

IEEE KEY MANAGEMENT SUMMIT 2008

Voltage Key Management
Architecture

Terence Spies
CTO

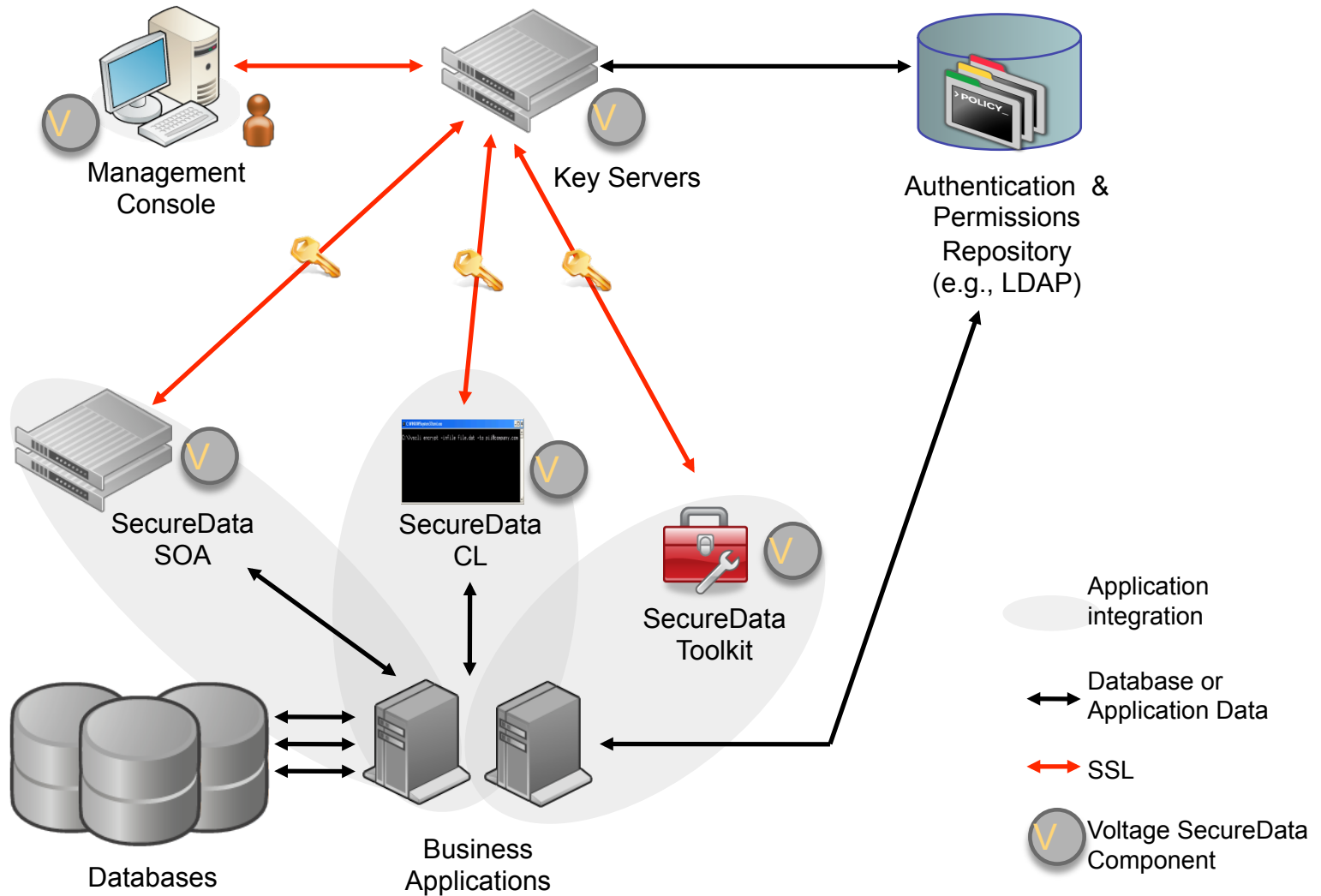
Voltage Security, Inc.

IEEE KEY MANAGEMENT SUMMIT 2008

Overview

- | Key Server Architecture
 - | Key Naming
 - | Key Generation
 - | Authentication
- | Applications
- | Programming Model
 - | Toolkit
 - | SOAP

Architecture



Key Management Functions

- | Key server runs “districts”
 - | Set of base keys, authentication rules
 - | Also includes branding for UI
- | Districts can build sym, DSA, IBE keys
 - | IBE keys for cross-domain
 - | Symmetric for single domain
- | Admin
 - | Logging, dual control polices
 - | Pushes encryption rules to clients

Key Naming

- | All keys have name and time
 - | Names: <name>@<district>
 - | Time: UTC rounded to policy granularity
 - | Hashed offset randomizes update time
- | Districts
 - | Domain name of key server
 - | Version number of base key
- | Load balancing via DNS
- | Domain name lets us use TLS directly

Key Generation

- | Heavy use of KDFs to create subkeys
- | Derivation allows:
 - | Stateless replication of KS function
 - | Single backup operation
 - | No loss of keys or db
- | IBE functions as PK “derivation”
- | Two rollover paradigms
 - | Update base key, update key time

Authentication

- | Core KM function!
- | All requests HTTPS
- | Authentication options
 - | Authenticate web connection
 - | Certificates
 - | Basic Auth
 - | Active Directory / Kerberos / LDAP
 - | Redirect to web auth
 - | User portal or custom application

Applications

- | Email
- | File sharing
- | Local file encryption
- | Database
 - | AES
 - | FPE
- | Integration APIs

Format-Preserving Encryption

- | Keyed permutation model
 - | Encrypt CCN -> CCN
 - | Encrypt SSN -> SSN
 - | Encrypt custom to custom
- | Untrusted applications see proxy data
- | Trusted applications can decrypt
- | Details: NIST modes development site

Programming Model

- | Three options
 - | C/C++/Java Toolkit
 - | SOAP Server
 - | Command Line
- | Rule #1: Simple, simple, simple
 - | Encrypt: policy + data
 - | Decrypt: credential + ciphertext
- | Policy: user, group, application

Web Services Example

```
VibeSimpleSOAPStub service =  
(VibeSimpleSOAPStub) new  
VibeSimple_ServiceLocator().getVibeSimpleSOAP();
```

```
String ccNum = "4391471208007120";
```

```
String keyName = "pci@company.com";
```

```
String encryptedCC =  
service.protectCreditCardNumber(ccNum, NULL,  
keyName, NULL, "UserPassword", "app1:54d8sd3");
```

Web Services

- | Integrates almost everywhere
 - | COBOL
 - | Z/OS, Tru64, etc, etc.
- | All operations logged/audited
- | Local and WS APIs use same model

Programming Model Design

- | Previous attempts
 - | BSafe, CryptoAPI
- | Lessons learned
 - | “Just encrypt it”
 - | Set central defaults
 - | Few need/want to choose algorithms
 - | ACLing model

Lessons Learned

- | Start with the applications
 - | Easy to pile on requirements in a vacuum
 - | How many X.509 extensions are used?
- | Simplify, simplify, simplify
 - | Unused security architecture = useless
 - | More knobs, more mistakes
 - | Most applications have simple needs
- | Key recovery is mandatory
 - | Regulatory driver