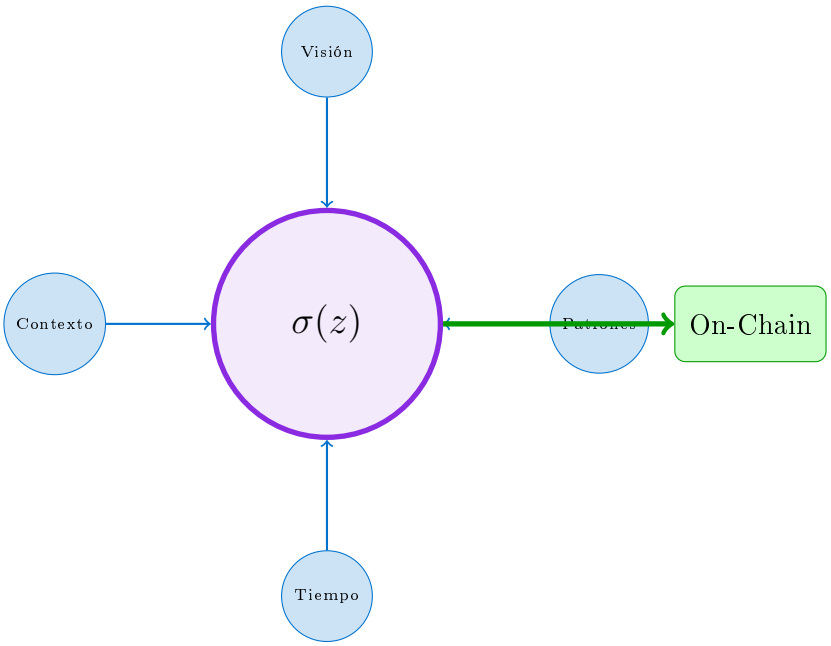
PULSE PROTOCOL

Sistema de Herencia Criptográfica Descentralizada

con Verificación Biométrica Pasiva mediante Inteligencia Artificial



Prueba de Vida Continua

Whitepaper Técnico v1.0

Febrero 2026

Construido sobre

Image

Stellar / Soroban

Contents

1 Resumen Ejecutivo **2**
1.1 El Problema . . . 2
1.2 La Solución: Pulse Protocol . . . 2

2 Arquitectura del Sistema **3**
2.1 Visión General . . . 3
2.2 Capa 1: Cliente Móvil . . . 3
2.2.1 Módulo de Visión Artificial . . . 3
2.2.2 Módulo de Análisis de Patrones . . . 4

3 Modelo de Inteligencia Artificial: El Perceptrón **4**
3.1 Fundamento Teórico . . . 4
3.2 Arquitectura del Perceptrón . . . 5
3.3 Formulación Matemática . . . 5
3.4 Vector de Features . . . 6
3.5 Entrenamiento del Modelo . . . 6
3.5.1 Fase de Calibración . . . 6
3.5.2 Función de Pérdida . . . 6
3.5.3 Actualización Continua (Online Learning) . . . 7

4 Almacenamiento On-Chain de Pesos **7**
4.1 Motivación . . . 7
4.2 Estructura de Datos en Soroban . . . 7
4.3 Representación de Punto Fijo . . . 8

5 Flujo de Herencia **8**
5.1 Estados del Sistema . . . 8
5.2 Descripción de Estados . . . 8
5.3 Período de Gracia y Notificaciones . . . 9

6 Smart Contracts en Soroban **9**
6.1 Contratos Principales . . . 9
6.2 Flujo de Interacción entre Contratos . . . 10

7 Privacidad y Seguridad **11**
7.1 Principios de Privacidad . . . 11
7.2 Modelo de Amenazas . . . 11
7.3 Consideraciones de Anti-Spoofing . . . 11

8 Stack Tecnológico **12**
8.1 Resumen del Stack . . . 12
8.2 Diagrama de Despliegue . . . 12

9 Roadmap de Desarrollo **13**
9.1 Fases del Proyecto . . . 13

10 Conclusión **13**

A Glosario **14**

B Referencias **14**

1 Resumen Ejecutivo

Visión

Pulse Protocol es un sistema de herencia criptográfica descentralizada que utiliza inteligencia artificial para verificar continuamente la "prueba de vida" del usuario mediante visión artificial y análisis de patrones de comportamiento, eliminando la necesidad de intermediarios legales y resolviendo el problema crítico de la pérdida permanente de activos digitales.

1.1 El Problema

La adopción masiva de criptomonedas ha creado un problema sin precedentes en la historia financiera: la pérdida irrecuperable de activos por fallecimiento del titular. Se estima que entre 3 y 4 millones de Bitcoin (aproximadamente el 20% del suministro total) están permanentemente perdidos, una proporción significativa debido a la muerte de sus propietarios sin transferir las claves privadas.

Los mecanismos tradicionales de herencia presentan limitaciones fundamentales:

1. **Dependencia de intermediarios**: Abogados, notarios y sistemas judiciales que no comprenden ni pueden acceder a activos criptográficos.
2. **Jurisdicción limitada**: Los testamentos tradicionales están sujetos a leyes locales, mientras que las criptomonedas son inherentemente globales.
3. **Rigidez**: Los testamentos son documentos estáticos que no pueden adaptarse a condiciones cambiantes.
4. **Falta de privacidad**: En muchas jurisdicciones, los testamentos se convierten en documentos públicos.

1.2 La Solución: Pulse Protocol

Pulse Protocol introduce un paradigma completamente nuevo: la **herencia programable con verificación biométrica pasiva**. El sistema utiliza un perceptrón entrenado individualmente para cada usuario, alimentado con señales de visión artificial y patrones de uso del dispositivo móvil.

Características diferenciadoras:

- **No intrusivo**: El usuario no realiza acciones especiales; el sistema observa patrones normales de vida.
- **Personalizado**: Cada usuario tiene un modelo de ML único, entrenado con sus propios patrones.
- **Gradual**: Detecta degradación (enfermedad prolongada) antes de eventos súbitos.
- **Resistente a fraude**: Combina múltiples señales biométricas y comportamentales.
- **Descentralizado**: La lógica de herencia vive en contratos inteligentes en Soroban (Stellar).

2 Arquitectura del Sistema

2.1 Visión General

Pulse Protocol se estructura en cuatro capas principales que interactúan para proporcionar un sistema robusto de herencia criptográfica:

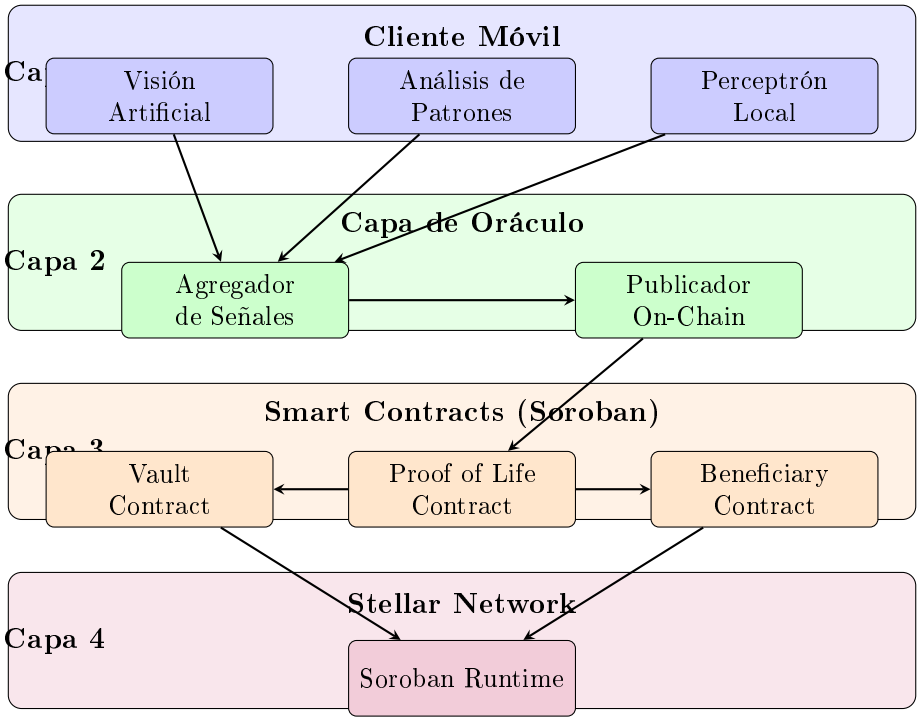


Figure 1: Arquitectura de cuatro capas de Pulse Protocol

2.2 Capa 1: Cliente Móvil

El cliente móvil es el núcleo de la recolección de datos y el procesamiento local de inteligencia artificial. Todos los datos biométricos sensibles se procesan exclusivamente en el dispositivo del usuario.

2.2.1 Módulo de Visión Artificial

El módulo de visión artificial implementa verificación facial continua con detección de "liveness" para prevenir ataques de suplantación.

```
class VisionModule:
    def __init__(self, user_face_encoding):
        self.reference_encoding = user_face_encoding
        self.liveness_detector = LivenessDetector()

    def verify(self, frame) -> VerificationResult:
        # Detectar rostro en el frame
        face = self.detect_face(frame)
        if not face:
            return VerificationResult(detected=False)

        # Verificar liveness (anti-spoofing)
        liveness_score = self.liveness_detector.check(
            blink_detection=True,
```

```
            micro_movements=True,
            depth_analysis=True  # Si disponible
        )

        # Comparar con encoding de referencia
        match_score = self.compare_encodings(
            face.encoding,
            self.reference_encoding
        )

        return VerificationResult(
            detected=True,
            match_score=match_score,       # 0.0 - 1.0
            liveness_score=liveness_score  # 0.0 - 1.0
        )
```

Listing 1: Pseudocódigo del módulo de visión

Señales capturadas:

- **Face Match Score**: Similitud con el rostro de referencia (0.0–1.0)
- **Liveness Score**: Probabilidad de que sea una persona real vs. foto/video
- **Contexto Visual**: Iluminación, ubicación aparente, ángulo típico

2.2.2 Módulo de Análisis de Patrones

Este módulo captura el "fingerprint comportamental" único del usuario sin acceder a contenido personal.

Table 1: Señales de patrones de comportamiento

Categoría	Señal	Descripción
Temporal	wake_time	Hora típica del primer uso
	sleep_time	Hora típica del último uso
	active_hours	Distribución de uso durante el día
Interacción	typing_speed	Velocidad promedio de escritura
	scroll_pattern	Patrón y velocidad de scroll
	touch_pressure	Presión típica en pantalla
	gesture_style	Estilo de gestos (swipe, tap, etc.)
Uso	app_frequency	Frecuencia de uso por aplicación
	session_duration	Duración típica de sesiones
	notification_response	Tiempo de respuesta a notificaciones
Movimiento	device_orientation	Cómo sostiene el dispositivo
	movement_signature	Patrón de acelerómetro al caminar

3 Modelo de Inteligencia Artificial: El Perceptrón

3.1 Fundamento Teórico

El corazón de Pulse Protocol es un perceptrón personalizado para cada usuario. Elegimos un perceptrón por las siguientes razones:

1. **Interpretabilidad**: Los pesos tienen significado directo (importancia de cada señal).
2. **Eficiencia computacional**: Puede ejecutarse en dispositivos móviles con batería limitada.
3. **Tamaño reducido**: Los pesos pueden almacenarse on-chain de forma económica.
4. **Robustez**: Menos propenso a overfitting que modelos más complejos.

3.2 Arquitectura del Perceptrón

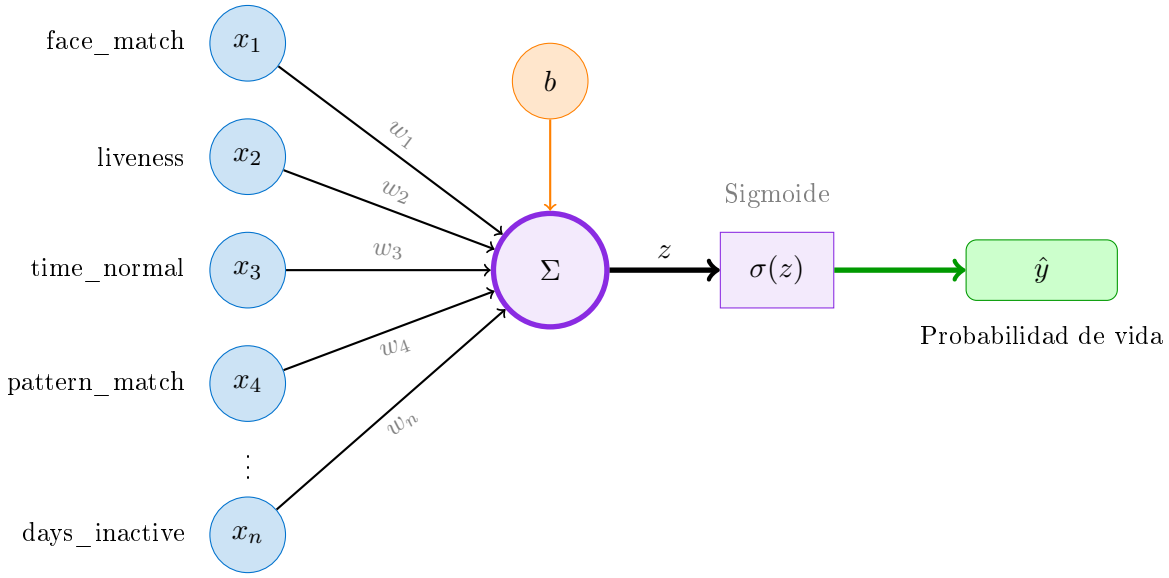


Figure 2: Arquitectura del perceptrón de Pulse Protocol

3.3 Formulación Matemática

El perceptrón calcula la probabilidad de vida mediante:

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b = \mathbf{w}^T \mathbf{x} + b \tag{1}$$

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

Donde:

- $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ es el vector de features normalizadas
- $\mathbf{w} = [w_1, w_2, ..., w_n]^T$ es el vector de pesos
- b es el término de sesgo (bias)
- σ es la función sigmoide
- $\hat{y} \in (0, 1)$ es la probabilidad de que el usuario esté vivo

3.4 Vector de Features

Definimos el vector de entrada $\mathbf{x}$ con las siguientes características normalizadas:

$$\mathbf{x} = \begin{bmatrix} x_1 & \text{face\_match\_score} & \in [0,1] \\ x_2 & \text{liveness\_score} & \in [0,1] \\ x_3 & \text{time\_of\_day\_normality} & \in [0,1] \\ x_4 & \text{location\_normality} & \in [0,1] \\ x_5 & \text{typing\_pattern\_match} & \in [0,1] \\ x_6 & \text{app\_usage\_match} & \in [0,1] \\ x_7 & \text{movement\_pattern\_match} & \in [0,1] \\ x_8 & \text{days\_since\_last\_verify} & \in [0,1] \\ x_9 & \text{session\_duration\_normal} & \in [0,1] \\ x_{10} & \text{interaction\_velocity\_normal} & \in [0,1] \end{bmatrix} \tag{3}$$

3.5 Entrenamiento del Modelo

3.5.1 Fase de Calibración

Durante las primeras 2-4 semanas, el sistema recolecta datos del usuario para establecer su "baseline" comportamental.

Algorithm 1 Calibración Inicial del Perceptrón

Require: Datos de comportamiento del usuario durante período de calibración
Ensure: Pesos iniciales $\mathbf{w}$, bias b

$\mathcal{D} \leftarrow \emptyset$ ▷ Dataset de entrenamiento
for cada día en período de calibración **do**
  for cada interacción del usuario **do**
    Extraer features $\mathbf{x}_t$
    $y_t \leftarrow 1$ ▷ Etiqueta: usuario vivo
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}_t, y_t)\}$
  end for
end for
Inicializar $\mathbf{w} \leftarrow \mathbf{0}$, $b \leftarrow 0$
Entrenar usando descenso de gradiente:
for cada epoch **do**
  for cada $(\mathbf{x}, y) \in \mathcal{D}$ **do**
    $\hat{y} \leftarrow \sigma(\mathbf{w}^T\mathbf{x} + b)$
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \cdot (\hat{y} - y) \cdot \mathbf{x}$
    $b \leftarrow b - \eta \cdot (\hat{y} - y)$
  end for
end for
return $\mathbf{w}$, b

3.5.2 Función de Pérdida

Utilizamos Binary Cross-Entropy para el entrenamiento:

$$\mathcal{L}(\mathbf{w}, b) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{4}$$

3.5.3 Actualización Continua (Online Learning)

El modelo se adapta gradualmente para aceptar cambios legítimos en el comportamiento del usuario:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \cdot \lambda \cdot (\hat{y}_t - y_t) \cdot \mathbf{x}_t \tag{5}$$

Donde $\lambda \in (0, 1)$ es un factor de "olvido" que permite adaptación lenta a cambios genuinos mientras resiste cambios abruptos sospechosos.

4 Almacenamiento On-Chain de Pesos

4.1 Motivación

Almacenar los pesos del perceptrón en la blockchain de Stellar proporciona:

- **Inmutabilidad auditada**: Historial completo de cambios
- **Propiedad del usuario**: El modelo le pertenece al usuario
- **Portabilidad**: Puede usarse desde cualquier dispositivo
- **Transparencia**: Los beneficiarios pueden auditar el sistema

4.2 Estructura de Datos en Soroban

```
#[contracttype]
#[derive(Clone)]
pub struct LifeModel {
    // Pesos del perceptron (fixed-point, 6 decimales)
    // Almacenados como i128 para precision
    pub weights: Vec<i128>,
    pub bias: i128,

    // Metadata del modelo
    pub version: u32,
    pub last_updated: u64,
    pub calibration_complete: bool,

    // Metricas de confianza
    pub total_verifications: u64,
    pub avg_confidence: u32,  // 0-10000 (2 decimales)

    // Thresholds personalizados
    pub alert_threshold: u32,     // Default: 3000 (0.30)
    pub critical_threshold: u32,  // Default: 1500 (0.15)
    pub grace_period_days: u32,   // Default: 30
}

#[contracttype]
#[derive(Clone)]
pub struct VerificationRecord {
    pub timestamp: u64,
    pub liveness_score: u32,  // 0-10000
    pub source: VerificationSource,
    pub oracle_signature: BytesN<64>,
}
```

```
#[contracttype]
#[derive(Clone)]
pub enum VerificationSource {
    FacialRecognition,
    BehaviorPattern,
    ManualCheckin,
    WitnessAttestation,
}
```

Listing 2: Estructura de datos del modelo en Soroban

4.3 Representación de Punto Fijo

Para almacenar pesos con precisión decimal en Soroban (que no soporta floats nativos), usamos representación de punto fijo:

$$w_{stored} = \lfloor w_{real} \times 10^6 \rfloor \tag{6}$$

Ejemplo: Un peso de 0.847523 se almacena como 847523 (i128).

5 Flujo de Herencia

5.1 Estados del Sistema

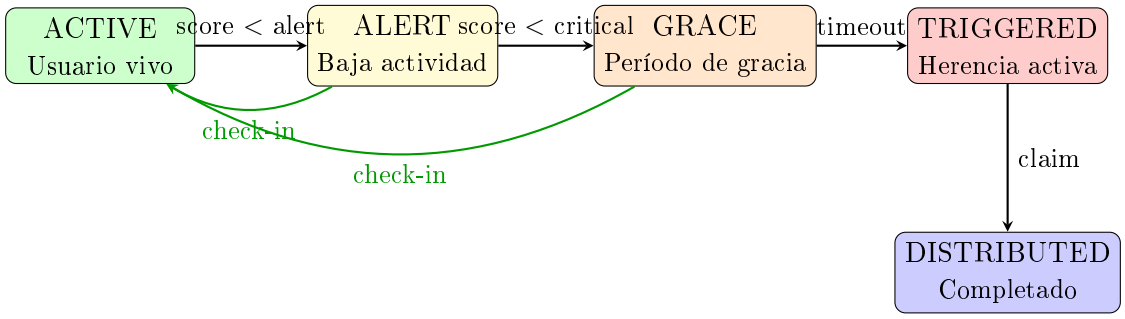


Figure 3: Máquina de estados del proceso de herencia

5.2 Descripción de Estados

1. **ACTIVE**: Estado normal. El perceptrón reporta alta probabilidad de vida ($\hat{y} > 0.7$). El sistema monitorea pasivamente.

2. **ALERT**: El score de vida ha bajado ($0.3 < \hat{y} < 0.7$). Posibles causas: vacaciones, hospitalización, cambio de dispositivo. El sistema incrementa la frecuencia de verificación.

3. **GRACE**: Score crítico ($\hat{y} < 0.3$) o período prolongado en ALERT. Se inicia el período de gracia configurable (default: 30 días). Se notifica a contactos de emergencia. El usuario puede cancelar con verificación exitosa.

4. **TRIGGERED**: El período de gracia expiró sin verificación exitosa. La herencia se activa. Los beneficiarios pueden comenzar a reclamar según las reglas programadas.

5. **DISTRIBUTED**: Todos los activos han sido reclamados. Estado terminal.

5.3 Período de Gracia y Notificaciones

Consideración Importante

El período de gracia es crítico para prevenir activaciones falsas. El sistema implementa múltiples capas de verificación antes de liberar activos.

Durante el período de gracia:

1. Se envían notificaciones diarias al usuario por todos los canales configurados
2. Se contacta a los "testigos de emergencia" designados
3. Se permite verificación por métodos alternativos (llamada telefónica, video con testigo)
4. El usuario puede extender el período de gracia una vez

6 Smart Contracts en Soroban

6.1 Contratos Principales

El protocolo se implementa en tres contratos principales que interactúan entre sí:

```
pub trait VaultTrait {
    // Crear nuevo vault
    fn create_vault(
        env: Env,
        owner: Address,
        initial_deposit: i128,
        token: Address
    ) -> VaultId;

    // Depositar activos
    fn deposit(
        env: Env,
        vault_id: VaultId,
        amount: i128,
        token: Address
    );

    // Retirar (solo si ACTIVE)
    fn withdraw(
        env: Env,
        vault_id: VaultId,
        amount: i128,
        token: Address
    );

    // Configurar beneficiarios
    fn set_beneficiaries(
        env: Env,
        vault_id: VaultId,
        beneficiaries: Vec<Beneficiary>
    );

    // Obtener estado actual
    fn get_status(env: Env, vault_id: VaultId) -> VaultStatus;
}
```

Listing 3: Interfaz del Vault Contract

```
pub trait ProofOfLifeTrait {
    // Registrar modelo de usuario
    fn register_model(
        env: Env,
        user: Address,
        initial_weights: Vec<i128>,
        bias: i128
    );

    // Actualizar verificacion (llamado por oraculo)
    fn submit_verification(
        env: Env,
        user: Address,
        score: u32,
        source: VerificationSource,
        oracle_sig: BytesN<64>
    );

    // Actualizar pesos del modelo
    fn update_model(
        env: Env,
        user: Address,
        new_weights: Vec<i128>,
        new_bias: i128
    );

    // Obtener score actual
    fn get_liveness_score(env: Env, user: Address) -> u32;

    // Check-in manual de emergencia
    fn emergency_checkin(env: Env, user: Address);
}
```

Listing 4: Interfaz del Proof of Life Contract

6.2 Flujo de Interacción entre Contratos

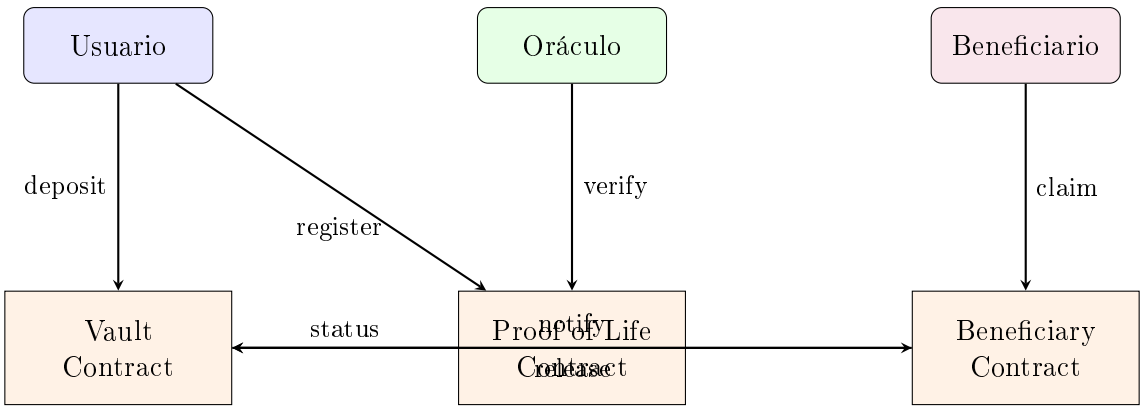


Figure 4: Interacción entre contratos del protocolo

7 Privacidad y Seguridad

7.1 Principios de Privacidad

Pulse Protocol sigue el principio de **"Privacy by Design"**:

Datos que NUNCA salen del dispositivo

- Imágenes faciales o datos biométricos raw
- Ubicación GPS exacta
- Contenido de mensajes o aplicaciones
- Historial de navegación
- Cualquier dato que identifique contactos

Datos que SÍ se procesan/envían

- Scores normalizados (0-1) sin contexto
- Pesos del modelo (no reversibles)
- Timestamps de verificación
- Hashes de patrones (no los patrones)

7.2 Modelo de Amenazas

Table 2: Análisis de amenazas y mitigaciones

Amenaza	Descripción	Mitigación
Suplantación facial	Atacante usa foto/video del usuario	Detección de liveness multi-señal, análisis de profundidad
Robo de dispositivo	Atacante obtiene acceso al teléfono	Patrones de comportamiento detectan anomalías, biometría del dispositivo
Colusión de testigos	Testigos declaran muerte falsa	Requiere múltiples fuentes independientes, período de gracia largo
Oráculo malicioso	Oráculo reporta datos falsos	Múltiples oráculos, verificación on-chain de firmas
Ataque al modelo	Envenenamiento gradual de pesos	Límites en tasa de cambio de pesos, auditoría de actualizaciones

7.3 Consideraciones de Anti-Spoofing

El módulo de liveness detection implementa múltiples técnicas:

1. **Detección de parpadeo**: Frecuencia y naturalidad del parpadeo

2. **Micro-movimientos**: Pequeños movimientos involuntarios del rostro
3. **Análisis de textura**: Detecta pantallas vs. piel real
4. **Challenge-response**: Solicitudes aleatorias ("mira a la izquierda")
5. **Análisis temporal**: Secuencias de frames para detectar videos pregrabados

8 Stack Tecnológico

8.1 Resumen del Stack

Table 3: Stack tecnológico completo

Capa	Tecnología	Justificación
Cliente Móvil		
Framework	React Native	Cross-platform, ecosistema maduro
ML Runtime	TensorFlow Lite	Inferencia eficiente on-device
Face Detection	ML Kit (Google)	Detección facial optimizada
Storage	Secure Enclave	Almacenamiento seguro de claves
Backend / Oráculo		
Lenguaje	Rust	Seguridad de memoria, rendimiento
API Framework	Actix-web	Async, alta concurrencia
Base de datos	PostgreSQL	Confiabilidad, SQL
Cache	Redis	Jobs, rate limiting
Stellar SDK	stellar-sdk (Rust)	Interacción con Soroban
Smart Contracts		
Plataforma	Soroban (Stellar)	Bajo costo, velocidad, Rust nativo
SDK	soroban-sdk	Desarrollo de contratos
Testing	soroban-test	Tests unitarios e integración
Infraestructura		
Hosting	Fly.io / Railway	Deploy global, bajo latencia
CI/CD	GitHub Actions	Automatización
Monitoreo	Grafana + Prometheus	Métricas y alertas

8.2 Diagrama de Despliegue

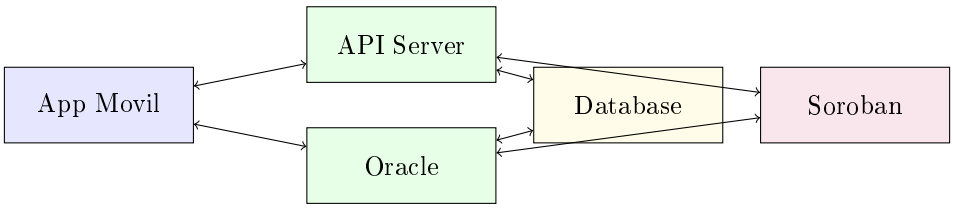


Figure 5: Diagrama de despliegue del sistema

9 Roadmap de Desarrollo

9.1 Fases del Proyecto

Table 4: Roadmap de desarrollo

Fase	Duración	Entregables
1: MVP	6-8 semanas	• Contratos básicos (Vault, Beneficiarios) • Check-in manual como prueba de vida • App móvil funcional mínima • Oráculo centralizado básico
2: AI Integration	6-8 semanas	• Módulo de visión artificial • Análisis de patrones de comportamiento • Perceptrón con entrenamiento local • Dashboard de métricas
3: Robustez	4-6 semanas	• Sistema de testigos • Período de gracia con notificaciones • Condiciones programables básicas • Auditoría de seguridad
4: Escalabilidad	Ongoing	• Descentralización del oráculo • Condiciones avanzadas • Integraciones (NFTs, multi-chain) • Gobernanza del protocolo

10 Conclusión

Pulse Protocol representa una innovación significativa en la intersección de DeFi, inteligencia artificial y planificación patrimonial digital. Al combinar:

- La seguridad y transparencia de la blockchain de Stellar
- La eficiencia de los smart contracts en Soroban
- La sofisticación de la visión artificial moderna
- La personalización de modelos de ML individuales

El protocolo ofrece una solución elegante a un problema que afecta a millones de holders de criptomonedas: ¿qué sucede con mis activos digitales cuando ya no esté?

La arquitectura propuesta prioriza la privacidad del usuario, la resistencia a fraudes y la flexibilidad para adaptarse a diferentes necesidades y jurisdicciones. El uso de un perceptrón personalizado como núcleo del sistema de prueba de vida permite una verificación continua y no intrusiva que se adapta a los patrones únicos de cada usuario.

"La verdadera herencia no es lo que dejamos a nuestros hijos,
sino lo que dejamos en ellos."

Pulse Protocol
Tu legado digital, protegido por inteligencia artificial.

A Glosario

Liveness Detection Técnica para verificar que un rostro pertenece a una persona real presente, no a una foto o video.

Perceptrón Modelo simple de red neuronal con una sola capa, utilizado para clasificación binaria.

Soroban Plataforma de smart contracts de Stellar, escrita en Rust.

Oráculo Servicio que proporciona datos externos a una blockchain.

Vault Contrato inteligente que custodia activos del usuario.

Prueba de Vida Verificación de que el propietario de un vault sigue vivo y activo.

B Referencias

1. Stellar Development Foundation. "Soroban Documentation." https://soroban.stellar.org/docs
2. TensorFlow. "TensorFlow Lite for Mobile." https://www.tensorflow.org/lite
3. Anthropic. "Constitutional AI: Harmlessness from AI Feedback." 2022.
4. Rosenblatt, F. "The Perceptron: A Probabilistic Model for Information Storage." Psychological Review, 1958.