ZIXT CHAT

Technical Overview

Quantum-Resistant Architecture

Zixt Chat is built on a foundation of post-quantum cryptography, designed to resist attacks from both classical and quantum computers. This document provides a technical overview of our architecture, security protocols, and implementation details.

Post-Quantum Cryptography

Zixt employs NIST-standardized post-quantum cryptographic algorithms:

ML-KEM-1024 (Module-Lattice Key Encapsulation Mechanism)

Based on the CRYSTALS-Kyber algorithm, ML-KEM-1024 provides 256-bit security against quantum attacks. It is used for all key exchange operations and message encryption in Zixt Chat.

ML-DSA-65 (Module-Lattice Digital Signature Algorithm)

Based on the CRYSTALS-Dilithium algorithm, ML-DSA-65 creates quantum-resistant digital signatures for message authentication and blockchain verification.

System Architecture

Zixt Chat is built on a distributed architecture with several key components:

- Secure Messaging Core: End-to-end encrypted messaging using ML-KEM-1024
- Blockchain Verification Layer: Message integrity verification using immutable ledgers
- Distributed Hash Table (DHT): Peer discovery and network resilience
- Authentication System: Multi-factor authentication with TOTP and PSK options
- Admin Management Console: Enterprise control and security management

Technical Implementation

Backend Technology Stack

Zixt Chat is built on a modern Python stack using Flask for web services, SQLAlchemy for database abstraction, and custom implementations of cryptographic protocols. The system runs on PostgreSQL for data persistence with Redis for caching and real-time features.

Distributed Message Delivery

Messages are encrypted with the recipient's public key, signed with the sender's private key, and delivered through a redundant network of peer nodes. Each message carries a unique signature that can be verified

against the blockchain record.

Blockchain Integration

Each conversation thread has its own blockchain for message verification. When a message is sent, a transaction is added to the blockchain containing a hash of the message and its ML-DSA-65 signature. This creates an immutable record that can be used to verify message integrity without revealing content.

DHT Network

The Distributed Hash Table (DHT) enables peer discovery and network resilience. Each node maintains connections to multiple peers, creating a redundant network that can continue to function even if individual nodes go offline.

Security Measures

In addition to post-quantum encryption, Zixt implements multiple layers of security:

- Multi-Factor Authentication: TOTP (Time-based One-Time Password) authentication compatible with standard authenticator apps
- Pre-Shared Key (PSK): Optional additional authentication layer using pre-shared keys
- Backup Codes: Secure recovery process using one-time backup codes
- Session Key Rotation: Regular rotation of session keys for forward secrecy
- Automatic Vulnerability Scanning: Real-time security monitoring and alerts
- Secure Key Storage: Encrypted private key storage using password-derived keys

Message Flow Architecture

A typical message flow in Zixt Chat follows these steps:

1. User A composes a message to User B
2. Message is encrypted with User B's ML-KEM-1024 public key
3. Message is signed with User A's ML-DSA-65 private key
4. Message hash and signature are recorded on the thread's blockchain
5. Encrypted message is distributed through the DHT network
6. User B's client receives the encrypted message
7. Message is decrypted using User B's ML-KEM-1024 private key
8. Signature is verified against User A's ML-DSA-65 public key
9. Message integrity is verified against the blockchain record

Scalability and Performance

Zixt Chat is designed for enterprise-scale deployment with these performance optimizations:

- Sharded Blockchain: Messages are distributed across multiple blockchain shards for improved throughput
- Distributed Consensus: Proof-of-Stake validation for efficient transaction verification
- Adaptive DHT Routing: Optimized peer discovery based on network conditions
- Message Caching: Local caching of frequently accessed messages for improved responsiveness
- Federation Capabilities: Support for cross-organization messaging with maintained security boundaries

Enterprise Integration

Zixt Chat can be integrated with existing enterprise systems:

- Directory Services: LDAP/Active Directory integration for user management
- Single Sign-On: SAML and OAuth2 support for enterprise authentication
- Audit Logging: Comprehensive event logging for compliance requirements
- API Access: Secure API for integration with third-party applications
- On-Premise Deployment: Support for fully on-premise deployment in secure environments

Technical Contacts

For technical inquiries, please contact:
Email: tech@zixt.app
Schedule a technical demo: https://zixt.app/