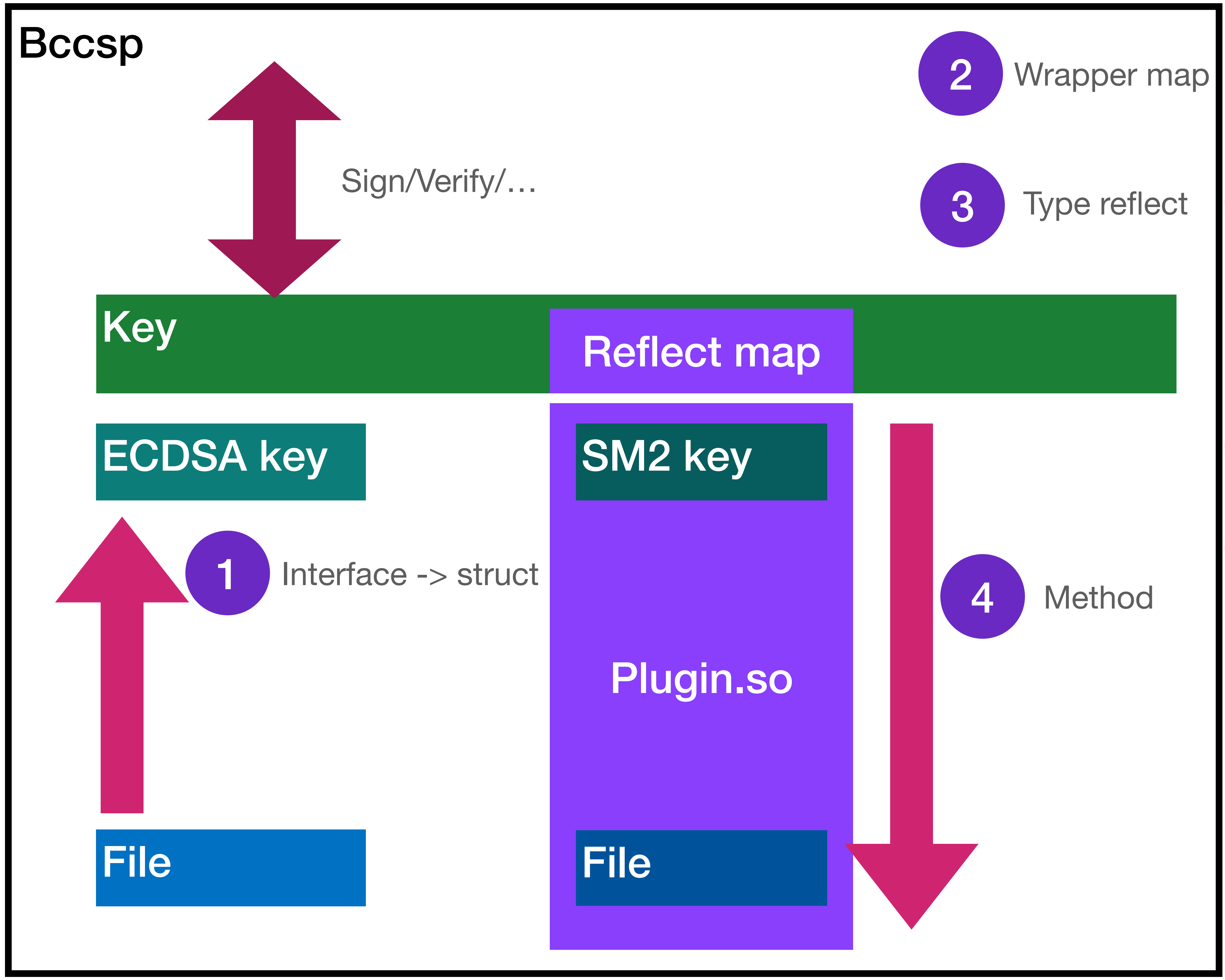
Modular Crypto Service

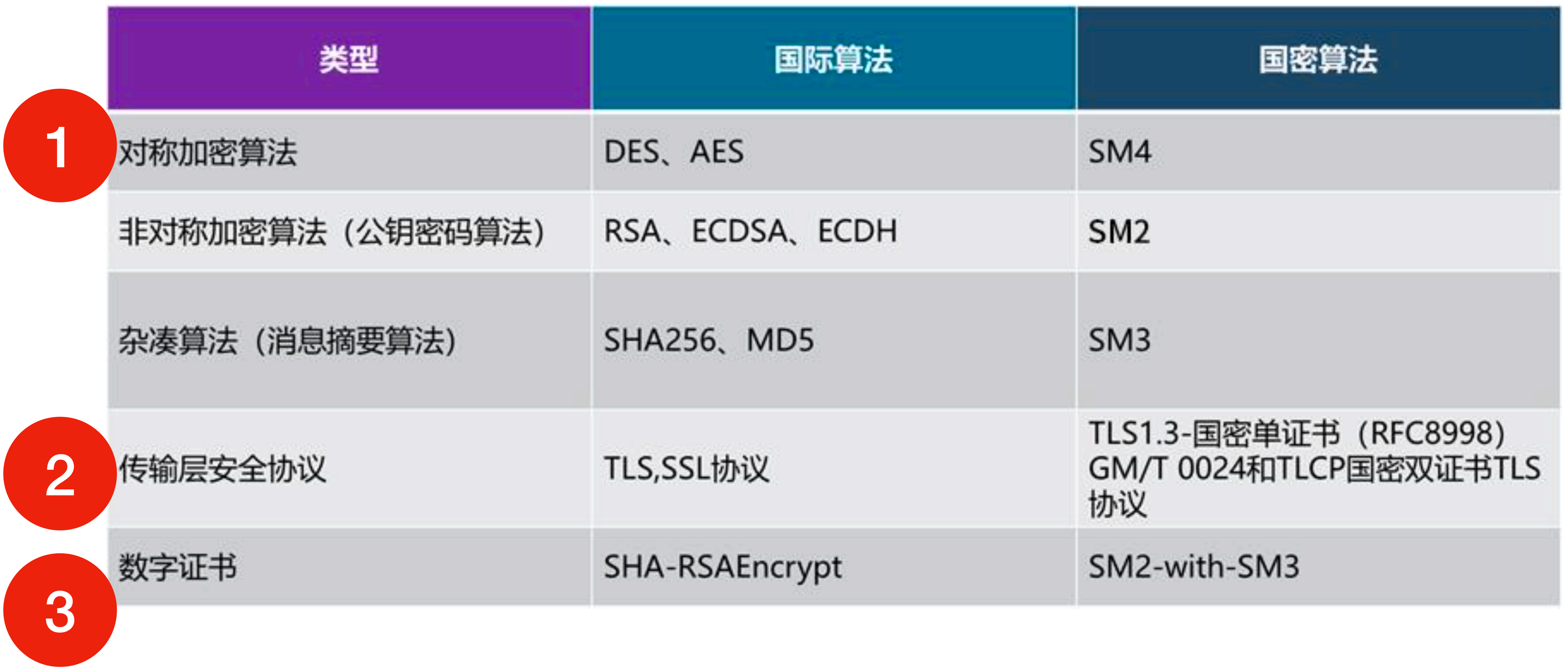
https://github.com/hyperledger/fabric-rfcs/pull/34

Modular Crypto Service

Back ground & Today just for no TLS part



国密算法与国际算法对比

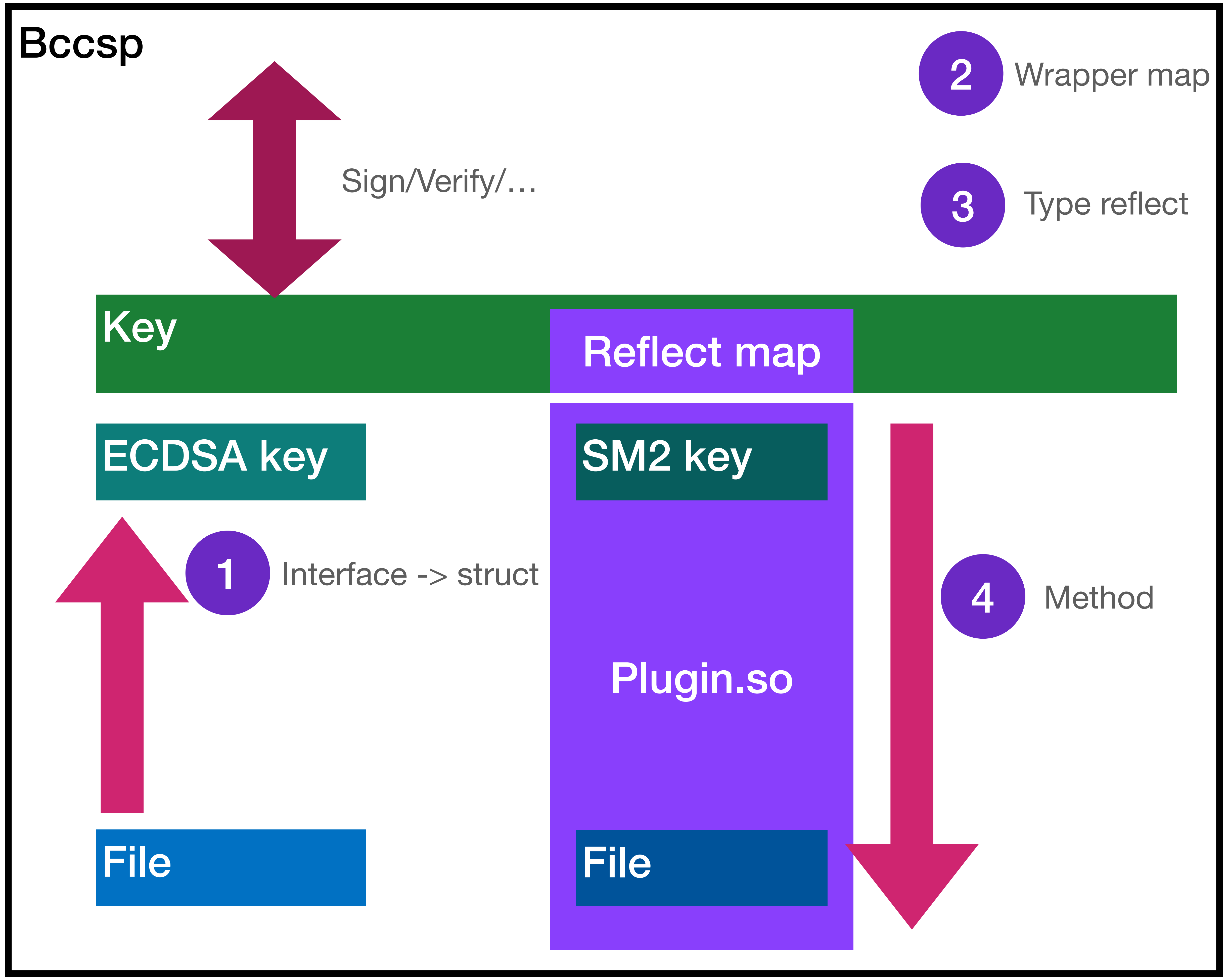


类型	国际算法	国密算法
对称加密算法	DES、AES	SM4
非对称加密算法（公钥密码算法）	RSA、ECDSA、ECDH	SM2
杂凑算法（消息摘要算法）	SHA256、MD5	SM3
传输层安全协议	TLS,SSL协议	TLS1.3-国密单证书（RFC8998） GM/T 0024和TLCP国密双证书TLS协议
数字证书	SHA-RSAEncrypt	SM2-with-SM3

Modular Crypto Service

High level design

- Behavioral Pattern
  - Strategy Pattern
- Structure Pattern
  - Prototype Pattern
  - Adapter Pattern
  - …



Bccsp

Sign/Verify/…

2 Wrapper map

3 Type reflect

Key

Reflect map

ECDSA key

SM2 key

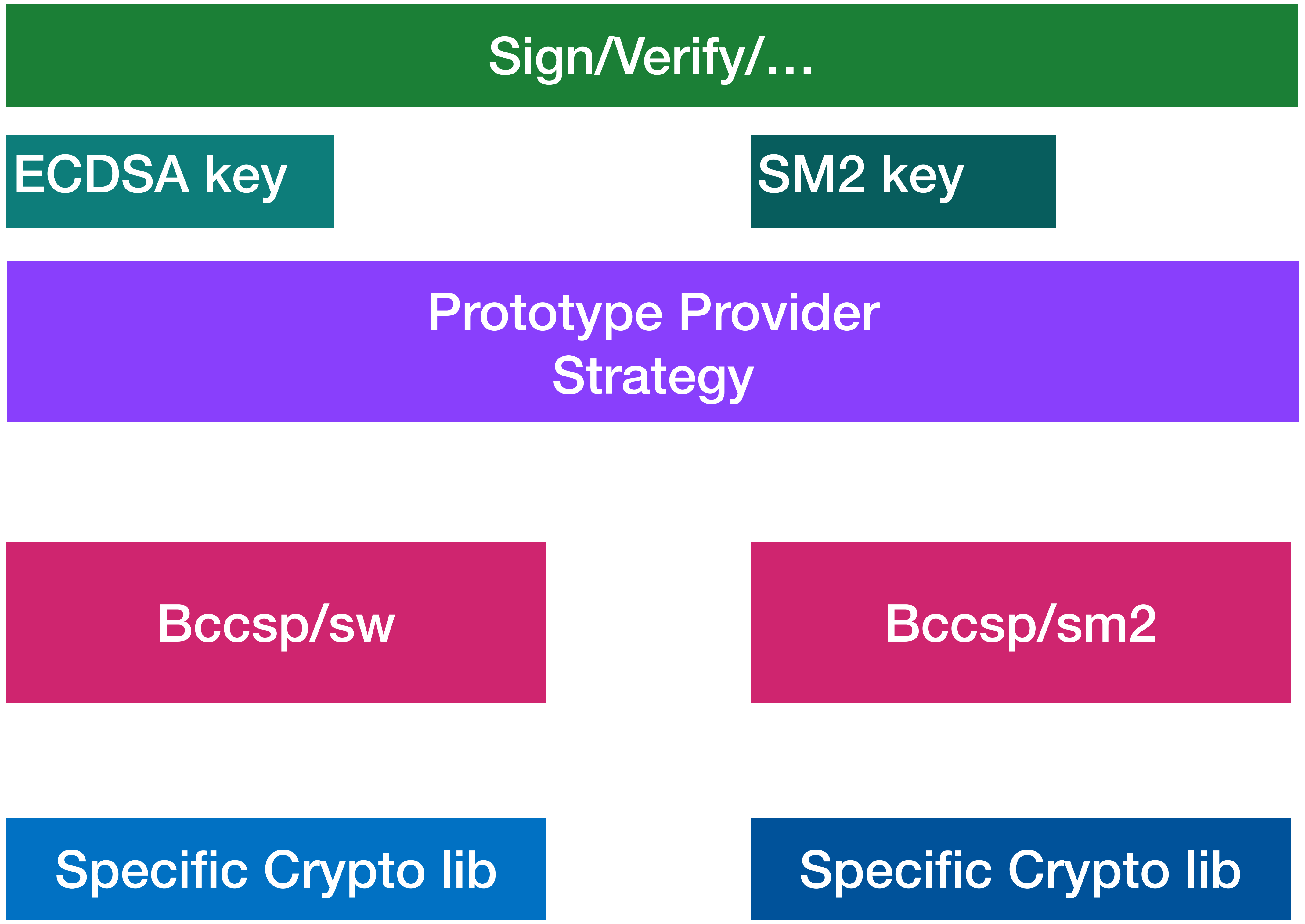
1 Interface -> struct

4 Method

Plugin.so

File

File



Sign/Verify/…

ECDSA key

SM2 key

Prototype Provider
Strategy

Bccsp/sw

Bccsp/sm2

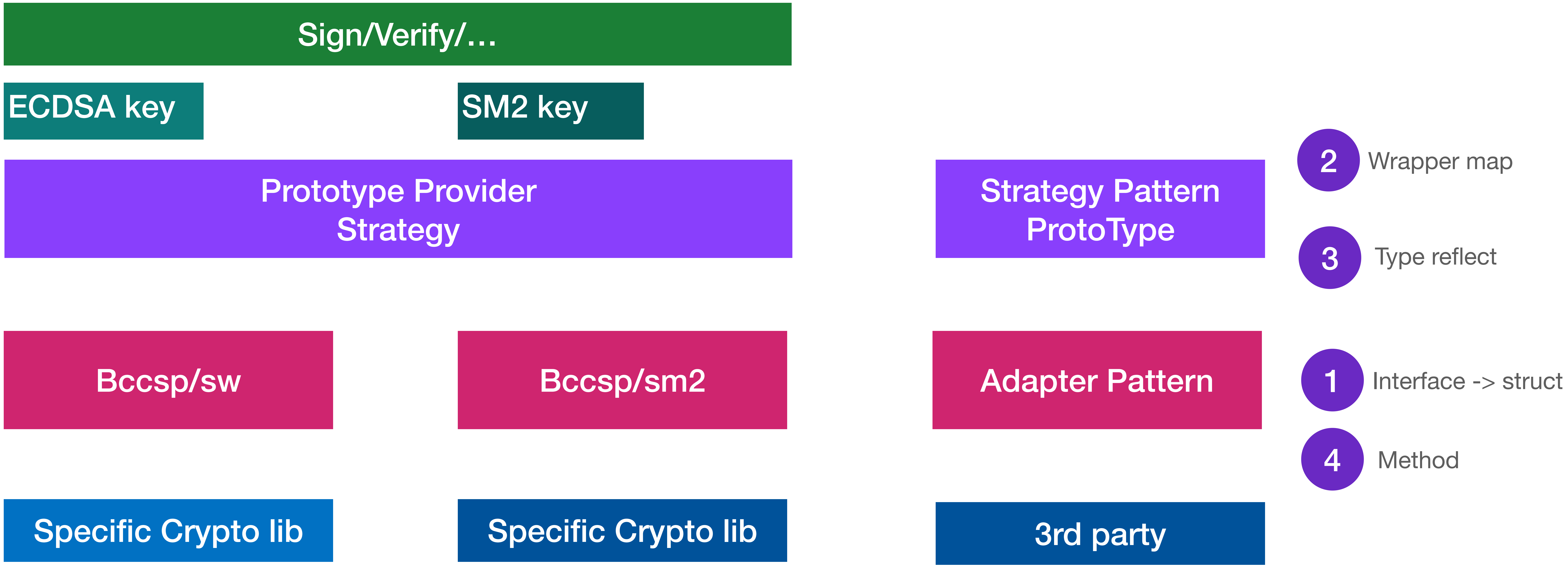
Specific Crypto lib

Specific Crypto lib

Modular Crypto Service

High level design

- Behavioral Pattern
  - Strategy Pattern
- Structure Pattern
  - Prototype Pattern
  - Adapter Pattern
  - …



Sign/Verify/…

ECDSA key

SM2 key

Prototype Provider
Strategy

Strategy Pattern
ProtoType

2 Wrapper map

3 Type reflect

Bccsp/sw

Bccsp/sm2

Adapter Pattern

1 Interface -> struct

4 Method

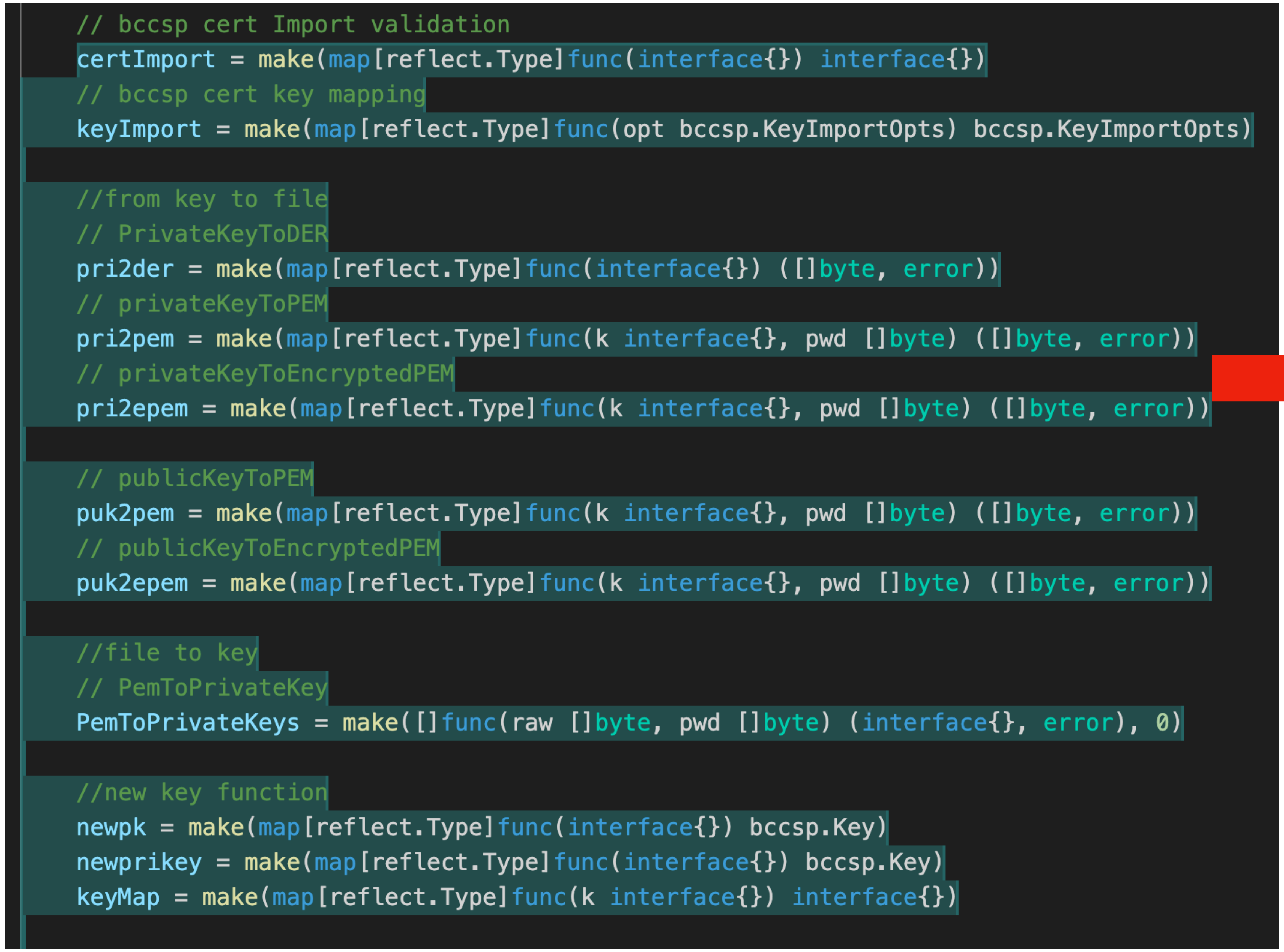
Specific Crypto lib

Specific Crypto lib

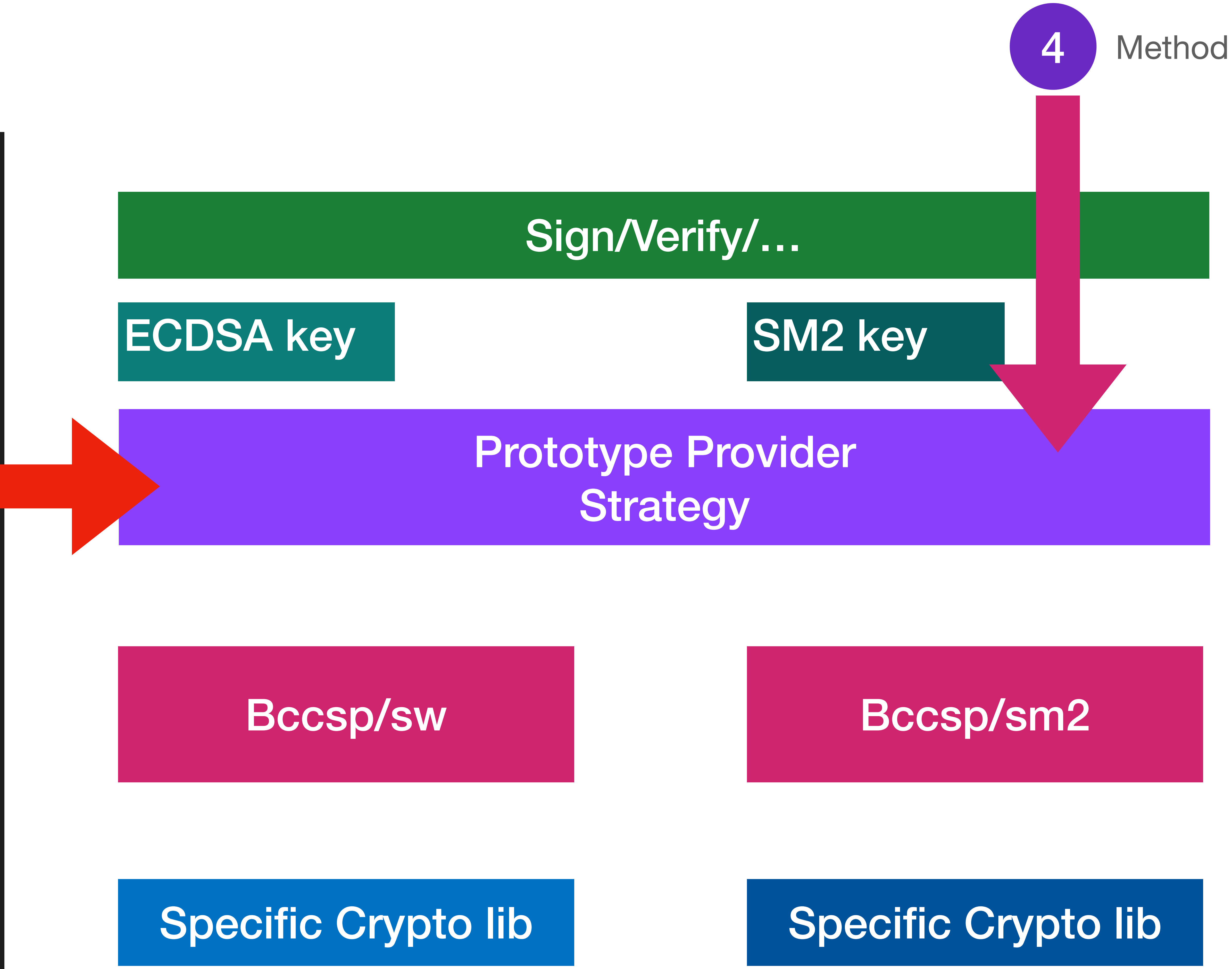
3rd party

Modular Crypto Service

The provider map

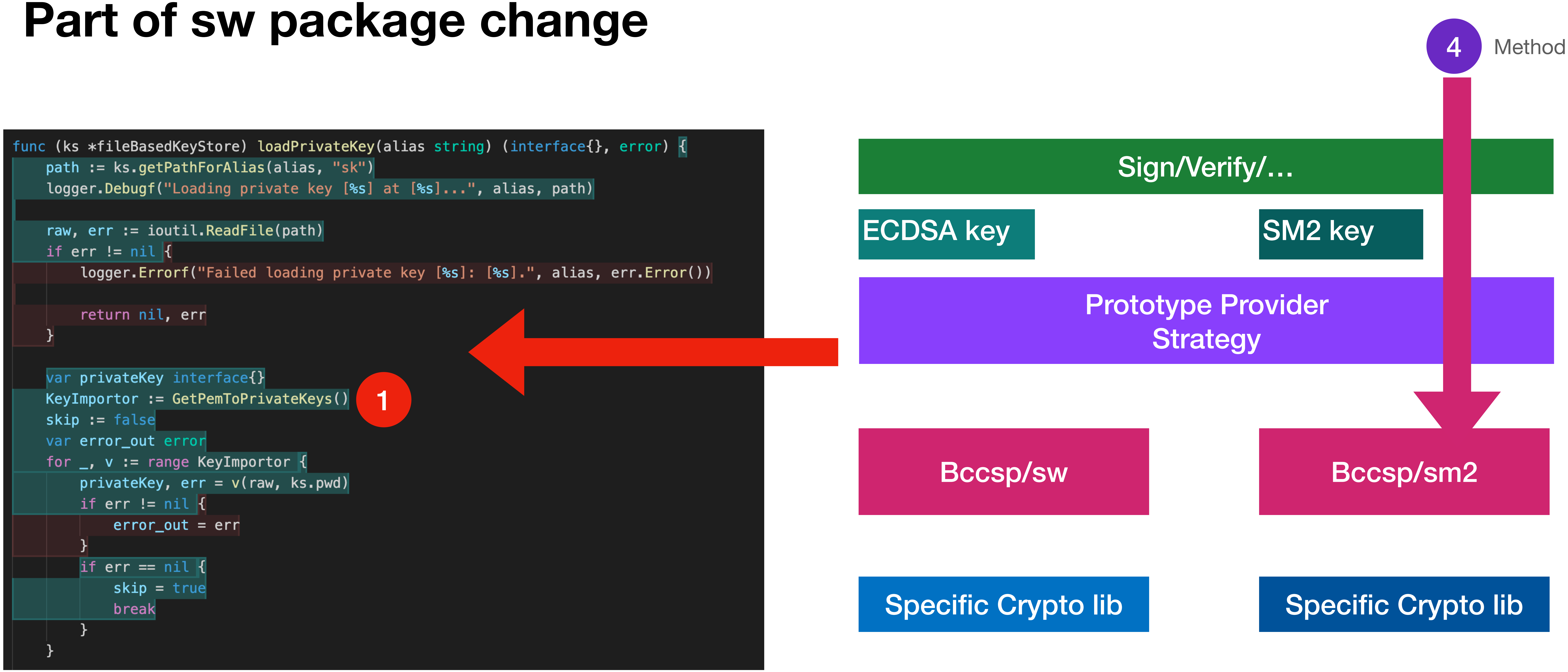


```
// bccsp cert Import validation
certImport = make(map[reflect.Type]func(interface{}) interface{})
// bccsp cert key mapping
keyImport = make(map[reflect.Type]func(opt bccsp.KeyImportOpts) bccsp.KeyImportOpts)

//from key to file
// PrivateKeyToDER
pri2der = make(map[reflect.Type]func(interface{}) ([]byte, error))
// privateKeyToPEM
pri2pem = make(map[reflect.Type]func(k interface{}, pwd []byte) ([]byte, error))
// privateKeyToEncryptedPEM
pri2epem = make(map[reflect.Type]func(k interface{}, pwd []byte) ([]byte, error))

// publicKeyToPEM
puk2pem = make(map[reflect.Type]func(k interface{}, pwd []byte) ([]byte, error))
// publicKeyToEncryptedPEM
puk2epem = make(map[reflect.Type]func(k interface{}, pwd []byte) ([]byte, error))

//file to key
// PemToPrivateKey
PemToPrivateKeys = make([]func(raw []byte, pwd []byte) (interface{}, error), 0)

//new key function
newpk = make(map[reflect.Type]func(interface{}) bccsp.Key)
newprikey = make(map[reflect.Type]func(interface{}) bccsp.Key)
keyMap = make(map[reflect.Type]func(k interface{}) interface{})
```



Modular Crypto Service

Part of sw package change



```
func (ks *fileBasedKeyStore) loadPrivateKey(alias string) (interface{}, error) {
    path := ks.getPathForAlias(alias, "sk")
    logger.Debugf("Loading private key [%s] at [%s]...", alias, path)

    raw, err := ioutil.ReadFile(path)
    if err != nil {
        logger.Errorf("Failed loading private key [%s]: [%s].", alias, err.Error())

        return nil, err
    }

    var privateKey interface{}
    KeyImportor := GetPemToPrivateKeys()
    skip := false
    var error_out error
    for _, v := range KeyImportor {
        privateKey, err = v(raw, ks.pwd)
        if err != nil {
            error_out = err
        }
        if err == nil {
            skip = true
            break
        }
    }
```

Modular Crypto Service

Discussion

Strategy Pattern ProtoType

Adapter Pattern

3rd party

	Go Plugin	Hardcode(build tag)	3rd party lib(proto)
Advantage	Less change in fabric	Go plugin limitation	
Disadvatage	Go plugin limitation	Less change in fabric	
Comments	From SW design considering, we are expected to remove hard code but configurable		