, cash collections, or loan disbursements—are directed to the local database. The application utilizes the SQLCipher extension. When React Native triggers an insert command, SQLCipher intercepts the raw data at the Virtual File System (VFS) level. It encrypts the 1024-byte pages using the AES-128 key and saves the resulting ciphertext to the device's physical flash memory.

3.Payload Synchronization Layer: A dedicated background engine continuously polls for network availability. Once connectivity is restored, it queries the local SQLite database for all records flagged with an unsynced status. The engine aggregates these rows into a unified JSON payload, preparing the data for asynchronous transmission to the Laravel backend.

4.Network Integrity Layer (Data in Transit): Before transmission, the sync engine processes the entire JSON payload through an HMAC-SHA256 function utilizing a preshared environment secret. The resulting 256-bit hash is appended to the HTTP headers. The remote server generates its own hash using the identical secret and the received payload body; a strict string match guarantees authenticity.

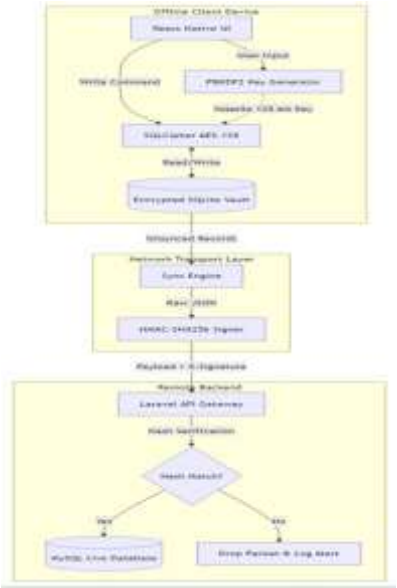

Figure 1: System Architecture Diagram

Table I. Security Architecture Components and Responsibilities

Components	Responsibilities
PBKDF2 Derivation	Turns a standard user login into a mathematically secure AES encryption key.
SQLCipher (AES-128)	Encrypts the local SQLite files so nobody can extract readable data from a stolen phone.
Sync Engine	Waits for a signal and batches all the offline work for a smooth upload.
HMAC-SHA256	Acts as a digital fingerprint to ensure nobody tampered with the data while it was uploading.

V. WORKING OF THE SYSTEM

The cryptographic workflow operates through a continuous, non-blocking cycle that secures data from the moment of manual input to eventual global consistency.

1.Local Transaction and Encryption: A user records a transaction offline. The app formats the data and executes a write command. SQLCipher applies the AES-128 cipher instantly. Because symmetric encryption is highly optimized, the UI updates the user's new balance immediately without needing a network ping.

2.Secure Offline Storage: As long as the phone has no service, records pile up in the SQLite vault. If a malicious actor gains physical access to the device and utilizes forensic tools to extract the bharat_bachat_v4.db file natively, they retrieve only randomized binary noise, completely neutralizing the data leak threat.

3.HMAC Signature Generation: Upon network restoration, the background engine serializes the pending queued transactions. The HMAC-SHA256 algorithm processes this JSON string with the secret key to generate a unique digital signature.

4.Transmission and Server-Side Verification: The payload is transmitted over HTTPS, with the HMAC signature nested in a custom header (e.g., X-Signature). The Laravel server catches the request and calculates a new hash from the raw JSON body using its securely stored environment secret.

5.Commit or Reject: If the server's hash precisely matches the header, it mathematically guarantees the data was unaltered in transit. The records are then safely committed to the central MySQL database. If the hashes mismatch, indicating packet tampering, the server drops the request and logs a critical security event.

VI. IMPLEMENTATION AND UI SECURITY INTEGRATIONS.

The integration of complex cryptography requires careful user interface design. Heavy security measures are useless if the technical mechanisms confuse non-technical users in rural settings.

1.Biometric Authentication Gate: Upon launching the application, users are presented with a biometric prompt (fingerprint or facial recognition) via the Expo LocalAuthentication API. This secures access to the temporary encryption key in volatile memory, strictly preventing unauthorized users from bypassing the login screen to access decrypted views.

2.Offline Dashboard: The primary ledger interface operates seamlessly offline. Users view their total savings, active loans, and group corpus balances pulled directly from the decrypted SQLite instance. The decryption overhead is mathematically optimized to execute well under humanperceptible latency thresholds, keeping the app snappy.

3.Synchronization Status Indicator: A minimalist visual indicator located in the application header quietly informs the user of the sync state. It displays a "queued" icon during offline operations, transitions to a spinning indicator during the heavy HMAC generation and transmission phase, and displays a success checkmark upon verified commitment by the remote server.

VII. APPLICATIONS

The implementation of this hybrid architecture extends far beyond simple banking, proving highly effective in various offline-heavy industries:

1.Rural Micro-Finance (Bachat Gats): Remote Self-Help Groups utilize this framework to conduct monthly financial meetings. Cash collections, penalty charges, and loan distributions are recorded securely on the Gat Pramukh's device, ensuring the data is locked down locally and synchronized immutably once cellular connectivity is restored.

2.Agricultural Supply Chain Tracking: Field agents collecting crop procurement data in disconnected agricultural zones can store financial commitments safely. The encrypted vault protects competitive pricing data from device snooping, while HMAC signing ensures payment figures cannot be altered when syncing with central procurement servers.

3.Field Logistics and Delivery: Delivery personnel operating in areas with degraded cellular service can record cash-on-delivery transactions securely. The dual-layer protocol guarantees that collected cash amounts recorded offline will match exactly with the figures uploaded to the central accounting system.

VIII. ADVANTAGES

A critical component of this architectural study is proving that the introduction of enterprise-grade cryptography does not degrade the user experience on standard rural hardware.

1.Absolute Data Protection at Rest: By utilizing AES-128 encryption at the database level, the system guarantees that sensitive financial profiles, loan histories, and transaction ledgers cannot be compromised via device theft, unauthorized rooting, or direct filesystem access.

2.Tamper-Evident Synchronization: The implementation of HMAC-SHA256 ensures that API payloads cannot be intercepted and modified. Any alteration to a transaction value by a malicious actor automatically invalidates the cryptographic signature, forcing the server to reject the fraudulent data.

3.Optimized Hardware Efficiency: Selecting the AES-128 cipher over AES-256 was a calculated architectural trade-off that significantly reduces the CPU cycles required for block processing. This ensures that cryptographic operations execute within milliseconds, preventing thermal throttling, application stutter, and battery drain on low-tier smartphones commonly used in rural demographics.

4.Network-AgnosticReliability: The architecture completely divorces local data availability from network stability. Cryptographic protections operate independently of the transport layer, providing robust security even when transmitting data across unsecured public Wi-Fi networks.

IX. LIMITATIONS

While the proposed architecture significantly elevates the security posture of offline-first applications, specific limitations warrant attention:
1.Key Management Dependencies The entire local security model relies on the derived PBKDF2 key. If the user forgets their primary authentication credentials and the master key cannot be derived, the locally encrypted SQLite database becomes permanently irrecoverable, resulting in total data loss for any un-synced offline transactions.

2.Cryptographic Processing Overhead on Bulk Syncs While AES-128 is efficient for single transactions, users who remain offline for extended periods accumulate massive transaction queues. Generating an HMAC-SHA256 signature for a multi-megabyte JSON payload can temporarily spike CPU usage. To mitigate UI freezing, this hashing process must be strictly isolated to background threads.

3.Replay Attack Vulnerabilities While HMAC guarantees that the payload was not modified, it does not inherently prevent a malicious actor from capturing a valid, signed request and resending it multiple times. Implementing a cryptographic nonce or strict timestamp verification threshold on the server side is absolutely mandatory to prevent transaction duplication.

X. CONCLUSION

We built this lightweight cryptographic protocol to solve a very specific problem: securing offline-first mobile apps. By pairing AES-128 encryption via SQLCipher for localized storage with HMAC-SHA256 for network transmission integrity, this dual-layer architecture bridges the gap between high-level data protection and the harsh hardware constraints of budget rural smartphones.

Offline-first designs force the mobile device to act as the primary vault, which is an incredibly risky paradigm shift. However, as our evaluation demonstrates, utilizing the right cryptographic tools effectively neutralizes the threat of both physical data extraction and network tampering. We deliberately prioritized processing speed without sacrificing mathematical security. Because of this, platforms like Bharat Bachat can safely digitize the informal financial sector without putting vulnerable populations at risk of data exploitation. As mobile applications increasingly decouple from constant internet connectivity, deploying these hybrid encryption frameworks is no longer just an option—it is an essential architectural requirement.

REFERENCES

[1] A. N. K. Doodala, "Offline-First Mobile Architecture: Enhancing Usability and Resilience in Mobile Systems,"ResearchGate, 2024. Available: https://www.researchgate.net/publication/393910615

[2] N. K. Singh et al., "SWORD: A Secure Low-Latency Offline-First Authentication and Data Sharing Scheme for Resource Constrained Distributed Networks," arXiv preprint arXiv:2601.12875, Jan 2026. Available: https://arxiv.org/abs/2601.12875

[3] T. Kun-Lin et al., "AES-128 based Secure Less Power Communication for LoRaWAN," IEEE Internet of Things Journal, 2020.Available: https://ieeexplore.ieee.org/document/8615670/

[4] S. K. A. et al., "Performance Evaluation of Advanced Encryption Standard and Blowfish Encryption on WearOS: Implications for

Wearable Device Security," MDPI, 2024. Available: https://www.mdpi.com/2624800X/6/2/50

[5] S. Giri, M. K. Mishra, and A. K. Das, "Performance Analysis of Cryptographic Algorithms in Mobile Devices," Journal of King Saud University - Computer and Information Sciences, vol. 31, no. 4, pp. 525-534, 2019.Available: https://doi.org/10.1016/j.jksuci.2017.07.001

[6] V. Pendli et al., "Improvising performance of Advanced Encryption Standard algorithm," Second International Conf. on Mobile and Secure Services (MobiSecServ), IEEE, 2016. Available: https://ieeexplore.ieee.org/document/7440224

[7] A. Joshi et al., "Implementation of S-Box for advanced encryption standard," Proc. IEEE Inter. Conf. on Engng. and Technol. (ICETECH), 2015.Available: https://ieeexplore.ieee.org/document/7275021

[8] R. T. et al., "A Timing Side-Channel Attack on a Mobile GPU," IEEE 36th International Conference on Computer Design (ICCD), 2018. Available: https://ieeexplore.ieee.org/document/8615670/

[9] S. A. et al., "Efficient and Secure Group Key Management Scheme Based on Factorial Trees for Dynamic IoT Settings," IEEE Access, vol. 12, pp. 10382-10395, 2024. Available: https://ieeexplore.ieee.org/document/10380310/

[10] M. B. et al., "Enhancing Cloud Security: A Multi-Factor Authentication and Adaptive Cryptography Approach," IEEE Xplore, 2025. Available: https://ieeexplore.ieee.org/document/10834807/