SSL Presentation

Practical SSL With Apache and PHP

... and OpenSSL, mod_ssl, X.509 Certificates.

Chris Snyder and Mike Southwell
New York PHP
22 November 2005

Part I: The Players

Cryptography basics

Certificates

Transport Layer Security

What you need:

- mod_SSL
- OpenSSL toolkit
- PHP --with-openssl

Basics: Hashing

not reversible
has native PHP support

md5('hello') : 5d41402abc4b2a76b9719d911017c592
sha1('hello') : aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d

Basics: Symmetric Encryption

Symmetric: both sides use the same key

Reversible
Requires out-of-band key exchange.
PHP needs mcrypt extension

- 3DES
- AES
- Blowfish
- RC4

We each have the same secret key.
You encrypt a value with this key, and send it to me.
The only way to decrypt it is with our shared secret key.

Basics: Asymmetric Encryption

Asymmetric: encryption and decryption use different keys

Reversible
Allows public knowledge of encryption key
PHP needs OpenSSL extension

RSA

I give you my public key.
You encrypt a value with my public key, and send it to me.
The only way to decrypt it is with my private key (which only I have).

Certificates

X509 Public Key Infrastructure

RFC3280 defines Certificates and Certificate Revocation Lists.

http://rfc.net/rfc3280.html

Certificate Authorities

Certificate Authorities provide secured identification of a server, and enable asymmetric encryption of messages between client and server.

Transport Layer Security

A 30-second history

Secure Sockets Layer was developed by Netscape in 1994 as a protocol which permitted persistent and secure transactions. In 1997 an Open Source version of Netscape's patented version was created, which is now OpenSSL. In 1999 the existing protocol was extended by a version now known as Transport Layer Security (TLS). By convention, the term "SSL" is used even when technically the TLS protocol is being used.

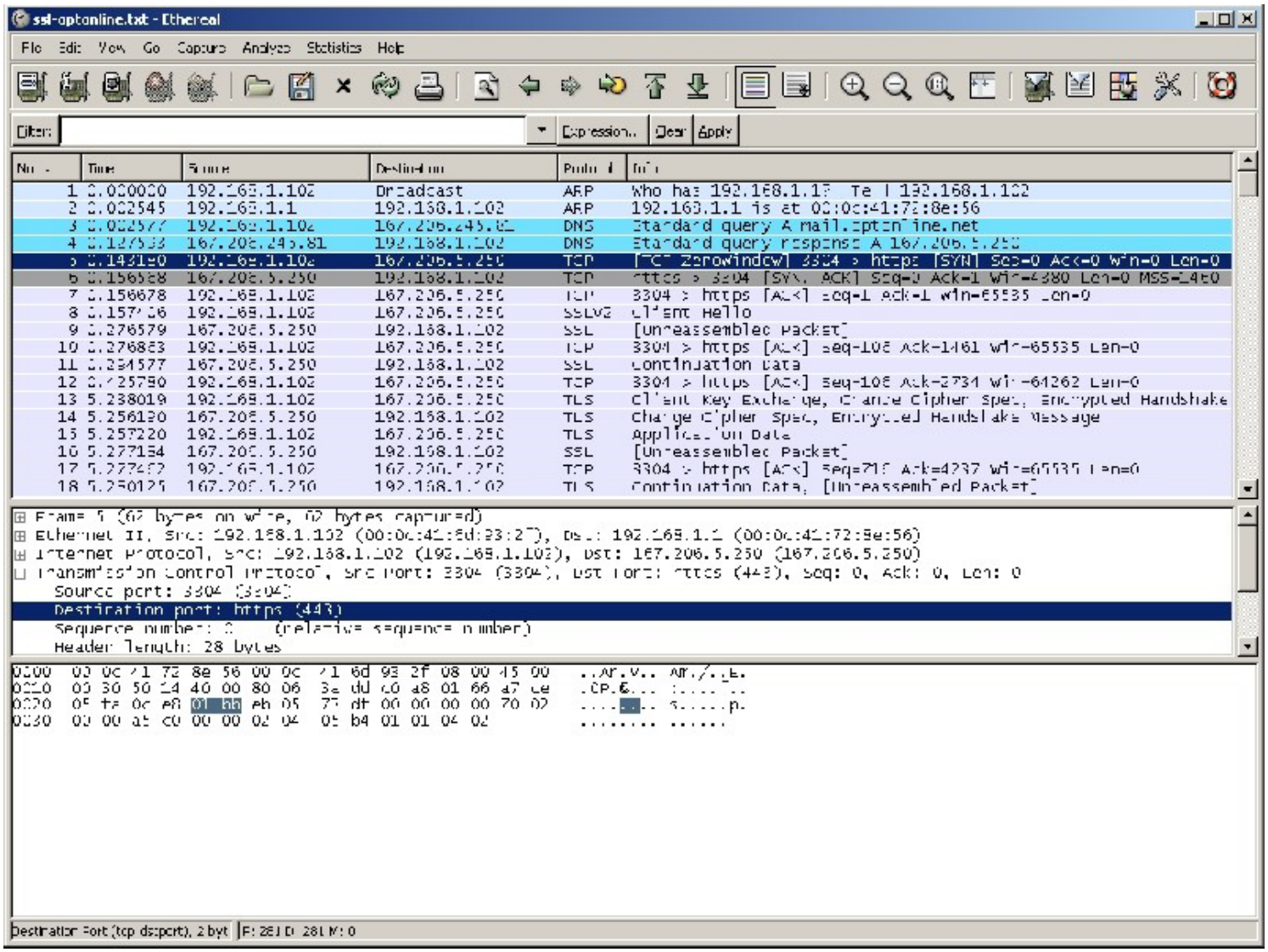
TLS: What It Does

- TLS encrypts messages.
- TLS makes message alteration detectable.
- TLS authenticates message senders/receivers.

TLS: The Handshake Protocol

SSL encrypts every transaction between the client browser and the server, which makes it possible to send sensitive information back and forth without fear that it will be readable by anybody who might intercept it. It can do this because, before it begins transferring encrypted information, the server engages in an elaborate negotiation with the client, called the Handshake Protocol. This negotiation has the following parts:

- The client sends a request to the server which, because it uses the https (as opposed to http) schema, initiates the negotiation.


ssl-optonline.txt - Ethereal
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Info
1 0.000000 192.168.1.102 Broadcast ARP Who has 192.168.1.1? Tell 192.168.1.102
2 0.002545 192.168.1.1 192.168.1.102 ARP 192.168.1.1 is at 00:0c:41:72:8e:56
3 0.002577 192.168.1.102 167.206.245.81 DNS Standard query A mail.optonline.net
4 0.127533 167.206.245.81 192.168.1.102 DNS Standard query response A 167.206.5.250
5 0.143150 192.168.1.102 167.206.5.250 TCP [TCP ZeroWindow] 3304 > https [SYN] Seq=0 Ack=0 Win=0 Len=0
6 0.156528 167.206.5.250 192.168.1.102 TCP https > 3304 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460
7 0.156678 192.168.1.102 167.206.5.250 TCP 3304 > https [ACK] Seq=1 Ack=1 Win=65535 Len=0
8 0.157206 192.168.1.102 167.206.5.250 SSLv2 Client Hello
9 0.276579 167.206.5.250 192.168.1.102 SSL [Unreassembled Packet]
10 0.276863 192.168.1.102 167.206.5.250 TCP 3304 > https [ACK] Seq=108 Ack=1461 Win=65535 Len=0
11 0.284577 167.206.5.250 192.168.1.102 SSL Continuation Data
12 0.425790 192.168.1.102 167.206.5.250 TCP 3304 > https [ACK] Seq=108 Ack=2734 Win=64262 Len=0
13 5.238019 192.168.1.102 167.206.5.250 TLS Client Key Exchange, Change Cipher Spec, Encrypted Handshake
14 5.256130 167.206.5.250 192.168.1.102 TLS Change Cipher Spec, Encrypted Handshake Message
15 5.257220 192.168.1.102 167.206.5.250 TLS Application Data
16 5.277154 167.206.5.250 192.168.1.102 SSL [Unreassembled Packet]
17 5.277462 192.168.1.102 167.206.5.250 TCP 3304 > https [ACK] Seq=715 Ack=4237 Win=65535 Len=0
18 5.250125 167.206.5.250 192.168.1.102 TLS Continuation Data, [Unreassembled Packet]
Frame 5 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 192.168.1.102 (00:0c:41:8d:93:2f), Dst: 192.168.1.1 (00:0c:41:72:8e:56)
Internet Protocol, Src: 192.168.1.102 (192.168.1.102), Dst: 167.206.5.250 (167.206.5.250)
Transmission Control Protocol, Src Port: 3304 (3304), Dst Port: https (443), Seq: 0, Ack: 0, Len: 0
Source port: 3304 (3304)
Destination port: https (443)
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
0000 00 0c 41 72 8e 56 00 0c 41 8d 93 2f 08 00 45 00
0010 00 30 50 24 40 00 80 06 3e dd c0 a8 01 66 a7 ce
0020 05 fa 0c e8 01 bb eb 05 77 df 00 00 00 00 70 02
0030 00 00 a5 c0 00 00 02 04 05 b4 01 01 04 02
Destination Port (tcp.dstport), 2 byt
P: 281 D: 281 M: 0

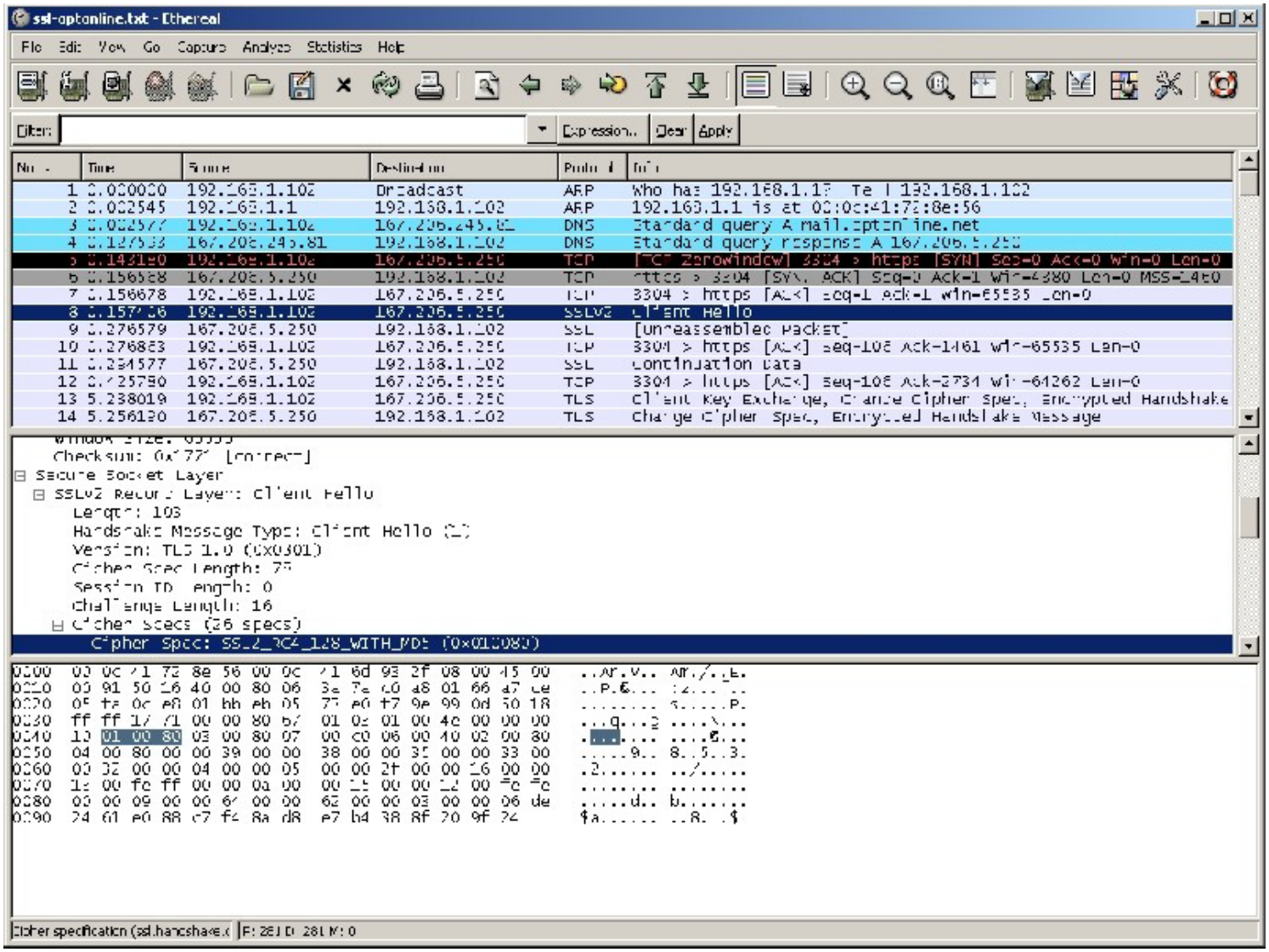
TLS: Server Handshake Response

The server responds with a plain-text message consisting of the following parts:

1. An exchange method to be used for passing back and forth the keys to be used for encrypting information. This is typically either RSA or Diffie-Hellman-Merkle. If it is RSA, the server must send along also a Certificate (discussed below).

2. The type of encryption to be used (RC4 or preferably 3DES).

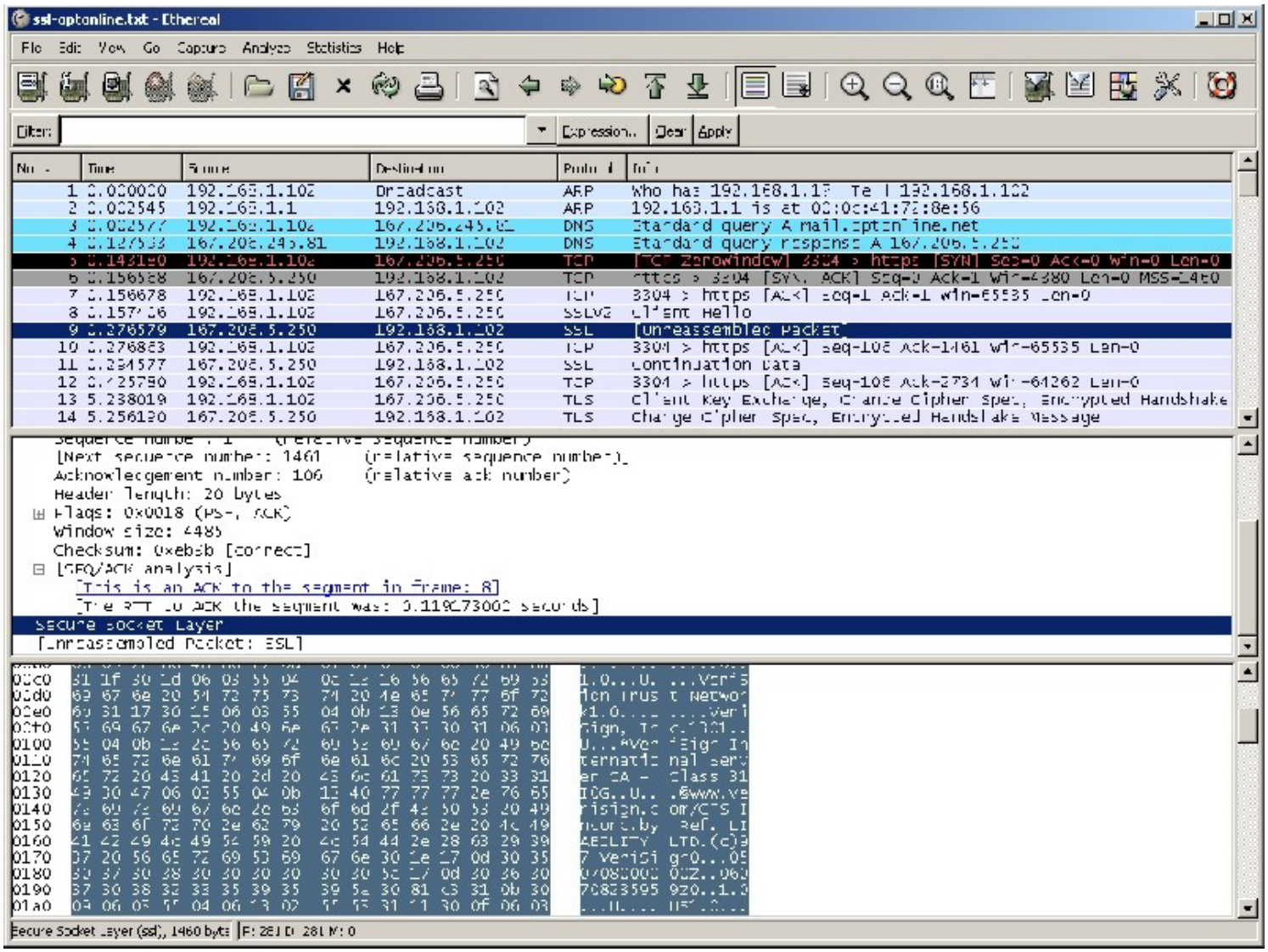
3. The technique to be used for calculating the Message Authentication Code, a checksum appended to messages and used to verify that the message contents havenâ€™t been tampered with. Typically MD5 or SHA-1.


ssl-optonline.txt - Ethereal
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Info
1 0.000000 192.168.1.102 Broadcast ARP Who has 192.168.1.1? Tell 192.168.1.102
2 0.002545 192.168.1.1 192.168.1.102 ARP 192.168.1.1 is at 00:0c:41:7f:8e:56
3 0.002577 192.168.1.102 167.206.245.81 DNS Standard query A mail.optonline.net
4 0.127533 167.206.245.81 192.168.1.102 DNS Standard query response A 167.206.5.250
5 0.143150 192.168.1.102 167.206.5.250 TCP [TCP ZeroWindow] 3304 > https [SYN] Seq=0 Ack=0 Win=0 Len=0
6 0.156528 167.206.5.250 192.168.1.102 TCP https > 3304 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460
7 0.156678 192.168.1.102 167.206.5.250 TCP 3304 > https [ACK] Seq=1 Ack=1 Win=65535 Len=0
8 0.157736 192.168.1.102 167.206.5.250 SSLv2 Client Hello
9 0.276579 167.206.5.250 192.168.1.102 SSL [Unreassembled Packet]
10 0.276863 192.168.1.102 167.206.5.250 TCP 3304 > https [ACK] Seq=108 Ack=1461 Win=65535 Len=0
11 0.284577 167.206.5.250 192.168.1.102 SSL Continuation Data
12 0.285790 192.168.1.102 167.206.5.250 TCP 3304 > https [ACK] Seq=108 Ack=2734 Win=64262 Len=0
13 5.238019 192.168.1.102 167.206.5.250 TLS Client Key Exchange, Change Cipher Spec, Encrypted Handshake
14 5.256130 167.206.5.250 192.168.1.102 TLS Change Cipher Spec, Encrypted Handshake Message
Checksum: 0x7f7f [correct]
Secure Socket Layer
SSLv2 Record Layer: Client Hello
Length: 103
Handshake Message Type: Client Hello (1)
Version: TLS 1.0 (0x0301)
Cipher Spec Length: 75
Session ID Length: 0
Challenge Length: 16
Cipher Specs (25 specs)
Cipher Spec: SSL2_RC4_128_WITH_MD5 (0x010080)
Cipher specification (ssl.handshake.c

TLS: Server Certificate

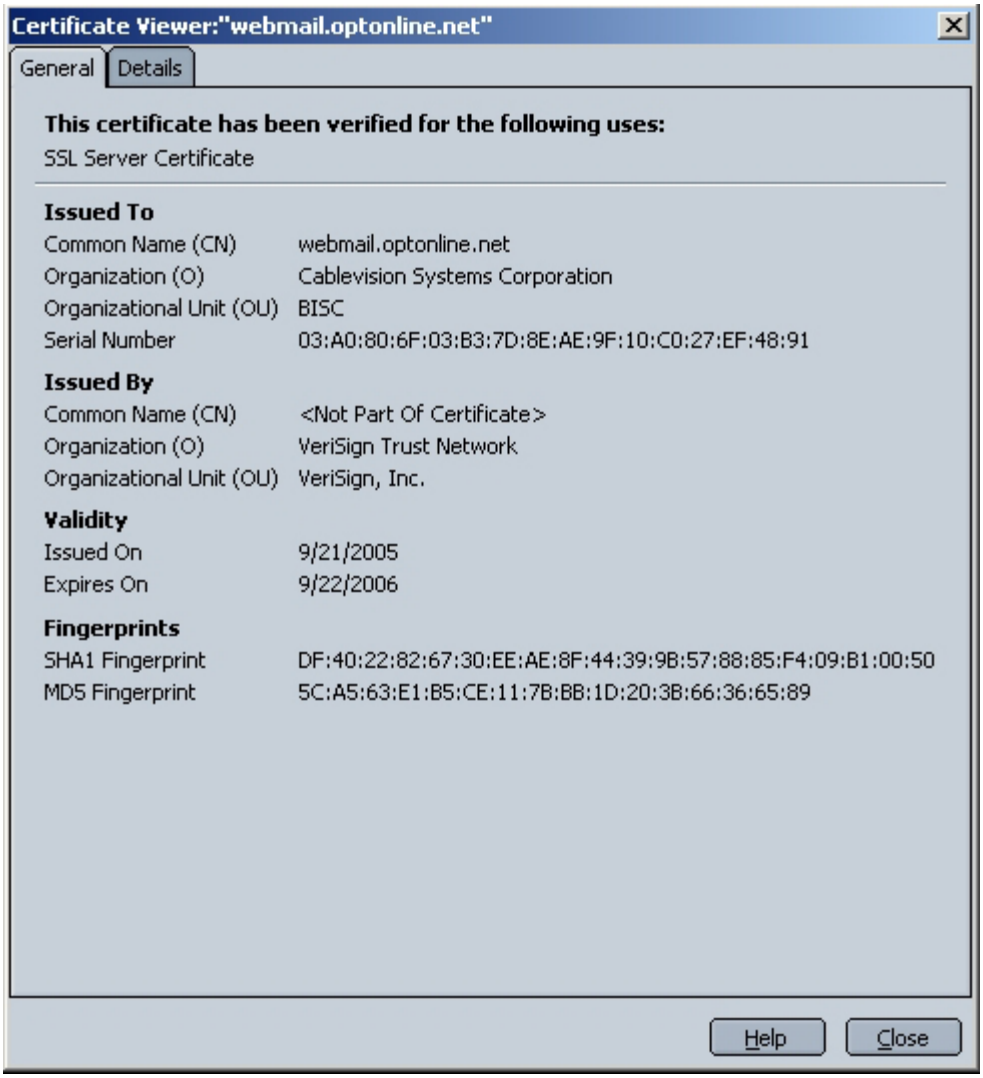
RSA key exchange enhances security by requiring the server to send a Certificate to the client. This Certificate is a binary collection of the following information:

1. Its identity
2. Its own attestation that it really is who it has said
3. A Certificate Authorityâ€™s attestation that the serverâ€™s attestation is true
4. Its public key, which may be used to encrypt the message encryption key, randomly generated by the client


ssl-optonline.txt - Ethereal
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
1 0.000000 192.168.1.102 Broadcast ARP Who has 192.168.1.1? Tell 192.168.1.102
2 0.002545 192.168.1.1 192.168.1.102 ARP 192.168.1.1 is at 00:0c:41:7f:8e:56
3 0.002577 192.168.1.102 167.206.245.81 DNS Standard query A mail.optonline.net
4 0.127533 167.206.245.81 192.168.1.102 DNS Standard query response A 167.206.5.250
5 0.143150 192.168.1.102 167.206.5.250 TCP [TCP ZeroWindow] 3304 > https [SYN] Seq=0 Ack=0 Win=0 Len=0
6 0.156528 167.206.5.250 192.168.1.102 TCP https > 3304 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460
7 0.156678 192.168.1.102 167.206.5.250 TCP 3304 > https [ACK] Seq=1 Ack=1 Win=65535 Len=0
8 0.157026 192.168.1.102 167.206.5.250 SSLv2 Client Hello
9 0.276579 167.206.5.250 192.168.1.102 SSL [Unreassembled Packet]
10 0.276863 192.168.1.102 167.206.5.250 TCP 3304 > https [ACK] Seq=106 Ack=1461 Win=65535 Len=0
11 0.284577 167.206.5.250 192.168.1.102 SSL Continuation Data
12 0.285790 192.168.1.102 167.206.5.250 TCP 3304 > https [ACK] Seq=106 Ack=2734 Win=64262 Len=0
13 5.238019 192.168.1.102 167.206.5.250 TLS Client Key Exchange, Change Cipher Spec, Encrypted Handshake
14 5.256130 167.206.5.250 192.168.1.102 TLS Change Cipher Spec, Encrypted Handshake Message
Sequence number: 1 (relative sequence number)
[Next sequence number: 1461 (relative sequence number)]
Acknowledgement number: 106 (relative ack number)
Header length: 20 bytes
Flags: 0x0018 (PSH, ACK)
Window size: 4485
Checksum: 0xeb5b [correct]
[SEQ/ACK analysis]
[This is an ACK to the segment in Frame: 8]
[The RTT to ACK the segment was: 0.119573000 seconds]
Secure Socket Layer
[Unreassembled Packet: SSL]
Secure Socket Layer (ssl), 1460 bytes

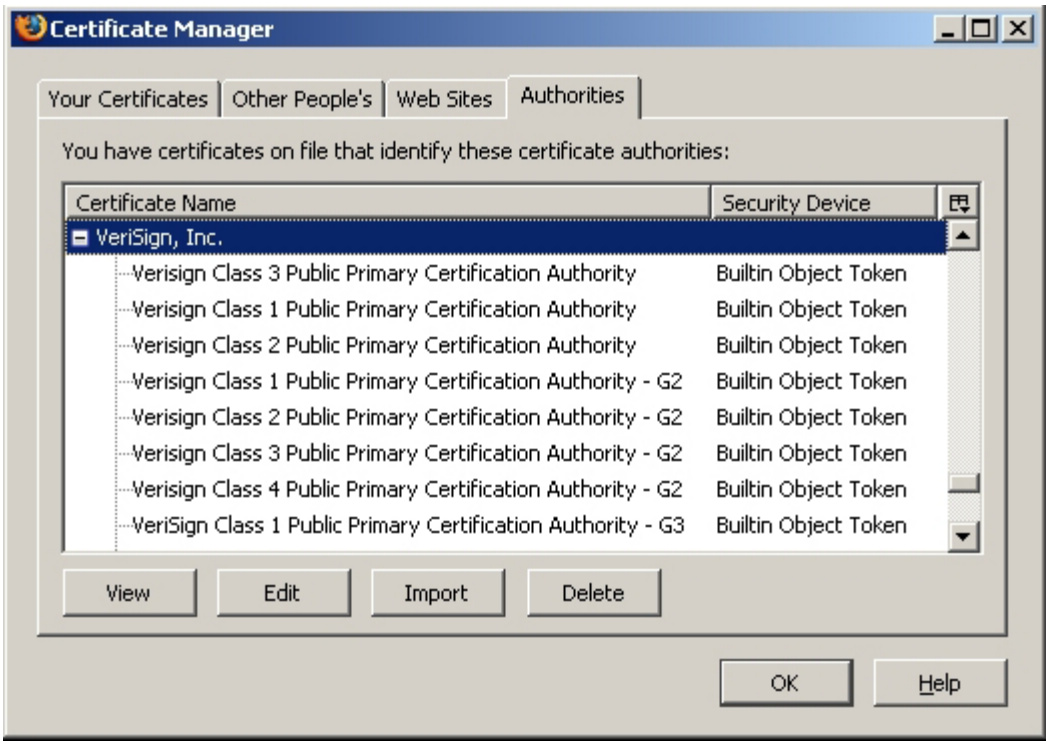
TLS: Server Certificate Verification

- The client browser recognizes the Certificate Authority and thus verifies the authenticity of the connection.



TLS: Root Certificates

It can do this because it has pre-installed certificates from many Authorities.

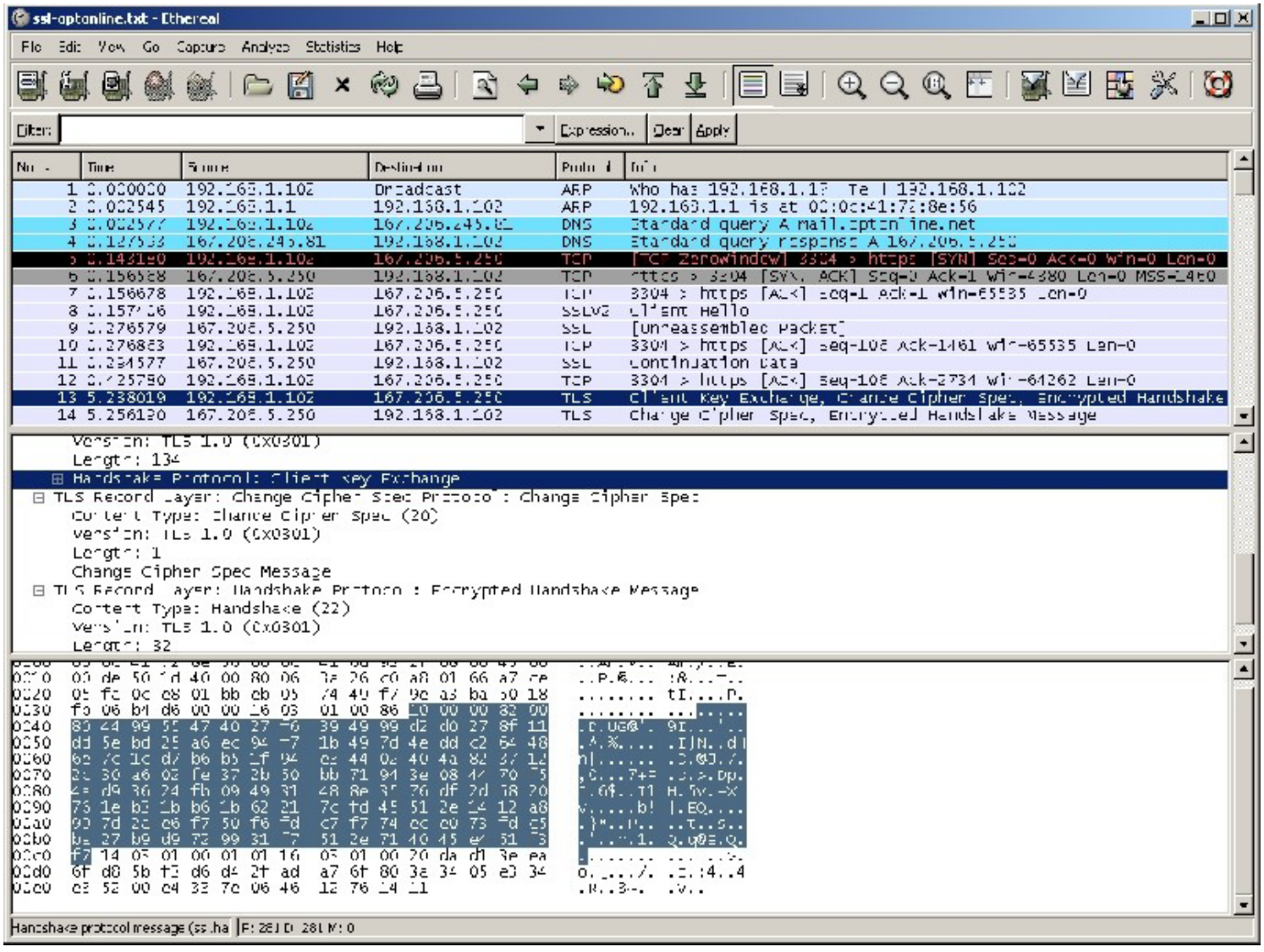


TLS: Failed Verification

If there is a conflict between the name on the certificate and the name of the server, the browser pops up a â€œDomain Name Mismatchâ€ notice, allowing the user to decide whether to continue.

TLS: Client Handshake Response

- Assuming the client does trust the server, it generates a random key to be used by the server for message encryption. It then encrypts that key with the serverâ€™s public key, and sends it back to the server.



TLS: Handshake Wrapup

At this point, the Handshake Protocol is finished. Both the server and the client know the following:

1. What key and encryption method to use for message encryption
2. What key and hashing method to use for message signing

There is, then, a great deal going on behind the scenes, which explains why SSL connections are sometimes slower than insecure ones.

TLS: Record Protocol

Now the Record Protocol starts to manage all future communication. The server finally is ready to respond to the original request, as follows:

1. It encrypts its http response.
2. It calculates a hash of its response
3. It appends that hash as a signature.

TLS: Record Protocol (cont'd)

Upon receiving the serverâ€™s response, the client does the following:

1. It detaches the signature.
2. It calculates a new hash of the response, and compares that to the detached signature.
3. (Assuming they match,) it decrypts the response.
4. It displays the requested html.

Apache's mod_ssl

Installed by default in Apache 2, and available for Apache 1.3, *mod_ssl* provides cryptography so that Apache can interact with the OpenSSL toolkit.

http://www.modssl.org/

OpenSSL

The OpenSSL project provides a toolkit and cryptography library for implementing the TLS protocol.

http://www.openssl.org/

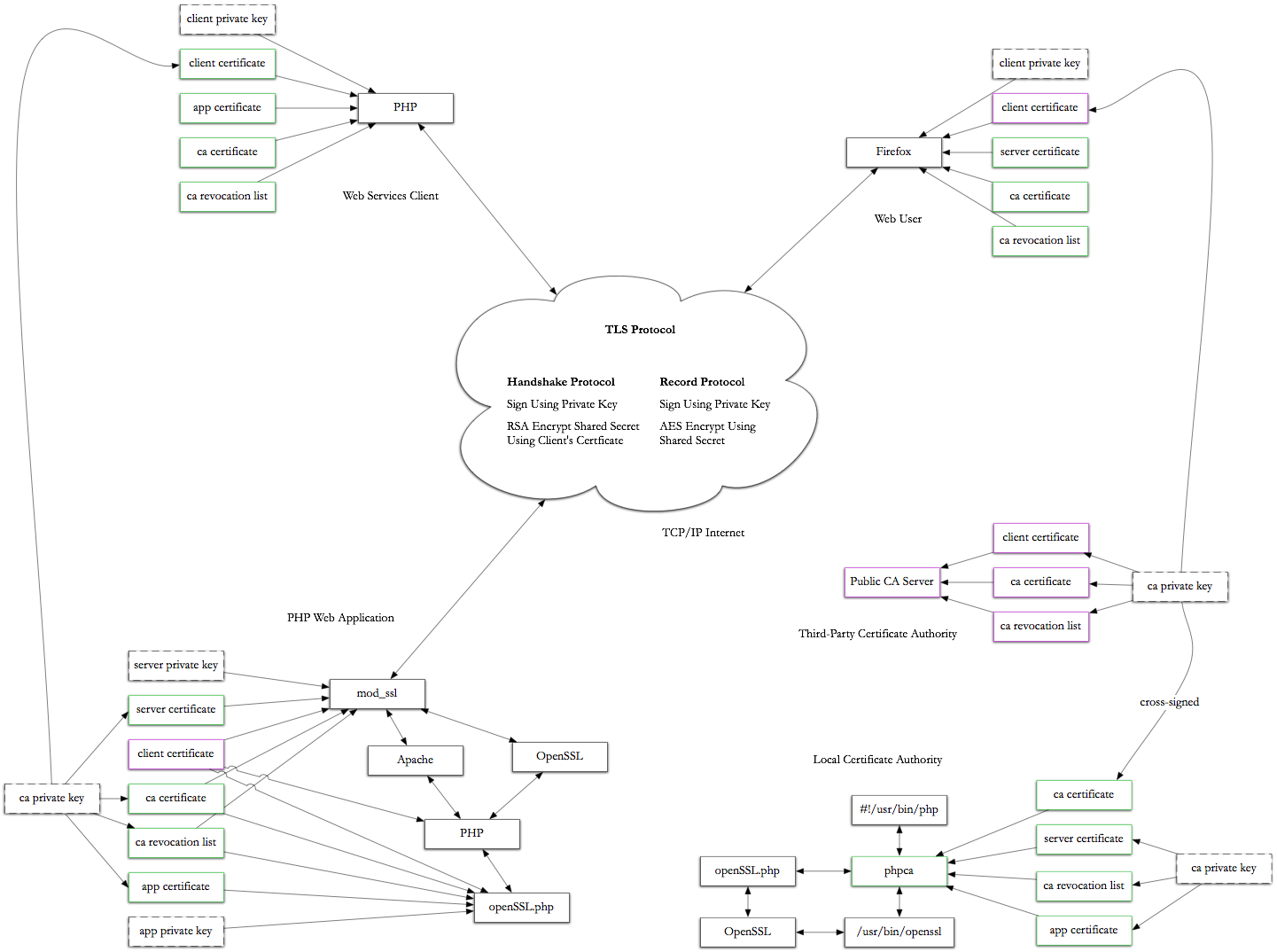
PHP's OpenSSL Support

PHP reports that it must be compiled *--with-openssl* in order to use its wrapper support for OpenSSL, although that appears not to be completely true.

http://www.php.net/openssl

The Big Picture

Green parts are created by local CA, pink parts by third-party CA.



Part II: SSL-enabled Webservers

HTTPS Requests and Responses are:

- Encrypted
- Tamper-proof
- Authenticated (Server, anyway)

HTTPS Requirements

- Apache with mod_ssl
- An IP Address
- Private server key (passphrase encrypted)
- Signed Server Certificate
- (optional) Certificate Authority Certificate(s)
- (optional) PassphraseDialog script

Using PHP to simplify PKI

phpca is a command line utility to create and manage a simple Certificate Authority for use with Apache's mod_ssl.

```
./phpca <command> [<server> | "ca"]
```

A Local Certificate Authority

```
./phpca cagen
```

A New Server Certificate

```
./phpca newserver ssl.example.org
```

Global SSL Directives

```
##
## Global SSL Directives
##

#LoadModule ssl_module modules/mod_ssl.so
Listen 443

# Random number generation
SSLRandomSeed startup file:/dev/urandom 512
SSLRandomSeed connect file:/dev/urandom 512

# Session Cache
SSLSessionCache none
SSLSessionCacheTimeout 600
SSLPassphraseDialog builtin
SSLMutex
file:/Applications/xampp/xamppfiles/logs/ssl_mutex

AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl    .crl
```

A Passphrase Dialog

Create /root/httpdkey:

```
#!/bin/sh
 echo 'my secret password'
```

Make sure only root has access:

```
chmod 700 /root/httpdkey
```

Use an external SSLPassphraseDialog:

```
SSLPassphraseDialog exec:/root/httpdkey
```

Manual

Apache Per-host SSL Directives

```
##
## SSL Virtual Host Context
##
<VirtualHost _default_:443>
  #  General setup for the virtual host
  DocumentRoot "/Users/csnyder/Sites/https"
  ServerName localhost
  ServerAdmin csnyder@chxo.com
  ErrorLog logs/sslerror.log

  #   SSL Configuration
  SSLEngine on
  SSLCipherSuite HIGH:MEDIUM
  SSLCertificateFile /etc/httpd/ssl/localhost.cert
  SSLCertificateKeyFile /etc/httpd/ssl/localhost.key
  SSLCertificateChainFile /etc/httpd/ssl/ca.cert
  SSLCARevocationFile /etc/httpd/ssl/ca.crl

  #   Client Authentication (Type):
  #SSLVerifyClient require
  #SSLVerifyDepth  10

  #   SSL Engine Options:
  <Files ~ "\.(cgi|shtml|phtml|php?)$">
      SSLOptions +StdEnvVars
  </Files>

  #   SSL Protocol Adjustments:
  SetEnvIf User-Agent ".*MSIE.*" \
           nokeepalive ssl-unclean-shutdown \
           downgrade-1.0 force-response-1.0

  #   Per-Server Logging:
  CustomLog logs/ssl_request_log \
            "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x
\"%r\" %b"
```

```
</VirtualHost>
```

Cipher Suite

DIY suite-ness!

```
openssl ciphers -v 'HIGH:MEDIUM'
```

Manual

Cross Your Fingers

Revoking A Server Certificate

```
./phpca revoke ssl.example.org
```

Part III: Application Level SSL

A PHP application can use OpenSSL's RSA support to:

- Sign values to prevent tampering
- Encrypt values to keep them private

Example: Stored Secrets

On storage, application needs:

- Application Certificate (Public Key) to encrypt values

On retrieval, application needs:

- Application Private Key to decrypt values
- ...and Private Key passphrase

Class openSSL

```
include_once( './openSSL.php' );
$server = new openSSL();
$server->privateKey( file_get_contents( $server_key ) );

$server->certificate( file_get_contents( $server_cert ) );
```

Signing

```
$value_signed = $ssl->sign( $value , $passphrase );
```

Looks like:

```
So long, and thanks for all the fish.
-----BEGIN openSSL.php SIGNATURE-----
e35fbZmTXYbVP33HY1CX31hxirAiENmEY4A0JHiGqPDfS87BmDQszGHP

CSvnMAyyqF7OuwRYnRS+VdiDNy2y3Uh2gqB7SNGXgWZnRNsMTeGX4OKX

NB5uwUQ+zNu7QjYfapQyliYoOWS5L+GcmFRHveTwAgw=
-----END openSSL.php SIGNATURE-----
```

Encrypting

Short values only! (56 chars max)

```
$secret = $ssl->encrypt( $value, $client_certificate );
```

Looks like:

```
Nlmc97wDavLSF7R078NcluVX8SrYMXdDFu9otugP5LDIPTW++VUQ5DOs
Z9qVS9+AL01gv1H2Ub8eNRcV67WgQw/5lMA92+T5KprSqV+CP/FXfcNC
nd3pq/H6TkP0LTF7zX9Q7GZg/4UuMHmc9EcGHhoOxBM=
```

Decrypting

```
$value = $ssl->decrypt( $secret, $passphrase );
```

Verifying

```
$verified = $ssl->verify( $signed_value,
$client_certificate );
```

Checking The Key Passphrase

```
$success = $ssl->checkKey( $passphrase );
```

Parsing X.509 Data

```
$ssl->certificate( $certificate_data );
$dn = $ssl->getDN();
print_r( $dn );
```

Looks like:

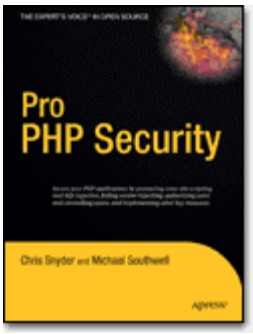
```
(
    [C] => US
    [ST] => New York
    [L] => New York City
    [O] => CHXO Internet
    [OU] => Certificate Authority
    [CN] => ca.chxo.com
    [emailAddress] => csnyder@chxo.com
)
```

Discussion

How do *you* use SSL?

Thank You!

For many many more details, please consult "Pro PHP Security", published in September by Apress.



Resources

Some of the many resources used in the making of this presentation.

Cryptography

- Diffie-Hellman-Merkle Key Exchange
- PKCS #7: Cryptographic Message Syntax Standard
- PKCS #12: Personal Information Exchange Syntax Standard
- RSA Cryptosystem
- S/MIME Specification 3.1 (RFC 3851)

Legal

- CNET: Netscape Patents Crypto Protocol (Sept. 16, 1997)
- United States Patent: 4,405,829 (RSA)
- United States Patent: 5,657,390 (SSL)

Manuals

- mod_ssl Directives
- OpenSSL Manuals
- PHP OpenSSL Functions

Other How-Tos

- Certificate Management and Generation with OpenSSL
- Generating an SSL Certificate with Apache+mod_ssl

SSL/TLS Protocol

- The SSLv2 Protocol
- The SSLv3 Protocol
- The TLS Protocol v1.0
- X.509 Public Key Infrastructure Spec (RFC 3280)

Download

Presentation files as a ZIP archive.

Contents:

```
psslwap/FCNYLicense.txt
psslwap/README
psslwap/cnf_template.php
psslwap/config-dist.php
psslwap/configure
psslwap/fcnyCLI.php
psslwap/gpl.txt
psslwap/openSSL.php
psslwap/openSSLDemo.php
psslwap/openssl.cnf
psslwap/phpca
psslwap/players.png
```