WebPKI and Trust

CS249i: The Modern Internet
November 10, 2021

Zane Ma
zanema@gatech.edu
https://zanema.com
Authentication

Definition: proving/verifying the identity of something

Keystone network security goal: confidentiality/privacy is meaningless without authentication

What something? What identifier? How to prove/verify?
# Authentication

Definition: proving/verifying the identity of something

<table>
<thead>
<tr>
<th>Something</th>
<th>Identifier</th>
<th>Proof Method</th>
<th>Proof Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Blood type</td>
<td>Blood test</td>
<td>Direct: functional test</td>
</tr>
<tr>
<td>Person</td>
<td>Name</td>
<td>ID card</td>
<td>Indirect: trusted entity</td>
</tr>
<tr>
<td>Client/User</td>
<td>Email address</td>
<td>Confirmation email</td>
<td>Direct: Request + Response</td>
</tr>
<tr>
<td>Client/User</td>
<td>Account owner</td>
<td>Password</td>
<td>Direct: Shared secret</td>
</tr>
<tr>
<td>Server</td>
<td>DNS Name</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Web Server Auth #1: Direct
Web Server Auth #1: Direct
Web Server Auth #1: Direct

Web Browser
Email Client

Web Server
Email Server
Web Server Auth #1: Direct
Web Server Auth #2: Indirect

Trust Establishment

Identity verification

Certificate Authority

Web Browser
Email Client

Web Server
Email Server

WebPKI and Trust • Zane Ma
TLS + Web PKI

1. Clients Establish Trust in CAs

Certificate Authority

CA Certificate

Root store inclusion

Web Browser
Email Client

Web Server
Email Server
TLS + Web PKI

1. Clients Establish Trust in CAs
   - Root store inclusion

2. CAs Verify Server Identity
   - Signs
   - CA Certificate
   - Leaf Certificate

- Certificate Authority
- Web Browser
- Email Client
- Web Server
- Email Server
1. Clients Establish Trust in CAs

2. CAs Verify Server Identity

3. TLS: Servers Send Proof of Identity
WebPKI and Trust • Zane Ma

Threat Model

Potential attacks:
1. Compromise private key of server
2. Trick CA to misissue certificate
3. Exploit certificate validation on client

2. CAs Verify Server Identity

Web Browser
Email Client

3. TLS

Web Server
Email Server

Certificate Authority
CA Certificate
Leaf Certificate

Signs

😈
Potential attacks:
1. Compromise private key of server
2. Trick CA to misissue certificate
3. Exploit certificate validation on client
Web PKI

1. Clients Establish Trust in CAs

2. CAs Verify Server Identity

3. TLS: Servers Send Proof of Identity
Certificate Issuance

Goals:

1) verify that a network identifier (i.e., IP address or DNS Name) controls some cryptographic public key

2) generate a certificate that attests to this linkage

How to verify? What does “control” mean?

But first, what’s a certificate?
Certificates

Signed document that attests to the connection between an identity and an authorized public key

TLS uses:
- X.509 certificate schema (what fields in what order)
- ASN.1 data format (field types, expression syntax)
- DER encoding (converting everything to bytes)

More info: https://letsencrypt.org/docs/a-warm-welcome-to-asn1-and-der/
Certificates

Certificate root
TBS certificate

Validity
  datetime:start “Apr 30 08:42:02 2018 GMT”

Issuer
  Field
    oid:commonName “Let’s Encrypt”
    string:name

Subject
  Field
    oid:commonName
    string:name “gatech.edu”

Subject Public Key Info
  oid:rsaEncryption
  Subject Public Key
    int:modulus
    int:exponent

Extensions

Signature
  oid:sha256WithRSAEnc.
  bytes:signatureValue
Certificates

<table>
<thead>
<tr>
<th>Certificate root</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBS certificate</td>
</tr>
<tr>
<td>Validity</td>
</tr>
<tr>
<td>datetime:start</td>
</tr>
<tr>
<td>&quot;Apr 30 08:42:02 2018 GMT&quot;</td>
</tr>
<tr>
<td>datetime:end</td>
</tr>
<tr>
<td>&quot;Dec 21 22:24:54 2019 GMT&quot;</td>
</tr>
<tr>
<td>Issuer</td>
</tr>
<tr>
<td>Field</td>
</tr>
<tr>
<td>oid:commonName</td>
</tr>
<tr>
<td>&quot;Let’s Encrypt&quot;</td>
</tr>
<tr>
<td>string:name</td>
</tr>
<tr>
<td>Subject</td>
</tr>
<tr>
<td>Field</td>
</tr>
<tr>
<td>oid:commonName</td>
</tr>
<tr>
<td>&quot;gatech.edu&quot;</td>
</tr>
<tr>
<td>Subject Public Key Info</td>
</tr>
<tr>
<td>oid:rsaEncryption</td>
</tr>
<tr>
<td>Subject Public Key</td>
</tr>
<tr>
<td>int:modulus</td>
</tr>
<tr>
<td>int:exponent</td>
</tr>
<tr>
<td>Extensions</td>
</tr>
<tr>
<td>Signature</td>
</tr>
<tr>
<td>oid:sha256WithRSAEnc.</td>
</tr>
<tr>
<td>bytes:signatureValue</td>
</tr>
</tbody>
</table>
Certificates

Certificate root
TBS certificate
Validity
  datetime:start “Apr 30 08:42:02 2018 GMT”

Issuer
  Field
    oid:commonName “Let’s Encrypt”
    string:name

Subject
  Field
    oid:commonName
    string:name “gatech.edu”

Subject Public Key Info
  oid:rsaEncryption
  Subject Public Key
    int:modulus
    int:exponent

Extensions

Signature
  oid:sha256WithRSAEnc.
  bytes:signatureValue

Subject identity and public key
Issuer (Certificate Authority)
Certificates

Certificate root
TBS certificate

Validity
  datetime:start “Apr 30 08:42:02 2018 GMT”

Issuer
  Field
    oid:commonName “Let’s Encrypt”
    string:name

Subject
  Field
    oid:commonName
    string:name “gatech.edu”

Subject Public Key Info
  oid:rsaEncryption
  Subject Public Key
    int:modulus
    int:exponent

Extensions

Signature
  oid:sha256WithRSAEnc.
  bytes:signatureValue
Certificates

Certificate root
TBS certificate
  Validity
    datetime:start “Apr 30 08:42:02 2018 GMT”
  Issuer
    Field
      oid:commonName “Let’s Encrypt”
      string:name
  Subject
    Field
      oid:commonName
      string:name “gatech.edu”
  Subject Public Key Info
    oid:rsaEncryption
    Subject Public Key
      int:modulus
      int:exponent
  Extensions: Permitted key usages, policies, additional identities
    (Subject Alternate Identity)

Subject identity and public key

Issuer (Certificate Authority)

Validity Period

Extensions: Permitted key usages, policies, additional identities
  (Subject Alternate Identity)
Certificates

Subject identity and public key

Issuer (Certificate Authority)

Validity Period

Extensions: Permitted key usages, policies, additional identities (Subject Alternate Identity)

Signature: TBS signed by Issuer public key
Certificate Chains

CA Certificate A

Issuer: Foo CA
Subject: Foo CA
Public key: 0xabc123...

Signature: 0x823cdf...

Self-signed
Certificate Chains

CA Certificate A
Issuer: Foo CA
Subject: Foo CA
Public key: 0xabc123...
Signature: 0x823cdf...

Self-signed

Leaf Certificate
Issuer: Foo CA
Subject: domain.com
Public key: 0xb21f35...
Signature: 0xa392b...
Certificate Chains

CA Certificate A

Issuer: Foo CA
Subject: Foo CA
Public key: 0xabc123...
Signature: 0x823cdf...
Self-signed

CA Certificate B

Issuer: Bar CA
Subject: Foo CA
Public key: 0xabc123...
Signature: 0xfa92bc4...
Cross-signed

Same Subject + pubkey
Different Issuer

Leaf Certificate

Issuer: Foo CA
Subject: domain.com
Public key: 0xb21f35...
Signature: 0xa392b...
Cross-Signing

CA Certificate A
Issuer: Foo CA
Subject: Foo CA
Public key: 0xabc123...
Signature: 0x823cdf...
Self-signed

Leaf Certificate
Issuer: Foo CA
Subject: domain.com
Public key: 0xb21f35...
Signature: 0xa392b...

Same Subject + pubkey
Different Issuer

CA Certificate B
Issuer: Bar CA
Subject: Foo CA
Public key: 0xabc123...
Signature: 0xfa92bc4...
Cross-signed
Cross-Signing

Broke: one certificate chain

Woke: many possible certificate chains

Broke: certificates as nodes, signatures as edges

Woke: certificates as edges, entities as nodes
Entity chains

Entity A
Name: Foo CA
Public key: 0xabc123…

Entity B
Name: domain.com
Public key: 0xb21f35…

Issuer: Foo CA
Subject: Foo CA
Public key: 0xabc123…
Signature: 0x823cdf…

CA Certificate A

Issuer: Foo CA
Subject: domain.com
Public key: 0xb21f35…
Signature: 0xa392b…

Leaf Certificate
Entity chains

Entity A
Name: Foo CA
Public key: 0xabc123...

Leaf Certificate

Entity B
Name: domain.com
Public key: 0xb21f35...

CA Certificate A
Entity chains

Entity A
Name: Foo CA
Public key: 0xabc123...

Leaf Certificate
Entity B
Name: domain.com
Public key: 0xb21f35...

CA Certificate A

Entity C
Name: Bar CA
Public key: 0xc82dae...

CA Certificate B
Certificate Issuance

Goals:

1) verify that a network identifier (i.e., IP address or DNS Name) controls some cryptographic public key

2) generate a certificate that attests to this linkage

How to verify? What does “control” mean?
Historical Issuance (ca. 2012)

Confirming the Applicant as the Domain Name Registrant directly with the Domain Name Registrar;

Communicating directly with Registrant via address, email, or telephone number provided by the Registrar;

Communicating directly with the Registrant using the contact information listed in the WHOIS record's “registrant”, “technical”, or “administrative” field;

Communicating with the Domain's administrator using an email address created by pre-pending ‘admin’, ‘administrator’, ‘webmaster’, ‘hostmaster’, or ‘postmaster’ followed by the Domain Name;

Relying upon a Domain Authorization Document;

Having the Applicant demonstrate practical control over the FQDN by making an agreed-upon change to information found on an online Web page identified by a uniform resource identifier containing the FQDN;

Using any other method of confirmation, provided that the CA maintains documented evidence that it establishes that the Applicant is the Registrant or has control over the FQDN to at least the same level of assurance as those methods previously described.
Historical Issuance (ca. 2012)

If CA == DNS Registrar
Confirming the Applicant as the Domain Name Registrant directly with the Domain Name Registrar:

- Communicating directly with Registrant via address, email, or telephone number provided by the Registrar;
- **WHOIS info directly from registrar**
  Communicating directly with the Registrant using the contact information listed in the WHOIS record’s “registrant”, “technical”, or “administrative” field;

  Communicating with the Domain’s administrator using an email address created by pre-pending ‘admin’, ‘administrator’, ‘webmaster’, ‘hostmaster’, or ‘postmaster’ followed by the Domain Name;

  Relying upon a Domain Authorization Document; **Removed**

  Having the Applicant demonstrate practical control over the FQDN by making an agreed-upon change to information found on an online Web page identified by a uniform resource identifier containing the FQDN;

  Using any other method of confirmation, provided that the CA maintains documented evidence that it establishes that the Applicant is the Registrant or has control over the FQDN to at least the same level of assurance as those methods previously described. **Removed**
Modern Issuance (ca. 2021)

<table>
<thead>
<tr>
<th>Contact Method</th>
<th>Contact Info Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone, Fax, Postal Mail</td>
<td>WHOIS, DNS (TXT, CAA), ‘admin’-like email</td>
</tr>
<tr>
<td>SMS Email</td>
<td></td>
</tr>
</tbody>
</table>

A. Manual Contact

- Step 1. CA sends token to client

B. Random Token

- Step 2. CA retrieves token from:
  - DNS server
  - HTTP / TLS server

Automatable
ACME

- Automates certificate issuance + revocation (to be discussed)

Open Source Software (OSS):

- ACME client
- ACME client + web server
- ACME CA

- certbot
- Caddy®
- Let’s Encrypt/Boulder
Let’s Encrypt

Source:

“Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web”


ACM Conference on Computer and Communications Security (CCS), November 2019
Let’s Encrypt

% of top sites that use Let’s Encrypt

Automation + open standards —> free + easy-to-use

Combined with browser carrots/sticks, LE has enabled massive HTTPS adoption
HTTPS Adoption

(14-day moving average, source: Firefox Telemetry)
Certificate Revocation

Why do we need to revoke certificates?

RFC 5280 Revocation Reasons

- unspecified (0)
- keyCompromise (1)
- cACompromise (2)
- affiliationChanged (3)
- superseded (4)
- cessationOfOperation (5)
- certificateHold (6)
- removeFromCRL (8)
- privilegeWithdrawn (9)
- aACompromise (10)

Only need to revoke unexpired certificates

Shorter certificate validity periods (enabled by ACME) leads to fewer revocations

How can we revoke certificates?

Engineering problem
Online Revocation Checks

Before validating TLS cert, contact CA and retrieve a Certificate Revocation List (**CRL**) or use Online Certificate Status Protocol (**OCSP**) to check revocation for a single certificate.

CRLs: require additional retrieval of large (~MBs) file that blocks TLS validation.

OCSP: generates lots of traffic to CA. Privacy-concerns.

What if CA is unavailable/inaccessible? Fail-open!
Online Revocation Checks

TLS: Servers Send Proof of Identity

Web Browser
Email Client

Web Server
Email Server

Certificate Authority
Online Revocation Checks

CRL / OCSP request

Certificate Authority

Web Browser
Email Client

Web Server
Email Server

TLS: Servers Send Proof of Identity

Leaf Certificate
OCSP Stapling

Certificate Authority

CA signed & time-windowed OCSP response

Web Browser
Email Client

Web Server
Email Server
OCSP Stapling

Certificate Authority

CA signed &
time-windowed
OCSP response

TLS: Servers Send
Proof of Identity + OCSP

Web Browser
Email Client

Leaf Certificate

Web Server
Email Server
OCSP Must Staple

Certificate ext: client must expect OCSP

Web Browser
Email Client

Leaf Certificate

TLS: Servers Send Proof of Identity

CA signed & time-windowed OCSP response

Web Server
Email Server

OCSP Must Staple
CRL Aggregation + Pushing

- Web Server
- Email Server
- Certificate Authority
- Web Browser
- Email Client

Aggregate CRL

Transfer CRL info

Solves TLS connection blocking problem! But CRLs are still large....

TLS: Servers Send Leaf Certificate Proof of Identity

Web Server Email Server
CRL Aggregation + Pushing

Web Server
Email Server

Certiﬁcate Authority

Web Browser
Email Client

Solves TLS connection blocking problem!
But CRLs are still large….

Aggregate CRL

Transfer CRL info

CRLite: Bloom ﬁlters!

James Larisch*
Bruce M. Maggs†

David Choffnes*
Alan Mislove*

Dave Levin†
Christo Wilson*

Web Server
Email Server

TLS: Servers Send
Proof of Identity

Aggregate
CRL

Transfer CRL info

Browser Update Server

Leaf Certificate
Certificate Revocation

• Browsers currently implement a hodgepodge of CRL / OCSP / OCSP+stapling, with Firefox having experimental support for CRLite

• Outstanding question: is it worth streamlining revocation versus reducing certificate lifetimes (i.e., certificates become a caching mechanism)

• CA certificate revocation is a separate process that uses bespoke CRL push methods (separate Apple, Chrome, Firefox, Microsoft mechanisms)
Web PKI

1. Clients Establish Trust in CAs

2. CAs Verify Server Identity

Certificate Authority

CA Certificate

Leaf Certificate

3. TLS: Servers Send Proof of Identity

Web Browser

Email Client

Web Server

Email Server

Root store inclusion

Signs
Web PKI

1. Clients Establish Trust in CAs

2. CAs Verify Server Identity

3. TLS: Servers Send Proof of Identity

Certificate Authority

CA Certificate

Leaf Certificate

Web Browser
Email Client

Web Server
Email Server

Root store inclusion

Signs
Trust Issues

Current web PKI design: every trust anchor is a single point of failure

- DigiNotar hack
- CNNIC MITM
- Certinomis Cross-sign StartCom
- WoSign / StartCom Issues
- Symantec Misissuance

2011 — 2021

Question: How to evaluate CA trustworthiness? Look at their issuance practices…
Certificate Transparency

Certs are public! But annoying to collect —> CAs can hide in shadows

Idea: require provable, explicit disclosure of certs before trusting; community will find bad things

Mechanism:
1. CAs submits “precertificates” to CT logs, which anyone can run.
2. CT logs return signed certificate timestamp (SCT) that are embedded into final certificate
3. During TLS, browsers verify SCT against a set of trusted logs

More info: https://certificate.transparency.dev/howctworks/
Tusting CAs

Question: How to evaluate CA trustworthiness?

0. Business / government relationships

1. Independent audits from traditional accounting firms
   Compliance with WebTrust/ETSI standards, CA/Browser Forum Baseline Requirements, root store + CA policy

2. Mozilla also performs public discussion

Zlint: See how well CAs follow technical standards / pass unit tests!

CA transparency: disclosure and measurement of CA behavior
Root stores for 77% of global CDN top 200 user agents

Additional default root store for dozens of libraries / TLS clients
Root store providers

User Agents
- Chrome
- Chrome Mobile
- Opera
- Firefox
- Safari
- Mobile Safari
- Edge
- IE
- Chromium

Web Browsers

OS
- Windows
- macOS
- Alpine
- iOS
- Android
- Ubuntu
- Debian
- Fedora
- Amazon Linux

Other TLS Clients / Libraries
- OpenSSL
- GnuTLS
- BoringSSL
- Mbed TLS
- curl
- wget
- okhttp
- LibreSSL

Libraries / Frameworks
- NSS
- Electron
- NodeJS
- Java

Default / configured
WebPKI and Trust • Zane Ma

Root store programs

User Agents
- Chrome
- Chrome Mobile
- Opera
- Firefox
- Safari
- Mobile Safari
- Edge
- IE
- Chromium

Root Store Providers
- Windows
- macOS
- Alpine
- iOS
- Android

Default / configured

Web Browsers
- Chrome
- Chrome Mobile
- Opera
- Firefox
- Safari
- Mobile Safari
- Edge
- IE
- Chromium

Other TLS Clients / Libraries
- OpenSSL
- GnuTLS
- BoringSSL
- Mbed TLS
- curl
- wget
- okhttp
- LibreSSL
- +10 more

Libraries / Frameworks
- NSS
- Electron
- NodeJS
- Java

Root Store Programs
- Microsoft
- Apple
- Mozilla
- Java

OS
- Windows
- macOS
- Alpine
- iOS
- Android

Other TLS Clients / Libraries
- OpenSSL
- GnuTLS
- BoringSSL
- Mbed TLS
- curl
- wget
- okhttp
- LibreSSL
- +10 more

Default / configured
Recap

Authentication/identity is the foundation for confidentiality / integrity

Web PKI exists as a scalability/interception-avoidance mechanism for global web server auth

Certificates are not trustable themselves; they are attestation links between entities (name + pub key)

Trusting the right CAs is imperative - brittle current PKI design

Transparency can lead to better security, and opens the door for new research!
Closing Remarks

• Major network security challenges: malware, Distributed Denial of Service (DDoS), social engineering

• What is the core issue? Distinguishing good data from bad data? Cat-and-mouse game, sometimes indistinguishable (DDoS)

• Distinguishing good originators/creators of data: web server auth is a good start, but what we really want is authenticating web content

• Interesting challenges: identifier selection, verification protocols, policies built on top of authentication
Research Opportunities

**Improving web server auth**

- Empirical evaluation of CA behavior + policy
- Program analysis/testing of web PKI software
- Breaking the fuzzy grey-areas of certificate issuance

**New web content auth**

- Take a holistic look at how current web content is identified and authenticated
- Exploring new protocol design, implementation, deployment
WebPKI and Trust

CS249i: The Modern Internet

Zane Ma
Georgia Institute of Technology
zanema@gatech.edu
https://zanema.com