Macaroons

Cookies with Contextual Caveats for Decentralized Authorization in the Cloud

Arnar Birgisson, Joe Politz, Úlfar Erlingsson, Ankur Taly, Michael Vrable, and Mark Lentczner
The Problem

- Cookies (and similar tokens) are ubiquitous in cloud authorization
- Good reasons: Simple, easily adopted
- But cookies are easy to steal, carry broad authority, and lack flexibility
- Macaroons = Better cookies, with arbitrary caveats, i.e. restrictions on access
Cookies with integrity

Standard technique:
Protect cookies with a MAC

9fad57c84ab8fec4813
Cookies with integrity

Standard technique:
Protect cookies with a MAC

$hmac(k_0, \text{9f}\ldots\text{13})$
Cookies with integrity

Standard technique:
Protect cookies with a MAC

hmac(k0, 9f..13)
Cookies with integrity

Standard technique:
Protect cookies with a MAC

hmac(k₀, 9f..13)
Hash-chaining

- Key enabling idea, dating back to Amoeba
- Macaroon: Key identifier + caveats + signature

\[ 9f.., \text{cat:grumpy} \]

\[ \text{hmac}(k_0, "9f..mpy") \]
Hash-chaining

- Key enabling idea, dating back to Amoeba
- Macaroon: Key identifier + caveats + signature
Hash-chaining

- Key enabling idea, dating back to Amoeba
- Macaroon: Key identifier + caveats + signature

9f.., cat:grumpy
expires: 2014-03-01

hmac(k0,"9f..mpy")
Hash-chaining

- Key enabling idea, dating back to Amoeba
- Macaroon: Key identifier + caveats + signature

$\text{hmac}(k_1, "exp\_2014-03-01")$
Verification

- Verify integrity by recomputing signature
Verification

- Verify integrity by recomputing signature
Verification

- Verify integrity by recomputing signature

\[
\text{hmac}(k_0, 9f.., \text{cat:grumpy}, \text{expires:2014-03-01}) = k_1
\]
Verification

- Verify integrity by recomputing signature

\[
\text{hmac}( k_0 , \text{9f.., cat:grumpy} ) = k_1 \\
\text{hmac}( k_1 , \text{expires:2014-03-01} ) = k_2 \\
k_2 = \text{fe3228a0bcee79}
\]
Verification

- Verify integrity by recomputing signature

9f.., cat:grumpy
expires:2014-03-01
fe3228a0bcee79 ✓
Verification

- Verify integrity by recomputing signature
- Check that all caveats are satisfied

| 9f.., cat:grumpy | ✓ |
| expires:2014-03-01 | ✓ |
| fe3228a0bcee79 | ✓ |
9f.., cat: grumpy
expires: 2014-03-01
fe3228a0bcee79
First-party caveats

- Can be added by any holder of a macaroon
- No constraints on the assertion language
- Naturally support attenuation and delegation
Delegation
Delegation

c3.., client:catfolio
hmac(k0,"c3..lio")

k0
Delegation

c3.., client:catfolio

hmac(k0,"c3..lio")
Delegation

c3.., client:catfolio
title:filename:grumpy.jpg
hmac(k1,”fil...jpg”)
Delegation

c3., client: catfolio
filename: grumpy.jpg
hmac(k1, "file...jpg")
Delegation

k0

c3.., client:catfolio
filename:grumpy.jpg
hmac(k1,"file...jpg")
Delegation

c3.., client: catfolio
filename: grumpy.jpg
hmac(k1, "file...jpg")
Third-party caveats

- So far, only the target server is trusted for authorization decisions.
- **Third-party caveats** require another service to check restrictions.
E(k2, kC) → E(kGoo, kC: catlovr)

kGoo

kC

k0

kId: client: catpics
filename: grumpy.jpg
E(kGoo, kC:catlovr)

E(k2, kC)

E(kGoo, kC:catlovr)

k0

kC

kId, client:catpics
filename:grumpy.jpg
E(kGoo, kC:catlovr)
Verification

- As before, check signature & caveats
- For each third-party caveat, recursively:
  - Verify its discharge macaroon
  - Check discharge macaroon caveats
Verification

- As before, check signature & caveats
- For each third-party caveat, recursively:
  - Verify its discharge macaroon
  - Check discharge macaroon caveats
Verification

- As before, check signature & caveats
- For each third-party caveat, recursively:
  - Verify its discharge macaroon
  - Check discharge macaroon caveats
Verification

- As before, check signature & caveats
- For each third-party caveat, recursively:
  - Verify its discharge macaroon
  - Check discharge macaroon caveats
Verification

- As before, check signature & caveats
- For each third-party caveat, recursively:
  - Verify its discharge macaroon
  - Check discharge macaroon caveats
Benefits

**Speed:** HMACs = Very light-weight, and fast.

**Timeliness:** Can require fresh credentials and revocation checks on every request.

**Flexibility:** Contextual confinements, attenuation, delegation, third-party caveats.

**Adoptability:** HMACs can run anywhere.
Example: Identity

Session, 2h

Discharge, 2s
Example: Identity

Logged out

Access denied
## Performance

<table>
<thead>
<tr>
<th></th>
<th>JS/Chrome</th>
<th>Node.js</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minting</td>
<td>27 μs</td>
<td>19 μs</td>
</tr>
<tr>
<td>Add caveat</td>
<td>~300 μs</td>
<td>56 μs</td>
</tr>
<tr>
<td>Verify</td>
<td>~360 μs</td>
<td>70 μs</td>
</tr>
<tr>
<td>To/from JSON</td>
<td>~3.5 μs</td>
<td>~4.0 μs</td>
</tr>
</tbody>
</table>
# Performance

<table>
<thead>
<tr>
<th>Task</th>
<th>JS/Chrome</th>
<th>Node.js</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minting</td>
<td>37k /s</td>
<td>53k /s</td>
</tr>
<tr>
<td>Add caveat</td>
<td>3.3k /s</td>
<td>18k /s</td>
</tr>
<tr>
<td>Verify</td>
<td>3.7k /s</td>
<td>14k /s</td>
</tr>
<tr>
<td>To/from JSON</td>
<td>380k /s</td>
<td>250k /s</td>
</tr>
</tbody>
</table>
Fun stuff in the paper

- Formal definitions and verification
- A variant based on public-key crypto instead of hash-chains
- Comparison to SPKI/SDSI
- Other uses for first- and third-party caveats
Macaroons are better cookies.
- Fast: Can be used anywhere, often.
- Flexible: Fit many applications, delegable.
- Secure:
  Caveats limit scope; Contextual caveats make stolen macaroons useless.