RSAConference2020

San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

SESSION ID: ACB-F03

Scaling Key Management: Thousands of Clients, Trillions of Operations

Yaron Sheffer

Director, Security Technologies Product Development
Intuit

@yaronsheffer

Intuit

- A global financial platform company
- Maker of turbotax quickbooks mint
- More than **50M customers** trust us with their financial data
- Steadfast commitment to security
- Strategic move to the AWS cloud
- Acquired Israeli security startup Porticor in Feb. 2015

IDPS, Intuit Data Protection Service

- Secret and Key Management service
  - Storage for secrets and encryption keys
  - Cryptographic operations

- For highly sensitive data we require both transparent disk-level and **application-level data encryption**

RSAConference2020

Core Features

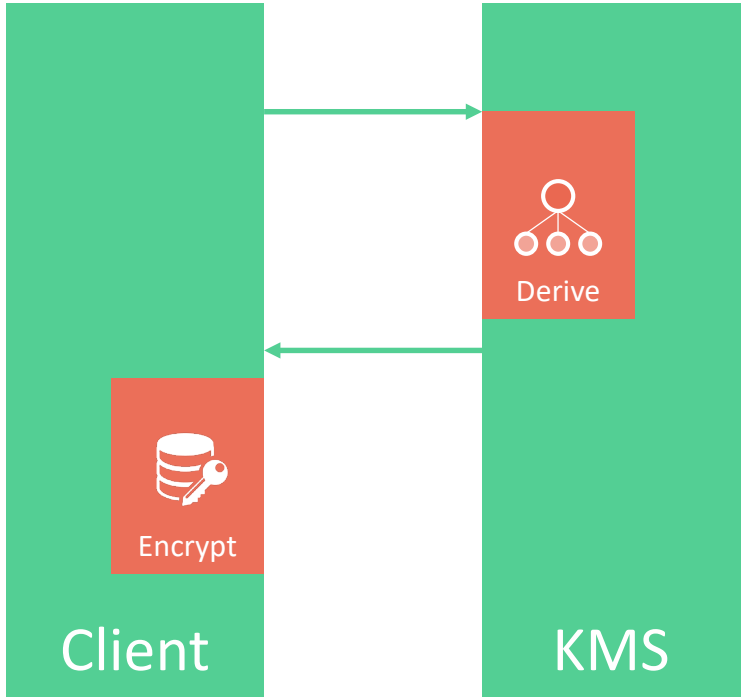
Where to Encrypt

- Fetch the key into the client, then encrypt
- Alternatively, send the data to be encrypted on the KM service

---

- Instead: RDLE – **Remote Derive, Local Encrypt**

- Fine-grain keys: implications on threat model
  - Spatial granularity
- Also on performance, scalability



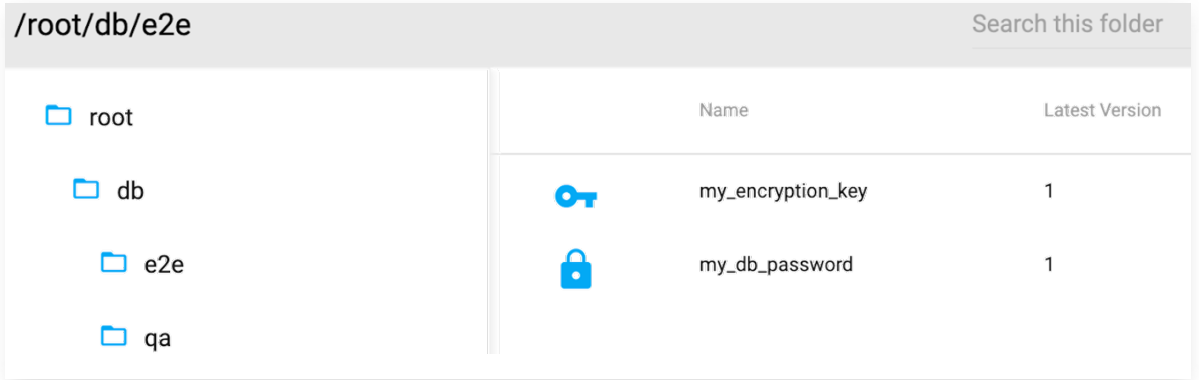
Temporal Granularity: Key Versions

- New key version every 90 days
- Key versioning vs. re-encryption (vs. “key rotation”)
- **Versioning is automated**, saved as part of a ciphertext header
  - No industry standard, sigh

- Versioning works well with probabilistic encryption
- More challenging with deterministic encryption

Managing Keys

- Each project receives a **strictly segregated namespace**
- A namespace is a hierarchy of folders and keys
  - Keys consist of multiple key versions
- Keys can be created, listed etc. with a UI and a REST API
- Role-based access control at the folder level
- Custom permissions
  e.g. read public-key only



Client Authentication

- Initially, had an OAuth1-like message signing
  - Assert long-term credentials, obtain a short-term bearer token
- Found out this is not workable: turtles all the way down
- Built a separate **policy authentication** service
  - Policy: AWS role plus additional properties
  - Good for: EC2 instances, Lambda functions, Docker containers, more
- Now being reused for other internal services

The Customer View

- IDPS only used by *internal* customers

- SDK: Java, Python, Go, JavaScript
  - Using the REST API directly is hard
- UI and a CLI tool for secret management

- No manual handling of keys
- Enterprise-wide governance

Why Have Your Own KMS?

RSAConference2020

Service Architecture

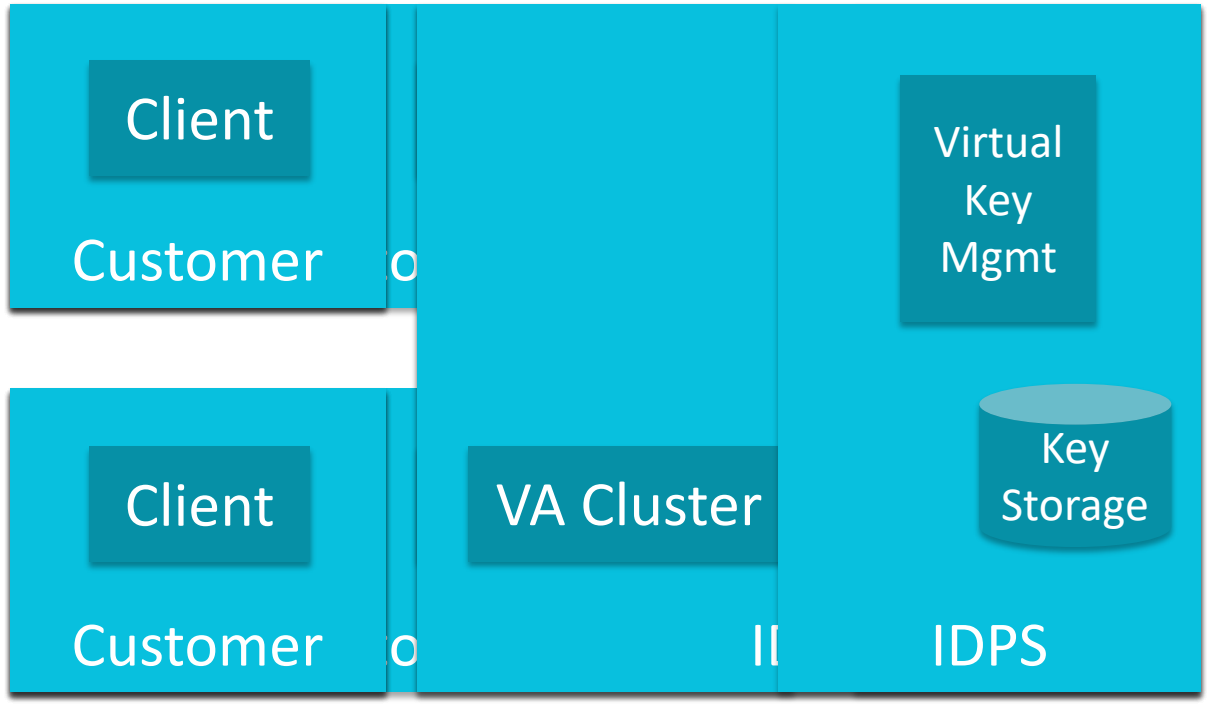
The Architectural Journey

1. Virtual Appliance (VA) owned by customer, master key

2. Fully managed service, including VA clusters

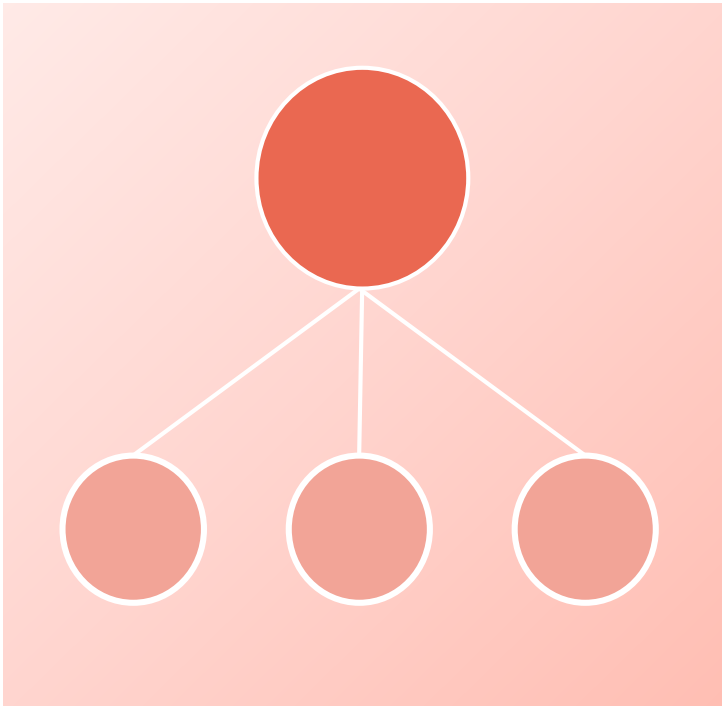
3. End-to-end secrecy

4. Multitenant key derivation



Evolving Key Derivation

- Need a deterministic function of a secret (“root key”) and a non-secret input, returning a secret derived key
  - With several security guarantees
- HKDF, RFC 5869
- How to do it in a shared environment? **Oblivious PRF**!
- The service uses a blinded version of the root key to derive a subkey
  - Which is then unblinded by the client



IDPS Technology

- EC2 instances
  - Formerly also hosted in a traditional data center
- RHEL, moving to Amazon Linux 2
- Go
  - Including most crypto code
- On the client side, primarily Java
  - Bouncy Castle for the crypto code
  - Apache Commons Crypto for performance
- Moved from CloudFormation to Terraform

RSAConference2020

Developed for Security

Security Principles

- IDPS is primarily about securing applications, not users
- Keys are only in the clear when used
- Cryptography is bog standard
  - When in doubt, call a cryptographer

Security Principles

- Humans are fallible
  - Dev/ops separation
  - No access to encryption keys, by anybody

Security Infrastructure: Intuit

- Large scale AWS account management
  - Including centralized SSO
- Automated policy enforcement
  - Both via tickets and real-time technical intervention
- Internal red team
- Strong compliance org/culture
- Cost tracking tools

Security Infrastructure: Unique to IDPS

- Zoning and containment: multiple separate AWS accounts
- SELinux
  - A strict custom policy reduces risk from code vulns and Golang zero days
- Dedicated instances
  - They come at a cost
- The Meltdown/Spectre family of issues validated these decisions

Security Infrastructure: Unique to IDPS (Cont.)

- Very conservative: no Lambda functions, no containers
  - But we support any *customer* deployment model
- Short term key backup allows rapid recovery of customer keys
- Custom disaster recovery aimed primarily at **malicious service interruption**

RSAConference2020

Wrap Up

Apply What You Have Learned Today: Recommendations

- In the next **two weeks** verify:
  - That you have a valid key management strategy, one that applies to modern automated key use
  - That you can identify its true security value, beyond “compliance”

- In the next **two months** make sure:
  - That this strategy remains relevant when you transition to the cloud
  - That security-critical infrastructure in the cloud is built with cloud ab/use cases in mind

Summary

- **Security-sensitive services can be run in the cloud**
- Consider, in depth, your particular cloud threat model
- But remember it's a moving target
- Security-optimized architecture, without hindering developer productivity

Security software in the cloud: Not easy but definitely fun!