UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA

Criptografía - *Grupo 2*

Profesor:
Dr. Alfonso Francisco de Abiega L'Eglisse

Proyecto:

Sistema de Votaciones

Integrantes:

Cruz Miranda Luis Eduardo

De la rosa Lara Gustavo

Domínguez Ríos Luis Daniel

Hernández Hernández Deissy Jovita

Mendoza Rodríguez Ángel Jesús

Nieto Rodríguez Tomás Andrés

Fecha:

20 de noviembre del 2025

Índice

Resumen

Este documento describe la implementación técnica de un sistema de votaciones seguro que utiliza técnicas criptográficas para garantizar la integridad, autenticidad y privacidad de los votos. El sistema implementa firmas digitales, cifrado asimétrico y tokens de un solo uso para crear un proceso de votación verificable y seguro.

1. Introducción

1.1. Propósito

El sistema de votaciones tiene como objetivo proporcionar una plataforma segura para la realización de procesos de votaciones digitales, implementando mecanismos criptográficos que garantizan:

- **Autenticidad**: Verificación de la identidad de votantes y creadores de encuestas
- **Integridad**: Protección contra modificaciones no autorizadas de los datos
- **Confidencialidad**: Secreto del voto mediante cifrado
- **No repudio**: Imposibilidad de negar la autoría de votos o encuestas

2. Arquitectura del Sistema

2.1. Componentes Principales

El sistema está compuesto por dos componentes principales:

2.1.1. Frontend (Cliente)

- Interfaz de usuario web responsive
- Gestión de claves criptográficas del usuario
- Operaciones de cifrado y firma en el cliente
- Comunicación con el backend via API REST

2.1.2. Backend (Servidor)

- Servidor Flask con endpoints API
- Gestión de usuarios y encuestas
- Verificación de firmas digitales
- Almacenamiento seguro de datos
- Proceso de escrutinio

3. Implementación Criptográfica

3.1. Generación y Gestión de Claves

3.1.1. Claves de Usuario

Cada usuario genera un par de claves RSA 2048 bits:

```
// Generaci n de par de claves en el cliente
const keyPair = await forge.pki.rsa.generateKeyPair({ bits: 2048 });
const publicKeyPem = forge.pki.publicKeyToPem(keyPair.publicKey);
const privateKeyPemEncrypted = forge.pki.encryptRsaPrivateKey(
    keyPair.privateKey,
    password
);
```

3.1.2. Claves de Encuestas

Cada encuesta genera su propio par de claves:

```
# Generaci n de claves para escrutinio en el servidor
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048,
    backend=default_backend()
)
public_key = private_key.public_key()
```

3.2. Firmas Digitales

3.2.1. Firma de Encuestas

Los administradores firman los datos de las encuestas:

```
// Firma en el cliente
const pollDataBytes = JSON.stringify(pollData,
   Object.keys(pollData).sort());
const md = forge.md.sha256.create();
md.update(pollDataBytes, 'utf8');
const signatureBytes = privateKey.sign(md);
const signatureBase64 = forge.util.encode64(signatureBytes);
```

3.2.2. Verificación en el Servidor

```
# Verificaci n en el servidor
creador_public_key.verify(
    firma_bytes,
    datos_encuesta_bytes,
    padding.PKCS1v15(),
    hashes.SHA256()
)
```

3.3. Cifrado de Votos

3.3.1. Cifrado en el Cliente

```
// Cifrado del voto con clave p blica de la encuesta
const publicKey = forge.pki.publicKeyFromPem(poll.clave_publica_pem);
const votoCifradoBytes = publicKey.encrypt(
    votoEnTextoPlano,
    'RSA-OAEP',
    { md: forge.md.sha256.create() }
);
```

3.3.2. Descifrado en el Escrutinio

```
# Descifrado durante el escrutinio
voto_descifrado_bytes = clave_privada.decrypt(
    voto_cifrado_bytes,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)
```

4. Flujos de Trabajo

4.1. Registro de Usuario

1. Usuario ingresa datos personales y contraseña
2. Sistema genera par de claves RSA 2048 bits
3. Clave privada se cifra con la contraseña del usuario
4. Clave pública se envía al servidor para registro
5. Primer usuario registrado se convierte en administrador

Figura 1: Diagrama de flujo del proceso de registro de usuario

4.2. Creación de Encuestas

1. Administrador autenticado crea nueva encuesta
2. Sistema genera par de claves para escrutinio
3. Datos de la encuesta se firman con clave privada del admin
4. Encuesta y clave pública de escrutinio se almacenan en servidor
5. Clave privada de escrutinio se guarda de forma segura

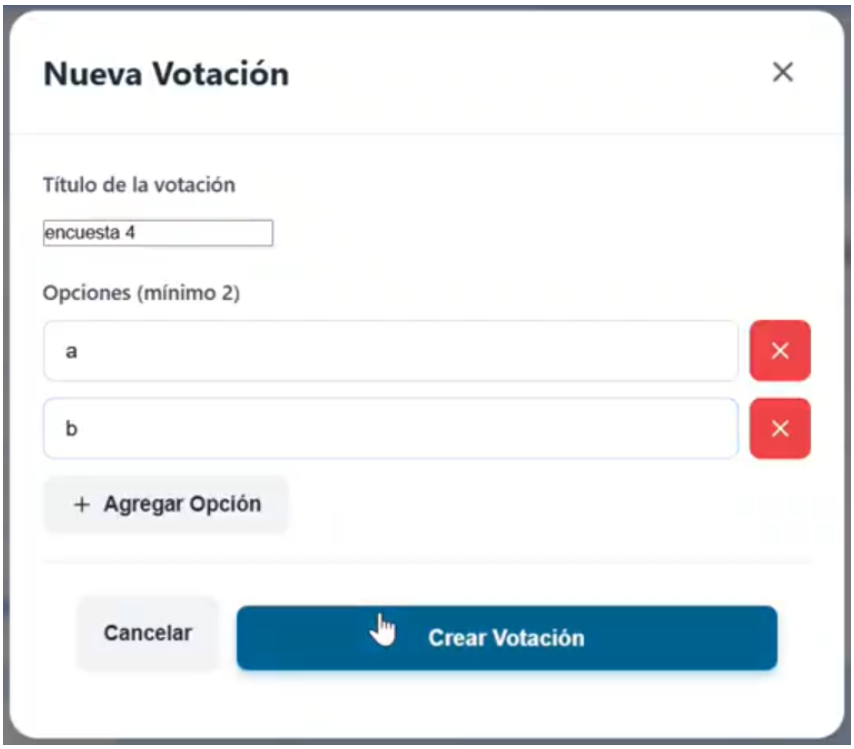

Figura 2: Diagrama de flujo del proceso de creación de encuestas

4.3. Proceso de Votación

4.3.1. Solicitud de Token

1. Usuario selecciona opción de voto
2. Sistema firma el ID de encuesta con clave privada del usuario
3. Se solicita token al servidor con la firma
4. Servidor verifica firma y genera token de un solo uso

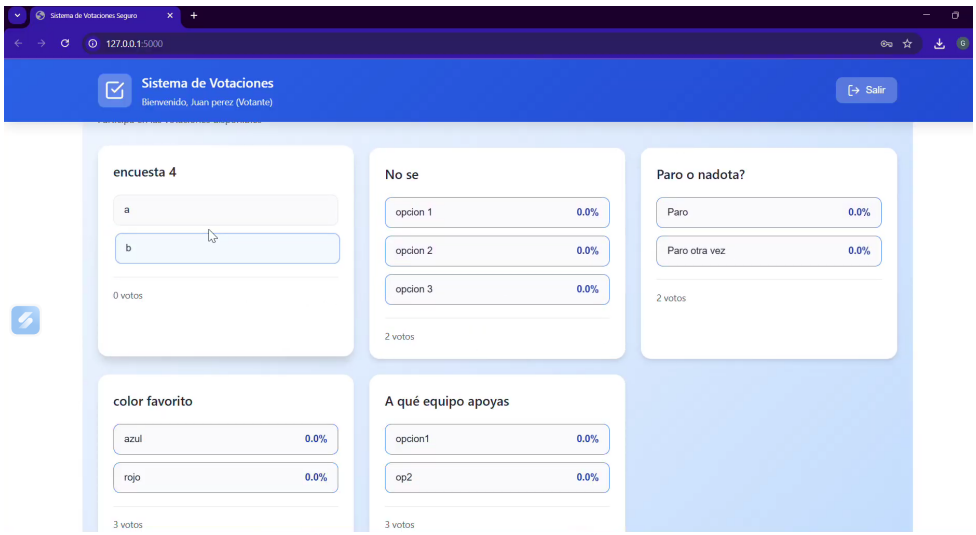

Figura 3: Diagrama de flujo del proceso de votación

4.3.2. Envío del Voto

1. Voto se cifra con clave pública de la encuesta
2. Voto cifrado se envía con el token
3. Servidor valida el token y almacena voto en urna digital
4. Token se invalida para prevenir uso múltiple

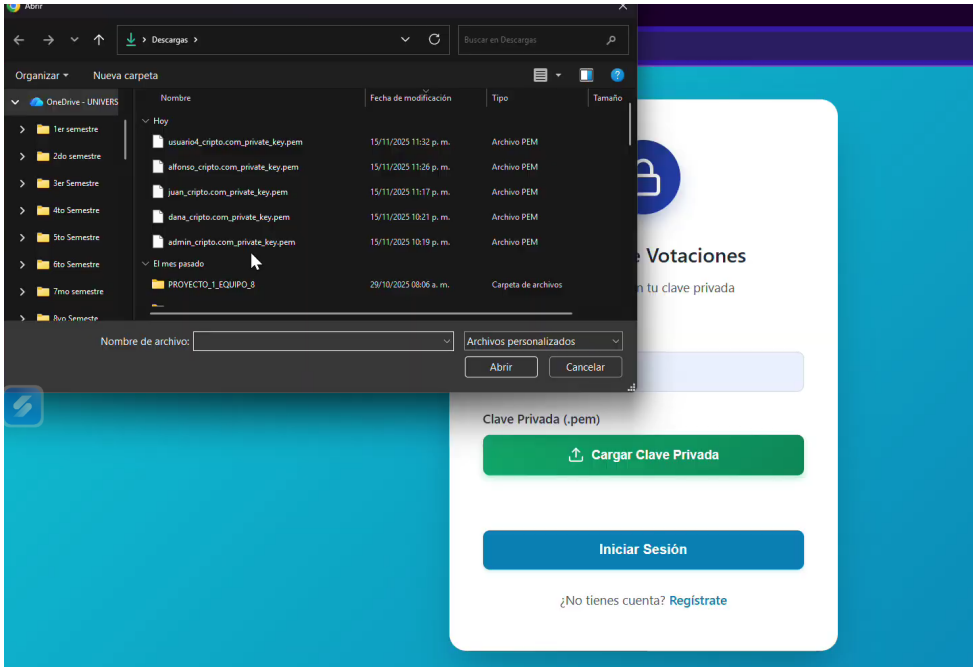

Figura 4: Diagrama de flujo del proceso de envío de voto

4.4. Escrutinio de Votos

1. Se accede al endpoint de escrutinio
2. Servidor recupera votos cifrados de la urna digital
3. Cada voto se descifra con la clave privada de escrutinio
4. Se cuentan los votos y se generan resultados
5. Resultados se presentan de forma anónima y agregada

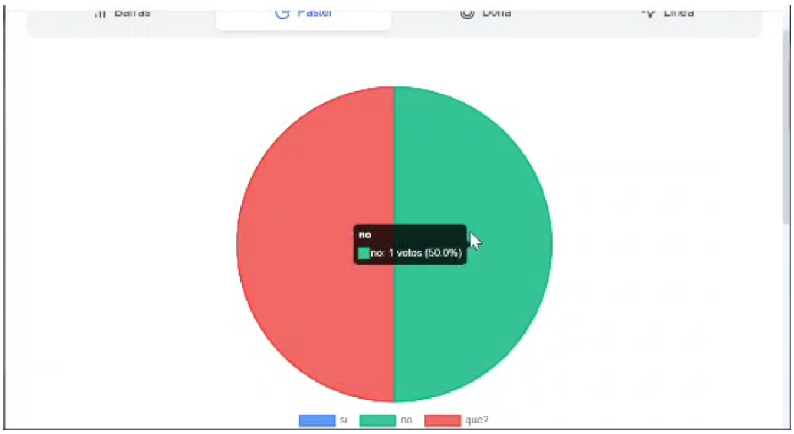

Figura 5: Gráfica de Patel de votos

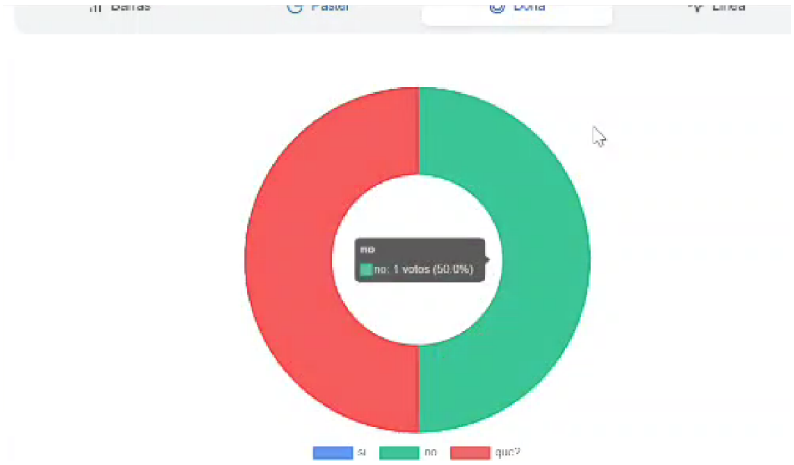

Figura 6: Gráfico Donut

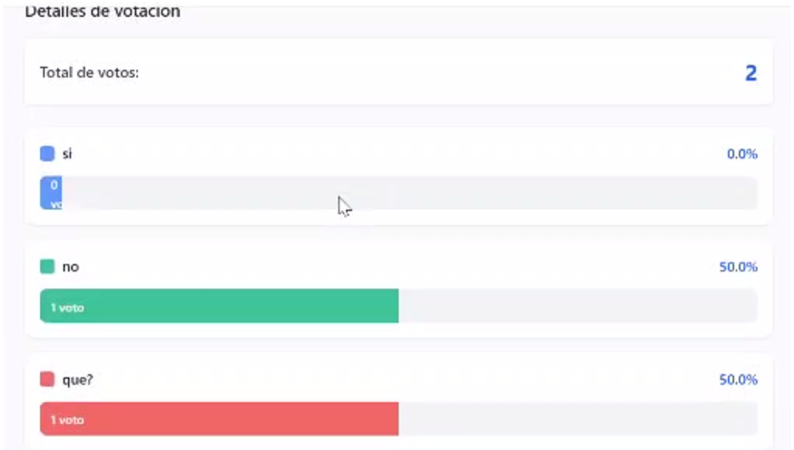

Figura 7: Gráfico de barras

5. Implementación Técnica

5.1. Frontend - Interfaz de Usuario

La interfaz está construida con HTML5, CSS3 y JavaScript vanilla, con un diseño moderno y responsive.

5.1.1. Estructura Principal

```
<div id="auth-form" class="auth-container">
  <!-- Formularios de login/registro -->
</div>

<div id="main-app" class="hidden">
  <header class="header">
    <!-- Barra de navegaci n -->
  </header>
  <main class="main-content">
    <!-- Lista de encuestas -->
    <div id="polls-container" class="polls-grid"></div>
  </main>
</div>
```

5.1.2. Gestión de Estado

```
// Estado global de la aplicaci n
let user = null;
let privateKey = null;
let polls = [];

// Inicializaci n de la aplicaci n
document.addEventListener('DOMContentLoaded', () => {
    initializeApp();
    setupEventListeners();
});
```

5.2. Backend - Servidor Flask

El servidor implementa la lógica de negocio y las operaciones criptográficas del lado del servidor.

5.2.1. Clase ServidorCentral

```
class ServidorCentral:
    def __init__(self):
        self.usuarios_registrados = {}
        self.encuestas_activas = []
        self.claves_privadas_encuestas = {}
        self.tokens_autorizados = {}
        self.votantes_autorizados = {}
        self.urna_digital = {}
```

5.2.2. Endpoints Principales

```python
@app.route('/registrar', methods=['POST'])
def endpoint_registrar():
    # Registro de usuarios

@app.route('/publicar-encuesta', methods=['POST'])
def endpoint_publicar():
    # Creaci n de encuestas

@app.route('/votar', methods=['POST'])
def endpoint_votar():
    # Recepci n de votos

@app.route('/contar-votos/<poll_id>', methods=['GET'])
def endpoint_contar_votos(poll_id):
    # Escrutinio de resultados
```

6. Consideraciones de Seguridad

6.1. Protecciones Implementadas

6.1.1. Autenticación y Autorización

- Verificación de firmas digitales para todas las operaciones críticas
- Tokens de un solo uso para prevenir votos duplicados
- Control de acceso basado en roles (admin/user)

6.1.2. Integridad de Datos

- Firmas SHA-256 con RSA para verificación de integridad
- Ordenamiento canónico de JSON antes de firmar
- Verificación de estructura de datos en el servidor

6.1.3. Confidencialidad

- Cifrado RSA-OAEP para votos
- Almacenamiento seguro de claves privadas
- Comunicación cliente-servidor via HTTPS (en producción)

6.2. Limitaciones y Mejoras Futuras

6.2.1. Para Entorno de Producción

- Implementar base de datos persistente
- Añadir logs de auditoría detallados
- Implementar rotación de claves
- Añadir protección contra ataques de tiempo
- Implementar backup seguro de claves

6.2.2. Mejoras Criptográficas

- Considerar uso de curvas elípticas para mejor rendimiento
- Implementar esquemas de umbral para el descifrado
- Añadir zero-knowledge proofs para verificación

7. Instalación y Configuración

7.1. Requisitos del Sistema

7.1.1. Software Requerido

- Python 3.8 o superior
- pip (gestor de paquetes de Python)
- Navegador web moderno (Chrome, Firefox, Safari, Edge)

7.1.2. Dependencias Python

```
Flask==2.3.3
cryptography==41.0.4
```

7.2. Instalación Paso a Paso

1. **Descargar el proyecto**
2. **Instalar dependencias:**

```
    pip install flask cryptography

```

3. **Ejecutar el servidor:**

```
    python servidor.py

```

4. **Acceder a la aplicación:**
   - Abrir navegador en `http://127.0.0.1:5000`
   - Registrar primer usuario (será administrador)
   - Crear encuestas y comenzar a votar

A. Códigos Fuente Relevantes

A.1. Frontend - Operaciones Criptográficas

```
// Ejemplo de firma digital en el cliente
async function signPollData(pollData, privateKey) {
    const pollDataBytes = JSON.stringify(pollData,
   Object.keys(pollData).sort());
    const md = forge.md.sha256.create();
    md.update(pollDataBytes, 'utf8');
```

```
    const signatureBytes = privateKey.sign(md);
    return forge.util.encode64(signatureBytes);
}
```

A.2. Backend - Verificación de Firmas

```
def verify_signature(public_key, data_bytes, signature_bytes):
    try:
        public_key.verify(
            signature_bytes,
            data_bytes,
            padding.PKCS1v15(),
            hashes.SHA256()
        )
        return True
    except InvalidSignature:
        return False
```

B. Conclusión

El sistema de votaciones criptográficas implementa técnicas criptográficas para crear un proceso de votación seguro, transparente y verificable. La arquitectura separa claramente las responsabilidades entre cliente y servidor, implementando las mejores prácticas de seguridad en cada componente.

Las características principales como las firmas digitales, el cifrado de votos y los tokens de un solo uso proporcionan un nivel robusto de seguridad adecuado para contextos educativos y puede servir como base para implementaciones más complejas en entornos productivos.