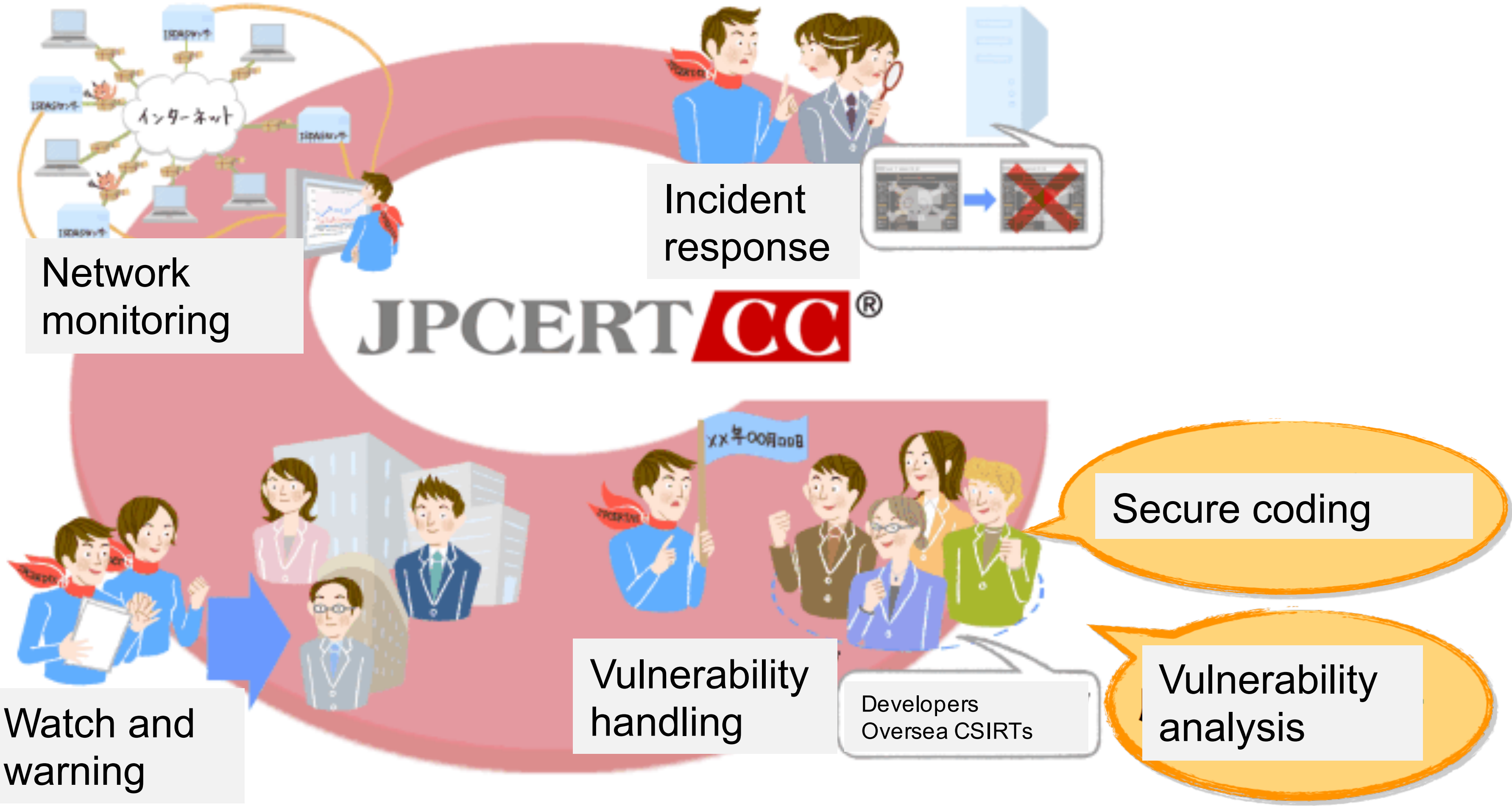
JavaOne2015 version

Case Studies and Lessons Learned from SSL/TLS Certificate Verification Vulnerabilities

JPCERT/CC Information Coordination Group
Yozo TODA (yozo.toda@jpcert.or.jp)

Activities of JPCERT/CC



Network monitoring

Incident response

Secure coding

Vulnerability analysis

Watch and warning

Vulnerability handling

Developers
Oversea CSIRTs

The speaker introduction

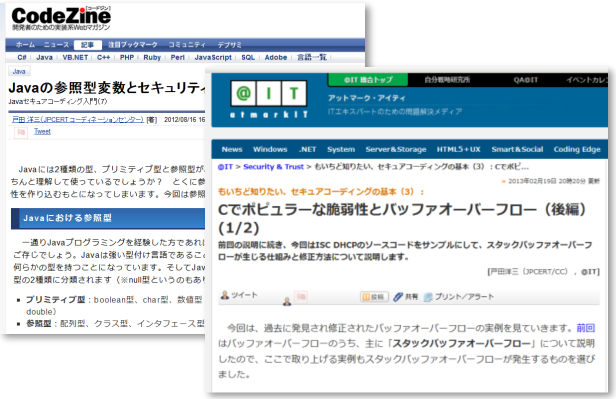
http://www.tomo.gr.jp/root/e9706.html

Yozo Toda

JPCERT/CC Vulnerability analysis team

- vulnerability analysis/handling
- secure coding
- co-op. with secure coding initiative of SEI, CMU

Secure Coding in C and C ++
Robert C. Seacord 著
C/C++
セキュアコーディング
ASCII
The CERT® C
Secure Coding Standard
Robert C. Seacord 著
CERT® C
セキュアコーディング
スタンダード
ASCII
THE CERT ORACLE SECURE CODING STANDARD FOR JAVA
Java™
セキュアコーディング
スタンダード
CERT™/Oracle®版
堅牢な
Javaコードを
書くための
ノウハウを集約。
Androidアプリ開発者必携。
ASCII



CodeZine[コードジン]
開発者のための実装系Webマガジン

ホーム　ニュース　記事　注目ブックマーク　コミュニティ　デブサミ

C# | Java | VB.NET | C++ | PHP | Ruby | Perl | JavaScript | SQL | Adobe | 言語一覧 |

Java

Javaの参照型変数とセキュリティ

Javaセキュアコーディング入門(7)

戸田 洋三(JPCERTコーディネーションセンター) [著]　2012/08/16 16

Tweet

Javaには2種類の型、プリミティブ型と参照型が
ちんと理解して使っているでしょうか？　とくに参
性を作り込むもとになってしまいます。今回は参照

Javaにおける参照型

一通りJavaプログラミングを経験した方であれ
ご存じでしょう。Javaは強い型付け言語であるこ
何らかの型を持つことになっています。そしてJav
型の2種類に分類されます（※null型というのもあ

- **プリミティブ型**：boolean型、char型、数値型
double）
- **参照型**：配列型、クラス型、インタフェース型

@IT 総合トップ　自分戦略研究所　QA@IT　イベントカレ

@IT atmarkIT

アットマーク・アイティ
ITエキスパートのための問題解決メディア

News　Windows　.NET　System　Server&Storage　HTML5＋UX　Smart&Social　Coding Edge

@IT > Security & Trust > もいちど知りたい、セキュアコーディングの基本（3）：Cでポピ…

» 2013年02月19日 20時20分 更新

もいちど知りたい、セキュアコーディングの基本（3）：

Cでポピュラーな脆弱性とバッファオーバーフロー（後編）(1/2)

前回の説明に続き、今回はISC DHCPのソースコードをサンプルにして、スタックバッファオーバーフローが生じる仕組みと修正方法について説明します。

[戸田洋三（JPCERT/CC），@IT]

ツイート　投稿　共有　プリント/アラート

今回は、過去に発見され修正されたバッファオーバーフローの実例を見ていきます。前回はバッファオーバーフローのうち、主に「**スタックバッファオーバーフロー**」について説明したので、ここで取り上げる実例もスタックバッファオーバーフローが発生するものを選びました。

Agenda

- ✓ Introduction
- ✓ Basics: SSL/TLS and Certificate Verification
- ✓ Vulnerabilities in the Real World
- ✓ Lessons Learned from Vulnerabilities
- ✓ References

INTRODUCTION

SSL/TLS

SSL/TLS technology becomes popular today, and is essential for privacy protection and data encryption.

- E-commerce and online banking sites support HTTPS connection.
- Most browsers support HTTP/2 on TLS only

But…
number of vulnerabilities are found on software supporting SSL/TLS.

From security vendors’ reports…

IOActive Research Blog (Jan. 8, 2014)

“40% of the audited apps did not validate the authenticity of SSL certificates presented. This makes them susceptible to Man in The Middle (MiTM) attacks.”

VERACODE, State of Software Security Volume6 (June 2015)

“cryptography issues are highly prevalent across all applications and may be used to allow an attacker to retrieve poorly protected data or hijack communication with an application.”

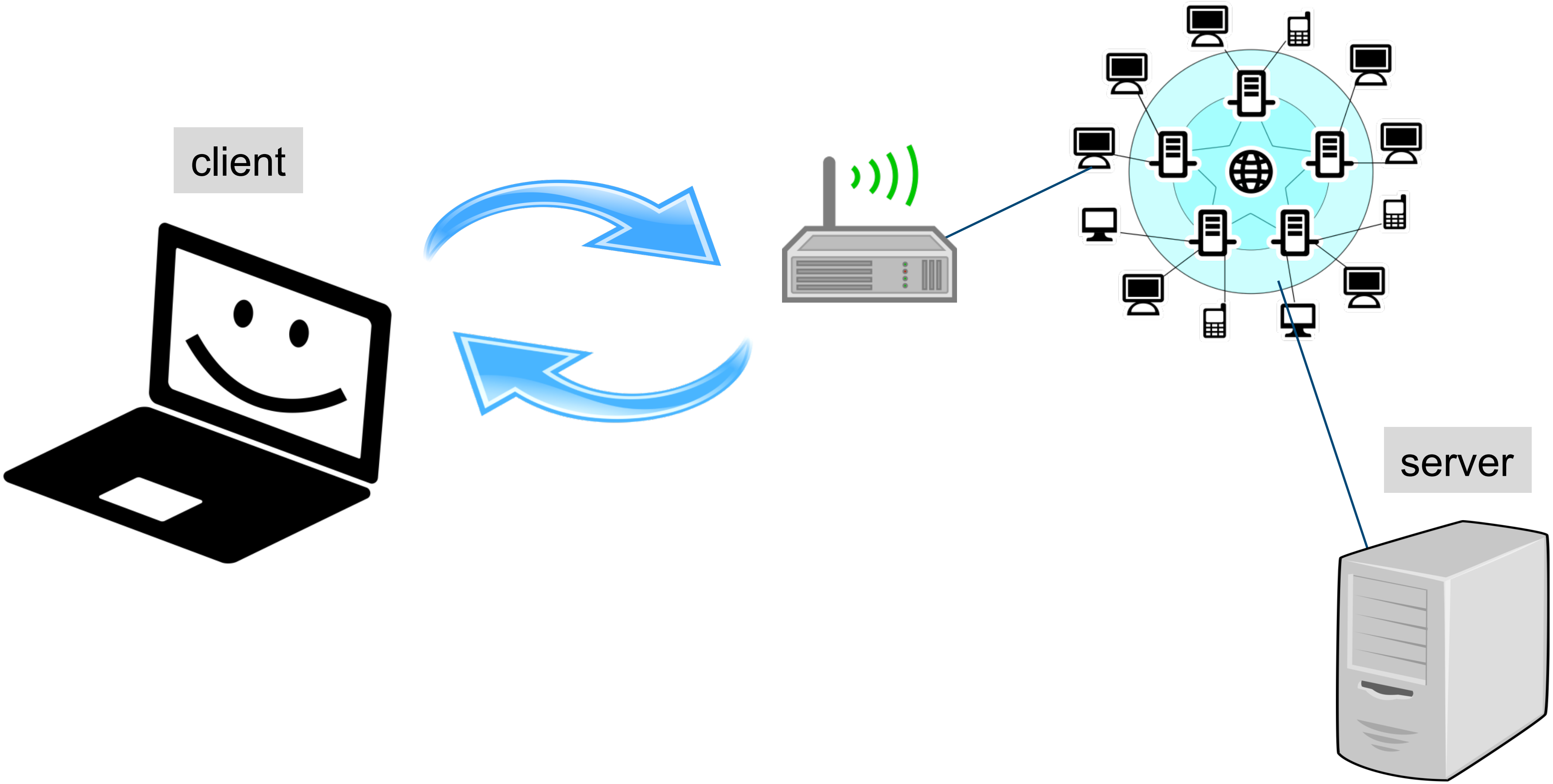
Vulnerability reports on JVN.JP

“improper certificate verification” issues in **jvn.jp** (2013,2014)

JVN#27388160: SumaHo for Android fails to verify SSL/TLS server certificates
JVN#48270605: Yahoo! Japan Box for Android issue where it fails to verify SSL server certificates
JVN#04560253: Yuko Yuko App for Android fails to verify SSL server certificates
JVN#17637243: Kindle App for Android fails to verify SSL server certificates
JVN#27702217: Ameba for Android contains an issue where it fails to verify SSL server certificates
JVN#72950786: Outlook.com for Android contains an issue where it fails to verify SSL server certificates
JVN#10603428: JR East Japan App for Android. contains an issue where it fails to verify SSL server certificates
JVN#16263849: Demaecan for Android. contains an issue where it fails to verify SSL server certificates
JVN#48810179: Denny's App for Android. contains an issue where it fails to verify SSL server certificates
JVN#97810280: KDrive Personal for Windows contains an issue where it fails to verify SSL server certificates
JVN#75084836: Yahoo! Japan Shopping for Android contains an issue where it fails to verify SSL server certificates
JVN#68156832: Yafuoku! Contains an issue where it fails to verify SSL server certificates
JVN#39218538: Pizza Hut Japan Official Order App for Android. co e where it fails to verify SSL server certificates
JVN#85812843: FileMaker Pro fails to verify SSL server
JVN#39707339: Opera fails to verify SSL server certifica
JVN#82029095: sp mode mail issue in the verification of

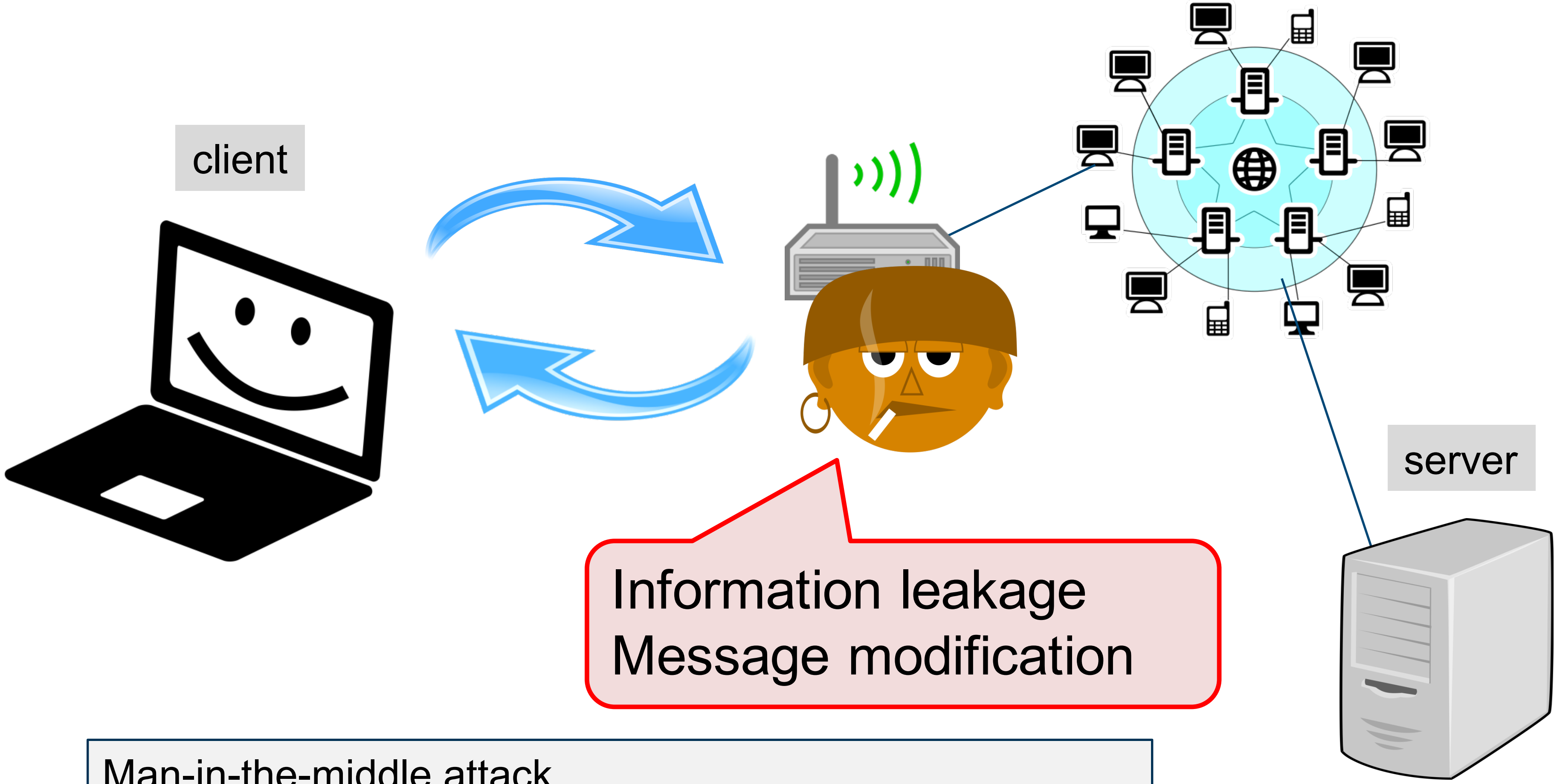
Many Reports on various Android apps

Why Certificate Verification Failure Concerns?



Why Certificate Verification Failure Concerns?

The failure allows **Man-in-the-middle attack**



Information leakage
Message modification

Man-in-the-middle attack
https://en.wikipedia.org/wiki/Man-in-the-middle_attack

SSL/TLS AND CERTIFICATE VERIFICATION

VULNERABILITIES IN THE REAL WORLD

LESSONS LEARNED FROM VULNERABILITIES

REFERENCES

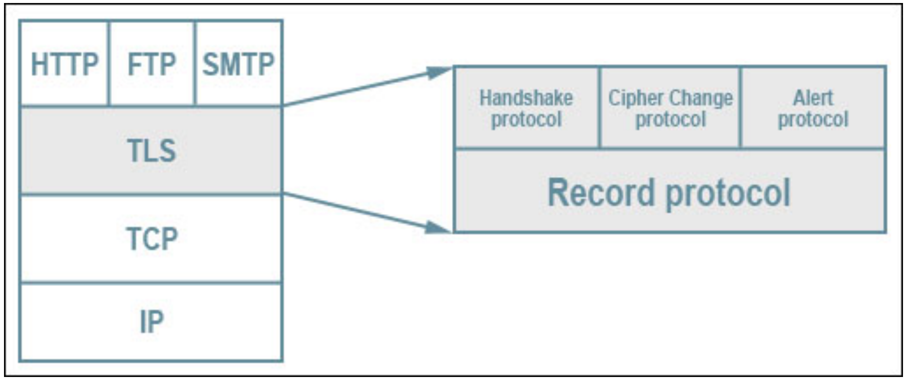
What is SSL/TLS?

https://en.wikipedia.org/wiki/Transport_Layer_Security

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols designed to provide communications security over a computer network.

They use X.509 certificates and hence asymmetric cryptography to authenticate the counterparty with whom they are communicating, and to negotiate a symmetric session key.

This session key is then used to encrypt data flowing between the parties.

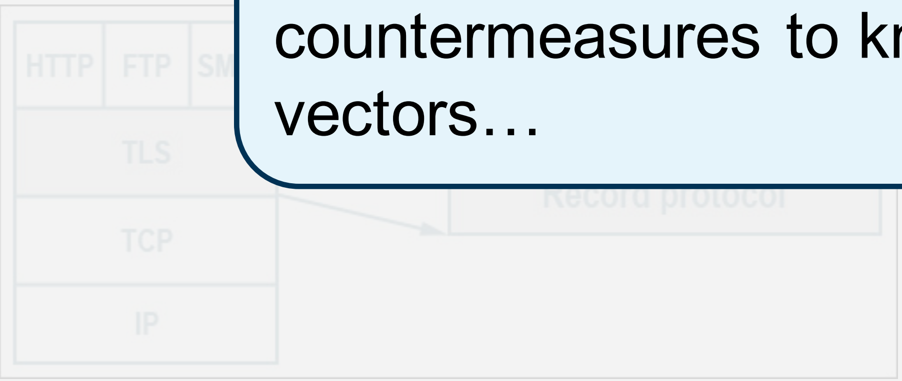


https://nl.wikipedia.org/wiki/Secure_Sockets_Layer

SSL/TLS versions

SSL 3.0 - RFC6101
TLS 1.0 - RFC2246
TLS 1.1 - RFC4346
TLS 1.2 - RFC5246
………..

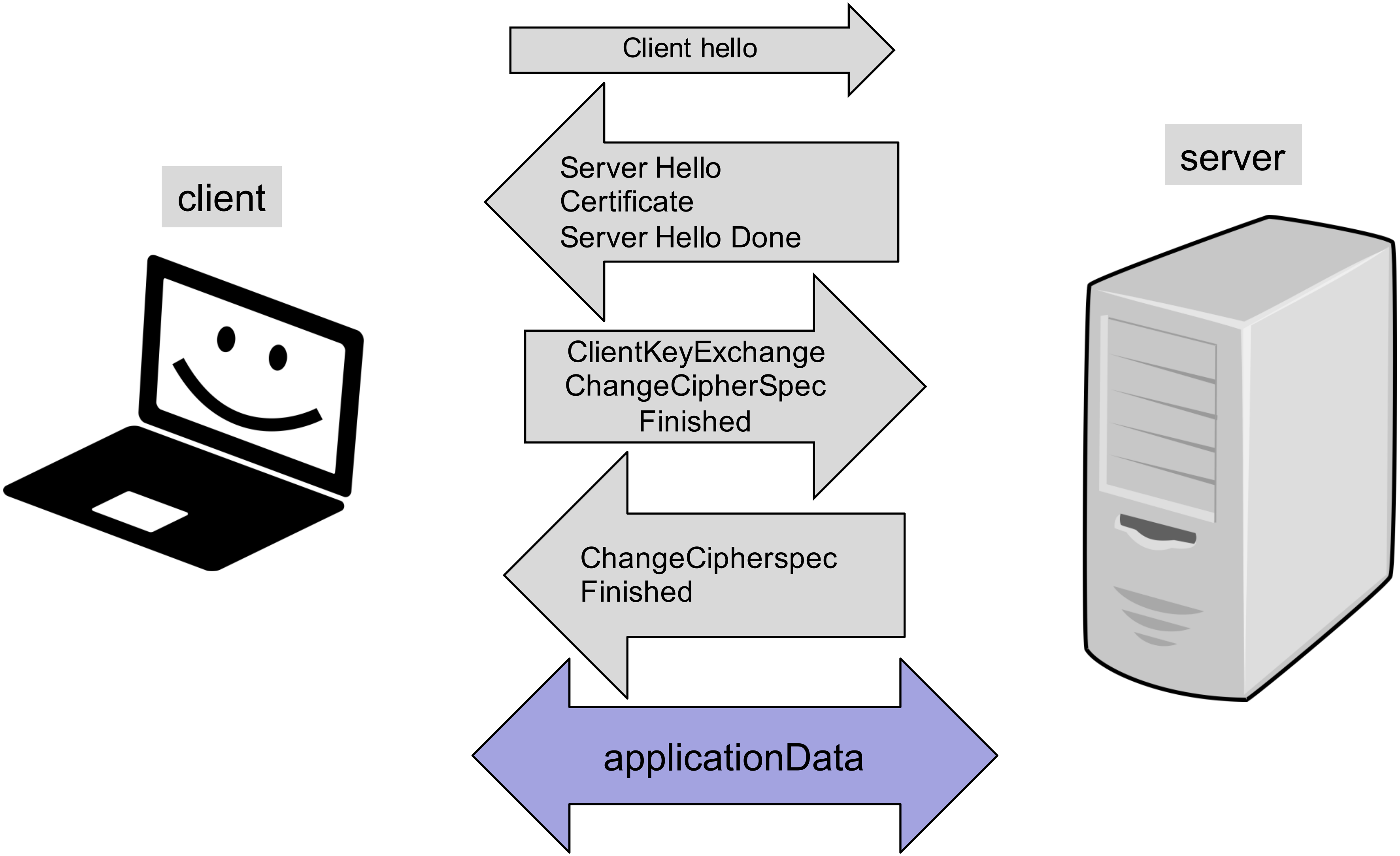
The protocol is still evolving; incorporating new cipher suites and countermeasures to known attack vectors…



https://nl.wikipedia.org/wiki/Secure_Sockets_Layer

SSL/TLS Transaction

This diagram is inspired from
http://www.secureworks.com/cyber-threat-intelligence/threats/transitive-trust/



client

server

Client hello

Server Hello
Certificate
Server Hello Done

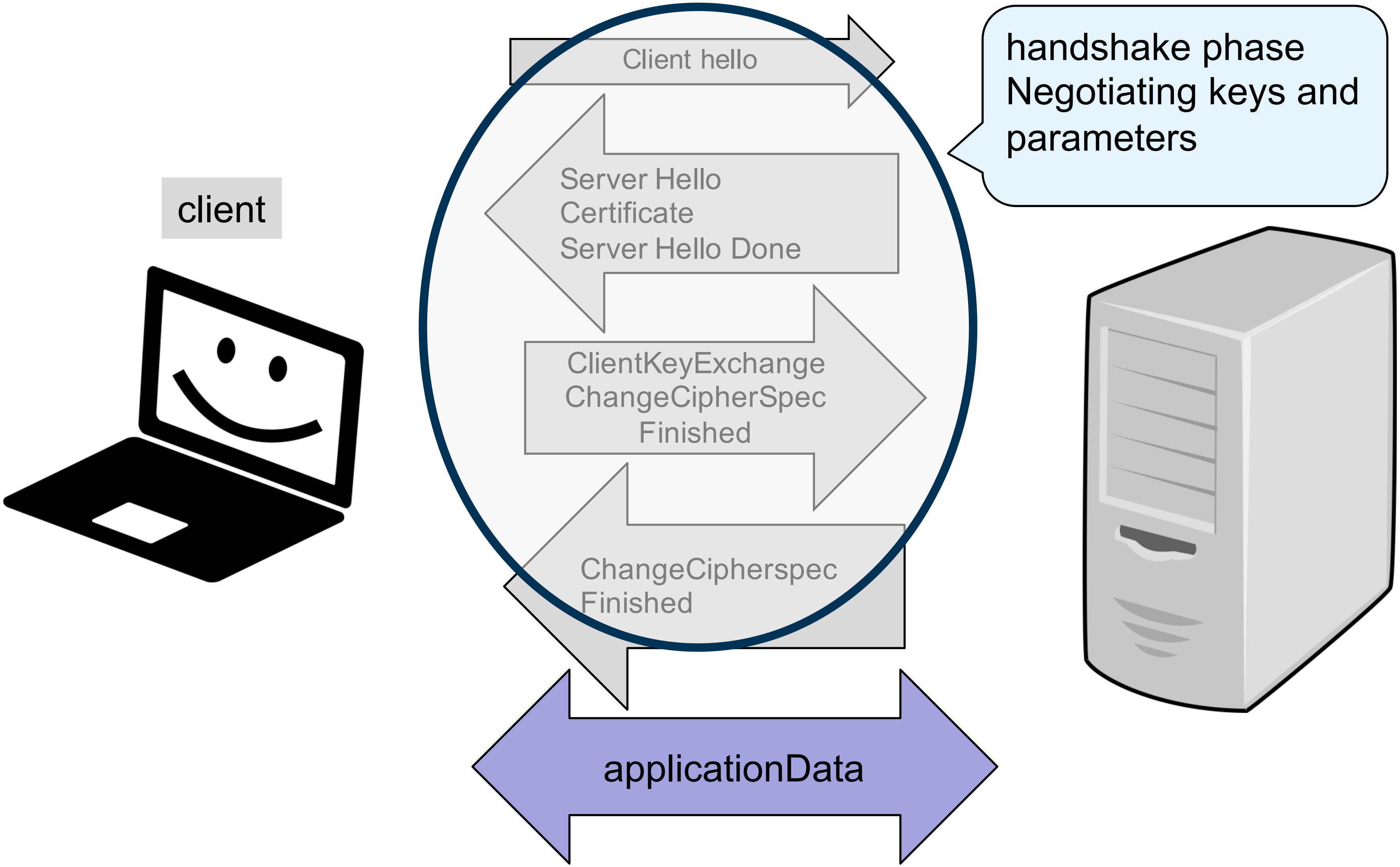
ClientKeyExchange
ChangeCipherSpec
Finished

ChangeCipherspec
Finished

applicationData

SSL/TLS Transaction

This diagram is inspired from
http://www.secureworks.com/cyber-threat-intelligence/threats/transitive-trust/



handshake phase
Negotiating keys and parameters

client

Client hello

Server Hello
Certificate
Server Hello Done

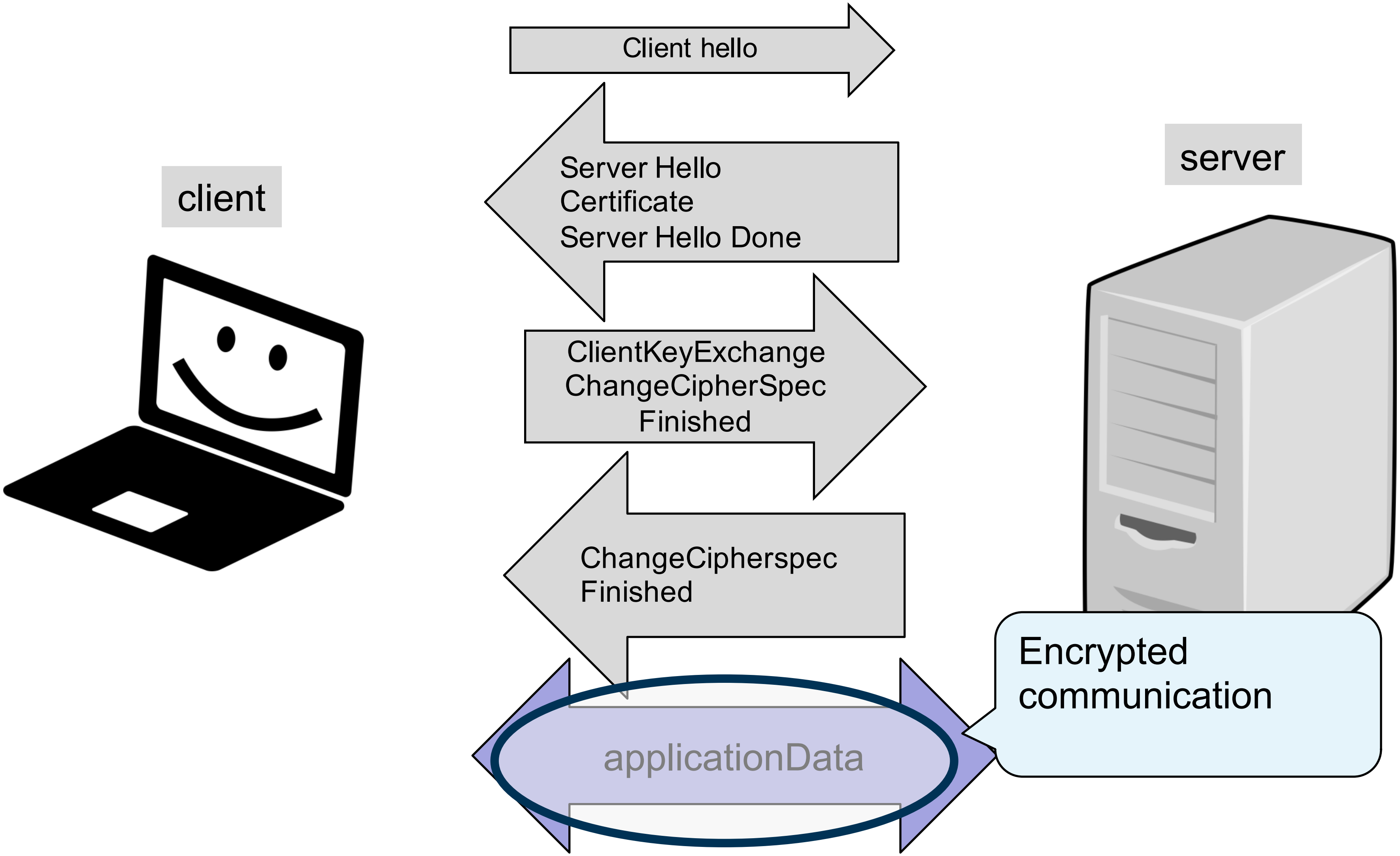
ClientKeyExchange
ChangeCipherSpec
Finished

ChangeCipherspec
Finished

applicationData

SSL/TLS Transaction

This diagram is inspired from
http://www.secureworks.com/cyber-threat-intelligence/threats/transitive-trust/



NetCat: sample client program with URLConnection class

```java
public class NetCat {
  public static void main(String[] argv) throws Exception {
    URI uri = new URI(argv[0]);
    URLConnection conn = uri.toURL().openConnection();

    BufferedReader reader =
      new BufferedReader (new InputStreamReader
               (conn.getInputStream(), “UTF-8”));
    String buffer = reader.readLine();
    System.out.println();
    while (null != buffer) {
      System.out.println(buffer);
      buffer = reader.readLine();
    }
  }
}
```

Sample session (1)

```
$ java NetCat http://www.jpcert.or.jp/

<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 ……
………………
```

```
$ java NetCat https://www.jpcert.or.jp/

<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 ……
………………
```

`URLConnection` supports both “http” and “https” protocol schemes.

Sample session (2)

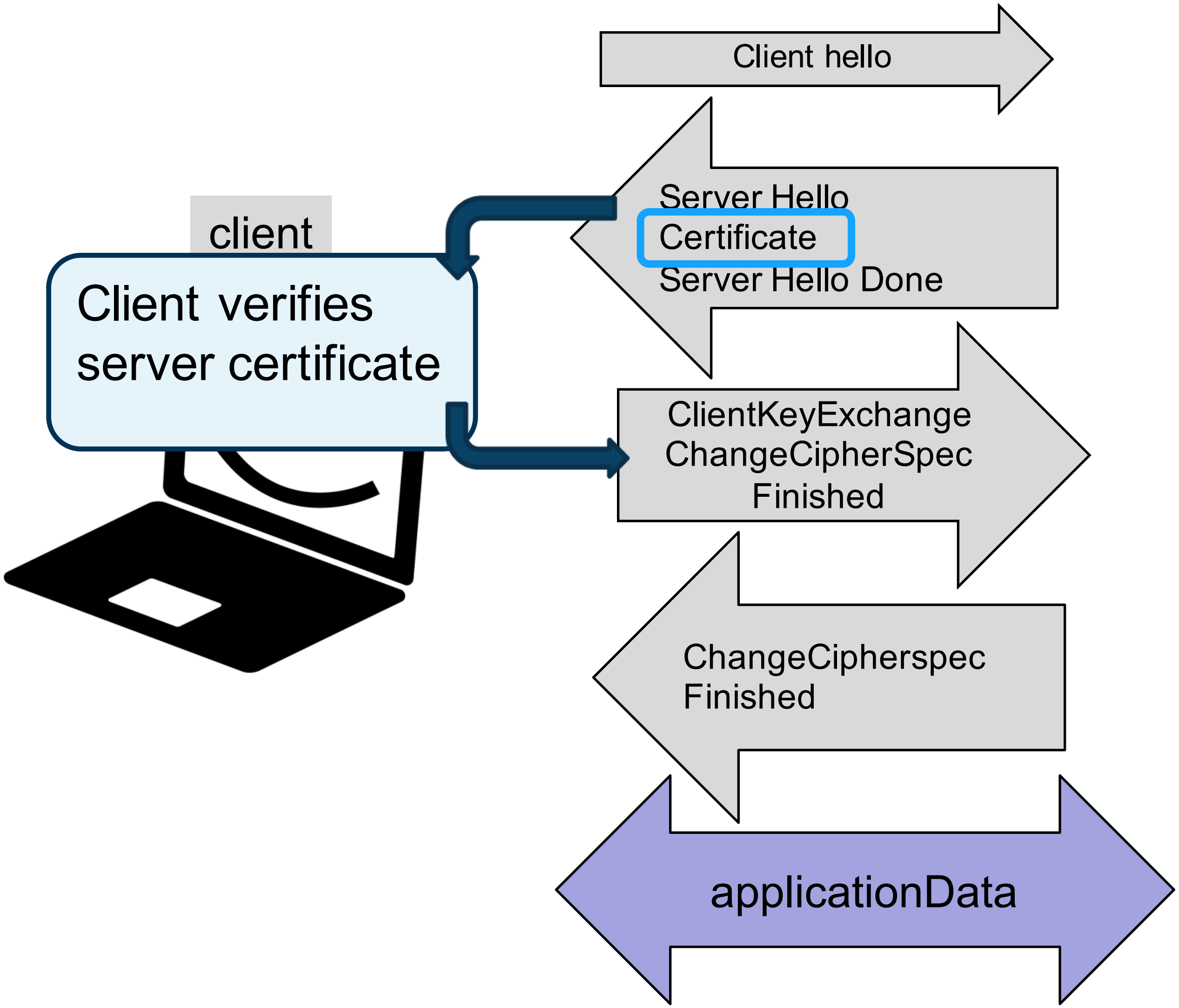
```
$ java NetCat https://www.php.net/
Exception in thread “main” javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find
valid certification path to requested target
        at sun.security.ssl.Alerts.getSSLException(Alerts.java:192)
        at sun.security.ssl.SSLSocketImpl.fatal (SSLSocketImpl.java:1937)
        …………
```

This server certificate is self-signed, hence certificate path validation failed.

In case of HTTPS, **`URLConnection`** verifies the server certificate.

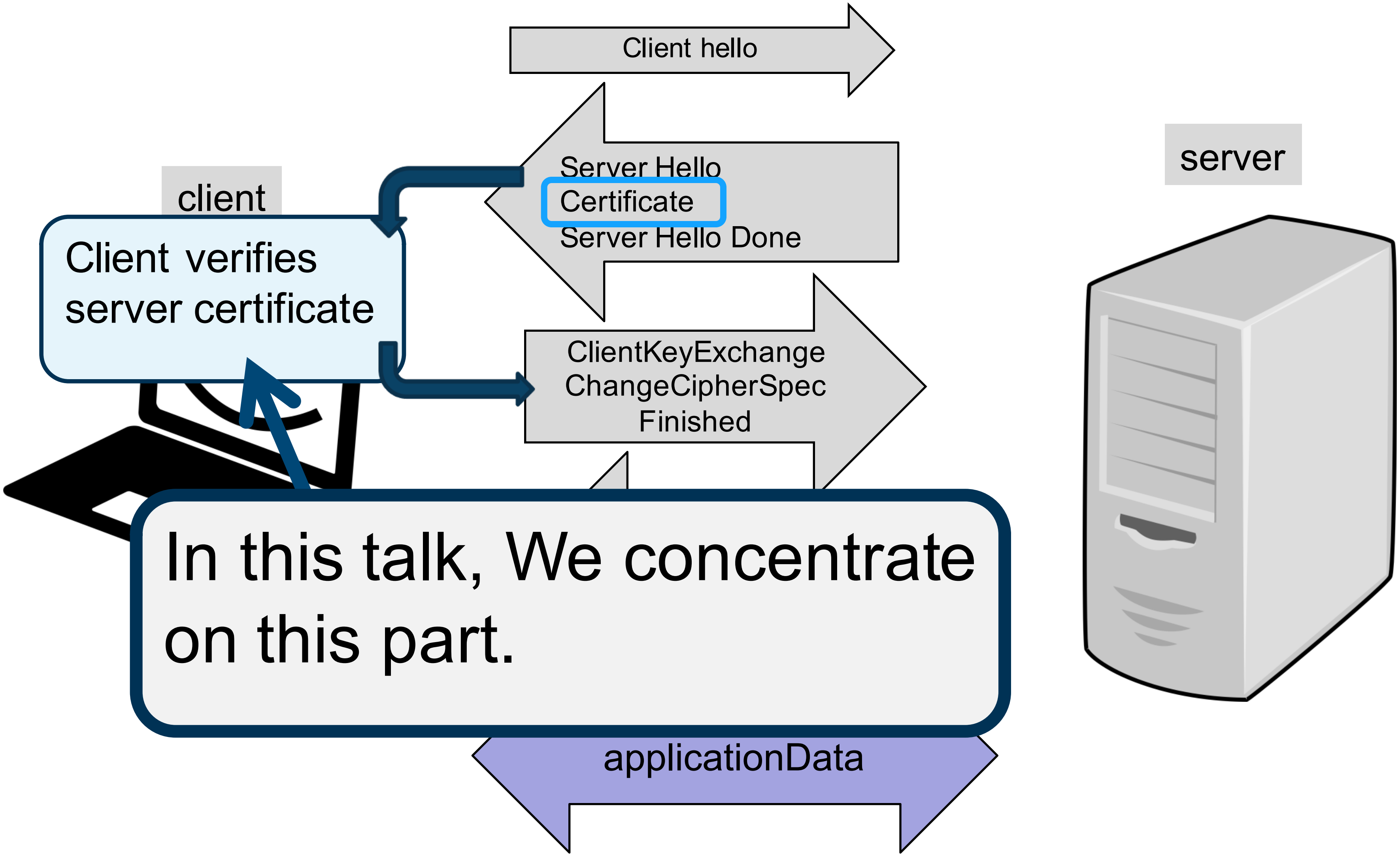
SSL/TLS Transaction

This diagram is inspired from
http://www.secureworks.com/cyber-threat-intelligence/threats/transitive-trust/



SSL/TLS Transaction

This diagram is inspired from http://www.secureworks.com/cyber-threat-intelligence/threats/transitive-trust/



Client verifies server certificate

In this talk, We concentrate on this part.

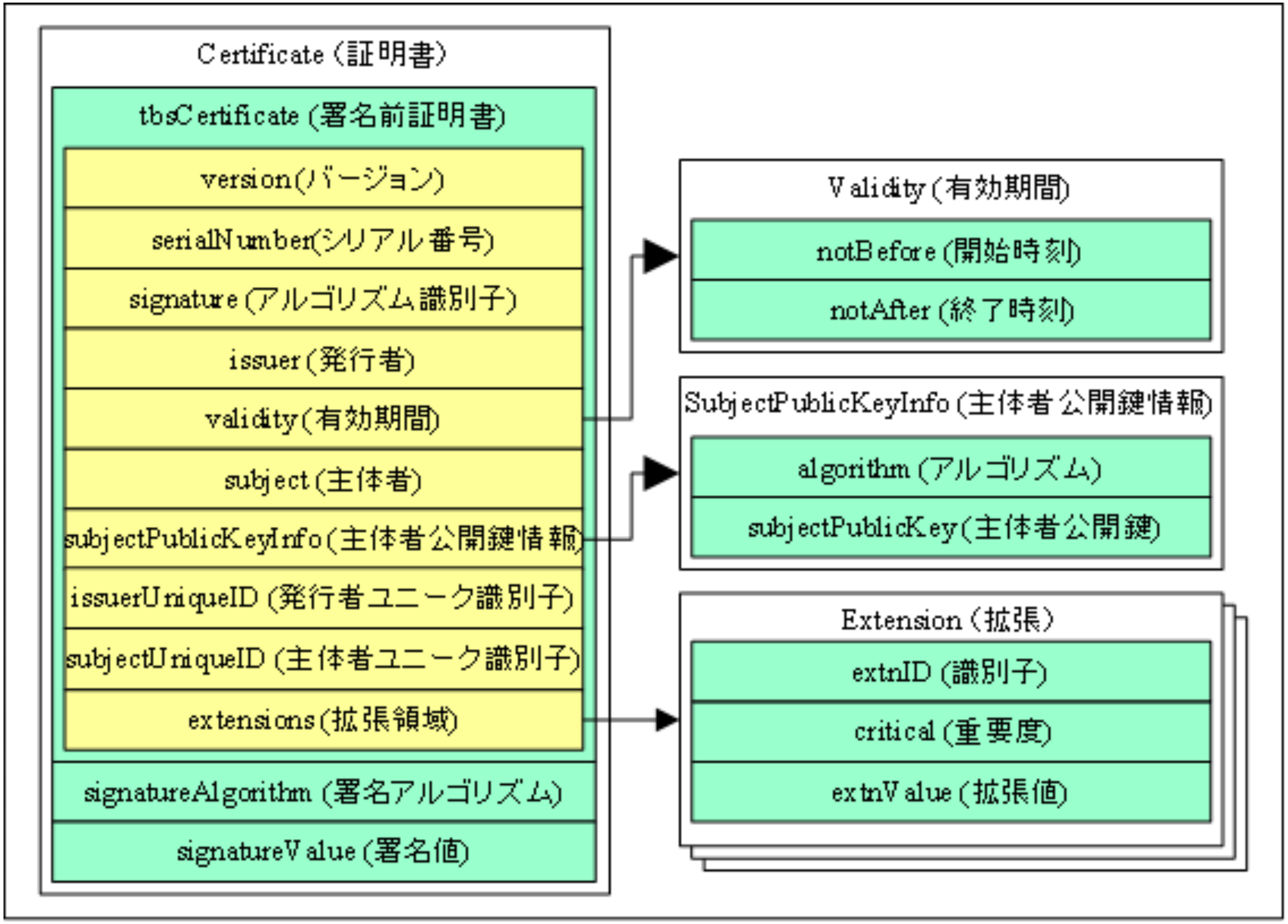
Server Certificates

- A ***server certificate*** contains **<u>the public key</u>** and **<u>the domain name</u>** of **the server** (when it is used in HTTPS)

- Some CA (Certificate Authority) guarantees the correspondence between the two

- ITU-T standard X.509
  - RFC5280, RFC6818
- Web browsers have a set of trusted CA certificates

Structure of X.509 v3 certificates



- Certificate（証明書）
  - tbsCertificate（署名前証明書）
    - version（バージョン）
    - serialNumber（シリアル番号）
    - signature（アルゴリズム識別子）
    - issuer（発行者）
    - validity（有効期間）
    - subject（主体者）
    - subjectPublicKeyInfo（主体者公開鍵情報）
    - issuerUniqueID（発行者ユニーク識別子）
    - subjectUniqueID（主体者ユニーク識別子）
    - extensions（拡張領域）
  - signatureAlgorithm（署名アルゴリズム）
  - signatureValue（署名値）

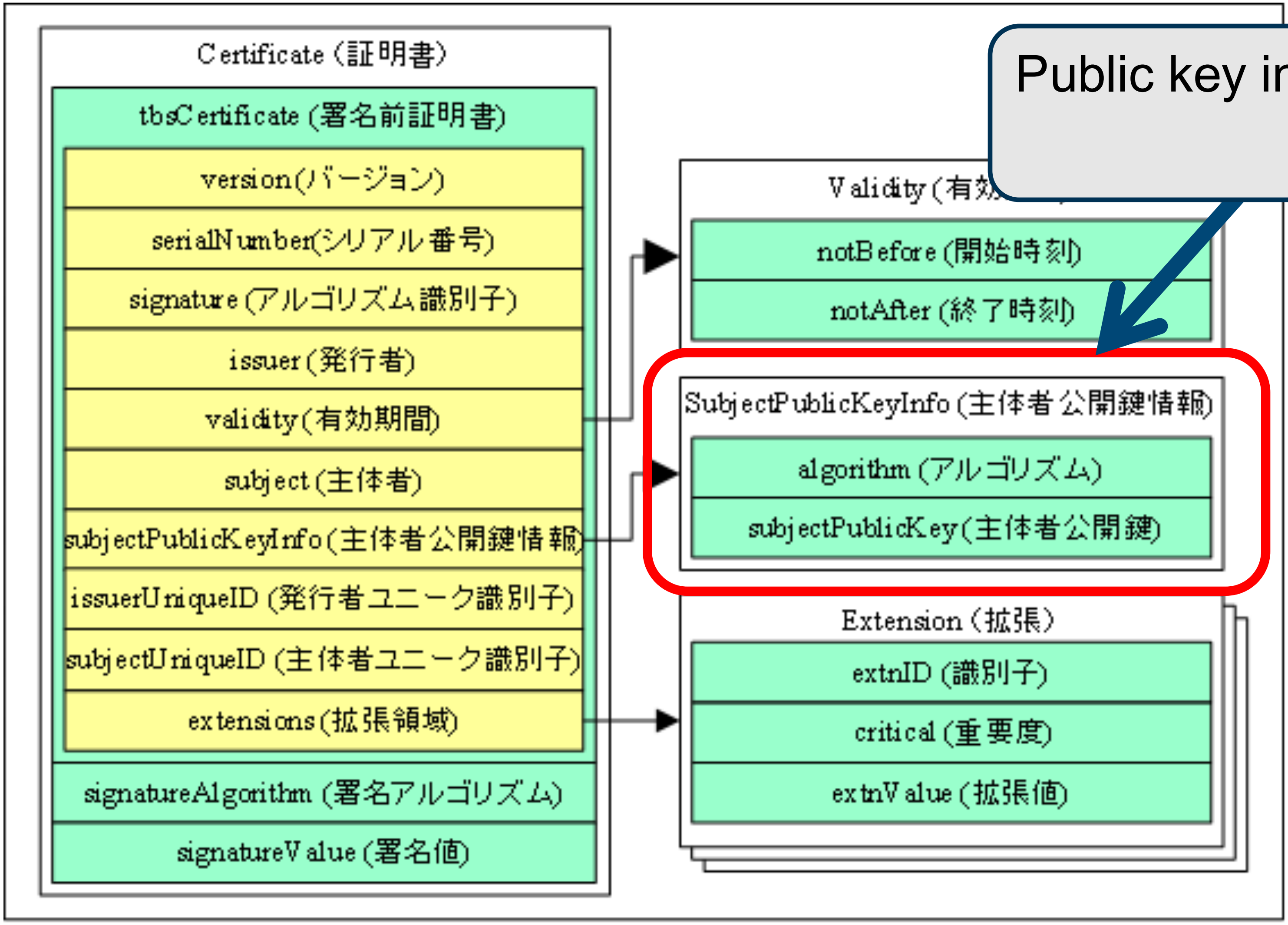
- Validity（有効期間）
  - notBefore（開始時刻）
  - notAfter（終了時刻）

- SubjectPublicKeyInfo（主体者公開鍵情報）
  - algorithm（アルゴリズム）
  - subjectPublicKey（主体者公開鍵）

- Extension（拡張）
  - extnID（識別子）
  - critical（重要度）
  - extnValue（拡張値）

https://www.ipa.go.jp/security/pki/033.html

Structure of X.509 v3 certificates



Certificate (証明書)

tbsCertificate (署名前証明書)

- version(バージョン)
- serialNumber(シリアル番号)
- signature (アルゴリズム識別子)
- issuer (発行者)
- validity(有効期間)
- subject (主体者)
- subjectPublicKeyInfo(主体者公開鍵情報)
- issuerUniqueID (発行者ユニーク識別子)
- subjectUniqueID (主体者ユニーク識別子)
- extensions(拡張領域)

signatureAlgorithm (署名アルゴリズム)

signatureValue (署名値)

Validity (有効

- notBefore (開始時刻)
- notAfter (終了時刻)

SubjectPublicKeyInfo (主体者公開鍵情報)

- algorithm (アルゴリズム)
- subjectPublicKey(主体者公開鍵)

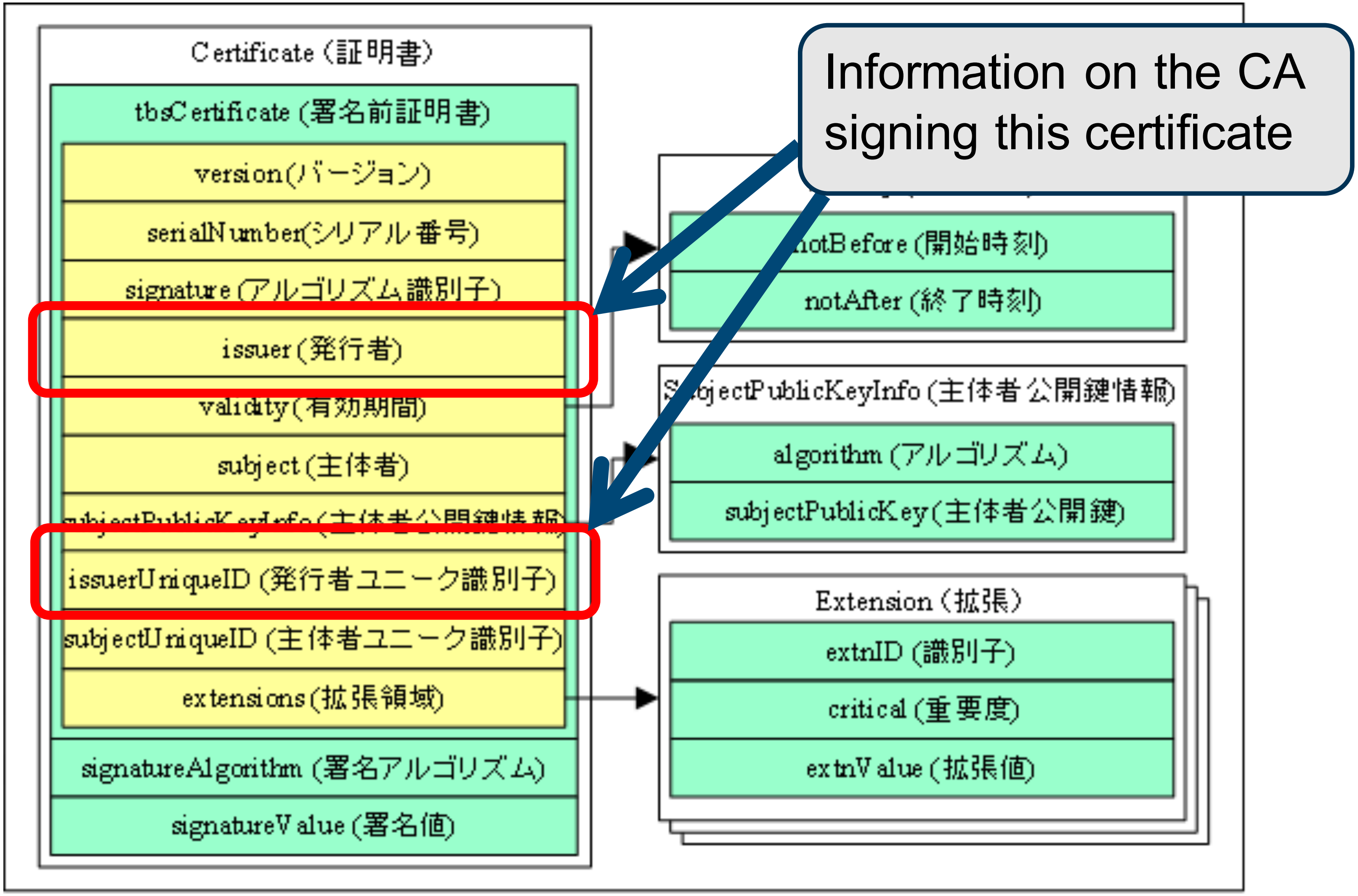
Extension (拡張)

- extnID (識別子)
- critical (重要度)
- extnValue (拡張値)

Public key information

https://www.ipa.go.jp/security/pki/033.html

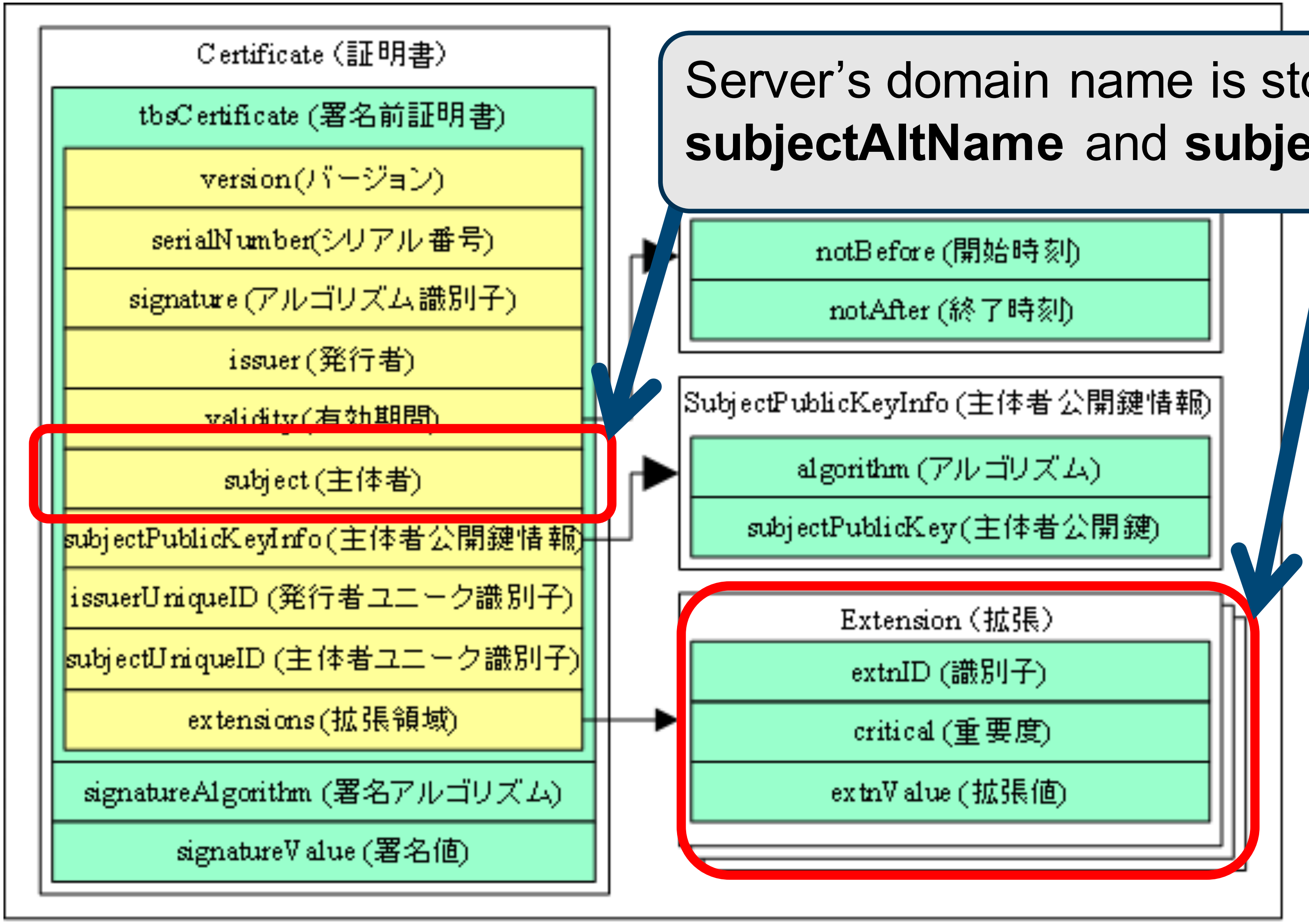
Structure of X.509 v3 certificates


Certificate (証明書)
tbsCertificate (署名前証明書)
version(バージョン)
serialNumber(シリアル番号)
signature (アルゴリズム識別子)
issuer (発行者)
validity (有効期間)
subject (主体者)
subjectPublicKeyInfo (主体者公開鍵情報)
issuerUniqueID (発行者ユニーク識別子)
subjectUniqueID (主体者ユニーク識別子)
extensions (拡張領域)
signatureAlgorithm (署名アルゴリズム)
signatureValue (署名値)
notBefore (開始時刻)
notAfter (終了時刻)
SubjectPublicKeyInfo (主体者公開鍵情報)
algorithm (アルゴリズム)
subjectPublicKey (主体者公開鍵)
Extension (拡張)
extnID (識別子)
critical (重要度)
extnValue (拡張値)

Information on the CA signing this certificate

https://www.ipa.go.jp/security/pki/033.html

Structure of X.509 v3 certificates



Certificate (証明書)

tbsCertificate (署名前証明書)

version(バージョン)

serialNumber(シリアル番号)

signature (アルゴリズム識別子)

issuer (発行者)

validity(有効期間)

subject (主体者)

subjectPublicKeyInfo(主体者公開鍵情報)

issuerUniqueID (発行者ユニーク識別子)

subjectUniqueID (主体者ユニーク識別子)

extensions(拡張領域)

signatureAlgorithm (署名アルゴリズム)

signatureValue (署名値)

notBefore (開始時刻)

notAfter (終了時刻)

SubjectPublicKeyInfo (主体者公開鍵情報)

algorithm (アルゴリズム)

subjectPublicKey(主体者公開鍵)

Extension (拡張)

extnID (識別子)

critical (重要度)

extnValue (拡張値)

Server’s domain name is stored at **subjectAltName** and **subject**

https://www.ipa.go.jp/security/pki/033.html

Example: www.jpcert.or.jp.

- Issuer:
    - C=US
    - O=Symantec Corporation
    - OU=Symantec Trust Network
    - CN=Symantec Class 3 EV SSL CA - G3
- **Subject**:
    - serialNumber=0100-05-006504
    - C=JP
    - postalCode=101-0054
    - ST=Tokyo
    - L=Chiyoda-ku
    - streetAddress="Hirose Bldg. 11F, 3-17 Kanda-nishikicho"
    - O="Japan Computer Emergency Response Team Coordination Center"
    - OU="System Administration Group"
    - **CN=www.jpcert.or.jp**
- X509v3 extensions:
    - **X509v3 Subject Alternative Name:**
        - **DNS:www.jpcert.or.jp**
    - X509v3 Basic Constraints:
        - CA:FALSE

CA Information

Server Information

Example: www.google.com.

- Issuer:
  - C=US
  - O=Google Inc
  - CN=Google Internet Authority G2
- **Subject**:
  - C=US
  - ST=California
  - L=Mountain View
  - O=Google Inc
  - **CN=google.com**
- X509v3 extensions:
  - **X509v3 Subject Alternative Name**:
    - **DNS:google.com, DNS:*.2mdn.net, DNS:*.android.com,**
    - **DNS:*.appengine.google.com, DNS:*.au.doubleclick.net,**
    - **DNS:*.cc-dt.com, DNS:*.cloud.google.com, DNS:*.de.doubleclick.net,**
    - **DNS:*.doubleclick.com, DNS:*.doubleclick.net,**
    - **DNS:*.fls.doubleclick.net, DNS:*.fr.doubleclick.net,**
    - **DNS:*.google-analytics.com, DNS:*.google.ac, DNS:*.google.ad,**
    - …….. (omitted) ……..
  - X509v3 Basic Constraints:
    - CA:FALSE

“Certificate Verification” contains 3 processes

- Verifies that the received server certificate is properly created
  - ⇒**certificate verification** (in a narrow sense)

- Verifies that there is a proper certificate path
  - ⇒**certificate path validation**

- Verifies that the server name contained in the certificate matches the server name to contact
  - ⇒**host name verification**

Certificate Verification (in a narrow sense)

- **Is this certificate valid?**
  - Correct ASN.1 data structure?
  - ~~Properly signed by some trusted CA?~~
  - Not expired?
  - Not revoked?

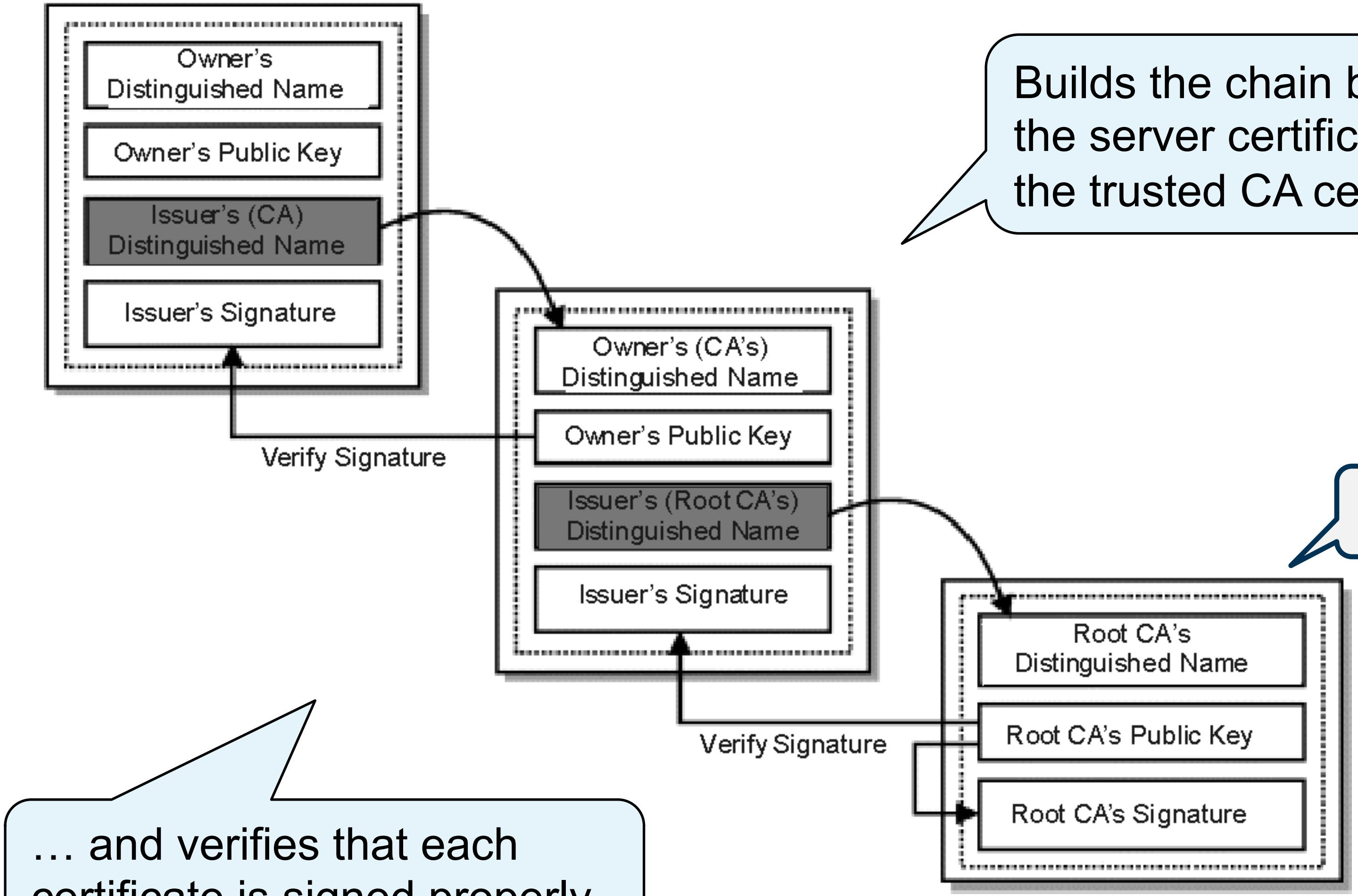
Certificate Path Validation

- Are there any certificate path(chain) starting from the certificate up to some trusted CA certificate?
- Is this certificate path valid?

RFC5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
6. Certification Path Validation
https://tools.ietf.org/html/rfc5280#section-6

Certification path validation algorithm
https://en.wikipedia.org/wiki/Certification_path_validation_algorithm

Certificate Path Validation


Owner's Distinguished Name
Owner's Public Key
Issuer's (CA) Distinguished Name
Issuer's Signature
Verify Signature
Owner's (CA's) Distinguished Name
Owner's Public Key
Issuer's (Root CA's) Distinguished Name
Issuer's Signature
Verify Signature
Root CA's Distinguished Name
Root CA's Public Key
Root CA's Signature

Builds the chain between the server certificate and the trusted CA certificate

Root CA

… and verifies that each certificate is signed properly.

https://security.stackexchange.com/questions/56389/ssl-certificate-framework-101-how-does-the-browser-actually-verify-the-validity

Hostname Verification

- Confirm the two identities match: the server name (domain name) to access and the server name stored in the certificate
- **subjectAltName** extension MUST be used if exists
- Matching algorithm is the same as the algorithm used in certificate path validation

RFC2818: HTTP Over TLS 3.1. Server Identity https://tools.ietf.org/html/rfc2818#section-3.1
RFC5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile 7. Processing Rules for Internationalized Names https://tools.ietf.org/html/rfc5280#section-7

SSL/TLS AND CERTIFICATE VERIFICATION

VULNERABILITIES IN THE REAL WORLD

LESSONS LEARNED FROM VULNERABILITIES

REFERENCES

Real Vulnerabilities: Pattern1

Vulnerability reports on JVN.JP

“improper certificate verification” issues in **jvn.jp** (2013,2014)

JVN#27388160: SumaHo for Android fails to verify SSL/TLS server certificates
JVN#48270605: Yahoo! Japan Box for Android issue where it fails to verify SSL server certificates
JVN#04560253: Yuko Yuko App for Android fails to verify SSL server certificates
JVN#17637243: Kindle App for Android fails to verify SSL server certificates
JVN#27702217: Ameba for Android contains an issue where it fails to verify SSL server certificates
JVN#72950786: Outlook.com for Android contains an issue where it fails to verify SSL server certificates
JVN#10603428: JR East Japan App for Android. contains an issue where it fails to verify SSL server certificates
JVN#16263849: Demaecan for Android. contains an issue where it fails to verify SSL server certificates
JVN#48810179: Denny's App for Android. contains an issue where it fails to verify SSL server certificates
JVN#97810280: KDrive Personal for Windows contains an issue where it fails to verify SSL server certificates
JVN#75084836: Yahoo! Japan Shopping for Android contains an issue where it fails to verify SSL server certificates
JVN#68156832: Yafuoku! Contains an issue where it fails to verify server certificates
JVN#39218538: Pizza Hut Japan Official Order App for Android. co where it fails to verify SSL server certificates
JVN#85812843: FileMaker Pro fails to verify SSL server
JVN#39707339: Opera fails to verify SSL server certifica
JVN#82029095: sp mode mail issue in the verification of

Many Reports on various Android apps

Vulnerable Code

```
 public static HttpClient getNewHttpClient() {
    DefaultHttpClient v6;
    try {
       KeyStore v5 = KeyStore.getInstance(KeyStore.getDefaultType());
       v5.load(null, null);
       MySSLSocketFactory mySSLSocket = new MySSLSocketFactory(v5);
       if(ApplicationDefineRelease.sAllowAllSSL) {
          ((SSLSocketFactory)mySSLScoket).setHostnameVerifier
                                        (SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);
       }

       BasicHttpParams v2 = new BasicHttpParams();
       HttpConnectionParams.setConnectionTimeout(((HttpParams)v2), 30000);
       ...
    }
    catch(Exception v1) {
       v6 = new DefaultHttpClient();
    }
    return ((HttpClient)v6);
}
```

Vulnerable Code

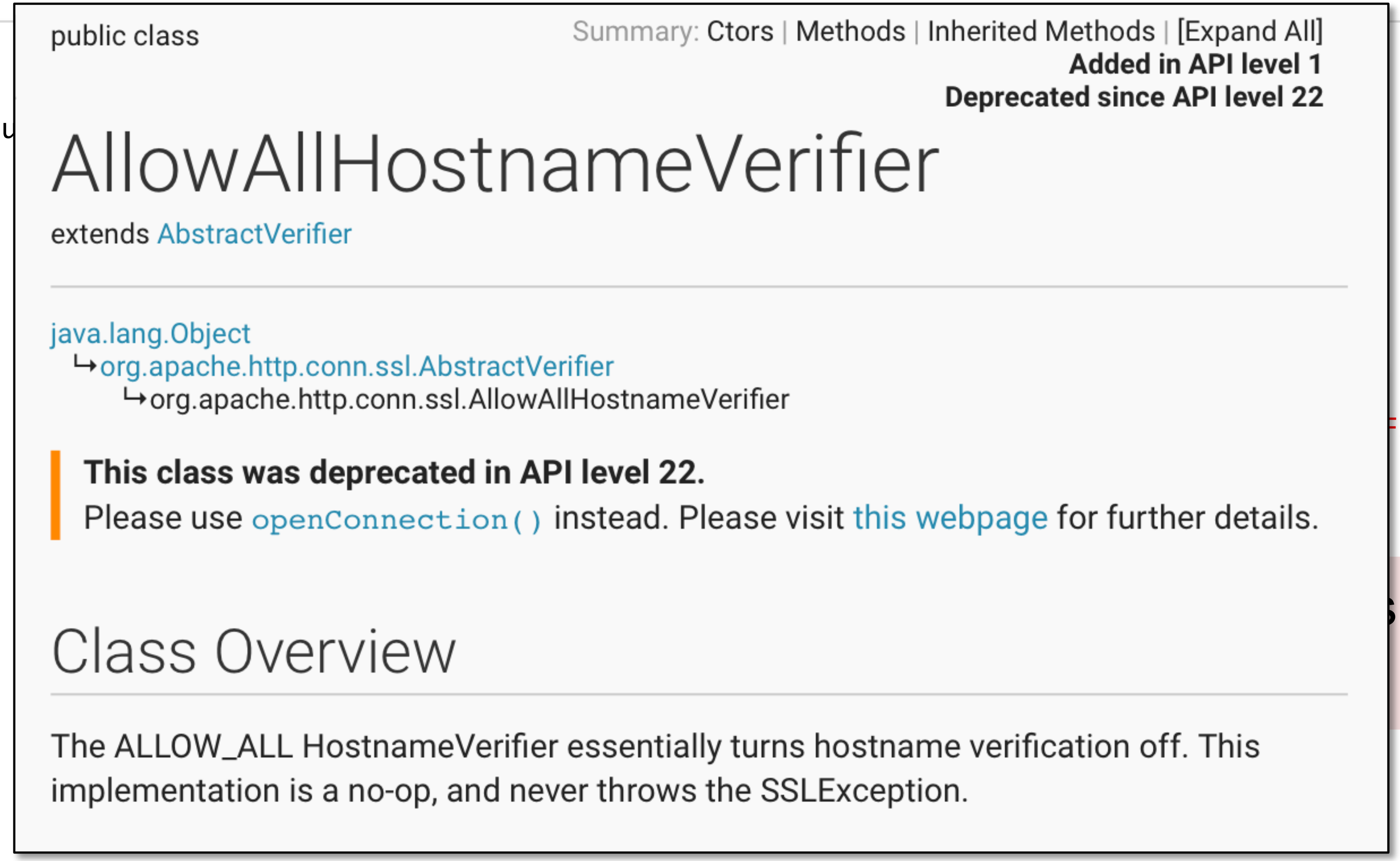
```
public static HttpClient getNewHttpClient() {
    DefaultHttpClient v6;
    try {
        KeyStore v5 = KeyStore.getInstance(KeyStore.getDefaultType());
        v5.load(null, null);
        MySSLSocketFactory mySSLSocket = new MySSLSocketFactory(v5);
                                                                                                  FIER);
        BasicHttpParams v2 = new BasicHttpParams();
        HttpConnectionParams.setConnectionTimeout(
        ...
    }
    catch(Exception v1) {
        v6 = new DefaultHttpClient();
    }
    return ((HttpClient)v6);
}
```

```
((SSLSocketFactory)mySSLScoket).setHostnameVerifier
          (SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);
```

Hostname verification is disabled!!

Vulnerable Code

```
pu
                                                                  FIER);
}
```



public class

Summary: Ctors | Methods | Inherited Methods | [Expand All]

Added in API level 1

Deprecated since API level 22

AllowAllHostnameVerifier

extends AbstractVerifier

java.lang.Object
 ↳org.apache.http.conn.ssl.AbstractVerifier
  ↳org.apache.http.conn.ssl.AllowAllHostnameVerifier

This class was deprecated in API level 22.
Please use `openConnection()` instead. Please visit this webpage for further details.

Class Overview

The ALLOW_ALL HostnameVerifier essentially turns hostname verification off. This implementation is a no-op, and never throws the SSLException.

Other Vulnerable Code Patterns

empty HostnameVerifier

```
HostnameVerifier hv = new HostnameVerifier() {
   @Override
   public boolean verify(String hostname, SSLSession session) {
      // always return true, any hostnames are accepted
      return true;
   }
};
```

Other Vulnerable Code Patterns

empty TrustManager

```
TrustManager tm = new X509TrustManager() {

   @Override
   public void checkClientTrusted(X509Certificate[] chain,
            String authType) throws CertificateException {
      // do nothing, any certificates are accepted
   }

   @Override
   public void checkServerTrusted(X509Certificate[] chain,
            String authType) throws CertificateException {
      // do nothing, any certificates are accepted
   }

   @Override
   public X509Certificate[] getAcceptedIssuers() {
      return null;
   }
};
```

SSL/TLS vulnerability as a research topic

ACM CCS2012

- Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security
  http://www2.dcsec.uni-hannover.de/files/android/p50-fahl.pdf
- The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software
  https://crypto.stanford.edu/~dabo/pubs/abstracts/ssl-client-bugs.html

ACM CCS2013

- Rethinking SSL Development in an Appified World
  http://android-ssl.org/files/p49.pdf

SSL/TLS vulnerability as a research topic

Many application mis-use SSL/TLS libraries!!

- disable certificate verification

- disable hostname verification

-……

the cause(s) of SSL/TLS related vulnerabilities

- Developer's lack of understanding SSL/TLS

- Releasing with the temporary configuration for internal testing

- Requirement from the customer

Real Vulnerabilities: Pattern2

Improper certificate path validation

Improper Certificate Path Validation: Fake ID

Android Fake ID Vulnerability Lets Malware Impersonate Trusted Applications, Puts All Android Users Since January 2010 At Risk

https://bluebox.com/technical/android-fake-id-vulnerability/

Presented at BlackHat 2014 USA

ANDROID FAKEID VULNERABILITY WALKTHROUGH

https://www.blackhat.com/us-14/archives.html#android-fakeid-vulnerability-walkthrough

This vulnerability is related to application-signing in Android OS…

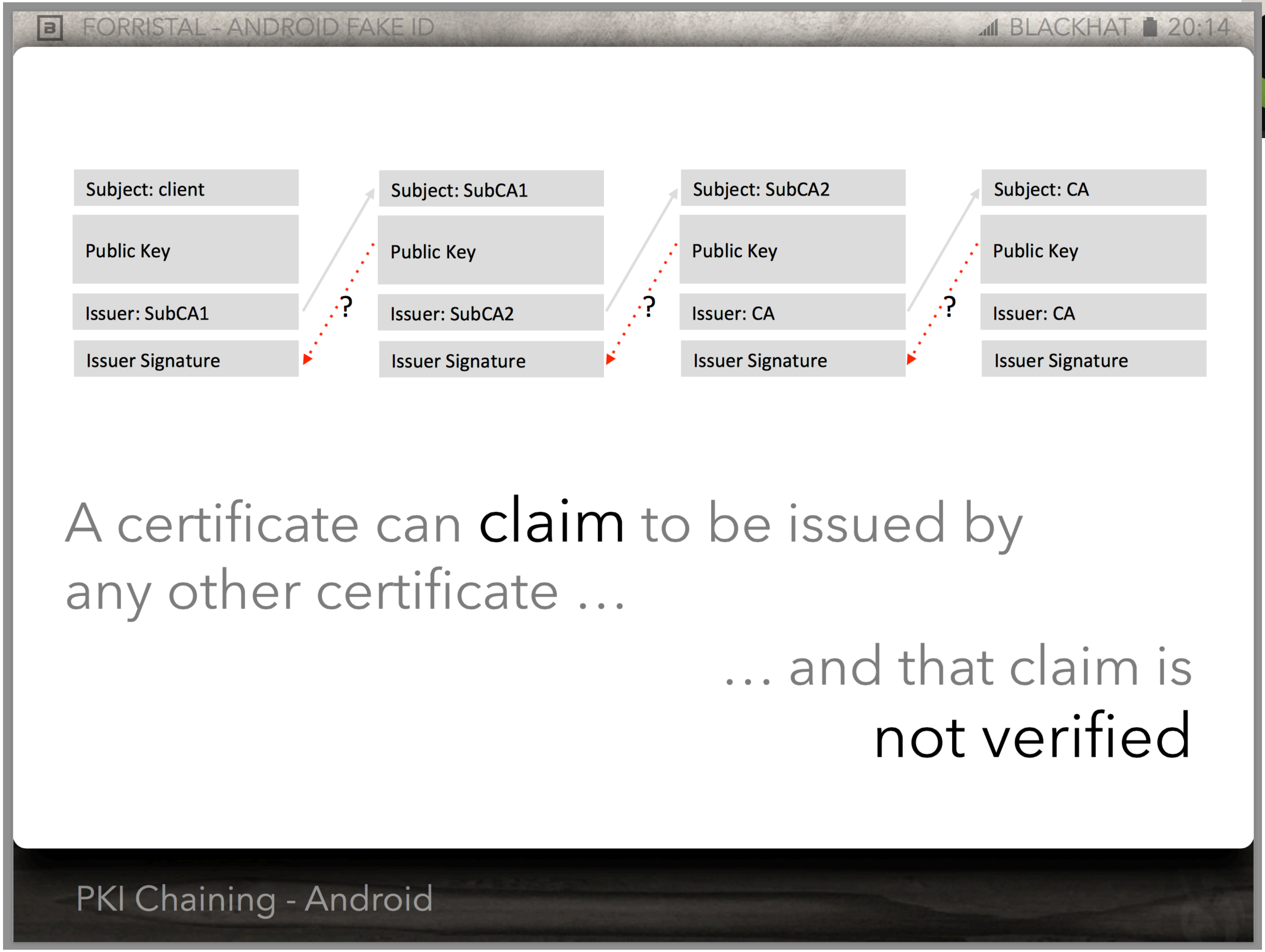
Improper Certificate Path Validation: Fake ID

- Every Android application is digitally signed
- Android OS verifies the signature as a part of installation process
  - Equivalent to certificate verification in SSL/TLS
- Verification code comes from Apache Harmony

Apache Harmony™

- This code has a problem on certificate path validation

“there is a conspicuous **absence of cryptographic verification** of any issuer cert claims, instead defaulting to **simple subjectDN to issuerDN string matching**.”



FORRISTAL - ANDROID FAKE ID

BLACKHAT 20:14

Subject: client	Subject: SubCA1	Subject: SubCA2	Subject: CA
Public Key	Public Key	Public Key	Public Key
Issuer: SubCA1	Issuer: SubCA2	Issuer: CA	Issuer: CA
Issuer Signature	Issuer Signature	Issuer Signature	Issuer Signature

A certificate can **claim** to be issued by any other certificate …

… and that claim is **not verified**

PKI Chaining - Android

From the presentation at BlackHat2014

JarUtils::findCert (vulnerable)

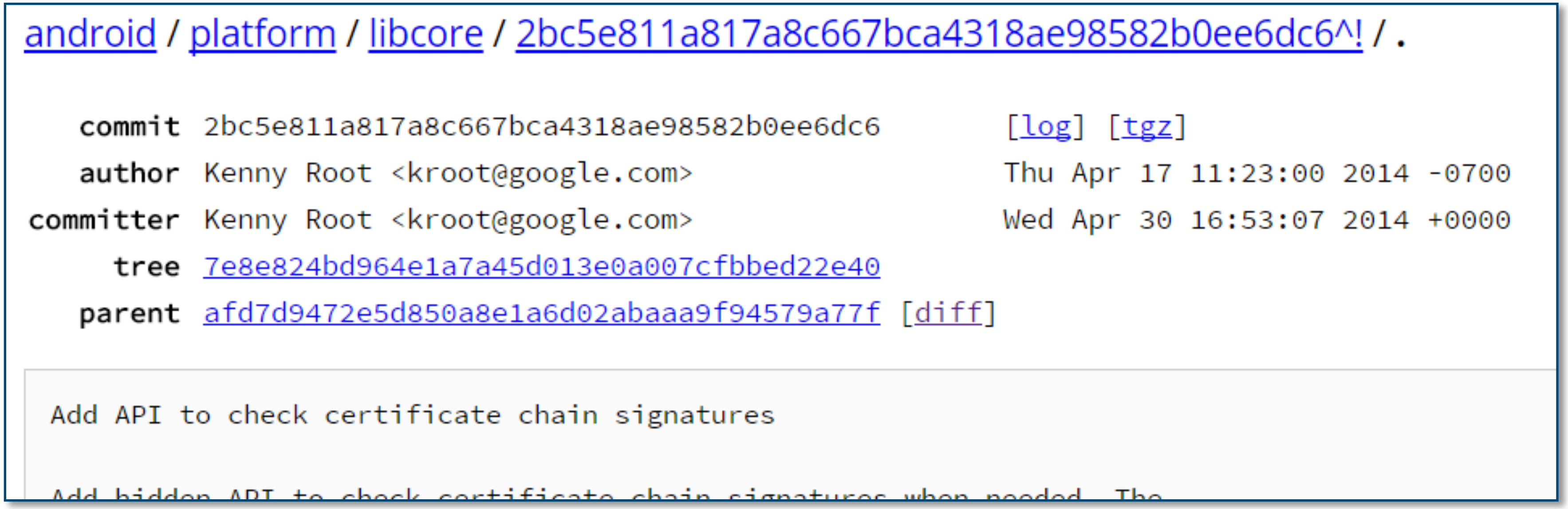
JarUtils.java

```java
private static X509Certificate
findCert(Principal issuer, X509Certificate[] candidates) {
    for (int i = 0; i < candidates.length; i++) {
          if (issuer.equals(candidates[i].getSubjectDN())) {
              return candidates[i];
          }
    }
```

Picks up a certificate just matching the subjectDN.
The signature is not validated.

Fixing Fake ID

The fixed code verifies the signature when picking up a certificate.



android / platform / libcore / 2bc5e811a817a8c667bca4318ae98582b0ee6dc6^! / .

```
    commit 2bc5e811a817a8c667bca4318ae98582b0ee6dc6      [log] [tgz]
    author Kenny Root <kroot@google.com>                 Thu Apr 17 11:23:00 2014 -0700
 committer Kenny Root <kroot@google.com>                 Wed Apr 30 16:53:07 2014 +0000
      tree 7e8e824bd964e1a7a45d013e0a007cfbbed22e40
    parent afd7d9472e5d850a8e1a6d02abaaa9f94579a77f [diff]
```

```
Add API to check certificate chain signatures

Add hidden API to check certificate chain signatures when needed. The
```

JarUtils::findCert (fixed)

JarUtils.java

```
private static X509Certificate
findCert(Principal issuer, X509Certificate[] candidates,
         X509Certificate subjectCert, boolean chainCheck) {
    for (int i = 0; i < candidates.length; i++) {
          if (issuer.equals(candidates[i].getSubjectDN())) {
              if (chainCheck) {
                  try {
                      subjectCert.verify(
                             candidates[i].getPublicKey());
                  } catch (Exception e) {
                      continue;
                  }
              }
              return candidates[i];
          }
    }
```

The signature is verified

Improper certificate path validation: Apple iOS

TWSL2011-007: iOS SSL Implementation Does Not Validate Certificate Chain

http://blog.spiderlabs.com/2011/07/twsl2011-007-ios-ssl-implementation-does-not-validate-certificate-chain.html

https://www3.trustwave.com/spiderlabs/advisories/TWSL2011-007.txt

“iOS's SSL certificate parsing contains a flaw where it fails to check the **basicConstraints** parameter of certificates in the chain.”

What is ‘**basicConstraints**’?

Example: www.jpcert.or.jp.

- Issuer:
  - C=US
  - O=Symantec Corporation
  - OU=Symantec Trust Network
  - CN=Symantec Class 3 EV SSL CA - G3
- **Subject**:
  - serialNumber=0100-05-006504
  - C=JP
  - postalCode=101-0054
  - ST=Tokyo
  - L=Chiyoda-ku
  - streetAddress="Hirose Bldg. 11F, 3-17 Kanda-nishikicho"
  - O="Japan Computer Emergency Response Team Coordination Center"
  - OU="System Administration Group"
  - **CN=www.jpcert.or.jp**
- X509v3 extensions:
  - **X509v3 Subject Alternative Name:**
    - **DNS:www.jpcert.or.jp**
  - X509v3 Basic Constraints:
    - CA:FALSE

Basic Constraints is specified in RFC5280.

What does basicConstraints indicate?

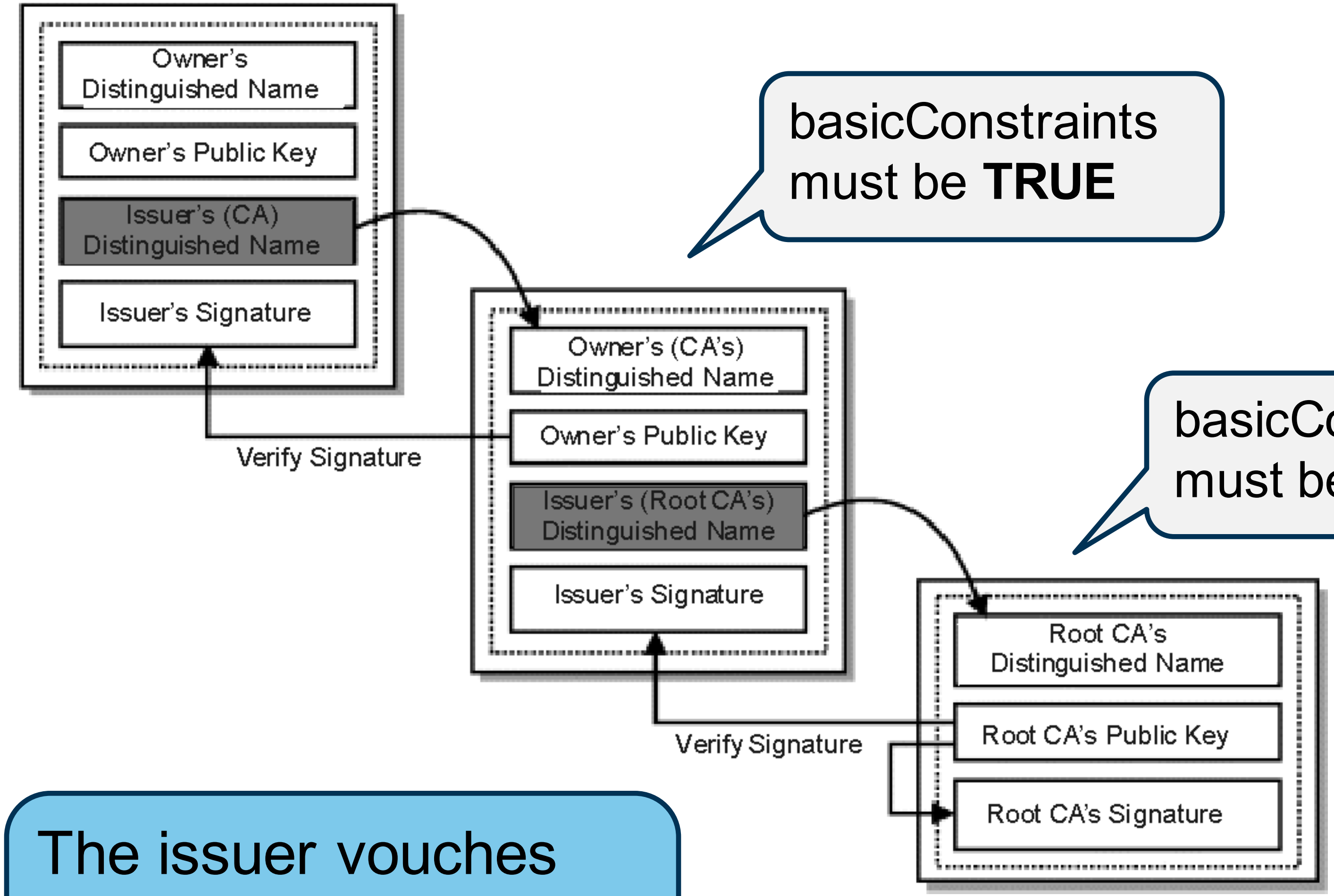
[from RFC5280 section 4.2.1.9]

(basicConstraints) indicates whether the certified public key may be used to verify certificate signatures.

If (basicConstraints is not present or the value is false), then the certified public key MUST NOT be used to verify certificate signatures.

CA certificates must have basicConstraints as TRUE, any other (nonCA) certificates must have basicConstraints as FALSE.

basicConstraints and Certificate Path Validation


Owner's Distinguished Name
Owner's Public Key
Issuer's (CA) Distinguished Name
Issuer's Signature
Verify Signature
Owner's (CA's) Distinguished Name
Owner's Public Key
Issuer's (Root CA's) Distinguished Name
Issuer's Signature
Verify Signature
Root CA's Distinguished Name
Root CA's Public Key
Root CA's Signature

basicConstraints must be **TRUE**

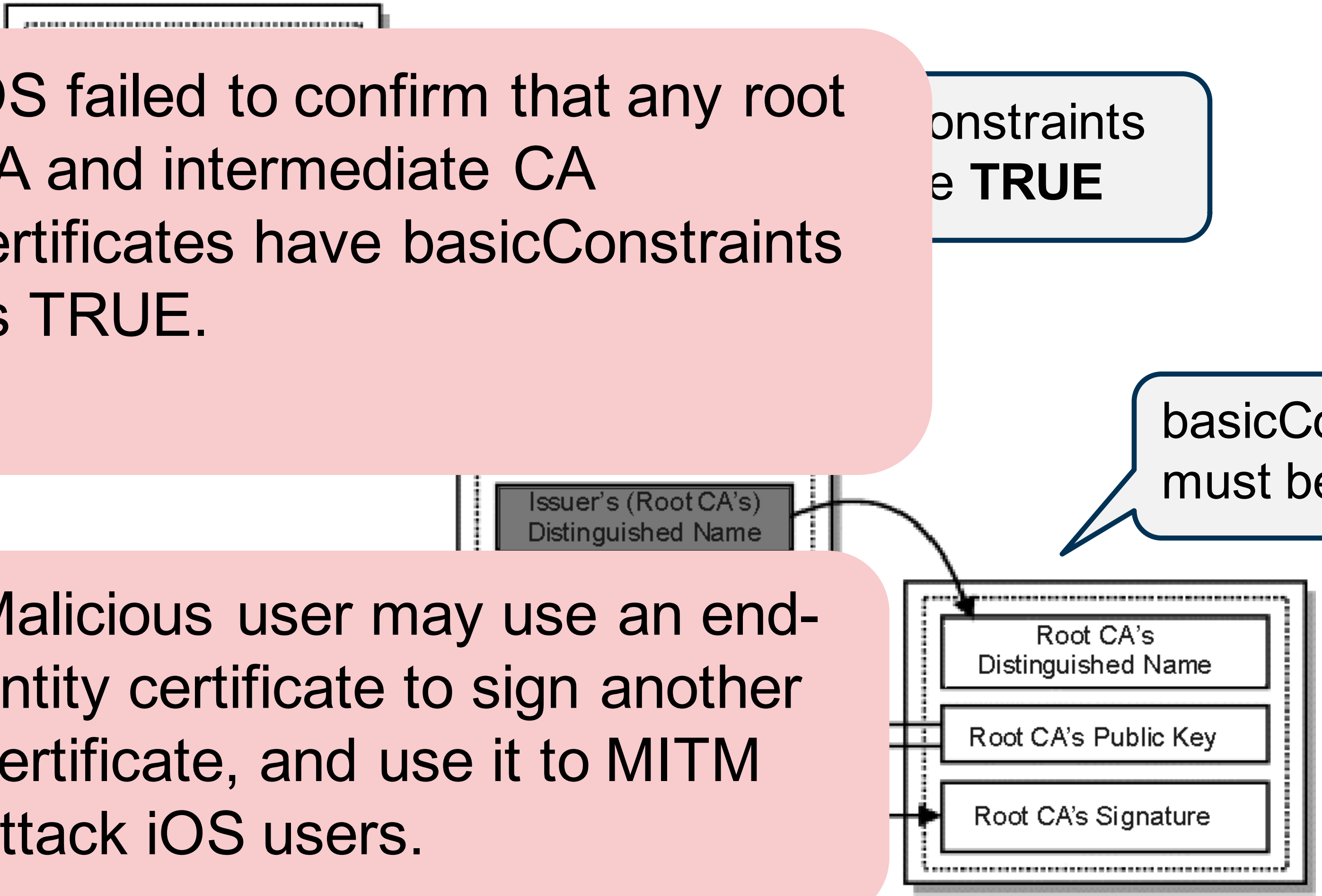
basicConstraints must be **TRUE**

The issuer vouches that the certificate is CA or not CA.

basicConstraints and Certificate Path Validation

iOS failed to confirm that any root CA and intermediate CA certificates have basicConstraints as TRUE.

onstraints
e **TRUE**

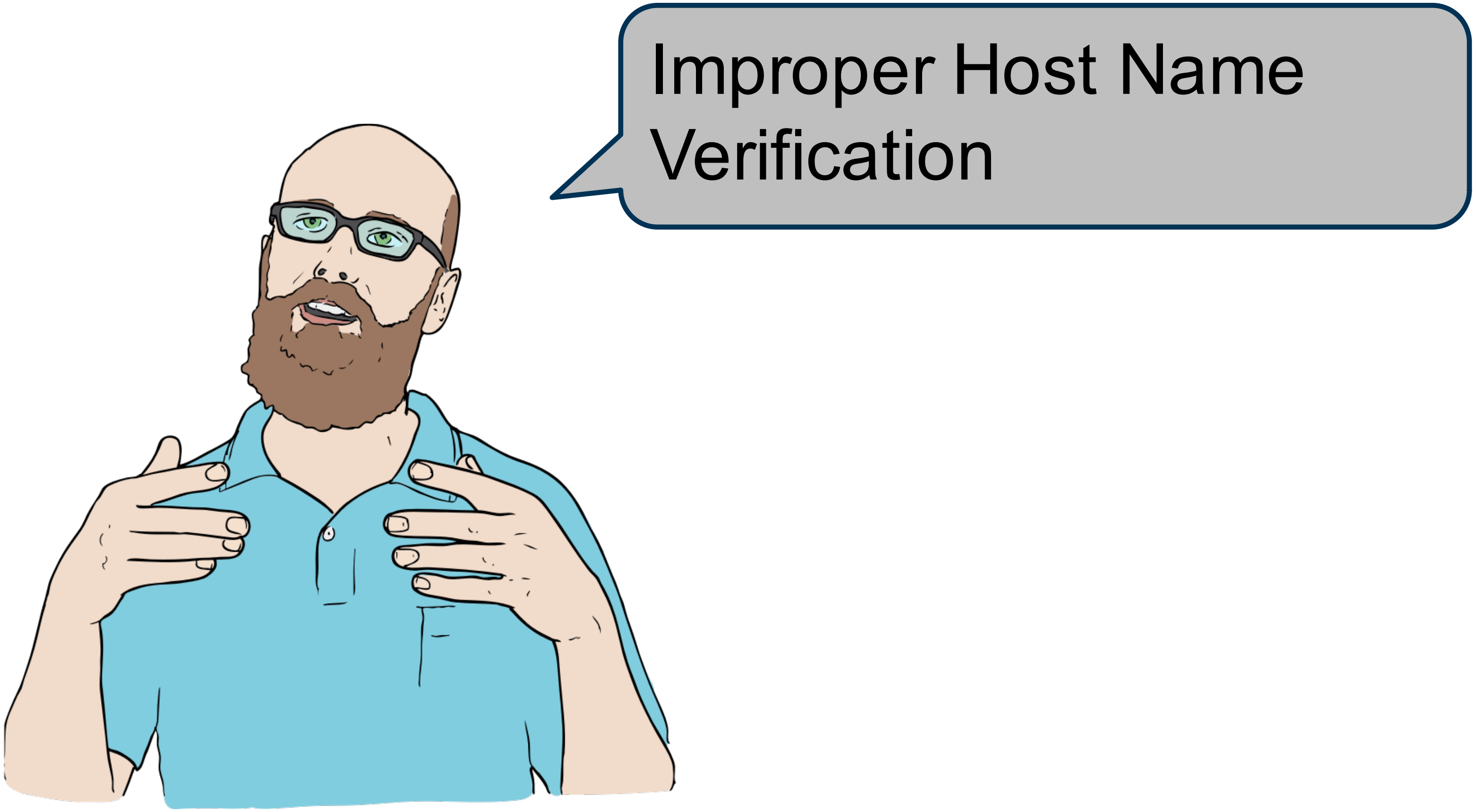

Issuer's (Root CA's) Distinguished Name
Root CA's Distinguished Name
Root CA's Public Key
Root CA's Signature

basicConstraints must be **TRUE**

Malicious user may use an end-entity certificate to sign another certificate, and use it to MITM attack iOS users.

that the certificate is CA or not CA.

Real Vulnerabilities: Pattern3



Apache HttpComponents and Apache Axis

CVE-2014-3577 Apache HttpComponents client: Hostname verification susceptible to MITM attack
http://seclists.org/fulldisclosure/2014/Aug/48

“Apache HttpComponents … may be susceptible to a 'Man in the Middle Attack' due to **a flaw in the default hostname verification** during SSL/TLS when a **specially crafted** server side certificate is used.”

Similar issues are reported for Apache Commons HttpClient (CVE-2012-6153,CVE-2012-5783)

Apache HttpComponents and Apache Axis

… a **specially crafted** server side certificate is used.”

↓

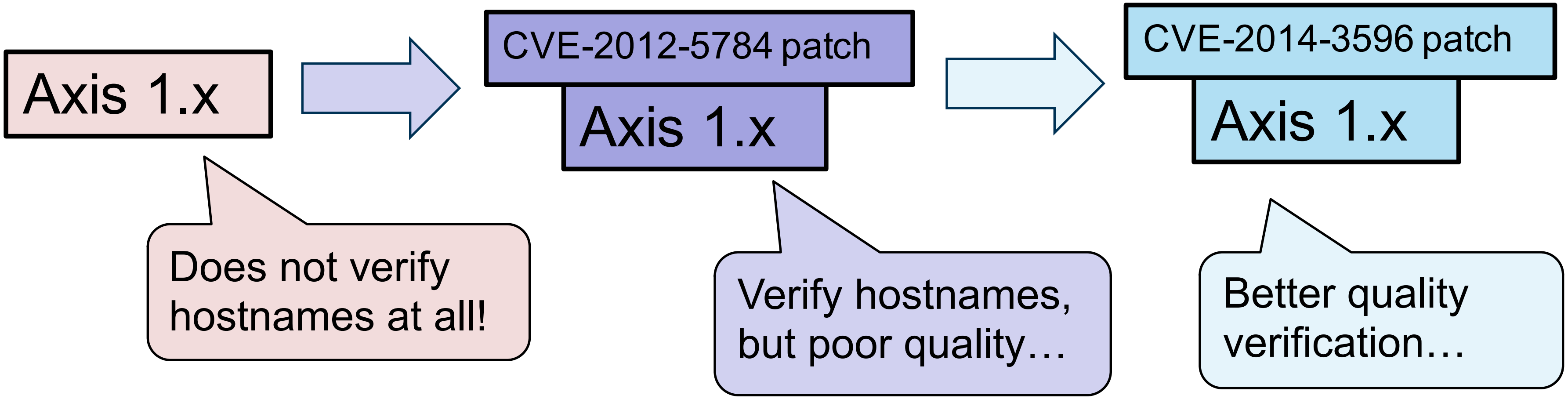
“a (crafted) DN with a O field such as

O="foo,CN=www.apache.org”

and …… ordered such that the O appears prior to the CN field would incorrectly match on the <www.apache.org> ..."

Apache HttpComponents and Apache Axis

[from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3596]
The getCN function in Apache Axis 1.4 and earlier does not properly verify that the server hostname matches a domain name in the subject's Common Name (CN) or subjectAltName field of the X.509 certificate, which allows man-in-the-middle attackers to spoof SSL servers via a certificate with a subject that specifies a common name in a field that is not the CN field.
NOTE: this issue exists because of an incomplete fix for CVE-2012-5784.



Axis 1.x

Does not verify hostnames at all!

CVE-2012-5784 patch

Axis 1.x

Verify hostnames, but poor quality…

CVE-2014-3596 patch

Axis 1.x

Better quality verification…

CVE-2012-5784 fix

```
private static void verifyHostName(final String host, X509Certificate cert)
                        throws SSLException {
            String cn = getCN(cert);
            String[] subjectAlts = getDNSSubjectAlts(cert);
            verifyHostName(host, cn.toLowerCase(Locale.US), subjectAlts);
}
```

```
private static String getCN(X509Certificate cert) {
            String subjectPrincipal = cert.getSubjectX500Principal().toString();
            return getCN(subjectPrincipal);
}
```

```
private static String getCN(String subjectPrincipal) {
            StringTokenizer st = new StringTokenizer(subjectPrincipal, ",");
            while(st.hasMoreTokens()) {
                        String tok = st.nextToken().trim();
                        if (tok.length() > 3) {
                                    if (tok.substring(0, 3).equalsIgnoreCase("CN=")) {
                                                return tok.substring(3);
                                    }
                        }
            }
            return null;
}
```

Recognizes the data as a comma-separated string list and searches “CN=“.
Hence it detects “CN=“ inside some attribute string.

CVE-2014-3596 fix(1)

```
private static void verifyHostName(final String host, X509Certificate cert)
                                throws SSLException {
                String[] cns = getCNs(cert);
                String[] subjectAlts = getDNSSubjectAlts(cert);
                verifyHostName(host, cns, subjectAlts);
}
```

```
private static String[] getCNs(X509Certificate cert) {
                String subjectPrincipal = cert.getSubjectX500Principal().toString();
                return getCNs(subjectPrincipal);
}
```

```
private static String[] getCNs(String subjectPrincipal) {
    ……..
}
```

CVE-2014-3596 fix(2)

```
private static void verifyHostName(final String host, X509Certificate cert) throws SSLException { …….. }
```

```
private static String[] getCNs(X509Certificate cert) { …….. }
```

```
private static String[] getCNs(String subjectPrincipal) {
    if (subjectPrincipal == null) {
        return null;
    }
    final List cns = new ArrayList();
    try {
        final LdapName subjectDN = new LdapName(subjectPrincipal);
        final List rdns = subjectDN.getRdns();
        for (int i = rdns.size() - 1; i >= 0; i--) {
            final Rdn rds = (Rdn) rdns.get(i);
            final Attributes attributes = rds.toAttributes();
            final Attribute cn = attributes.get("cn");
            if (cn != null) {
                try {
                    final Object value = cn.get();
                    if (value != null) {
                        cns.add(value.toString());
                    }
                }
                catch (NamingException ignore) {}
            }
        }
    }
    catch (InvalidNameException ignore) { }
    return cns.isEmpty() ? null : (String[]) cns.toArray(new String[ cns.size() ]);
}
```

This code uses **LdapName** class to find **CN** attribute.

Another Improper hostname verification: CVE-2013-4073 Ruby

Hostname check bypassing vulnerability in SSL client (CVE-2013-4073)
https://www.ruby-lang.org/en/news/2013/06/27/hostname-check-bypassing-vulnerability-in-openssl-client-cve-2013-4073/

Ruby
A PROGRAMMER'S BEST FRIEND

"Ruby's SSL client implements hostname identity check but it does not properly handle hostnames in the certificate that contain null bytes."

SSL/TLS AND CERTIFICATE VERIFICATION

VULNERABILITIES IN THE REAL WORLD

LESSONS LEARNED FROM VULNERABILITIES

REFERENCES

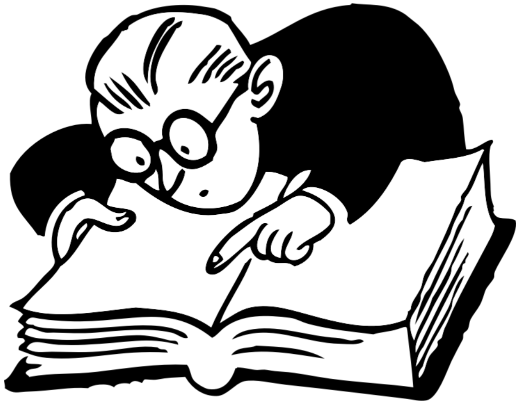
Point1: Do Verify Certificates

- Certificate Verification is THE mandatory procedure for SSL/TLS communication
- Be careful if disabling verification for debugging
  - Check the configuration for release builds
  - Your release build behaves properly?
- For Java/Android applications
  - Don't ignore **SSLException**
  - Don't disable **TrustManager**
  - Don't disable **HostnameVerifier**

Point2: Verify Certificate Path and Hostname Properly

BE CAREFUL!
Certificate path validation and hostname verification are complicated tasks.

- Basic Principle: When using third-party libraries, use them as is, customization should be as smallest as possible
- When you need to implement the verification procedure by yourself
  - Understand the specification properly
  - Test verification behaviors carefully
  - Include test patterns reflecting the known attack vectors



Best Practice for Using Cryptography

"In general, try using the highest level of pre-existing framework implementation that can support your use case.

....

If you cannot avoid implementing your own protocol, we strongly recommend that you *do not* implement your own cryptographic algorithms."

https://developer.android.com/guide/practices/security.html#Crypto

Note: Debugging with Proxy Tools

Proxy tools are useful for testing verification behavior

- Responding with a self-signed certificate or a dynamically generated certificate
- Certificates with improper hostnames
- Expired certificates
- Revoked certificates

•Famous / popular proxy tools are Burp proxy, dsniff, Fiddler, mitmproxy, …

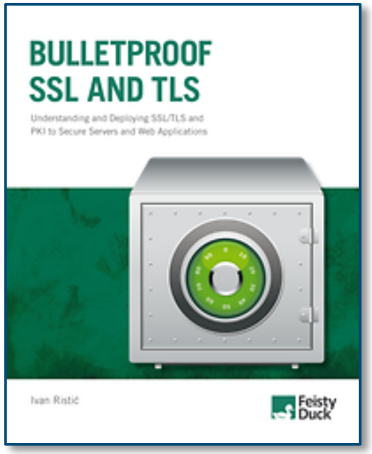
SSL/TLS AND CERTIFICATE VERIFICATION

VULNERABILITIES IN THE REAL WORLD

LESSONS LEARNED FROM VULNERABILITIES

REFERENCES

BOOKS


BULLETPROOF SSL AND TLS
Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications
Ivan Ristić
Feisty Duck

- Bulletproof SSL and TLS
  - https://www.feistyduck.com/books/bulletproof-ssl-and-tls/

And if you can read Japanese…

- マスタリングTCP/IP SSL/TLS編
  - http://shop.ohmsha.co.jp/shop/shopdetail.html?brandcode=000000001666&search=4-274-06542-1

WWW resources

- Introduction to Public-Key Cryptography
  - https://developer.mozilla.org/en-US/docs/Introduction_to_Public-Key_Cryptography
- Exciting Updates to Certificate Verification in Gecko
  - https://blog.mozilla.org/security/2014/04/24/exciting-updates-to-certificate-verification-in-gecko/
- Japan smartphone Security Association (JSSEC), Android Application Secure Design/Secure Coding Guidebook
  - https://www.jssec.org/dl/android_securecoding_en_20140701.pdf
- OnionKit by Android Library Project for Multi-Layer Network Connections (Better TLS/SSL and Tor)
  - https://github.com/guardianproject/OnionKit

WWW resources

- SSL Vulnerabilities: Who listens when Android applications talk?
  - http://www.fireeye.com/blog/technical/2014/08/ssl-vulnerabilities-who-listens-when-android-applications-talk.html
- Defeating SSL Certificate Validation for Android Applications
  - https://secure.mcafee.com/us/resources/white-papers/wp-defeating-ssl-cert-validation.pdf
- CERT/CC Vulnerability Note VU#582497: Multiple Android applications fail to properly validate SSL certificates
  - https://www.kb.cert.org/vuls/id/582497

WWW resources (Certificate and Public Key Pinning)

- OWASP, Certificate and Public Key Pinning
  - https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
- OWASP, Pinning Cheat Sheet
  - https://www.owasp.org/index.php/Pinning_Cheat_Sheet
- Java Pinning (Flowdalic / java-pinning)
  - https://github.com/Flowdalic/java-pinning
- Android Pinning by Moxie Marlinspike (moxie0 / AndroidPinning)
  - https://github.com/moxie0/AndroidPinning

JPCERT Coordination Center
(https://www.jpcert.or.jp/)

Secure Coding
(https://www.jpcert.or.jp/securecoding/)

Contact: secure-coding@jpcert.or.jp