

## App Information

- App Name: GroopTroop
- Bundle ID: com.nick.grooptroop
- Version: 1.0.0

## Encryption Overview

GroopTroop implements end-to-end encryption for group messaging using standard, internationally accepted cryptographic algorithms. The app does not use any proprietary encryption methods.

### Encryption Implementation Details

#### Cryptographic Libraries Used

```
// Primary encryption library
import nacl from 'tweetnacl'; // NaCl (Networking and Cryptography Library)
import util from 'tweetnacl-util';
import * as Crypto from 'expo-crypto'; // React Native crypto wrapper for iOS/Android APIs
```

#### Libraries:

TweetNaCl: Industry-standard NaCl implementation (Daniel J. Bernstein's Networking and Cryptography Library)

Expo Crypto: React Native wrapper for platform-native cryptographic APIs

Platform APIs: iOS CryptoKit and Android cryptographic services

#### Encryption Algorithms

Component	Algorithm	Key Size	Standard
Symmetric Encryption	XSalsa20 + Poly1305 (via NaCl secretbox)	256-bit	RFC 7539, NaCl standard

Asymmetric Encryption	Curve25519 + XSalsa20 + Poly1305 (via NaCl box)	256-bit	RFC 7748, NaCl standard
Key Generation	Platform CSPRNG (iOS: SecRandomCopyBytes, Android: SecureRandom)	256-bit	FIPS 140-2
Hashing	SHA-256	256-bit	FIPS 180-4

## Key Management Architecture

### User Key Pair Generation

```
// Each user gets an asymmetric key pair for secure key exchange
static async generateAndStoreUserKeys(userId: string): Promise<UserKeys | null> {
  const keyPair = nacl.box.keyPair(); // Curve25519 key pair
  // Keys stored securely using platform keychain
}
```

### Group Key Generation

```
// Each group gets a symmetric key for message encryption
static async generateGroopKey(groopId: string): Promise<string | null> {
  const key = await Crypto.getRandomBytesAsync(nacl.secretbox.keyLength); // 256-bit
  // Stored securely using iOS Keychain/Android Keystore
}
```

## Encryption Processes

### Message Encryption (Group Messages)

Algorithm: XSalsa20 stream cipher with Poly1305 MAC

Process:

- Generate random 192-bit nonce using platform CSPRNG
- Encrypt message with group symmetric key
- Authenticate with Poly1305 MAC
- Format: nonce (24 bytes) + encrypted\_data + mac (16 bytes)

## Key Exchange (New Member Onboarding)

Algorithm: Curve25519 ECDH + XSalsa20 + Poly1305

Process:

- Admin encrypts group key using new member's public key
- Uses sender's private key and recipient's public key
- Creates shared secret via ECDH
- Encrypts group key with shared secret

## Security Features

### Forward Secrecy

- New random nonce for each message
- Group keys can be rotated independently
- No key reuse across messages

### Authentication

- All encrypted data includes Poly1305 MAC for integrity
- Public key cryptography ensures sender authenticity
- Platform-level secure storage protects private keys

### Key Storage Security

// Keys stored using platform secure storage

- - iOS: Keychain Services (kSecAttrAccessibleWhenUnlockedThisDeviceOnly)
- - Android: Android Keystore (AES encryption with TEE/HSM when available)



## Platform Integration

### iOS Implementation






- Uses iOS CryptoKit and Security Framework
- Keys stored in iOS Keychain with device-only access
- CSPRNG via SecRandomCopyBytes
- Respects App Transport Security requirements

## Android Implementation

- Uses Android Keystore and Crypto APIs
- Hardware-backed security when available (TEE/HSM)
- CSPRNG via SecureRandom
- Follows Android security best practices

## Compliance & Standards

### International Standards Compliance

-  FIPS 140-2: Key generation using certified CSPRNGs
-  RFC 7748: Curve25519 elliptic curve cryptography
-  RFC 7539: ChaCha20-Poly1305 authenticated encryption
-  NaCl Standard: Daniel J. Bernstein's NaCl specification
-  IETF Standards: All algorithms approved by IETF

### Export Control Classification

- ECCN: 5D002 (standard cryptographic software)
- Classification: Mass market cryptographic software
- Export License: Not required (standard algorithms, publicly available)

## Implementation Verification

### Code References

- Encryption Service: EncryptionService.tsx
- Key Exchange Service: KeyExchangeService.tsx
- Authentication Service: AuthService.tsx
- Info.plist Setting: ITSAAppUsesNonExemptEncryption = false

### Algorithm Verification

# Verify TweetNaCl version and authenticity

- npm list tweetnacl

# Current: tweetnacl@1.0.3 (official implementation)

# Verify Expo Crypto integration

- npm list expo-crypto

# Uses platform-native cryptographic APIs



## Technical Summary

GroopTroop uses only standard, internationally accepted cryptographic algorithms:

- No proprietary encryption - All algorithms are IEEE/IETF/NIST approved
- Standard implementations - TweetNaCl is the reference NaCl implementation
- Platform integration - Uses iOS/Android native crypto APIs via React Native
- Export compliance - Qualifies for mass market software exemption
- Open source libraries - All cryptographic code is publicly auditable



## Conclusion

GroopTroop's encryption implementation uses only standard, well-vetted cryptographic algorithms and libraries. The app does not implement any custom cryptographic protocols or use any proprietary encryption methods. All encryption is performed using internationally accepted standards through established, open-source libraries.

This implementation qualifies for the standard software exemption under U.S. export control regulations and does not require additional export documentation.

Document Prepared: June 2025

App Version: 1.0.0

Contact: Nick.sanders.a@gmail.com