What are "Macaroons"?
for what might they be useful

06.08.2019

Nima Azimihashemi

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Es war einmal …

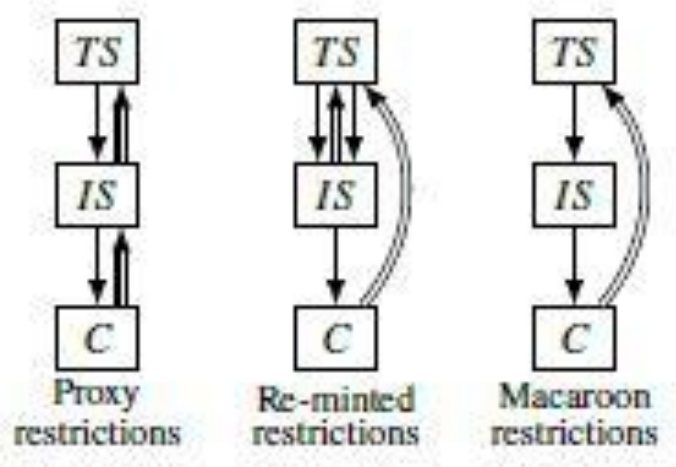
1976

Public key cryptography

Und 2014
Google Team backte das Macaron

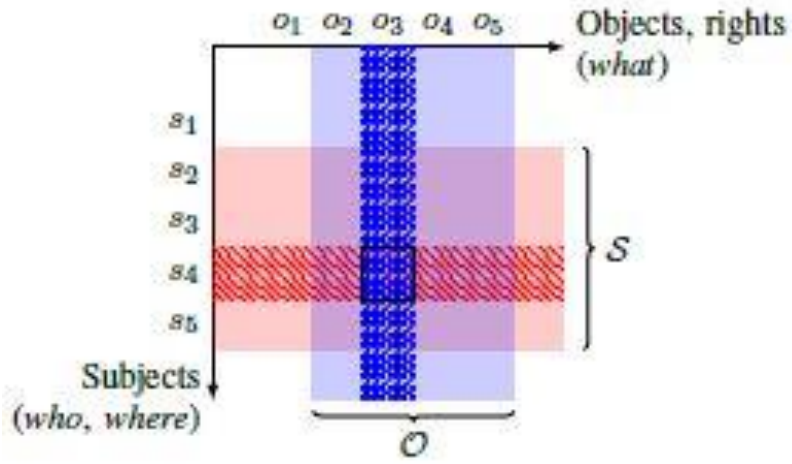
Eigenschaften

- Delegation with Contextual Caveats
  - SPKI / SDSI



Eigenschaften

- Attennuation (Verdünnung)
  - OAuth/OAuth2



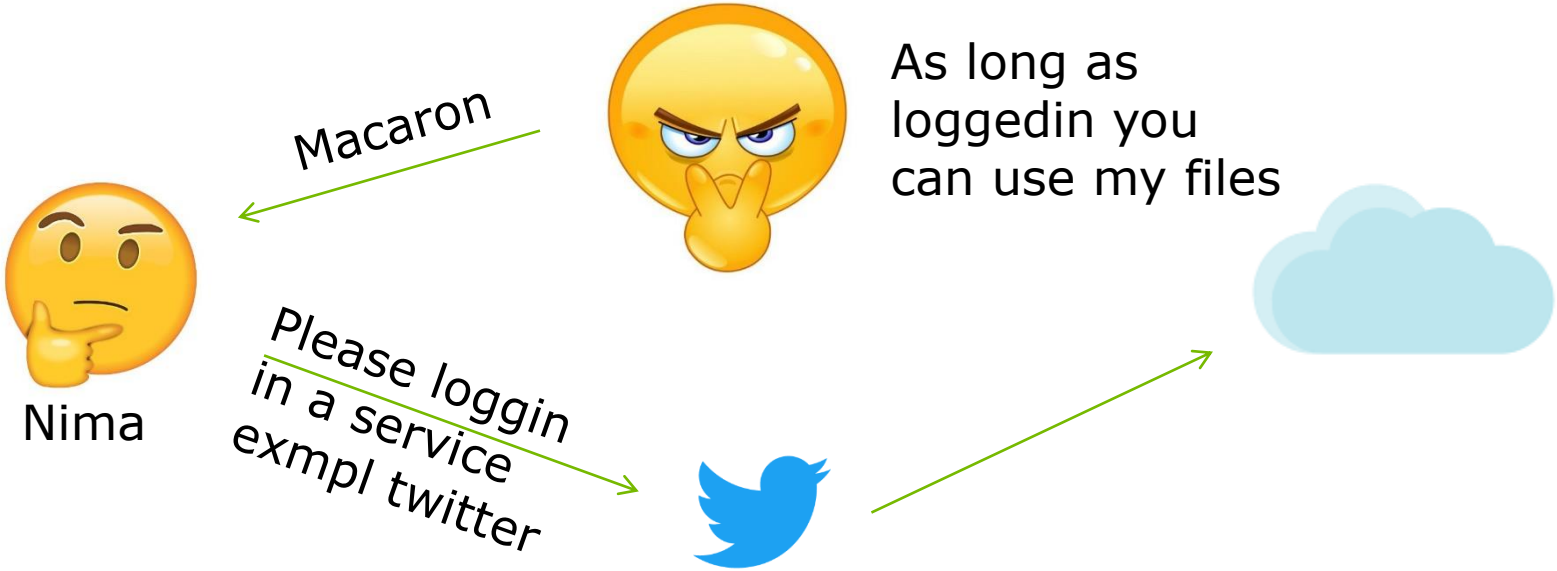
Eigenschaften

- Proof carrying
  - HMAC (Hashbased message authentication)

$$
\begin{aligned}
&\mathsf{CreateMacaroon}(k, id, L)\\
&\quad sig := \mathsf{MAC}(k, id)\\
&\quad \textbf{return } \mathsf{macaroon}_{@L}\langle id, [\,], sig\rangle
\end{aligned}
$$

Eigenschaften

- 3. party Caveat



Pseudcode

```
CreateMacaroon(k, id, L)
  sig := MAC(k, id)
  return macaroon_@L⟨id, [ ], sig⟩

M.addCaveatHelper(cId, vId, cL)
  macaroon_@L⟨id, C, sig⟩ ← M      // ← is pattern matching
  C := cav_@cL⟨cId, vId⟩
  sig' := MAC(sig, vId :: cId)     // :: is pair concatenation
  return macaroon_@L⟨id, C ▷ C, sig'⟩      // ▷ is list append

M.AddThirdPartyCaveat(cK, cId, cL)
  vId := Enc(M.sig, cK)
  return M.addCaveatHelper(cId, vId, cL)

M.AddFirstPartyCaveat(a)
  return M.addCaveatHelper(a, 0, ⊤)

M.PrepareForRequest(𝓜)
  𝓜^scaled := ∅
  for M' ∈ 𝓜
    macaroon_@L'⟨id', C', sig'⟩ ← M'
    sig'' := M.bindForRequest(sig')
    𝓜^scaled := 𝓜^scaled ∪ {macaroon_@L'⟨id', C', sig''⟩}
  return 𝓜^scaled

M.Verify(TM, k, 𝒜, 𝓜)
  cSig := MAC(k, M.id)
  for i := 0 to |M.C| − 1
    cav_@L⟨cId, vId⟩ ← M.C[i]
    cK := Dec(cSig, vId)
    if (vId = 0)
      assert (∃a ∈ 𝒜 : a = cId)
    else
      assert (∃M' ∈ 𝓜 : M'.id = cId ∧ M'.Verify(TM, cK, 𝒜, 𝓜))
    cSig := MAC(cSig, vId :: cId)
  assert (M.sig = TM.bindForRequest(cSig))
  return true
```

Fig. 8. The operations essential to authorization using macaroon credentials.

Beispiel:

```ruby
require 'macaroons'
require 'securerandom'

# Deserialize the macaroon

macaroon = Macaroon.from_binary(ENV[HTTP_AUTHENTICATION])

verifier = Macaroon::Verifier.new

verifier.satisfy_exact("username = k4nd4lf")
verifier.satisfy_exact("timestamp = 1464568512")

puts verifier.verify(
    macaroon: macaroon,
    key: Settings::SECRET #this is your secret key, the one you used to sign
)

=> true
```

```ruby
require 'macaroons'
require 'securerandom'

secret     = Settings::SECRET
identifier = "users"
location   = "http://my.protected.service.com"

macaroon = Macaroon.new(key: secret, identifier: identifier, location: locat

macaroon.add_first_party_caveat("username = k4nd4lf")
macaroon.add_first_party_caveat("timestamp = #{(Time.now + 10).to_i}")

puts macaroon.serialize
```

Referenzen

- Official Google AI Publication Page: https://ai.google/research/pubs/pub41892
- Github Implement.: https://github.com/rescrv/libmacaroons

- Cryptography: https://cs.stanford.edu/people/eroberts/courses/soco/projects/public-key-cryptography/history.html
- HMAC: https://dev.to/mathijspim/hmac-authentication-better-protection-for-your-api-4e0
- SPKI/SDSI: http://world.std.com/~cme/html/spki.html
- OAuth2: https://blog.oio.de/2018/08/20/eine-einfuhrung-in-oauth-2/

- Articles from other authors about Macaroons:
  https://blog.bren2010.io/2014/12/04/macaroons.html
  http://tech.tmh.io/concept/2016/06/07/macaroons-a-new-flavor-for-authorization.html
  https://www.rubydoc.info/gems/macaroons/0.6.1
  http://hackingdistributed.com/2014/05/21/my-first-macaroon/
  http://hackingdistributed.com/2014/05/16/macaroons-are-better-than-cookies/

- HTW Musterfolie: https://corporatedesign.htw-berlin.de/musterdokumente/powerpoint-vorlagen/
- How to bake a Macaron: https://en.wikipedia.org/wiki/Macaron ;)

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

www.htw-berlin.de