

OpenSSL mit Delphi

Max Kleiner

http://max.kleiner.com/download/openssl_opengl.pdf



OpenSSL is an org

<http://www.openssl.org>

free library providing cryptographic functions

it's not the only one, alternatives: Crypto++ and
Cryptlib of Peter Guttman

the important feature is the complete
implementation of the protocols/handshaking
of SSLv2, SSLv3 and TLSv1

Der Computer arbeitet deshalb so schnell, weil er nicht denkt.

Achtung: Lesen kann ihre Dummheit gefährden

Lang ist der Weg durch Lehren, kurz und wirksam durch Beispiele

Delphi 2007: eine IDE mit dramatischer Tiefe und gutem Rückhalt

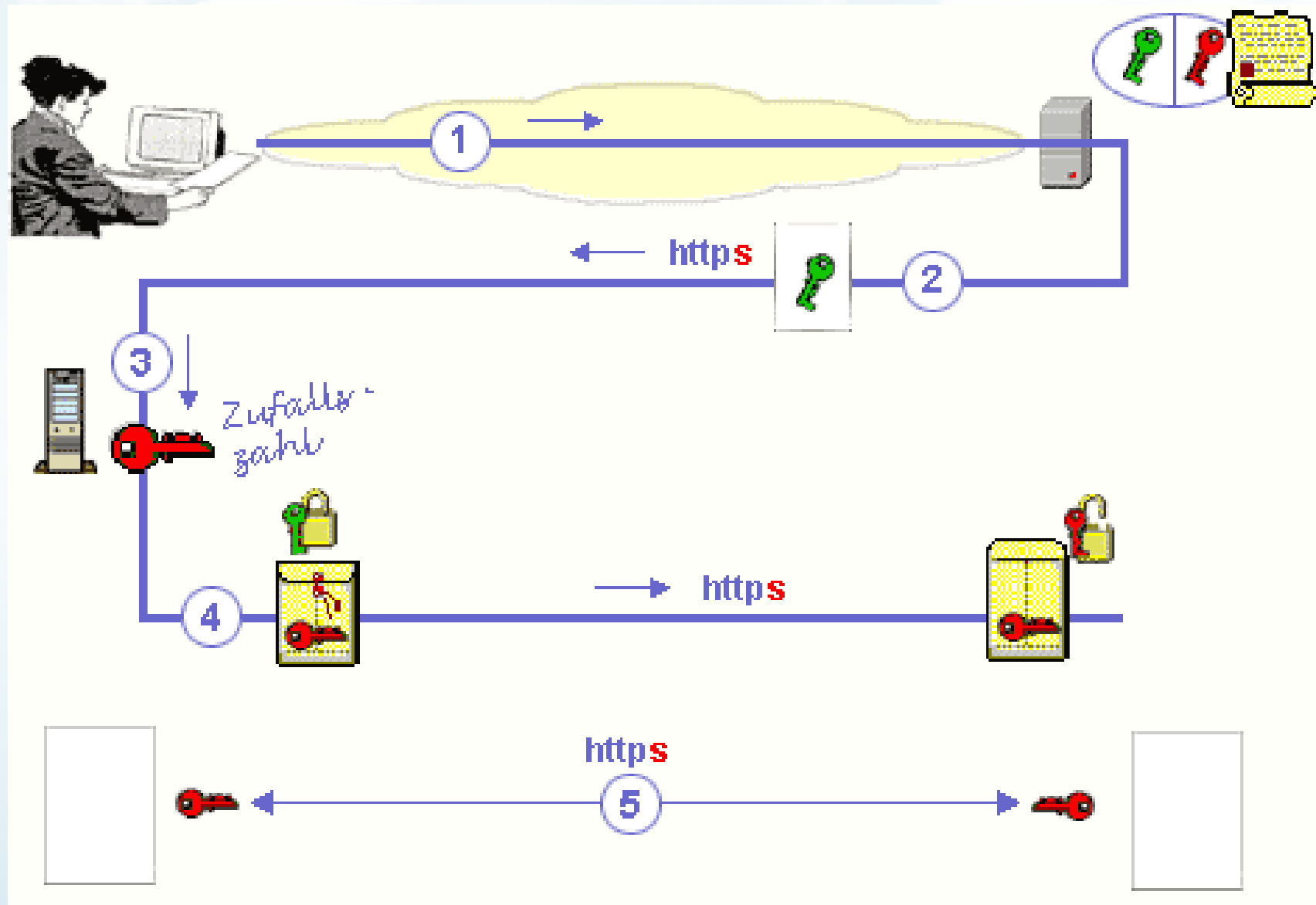
<http://sourceforge.net/projects/delphiwebstart>

DelphiWebStart (DWS) is an Application Loader with TCP Sockets based on a SmallClient which is first spread over the Web, VPN or Intranet. Then a user can download data (exes, maps, files etc.) from a easy list and start it. DWS 1.8 supports OpenSSL.

From DelphiWebStart to DataWebSecure (DEMO)

the important feature is the complete implementation of the protocols SSLv2, SSLv3, TLSv1 and a non-blocking IO abstraction (BIO)

This is OpenSSL



Vorbereiten der CA

1. Generieren eines Schlüsselpaares für die CA
2. Verteilen des CA-Zertifikates auf alle Browser

Vorbereiten des Webservers

3. generieren eines Schlüsselpaares für den Webserver
4. Zertifizierung des Webservers nach Prüfung durch die CA

Unsymmetrischer Sitzungsaufbau

1. Aufbau der Verbindung `https://www.ar.admin.ch` auf Port 443
2. Übertragen des Webserver-Zertifikats zum Browser
3. Prüfen der Signatur des Zertifikats anhand des von der CA hinterlegten Schlüssels, bei Erfolg ist die Identität des Webservers festgestellt
4. Generieren eines temporären Sitzungsschlüssels
5. Senden des Schlüssels in einer nur für den Webserver lesbaren Art
6. Entschlüsseln des Sitzungsschlüssels

Symmetrischer SSL-Tunnel

7. Symmetrische Ver- und Entschlüsselung beim Client
8. Symmetrische Ver- und Entschlüsselung beim Server

DEMO: CrypTool Signieren

Anwendungsbeispiele (I)

Verschlüsselung mit RSA – Mathematischer Hintergrund / Verfahren

- Öffentlicher Schlüssel (public key): (n, e)
- Privater Schlüssel (private key): (d)

wobei:

p, q große zufällig gewählte Primzahlen mit $n = p \cdot q$;

d wird unter den NB $\text{ggT}[\varphi(n), e] = 1$; $e \cdot d \equiv 1 \pmod{\varphi(n)}$; bestimmt.

Ver- und Entschlüsselungs-Operation: $(m^e)^d \equiv m \pmod{n}$

- n ist der Modulus, dessen Schlüssellänge beim RSA-Verfahren angegeben wird.
- ggT = größter gemeinsamer Teiler.
- $\varphi(n)$ ist die Eulersche Phi-Funktion.

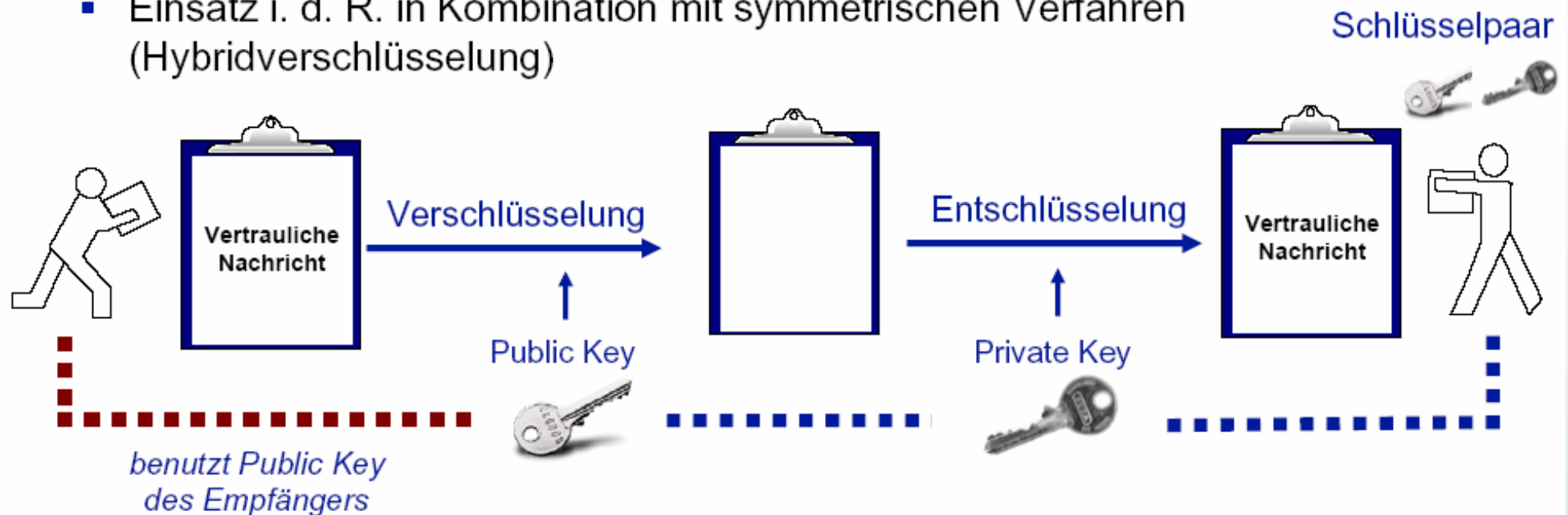
Vorgehen:

- Transformation von Nachrichten in binäre Repräsentation
- Nachricht $m = m_1, \dots, m_k$ blockweise verschlüsseln, wobei für alle m_j gilt:
 $0 \leq m_j < n$; also maximale Blockgröße r so, dass gilt: $2^r \leq n$

Anwendungsbeispiele (I)

Verschlüsselung mit RSA

- **Grundlage** für z.B. SSL-Protokoll (Zugriff auf gesicherte Web Seiten)
- **Asymmetrische Verschlüsselung mit RSA**
 - Jeder Benutzer hat ein Schlüsselpaar – einen öffentlichen und einen privaten.
 - Sender verschlüsselt mit dem öffentlichen Schlüssel (*public key*) des Empfängers.
 - Empfänger entschlüsselt mit seinem privaten Schlüssel (*private key*).
- Einsatz i. d. R. in Kombination mit symmetrischen Verfahren (Hybridverschlüsselung)



Algorithms implemented

Block ciphers: DES, 3DES, DESX, CAST, RC2, RC5,
IDEA (patent), Blowfish, AES

stream cipher: RC4, RC5 (patent)

digests: MD2, MD4, MD5, SHA-1, RIPEMD 160,
MDC2 (for smartcards, IBM patent)

asymmetric cryptosyst.: RSA, DSA, DH

MAC: HMAC

Standards implemented

PKCS 1 (full), PKCS 7 (almost complete for the types actually used: Data, Signed and Enveloped), PKCS 8 (full), PKCS10 (full) and PKCS 12

X509v3, CSRs, CRLs

ASN.1 with DER encoding (not complete)

PEM based ASCII-binary encoding

SSLv3 and TLSv1 (practically identical)

Command Shell implemented

OpenSSL includes a command line utility that can be used to perform a variety of cryptographic functions like generating your machine certificate in [CERT].
First you need a RootCA (selfsigned)

```
// we generate the private key of the CA:  
1. openssl genrsa -des3 -out CA_pvk.pem 1024  
// we sign the private to make a certificate of CA  
2. openssl -new -x509 -days 365 -key CA_pvk.pem -out CA_cert.pem  
// we need the host private key  
3. openssl genrsa -des3 -out host_pvk.pem 1024  
// we sign the host private from the CA (machine certificate)  
4. openssl req -new key host_pvk.pem -out host_csr.pem  
5. openssl ca -out host_cert.pem - in host_csr.pem -cert CA_cert.pem -keyfile  
CA_pvk.pem  
in this way we get:  
[CERT]  
ROOTCERT=cert\CA_cert.pem  
SCERT=cert\host_cert.pem  
RSAKEY=cert\host_pvk.pem
```

asn1parse

parse an ASN.1 sequence

ca

Certificate Authority (CA) management

ciphers

cipher suite description

crl

Certificate Revocation List (CRL) management

crl2pkcs7

CRL to PKCS#7 conversion

dgst

message digest calculation

dh

Diffie-Hellman parameter management. Obsoleted
by dhparam

dsa

DSA data management

dsaparam

DSA parameter generation

enc

encoding with ciphers

genrsa

generation of RSA parameters

ocsp

Online Certificate Status Protocol utility

passwd

generation of hashed passwords

pkcs12

PKCS#12 data management

pkcs7

PKCS#7 data management

rand

generate pseudo-random bytes

req

X.509 Certificate Signing Request (CSR)
management

rsa

RSA data management

rsautl

RSA utility for signing, verification, encryption, and
decryption

smime

S/MIME mail processing

speed

algorithm speed measurement

verify

X.509 certificate verification

version

OpenSSL version information

x509

X.509 certificate data management

Situation is slowly improving

best source: <http://www.openssl.org/docs/>, updated man page
(sometimes too updated)

for the SSL topic a book in depth is available from Eric Escorla
(<http://www.rtfm.com/>) (addison wesley 2001)

<http://www.rtfm.com/openssl-examples/>

<http://www2.linuxjournal.com/cgi-bin/frames.pl/index.html>

good support with mailing list:

<http://www.openssl.org/support/>

the code of the various demo applications !!!

the file openssl.txt in the directory doc

OpenSSL with Indy Version

current version 0.9.8h (28.5.2008) and Indy10

(Coming Soon: [Indy 10 for FreePascal and the Lazarus IDE](#))

Now fits OpenSSL 0.9.8g (Stable) with Indy 9 and 10

(rename IdSSLOpenSSLHeaders9.pas or

IdSSLOpenSSLHeaders10.pas to

IdSSLOpenSSLHeaders.pas depending on your Indy
Version). (<http://www.indyproject.org/>)

Due changes to the Object Hierarchy you can choice between
Intercept or IOHandler:

Difference Interceptor or Handler

TIdTCPConnection.IOHandler or TIdTCPConnection.Intercept properties.

Intercept is used to perform operations that can include logging send and receive operations, or provide Secure Socket Layer (SSL) support for the connection.

IOHandler is used in methods that perform low-level read or write operations like ReadFromStack and WriteBuffer. IOHandler is also used in methods that *access* the connection status like CheckForDisconnect, Connected, DisconnectSocket, and Disconnect.

Each IOHandler can override additional higher level methods to provide optimizations!

Some remarks

The SSL capabilities of Indy 10 are now completely pluggable. Prior to Indy 10, the SSL support was pluggable at the TCP level, however protocols such as HTTP which used SSL for HTTPS were fixed to use Indy's default SSL implementation of OpenSSL.

Indy 10 continues to include support for OpenSSL, so the SSL capabilities of Indy are completely pluggable at the core and protocol level for other implementations or frameworks.

Verify a signature isn't included, you must specify and implement it on the callback function `onVerifyPeer()` or use the command line function!

Intelicom.si published their recommendation regarding how to compile OpenSSL with the Indy modification

Remarks of properties (SSLOptions)

Set `VerifyDepth` to 2 means that we accept the server certificate up to 2 levels of Certificate Chain (RootCA --> CA --> ServerCert)

Set property `Method` to `sslvSSLv23` means the ssl protocol will negotiate the proper mode automatically.

If `[sslvrfPeer]` on `Server` is true and we use `OnVerifyPeer()`, think about cross certification, means the server will request a client certificate too!

A closer look to indy9/indy10

```
{$IFDEF INDY10}
```

```
IOHandler:=
```

```
  TIdSSLIOHandlerSocketOpenSSL.Create(SoapClient.HTTPWebNode.  
                                       HttpClient);
```

```
{$ELSE}
```

```
IOHandler:=
```

```
  TIdSSLIOHandlerSocket.Create(SoapClient.HTTPWebNode.HttpClient);
```

```
{$ENDIF}
```

```
  with {$IFDEF INDY10}
```

```
    TIdSSLIOHandlerSocketOpenSSL
```

```
{$ELSE}
```

```
  TIdSSLIOHandlerSocket
```

```
{$ENDIF} (IOHandler), SSLOptions do  
  begin
```

That's all Folks ;))

Fazit:

Der Transformator OpenSSL lässt sich nun in jede Lok integrieren!