VAULT Windows Desktop Application - Complete Development Plan

Title: VAULT Windows Desktop Application - Production-Ready Development Plan

Version: 1.0.0

Date: February 1, 2026

Status: PRODUCTION-READY

Security Level: **MILITARY-GRADE**

Platform: Windows Desktop (.NET 8 WPF)

1. Executive Summary

This document outlines the comprehensive development strategy for the VAULT Windows Desktop Application. VAULT is a military-grade, end-to-end encrypted messaging platform designed for absolute privacy and zero-trace data handling. The Windows client is built to leverage native OS performance while maintaining strict security boundaries.

Key highlights of the implementation include:

- **Military-Grade Security:** Full implementation of the Signal Protocol for asynchronous messaging combined with Post-Quantum Cryptography (ML-KEM-768) to protect against future threats.
- **Zero-Knowledge Architecture:** The server stores no metadata or message content. Identity verification is handled via Zero-Knowledge Proofs.
- **Native Performance:** Built using .NET 8 and WPF (Windows Presentation Foundation) for a responsive, high-performance user experience that integrates seamlessly with the Windows environment.
- **Cost-Effective Deployment:** A strategic deployment plan utilizing free-tier services (Mailgun for transactional emails, Render/Railway for backend services) to ensure low operating costs without compromising functionality.

2. Architecture Overview

2.1 Application Architecture

The application follows a strictly layered architecture based on Clean Architecture principles to ensure separation of concerns, testability, and maintainability.

- **Presentation Layer (WPF MVVM):** Handles UI rendering and user interaction using the Model-View-ViewModel pattern. This layer contains no business logic.

- **Domain Layer (Core):** Contains the pure business logic, entities, and interfaces. This layer has no dependencies on external frameworks or the UI.
- **Data/Infrastructure Layer:** Implements the interfaces defined in the Domain layer. Manages data persistence (SQLite), network communication (WebSocket/REST), and cryptographic operations.

2.2 Technology Stack Decisions

Component	Technology	Justification
Language	C# 12 (.NET 8)	Long-term support (LTS), high performance, strong type safety, and rich ecosystem.
UI Framework	WPF + ModernWPF	Mature desktop framework with hardware acceleration and "Modern" UI styling capabilities.
Architecture	MVVM + Clean Arch	Standard industry pattern for decoupled, testable, and maintainable XAML-based applications.
Cryptography	libsignal-protocol-dotnet + BouncyCastle	Proven implementation of the Signal Protocol and comprehensive crypto library for PQC support.
Database	SQLite + SQLCipher	Serverless, local storage with transparent 256-bit AES encryption for data at rest.
Network	SignalR / WebSocket	Real-time, bi-directional communication required for instant messaging and presence.
DI Container	Microsoft.Extensions.DI	Lightweight, built-in dependency injection container for managing object lifecycles.
Email Service	Mailgun	Reliable transactional email with a generous free tier (100 emails/day) for auth flows.
Identity	ASP.NET Core Identity	Robust identity management system supporting multi-device login scenarios.

2.3 Data Flow Architecture

The message flow ensures that unencrypted data never leaves the client device:

1. **Input:** User types a message in the WPF UI.
2. **Processing:** The ViewModel passes the plaintext to the Domain layer.
3. **Encryption:** The Cryptography service (Infrastructure) encrypts the payload using the recipient's public identity key and the current session state (Double Ratchet).

4. **Transmission:** The encrypted binary blob is sent via WebSocket to the Relay Server.
5. **Delivery:** The server routes the blob to the recipient without inspection (it holds no decryption keys).

3. Core Features Implementation

3.1 Messaging Features

- **1-on-1 Encrypted Chat:** Standard messaging using Signal Protocol with forward secrecy.
- **Group Messaging:** Implementation of Messaging Layer Security (MLS) IETF standard to support efficient E2EE for groups up to 10,000 members.
- **Media Sharing:** AES-256-GCM encrypted transfer of photos, videos, and documents. Thumbnails generated locally in secure memory.
- **Voice Messages:** On-device encrypted audio recording using secure temporary storage.
- **Voice/Video Calls:** Peer-to-peer WebRTC implementation with SRTP (Secure Real-time Transport Protocol).
- **Message Reactions:** Private reactions implemented as distinct encrypted message types.
- **Message Editing & Deletion:** Secure protocols for editing history and "Delete for Everyone" with cryptographic revocation instructions.
- **Forwarding:** Secure re-encryption of content for new recipients without exposing plaintext.

3.2 Security Features (9 Layers)

Layer	Implementation Details
1. App Hardening	Code obfuscation, anti-debugging checks, and tamper detection.
2. Transport	TLS 1.3 only, strict Certificate Pinning, and custom trust validation.
3. Protocol	Signal Protocol with Double Ratchet Algorithm for perfect forward secrecy.
4. Post-Quantum	Hybrid key exchange using X25519 and ML-KEM-768 (Kyber).
5. Zero-Knowledge	zk-SNARKs implementation for proving identity without revealing keys.
6. Hardware	Integration with Windows TPM 2.0 for root key protection.
7. Memory	SecureString for keys, immediate zeroization of buffers after use.
8. Storage	SQLCipher for DB encryption; DPAPI for local secret management.
9. Physical	Windows Hello integration (Biometric/PIN) and configurable auto-lock.

3.3 Premium Features

- **Disappearing Messages:** Granular TTL control (5 seconds to 1 week).
- **Scheduled Messages:** Local queuing of encrypted messages for future delivery.

- **Custom Themes:** Built-in High-Contrast, Dark, Light, and AMOLED Black modes.
- **Encrypted Backup:** Client-side AES-256 encrypted backups uploadable to user cloud storage.
- **Full-Text Search:** Secure, local-only inverted index search (FTS5).
- **Multi-Device Sync:** Real-time synchronization of message history across up to 5 devices.
- **Decoy Mode:** Alternative password entry opens a dummy account with fake data.

4. Project Structure

```
VaultWindows/
├── src/
│   ├── Vault.Desktop/          # Main WPF Application (Presentation)
│   │   ├── Views/              # XAML Windows and Controls
│   │   ├── ViewModels/         # MVVM ViewModels
│   │   └── App.xaml
│   ├── Vault.Core/             # Domain Layer
│   │   ├── Entities/           # Business Objects
│   │   └── Interfaces/         # Repository & Service Interfaces
│   ├── Vault.Infrastructure/   # Data Layer
│   │   ├── Data/               # SQLite Context
│   │   └── Repositories/       # Data Access Implementation
│   ├── Vault.Cryptography/     # Security Layer
│   │   ├── Signal/             # Signal Protocol Implementation
│   │   └── Primitives/         # AES, Hashing, Random
│   └── Vault.Services/         # Service Integrations
│       ├── Network/            # WebSocket Client
│       └── Email/              # Mailgun Service
├── tests/                      # Unit and Integration Tests
├── docs/                       # Technical Documentation
└── deployment/                 # Installer Scripts (WiX/Inno Setup)
```

5. Security Implementation Details

5.1 Cryptographic Components

The core cryptographic engine uses a hybrid approach. The established Signal Protocol handles session management and ratcheting. To address future quantum threats, the initial key exchange is augmented with ML-KEM-768 (Kyber), ensuring that today's encrypted traffic cannot be decrypted by future quantum computers.

5.2 Secure Storage

Data at rest is protected using a multi-tiered approach:

- **Database:** The SQLite database is encrypted via SQLCipher. The encryption key is never stored in plaintext.
- **Key Management:** The database key is wrapped using a master key derived from the user's password and stored securely in the Windows Data Protection API (DPAPI).
- **Files:** Media files are stored individually on the file system, encrypted with AES-256-GCM with unique keys stored in the encrypted database.

5.3 Network Security

All network communications are strictly enforced over TLS 1.3. Certificate pinning is implemented to prevent Man-in-the-Middle (MitM) attacks by compromising Certificate Authorities. WebSocket connections are authenticated using short-lived JWTs derived from the cryptographic identity proof.

6. Email Integration (Mailgun)

6.1 Provider Selection: Mailgun

Mailgun has been selected for transactional email services due to its developer-friendly API and robust free tier.

- **Free Tier Limits:** 100 emails per day (approx. 3,000/month), sufficient for initial user base and testing.
- **Requirements:** No credit card required for the foundational trial/flex plan.
- **Integration:** RESTful API implementation using `Vault.Services`.

6.2 Email Features

The email service handles critical authentication workflows:

- **Sender Identity:** noreply@b2g-vault
- **Account Verification:** OTP codes for new device registration.
- **Security Alerts:** Notifications for new logins or key changes.
- **Device Authorization:** Magic links to authorize new desktop clients.

7. Identity & Authentication

7.1 Identity Model

VAULT uses a "One Identity, Multi-Device" model. A single user account (Identity Key) acts as a root of trust for multiple Device Keys (Signed PreKeys).

- **Sessions:** Each device maintains its own Double Ratchet session with other users.
- **Syncing:** The "Sesame" algorithm manages sending messages to all of a user's active devices.

7.2 Authentication Methods

- **Primary:** Email and strong password (hashed with Argon2id).
- **Local Access:** Windows Hello (Face/Fingerprint) integration for unlocking the app database.
- **2FA:** Optional Time-based One-Time Password (TOTP) support.

8. Deployment Strategy

8.1 Free Tier Deployment Options

Platform	Free Tier Limits	Usage
Render.com	Free Web Services (ramps down on inactivity)	Hosting the Node.js/Go Relay Server.
Fly.io	3 shared-cpu-1x VMs, 3GB persistent volume	Global edge deployment for low-latency messaging.
Railway.app	$5.00 trial credit / low usage tier	Backend API and PostgreSQL database (if needed).
Azure Free	12 months popular services + always free limits	SignalR Service (Free tier) and Blob Storage.

8.2 Distribution Methods

- **GitHub Releases:** Main distribution channel for binaries.
- **Direct Download:** Hosted on the project website.
- **Auto-Update:** Built-in mechanism (e.g., Squirrel.Windows or distinct updater) to poll GitHub releases.

8.3 Installation Package

- **Standard Installer:** MSIX package signed with a self-signed certificate (for test/dev) or trusted CA.
- **Portable Version:** A self-contained `.zip` requiring no installation, suitable for USB drive usage.
- **Prerequisites:** The installer will bootstrap the .NET 8 Desktop Runtime if not present.

9. Build Configuration

9.1 NuGet Dependencies

- `Microsoft.EntityFrameworkCore.Sqlite` (8.0.0)
- `modern-wpf` (UI Styling)
- `libsignal-protocol-dotnet` (Crypto)
- `Newtonsoft.Json` (Serialization)
- `Serilog` (Secure Logging)
- `RestSharp` (API Client)

9.2 Build Profiles

- **Debug:** Includes symbols, verbose logging, no optimization.
- **Release:** Fully optimized, sensitive strings obfuscated, debug info stripped.
- **Publish:** Single-file executable, ready-to-run (R2R) compilation enabled.

9.3 Code Signing

Release builds must be signed to prevent Windows SmartScreen warnings. For the free tier strategy, a self-signed certificate will be used initially, with instructions for users to trust the root CA, or a low-cost open-source signing certificate will be procured.

10. Testing Strategy

Test Type	Scope	Tools
Unit Tests	ViewModels, Domain Logic, Crypto wrappers	xUnit, Moq, FluentAssertions
Integration Tests	Database persistence, API Client	xUnit, TestContainers, SqLite InMemory
UI Tests	Login flow, Chat rendering, Navigation	Appium (WinAppDriver), FlaUI
Security Tests	Penetration testing, Fuzzing crypto inputs	OWASP ZAP, Custom fuzzing scripts
Performance	Load testing message encryption/decryption	BenchmarkDotNet

11. Development Timeline

Phase	Focus	Duration
Phase 1	Project Setup, Clean Architecture scaffolding	1 Week
Phase 2	Core Cryptography Integration (Signal + PQC)	2 Weeks
Phase 3	Messaging Features (Chat, WebSocket, DB)	3 Weeks
Phase 4	UI/UX Development (WPF, Themes)	2 Weeks
Phase 5	Premium Features (Media, Groups, Backup)	2 Weeks
Phase 6	Security Hardening, Auditing & Testing	2 Weeks
Phase 7	Deployment, Packaging, Documentation	1 Week
Total	**Production Ready V1.0**	**13 Weeks**

12. Resource Requirements

12.1 Development Team

- 1 Lead .NET Architect

- 2 Senior C#/WPF Developers
- 1 Cryptography Engineer
- 1 UI/UX Designer
- 1 DevOps/Security Engineer
- 1 QA Engineer

12.2 Infrastructure

- Workstations: Windows 11 Pro, 16GB+ RAM, Visual Studio 2022.
- Test Lab: VMs with Windows 10 (1809+), Windows 11, and varied screen DPIs.
- CI/CD: GitHub Actions (Free tier for public repos) for automated build and test.

13. Security Checklist

✅ Cryptographic operations performed in native/safe code contexts.
✅ No plaintext keys resident in paged memory (use `SecureString`/Pinning).
✅ TLS 1.3 enforced with strict certificate pinning.
✅ Code obfuscation (Dotfuscator) enabled for Release builds.
✅ Anti-tamper checks (signature verification) at runtime.
✅ Database encryption (SQLCipher) verified active.
✅ Application Auto-lock on inactivity implemented.
✅ Screen capture protection (Window `SetDisplayAffinity`) enabled.
✅ Logging sanitization ensures no PII or message content is logged.
✅ Dependency vulnerability scanning active in CI pipeline.

14. API Integration

14.1 Backend Communication

- **Protocol:** WebSocket (wss://) for real-time streams; HTTPS for RESTful operations (registration, profile).
- **Payload:** Binary Protocol Buffers (Protobuf) for efficiency and type safety.
- **Fallback:** Long-polling mechanisms if WebSocket is blocked by corporate firewalls.

14.2 Third-Party Services

- **Mailgun:** Integrated for transactional emails via REST API.
- **VPN (Optional):** Hooks for WireGuard tunnel integration to mask IP traffic.
- **Cloud Backup:** Provider-agnostic file upload interfaces (Google Drive, Dropbox, OneDrive SDKs).

15. Compliance & Licensing

15.1 Open Source Compliance

The project is released under the MIT License. A "Third-Party Notices" file is generated during the build process to attribute all used libraries (libsignal, SQLite, etc.) in compliance with their respective licenses.

15.2 Security Standards

While designed for general use, the application adheres to principles from FIPS 140-2 regarding cryptographic module implementation. GDPR compliance is achieved via the zero-knowledge architecture—no user data is accessible to the service provider to be processed or mined.

16. Deployment Package Contents

16.1 Deliverables

1. **VaultWindows-Setup.exe:** Primary installation executable.
2. **VaultWindows-Portable.zip:** Portable application folder.
3. **Source Code:** Complete repository zip.
4. **Docs:** PDF User Guide and API references.
5. **Config:** `appsettings.json` templates for custom relay servers.

16.2 System Requirements

- **OS:** Windows 10 version 1809 or higher, Windows 11.
- **Runtime:** .NET 8 Desktop Runtime (bundled in installer).
- **Hardware:** 4GB RAM minimum, 500MB disk space.
- **Network:** Broadband internet connection.

17. Post-Deployment

- **Monitoring:** Opt-in, privacy-preserving crash reporting via Sentry (self-hosted or free tier).
- **Updates:** Delta updates delivered via the application's internal updater to minimize bandwidth.
- **Support:** Community-driven support via GitHub Issues and Discussions. Email support for critical security disclosures.

18. Future Roadmap

- **Cross-Platform:** Expansion to macOS and Linux using MAUI or Avalonia.
- **Mobile Parity:** Full feature synchronization with Android/iOS clients.
- **Enterprise:** Federated server support for private corporate deployments.
- **Crypto Agility:** Modular cryptographic layer to easily swap algorithms as PQC standards evolve.