SOC 2 Controls Matrix

Divy It Up, LLC

Document Creation Date:	March 18th, 2026
Effective Date:	March 19th, 2026
Version:	1.1
Classification:	Public
System:	**Divy It Up** – Cryptographic Key Management & Data Protection
Scope:	Security & Confidentiality Criteria

Document Purpose

Below we map the SOC 2 Controls Matrix mapped explicitly to the Trust Services Criteria (TSC) with clear linkage to the Divy It Up implemented architecture (AWS Key Management Service (KMS), envelope encryption, federated identity, least privilege). Our goal is to clearly show: Control → Description → Implementation → Evidence → Owner.

This is meant to complement and clarify our SOC 2 Control Narrative Document by clearly detailing the Control and the Ownership.

CC1 – Control Environment

CC1.2 – Integrity and Ethical Values

Control:

Management establishes and enforces security-first principles for handling financial data.

Implementation:

- Security architecture mandates encryption of all sensitive financial data at ingestion
- No plaintext storage permitted
- Separation of duties enforced between key admins and application roles

Evidence:

- Security architecture documentation
- Codebase enforcement (encryption wrapper usage)
- KMS key policy showing role separation

Owner: Engineering Leadership

CC2 – Communication and Information

CC2.1 – Internal Communication of Security Responsibilities

Control:

Security responsibilities for cryptographic systems are defined and communicated.

Implementation:

- Defined roles:
    - Key Administrator (AWS Identity & Access Management (IAM) user)
    - Backend Service Role (cryptographic operations)
- No developer-level access to production decryption capabilities

Evidence:

- IAM role definitions
- KMS key policy
- Internal documentation of role responsibilities

Owner: Engineering / Security

CC3 – Risk Assessment

CC3.2 – Identification of Risks

Control:

Risks related to financial data exposure and key compromise are identified and mitigated.

Implementation:

- Threat model includes:
    - Credential leakage
    - Insider misuse
    - Data exfiltration
- Mitigations:
    - No long-lived credentials
    - No plaintext persistence
    - KMS-managed keys

Evidence:

- Architecture documentation
- IAM configuration
- Absence of static credentials in environment configs

Owner: Security / Engineering

CC4 – Monitoring Activities

CC4.1 – Ongoing Monitoring

Control:

Cryptographic operations and access are continuously monitored.

Implementation:

- All KMS operations logged via AWS CloudTrail:
    - GenerateDataKey
    - Decrypt
    - Role assumption events

- Logs include identity and context

Evidence:

- CloudTrail logs
- Audit log samples showing KMS usage
- Alerting configuration (if implemented)

Owner: DevOps / Security

CC5 – Control Activities

CC5.2 – Logical Access Controls

Control:

Access to cryptographic functions is restricted to authorized principals.

Implementation:

- Single backend IAM role allowed:
  - kms:GenerateDataKey
  - kms:Decrypt
- No human user has decrypt permissions
- KMS key policy enforces explicit principal allow-list

Evidence:

- IAM role policy
- KMS key policy JSON
- AWS IAM console screenshots

Owner: DevOps

CC5.3 – Segregation of Duties

Control:

Administrative and operational responsibilities are separated.

Implementation:

- Key administrators:
  - Manage key lifecycle
  - Cannot decrypt data
- Application role:
  - Can decrypt
  - Cannot modify key policy

Evidence:

- IAM policies showing separation
- KMS key policy statements
- Role definitions

Owner: Security / DevOps

CC6 – Logical and Physical Access Controls

CC6.1 – Logical Access Security

Control:

System access is restricted based on least privilege.

Implementation:

- IAM role scoped to:
  - Specific KMS key ARN
  - Minimal required actions
- No wildcard permissions

Evidence:

- IAM policy JSON
- AWS IAM policy simulator results

Owner: DevOps

CC6.2 – Authentication Mechanisms

Control:

Strong authentication is enforced for system access.

Implementation:

- Federated identity via OIDC
- AWS STS issues short-lived credentials
- No static AWS credentials used

Evidence:

- IAM role trust policy (OIDC provider)
- Absence of AWS keys in environment variables
- STS logs

Owner: DevOps

CC6.6 – Least Privilege

Control:

Access rights are limited to necessary functions.

Implementation:

- Backend role limited to:
  - GenerateDataKey
  - Decrypt
- No broad AWS permissions
- No cross-service access

Evidence:

- IAM role policy
- KMS key policy
- Access review documentation

Owner: Security

CC6.7 – Credential Management

Control:

Credentials are securely managed and rotated.

Implementation:

- No long-lived credentials
- Short-lived STS tokens issued via OIDC
- Automatic expiration

Evidence:

- IAM role configuration
- OIDC trust policy
- AWS STS logs

Owner: DevOps

CC7 – System Operations

CC7.2 – Change Management

Control:

Changes to cryptographic configuration are controlled and auditable.

Implementation:

- KMS key policy changes require admin permissions
- IAM changes logged in AWS
- Infrastructure changes tracked

Evidence:

- CloudTrail logs for:
  - PutKeyPolicy
  - IAM changes
- Change management records

Owner: DevOps / Security

CC7.3 – Monitoring for Anomalies

Control:

System monitors for unusual or unauthorized activity.

Implementation:

- CloudTrail captures:

    - Unexpected decrypt activity
    - Unauthorized role assumptions
- Logs reviewed periodically or via alerts

Evidence:

- CloudTrail logs
- Alerting configuration (if applicable)

Owner: Security

CC8 – Change Management (System Components)

CC8.1 – Encryption of Sensitive Data

Control:

Sensitive data is encrypted using strong cryptographic techniques.

Implementation:

- AES-256-GCM encryption
- Envelope encryption using AWS KMS
- Per-record data keys

Evidence:

- Encryption code implementation
- KMS usage logs
- Database schema (no plaintext fields)

Owner: Engineering

CC8.2 – Data Confidentiality

Control:

Confidential data is protected from unauthorized disclosure.

Implementation:

- Only encrypted values stored
- Masked values used for UI
- No sensitive data returned to client

Evidence:

- API response samples
- Database inspection
- Code review

Owner: Engineering

CC8.3 – Key Management

Control:

Cryptographic keys are securely managed.

Implementation:

- AWS KMS CMK used
- Key material never exposed
- Access controlled via key policy + IAM

Evidence:

- KMS configuration
- Key policy JSON
- AWS documentation

Owner: DevOps / Security

CC9 – Risk Mitigation

CC9.2 – Data Retention and Minimization

Control:

Sensitive data is minimized and retained only as necessary.

Implementation:

- Only required financial fields stored
- No redundant duplication
- Preference for tokenization where possible

Evidence:

- Database schema
- Data retention policy

Owner: Engineering / Compliance

Summary for Auditors

This control environment demonstrates:

- Strong encryption controls (CC8) via AES-256-GCM + KMS
- Strict access control (CC6) via IAM roles and OIDC federation
- Separation of duties (CC5) between admins and application roles
- Comprehensive logging (CC4/CC7) via CloudTrail
- Credential security (CC6.7) via elimination of static secrets