HyperLedger Fabric
密码算法及相关应用场景解析

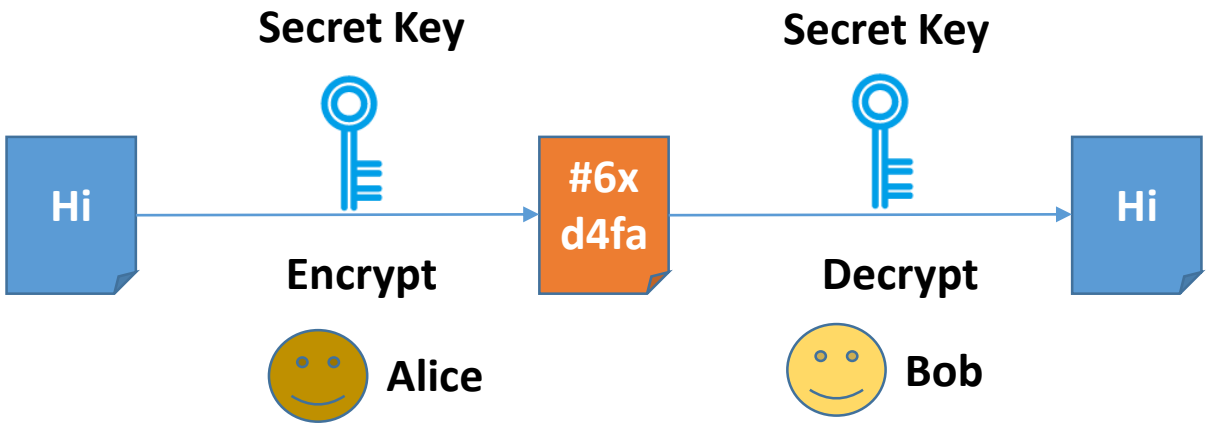
刘地军
中国网安
flyinox31@gmail.com

CONTENTS

密码算法基本概念

对称密码算法

- 加解密双方共享一个秘钥
- 快速
- 如何共享秘钥



密码算法基本概念

哈希函数

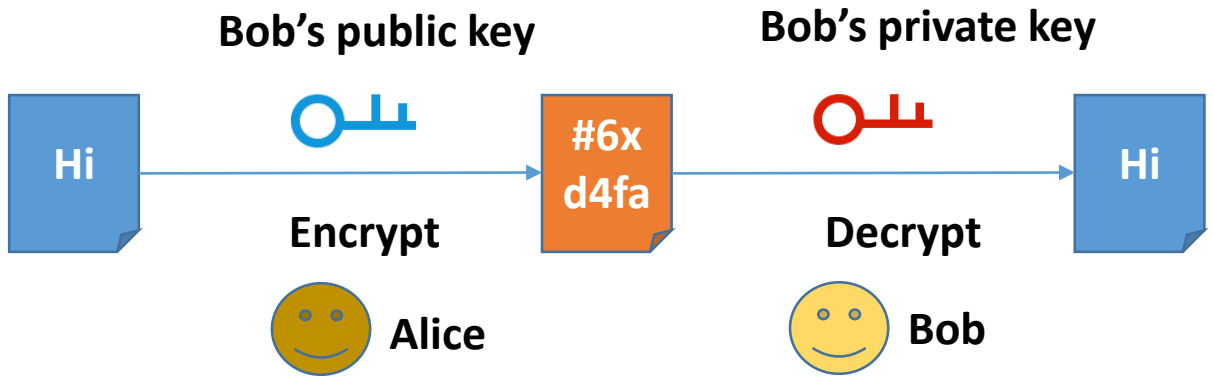
- 任意长度数据 =》 固定长度数据
- 单向
- 抗碰撞

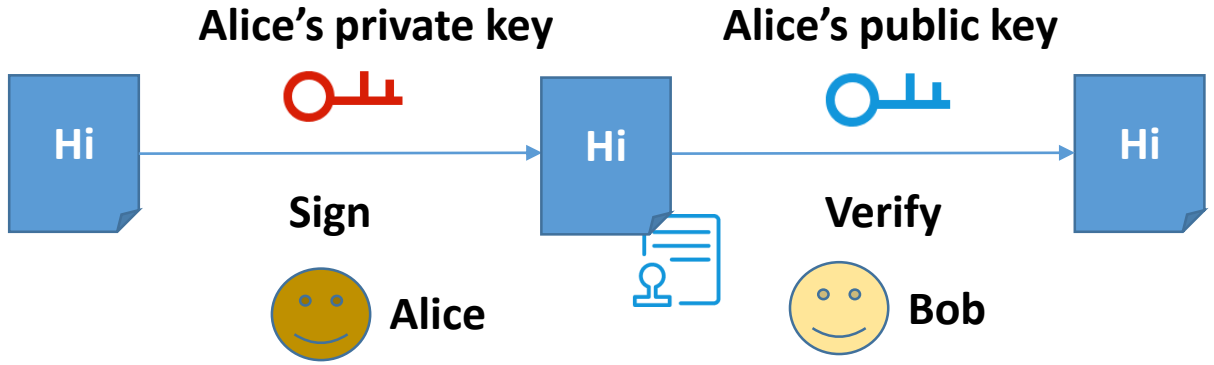
- 完整性校验
- 数字签名输入
- 生成索引

密码算法基本概念

非对称密码算法

- 公私钥对
- 加解密/数字签名
- 防篡改、伪造、防抵赖

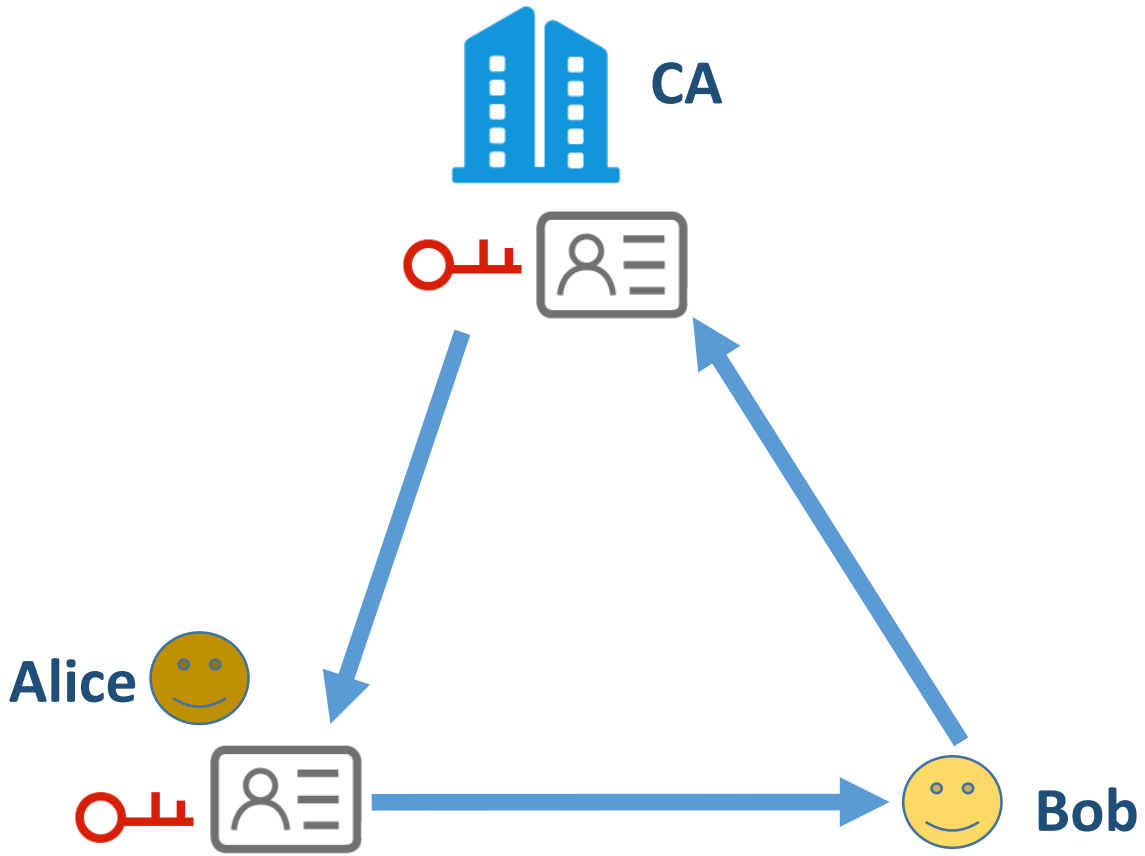




密码算法基本概念

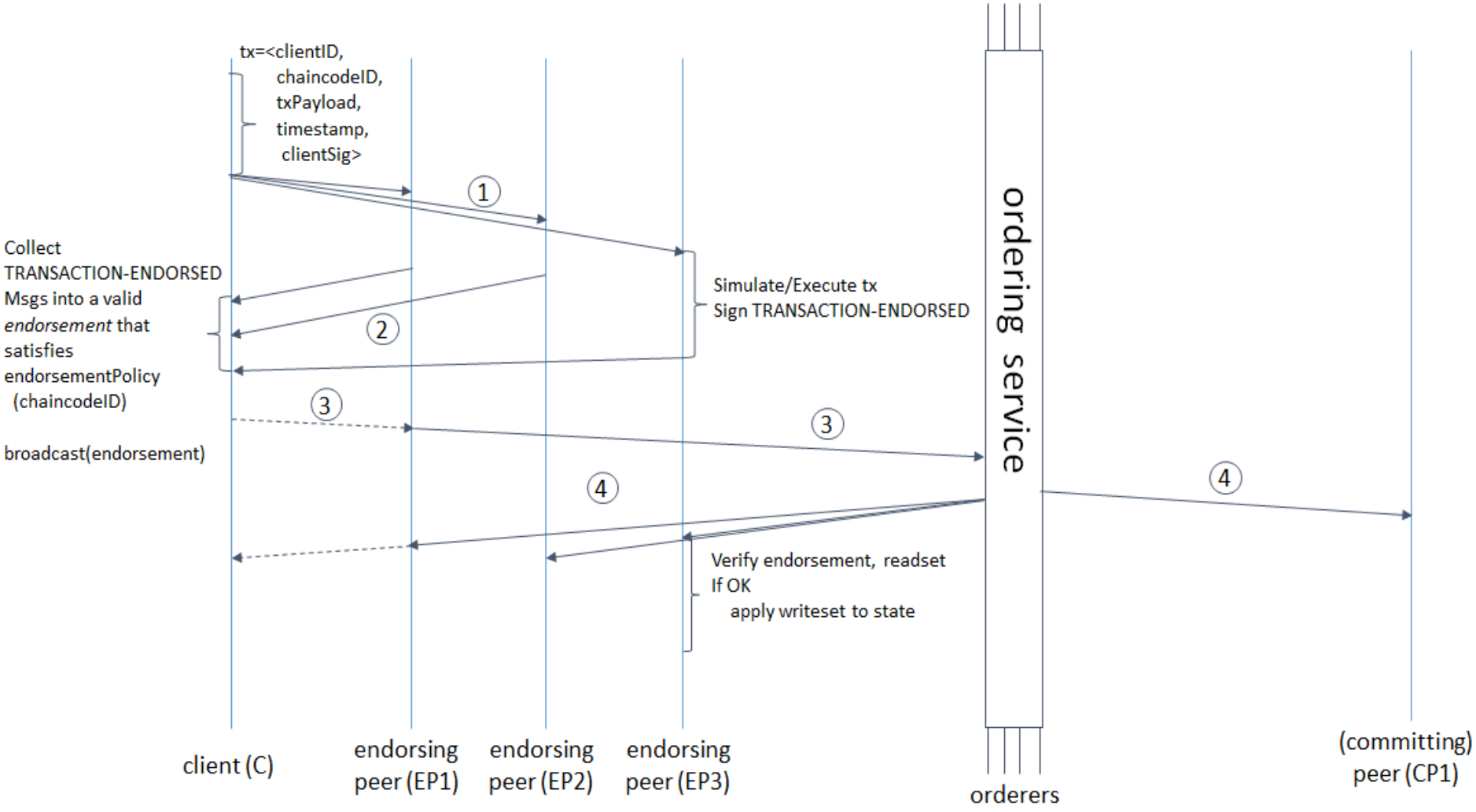
数字证书及PKI

- CA为用户签发证书
- 分发证书
- 其他用户通过CA验证证书



Fabric密码算法应用场景 – 数字签名

Transaction flow



Fabric密码算法应用场景 – 数字签名

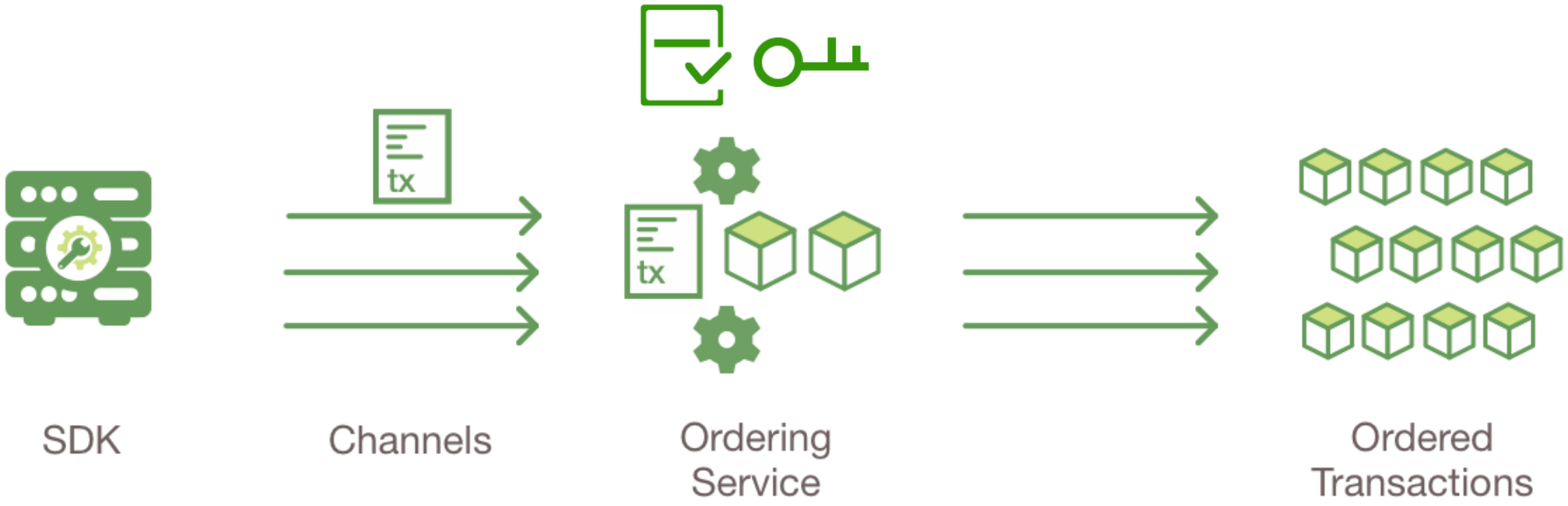
EndorseProposal

Fabric密码算法应用场景 – 数字签名

Simulate/execute and Endorse

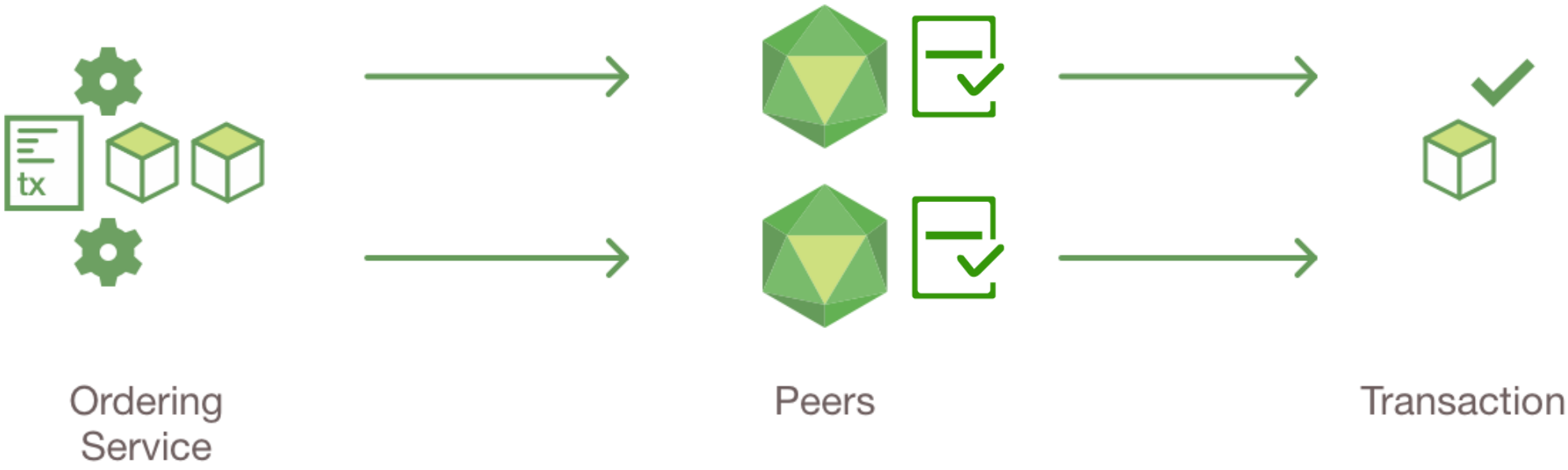
Fabric密码算法应用场景 – 数字签名

Consensus and generate block



Fabric密码算法应用场景 – 数字签名

Validate tx and commit to block



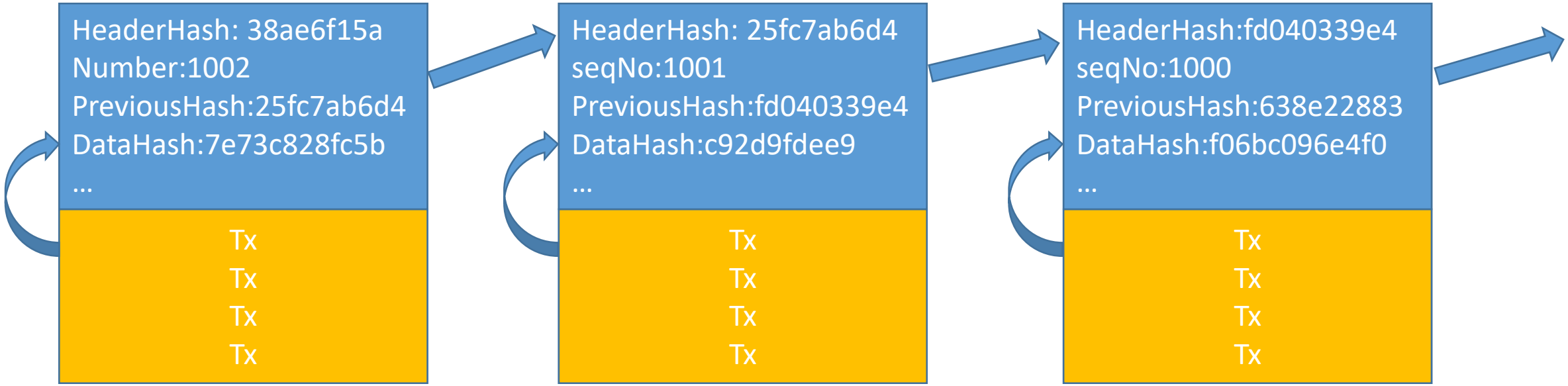
Fabric密码算法应用场景 – 哈希函数

哈希函数

- 区块数据处理
- 签名前置处理
- 产生唯一索引

Fabric密码算法应用场景 – 哈希函数

保证账本数据不可篡改？



Fabric密码算法应用场景 – 加密

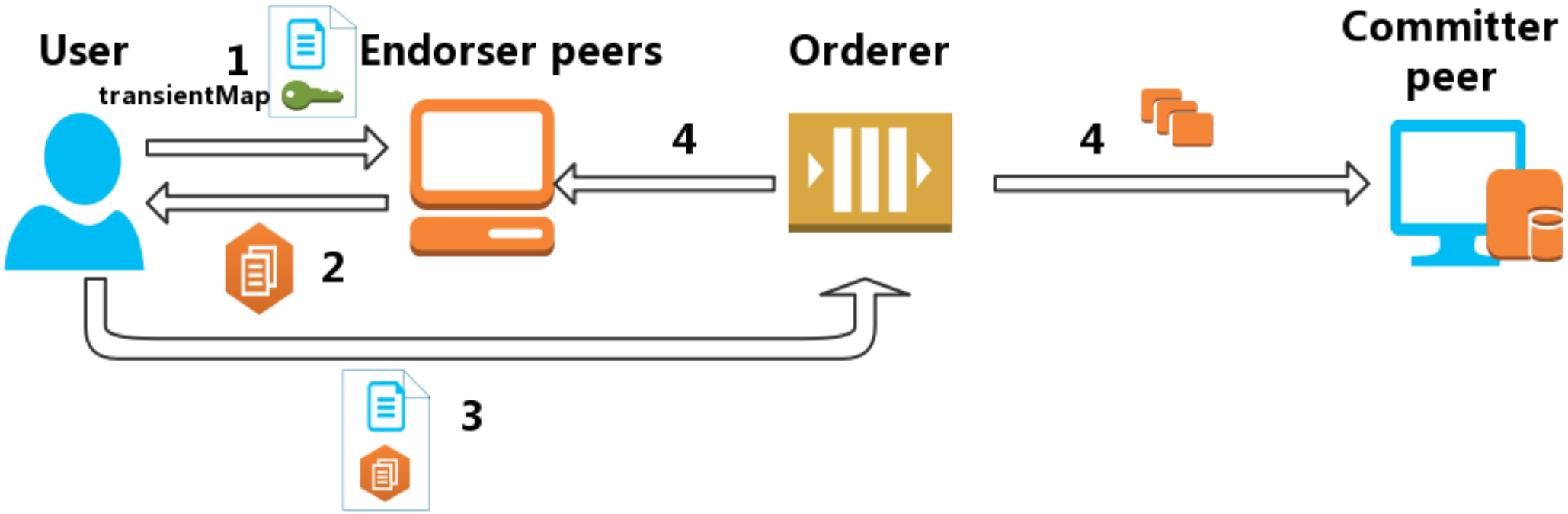
- **加密没有用于Fabric核心**

  通过channel保证账本的数据隐私性

- **在chaincode层面提供应用级加密支持**

  用于channel内部的数据隐私保护

Fabric密码算法应用场景 – 加密



BCCSP介绍 – 设计目标

设计目标：

- BCCSP为Fabric提供统一的密码服务入口
- 提供不同类型的实现方式
- 提供多种类的算法

BCCSP介绍 – 接口

```
type BCCSP interface{
    KeyGen(opts KeyGenOpts) (k Key, err error)
    KeyDeriv(k Key, opts KeyDerivOpts) (dk Key, err error)
    KeyImport(raw interface{}, opts KeyImportOpts) (k Key, err error)
    GetKey(ski []byte) (k Key, err error)
    Hash(msg []byte, opts HashOpts) (hash []byte, err error)
    GetHash(opts HashOpts) (h hash.Hash, err error)
    Sign(k Key, digest []byte, opts SignerOpts) (signature []byte, err error)
    Verify(k Key, signature, digest []byte, opts SignerOpts) (valid bool, err error)
    Encrypt(k Key, plaintext []byte, opts EncrypterOpts) (ciphertext []byte, err error)
    Decrypt(k Key, ciphertext []byte, opts DecrypterOpts) (plaintext []byte, err error)
}
```

Key lifecycle

Crypto operations

BCCSP介绍 – 秘钥管理

KeyGen(opts KeyGenOpts) (k Key, err error)

KeyDeriv(k Key, opts KeyDerivOpts) (dk Key, err error)

KeyImport(raw interface{}, opts KeyImportOpts) (k Key, err error)

GetKey(ski []byte) (k Key, err error)

BCCSP介绍 – 密码操作

Hash(msg []byte, opts HashOpts) (hash []byte, err error)

GetHash(opts HashOpts) (h hash.Hash, err error)

Sign(k Key, digest []byte, opts SignerOpts) (signature []byte, err error)

Verify(k Key, signature, digest []byte, opts SignerOpts) (valid bool, err error)

Encrypt(k Key, plaintext []byte, opts EncrypterOpts) (ciphertext []byte, err error)

Decrypt(k Key, ciphertext []byte, opts DecrypterOpts) (plaintext []byte, err error)

BCCSP介绍 – 实现方案

1. SW(software)

软件实现方案，提供软件算法集

2. PKCS11

硬件对接方案，通过PKCS11对接厂商密码硬件

3. Plugin

插件式BCCSP实现方案

MSP相关

- 基于X509的PKI
- 为每个organization提供自己的MSP功能
- 能够获取channel中其他org的msp，验证从属于其中的实体身份

MSP相关

```
// with, and verifying signatures that correspond to these certificates.///
type Identity interface {
    // ExpiresAt returns the time at which the Identity expires.
    // If the returned time is the zero value, it implies
    // the Identity does not expire, or that its expiration
    // time is unknown
    ExpiresAt() time.Time

    // GetIdentifier returns the identifier of that identity
    GetIdentifier() *IdentityIdentifier

    // GetMSPIdentifier returns the MSP Id for this instance
    GetMSPIdentifier() string
    // Validate uses the rules that govern this identity to validate it.
    // E.g., if it is a fabric TCert implemented as identity, validate
    // will check the TCert signature against the assumed root certificate
    // authority.
    Validate() error

    // GetOrganizationalUnits returns zero or more organization units or
    // divisions this identity is related to as long as this is public
    // information. Certain MSP implementations may use attributes
    // that are publicly associated to this identity, or the identifier of
    // the root certificate authority that has provided signatures on this
    // certificate.
    // Examples:
    //  - if the identity is an x.509 certificate, this function returns one
    //    or more string which is encoded in the Subject's Distinguished Name
    //    of the type OU
    // TODO: For X.509 based identities, check if we need a dedicated type
    //       for OU where the Certificate OU is properly namespaced by the
    //       signer's identity
    GetOrganizationalUnits() []*OUIdentifier

    // Verify a signature over some message using this identity as reference
    Verify(msg []byte, sig []byte) error

    // Serialize converts an identity to bytes
    Serialize() ([]byte, error)

    // SatisfiesPrincipal checks whether this instance matches
    // the description supplied in MSPPrincipal. The check may
    // involve a byte-by-byte comparison (if the principal is
    // a serialized identity) or may require MSP validation
    SatisfiesPrincipal(principal *msp.MSPPrincipal) error
}
```

MSP相关

```go
// MSP is the minimal Membership Service Provider Interface to be implemented
// to accommodate peer functionality
type MSP interface {

    // IdentityDeserializer interface needs to be implemented by MSP
    IdentityDeserializer

    // Setup the MSP instance according to configuration information
    Setup(config *msp.MSPConfig) error

    // GetVersion returns the version of this MSP
    GetVersion() MSPVersion

    // GetType returns the provider type
    GetType() ProviderType

    // GetIdentifier returns the provider identifier
    GetIdentifier() (string, error)

    // GetSigningIdentity returns a signing identity corresponding to the provided identifier
    GetSigningIdentity(identifier *IdentityIdentifier) (SigningIdentity, error)

    // GetDefaultSigningIdentity returns the default signing identity
    GetDefaultSigningIdentity() (SigningIdentity, error)

    // GetTLSRootCerts returns the TLS root certificates for this MSP
    GetTLSRootCerts() [][]byte

    // GetTLSIntermediateCerts returns the TLS intermediate root certificates for this MSP
    GetTLSIntermediateCerts() [][]byte

    // Validate checks whether the supplied identity is valid
    Validate(id Identity) error

    // SatisfiesPrincipal checks whether the identity matches
    // the description supplied in MSPPrincipal. The check may
    // involve a byte-by-byte comparison (if the principal is
    // a serialized identity) or may require MSP validation
    SatisfiesPrincipal(id Identity, principal *msp.MSPPrincipal) error
}
```

国密算法支持 – 简介

国密算法是国家商用密码算法的简称

自2012年以来，国家密码管理局以《中华人民共和国密码行业标准》的方式，
陆续公布了SM2/SM3/SM4等密码算法标准及其应用规范

其中“SM”代表“商密”，即用于商用的、不涉及国家秘密的密码技术。

SM2为基于椭圆曲线密码的公钥密码算法标准，包含数字签名、密钥交换和公钥加密，
用于替换RSA/Diffie-Hellman/ECDSA/ECDH等国际算法

SM3为密码哈希算法，用于替代MD5/SHA-1/SHA-256等国际算法

SM4为分组密码，用于替代DES/AES等国际算法

国密算法支持 – 算法实现方式

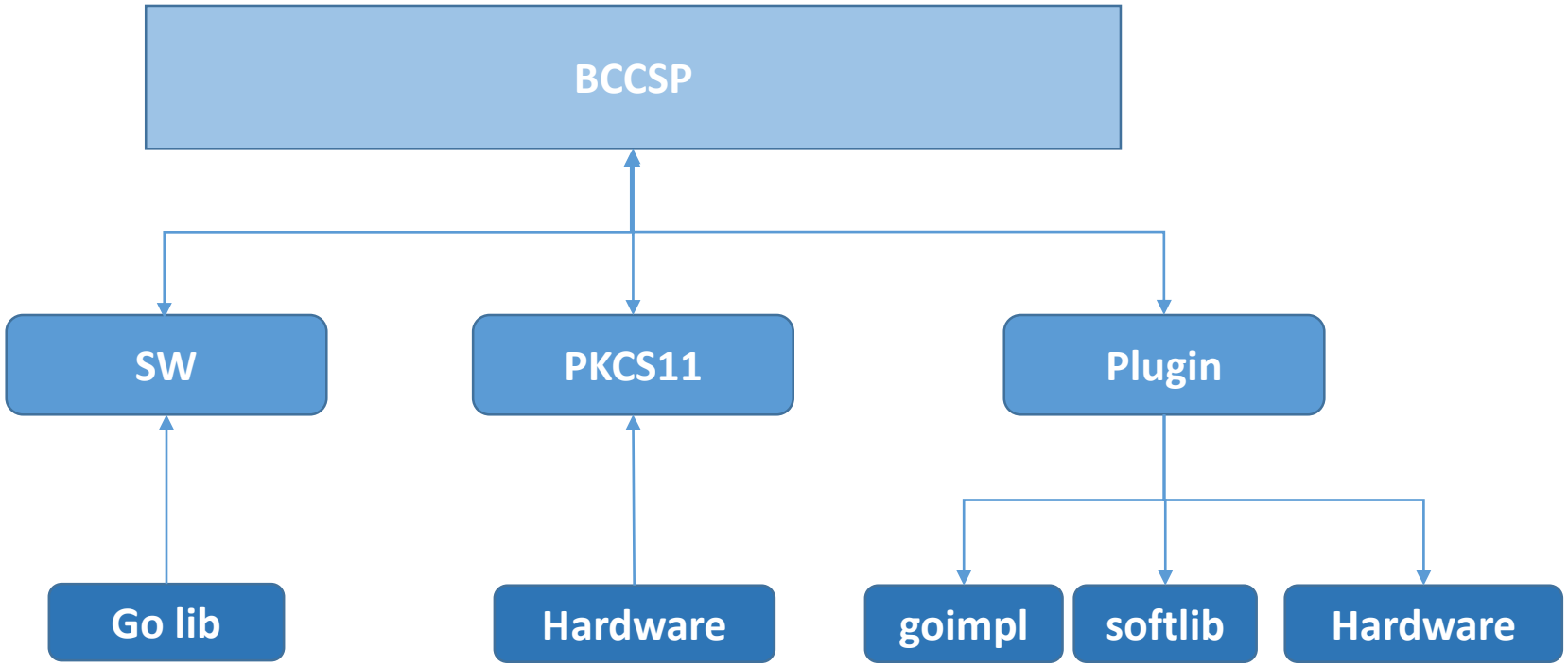
- 软件

  c、go、java、gmssl

- 硬件

  PCIE密码卡、密码机、USBKEY

国密算法支持 – BCCSP实现



国密算法支持 – MSP支持

Trouble:

基于X509 现阶段不支持国密

Alternative solutions:

- 在go lib层面做支持
- 改现有的msp impl
- 重新设计新msp impl
- Msp plugin方式

国密算法支持

- 工具链支持
- CA支持
- SDK支持

END

THANKS