ADR-005: Token-Based Voter Authentication

Submitters

- Luke Doyle (D00255656)
- Hafsa Moin (D00256764)

Change Log

- approved 2026-02-11

Referenced Use Case(s)

- UVote Voter Authentication Requirements - Secure, low-friction authentication for one-time voters in small-scale elections.

Context

Voters in small-scale elections such as student councils, NGO boards, and community groups are one-time users who need to cast a single vote and may never use the system again. The authentication mechanism must balance security (only legitimate voters can vote, each voter votes once) with usability (minimum friction to complete the voting action). Forced account creation, password management, or multi-step verification would cause voter abandonment in the target user groups.

The project proposal identifies identity fraud as one of three critical barriers to election integrity, covering impersonation, synthetic identity fraud, and account takeover. Traditional password-based authentication is vulnerable to all three. For one-time voters who will never log in again, passwords add friction without proportional security benefit.

Prior experience with token-based authentication during a placement at IFM informed the approach here. Working with JWT tokens in a production environment gave direct familiarity with how cryptographic tokens can be used to authenticate users without requiring persistent credentials, and how single-use and expiry controls can be applied to limit their validity window. This experience supported the decision to adopt a token URL model for voter authentication rather than a more traditional account-based approach.

The selected approach is cryptographic token URLs: the admin imports voters via CSV, the system generates a unique 256-bit token per voter and sends it as a clickable email link, and the token is invalidated after use.

Proposed Design

Services and modules impacted:

Token generation is handled in `shared/security.py`. Token validation is handled by the auth-service. The voting-service marks tokens as used after a ballot is submitted. The audit-service logs token usage events.

Token generation (`shared/security.py`):

```
import secrets
from datetime import datetime, timedelta

def generate_voting_token(length: int = 32) -> str:
    """Generate a 256-bit URL-safe token."""
    return secrets.token_urlsafe(length)

def generate_token_expiry(hours: int = 168) -> datetime:
    """Default: 7 days (168 hours)."""
    return datetime.now() + timedelta(hours=hours)

```

Token validation (auth-service):

```
@app.get("/tokens/{token}/validate")
```

```
async def validate_voting_token(token: str):
    row = await conn.fetchrow(
        "SELECT ... FROM voting_tokens vt "
        "JOIN elections e ON e.id = vt.election_id "
        "WHERE vt.token = $1", token
    )
    # Check: exists, not used, not expired, election is open

```

Model and schema impact:

The `voting_tokens` table stores the token value, `is_used` boolean, `used_at` timestamp, expiry timestamp, and the associated election and voter. The `voter_mfa` table stores the date of birth used for second-factor verification after token validation.

Configuration:

- Token length: 32 bytes (256 bits of entropy, 43 URL-safe characters)
- Token expiry: 7 days (configurable via environment variable)
- Single-use: `is_used` boolean and `used_at` timestamp in `voting_tokens` table
- MFA: date of birth verification via `voter_mfa` table after token validation

Integration points:

- ADR-006 (JWT): Admin authentication uses JWT; token URLs are used for voters only
- ADR-015 (Anonymity): The voting token validates identity; the blind ballot token provides anonymity
- ADR-007 (Audit): Token usage is logged in the `audit_log` table with timestamp and election, without recording vote choice

Considerations

Password-based account creation (not selected): Voters register with email and password, then log in to vote. This is a well-understood model but requires a minimum of five steps (register, verify email, set password, log in, vote). Research on voter behaviour in online elections indicates that each additional step reduces participation. For a one-time action, requiring permanent credentials adds friction with no corresponding security benefit, given that credentials can be phished, shared, or reused across sites.

OAuth 2.0 via Google or Microsoft (not selected): Delegating authentication to an external identity provider reduces friction for users who already have accounts with those providers. It was not selected because it introduces an external dependency (an OAuth provider outage would block voting), raises privacy concerns around third-party tracking of voting behaviour, and cannot restrict access to only the voters registered by the admin without additional validation logic. It also does not satisfy the requirement for a self-contained system.

OTP via SMS or email (not selected): Sending a one-time password to the voter's registered contact provides moderate friction and proves control of the registered channel. SMS was ruled out by the project's budget constraint (per-message cost). Email OTP was considered but requires the voter to switch between email and browser, introduces a manual entry step that can fail due to typos or expiry, and is functionally equivalent to a clickable link with added inconvenience. The insight that the OTP itself could simply be embedded in a URL led directly to the token URL approach.

Token forwarding risk: A voter could forward their email link to another person. This is mitigated by the date of birth verification step, which requires the recipient to know the registered voter's date of birth before proceeding. Single-use enforcement and audit logging provide additional controls.

Email dependency: Token delivery relies on the SMTP infrastructure. This is mitigated by retry logic in the email service, a 7-day validity window that accommodates delivery delays, and an admin interface that allows individual tokens to be resent.

No self-service recovery: A voter who does not receive their token must contact the admin for reissuance. This is accepted given that the admin interface provides a one-click resend function and the token status is visible per voter.

Decision

Cryptographic token URLs were selected as the voter authentication mechanism for UVote.

The primary driver is minimising friction for one-time voters. A single click from the email link to the ballot is the lowest possible barrier to participation, and removes the credential management overhead that is the main cause of drop-off in online voting systems. Research from established e-voting implementations, including Estonia's iVoting system, supports the use of invitation-based token models for this reason.

Prior experience with JWT token-based authentication during a placement at IFM provided practical grounding in how cryptographic tokens can authenticate users without persistent credentials and how expiry and single-use controls constrain their validity. This informed confidence in the approach and its implementation using Python's `secrets` module.

The security properties of the selected approach are strong. Python's `secrets.token_urlsafe(32)` generates 256-bit tokens using the operating system's cryptographically secure pseudorandom number generator (CSPRNG). Single-use enforcement means a captured token cannot be replayed. The 7-day expiry limits the window in which a token remains valid. Date of birth verification as a second factor addresses the impersonation risk without adding visible friction beyond a single form field.

The trade-off of removing self-service account recovery in exchange for minimum friction is accepted. The admin interface provides token reissuance, and the receipt token system (ADR-015) allows voters to verify their participation without requiring a persistent account.

A review is scheduled for the end of Stage 2 (April 2026) to assess whether date of birth verification is sufficient as a second factor or whether additional verification is needed.

Other Related ADRs

- ADR-006: Admin Authentication (JWT) - separate authentication mechanism for administrators
- ADR-007: Audit Logging - token usage events logged here
- ADR-015: Blind Ballot Token Anonymity - identity-ballot separation using a second token

References

- Python secrets module documentation
- Estonian iVoting System - token-based invitation model
- EdgeX Foundry ADR Template