Assertion-Carrying Certificates

Waqar Aqeel, Zachary Hanif, James Larisch, Olamide Omolola, Taejoong Chung, Dave Levin, Bruce Maggs, Alan Mislove, Bryan Parno, Christo Wilson
The Public Key Infrastructure is how users know with whom they are communicating online.
Certificates encapsulate identity (who hosts are) and capability (what they can do)
Certificates encapsulate identity (who hosts are) and capability (what they can do)

Traditional PKI roles

**Subject Name**
Who the cert is about

**Issuer Name**
Who vetted the subject’s identity

**Expiration Dates**
When is the certificate no longer valid

**Public key and signature**
Attestation of cryptographic identity
The PKI has had to evolve to meet new threats, deployments, and opportunities.

**Traditional PKI roles**

- **Subject Name**: Who the cert is about
- **Issuer Name**: Who vetted the subject’s identity
- **Expiration Dates**: When is the certificate no longer valid
- **Public key and signature**: Attestation of cryptographic identity

**New additions to the PKI**

- **Key Usage**: Certificate signing, authentication
- **Subject Alternate Names**: Support deployments in CDNs
The PKI has had to evolve to meet new threats, deployments, and opportunities.

### Traditional PKI roles

**Subject Name**
Who the cert is about

**Issuer Name**
Who vetted the subject's identity

**Expiration Dates**
When is the certificate no longer valid

**Public key and signature**
Attestation of cryptographic identity

### New additions to the PKI

**Key Usage**
Certificate signing, authentication

**Subject Alternate Names**
Support deployments in CDNs

**Revocation Information**
New ways to deliver revocations

**Certificate Transparency**
Allows greater insight into CA (mis)behavior
The PKI must continue to evolve but adding new features is slow and laborious

<table>
<thead>
<tr>
<th>Traditional PKI roles</th>
<th>New additions to the PKI</th>
<th>Future additions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subject Name</strong></td>
<td><strong>Key Usage</strong></td>
<td><strong>Naming constraints</strong></td>
</tr>
<tr>
<td>Who the cert is about</td>
<td>Certificate signing, authentication</td>
<td>Let non-CAs issue their own certs, limited to domains they control</td>
</tr>
<tr>
<td><strong>Issuer Name</strong></td>
<td><strong>Subject Alternate Names</strong></td>
<td><strong>Signed exchanges</strong></td>
</tr>
<tr>
<td>Who vetted the subject’s identity</td>
<td>Support deployments in CDNs</td>
<td>Sign-over the hosting of some resources to a third party</td>
</tr>
<tr>
<td><strong>Expiration Dates</strong></td>
<td><strong>Revocation Information</strong></td>
<td><strong>Multi-rooted certificates</strong></td>
</tr>
<tr>
<td>When is the certificate no longer valid</td>
<td>New ways to deliver revocations</td>
<td>Minimize the reliance on a small set of trusted certificate authorities</td>
</tr>
<tr>
<td><strong>Public key and signature</strong></td>
<td><strong>Certificate Transparency</strong></td>
<td><strong>And many more!</strong></td>
</tr>
<tr>
<td>Attestation of cryptographic identity</td>
<td>Allows greater insight into CA (mis)behavior</td>
<td></td>
</tr>
</tbody>
</table>
Is there one extension we could add that would make the PKI:

- More evolvable
- More customizable to new deployments
- Easier to formally verify

**Insight:** A certificate is a set of constraints

<table>
<thead>
<tr>
<th>Name</th>
<th>Validity period</th>
<th>Allowed usages</th>
</tr>
</thead>
</table>

Why not encode constraints in small programs in the certificate?
Assertion-Carrying Certificates (ACCs)
Assertion-Carrying Certificates (ACCs)
Add small programs that must be run as part of the certificate’s validation

Rules

/* Does String end with Suffix */
endsWith(String, Suffix) :-
    string_concat(_, Suffix, String).

/* Is certificate Y a descendant of X */
descendant(X, Y) :-
    sign(X, Y),
    sign(Y, X), descendant(X, Y).

/* Does the certificate’s name end in Suffix */
nameConstrained(Cert, Suffix) :-
    hasName(Cert, Name), endsWith(Name, Suffix).

Assertions

/* Each descendant X of cert is name-constrained */
forall(descendant(cert, X),
    nameConstrained(X, '.example.com')).
Assertion-Carrying Certificates (ACCs)
Add small programs that must be run as part of the certificate’s validation

Rules

```prolog
/* Does String end with Suffix */
endsWith(String, Suffix) :-
    string_concat(_, Suffix, String).

/* Is certificate Y a descendant of X */
descendant(X,Y) :-
    signs(X,Y);
    signs(X,Z), descendant(Z,Y).

/* Does the certificate’s name end in Suffix */
nameConstrained(Cert, Suffix) :-
    hasName(Cert, Name), endsWith(Name, Suffix).
```

Define new capabilities
What it means to be name-constrained

Assertions

```prolog
/* Each descendant X of cert is name-constrained */
forall(descendant(cert,X),
    nameConstrained(X, "example.com").
```
Assertion-Carrying Certificates (ACCs)

Language goals

- All constraints across *all* certs in the chain must hold
  - Certs can never relax constraints further up the chain
  - Browsers can add their own constraints, as well

- The language should be concise and expressive
  - *Does not* need to be Turing-complete
  - *Should* be formally verifiable
  - *Must not* broaden the attack surface

- A logic-based programming language is a natural fit
## Assertion-Carrying Certificates (ACCs)
What is the appropriate constraint language?

<table>
<thead>
<tr>
<th>Prolog</th>
<th>Datalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Non-Turing-complete</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Declarative</td>
<td>✓</td>
</tr>
<tr>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Termination guaranteed</td>
<td>✓</td>
</tr>
<tr>
<td>½</td>
<td>✓</td>
</tr>
<tr>
<td>Amenable to static analysis</td>
<td>½</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fully expressive</td>
<td>½</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Negation</td>
<td>½</td>
</tr>
<tr>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Unbounded lists, numbers, strings</td>
<td></td>
</tr>
</tbody>
</table>

We might not need these
Assertion-Carrying Certificates (ACCs)
Allow for a far more agile PKI

Today’s PKI is slow to evolve

ACCs add small programs that must be run as part of the certificate’s validation

Ongoing and Future Efforts

Implementing long-desired features
Naming constraints, signed exchanges, and more

Re-implementing various browsers’ validation logic in Prolog/Datalog
Chrome, Firefox, mbedTLS — in far fewer lines of code

Exploring ways to verify correctness:
- Static analysis
- Certificate fuzzing
- Using the languages’ imputation

Is there any certificate that is valid but where constraint X does not hold?