Cryptographic Key Management & Data Protection Architecture

Divy It Up, LLC

Document Creation Date:	March 15th, 2026
Effective Date:	March 15th, 2026
Version:	1.2
Classification:	Public

Security Design Overview

1. Executive Summary

Divy It Up employs a **defense-in-depth cryptographic architecture** built on **AWS Key Management Service (KMS)** to protect sensitive financial data, including:

- Bank account numbers
- Routing numbers
- Third-party access tokens (e.g., Plaid access tokens)

All sensitive data is encrypted using **envelope encryption with AWS-managed key material**, ensuring that:

- No plaintext financial data is stored at rest
- Cryptographic keys are never exposed to application developers or infrastructure
- Decryption is tightly controlled, audited, and restricted to specific backend workflows

This architecture aligns with industry best practices for handling financial data and is consistent with principles required for secure ACH and banking integrations.

---

2. Design Principles

The system is designed around the following core security principles:

Least Privilege Access

Only a single, tightly scoped backend role is permitted to perform cryptographic operations. No human users or broad services have access to decrypt sensitive data.

Separation of Duties

- **Key administrators** manage key lifecycle and policy

- **Application roles** perform encryption/decryption
- No principal has both unrestricted administrative and cryptographic authority

No Long-Lived Credentials

All AWS access is performed using **short-lived, automatically rotated credentials** issued via secure identity federation.

Zero Trust for Application Layers

- Sensitive values are encrypted immediately upon receipt
- Plaintext values are never persisted
- Decryption occurs only in memory and only when required

---

3. Encryption Architecture

3.1 Envelope Encryption Model

Divy It Up uses **envelope encryption**, the AWS-recommended pattern for application-layer encryption.

Encryption Flow

1. Application requests a **data encryption key (DEK)** from AWS KMS
2. KMS returns:
    - A **plaintext DEK** (used transiently in memory)
    - An **encrypted DEK** (bound to the KMS master key)
3. Application encrypts sensitive data locally using AES-256-GCM
4. Application stores:
    - Ciphertext
    - Encrypted DEK
    - Initialization vector (IV) and authentication tag

Decryption Flow

1. Application retrieves encrypted data + encrypted DEK
2. Encrypted DEK is sent to AWS KMS for decryption

3. KMS returns the plaintext DEK (in-memory only)
4. Application decrypts data in memory and immediately discards key material

At no point is the master key or plaintext data stored persistently.

3.2 Cryptographic Standards

- Encryption Algorithm: **AES-256-GCM**
- Key Management: **AWS KMS Customer-Managed Keys**
- Key Generation: **AWS KMS GenerateDataKey API**
- Authenticated Encryption: Ensures both confidentiality and integrity

---

4. Key Management (AWS KMS)

4.1 Customer-Managed Keys

All encryption operations are backed by a **dedicated AWS KMS Customer-Managed Key (CMK)**:

- Strictly scoped to financial data use cases
- Isolated from other application keys
- Managed under explicit key policies

4.2 Key Policies

Access to the KMS key is governed by:

- **Explicit principal allow-listing**
- No wildcard or broad account-level usage
- Separate roles for:
    - Key administration
    - Cryptographic operations

4.3 Key Rotation

- Automatic key rotation is enabled where applicable
- Historical key material remains available for decryption of previously encrypted data

5. Identity & Access Management

5.1 Federated Identity Model

The application uses **OpenID Connect (OIDC) federation** with AWS Security Token Service (STS):

- No static AWS credentials are stored or distributed
- Application instances obtain **short-lived credentials at runtime**
- Credentials are valid only for minutes and automatically expire

5.2 Role-Based Access Control

A single backend IAM role is authorized to:

- kms:GenerateDataKey
- kms:Decrypt
- kms:DescribeKey

This role is:

- Restricted to specific environments (e.g., production only)
- Bound to a specific workload identity
- Not assumable by users or external systems

5.3 Elimination of Credential Risk

This model eliminates:

- Credential leakage risk
- Key rotation overhead
- Long-lived access tokens in infrastructure

---

6. Data Handling Controls

6.1 Encryption Scope

The following fields are encrypted at the application layer:

- Bank account number

- Routing number
- Financial access tokens

Non-sensitive metadata (e.g., masked account numbers) may be stored in plaintext for usability.

6.2 In-Memory Decryption Only

- Decryption occurs **only in volatile memory**
- Data is used immediately and never persisted post-decryption
- Memory buffers are cleared after use

6.3 Logging & Observability Controls

The system enforces strict controls to prevent data leakage:

- No sensitive values are logged
- No plaintext values appear in application logs, traces, or analytics
- Encryption context is used for auditability without exposing secrets

---

7. Encryption Context & Data Binding

Each encryption operation includes a structured **encryption context**, such as:

- Data purpose (e.g., plaid_access_token)
- Environment (e.g., production)
- Internal user identifier

This provides:

- Additional cryptographic binding of data to its intended use
- Protection against misuse or replay of encrypted values
- Enhanced audit traceability in AWS logs

---

8. Audit & Monitoring

All cryptographic operations are logged via **AWS CloudTrail**, including:

- Key usage (GenerateDataKey, Decrypt)
- Role assumption events
- Source identity and context

This provides:

- Full traceability of sensitive data access
- Detection of anomalous or unauthorized activity
- Compliance-ready audit logs

---

9. Security Posture Summary

This architecture provides the following guarantees:

Confidentiality

Sensitive financial data is never stored in plaintext and is protected by strong encryption at all times.

Integrity

Authenticated encryption ensures that any tampering with ciphertext is detected.

Access Control

Only explicitly authorized backend systems can decrypt data, under tightly scoped permissions.

Credential Security

No long-lived credentials exist; all access is ephemeral and federated.

Auditability

All key usage is logged, attributable, and reviewable.

---

10. Alignment with Financial Security Expectations

This design aligns with expectations for:

- ACH and bank-integrated systems
- Fintech-grade infrastructure

- SOC 2 and similar control frameworks
- Secure handling of account and routing numbers

Specifically, it demonstrates:

- Proper key management separation
- Use of managed HSM-backed key services (AWS KMS)
- Strong encryption standards
- Strict identity-based access controls

---

11. Conclusion

Divy It Up's cryptographic architecture ensures that:

- Sensitive financial data is **never exposed unnecessarily**
- Access is **strictly controlled and auditable**
- The system is built on **modern, cloud-native security primitives**

By combining AWS KMS, federated identity, and envelope encryption, the platform achieves a **high-assurance security posture appropriate for financial applications**.