Key Management Death Match?

Marc Massar, CISSP, NSA-IAM

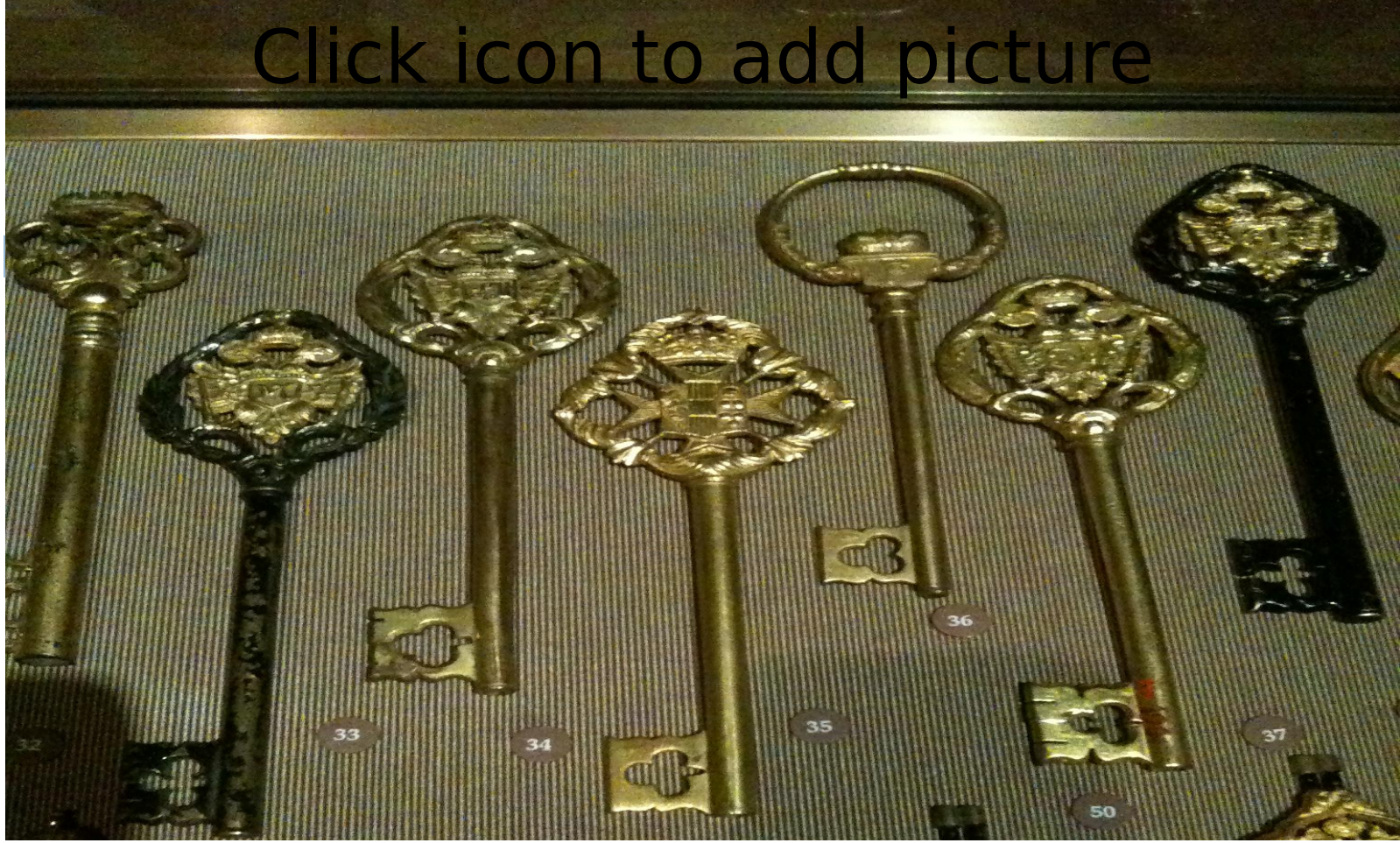
DEEPSEC IDSC2009

Competing KM Standards Technical Deep Dive

Introduction

- The Problem – Why So Many Key Management Products?
- More Problems – Interoperability
- The Contenders
- Details of the Standards/Protocols


Im In Ur Hausse
Stealin yur keyz
ICANHASCHEEZBURGER.COM



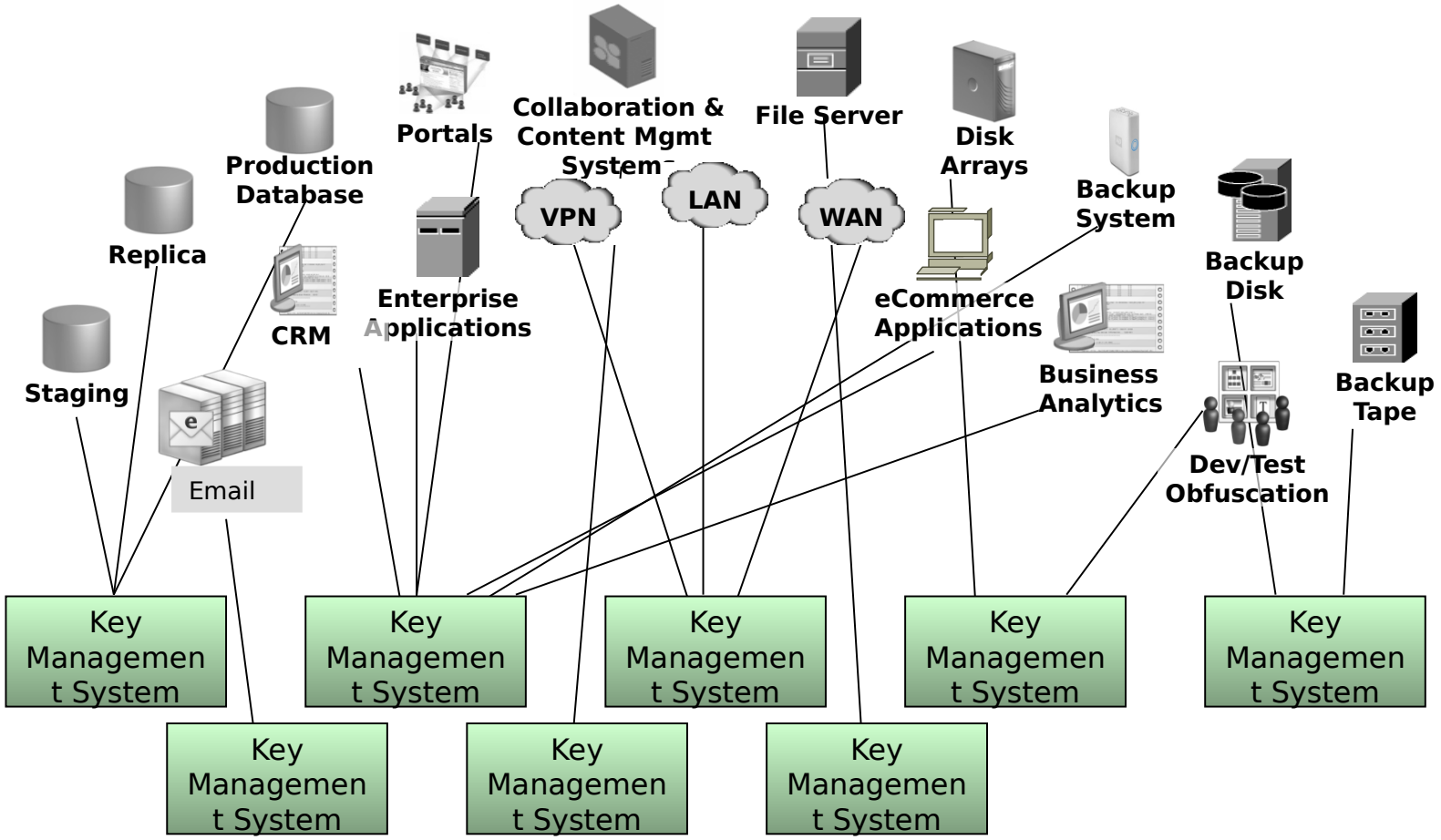
4

So Many Keys, So Little Time

11/19/2009

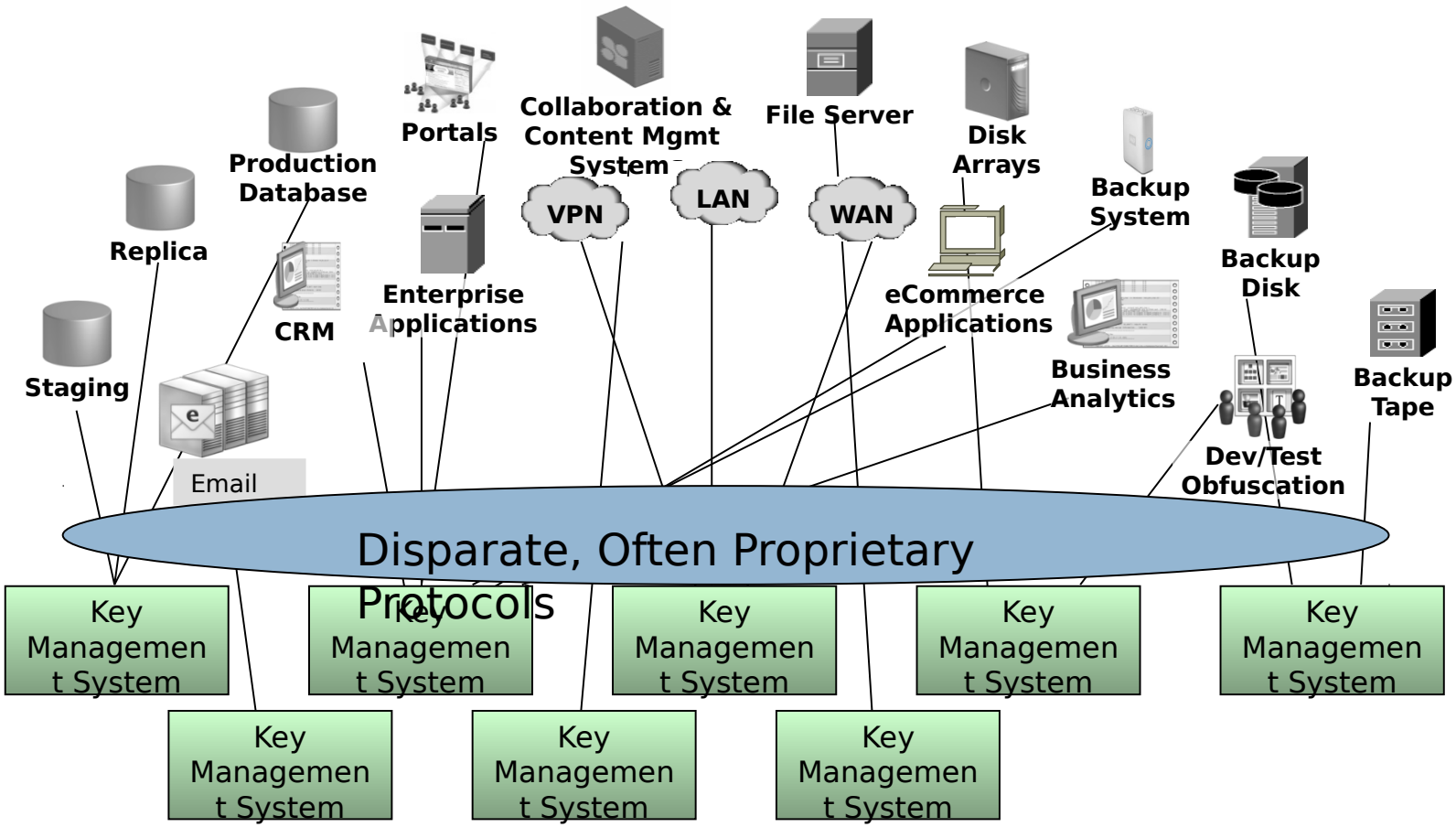
The Problem – The Many Uses of Crypto

Enterprise Cryptographic Environments

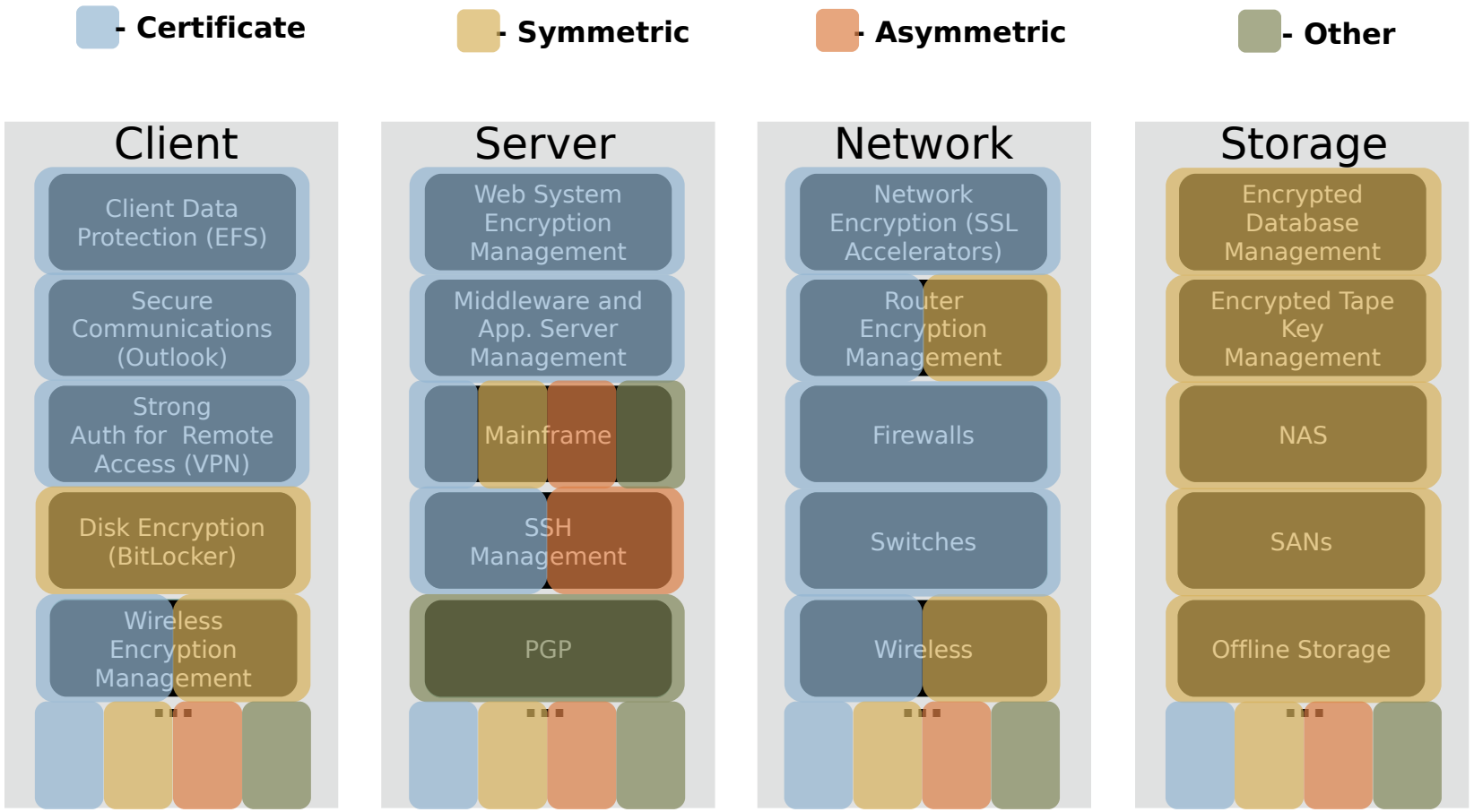


And More – Interoperability

Enterprise Cryptographic Environments



And More – The Many Uses of Crypto



Encryption is Business Critical

- You don’t encrypt worthless information – you only encrypt the most important information
- Encryption Impacts Your Business Processes!
- It Matters Who Encrypts…And Matters More Who Decrypts!
- Value Transfer from Data to Keys

Click icon to add picture

The Contenders

The Contenders – KMIP

Round 1 – KMIP

KMIP is an open standard backed by OASIS with members including IBM, EMC/RSA, Thales, LSI, NIST, SafeNet, and quite a few more. KMIP came out of the IEEE 1619.3 effort

- Key Management Interoperability Protocol
  - …will develop specification(s) for the interoperability of KM services with KM clients…will address anticipated customer requirements for key lifecycle management (generation, refresh, distribution, tracking of use, life-cycle policies including states, archive, and destruction), key sharing, and long-term availability of crypto objects of all types…
- In Scope – Just about everything
- Out of Scope – Implementation and framework details

The Contenders – IEEE 1619.3

Round 2 - IEEE 1619.3

Formed as a sub-committee of IEEE 1619 SISWG (Security in Storage Working Group). Members include Cisco, EMC/RSA, LSI, Vormetric, and others.

- 1619.3 – A Sub-Committee of 1619 Working Group
  - …standard defines methods for the storage, management, and distribution of cryptographic keys used for the protection of stored data. This standard augments existing KM methodologies to address issues specific to cryptographic protection of stored data. This includes stored data protected by compliant implementations of other standards in the IEEE 1619 family.
- In Scope – Protection of stored data (interfaces, methods, and algorithms)
- Out of Scope – Transport Encryption, non-storage use cases

The Contenders - EKMI

Round 3 – EKMI

Another OASIS committee. This one formed before KMIP and includes members from Red Hat, CA, Wells Fargo, PayPal, and PrimeKey.

- Enterprise Key Management Infrastructure
  - …TC will create use-case(s) that describe how and where the protocols it intends to create, will be used
  - …TC will define symmetric key management protocols…
  - …ensure cross-implementation interoperability, the TC will create a test suite…will allow different implementations of this protocol to be certified…
  - …TC will provide guidance on how a symmetric key-management infrastructure may be secured using asymmetric keys…
  - …in conjunction with other standards organizations that focus on disciplines outside the purview of OASIS, the TC will provide input on how such enterprise KM infrastructures may be managed…
  - …conduct other activities that educate users…
- In Scope – All symmetric secrets secured using the defined KM Infrastructure
- Out of Scope – Asymmetric KM, some implementation details

The Contenders – IETF KeyProv

Round 4 – IETF KeyProv

Provisioning of Symmetric Keys. This committee has been inactive and active again recently and has participation from NIST, ActivIdentity, and others. Released DSKPP – Dynamic Symmetric Key Provisioning Protocol and PSKC – Portable Symmetric Key Container

- Provisioning of Symmetric Keys
  - …to define protocols and data formats necessary for provisioning of symmetric cryptographic keys and associated attributes…consider use cases related to use of Shared Symmetric Key Tokens. Other use cases may be considered for the purpose of avoiding unnecessary restrictions in the design and ensure…future extensibility.
- In Scope – Provisioning of Symmetric keys (think existing devices)
- Out of Scope – Asymmetric keys, specific implementations

14

Sizing Up the Competition

KMIP Overview

- Community Draft Level – version 1.0
- Binary Protocol
- TTLV – Tag Type Length Value
- Standard defines – Objects, Attributes, and Operations
  - Objects – Base Objects like Key Block, Key Value, Key Wrapping Data
  - Objects – Managed Objects like Certificates, Keys, Key parts, template data
  - Attributes – Identifier, State, Usage Limits, Algorithm, Length, Issuer, Application data
  - Operations – Create, Register, Re-Key, Derive, Get, Modify Attributes, Activate, Revoke, Destroy
- List not all inclusive

IEEE 1619.3 Overview

- Draft 7 August 2009 – Probably a draft 8 soon
- Defines a KM architecture model, KM Conceptual model, Lifecycle model, KM Sequence models, Object models, and Operation models
  - These models are all specific to “data at rest”
- Does NOT define a message between actors
  - Proposed adoption of KMIP binary protocol for communication between actors
- Defines Key naming extensively – global uniqueness
- Also calls for XML message not yet defined…likely to adopt something that one of the other committees proposes (EKMI, IETF, or XML from KMIP)

EKMI Overview

- SKSML – Symmetric Key Services Markup Language 1.0 PR02 Draft 8
  - Mobile – SKSML available as well
- Committee defines not only the semantics of symmetric key exchange (XML – SKSML) but also the components required to make that exchange secure
  - SKMS, SKS, SKCL, uses PKI as the trust mechanism for key exchange
- Very well defined set of Requests/Responses to cover multiple use cases

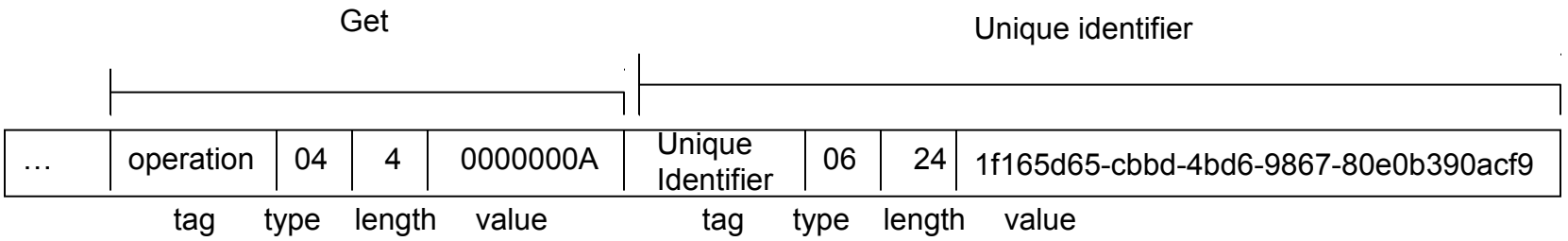
IETF KeyProv Overview

- Leverages RFC4758 – CT-KIP (Cryptographic Token Key Initialization Protocol)
- DSKPP – 1.0 draft 9
- PSKC – 1.0 draft 4
- Symmetric Key Format doc – 1.0 draft 6
- Does not define an architecture per se, but does outline the use cases around provisioning keys to Internet accessible cryptographic systems
  - Covers typical Client – Server interactions
  - Defines “entities” that are actors in the use cases
- DSKPP allows for 2-pass and 4-pass messages between client and server

19

The Messages

Message Layout – KMIP



Tag	Type	Length	Value
Attribute	Structure	<varies>	

Tag	Type	Length	Value
Attribute Name	String	<varies>	“Application Specific ID”
Attribute Index	Integer	4	2
Attribute Value	Structure	<varies>	

Tag	Type	Length	Value
App. Name	String	<varies>	“ssl”
App. ID	String	<varies>	“www.example.com”

Message Sample – KMIP

42007801000001204200770100000038420069010000002042006A020000000400000001000000004200 6B020000000400000000000000004200 0D020000000400000001000000004200 0F01000000D842005C050000000400000001000000004200 7901000000C042005705000000040000000200000000420091010000000A842000801000000304 2000A07000000174372797074 6F6772617068696320416C676F726974686D0042000B050000000400000000300000000420008010000003042000A07000000144372797074 6F6772617068696320 4C656E67746800000000420 00B020000000400000000C00000000

Create Key with the following data - In: objectType=’00000002’ (Symmetric Key), attributes={ CryptographicAlgorithm=’00000003’ (AES), CryptographicLength=’128’, CryptographicUsageMask=‘0000000C’ }

```
Tag: Request Message (0x420078), Type: Structure (0x01), Data:
  Tag: Request Header (0x420077), Type: Structure (0x01), Data:
    Tag: Protocol Version (0x420069), Type: Structure (0x01), Data:
      Tag: Protocol Version Major (0x42006A), Type: Integer (0x02), Data: 0x00000001 (1)
      Tag: Protocol Version Minor (0x42006B), Type: Integer (0x02), Data: 0x00000000 (0)
    Tag: Batch Count (0x42000D), Type: Integer (0x02), Data: 0x00000001 (1)
  Tag: Batch Item (0x42000F), Type: Structure (0x01), Data:
    Tag: Operation (0x42005C), Type: Enumeration (0x05), Data: 0x00000001 (Create)
    Tag: Request Payload (0x420079), Type: Structure (0x01), Data:
      Tag: Object Type (0x420057), Type: Enumeration (0x05), Data: 0x00000002 (Symmetric Key)
      Tag: Template-Attribute (0x420091), Type: Structure (0x01), Data:
        Tag: Attribute (0x420008), Type: Structure (0x01), Data:
          Tag: Attribute Name (0x42000A), Type: Text String (0x07), Data: Cryptographic Algorithm
          Tag: Attribute Value (0x42000B), Type: Enumeration (0x05), Data: 0x00000003 (AES)
        Tag: Attribute (0x420008), Type: Structure (0x01), Data:
          Tag: Attribute Name (0x42000A), Type: Text String (0x07), Data: Cryptographic Length
          Tag: Attribute Value (0x42000B), Type: Integer (0x02), Data: 0x00000080 (128)
        Tag: Attribute (0x420008), Type: Structure (0x01), Data:
          Tag: Attribute Name (0x42000A), Type: Text String (0x07), Data: Cryptographic Usage Mask
          Tag: Attribute Value (0x42000B), Type: Integer (0x02), Data: 0x0000000C (Encrypt, Decrypt)
```

Object Naming Sample – IEEE 1619.3

- Can format all objects as URIs
- SO_GUID = SO_Family "://" SO_Domain SO_Context SO_Handle / SO_Directory
- km://example.org/key/dir1/dir2/key123
- km://example.com/key/dir1/%00%00%EA%05
- km://traders.bigbank.com/key/000102030405060708090A0B0C0D0E0F
- km://example.net/policy/storsecpolicy/kmspolicy/keypolicy3
- Allows for integration with EKMI SKSML
- Note that IEEE 1619.3 will adopt KMIP binary encoding

Request XSD – EKMI


```
<xsd:element name="SymkeyRequest">
    <xsd:annotation>
      <xsd:documentation>
        This element requests a new, or an existing, symmetric
        encryption key from an SKS server.  It contains a GlobalKeyID
        child element, which is the global key identifier (GlobalKeyID)
        of the requested key and an optional KeyClasses element
        containing a list of KeyClass elements.  The number of
        KeyClass elements indicates the number of symmetric keys
        being requested by the client.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:choice>
        <xsd:sequence>
          <xsd:element name="GlobalKeyID" type="ekmi:GlobalKeyIDType"
minOccurs="1" maxOccurs="unbounded">
            <xsd:annotation>
              <xsd:documentation>
              The global key-identifier being requested.  A
              GlobalKeyID of 10514-0-0 is a request for a new
              symmetric key; all other values indicate an
              existing symmetric key.
              </xsd:documentation>
            </xsd:annotation>
          </xsd:element>
```

Note the XSD is continued on next slide. Trailing elements were truncated on next slide, so XSD is not complete.

Request XSD – EKMI


```
<xsd:element name="KeyClasses" type="ekmi:KeyClassesType" minOccurs="0" maxOccurs="1">
                <xsd:annotation>
                   <xsd:documentation>
                   An optional qualifier that indicates the types of
                   symmetric keys being requested by the client
                   application.  KeyClasses are application-defined
                   and site-specific.
                   </xsd:documentation>
                </xsd:annotation>
             </xsd:element>
             <xsd:element name="X509EncryptionCertificate" type="ekmi:X509CertificateType" minOccurs="0"
maxOccurs="1">
                <xsd:annotation>
                   <xsd:documentation>
                   An optional X509-compliant digital certificate sent
                   by SKMS clients and used by SKS servers to encrypt
                   the symmetric-key payload when responding to the
                   client.
                   </xsd:documentation>
                </xsd:annotation>
             </xsd:element>
          </xsd:sequence>
          <xsd:sequence>
             <xsd:element name="SymkeyRequestID" type="ekmi:SymkeyRequestIDType" minOccurs="1"
maxOccurs="unbounded">
                <xsd:annotation>
                   <xsd:documentation>
                   This element indicates that the client is checking
                   on the status of a previous request from which it
                   received a SymkeyRequestID from the SKS server.
```

2-Pass Key Request Sample – KeyProv


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <dskpp:KeyProvClientHello
       xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc"
       xmlns:dskpp="urn:ietf:params:xml:ns:keyprov:dskpp"
       xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       Version="1.0">
       <dskpp:DeviceIdentifierData>
           <dskpp:DeviceId>
               <pskc:Manufacturer>TokenVendorAcme</pskc:Manufacturer>
               <pskc:SerialNo>987654321</pskc:SerialNo>
               <pskc:StartDate>2009-09-01Z</pskc:StartDate>
               <pskc:ExpiryDate>2014-09-01Z</pskc:ExpiryDate>
           </dskpp:DeviceId>
       </dskpp:DeviceIdentifierData>
       <dskpp:SupportedKeyTypes>
           <dskpp:Algorithm>
               urn:ietf:params:xml:ns:keyprov:pskc#hotp
           </dskpp:Algorithm>
           <dskpp:Algorithm>
          http://www.rsa.com/rsalabs/otps/schemas/2005/09/otps-wst#SecurID-AES
           </dskpp:Algorithm>
       </dskpp:SupportedKeyTypes>
       <dskpp:SupportedEncryptionAlgorithms>
           <dskpp:Algorithm>
               http://www.w3.org/2001/04/xmlenc#rsa_1_5
           </dskpp:Algorithm>
       </dskpp:SupportedEncryptionAlgorithms>
```

2-Pass Key Request Sample – KeyProv


```
<dskpp:SupportedMacAlgorithms>
          <dskpp:Algorithm>
              http://www.ietf.org/keyprov/dskpp#dskpp-prf-sha256
          </dskpp:Algorithm>
      </dskpp:SupportedMacAlgorithms>
      <dskpp:SupportedProtocolVariants>
          <dskpp:TwoPass>
              <dskpp:SupportedKeyProtectionMethod>
                  urn:ietf:params:xml:schema:keyprov:dskpp#transport
              </dskpp:SupportedKeyProtectionMethod>
              <dskpp:Payload>
                  <ds:KeyInfo>
                      <ds:X509Data>
                          <ds:X509Certificate>
  INSERT X509 CERTIFICATE HERE
                          </ds:X509Certificate>
                      </ds:X509Data>
                  </ds:KeyInfo>
              </dskpp:Payload>
          </dskpp:TwoPass>
      </dskpp:SupportedProtocolVariants>
      <dskpp:SupportedKeyPackages>
          <dskpp:KeyPackageFormat>
              urn:ietf:params:xml:ns:keyprov:pskc#KeyContainer
          </dskpp:KeyPackageFormat>
      </dskpp:SupportedKeyPackages>
```

2-Pass Key Request Sample – KeyProv


```
<dskpp:AuthenticationData>
          <dskpp:ClientID>AC00000A</dskpp:ClientID>
          <dskpp:AuthenticationCodeMac>
              <dskpp:Nonce>

ESIzRFVmd4iZqrvM3e7/ESIzRFVmd4iZqrvM3e7/ESI=
              </dskpp:Nonce>

<dskpp:IterationCount>100000</dskpp:IterationCount>
              <dskpp:Mac
                  MacAlgorithm=
                  "
http://www.ietf.org/keyprov/dskpp#dskpp-prf-sha256">
                  3eRz51ILqiG+dJW2iLcjuA==
              </dskpp:Mac>
          </dskpp:AuthenticationCodeMac>
       </dskpp:AuthenticationData>
   </dskpp:KeyProvClientHello>
```

28

Overlap

Where They Overlap

- Storage – KMIP and IEEE 1619.3
- Application Friendly – XML
  - KeyProv, EKMI, Future KMIP
- Broadest scope – KMIP and EKMI
- HSM Integration – KMIP and 1619.3

- No Easy Answers
- And some areas are not covered – Financial Use Cases are largely ignored

Questions?

marc@techandbull.com

http://www.linkedin.com/in/marcmassar