



SAA-C03 Secure 1: IAM, Organizations, Encryption

SAA-C03 Domain 1 Design Secure Architectures (30 %), tasks 1.1 secure access and 1.3 data security | IAM users, roles, policies, federation, Organizations and SCPs, KMS, Secrets Manager, ACM, threat-detection services

SAA-C03 Domain 1 IAM KMS Organizations

Where this sheet sits in the exam

Domain 1 **Design Secure Architectures** is the heaviest SAA-C03 domain (30 % of scored questions). Task 1.1 asks you to design **secure access** (IAM, federation, multi-account), task 1.2 **secure workloads** (VPC security, sheet 02), task 1.3 **data security controls** (encryption, keys, backups, compliance). IAM evaluation logic in full depth is taught in the Security Specialty chapter; this sheet gives what an architect must decide.

	EC2 (IaaS)	RDS (managed)	Lambda (serverless)	S3 (storage)
Data classification, encryption choice	Customer	Customer	Customer	Customer
IAM: who may call which API	Customer	Customer	Customer	Customer
Network rules (SG, NACL, endpoints)	Customer	Customer	Shared	AWS
Guest OS patching	Customer	AWS	AWS	AWS
Database engine patching	Customer	Shared	AWS	AWS
Runtime and scaling	Customer	Shared	AWS	AWS
Hypervisor, hosts, data centres	AWS	AWS	AWS	AWS

Customer Shared AWS responsibility shifts

Shared responsibility moves to AWS as the service gets more managed; data and IAM always stay with you. RDS 'shared': AWS patches, you pick the maintenance window and major-version upgrades.

IAM building blocks

Identity	Credentials	Use it for	Avoid
Root user	email + password, MFA	only root-only tasks: close account, change support plan, restore IAM admin	access keys; daily use
IAM user	password and/or up to 2 access keys	break-glass, legacy tools that cannot use roles	humans in multi-account setups (use Identity Center)
IAM group	none (collection of users)	attach policies to many users	groups cannot be principals or nested
IAM role	temporary STS credentials	EC2 instance profile, Lambda execution role, cross-account, federation	sharing long-term keys instead

Identity Center user	SSO session → role per permission set owned by an AWS service	workforce access to many accounts	per-account IAM users
Service-linked role		Auto Scaling, ELB, RDS act on your behalf	editing its permissions (you cannot)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadReportsFromVpcOnly",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::reports",
        "arn:aws:s3:::reports/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceVpc": "vpc-0a1b2c"
        },
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    }
  ]
}
```

ListBucket needs the bucket ARN, GetObject the /* object ARN

only through the VPC endpoint and over TLS

Policy grammar: Effect, Action, Resource, Condition (Principal only in resource-based policies). Version 2012-10-17 enables policy variables like `${aws:username}`.

Request		
SAME ACCOUNT		
PRINCIPAL	role/DataEngineer	
ACTION	s3:PutObject	
RESOURCE	arn:aws:s3:::finance-b/reports/q3.csv	
CONTEXT	aws:RequestedRegion = eu-west-3; aws:SecureTransport = true	
LAYER	STATEMENTS (✓ MATCHES THE REQUEST)	RESULT
Organization guardrail (SCP)	✓ S1 Allow * on * -- S2 Deny * on * if StringNotEquals aws:RequestedRegion eu-west-3, eu-central-1 (condition StringNotEquals aws:RequestedRegion false)	allows
Workloads OU guardrail		
Permission boundary	-- S1 Allow s3:Get*, s3:List*, athena:* on * (action not matched)	no allow: implicit deny ← decides
DataBoundary		
Identity policy	✓ S1 Allow s3:* on arn:aws:s3:::finance-b/*	grants (allow)
DataEngineerPolicy		

DENY implicit deny: DataBoundary (Permission boundary) does not allow s3:PutObject; every guardrail must allow

Order: 1 explicit deny in any layer → deny. 2 every guardrail present (SCP, RCP, permission boundary, session policy) must allow, guardrails never grant. 3 an identity or resource policy must allow (both across accounts). 4 otherwise implicit deny.

* matches any run of characters, ? exactly one; actions ignore case, resources do not.

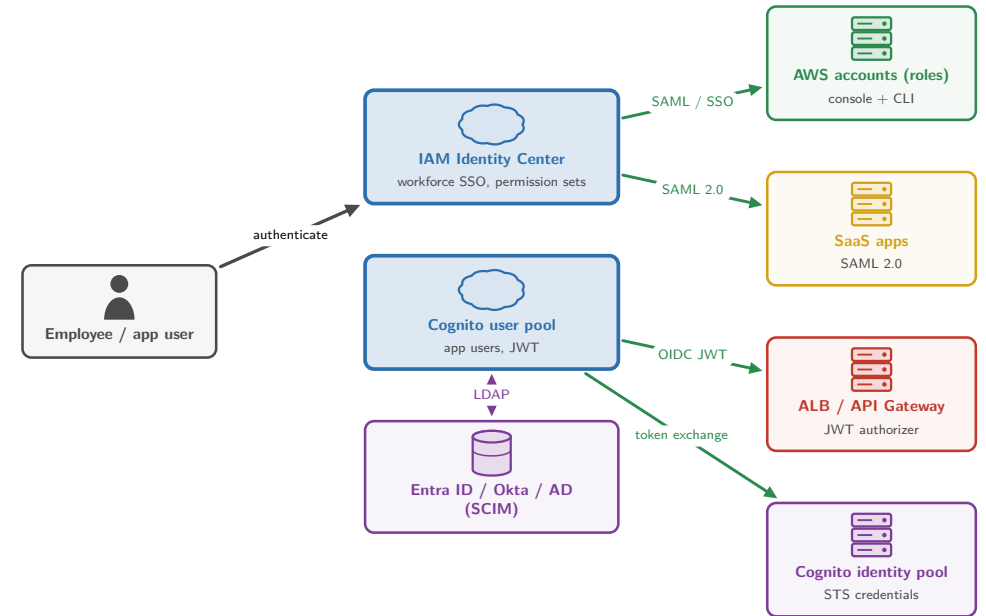
Same account: a resource policy that names the IAM user or the role session itself allows even when the boundary or session policy does not; naming the role (not the session) keeps those limits.

Evaluation: explicit deny anywhere wins; every guardrail (SCP, RCP, boundary, session policy) must allow; then an identity or resource allow is needed; else implicit deny. Here the identity policy allows s3:*, but the permissions boundary does not allow PutObject → implicit deny.

EXAM TIP

Cross-account: the request needs an allow on **both** sides (the caller's identity policy AND the resource policy or the role trust policy). Same account: either side is enough. 'Grant an application on EC2 access to S3' → instance profile role, never access keys in user data or code.

Federation and application identity

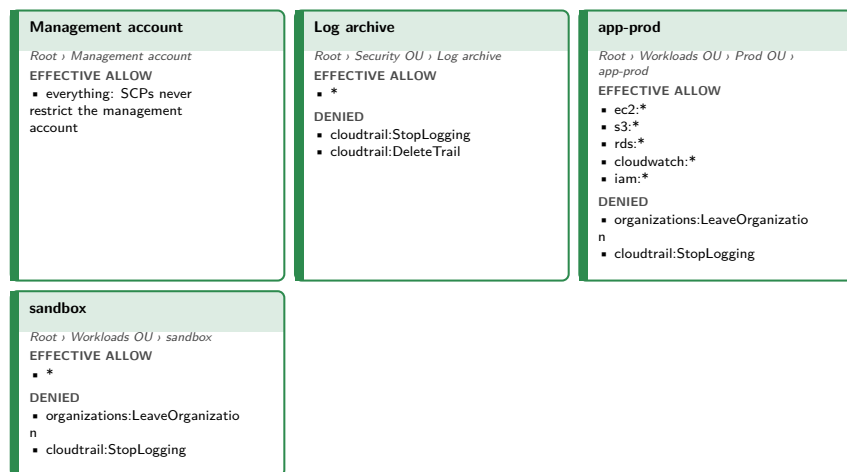
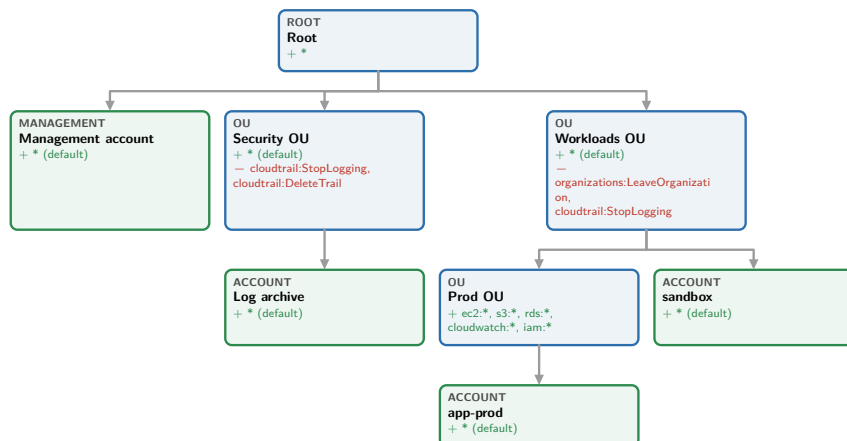


Workforce identities go through IAM Identity Center; customer identities through Cognito. Identity pools turn tokens into temporary AWS credentials for direct S3 or DynamoDB access.

Cognito user pool sign-up, sign-in, JWT tokens Tell: Does the client need to call AWS APIs directly? → identity pool	vs	Cognito identity pool temporary AWS credentials
IAM Identity Center workforce SSO into many accounts Tell: Many accounts in an organization → Identity Center	vs	IAM SAML federation one account, IdP → role directly
Resource-based policy caller keeps its own permissions Tell: Must the caller still write to its own account at the same time? → resource policy	vs	AssumeRole cross-account caller gives up its permissions for the role's
Permissions boundary caps one user or role Tell: Delegating role creation to developers → boundary; whole account → SCP	vs	SCP caps every principal of an account/OU
AWS Directory Service AD Connector proxy to on-prem AD, no cache Tell: Must work when the VPN is down / needs AD-aware apps in AWS → Managed AD	vs	AWS Managed Microsoft AD real AD in AWS, trusts on-prem

Multi-account with AWS Organizations

INTERSECTION: an action is allowed only if EVERY level allows it; a deny anywhere above applies below



Guardrails only filter: the identity policies in each account must still grant the action.
No assignments listed = the default full-access guardrail (*) stays attached.

SCPs intersect down the path: app-prod may only use what Root and Workloads AND Prod allow, minus every deny above it. The management account is never restricted. Typical OUs: Security (log archive, tooling), Infrastructure (network), Workloads (prod, non-prod).

Need AWS answer

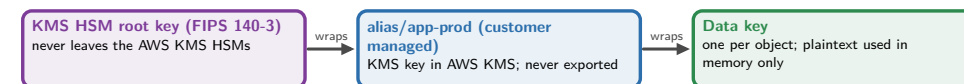
Stop any account from leaving the org or disabling CloudTrail	SCP with Deny on organizations:LeaveOrganization, cloudtrail:StopLogging
Restrict all workloads to EU regions	SCP Deny with aws:RequestedRegion NotEquals eu-*, exempting global services

Landing zone with guardrails in a few clicks	AWS Control Tower (preventive SCP, detective Config, proactive Hooks controls)
Share a subnet or a Transit Gateway with other accounts	AWS RAM (Resource Access Manager)
One bill, volume discounts, shared RIs and Savings Plans	consolidated billing (on by default in Organizations)
Restrict a bucket to principals of the organization	bucket policy condition aws:PrincipalOrgID
Cap what any external principal can do on your resources	resource control policies (RCPs), 2024+, for S3, STS, KMS, SQS, Secrets Manager

WARNING

An SCP grants nothing: with the default FullAWSAccess SCP removed and only an Allow-list SCP attached, every action not listed is denied, including for the account's administrators. The management account is never restricted by SCPs: keep workloads out of it.

Data protection: KMS, secrets, certificates



Encrypt (write)

- GenerateDataKey on the KMS key → plaintext data key + encrypted copy
- encrypt the object locally with the plaintext data key (AES-256-GCM)
- erase the plaintext key from memory
- store the ciphertext with the encrypted data key

Decrypt (read)

- read the encrypted data key stored with the object
- Decrypt: the service unwraps it with the KMS key (access policy and audit log apply)
- decrypt the object locally with the plaintext key, then erase it

STORED WITH THE OBJECT



Never stored: the plaintext data key or the KMS key itself.

OBJECTS: 200,000	GENERATEDATAKEY CALLS	DECRYPT CALLS	TOTAL REQUESTS
Without caching	200,000	200,000	400,000
Cache 1 key per 100 objects	2,000	2,000	4,000 (—99.0%)

calls = N without caching; calls = $\lceil N / R \rceil$ with caching, where N = objects written (each read once) and R = objects encrypted per cached data key. Caching cuts requests (quota and per-request price) but widens what one leaked key exposes.

ROTATION: every 365 days, key age 400 days

1 rotation so far → 2 key versions (the old ones are kept to decrypt older data); next rotation in 330 days. Rotation changes the key material only: the key id stays, existing data is not re-encrypted.

Envelope encryption: GenerateDataKey returns a plaintext and an encrypted data key; data is encrypted locally and the encrypted key is stored with it. KMS only ever sees small data keys, so object size is unlimited; caching a data key for 100 objects cuts KMS calls 100x (S3 Bucket Keys do this for SSE-KMS).

Option	Who holds the key	Audit / control	Pick it when
SSE-S3	S3 (AES-256), default since Jan 2023	none per key	encryption at rest required, nothing more
SSE-KMS (AWS managed key aws/s3)	KMS, AWS-managed policy	CloudTrail per use	audit trail, no cross-account need
SSE-KMS customer managed key	KMS, your key policy	CloudTrail, rotation, disable, cross-account grants	separation of duties, cross-account, revoke access
DSSE-KMS	two layers of KMS encryption	CloudTrail	compliance demanding dual-layer
SSE-C	you send the key with each request (HTTPS)	you	keys must stay outside AWS; S3 does not store them
Client-side	your app, before upload	you	AWS must never see plaintext
CloudHSM	single-tenant HSM cluster you manage (FIPS 140-3 level 3)	you own users and keys	custom key store, Oracle TDE, SSL offload, full key control

Automatic key rotation — Customer managed KMS keys rotate yearly by default when enabled (configurable 90-2560 days); old material is kept so existing ciphertext still decrypts. AWS managed keys rotate yearly.

Multi-Region key — KMS keys with the same key ID and material in several regions, to decrypt in a DR region without re-encrypting.

Secrets Manager vs Parameter Store — Secrets Manager: rotation, cross-region replicas, per-secret fee. SSM Parameter Store SecureString: free standard tier (4 KB, 10,000 parameters), no native rotation.

ACM (AWS Certificate Manager) — Free public TLS certificates with auto-renewal on ELB, CloudFront, API Gateway; for CloudFront the certificate must be in us-east-1. Not installable on EC2 directly.

```
Encrypt an existing unencrypted EBS volume or RDS instance
# EBS: snapshot -> encrypted copy -> new volume
aws ec2 create-snapshot --volume-id vol-0abc
aws ec2 copy-snapshot --source-snapshot-id snap-0abc \
  --source-region eu-west-3 --encrypted \
  --kms-key-id alias/app-prod
aws ec2 create-volume --snapshot-id snap-0new \
  --availability-zone eu-west-3a
# Account default for new volumes
aws ec2 enable-efs-encryption-by-default
# RDS: same idea, snapshot -> copy --kms-key-id -> restore
```

🔍 Threat detection and compliance services

Service	What it does	Exam keyword
Amazon GuardDuty	ML threat detection on CloudTrail, VPC Flow Logs, DNS logs, S3, EKS, RDS logins, Lambda, runtime	"detect compromised instances, crypto-mining"
Amazon Inspector	vulnerability scanning of EC2, ECR images, Lambda code (CVE, network reachability)	"scan for software vulnerabilities"
Amazon Macie	discovers and classifies sensitive data (PII) in S3	"find PII in buckets"
AWS Security Hub	aggregates findings, CSPM checks (CIS, AWS FSBP)	"single pane, compliance score"
Amazon Detective	graph investigation of GuardDuty findings	"root cause of a finding"

AWS Config	resource configuration history and compliance rules, auto-remediation via SSM	"track configuration changes, noncompliant resources"
AWS CloudTrail	API call history; org trail to a log-archive bucket	"who did what, when"
AWS WAF / Shield	layer-7 rules on CloudFront, ALB, API Gateway, AppSync, Cognito / DDoS (Standard free, Advanced paid with response team)	"SQL injection, rate limit / DDoS"
AWS Artifact	AWS compliance reports (SOC, ISO, PCI) and agreements	"download the SOC 2 report"

Least privilege for a cross-account reporting job

PROBLEM

A Lambda in account A (111111111111) must read objects under s3://finance-b/reports/ in account B, objects encrypted with a customer managed KMS key in B.

GIVEN

- Lambda execution role arn: aws:iam::111111111111:role/report-fn
- Bucket finance-b, KMS key alias/finance in account B

1

Identity policy on report-fn (account A)

Allow s3:GetObject on arn:aws:s3:::finance-b/reports/*; kms:Decrypt on the key ARN in B

Why: cross-account needs the caller's own allow

2

Bucket policy on finance-b (account B)

Principal role/report-fn, Allow s3:GetObject, Resource finance-b/reports/*

Why: the resource side must also allow

3

Key policy on alias/finance (account B)

Principal role/report-fn, Allow kms:Decrypt

Why: a KMS key must explicitly trust external principals

4

Check guardrails

no SCP or boundary denies s3:GetObject or kms:Decrypt

Why: an explicit deny anywhere wins

ANSWER

3 policy statements: identity (A), bucket (B), key policy (B). With an AWS managed aws/s3 key the job is impossible: its key policy cannot be edited for other accounts.

CHECK

✔ Test with aws s3 cp s3://finance-b/reports/q3.csv - from the function role; AccessDenied on KMS = missing key-policy grant.

🚩 Myths, takeaways, self-test

MYTH

❌ *An SCP Allow gives permissions to the account's users*

❌ *The root user should keep access keys for emergencies*

❌ *SSE-S3 means nobody at AWS-level can read my data*

❌ *KMS can encrypt my 2 GB file directly*

FACT

✅ **SCPs only cap permissions**

Users still need an identity or resource policy that allows the action.

✅ **Root has no access keys; protect it with MFA and lock it away**

Break-glass = a dedicated IAM role or user with MFA, monitored by CloudTrail alarms.

✅ **SSE-S3 protects disks, not access**

Anyone with s3:GetObject reads plaintext; use SSE-KMS with a key policy for a second gate.

✅ **KMS Encrypt accepts at most 4 KB**

Use GenerateDataKey and envelope encryption (the SDK and the AWS Encryption SDK do it).

Key takeaways

01

Roles, not keys

instance profiles, execution roles, Identity Center, STS

02

Deny wins, guardrails cap, one allow grants

SCP and boundary never grant

03

Cross-account = both sides allow

and the KMS key policy too

04

Customer managed KMS key for control

audit, revoke, cross-account, rotation

05

Name the detector

GuardDuty threats, Inspector CVEs, Macie PII, Config drift

Q1 An app on EC2 needs DynamoDB access: what do you configure?

A An IAM role attached through an instance profile; the SDK picks up rotating credentials from the instance metadata (IMDSv2).

Q2 Developers may create roles for their Lambdas but must never exceed a set of permissions. How?

A Require a permissions boundary on every role they create (condition iam:PermissionsBoundary on iam:CreateRole).

Q3 Mobile app users must upload photos straight to their own S3 prefix. Which services?

A Cognito user pool for sign-in, identity pool for temporary credentials, IAM policy with `#{cognito-identity.amazonaws.com:sub}` in the prefix.

Q4 Which condition key limits a bucket to accounts of your organization?

A `aws:PrincipalOrgID` in the bucket policy.

Q5 Where must the ACM certificate of a CloudFront distribution live?

A `us-east-1` (N. Virginia), whatever the origin region.

Q6 You must rotate RDS database passwords every 30 days without code changes. Service?

A Secrets Manager with managed rotation; the app reads the secret at runtime.

Q7 Does an SCP restrict the management account?

A No. SCPs apply to member accounts only (including their root users).

Q8 Unencrypted EBS volume must become encrypted. Steps?

A Snapshot, copy the snapshot with encryption and a KMS key, create a volume from it, swap it in.