

SSL/TLS

Presenter: Yinzhi Cao

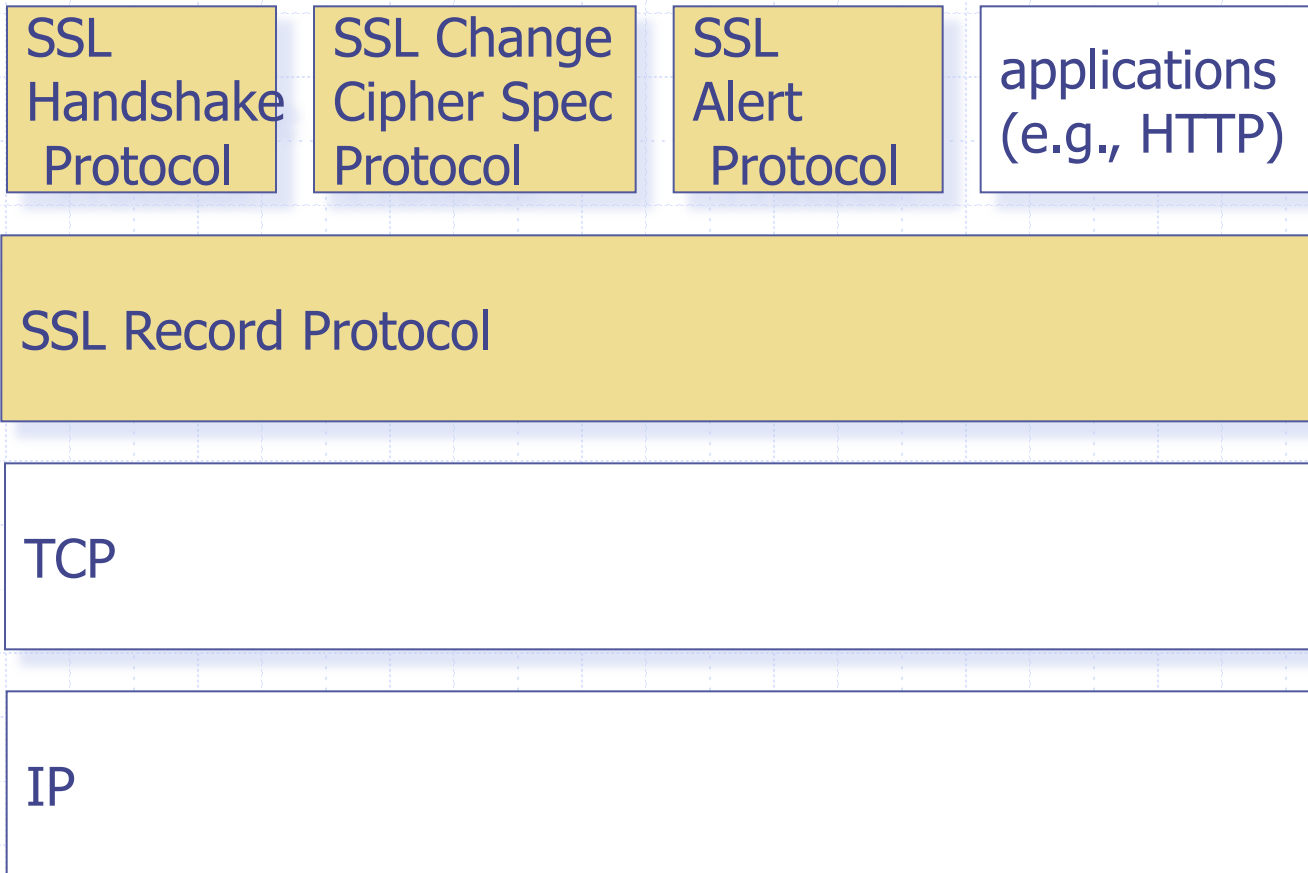
Acknowledgement

- ◆ <http://theory.stanford.edu/~jcm/slides/tecs-05/02-SSL.ppt>
- ◆ http://www.hit.bme.hu/~buttyan/courses/BMEVIHI9367/SSL_TLS.ppt
- ◆ <https://lyle.smu.edu/~nair/courses/7349/ssl.ppt>
- ◆ <http://www.cs.utexas.edu/~shmat/courses/cs6431/ssl.ppt>

What are SSL and TLS?

- ◆ SSL – Secure Socket Layer
- ◆ TLS – Transport Layer Security
- ◆ both provide a secure transport connection between applications (e.g., a web server and a browser)
- ◆ SSL was developed by Netscape
- ◆ SSL version 3.0 has been implemented in many web browsers (e.g., Netscape Navigator and MS Internet Explorer) and web servers and widely used on the Internet
- ◆ SSL v3.0 was specified in an Internet Draft (1996)
- ◆ it evolved into TLS specified in RFC 2246
- ◆ TLS can be viewed as SSL v3.1

SSL architecture



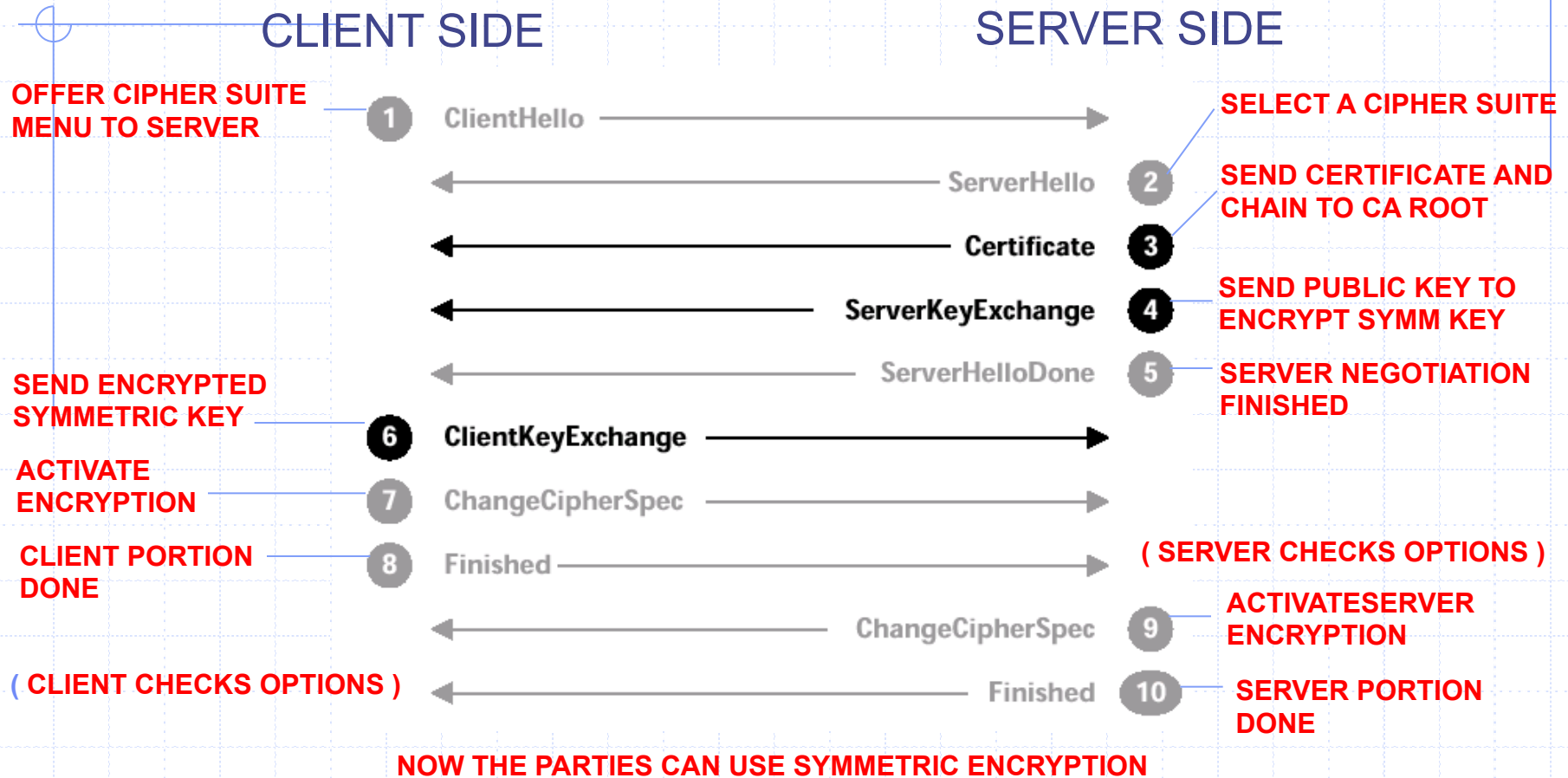
Handshake

- ◆ Negotiate Cipher-Suite Algorithms
 - Symmetric cipher to use
 - Key exchange method
 - Message digest function
- ◆ Establish and share master secret
- ◆ Optionally authenticate server and/or client

Handshake Phases

- ◆ Hello messages
- ◆ Certificate and Key Exchange messages
- ◆ Change CipherSpec and Finished messages

SSL Messages

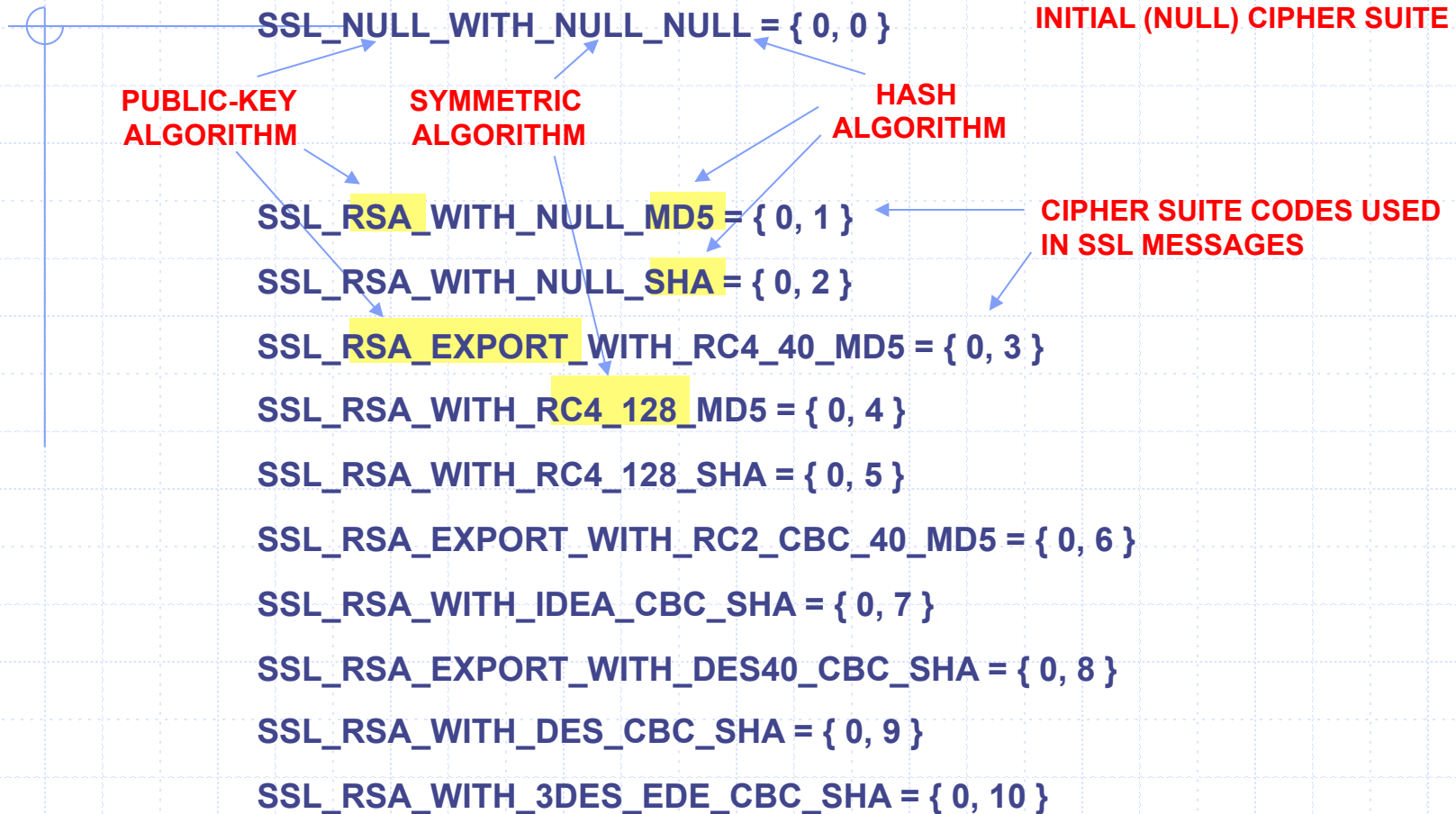


SOURCE: THOMAS, *SSL AND TLS ESSENTIALS*

Client Hello

- Protocol version
 - ◆ SSLv3(major=3, minor=0)
 - ◆ TLS (major=3, minor=1)
- Random Number
 - ◆ 32 bytes
 - ◆ First 4 bytes, time of the day in seconds, other 28 bytes random
 - ◆ Prevents replay attack
- Session ID
 - ◆ 32 bytes – indicates the use of previous cryptographic material
- Compression algorithm

Client Hello - Cipher Suites



Server Hello

- ◆ Version
- ◆ Random Number
 - Protects against handshake replay
- ◆ Session ID
 - Provided to the client for later resumption of the session
- ◆ Cipher suite
 - Usually picks client's best preference – No obligation
- ◆ Compression method

Certificates

- ◆ Sequence of X.509 certificates
 - Server's, CA's, ...
- ◆ X.509 Certificate associates public key with identity
- ◆ Certification Authority (CA) creates certificate
 - Adheres to policies and verifies identity
 - Signs certificate
- ◆ User of Certificate must ensure it is valid

Validating a Certificate

- ◆ Must recognize accepted CA in certificate chain
 - One CA may issue certificate for another CA
- ◆ Must verify that certificate has not been revoked
 - CA publishes Certificate Revocation List (CRL)

Client Key Exchange

◆ Premaster secret

- Created by client; used to “seed” calculation of encryption parameters
- 2 bytes of SSL version + 46 random bytes
- Sent encrypted to server using server’s public key

This is where the attack happened in SSLv2



Change Cipher Spec & Finished Messages

◆ Change Cipher Spec

- Switch to newly negotiated algorithms and key material

◆ Finished

- First message encrypted with new crypto parameters
- Digest of negotiated master secret, the ensemble of handshake messages, sender constant
- HMAC approach of nested hashing

SSL Encryption

◆ Master secret

- Generated by both parties from premaster secret and random values generated by both client and server

◆ Key material

- Generated from the master secret and shared random values

◆ Encryption keys

- Extracted from the key material

Generating the Master Secret

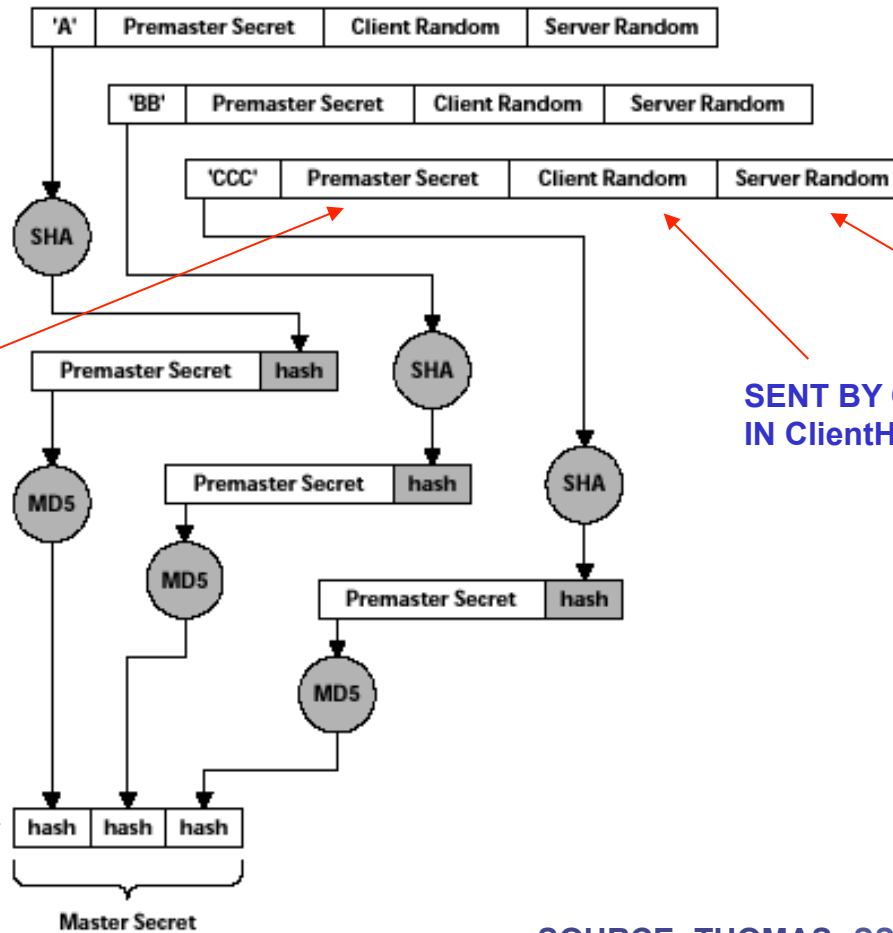
SERVER'S PUBLIC KEY IS SENT BY SERVER IN ServerKeyExchange

CLIENT GENERATES THE PREMASTER SECRET

ENCRYPTS WITH PUBLIC KEY OF SERVER

CLIENT SENDS PREMASTER SECRET IN ClientKeyExchange

MASTER SECRET IS 3 MD5 HASHES CONCATENATED TOGETHER = 384 BITS



SENT BY SERVER IN ServerHello

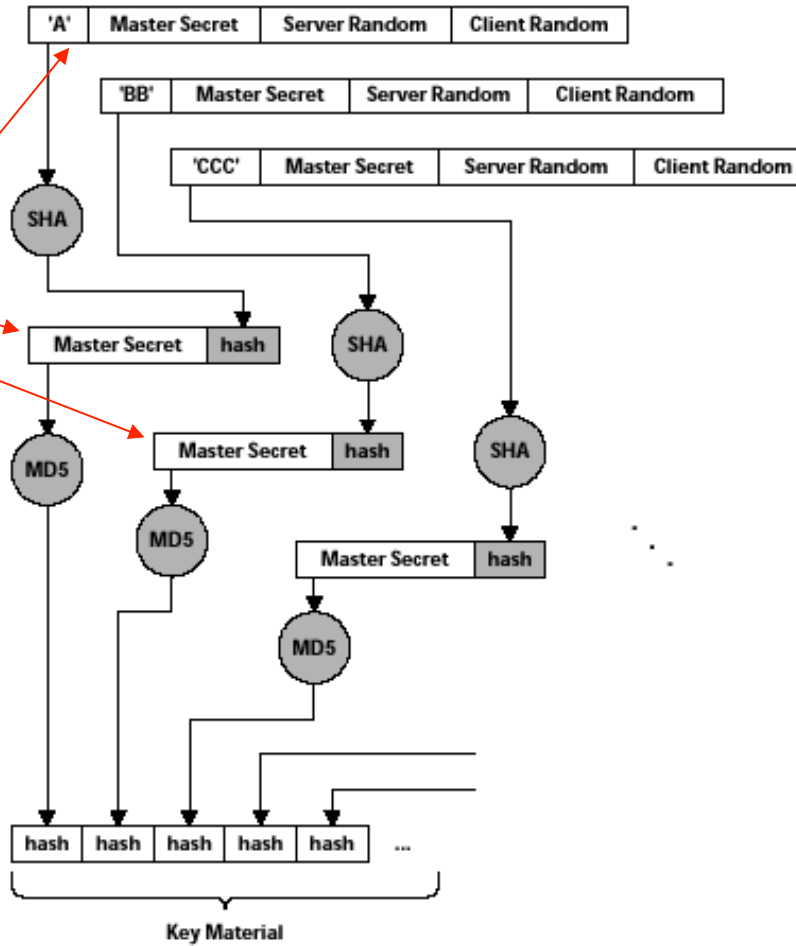
SENT BY CLIENT IN ClientHello

SOURCE: THOMAS, SSL AND TLS ESSENTIALS

Generation of Key Material

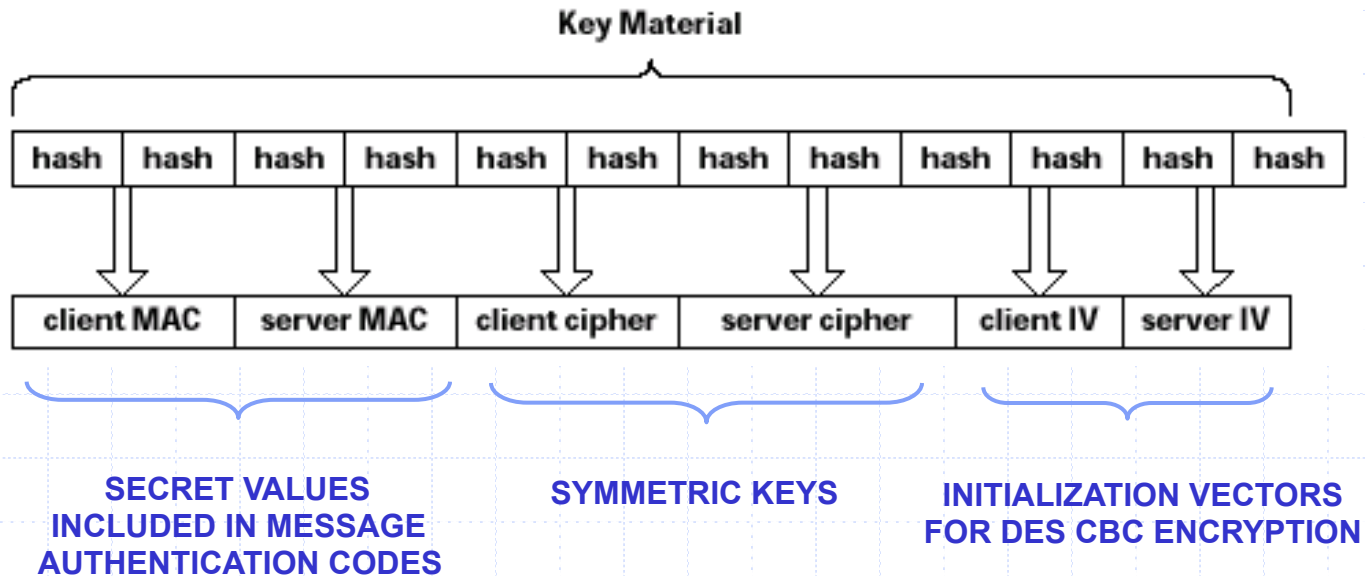
JUST LIKE FORMING THE MASTER SECRET

EXCEPT THE MASTER SECRET IS USED HERE INSTEAD OF THE PREMASTER SECRET



SOURCE: THOMAS, *SSL AND TLS ESSENTIALS*

Obtaining Keys from the Key Material

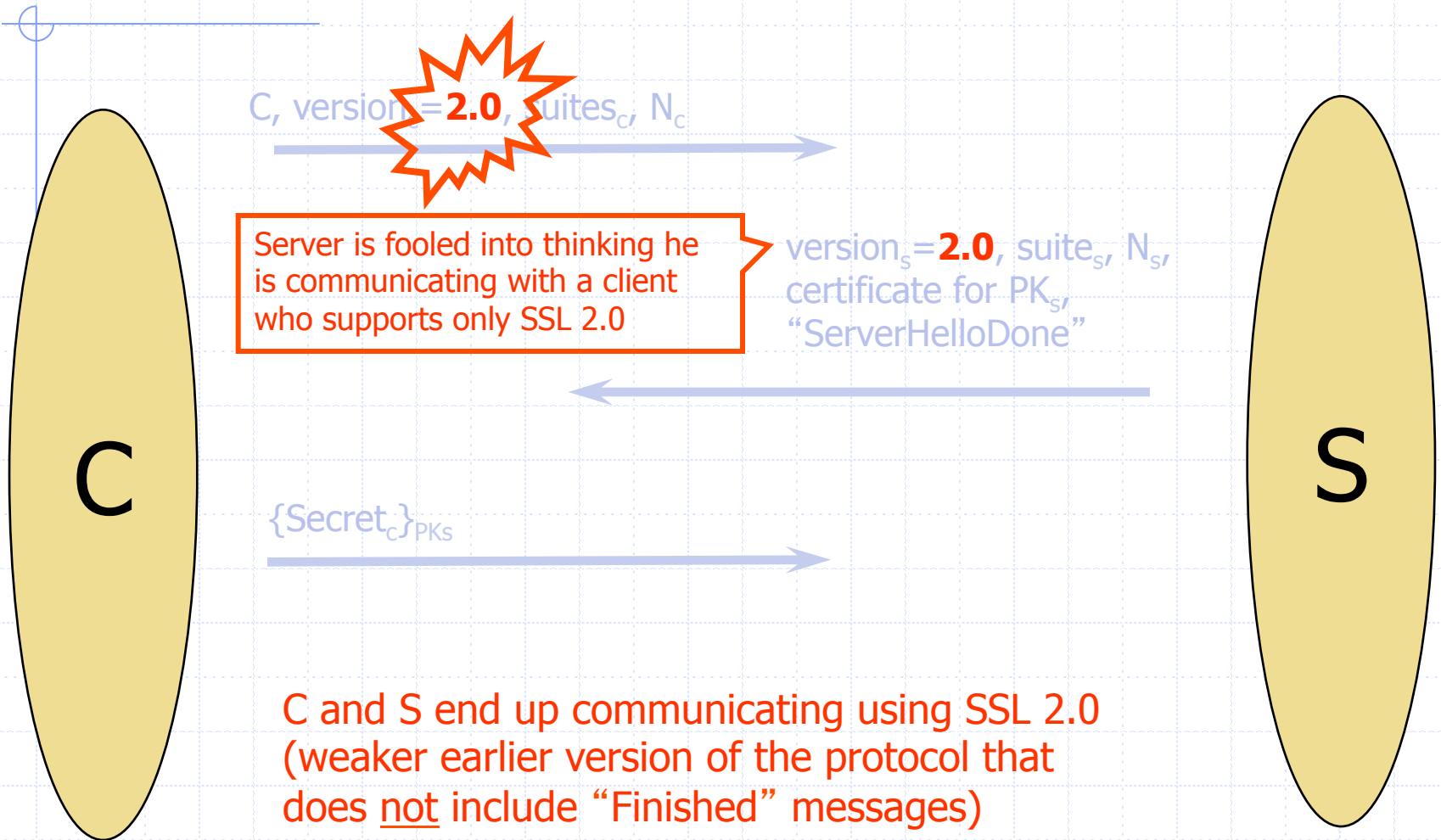


SOURCE: THOMAS, *SSL AND TLS ESSENTIALS*

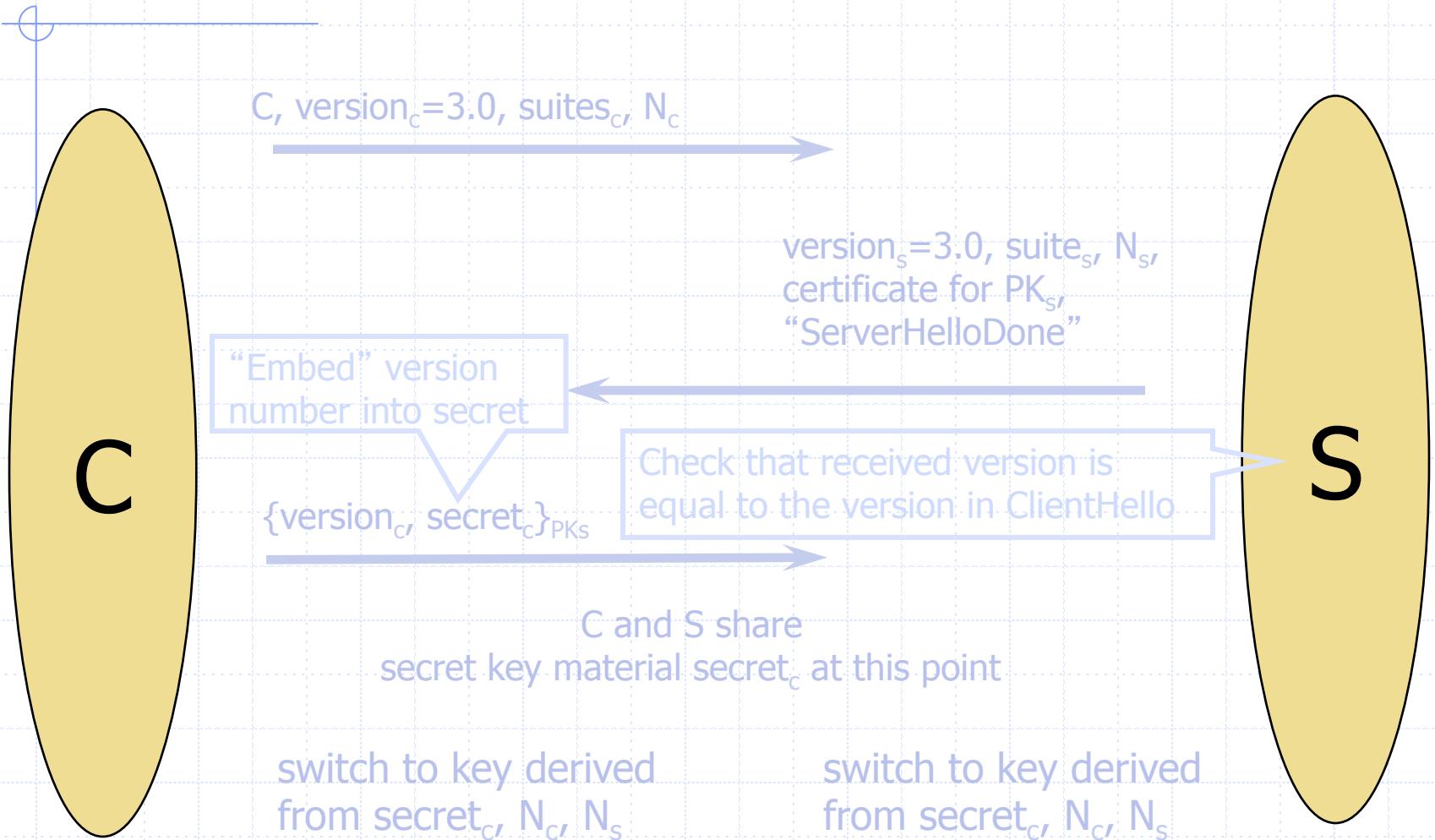
SSL 2.0 Weaknesses (Fixed in 3.0)

- ◆ Cipher suite preferences are not authenticated
 - “Cipher suite rollback” attack is possible
- ◆ Weak MAC construction, MAC hash uses only 40 bits in export mode
- ◆ SSL 2.0 uses padding when computing MAC in block cipher modes, but padding length field is not authenticated
 - Attacker can delete bytes from the end of messages
- ◆ No support for certificate chains or non-RSA algorithms

Version Rollback Attack

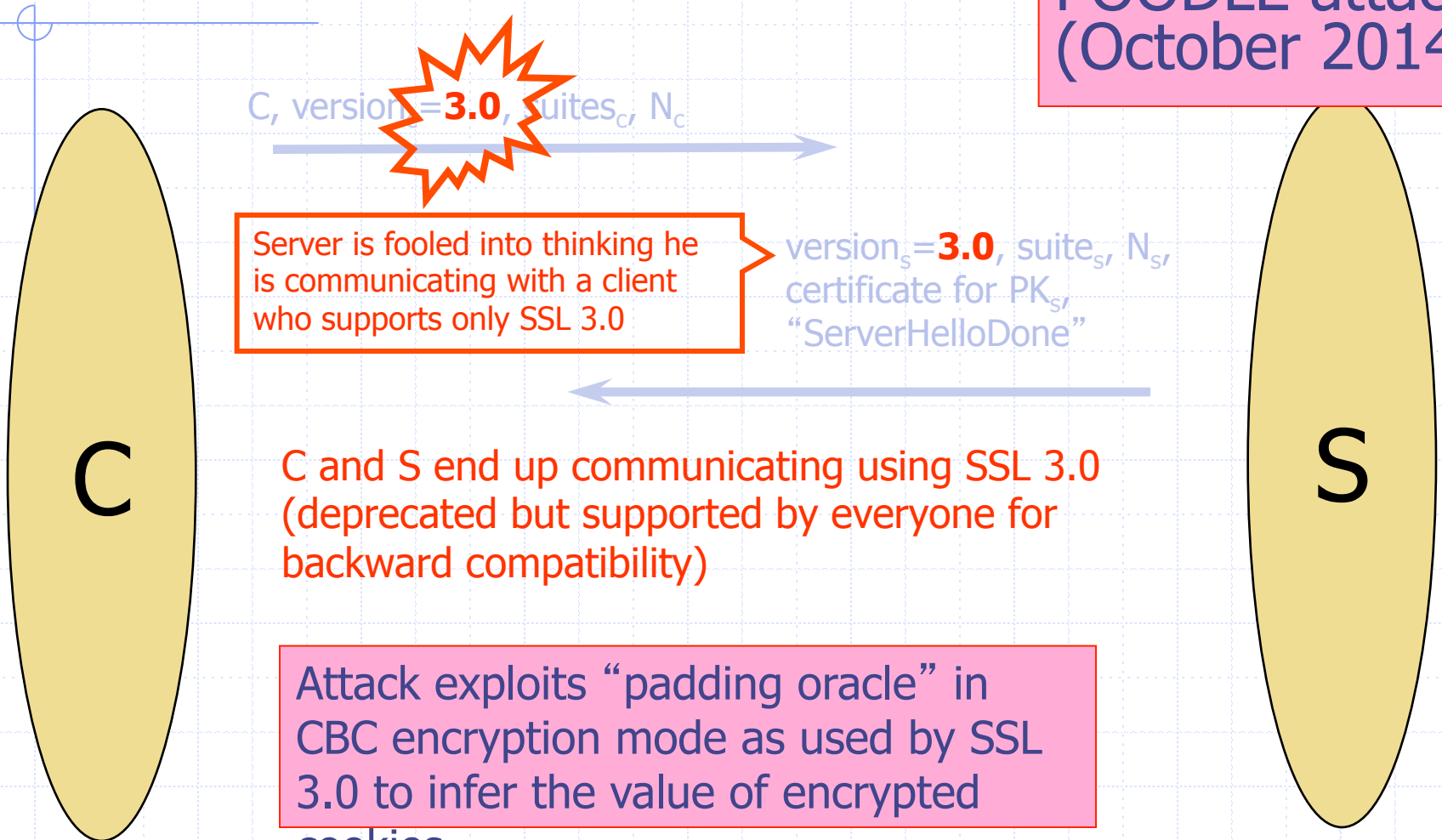


Version Check in SSL 3.0



TLS Version Rollback

POODLE attack
(October 2014)



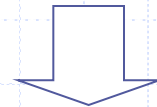
Many "padding oracle" attacks over the years: BEAST, CRIME, ...

“Chosen-Protocol” Attacks

- ◆ Why do people release new versions of security protocols? Because the old version got broken!
- ◆ New version must be **backward-compatible**
 - Not everybody upgrades right away
- ◆ Attacker can fool someone into using the old, broken version and exploit known vulnerabilities
 - Similar: fool victim into using weak crypto algorithms
- ◆ Defense is hard: must authenticate version early
- ◆ Many protocols had “version rollback” attacks
 - SSL, SSH, GSM (cell phones)

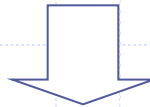
SSL Record Protocol – processing overview

application data



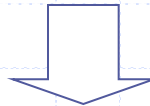
fragmentation

SSLPlaintext



compression

SSLCompressed



msg authentication and encryption (with padding if necessary)

SSLCiphertext



Header

◆ type

- the higher level protocol used to process the enclosed fragment
- possible types:
 - ◆ change_cipher_spec
 - ◆ alert
 - ◆ handshake
 - ◆ application_data

◆ version

- SSL version, currently 3.0

◆ length

- length (in bytes) of the enclosed fragment or compressed fragment
- max value is $2^{14} + 2048$

MAC

MAC = hash(MAC_write_secret | pad_2 |
hash(MAC_write_secret | pad_1 | seq_num | type | length |
fragment))

- ◆ similar to HMAC but the pads are concatenated
- ◆ supported hash functions:
 - MD5
 - SHA-1
- ◆ pad_1 is 0x36 repeated 48 times (MD5) or 40 times (SHA-1)
- ◆ pad_2 is 0x5C repeated 48 times (MD5) or 40 times (SHA-1)

Encryption

◆ supported algorithms

■ block ciphers (in CBC mode)

- ◆ RC2_40
- ◆ DES_40
- ◆ DES_56
- ◆ 3DES_168
- ◆ IDEA_128
- ◆ Fortezza_80

■ stream ciphers

- ◆ RC4_40
- ◆ RC4_128

◆ if a block cipher is used, than padding is applied

- last byte of the padding is the padding length

TLS Heartbeat

A way to keep TLS connection alive without constantly transferring data

If you are alive, send me
this 5-letter word: "xyzyz"

"xyzyz"

C

Per RFC 6520:

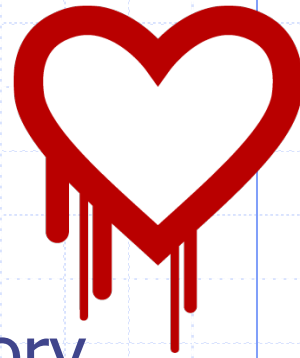
```
struct {  
    HeartbeatMessageType type;  
    uint16 payload_length;  
    opaque payload[HeartbeatMessage.payload_length];  
    opaque padding[padding_length];  
} HeartbeatMessage;
```

OpenSSL omitted to check that this value matches the actual length of the heartbeat message

S



Heartbleed Consequences



- ◆ Attacker can obtain chunks of server memory
 - Passwords, contents of other users' communications, even the server's private RSA key
 - Why is the RSA key still in memory? Long story:

<https://www.lightbluetouchpaper.org/2014/04/25/heartbleed-and-rsa-private-keys/>

- ◆ Assisted by a custom allocator that does not zero out malloc'd memory (for "performance," natch!)

SSL Alert Protocol

- ◆ each alert message consists of 2 fields (bytes)
- ◆ first field (byte): “warning” or “fatal”
- ◆ second field (byte):
 - fatal
 - ◆ unexpected_message
 - ◆ bad_record_MAC
 - ◆ decompression_failure
 - ◆ handshake_failure
 - ◆ illegal_parameter
 - warning
 - ◆ close_notify
 - ◆ no_certificate
 - ◆ bad_certificate
 - ◆ unsupported_certificate
 - ◆ certificate_revoked
 - ◆ certificate_expired
 - ◆ certificate_unknown
- ◆ in case of a fatal alert
 - connection is terminated
 - session ID is invalidated → no new connection can be established within this session

Most Common Use of SSL/TLS

Wells Fargo Account Summary - Microsoft Internet Explorer

Address: https://online.wellsfargo.com/mn1_aa1_on/cgi-bin/session.cgi?sessargs=coAn76ax52xltPX8uoCT8rRBfMMdJldx

Home | Help Center | Contact Us | Locations | Site Map | Apply | Sign Off

Account Summary

Last Log On: January 06, 2004

Wells Fargo Accounts | OneLook Accounts

Tip: Select an account's balance to access the Account History.

NEW [Enroll for Online Statements](#) [My Message Center](#)

Cash Accounts

Account	Account Number	Available Balance
Checking Add Bill Pay		
Total		

To end your session, be sure to Sign Off.

Account Summary | Brokerage | Bill Pay | Transfer | My Message Center | Sign Off
Home | Help Center | Contact Us | Locations | Site Map | Apply

© 1995 - 2003 Wells Fargo. All rights reserved.

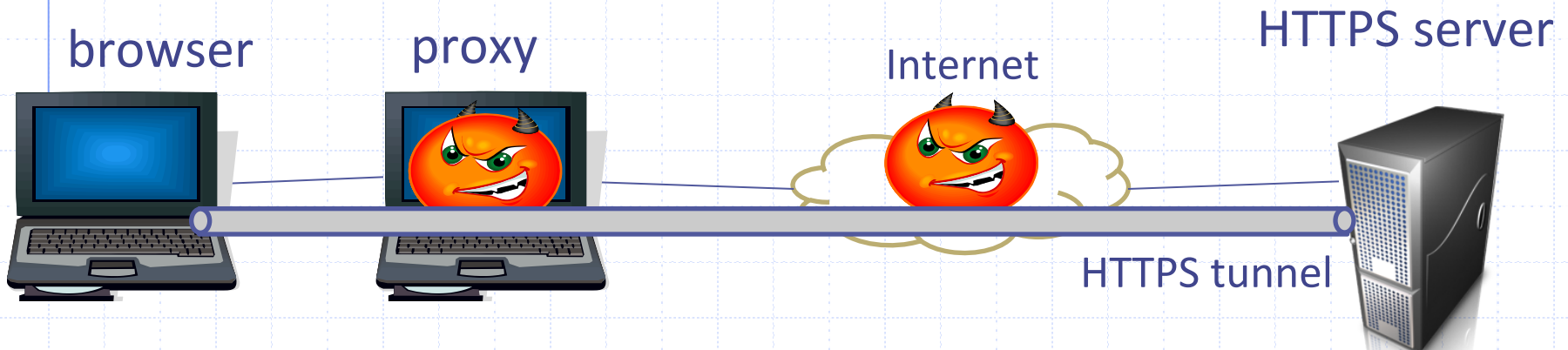
Stay organized with FREE 24/7 access to Online Statements. Sign up today.

Sign up for the Wells Fargo Rewards® program and get 2,500 points. Learn More.



HTTPS and Its Adversary Model

- ◆ HTTPS: **end-to-end** secure protocol for Web
- ◆ Designed to be secure against network attackers, including man-in-the-middle (MITM) attacks



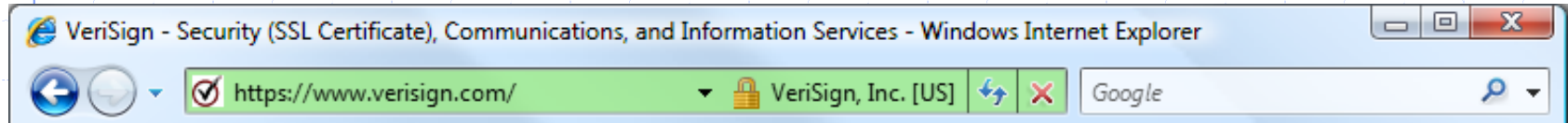
- ◆ HTTPS provides encryption, authentication (usually for server only), and integrity checking

The Lock Icon



- ◆ Goal: identify secure connection
 - SSL/TLS is being used between client and server to protect against active network attacker
- ◆ Lock icon should only be shown when the page is secure against **network attacker**
 - Semantics subtle and not widely understood by users
 - Problem in user interface design

HTTPS Security Guarantees




- ◆ The origin of the page is what it says in the address bar
 - User must interpret what he sees
- ◆ Contents of the page have not been viewed or modified by a network attacker

Evolution of the Lock in Firefox

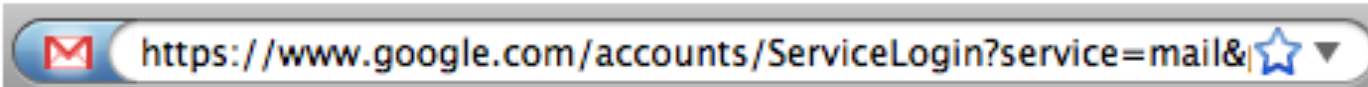
[Schultze]


Firefox 3.6



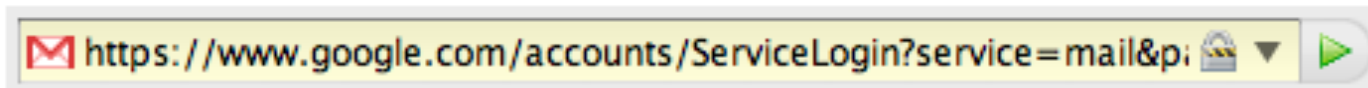
bottom-right corner of browser window (status bar): 


Firefox 3.0



bottom-right corner of browser window:  www.google.com

Firefox 2.0

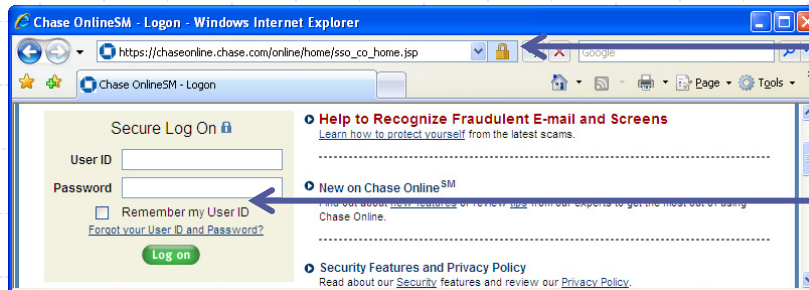


bottom-right corner of browser window:  www.google.com

Combining HTTPS and HTTP

◆ Page served over HTTPS but contains HTTP

- IE 7: no lock, "mixed content" warning
- Firefox: "!" over lock, no warning by default
- Safari: does not detect mixed content



Lock icon

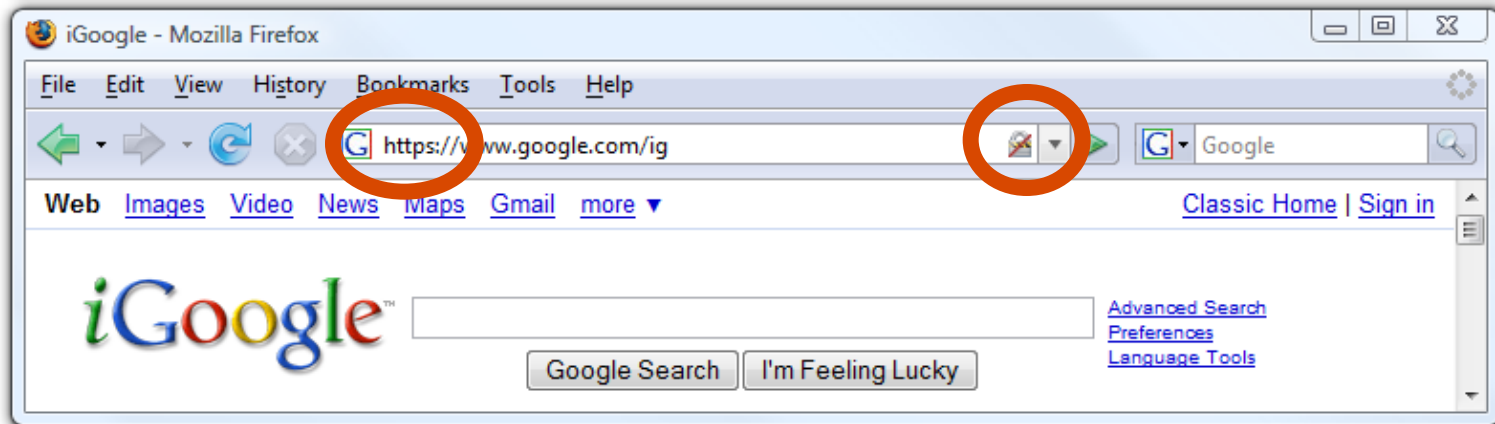
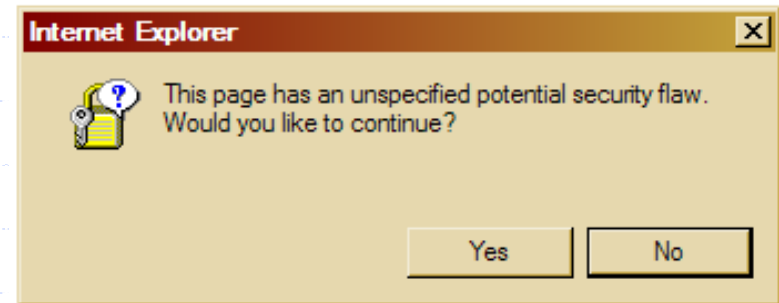
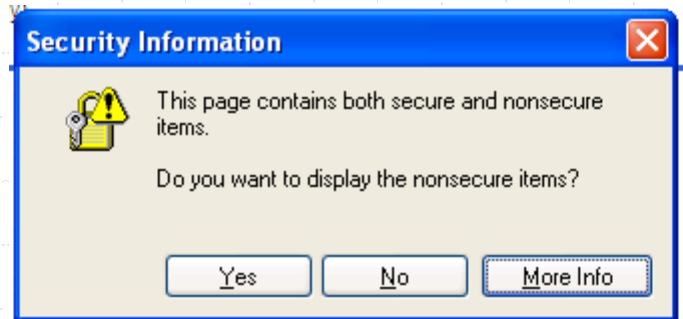
Flash file served over HTTP

- Flash does not trigger warning in IE7 and FF

◆ Network attacker can now inject scripts, hijack session

Can script embedding page!

Mixed Content: UI Challenges



Mixed Content and Network Attacks

◆ Banks: after login, all content served over HTTPS

◆ Developer error: somewhere on bank site write

```
<script src=http://www.site.com/script.js> </script>
```

- Active network attacker can now hijack any session (how?)

◆ Better way to include content:

```
<script src=//www.site.com/script.js> </script>
```

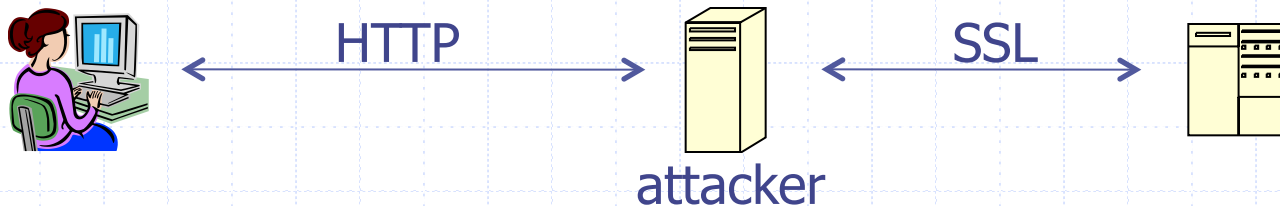
- Served over the same protocol as embedding page

HTTP → HTTPS and Back

◆ Typical pattern: HTTPS upgrade

- Come to site over HTTP, redirect to HTTPS for login
- Browse site over HTTP, redirect to HTTPS for checkout

◆ **sslstrip**: network attacker downgrades connection

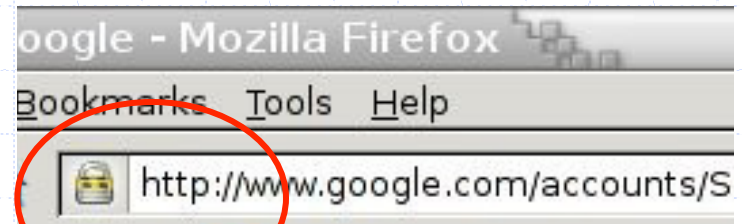
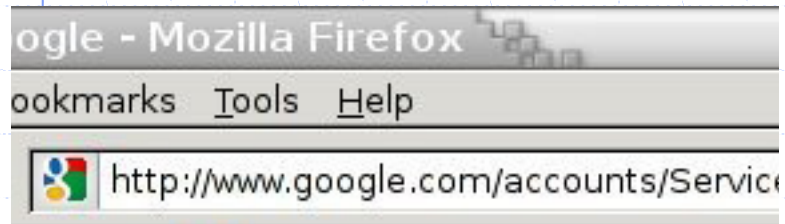


- Rewrite `` to ``
- Redirect Location: `https://...` to Location: `http://...`
- Rewrite `<form action=https://... >` to `<form action=http://...>`

Can the server detect this attack?

Will You Notice?

[Moxie Marlinspike]



Clever favicon inserted
by network attacker

Motivation

https://

The screenshot shows the Wells Fargo website in Internet Explorer. The address bar displays `https://www.wellsfargo.com/`. A red box highlights the `https://` part of the URL. A red arrow points from the lock icon in the address bar to a separate lock icon on the right. The website content includes a navigation menu with 'Personal', 'Small Business', 'Commercial', and 'About Us'. A large green banner reads 'Checking and much more' with the subtext 'Choose a Wells Fargo Checking Package®'. Below this, there are sections for 'Savings & CDs', 'Personal Loans', 'Mutual Funds', 'Open an Account', 'Check Today's Rates', and 'Money management help'. A pink box is overlaid on the page with the text: 'Whose public key is used to establish the secure session?'.

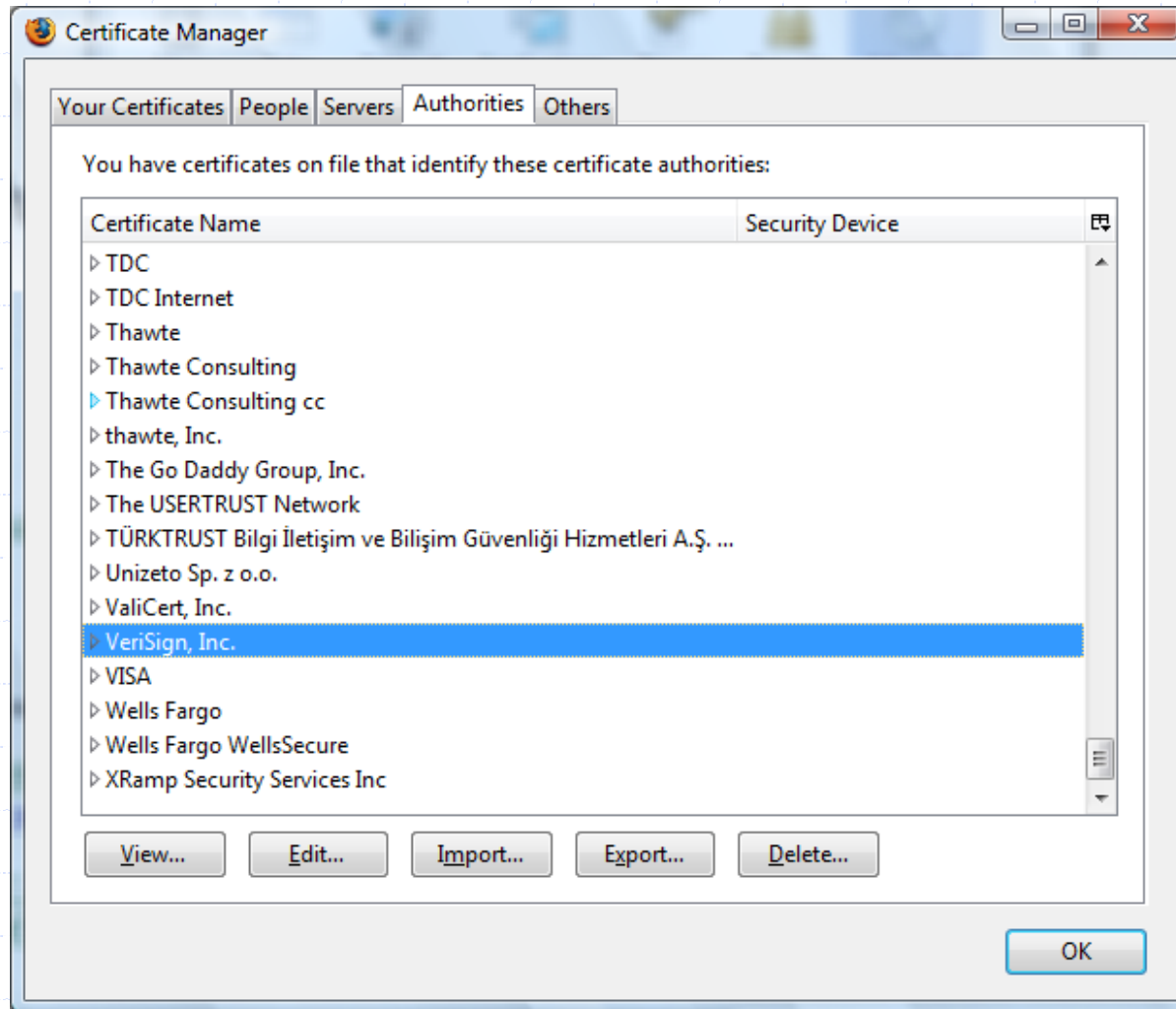
Whose public key is used to establish the secure session?



Distribution of Public Keys

- ◆ Public announcement or public directory
 - Risks: forgery and tampering
- ◆ Public-key certificate
 - Signed statement specifying the key and identity
 - ◆ $\text{sig}_{\text{Alice}}(\text{"Bob"}, \text{PK}_B)$
- ◆ Common approach: certificate authority (CA)
 - An agency responsible for certifying public keys
 - Browsers are pre-configured with 100+ of trusted CAs
 - A public key for any website in the world will be accepted by the browser if certified by one of these CAs

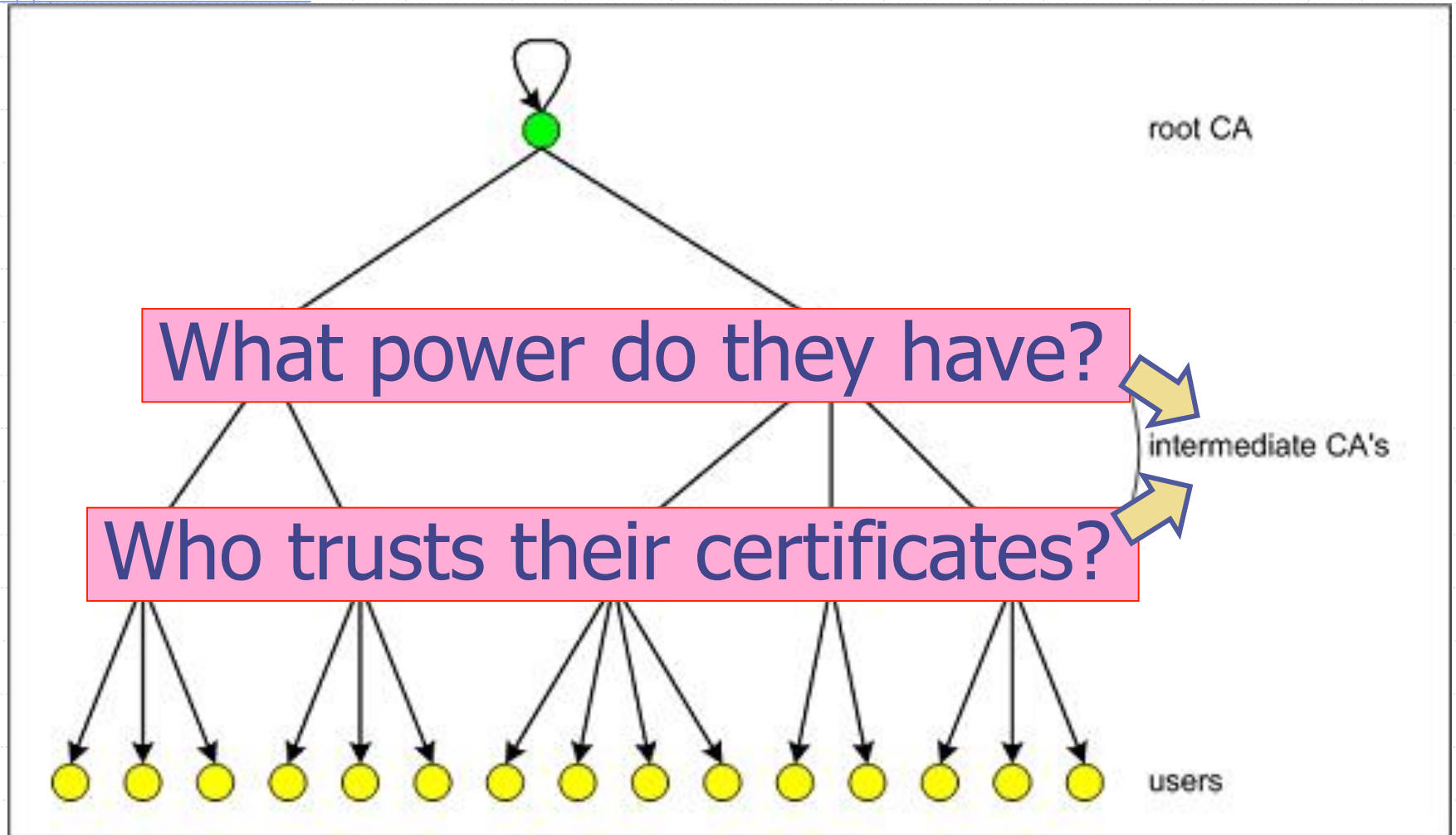
Trusted Certificate Authorities



CA Hierarchy

- ◆ Browsers, operating systems, etc. have trusted **root certificate authorities**
 - Firefox 3 includes certificates of 135 trusted root CAs
- ◆ A Root CA signs certificates for intermediate CAs, they sign certificates for lower-level CAs, etc.
 - Certificate **"chain of trust"**
 - ◆ $\text{sig}_{\text{Verisign}}(\text{"UT Austin"}, \text{PK}_{\text{UT}}), \text{sig}_{\text{UT}}(\text{"Vitaly S."}, \text{PK}_{\text{Vitaly}})$
- ◆ CA is responsible for verifying the identities of certificate requestors, domain ownership

Certificate Hierarchy



Example of a Certificate

Important fields

Certificate Signature Algorithm

Issuer

▲ Validity

Not Before

Not After

Subject

▲ Subject Public Key Info

Subject Public Key Algorithm

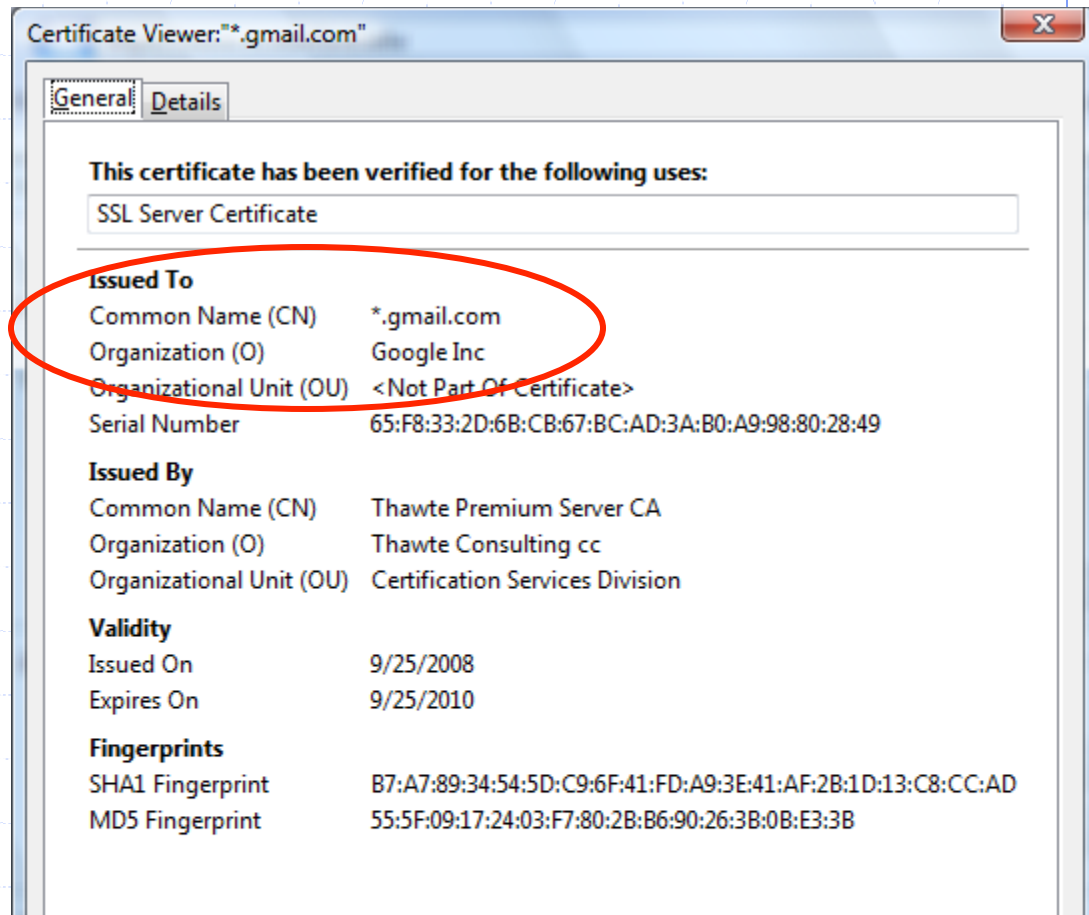
Subject's Public Key

▲ Extensions

Field Value

Modulus (1024 bits):

```
ac 73 14 97 b4 10 a3 aa f4 c1 15 ed cf 92 f3 9a
97 26 9a cf 1b e4 1b dc d2 c9 37 2f d2 e6 07 1d
ad b2 3e f7 8c 2f fa a1 b7 9e e3 54 40 34 3f b9
e2 1c 12 8a 30 6b 0c fa 30 6a 01 61 e9 7c b1 98
2d 0d c6 38 03 b4 55 33 7f 10 40 45 c5 c3 e4 d6
6b 9c 0d d0 8e 4f 39 0d 2b d2 e9 88 cb 2d 21 a3
f1 84 61 3c 3a aa 80 18 27 e6 7e f7 b8 6a 0a 75
e1 bb 14 72 95 cb 64 78 06 84 81 eb 7b 07 8d 49
```



Common Name

- ◆ Explicit name: `www.foo.com`
- ◆ Wildcard: `*.foo.com` or `www*.foo.com`
- ◆ Matching rules
 - Firefox 3: `*` matches anything
 - Internet Explorer 7: `*` must occur in the leftmost component, does not match `\.`
 - ◆ `*.foo.com` matches `a.foo.com`, but not `a.b.foo.com`

International Domain Names

- ◆ Rendered using international character set
- ◆ Chinese character set contains characters that look like / ? = .
 - What could go wrong?
- ◆ Can buy a certificate for *.foo.cn, create any number of domain names that look like www.bank.com/accounts/login.php?q=me.foo.cn
 - What does the user see?
 - *.foo.cn certificate works for all of them!

Example

[Moxie Marlinspike]

Personal Banking - PNC Bank - Mozilla Firefox

File Edit View History Bookmarks Tools Help

https://www.pnc.com/webapp/unsec/homepag

Most Visited Getting Started Latest Headlines

PNC
LEADING THE WAY

HOME SECURITY ASSURANCE LOCATE PNC CONTACT US CUSTOMER SERVICE

Search

PERSONAL SMALL BUSINESS CORPORATE & INSTITUTIONAL ABOUT PNC

Online Banking Sign On

User ID: **SIGN ON**

▶ Forgot Your User ID or Password?

New to Online Banking? ▶ Learn More
▶ Get Started Now! ▶ View Demo

Sign On to Other Services:
Select Service

PNC Security Assurance Products and Services Solutions

Important FDIC Information
PNC Bank is participating in the FDIC's Transaction Account Guarantee Program. more ▶

Two of America's best-known banks. Now simply one of America's best.
Making the transition to PNC as easy as possible for you.

PNC's wide range of services can make banking easier, and more convenient than ever. See why PNC's the smart choice for help in meeting your financial goals.

- ▶ Online Banking and Bill Pay
- ▶ Checking
- ▶ Savings
- ▶ Loans and Lines of Credit
- ▶ Cards

Whatever challenges and opportunities lie ahead, PNC can help. See why working with PNC to plan for life's greatest milestones is the smart choice.

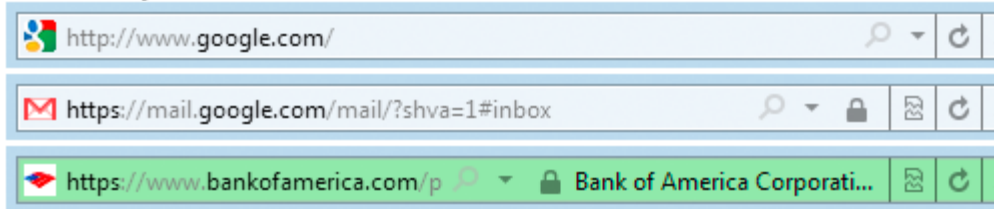
- ▶ Making the Most of Your Money
- ▶ Virtual Wallet
- ▶ Planning for Retirement
- ▶ Saving for Education
- ▶ Buying a Home

Done www.pnc.com/webapp/unsec/homepage.var.cn

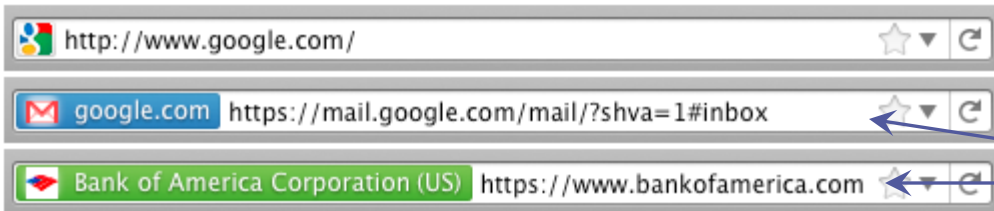
Meaning of Color

[Schultze]

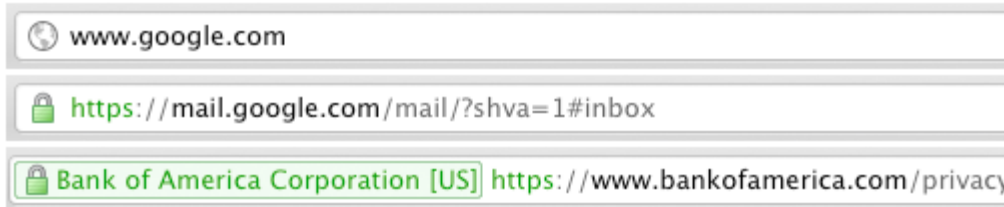
Internet Explorer 9



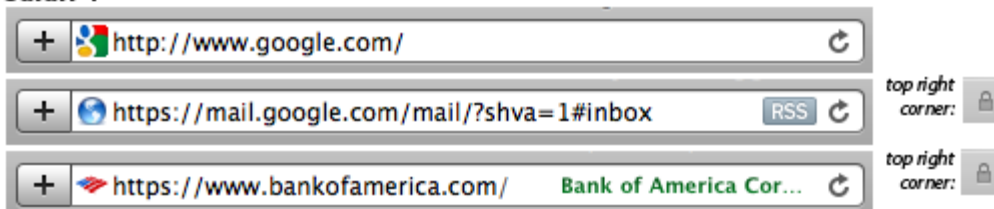
Firefox 4



Chrome 8



Safari 4



What is the difference?

Domain Validation (DV)
certificate

VS.

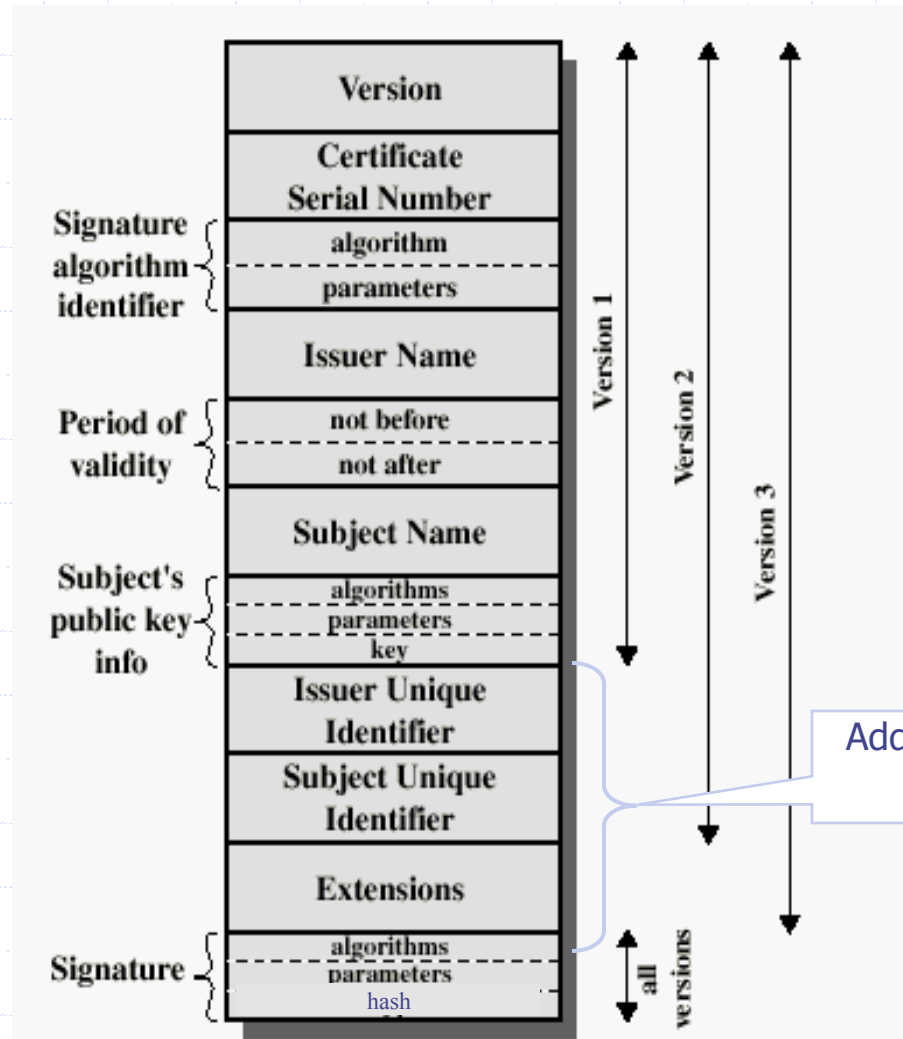
Extended Validation (EV)
certificate

Means what?

X.509 Authentication Service

- ◆ Internet standard (1988-2000)
- ◆ Specifies certificate format
 - X.509 certificates are used in IPsec and SSL/TLS
- ◆ Specifies certificate directory service
 - For retrieving other users' CA-certified public keys
- ◆ Specifies a set of authentication protocols
 - For proving identity using public-key signatures
- ◆ Can use with any digital signature scheme and hash function, but must hash before signing

X.509 Certificate



Added in X.509 versions 2 and 3 to address usability and security problems

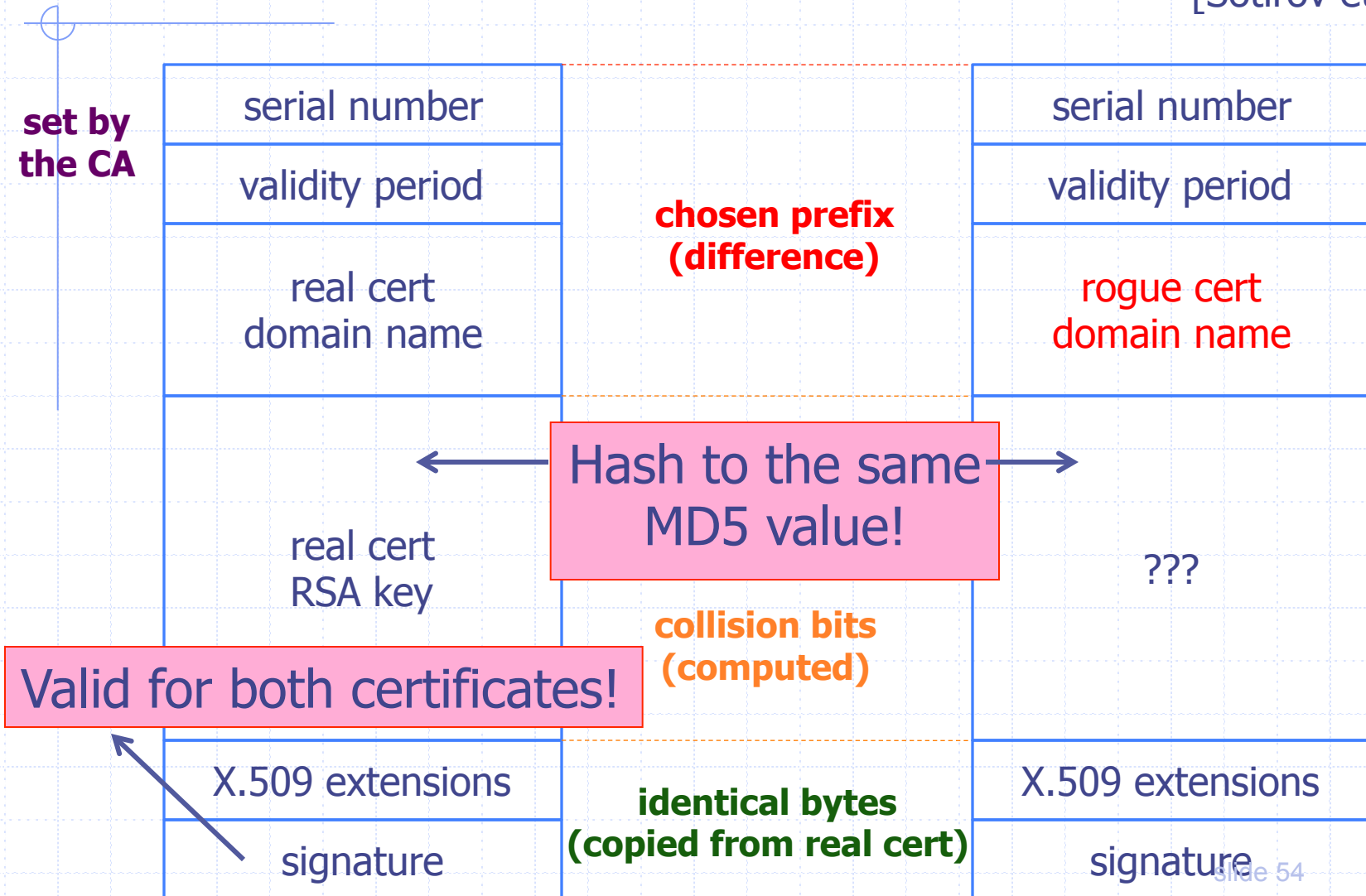
Back in 2008

[Sotirov et al. “MD5 Considered Harmful Today: Creating a Rogue CA Certificate”]

- ◆ Many CAs still used MD5
 - RapidSSL, FreeSSL, TrustCenter, RSA Data Security, Thawte, verisign.co.jp
- ◆ Sotirov et al. collected 30,000 website certificates
- ◆ 9,000 of them were signed using MD5 hash
- ◆ 97% of those were issued by RapidSSL

Colliding Certificates

[Sotirov et al.]



Generating Collisions

[Sotirov et al.]

1-2 days on a cluster of
200 PlayStation 3's

Equivalent to 8000
desktop CPU cores or
\$20,000 on Amazon EC2



Generating Colliding Certificates

[Sotirov et al.]

- ◆ RapidSSL uses a fully automated system
 - \$69 for a certificate, issued in 6 seconds
 - Sequential serial numbers
- ◆ Technique for generating colliding certificates
 - Get a certificate with serial number S
 - Predict time T when RapidSSL's counter goes to $S+1000$
 - Generate the collision part of the certificate
 - Shortly before time T buy enough (non-colliding) certificates to increment the counter to $S+999$
 - Send colliding request at time T and get serial number $S+1000$

Creating a Fake Intermediate CA

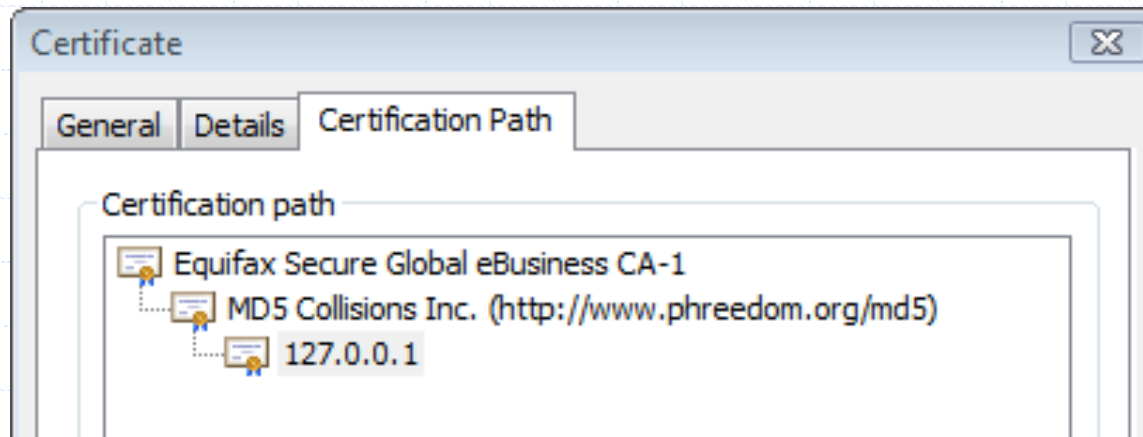
[Sotirov et al.]

serial number		
validity period		rogue CA cert
real cert domain name	chosen prefix (difference)	rogue CA RSA key
		rogue CA X.509 extensions ← CA bit!
real cert RSA key	collision bits (computed)	Netscape Co Extensio (contents ignored by browsers)
X.509 extensions		
signature	identical bytes (copied from real cert)	signature

We are now an intermediate CA. WOOT!

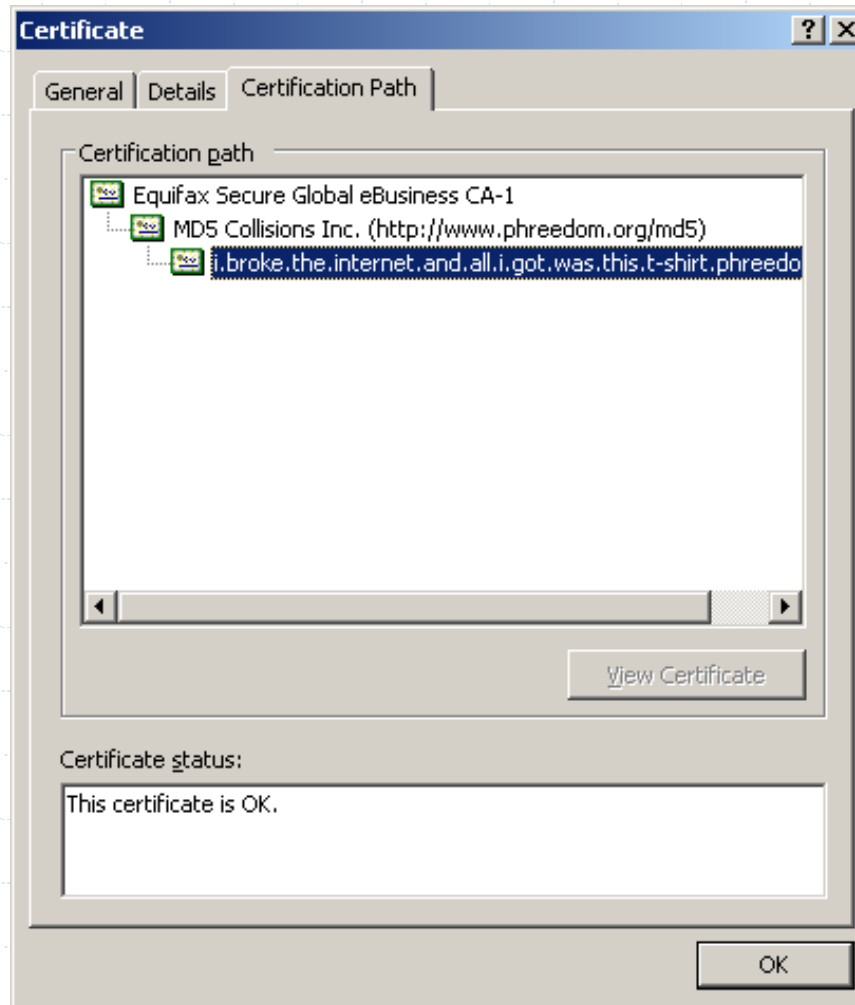
Result: Perfect Man-in-the-Middle

- ◆ This is a “skeleton key” certificate: it can issue fully trusted certificates for any site (why?)



- ◆ To take advantage, need a network attack
 - Insecure wireless, DNS poisoning, proxy auto-discovery, hacked routers, etc.

A Rogue Certificate



SSLint

- Certificate chain validation
- Server domain name / hostname validation
- Our findings:
 - ✓ We detected **27** previous unknown vulnerable apps out of **485** Ubuntu apps.
 - ✓ All vulnerabilities fall into the two categories mentioned above.

OpenSSL API

```
const SSL_METHOD *method;
SSL_CTX *ctx;
SSL *ssl;

...
//select protocol
method = TLSv1_client_method();

...
//Create CTX
ctx = SSL_CTX_new(method);

...
//Create SSL
ssl = SSL_new(ctx);

...
/*set SSL_VERIFY_PEER flag to
Enforce certificate chain
validation during handshake*/
SSL_CTX_set_verify(ctx,
SSL_VERIFY_PEER,...);

...
//Start SSL handshake
SSL_connect(ssl);

...
```

```
const SSL_METHOD *method;
SSL_CTX *ctx;
SSL *ssl;
X509 *cert = NULL;

...
//select protocol
method = TLSv1_client_method();

...
//Create CTX
ctx = SSL_CTX_new(method);

...
//Create SSL
ssl = SSL_new(ctx);

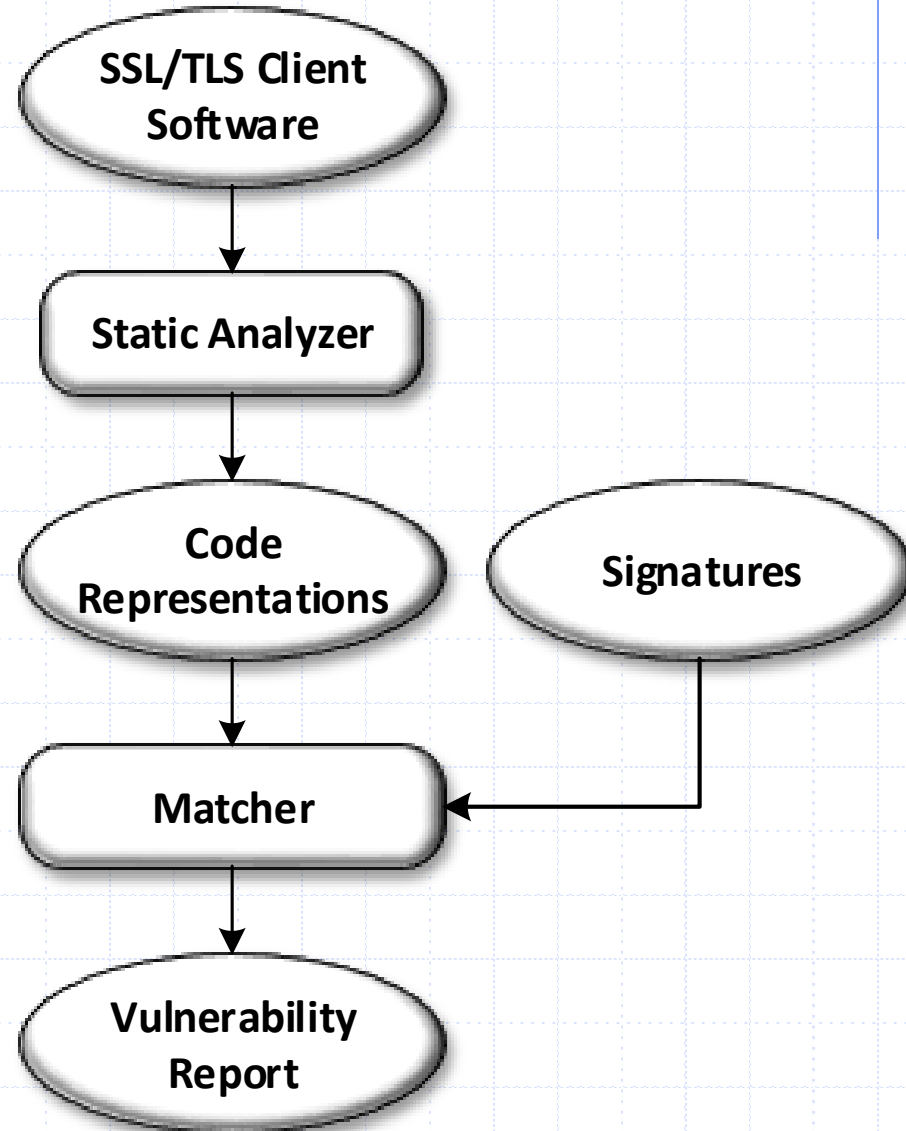
...
//Start handshake
SSL_connect(ssl);

...
cert = SSL_get_peer_certificate(ssl);
if (cert != NULL){
    if(SSL_get_verify_result(ssl)
        == X509_V_OK){
        //Validation succeeds.
    }else{
        //Validation fails and terminate connecti
    }
}
else{
    //Validation fails and terminate connecti
}
```

Solution

- Check whether validation APIs are called correctly
- Encode "correct" usage in a signature and match this signature

Pass if match succeeds



Code Representations & Signatures

- Simple pattern matching (e.g., regular expressions) not sufficient
- APIs are connected by parameters and return values

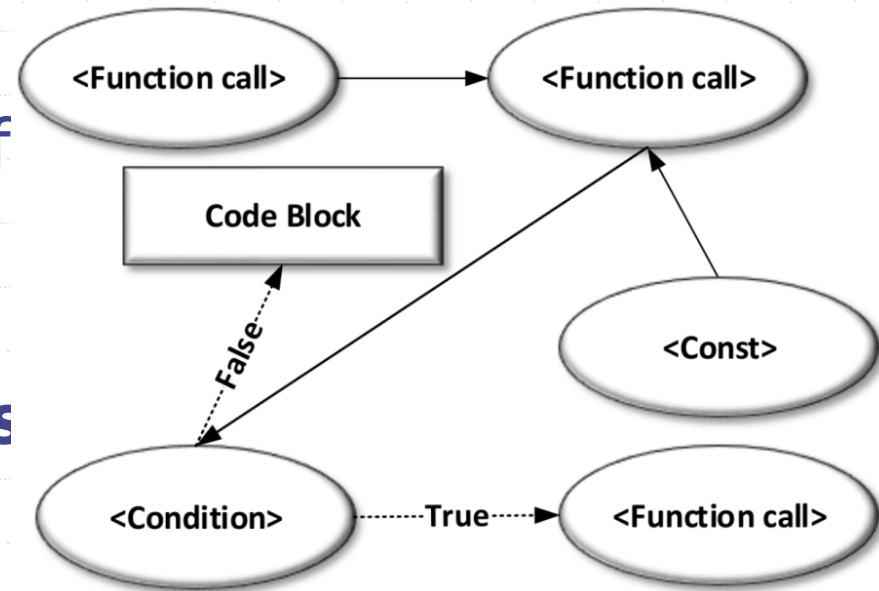
Need to track data flow

- Need to check API call sequences

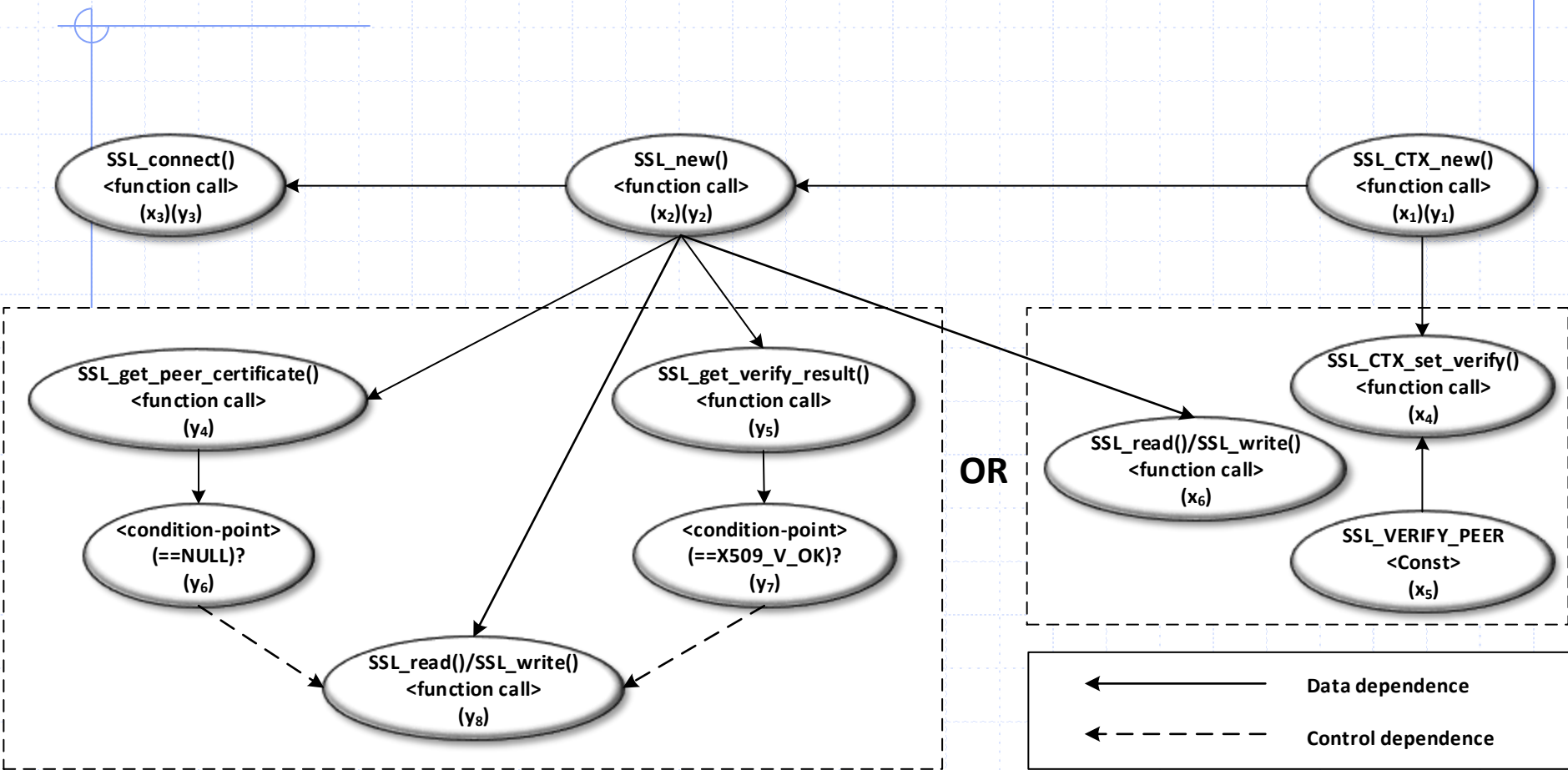
Need to track control flow

Code Representations & Signatures

- **Program dependence graphs (PDGs)**
 - Nodes are program statements
 - Edges are control and data dependencies
 - A is control dependent on B if B can directly affect A's execution
 - A is data dependent on B if value assigned in B can be referenced from A
- **A signature matches nodes and edges of a PDG**



Signature for OpenSSL



SSLint Implementation

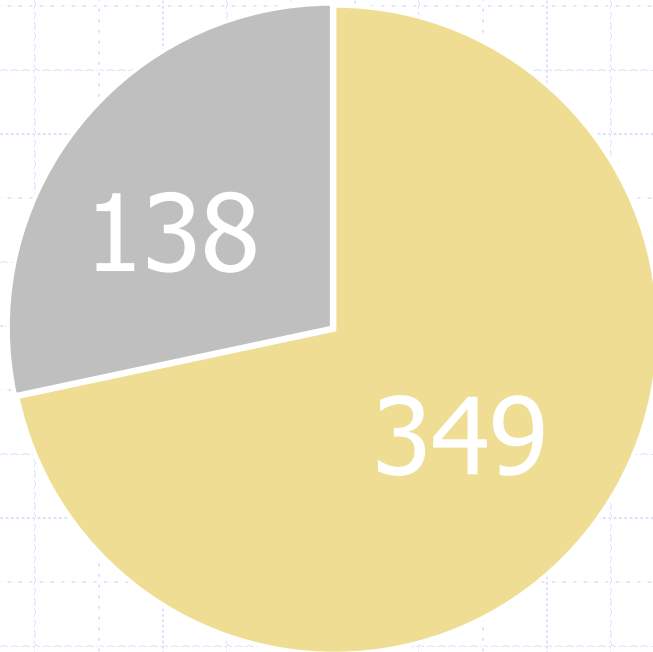
- Certificate Validation Vulnerability Scanner
- CodeSurfer provides static analysis
- Generated PDGs matched with signatures
 - Signature Expressions motivated from Cypher, a graph query language
 - Custom algorithm to perform the matches

Evaluation

- Signatures implemented for OpenSSL and GnuTLS
 - the most popular two SSL/TLS libraries
- Scanned the entire Ubuntu distribution
 - 485 applications using OpenSSL and GnuTLS
- Detected 27 vulnerabilities
 - All reported
 - Many fixed or acknowledged

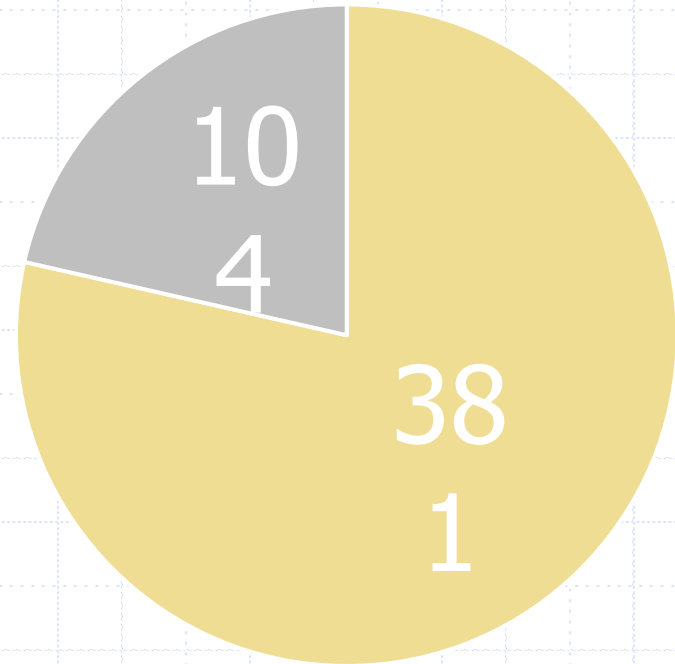
Evaluation

SSL/TLS apps in Ubuntu 12.04



■ OpenSSL app ■ GnuTLS app

Analysis Coverage



■ App successfully analyzed
■ App failed to analyze

Results

- **Vulnerable E-mail Software:**
 - *Xfce4-Mailwatch-Plugin, Mailfilter, Exim, DragonFly Mail Agent, spamc*
- **Vulnerable IRC Software:**
 - *Enhanced Programmable ircII client (EPIC), Scrollz*
- **Other Vulnerable Software:**
 - Web(https): Prayer front end, xxxterm*
 - Database: FreeTDS*
 - Admin tool: nagircbot, nagios-nrpe-plugin, syslog-ng*
 - Performance testing tool: siege, httpperf, httping*

Results

App Name	LoC	Vulnerability Type	SSL library	Dynamic Auditing	Developer Feedback
dma	12,504	Certificate Validation	OpenSSL	Proved	Confirmed
exim4	94,874	Hostname Validation	OpenSSL GnuTLS	Proved	Fixed
xfce4-mailwatch-plugin	9,830	Certificate Validation Hostname Validation	GnuTLS	Proved	
spamc	5,472	Certificate Validation	OpenSSL		Confirmed
prayer	45,555	Certificate Validation	OpenSSL		Confirmed
epic4	56,168	Certificate Validation	OpenSSL	Proved	Fixed
epic5	65,155	Certificate Validation	OpenSSL	Proved	Fixed
scrollz	78,390	Certificate Validation Hostname Validation	OpenSSL GnuTLS	Proved	Confirmed
xxxterm	23,126	Hostname Validation	GnuTLS	Proved	Confirmed
httping	1,400	Certificate Validation	OpenSSL	Proved	Confirmed
pavuk	51,781	Certificate Validation	OpenSSL		Confirmed
crtmpserver5	57,377	Certificate Validation	OpenSSL		Confirmed
freetds-bin	80,203	Certificate Validation Hostname Validation	GnuTLS	Proved	Confirmed

Results

App Name	LoC	Vulnerability Type	SSL library	Dynamic Auditing	Developer Feedback
nagircbot	3,307	Certificate Validation	OpenSSL	Proved	
picolisp	14,250	Certificate Validation	OpenSSL		Fixed
nagios-nrpe-plugin	3,145	Certificate Validation	OpenSSL		Confirmed
citadel-client	56,866	Certificate Validation	OpenSSL	Proved	
mailfilter	4,773	Certificate Validation	OpenSSL	Proved	
suck	12,083	Certificate Validation	OpenSSL	Proved	
proxytunnel	2,043	Certificate Validation Hostname Validation	GnuTLS	Proved	
siege	8,581	Certificate Validation	OpenSSL	Proved	
httperf	6,692	Certificate Validation	OpenSSL	Proved	
syslog-ng	115,513	Certificate Validation	OpenSSL	Proved	
medusa	18,811	Certificate Validation	OpenSSL	Proved	
hydra	23,839	Certificate Validation	OpenSSL	Proved	
ratproxy	4,069	Certificate Validation	OpenSSL	Proved	
dsniff	24,625	Certificate Validation	OpenSSL	Proved	

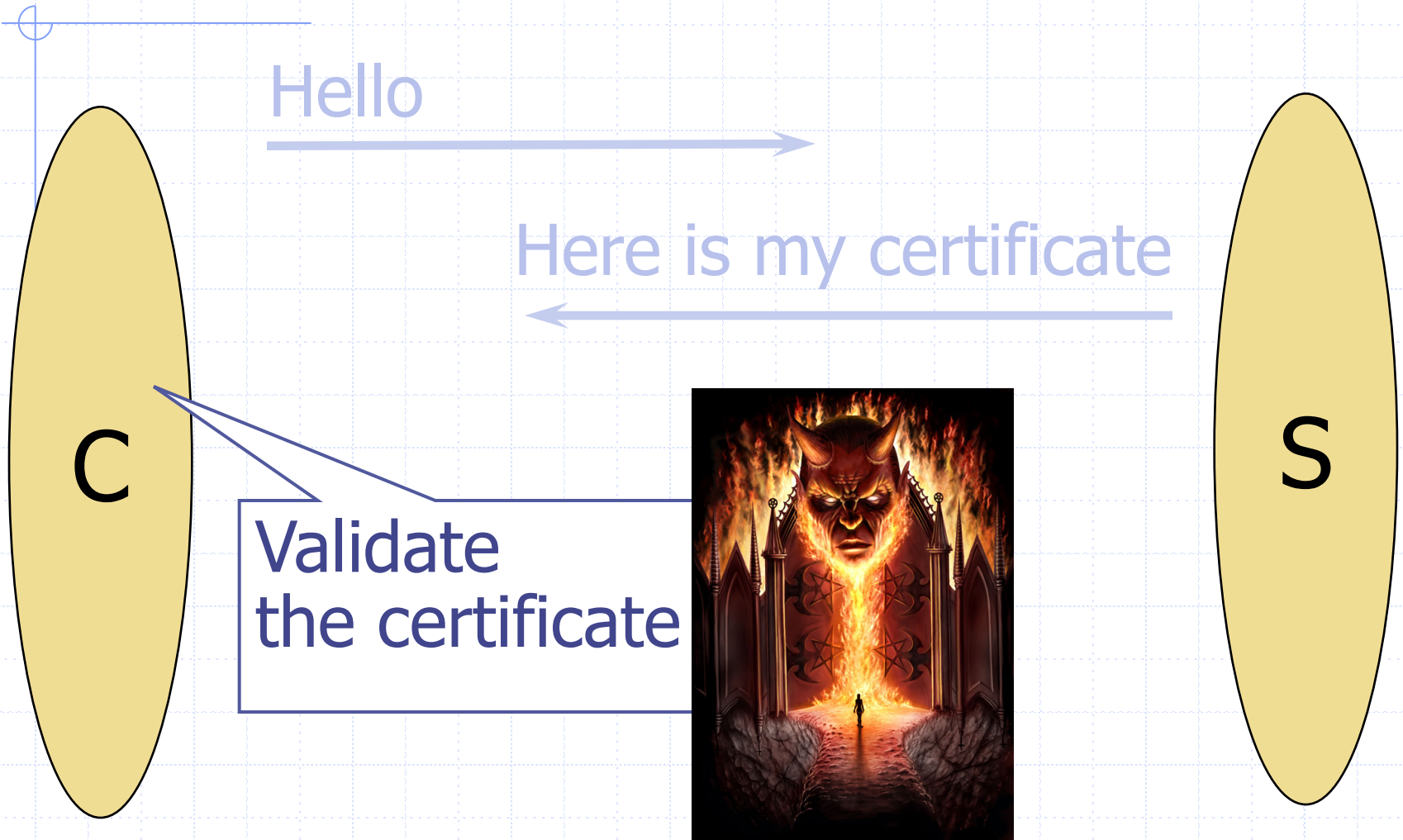


Backup Slides

Flame

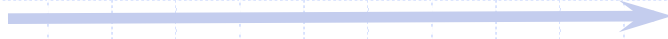
- ◆ Cyber-espionage virus (2010-2012)
- ◆ Signed with a fake intermediate CA certificate that appears to be issued by Microsoft and thus accepted by any Windows Update service
 - Fake intermediate CA certificate was created using an MD5 chosen-prefix collision against an obscure Microsoft Terminal Server Licensing Service certificate that was enabled for **code signing** and still used MD5
- ◆ MD5 collision technique possibly pre-dates Sotirov et al.'s work
 - Evidence of state-level cryptanalysis?

SSL/TLS Handshake

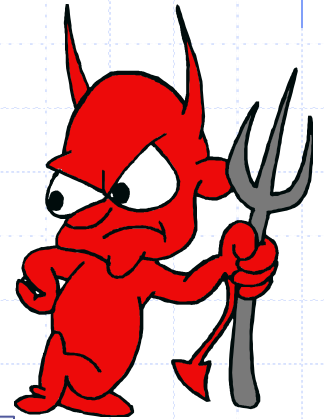


SSL/TLS Handshake

Hello



I am Chase.com
Here is my certificate



Issued by GoDaddy to
AllYourSSLAreBelongTo.us

CHASE 

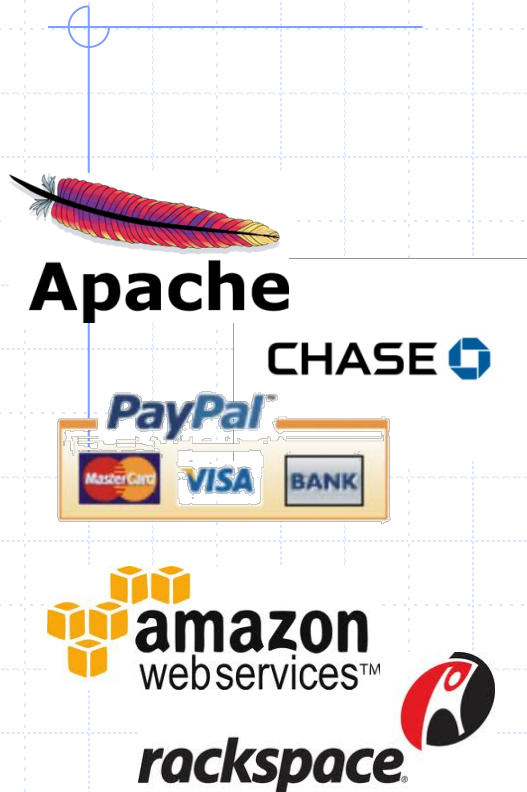
Android
app

Ok!



DAFUQ

Failing to Check Hostname



“Researchers at the University of Texas at Austin and Stanford University have discovered that poorly designed APIs used in SSL implementations are to blame for vulnerabilities in many critical non-browser software packages. Serious security vulnerabilities were found in programs such as Amazon’s EC2 Java library, Amazon’s and PayPal’s merchant SDKs, Trillian and AIM instant messaging software, popular integrated shopping cart software packages, Chase mobile banking software, and several Android applications and libraries. **SSL connections from these programs and many others are vulnerable to a man in the middle attack...**”

- Threatpost (Oct 2012)

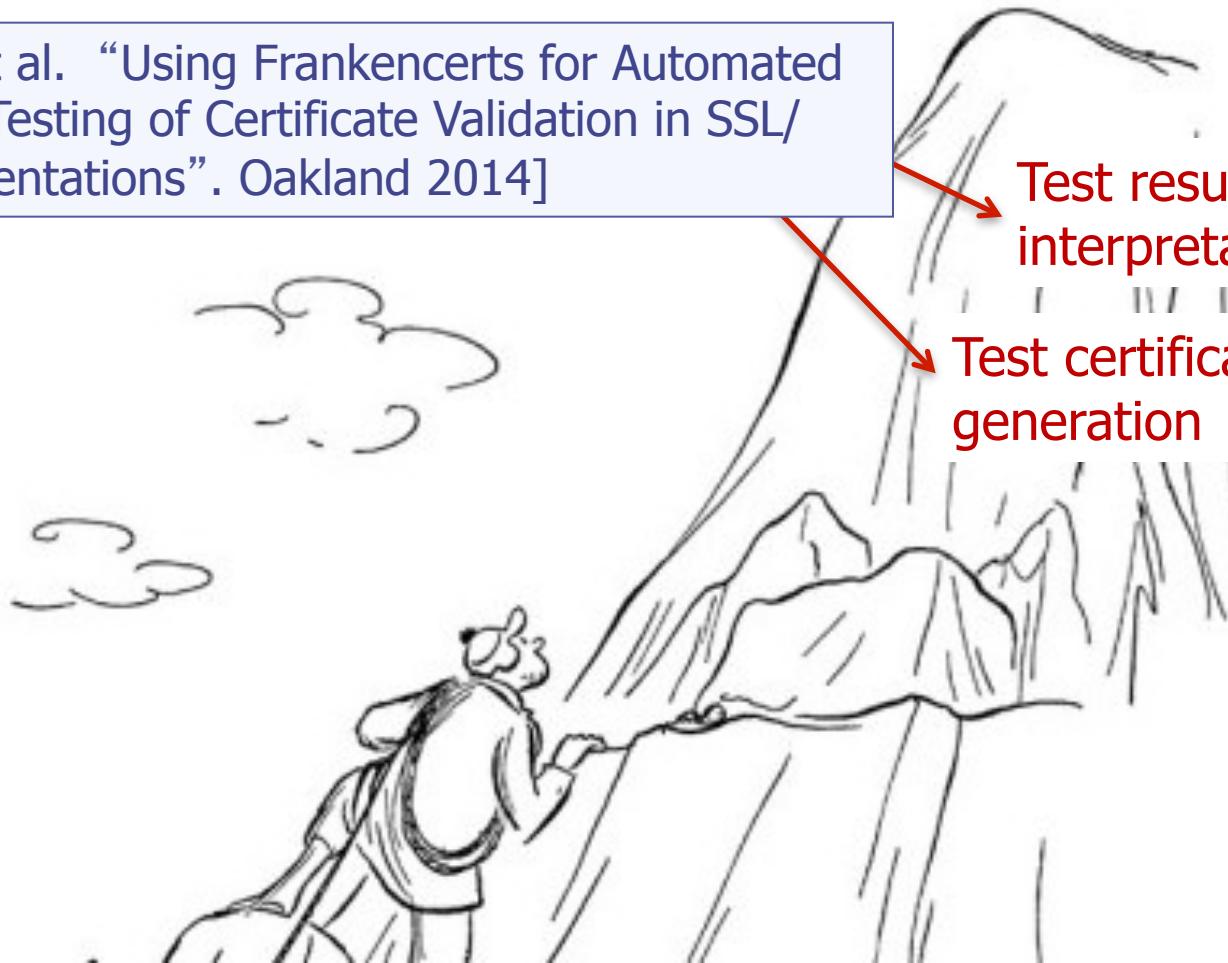
Major payment processing gateways,
client software for cloud computing,
integrated e-commerce software, etc.

Testing Certificate Validation Code

[Brubaker et al. “Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations”. Oakland 2014]

Test result
interpretation

Test certificate
generation



Generating Test Certificates

◆ Requirements

- Must generate “semantically bad” certificates
- Should be syntactically correct, otherwise will fail during parsing and won’t exercise most of the certificate validation code
- Must scale to millions of certificates

◆ Idea



- X.509 certificates contain structured data, can we exploit this?

X.509 Certificate Structure

- ◆ Multilayered structured data
- ◆ Syntactic constraints for each piece
 - Ex: Version must be an integer
- ◆ Semantic constraints for individual piece or across multiple pieces
 - Ex: Version must be 0, 1, or 2
 - Ex: if version!=2, extensions must be NULL

Version
Serial Number
Signature
Algorithm
Identifier
Issuer Name
Validity Period
Subject Name
Public Key
Information
Issuer Unique ID
Subject Unique ID
Extensions

X.509 Standards... Ugh!



Idea: Random Re-assembly

Create X.509 certs using randomly picked syntactically valid pieces

↑
Likely to violate some semantic constraints and will thus generate “bad” test certs just as we wanted

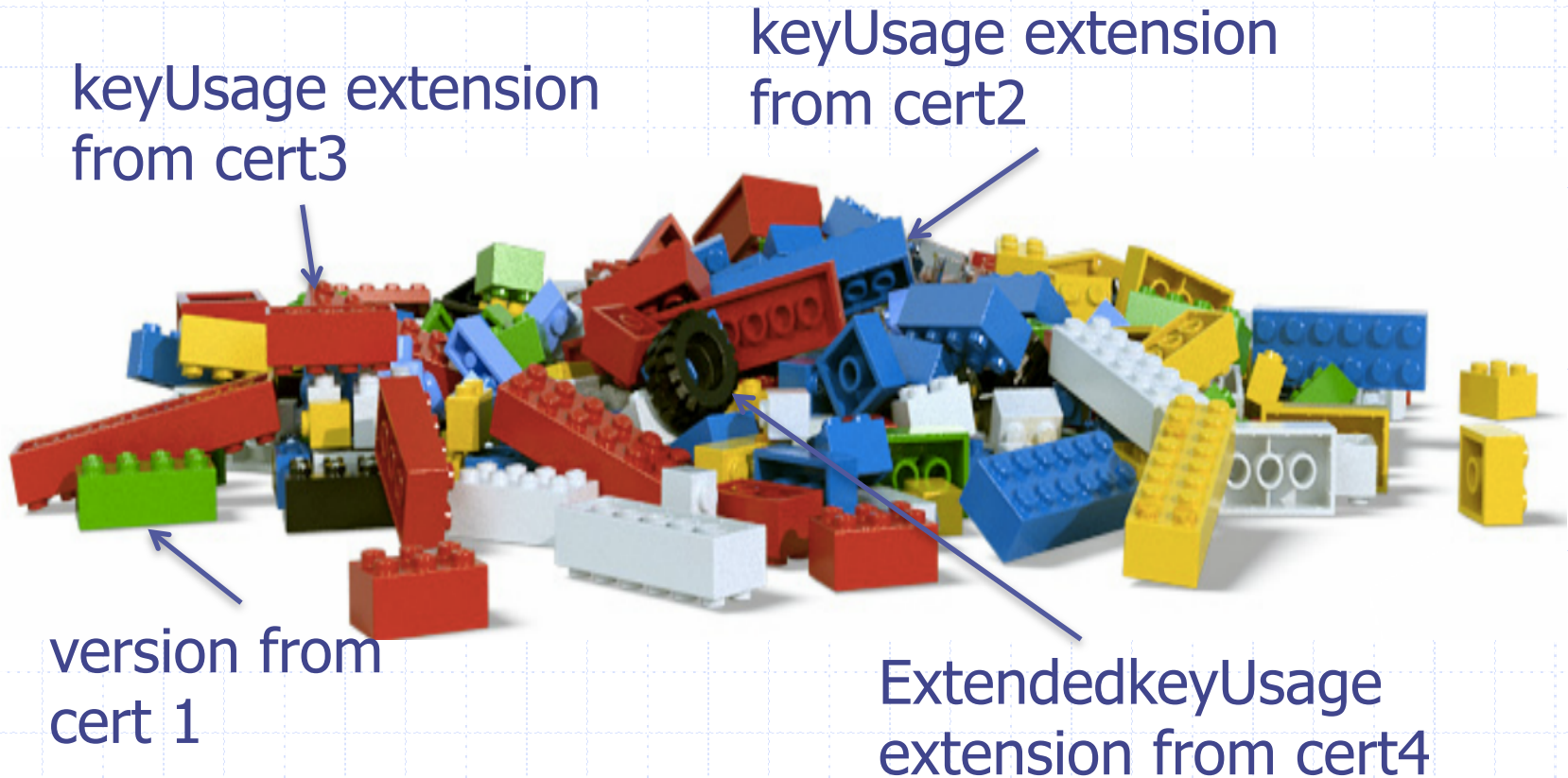
Wait, how can we generate a large set of such syntactically valid pieces without reading X.509 specs?

1. Scan the Internet

Collect 243,246 X.509 server certificates



2. Extract Syntactically Valid Pieces



3. Frankencerts

Generate 8 million **frankencerts** from random combinations of certificate pieces



Differential Testing

- ◆ Multiple implementations of SSL/TLS should implement the same certificate validation logic
- ◆ If a certificate is accepted by some and rejected by others, what does this mean?

Find the Rotten One



No false positives, although some discrepancies might be due to different interpretations of X.509

Results of Differential Testing

- ◆ 14 different SSL/TLS implementations
- ◆ 208 discrepancies due to 15 root causes
- ◆ Multiple bugs
 - Accepting fake and unauthorized intermediate certificate authorities → attacker can impersonate any website!
 - Accepting certificates not authorized for use in SSL or not valid for server authentication
 - Several other issues

Results Summary

Problem	Certificates triggering the problem occur in the original corpus	OpenSSL	PolarSSL	GnuTLS	CyaSSL	MatrixSSL	NSS	OpenJDK, Bouncy Castle	Browsers
Untrusted version 1 intermediate CA certificate	No	reject	reject	accept	reject	accept	reject	reject	reject
Untrusted version 2 intermediate CA certificate	No	reject	reject	reject	reject	accept	reject	reject	reject
Version 1 certificate with valid basic constraints	No	accept	reject	accept	accept	accept	reject	reject	Firefox: reject Opera, Chrome: accept
Intermediate CA not authorized to issue further intermediate CA certificates, but followed in the chain by an intermediate CA certificate	No	reject	reject	reject	reject	accept	reject	reject	reject
... followed by a leaf CA certificate	No	reject	reject	accept	reject	accept	reject	reject	reject
Intermediate CA not authorized to issue certificates for server's hostname	No	reject	reject	accept	accept	accept	reject	reject	reject
Certificate not yet valid	Yes	reject	accept	reject	reject	reject	reject	reject	reject
Certificate expired in its timezone	Yes	reject	accept	reject	reject	accept	reject	reject	reject
CA certificate not authorized for signing other certificates	No	reject	reject	accept	accept	accept	reject	reject	reject
Server certificate not authorized for use in SSL/TLS handshake	Yes	reject	accept	accept	accept	accept	reject	reject	reject
Server certificate not authorized for server authentication	Yes	reject	accept	accept	accept	accept	reject	reject	reject
Certificate with unknown critical extension	No	reject	reject	accept	accept	accept	reject	reject	reject
Certificate with malformed extension value	No	accept	reject	accept	accept	accept	reject	reject	reject
Certificate with the same issuer and subject and a valid chain of trust	No	reject	reject	accept	reject	accept	reject	reject	reject
Issuer name does not match AKI	No	reject	accept	accept	accept	accept	reject	reject	reject
Issuer serial number does not match AKI	No	reject	accept	reject	accept	accept	reject	reject	reject

Version 1 CA certificates

If an SSL/TLS implementation encounters a version 1 (v1) CA certificate that cannot be validated out of band, it must reject it

RFC 5280 Section 6.1.4(k)

v1 CA certificates do not support the CA bit:
anybody with a valid v1 certificate can
pretend to be a CA

Exhibit 1: GnuTLS

```
/* Disable V1 CA flag to prevent version 1 certificates in a supplied
chain. */
flags &= ~(GNUTLS_VERIFY_ALLOW_X509_V1_CA_CRT);
ret = _gnutls_verify_certificate2 (flags,..)

int _gnutls_verify_certificate2(flags, ..)
{
    if (!(flags & GNUTLS_VERIFY_DISABLE_CA_SIGN) &&
        ((flags & GNUTLS_VERIFY_DO_NOT_ALLOW_X509_V1_CA_CRT)
         || issuer_version != 1))
    {
        /*check the CA bit */
    }
}
```

Exhibit 2: Google Chrome

 ~~https://~~www.google.com



The site's security certificate has expired!

You attempted to reach **www.google.com**, but the server presented an expired certificate. No information is available to indicate whether that certificate has been compromised since its expiration. This means Google Chrome cannot guarantee that you are communicating with **www.google.com** and not an attacker. Your computer's clock is currently set to Wednesday, May 7, 2014 8:33:18 PM. Does that look right? If not, you should correct the error and refresh this page.

You should not proceed, **especially** if you have never seen this warning before for this site.

Proceed anyway

Back to safety

▶ [Help me understand](#)

OK to click through?

Exhibit 2: Google Chrome

The screenshot shows a Chrome browser window with the address bar displaying <https://www.google.com>. A security warning dialog is open, titled "Certificate Viewer: www.google.com". The dialog has two tabs: "General" and "Details". The "General" tab is selected, showing the following information:

This certificate has been verified for the following usages:

Issued To

Common Name (CN)	www.google.com
Organization (O)	Google Inc.
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	00:BC:BA:57:5A:51:B4:D5:31

Issued By

Common Name (CN)	www.foobar.com
Organization (O)	foobar inc.
Organizational Unit (OU)	<Not Part Of Certificate>

Validity Period

Issued On	2/5/12
Expires On	2/5/14

Fingerprints

SHA-256 Fingerprint	B9 4B 94 80 9F 99 B3 90 CD DC CD BA FF 4F E4 06 8B 0E AC 26 81 A9 A2 04 15 0C 18 22 71 7E EB AD
---------------------	--

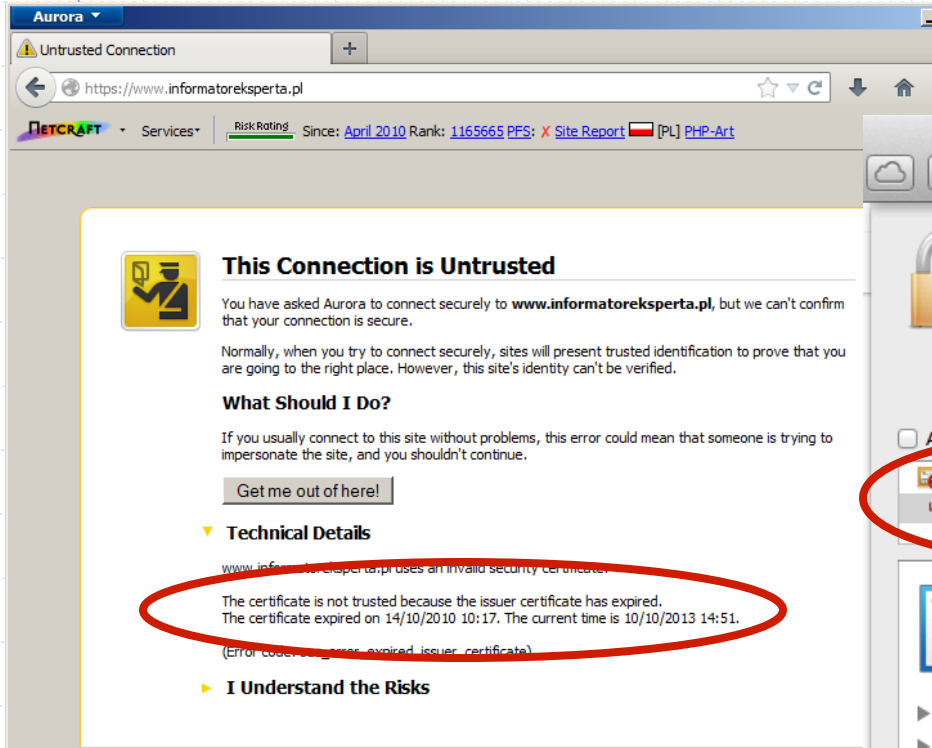
The text "untrusted CA" is written in red next to the "www.foobar.com" entry in the "Issued By" section, which is circled in red. In the background, a security warning is visible with a yellow triangle icon and the text "The site's security is not guaranteed because the certificate is not trusted by your computer's operating system. If not, you should correct the certificate information. You should not proceed with this page unless you are sure the information is correct." A "Proceed anyway" button and a "Help me understand" link are also visible.

Exhibit 2: Root Cause

- ◆ Chrome on Linux uses a modified version of NSS
- ◆ If a certificate is issued by an untrusted CA and is expired, the NSS certificate validation code returns only the “expired” error
- ◆ Firefox uses a glue layer called Personal Security Manager (PSM) over NSS and thus is not affected

Another Bad Warning

<http://news.netcraft.com/archives/2013/10/16/us-government-aiding-spying-against-itself.html>



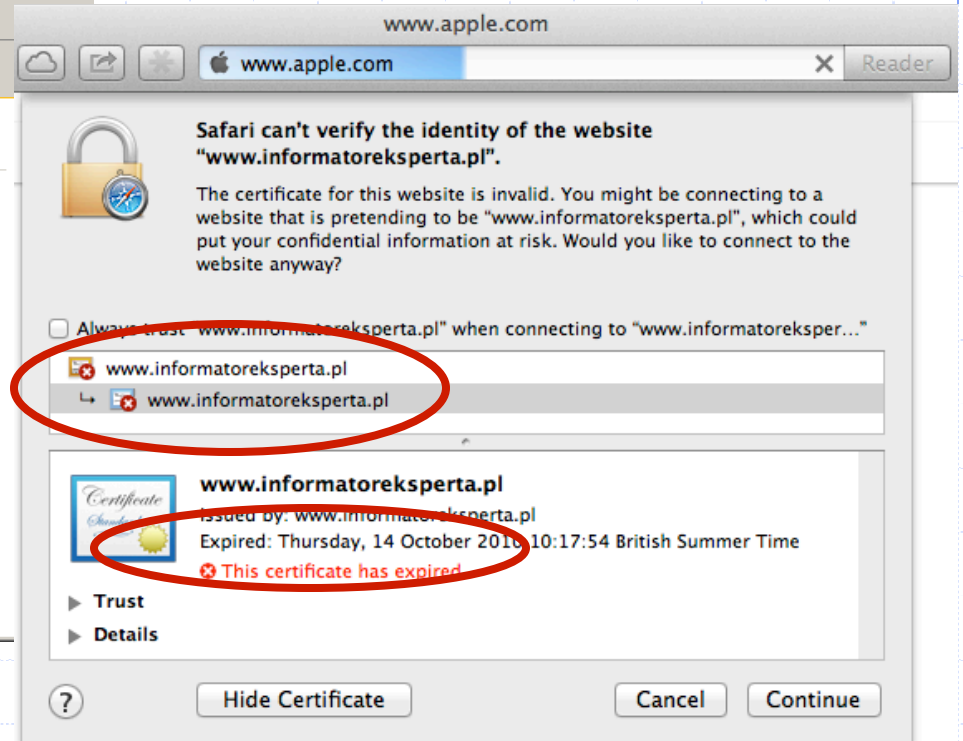
Aurora
Untrusted Connection +
https://www.informatoreksperta.pl
NETCRAFT Services Risk Rating Since: April 2010 Rank: 1165665 PFS: X Site Report [PL] PHP-Art

This Connection is Untrusted
You have asked Aurora to connect securely to **www.informatoreksperta.pl**, but we can't confirm that your connection is secure.
Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?
If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.
[Get me out of here!](#)

Technical Details
www.informatoreksperta.pl proposes an invalid security certificate.
The certificate is not trusted because the issuer certificate has expired.
The certificate expired on 14/10/2010 10:17. The current time is 10/10/2013 14:51.
(Error code: ssl_error_expired_issuer_certificate)

I Understand the Risks



www.apple.com
www.informatoreksperta.pl Reader

Safari can't verify the identity of the website "www.informatoreksperta.pl".
The certificate for this website is invalid. You might be connecting to a website that is pretending to be "www.informatoreksperta.pl", which could put your confidential information at risk. Would you like to connect to the website anyway?

Always trust "www.informatoreksperta.pl" when connecting to "www.informatoreksper..."

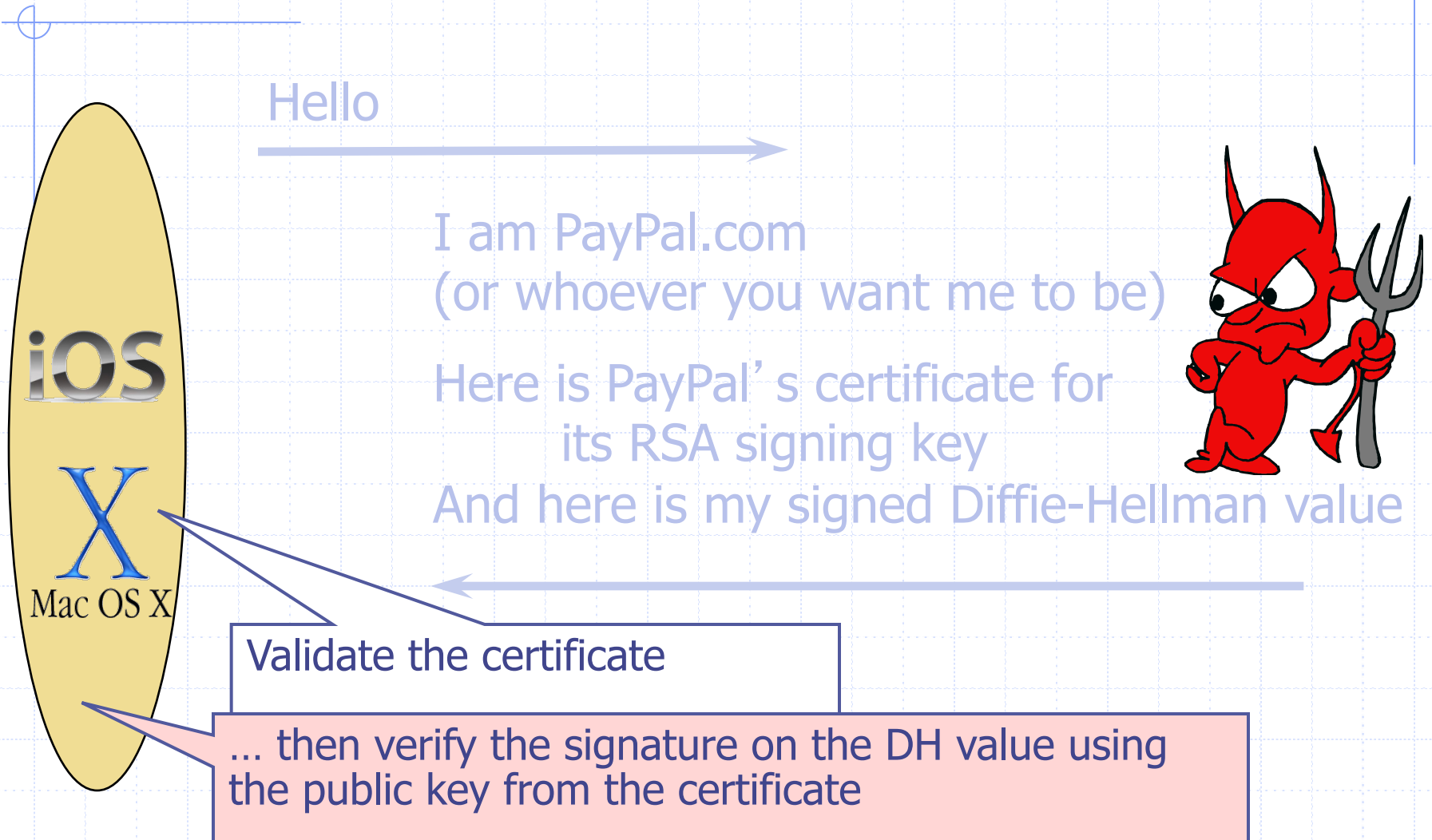
www.informatoreksperta.pl
www.informatoreksperta.pl

www.informatoreksperta.pl
Issued by: www.informatoreksperta.pl
Expired: Thursday, 14 October 2010 10:17:54 British Summer Time
This certificate has expired.

Trust
Details

Hide Certificate Cancel Continue

What Happens After Validation?



Goto Fail



Here is PayPal's certificate
And here is my signed Diffie-Hellman value

... verify the signature on the DH value using the public key from the certificate

iOS

X

Mac OS X

```
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail; ...
err = sslRawVerify(...);
...
fail: ... return err ...
```



Signature is verified here

Complete Fail Against MITM

- ◆ Discovered in February 2014
- ◆ All OS X and iOS software vulnerable to man-in-the-middle attacks
 - Broken TLS implementation provides no protection against the very attack it was supposed to prevent
- ◆ What does this tell you about quality control for security-critical software?



Certificate Revocation

- ◆ Revocation is very important
- ◆ Many valid reasons to revoke a certificate
 - Private key corresponding to the certified public key has been compromised
 - User stopped paying his certification fee to the CA and the CA no longer wishes to certify him
 - CA has been compromised
- ◆ Expiration is a form of revocation, too
 - Many deployed systems don't bother with revocation
 - Re-issuance of certificates is a big revenue source for certificate authorities

Certificate Revocation Mechanisms

◆ Online revocation service

- When a certificate is presented, recipient goes to a special online service to verify whether it is still valid

◆ Certificate revocation list (CRL)

- CA periodically issues a signed list of revoked certificates
- Can issue a "delta CRL" containing only updates

Q: Does revocation protect against forged certificates?

Comodo



- ◆ Comodo is one of the trusted root CAs
 - Its certificates for any website in the world are accepted by every browser
- ◆ Comodo accepts certificate orders submitted through resellers
 - Reseller uses a program to authenticate to Comodo and submit an order with a domain name and public key, Comodo automatically issues a certificate for this site

Comodo Break-In



- ◆ An Iranian hacker broke into instantSSL.it and globalTrust.it resellers, decompiled their certificate issuance program, learned the credentials of their reseller account and how to use Comodo API
 - username: gtadmin, password: globaltrust
- ◆ Wrote his own program for submitting orders and obtaining Comodo certificates
- ◆ On March 15, 2011, got Comodo to issue 9 rogue certificates for popular sites
 - mail.google.com, login.live.com, login.yahoo.com, login.skype.com, addons.mozilla.org, "global trustee"

Consequences

- ◆ Attacker needs to first divert users to an attacker-controlled site instead of Google, Yahoo, Skype, but then...
 - For example, use DNS to poison the mapping of mail.yahoo.com to an IP address
- ◆ ... “authenticate” as the real site
- ◆ ... decrypt all data sent by users
 - Email, phone conversations, Web browsing

Q: Does HTTPS help? How about EV certificates?

Message from the Attacker

<http://pastebin.com/74KXCaeZ>

I'm single hacker with experience of 1000 hacker, I'm single programmer with experience of 1000 programmer, I'm single planner/project manager with experience of 1000 project managers ...

When USA and Isarel could read my emails in Yahoo, Hotmail, Skype, Gmail, etc. without any simple little problem, when they can spy using Echelon, I can do anything I can. It's a simple rule. You do, I do, that's all. You stop, I stop. It's rule #1 ...

Rule#2: So why all the world got worried, internet shocked and all writers write about it, but nobody writes about Stuxnet anymore?... So nobody should write about SSL certificates.

Rule#3: I won't let anyone inside Iran, harm people of Iran, harm my country's Nuclear Scientists, harm my Leader (which nobody can), harm my President, as I live, you won't be able to do so. as I live, you don't have privacy in internet, you don't have security in digital world, just wait and see...

DigiNotar Break-In



- ◆ In June 2011, the same “ComodoHacker” broke into a Dutch certificate authority, DigiNotar
 - Message found in scripts used to generate fake certificates:
“THERE IS NO ANY HARDWARE OR SOFTWARE IN THIS WORLD EXISTS WHICH COULD STOP MY HEAVY ATTACKS MY BRAIN OR MY SKILLS OR MY WILL OR MY EXPERTISE”
- ◆ Security of DigiNotar servers
 - All core certificate servers in a single Windows domain, controlled by a single admin password (Pr0d@dm1n)
 - Software on public-facing servers out of date, unpatched
 - Tools used in the attack would have been easily detected by an antivirus... if it had been present

Consequences of DigiNotar Hack

- ◆ Break-in not detected for a month
- ◆ Rogue certificates issued for *.google.com, Skype, Facebook, www.cia.gov, and 527 other domains
- ◆ 99% of revocation lookups for these certificates originated from Iran
 - Evidence that rogue certificates were being used, most likely by Iranian government or Iranian ISPs to intercept encrypted communications
 - ◆ Textbook man-in-the-middle attack
 - 300,000 users were served rogue certificates

Another Message from the Attacker

<http://pastebin.com/u/ComodoHacker>

Most sophisticated hack of all time ... I'm really sharp, powerful, dangerous and smart!

My country should have control over Google, Skype, Yahoo, etc. [...] I'm breaking all encryption algorithms and giving power to my country to control all of them.

You only heards Comodo (successfully issued 9 certs for me -thanks by the way-), DigiNotar (successfully generated 500+ code signing and SSL certs for me -thanks again-), StartCOM (got connection to HSM, was generating for twitter, google, etc. CEO was lucky enough, but I have ALL emails, database backups, customer data which I'll publish all via cryptome in near future), GlobalSign (I have access to their entire server, got DB backups, their linux / tar gzipped and downloaded, I even have private key of their OWN globalsign.com domain, hahahaha) ... BUT YOU HAVE TO HEAR SO MUCH MORE! SO MUCH MORE! At least 3 more, AT LEAST!

TrustWave



- ◆ In Feb 2012, admitted issuing an intermediate CA certificate to a corporate customer
 - Purpose: “re-sign” certificates for “data loss prevention”
 - Translation: forge certificates of third-party sites in order to spy on employees’ encrypted communications with the outside world
- ◆ Customer can now forge certificates for any site in world... and they will be accepted by any browser!
 - What if a “re-signed” certificate leaks out?
- ◆ Do other CAs do this?

TurkTrust



- ◆ In Jan 2013, a rogue *.google.com certificate was issued by an intermediate CA that gained its authority from the Turkish root CA TurkTrust
 - TurkTrust accidentally issued intermediate CA certs to customers who requested regular certificates
 - Ankara transit authority used its certificate to issue a fake *.google.com certificate in order to filter SSL traffic from its network
- ◆ This rogue *.google.com certificate was trusted by every browser in the world