Kryptografie na aplikační vrstvě

streamové šifrování (SSL, IpSec, ...) nezajišťuje přístup k vlastním datům, tudíž nelze využít zavedenou sémantiku provedených kryptografických transformací

- zejména nepopiratelnost
- anonymita příjemce
- ...

Prokazatelnost operace a z ní plynoucí sémantika

- prováděné na serveru – utajení, integrita, původ, nikdy nepopiratelnost
- prováděné přímo na pracovní stanici, nebo tokenem uživatele – jako předchozí, navíc nepopiratelnost, soulad s požadavku ZEP na (zaručený) el. podpis, ...

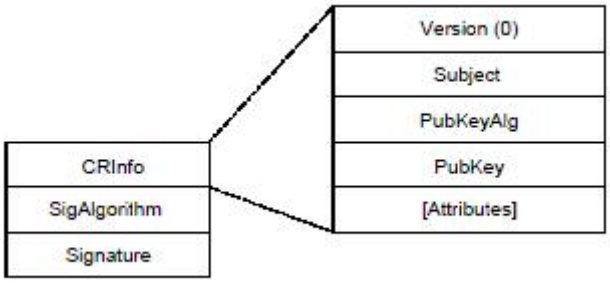
Základním problémem je uložení klíčů

- cryptoapi – jednoduché, levné, minimálně pružné, velmi málo bezpečné
- soubory – výrazně mobilnější
- karty a jiné tokeny – bezpečné, mobilní, dosud drahé

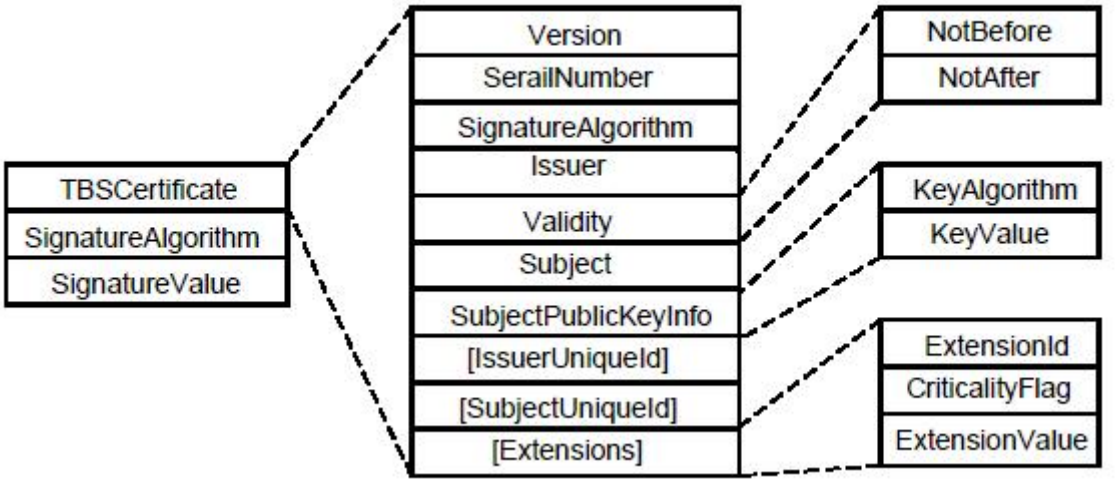
Kryptografické knihovny

- nativní kód daného OS – rychlé, problémy s instalací
- Java – portabilní, lze spustit i na stanici, kde uživatel není administrátor

PKCS10 – certificate signing request



X.509 Certifikát

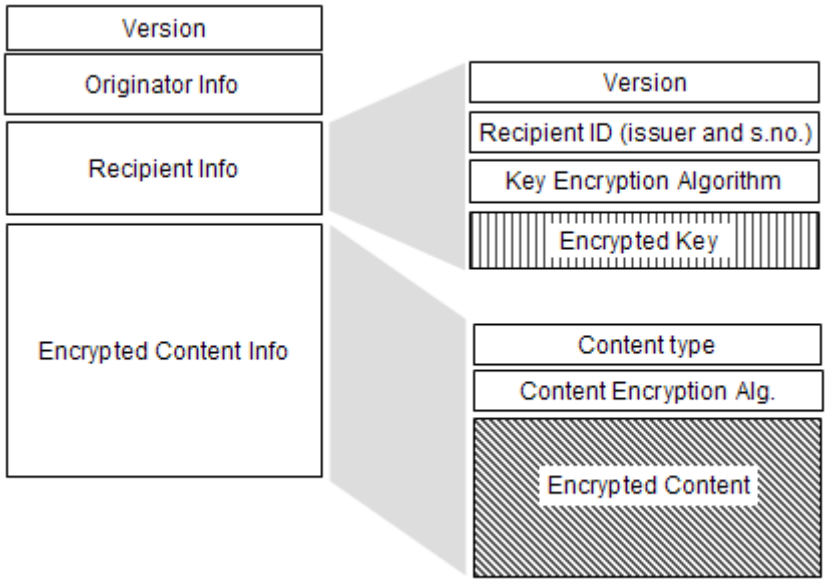


PKCS #7

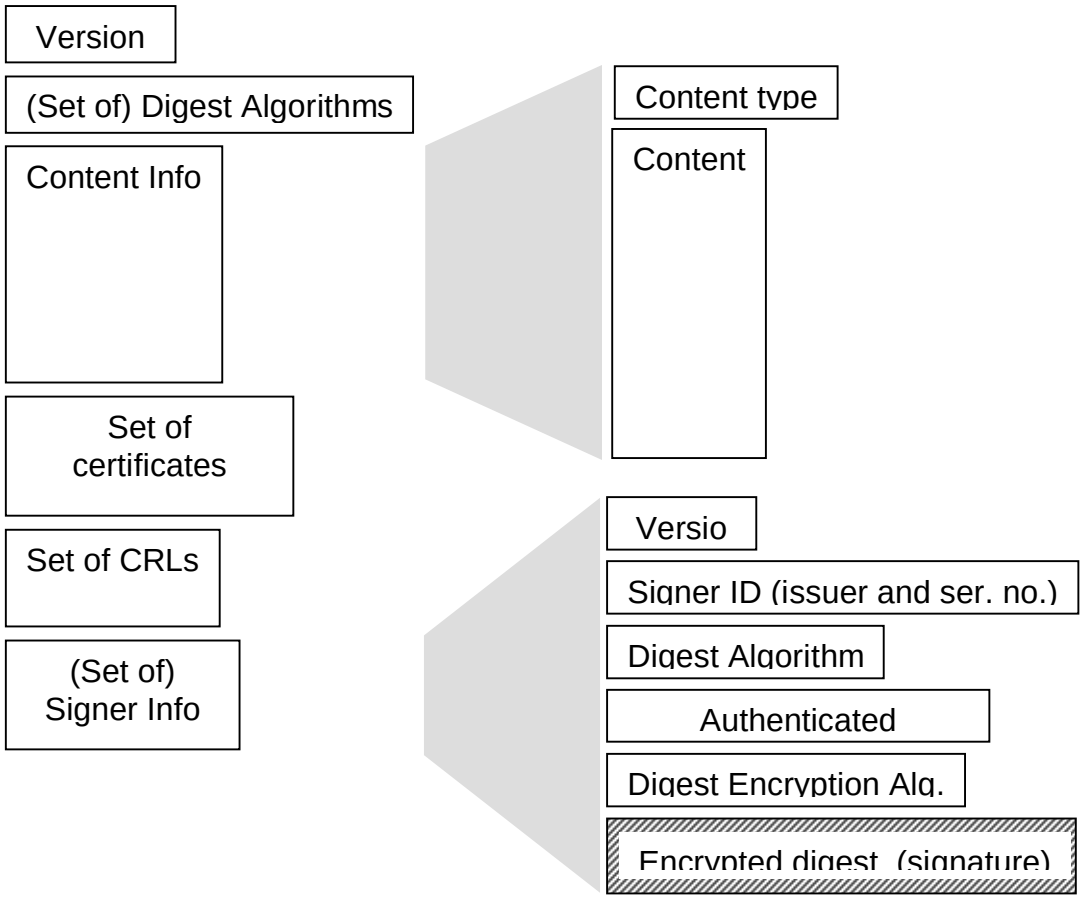
Dnes pravděpodobně daleko nejrozšířenější formát kryptografických dat
Obecně pracuje tak, že vezme data a bez jakékoliv větší interpretace je „obalí“ nějakou kryptografickou transformací
je základem pro S/MIME, PEM, ... v podstatě i XML
Definuje šest formátů zpráv:

- Data – oktet string
- Signed Data – data s připojeným podpisem spočteným nad hashem a odpovídajícím CRT
- Enveloped data – sealovaná data veřejným klíčem (více) příjemců
- Signed-and-enveloped data – podepsaná data následně sealovaná
- Digested data – hash nad daty
- Encrypted data – symetrické šifrování, neřeší se key mamagement klíčů

Enveloped data



Signed data



Signed and enveloped data

složení obou předchozích formátů

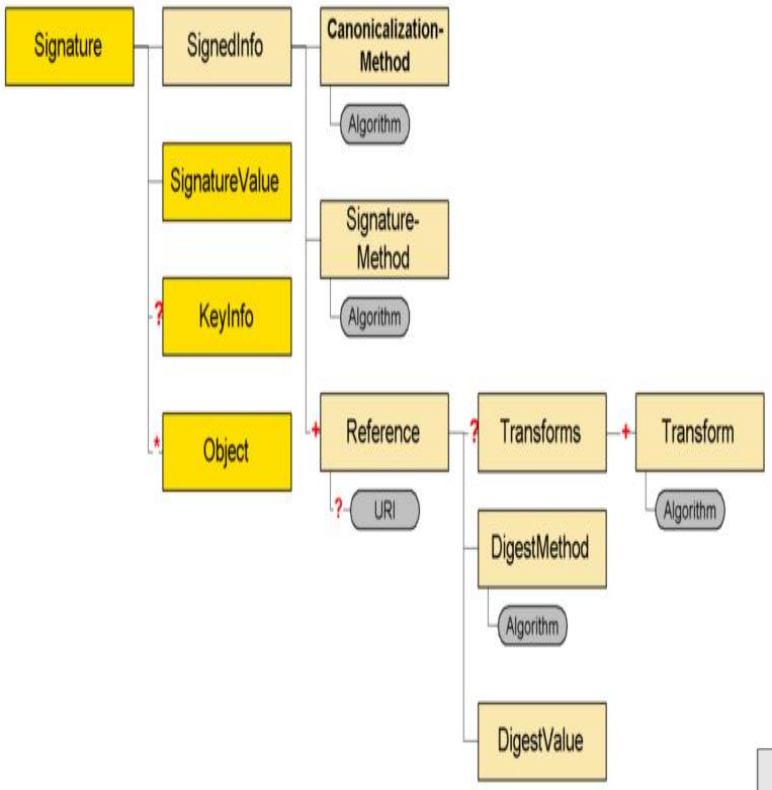
```
SignedAndEnvelopedData ::= SEQUENCE {
   • version Version,
   • recipientInfos RecipientInfos,
   • digestAlgorithms DigestAlgorithmIdentifiers,
   • encryptedContentInfo EncryptedContentInfo,
   • certificates
        [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
   • crls
       [1] IMPLICIT CertificateRevocationLists OPTIONAL,
   • signerInfos SignerInfos
}
```

Zbylé formáty jsou v podstatě triviální

XML signature/encryption

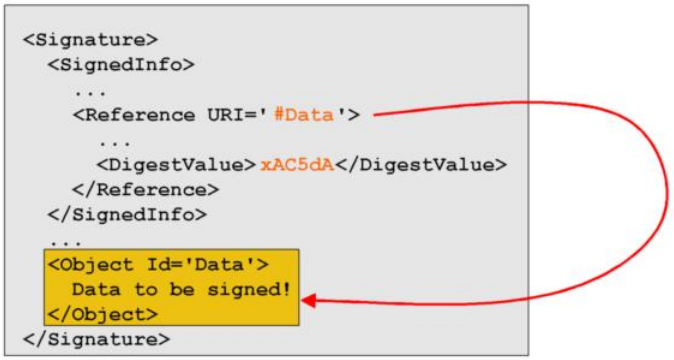
v podstatě jde o PKCS#7 objekty řízené nástroji xml – zejména XSLT a Xpath kryptografické transformace jsou aplikovány s „porozuměním“ pro strukturu dokumentu pouze na relevantní části, vlastní výsledek transformace je součástí dokumentu

Struktura podpisu



Druhy podpisů

definují se celkem tři druhy podpisů:



- enveloping signature

podepsaný dokument je součástí podpisu (podobné, jako PKCS#7)

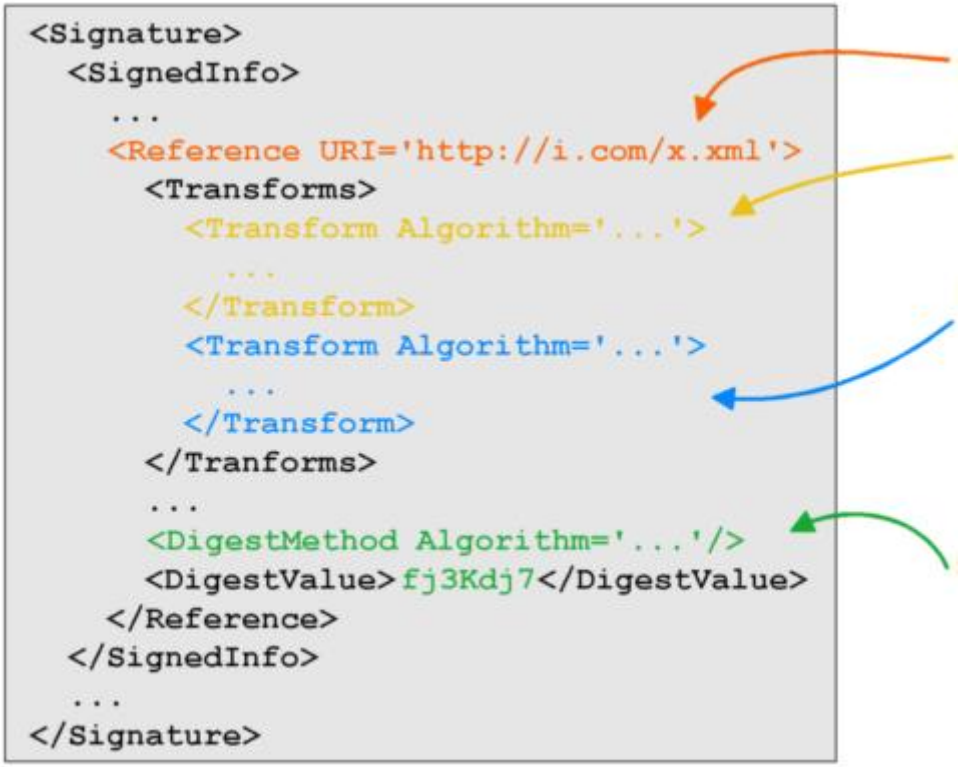
- enveloped signature

podpis je součástí dokumentu, který je podepisován

- detached signature

podpis se vztahuje k datům, která nesou součástí dokumentu, ve kterém se podpis nachází

Celková strategie provedení podpisu



- načtení dat z místa určeného referencí
- načtená data jsou podrobena první transformace
- výsledek předchozí transformace je vstupem následující transformace
- výsledek poslední transformace je podroben hashování

Pravidla tvorby podpisu

```
<Signature>
  <SignedInfo>
    <CanonicalizationMethod
      URI='...'/>
    <SignatureMethod URI='...'/>
    <Reference URI='#Data'>
      <Transforms>
        <Transform URI='...'>
          ...
        </Transfrom>
      <Transforms>
      <DigestMethod URI='...'/>
      <DigestValue>
        ...
      </DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>
    ...
  </SignatureValue>
  <KeyInfo>...</KeyInfo>
  <Object Id='Data'>
    Data being signed!
  </Object>
</Signature>
```

Každá podepisovaná data:

- Aplikuj Transforms
- Z výsledku spočítej hash
- Vytvoř Reference klauzuli obsahující **Transforms**, **DigestMethod** a **DigestValue**

- Vytvoř **SignedInfo** klauzuli s **SignatureMethod**, **CanonicalizationMethod** a **Reference**
- Kanonizuj **SignedInfo** a spočítej hodnotu podpisu
- Sestav **Signature** obsahující **SignedInfo**, **SignatureValue** a případně **KeyInfo** a **Object**

XML encryption

Podobně jako v případě XML signature, nejprve se XSLT, XPATH a podobnou transformací vybere příslušná část původního formuláře a ta se šifruje.
Šifrovat je možné

- element
- obsah elementu
- celý dokument (jako oktet string)

```
<?xml version="1.0"?>
<!DOCTYPE customer_order SYSTEM "custord.dtd">
<customer_order>
  <items>
    <item>
      <name>Turnip Twaddler</name>
      <qty>3</qty>
      <price>9.95</price>
    </item>
    <item>
      <name>Snipe Curdler</name>
      <qty>1</qty>
      <price>19.95</price>
    </item>
  </items>
```

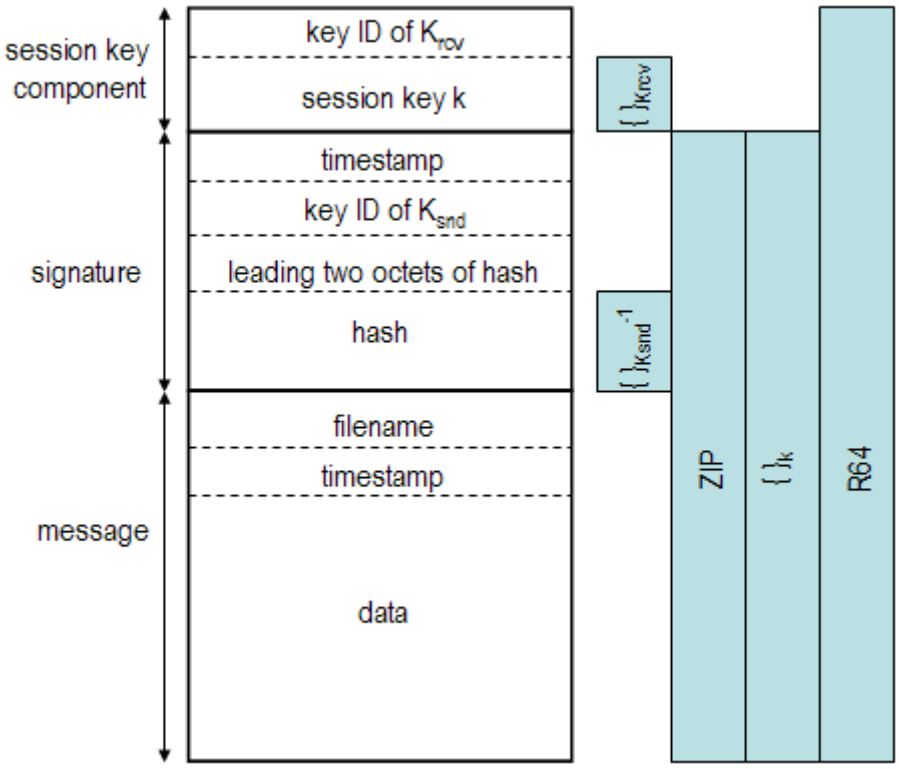
```
  <customer>
    <name>Doug Tidwell</name>
    <street>1234 Main Street</street>
    <city state="NC">Raleigh</city>
    <zip>11111</zip>
  </customer>
  <credit_payment>
    <card_issuer>American Express</card_issuer>
    <card_number>1234 567890 12345</card_number>
    <expiration_date month="10" year="2004"/>
  </credit_payment>
</customer_order>
```

```
<?xml version="1.0"?><customer_order>
  <items>
    <item>
      <name>Turnip Twaddler</name>
      <qty>3</qty>
      <price>9.95</price>
    </item>
    <item>
      <name>Snipe Curdler</name>
      <qty>1</qty>
      <price>19.95</price>
    </item>
  </items>
  <customer>
    <name>Doug Tidwell</name>
    <street>1234 Main Street</street>
    <city state="NC">Raleigh</city>
    <zip>11111</zip>
  </customer>
  <EncryptedElement algorithm="DES/CBC/PKCS5Padding" contentType="text/xml"
  encoding="base64" iv="S5Rirg//pNQ=">vJqNpDrQT1vmCVbyGJfIwdIDBYoGXGmutgz6TVGoPuKVG7I
  xNEN50iKw8pmtxFixz5hOChOXgTtPqktQhEHO5+vLOLAFgIioDIRQGHHmHng3CLd+8tvrT8wxPBCRSMUpx4
  d2TGXW2tqSepam0ZxdmwUXwNSAgaR8hmiromD+bh+tDomPv7eFZ4no5ft3JG3t0trLlwVupF/5vaIJimUSm
  uUkkgyG8x9AcS/kXJxHpmM=peqGzIMf+8A=</EncryptedElement>
</customer_order>
```

PGP

Nápad Philla Zimmermanna, původně opensource, dnes vlastní Computer Associates, celá řada free reimplementací, nejlépe GPG

Formát zprávy v PGP formátování



S/MIME

založeno na PKCS#7 datových zprávách
rozšíření MIME formátu o kryptografické datové typy:

- šifrovaná obálka (application/pkcs7-mime; smime-type = enveloped-data)
  - zašifrovaná (sealing) data
- podepsaná data (application/pkcs7-mime; smime-type = signed-data)
  - digitální podpis (“hashování a podpis”)
  - data + podpis kódována do base64
- čitelná podepsaná data (multipart/signed)
  - digitální podpis
  - jen vlastní podpis je kódován do base64
  - příjemce bez podpory S/MIME může aspoň číst
- podepsaná a zašifrovaná data
  - podepsané a zašifrované bloky dat lze libovolně hnízdit

Definovaná kryptografie

- digest funkce
  - povinně: SHA-1
  - doporučeno (příjemce): MD5 (zpětná kompatibilita)

- digitální podpis
  - povinně: DSS (DSA)
  - doporučeno: RSA
- asymmetrické šifrování
  - povinně: ElGamal
  - doporučeno: RSA
- symmetrické šifrování
  - odesílatel:
    - doporučeno: 3DES, RC2/40
  - příjemce:
    - povinně: 3DES
    - doporučeno: RC2/40

Toolkity

- OpenSSL
- Crypto++
- CryptoLib
- XMLSec
- Baltimore
- Entrust
- IBM XML Security Suite
- Phaos
- RSAREF

Použití kryptografických transformací v praxi

CryptoApi

CALG_3DES	Triple DES encryption algorithm.
CALG_3DES_112	Two-key triple DES encryption.
CALG_DES	DES encryption algorithm.
CALG_HMAC*	MAC keyed-hash algorithm.
CALG_MAC*	MAC keyed-hash algorithm.
CALG_MD2*	MD2 hashing algorithm.
CALG_MD4	MD4 hashing algorithm.
CALG_MD5*	MD5 hashing algorithm.
CALG_RC2*	RC2 block encryption algorithm.
CALG_RC4*	RC4 stream encryption algorithm.

CALG_RC5	RC5 block encryption algorithm.
CALG_RSA_KEYX*	RSA public-key exchange algorithm.
CALG_RSA_SIGN*	RSA public-key signature algorithm.
CALG_SHA*	SHA hashing algorithm.
CALG_SHA1*	Same as CALG_SHA.
CALG_SSL3_SHAMD5	SLL3 client authentication algorithmUsed by the schannel.dll operations system. This ALG_ID should not be used by applications.

*

SSL

pokud server poskytuje RSA certifikát pro výměnu klíčů

```
SSL_RSA_WITH_NULL_MD5
SSL_RSA_WITH_NULL_SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
SSL_RSA_WITH_IDEA_CBC_SHA
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_RSA_WITH_DES_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA
```

Pro server-authenticated (a případně client-authenticated) Diffie-Hellman. *DH* znamena použití certifikátu s Diffie-Hellman parametery podepsanými autoritou. *DHE* značí ephemeral Diffie-Hellman, kde Diffie-Hellman parametery jsou podepsány DSS nebo RSA certifikátem podepsaným autoritou.

```
SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_DSS_WITH_DES_CBC_SHA
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA
SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_RSA_WITH_DES_CBC_SHA
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
```

Pro anonymní Diffie-Hellman komunikaci bez autentizace žádné ze stran.

```
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
SSL_DH_anon_WITH_RC4_128_MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_anon_WITH_DES_CBC_SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
```

Použití Fortezzy

```
SSL_FORTEZZA_DMS_WITH_NULL_SHA
SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA
```

OpenSSL

SYMMETRIC CIPHERS
blowfish, *cast*, des, *idea*, *rc2*, rc4, *rc5*,*rijndael (aes)*
PUBLIC KEY CRYPTOGRAPHY AND KEY AGREEMENT

dsa(3), dh(3), rsa(3)
CERTIFICATES
x509(3), *x509v3(3)*
AUTHENTICATION CODES, HASH FUNCTIONS
hmac(3), md2(3), md4(3), md5(3), mdc2(3), ripemd(3), sha(3)

F-secure SSH

Algorithms

- AES
- 3DES
- Blowfish
- DES
- MD5
- SHA-1
- RSA
- DSA
- Diffie-Hellman

Authentication

- Traditional password
- User public key (RSA and DSA)
- User-key generation support
- PAM (F-Secure proprietary implementation)
- PGP keys
- Kerberos
- NIS and NIS+ environment
- RSA SecurID (F-Secure proprietary implementation)
- Host-based authentication
- Custom plugins
- RADIUS
- Keyboard-interactive PAM
- Keyboard-interactive RSA SecurID
- x.509
- LDAP/Active Directory
- PKCS #12
- CMPv2 enrollment

- PKCS #11
- PKCS #7
- SSH authentication agent
- Agent forwarding

Checkpoint VPN-1

Encryption Algorithm

- Rijndael (Advanced Encryption Standard - AES) * 128- and 256-bit
- Triple DES* 168-bit DES 56-bit DES-40* 40-bit
- CAST-40* 40-bit

User Authentication

- X.509 Digital Certificates
- Pre-shared Secret Hybrid Mode IKE *
- RADIUS TACACS/TACACS+
- Token-based (two-factor)
- Operating System Password
- FireWall-1 Password
- S/Key

Public Key Algorithms

- RSA 512- to 1536-bit*
- Diffie-Hellman 512- to 1536-bit*
- Key Management
- IKE (ISAKMP/Oakley)