ISSN : 2455-135X

A Non-Interactive PKCE Architecture for Secure OAuth2 Authorization in Mobile Applications

Sangeeta Manna (nee Datta), BSc. Hons. (Cal.), Master of Computer Applications (UTech. WB. India)

Senior Software Programmer in Banking, Insurance & Service-related IT Industry

ABSTRACT

Modern mobile applications frequently adopt OAuth2 with Proof Key for Code Exchange (PKCE) to protect authorization code flows from interception attacks. However, many production mobile systems already provide a native username and password login interface, and migrating to a fully browser-based OAuth login experience can significantly impact usability and require major architectural redesign. This paper proposes a Non-Interactive PKCE architecture that preserves the native mobile authentication experience while maintaining strong OAuth2 security guarantees. The architecture separates authentication and authorization responsibilities by introducing a cryptographically protected trust artifact called oaCode, generated by an App Authentication Server (AUS). The oaCode securely transfers authenticated context to an App OAuth2 Server (AOS), allowing the OAuth authorization flow to proceed without additional user interaction. PKCE verification is enforced during token exchange to ensure that intercepted authorization codes cannot be redeemed by malicious applications. The proposed architecture demonstrates how existing mobile authentication systems can integrate OAuth2 authorization while maintaining both security and usability.

Keywords

OAuth2, PKCE, Mobile Security, Authorization Code Flow, Authentication Architecture, Identity Server, Secure Mobile Applications, oaCode (OAuth Authorization Code Bridge)

I. INTRODUCTION

Mobile applications increasingly depend on secure authorization frameworks to access protected APIs and cloud services. OAuth2 has become the industry standard for delegated authorization, and its **Authorization Code Flow with Proof Key for Code Exchange (PKCE)** is widely recommended for public clients such as mobile applications.

PKCE mitigates authorization code interception attacks by cryptographically binding the authorization request to the client instance that initiated the authentication flow. Because mobile applications cannot safely store client secrets, PKCE ensures that only the legitimate application instance can exchange an authorization code for tokens.

Despite these advantages, many real-world mobile applications already implement their own **native credential-based login interface**. Replacing this experience with a browser-based OAuth login flow can disrupt the user experience and require substantial redesign of existing authentication systems.

This research proposes a **Non-Interactive PKCE architecture** that allows mobile applications to preserve their native login experience while still enforcing OAuth2 [1] PKCE-based authorization security. The architecture introduces a secure bridge between authentication and authorization, enabling seamless integration of existing identity systems with modern OAuth frameworks.

II. PROBLEM STATEMENT

Integrating OAuth2 with existing mobile authentication systems presents several practical challenges.

Traditional OAuth implementations assume that user authentication occurs directly within the authorization server. In many enterprise environments, however, authentication is handled by independent identity services that manage credential validation, account status verification, and security policies.

When mobile applications attempt to integrate OAuth authorization [2] with these systems, several issues arise:

1. Existing applications already rely on native credential-based login interfaces.
2. Migrating to browser-based OAuth authentication flows may degrade user experience.

ISSN : 2455-135X

3. Authorization codes may be intercepted during redirect flows if additional security mechanisms are not applied.
4. OAuth authorization servers should not handle user passwords directly in many enterprise architectures.

Therefore, a mechanism is required that preserves native authentication while enabling secure OAuth authorization and enforcing PKCE protection during token issuance.

III. AIMS AND OBJECTIVES

The primary aim of this research is to design a secure mobile authorization architecture that integrates existing authentication systems with OAuth2 PKCE authorization flows.

The objectives of this research are:

- To separate authentication and authorization responsibilities in mobile security architectures.
- To introduce a secure mechanism for transferring authenticated user context between systems.
- To enforce PKCE verification during token exchange to prevent authorization code interception.
- To maintain the existing native mobile login experience without additional authentication prompts.
- To ensure that user credentials are never processed within the OAuth authorization system.

IV. METHODOLOGY & DESIGN

The proposed architecture consists of three primary components:

Component	Responsibility
App Authentication Server (AUS)	User identity verification
App OAuth2 Server (AOS)	OAuth authorization handling
Identity Server (IS4)	Secure token issuance

The architecture introduces a trust artifact called **oaCode** that bridges the authentication and authorization processes.

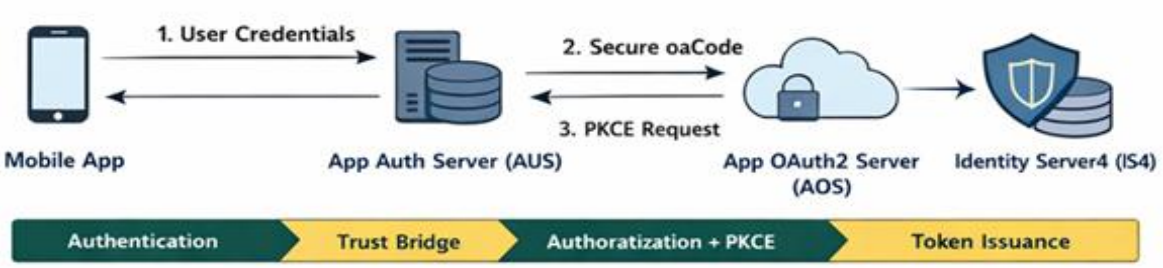

Figure 1: System Architecture Diagram

Authentication Phase

The authentication process begins within the mobile application where the user enters their credentials through the application's native interface. These credentials are transmitted to the **App Authentication Server (AUS)**.

During this phase:

- OAuth authorization has not yet started.
- No access or refresh tokens are issued.

ISSN : 2455-135X

- No browser-based authentication is required.

The AUS performs credential validation, verifies account status, and ensures that security prerequisites are satisfied.

Trust Bridge Generation

After successful authentication, the AUS generates a short-lived artifact called **oaCode**. The oaCode acts as a cryptographically protected bridge that transfers authenticated user context to the OAuth authorization system.

The oaCode possesses the following properties:

- encrypted payload
- JWT-backed structure
- strict expiration window
- single-purpose usage

Importantly, oaCode does not grant access privileges and cannot be used to obtain tokens directly.

OAuth Authorization Initiation

Once the mobile application receives oaCode, it initiates the OAuth authorization process with PKCE [2]. The application generates a high-entropy **code_verifier** and derives a **code_challenge** using the SHA-256 transformation.

The authorization request includes:

- client identifier
- redirect URI
- requested scopes
- code challenge
- state parameter
- oaCode

The request is then sent to the **App OAuth2 Server (AOS)**.

Authorization Processing

Upon receiving the authorization request, the AOS validates and decrypts the oaCode. If the oaCode is valid and within its time window, the server establishes a trusted authenticated context.

ISSN : 2455-135X

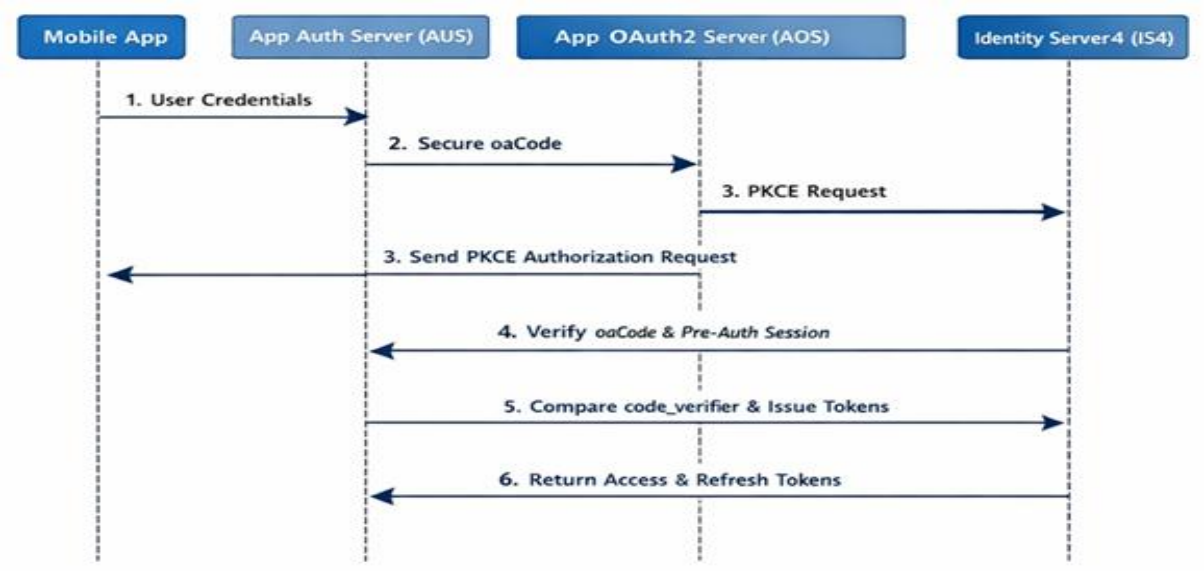

Figure 2: PKCE Flow Sequence Diagram

Because authentication has already been verified by AUS, the authorization server does not prompt the user for credentials or display consent screens. The OAuth server proceeds directly to issuing an **authorization code**.

The authorization code is returned to the mobile application through the registered deep link redirect.

V. RELATED WORKS & IMPLEMENTATION

OAuth2 has become the standard framework for secure API authorization. The **PKCE extension (RFC 7636)** was introduced to address authorization code interception attacks, particularly for public clients such as mobile and browser-based applications. Traditional OAuth architectures assume that authentication occurs inside the authorization server itself. However, many enterprise environment's separate identity management from authorization infrastructure.

Several related mechanisms exist, including session hints, login hints, and backchannel authentication techniques. These approaches assist in reducing user interaction but do not provide a secure cryptographic bridge between independent authentication and authorization systems. The architecture proposed in this research introduces a unique mechanism in the form of **oaCode**, which securely transfers authenticated identity context from the authentication server to the OAuth authorization pipeline. This enables non-interactive OAuth authorization while maintaining PKCE-based protection during token exchange.

VI. RESULT ANALYSIS

The proposed architecture provides multiple security and usability benefits.

First, authentication and authorization responsibilities are clearly separated. The authentication server handles identity verification, while the OAuth server focuses exclusively on authorization and token management.

Second, the oaCode artifact securely transfers authentication state without exposing user credentials to the OAuth system. Third, PKCE ensures that intercepted authorization codes cannot be redeemed by unauthorized applications, as token issuance requires the original **code_verifier** generated by the mobile client.

Fourth, the architecture preserves the native mobile login experience, preventing additional authentication prompts or browser-based login flows.

These characteristics demonstrate that the architecture achieves a balance between **enterprise system integration, user experience, and modern OAuth security requirements**.

VII. SECURITY ANALYSIS

The proposed architecture was designed with multiple security layers to mitigate common threats associated with OAuth2 authorization flows in mobile applications. This section evaluates how the system addresses potential attack vectors.

ISSN : 2455-135X

Authorization Code Interception

One of the primary threats in OAuth2 authorization flows is the interception of authorization codes during redirect operations. If an attacker obtains the authorization code before the legitimate application redeems it, they may attempt to exchange it for access tokens. The proposed architecture mitigates this threat using Proof Key for Code Exchange (PKCE). During authorization initiation, the mobile application generates a high-entropy code verifier and derives a corresponding code challenge using SHA-256. The authorization server stores the challenge and validates the verifier during token exchange. Because the attacker does not possess the original code verifier, intercepted authorization codes cannot be redeemed successfully.

Credential Exposure Prevention

Traditional OAuth implementations often require user credentials to be submitted directly to the authorization server. In the proposed architecture, authentication occurs exclusively within the App Authentication Server (AUS). The OAuth authorization server never receives or processes user passwords, thereby reducing the attack surface associated with credential leakage.

Authentication Context Protection

The oaCode artifact functions as a secure bridge between authentication and authorization systems. The oaCode is encrypted, time-bound, and cryptographically protected, ensuring that the authentication context cannot be altered or reused outside its intended authorization window.

Replay Attack Prevention

Replay attacks occur when an attacker attempts to reuse previously captured authentication or authorization artifacts. The proposed architecture prevents replay attacks through the following mechanisms:

- oaCode expiration and single-use validation
- PKCE code [5] verifier binding
- short-lived authorization codes

These mechanisms collectively ensure that previously captured artifacts cannot be reused to obtain tokens.

Token Issuance Control

Access tokens and refresh tokens are issued only after successful PKCE verification during the token exchange phase. This ensures that only the legitimate application instance that initiated the authorization request can obtain tokens.

Through these security controls, the architecture provides layered protection against common OAuth-related threats while maintaining usability for mobile applications.

VIII. FUTURE WORK

Future improvements to this architecture may include:

- Integration with multi-factor authentication systems
- biometric authentication support for mobile devices
- dynamic risk-based authorization policies
- stronger cross-system trust validation mechanisms
- integration with distributed identity providers

Further research may also explore large-scale deployment scenarios and evaluate performance characteristics in high-volume mobile environments.

IX. CONCLUSION

This paper presents a **Non-Interactive PKCE architecture** designed for mobile applications that already maintain native authentication systems. By separating authentication from authorization and introducing a secure trust artifact known as

ISSN : 2455-135X

oaCode, the architecture enables seamless integration of existing identity infrastructures with OAuth2 PKCE authorization flows.

The proposed design ensures that token issuance remains strictly protected by PKCE verification while preserving the native mobile login experience. This approach demonstrates that strong OAuth security guarantees can coexist with enterprise authentication architectures without requiring disruptive changes to existing mobile applications.

The architecture provides a practical and secure framework for integrating modern OAuth authorization mechanisms into real-world mobile systems.

References

[1] Lodderstedt, T., McGloin, M., & Hunt, P.

OAuth 2.0 Threat Model and Security Considerations.

[2] Authorization Code Flow with PKCE.

https://auth0.com/docs/flows/proof-key-for-code-exchange-pkce

[3] IdentityServer4 Documentation.

https://identityserver4.readthedocs.io

[4] OAuth 2.0 Security Best Current Practice,

Lodderstedt, T., et al.
OAuth 2.0 Security Best Current Practice.

[5] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., & Mortimore, C.
Proof Key for Code Exchange by OAuth Public Clients (PKCE).
Internet Engineering Task Force (IETF), 2015.
https://datatracker.ietf.org/doc/html/rfc7636
