Introduction to Macaroons

illuminated

Dr. Neil Madden

Author *API Security in Action*

Found “Psychic Signatures” vulnerability in Java ECDSA

AppSec and applied crypto specialist

Contributor to OAuth and JOSE working groups

illuminated

Macaroons

Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud

Arnar Birgisson · Joe Gibbs Politz · Úlfar Erlingsson · Ankur Taly · Michael Vrable · Mark Lentczner ·

Network and Distributed System Security Symposium, Internet Society (2014)

Cookies & authentication tokens

illuminated

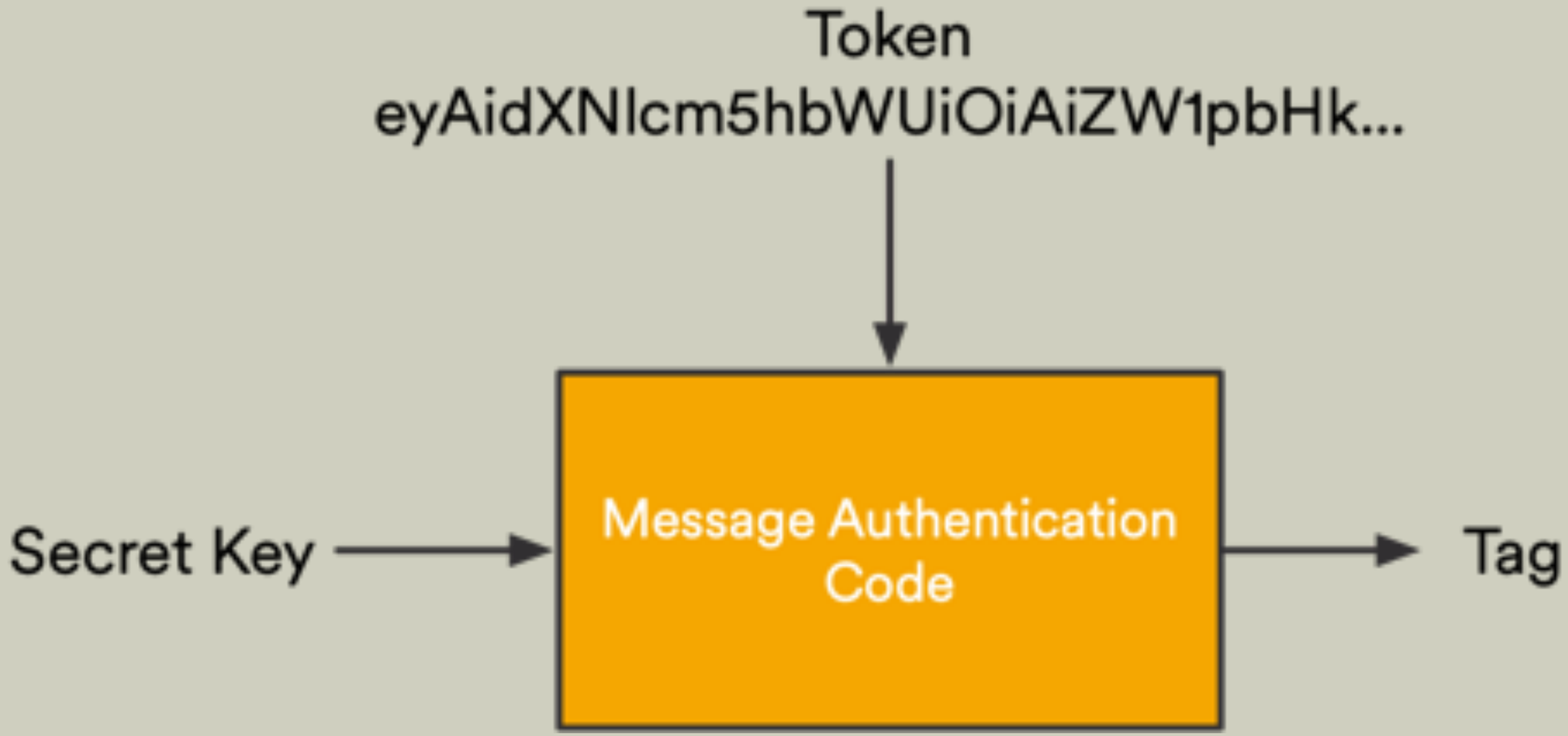
Database token storage

Token	Username	Login Time	Login Method
xyz...	emily.p	1234567	Password
abc...	peter.f	4561234	MFA
...	...	...	...

```
var json = {
    "username": "emily.p",
    "logintime": 1234567,
    "authmethod": "password",
    "fullname": "Emily Peacock",
    ...
}
var cookie = base64url(json);
```

```
eyAidXNlcm5hbWUiOiAiZW1pbHk...
```

Message Authentication Codes



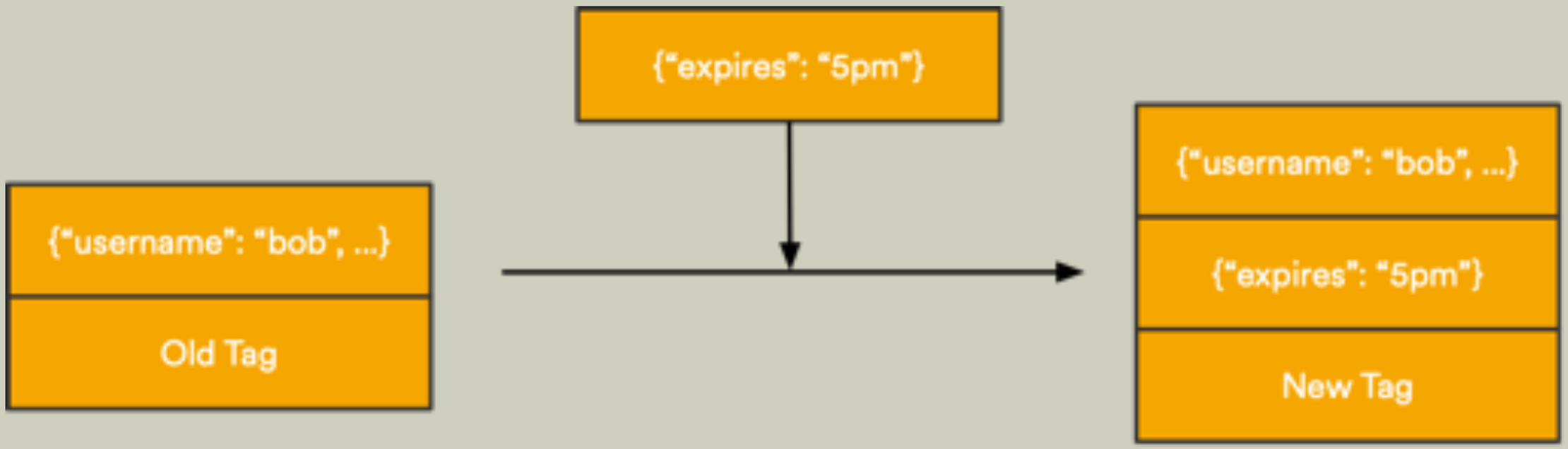
illuminated

Macaroons

```
var claims = {
    "username": ...
    ...
}
var encoded = base64url(claims);
var tag = hmac(key, encoded);
var macaroon = encoded + ':' +
base64url(tag);
```

illuminated

Caveats


{“expires”: “5pm”}
{“username”: “bob”, ...}
Old Tag
{“username”: “bob”, ...}
{“expires”: “5pm”}
New Tag

Caveats should only ever *restrict*

Bearer tokens

and API clients

Proof of Possession

Cryptography to the rescue?

© Mybloodtypeiscoffee, CC BY-SA 4.0, via Wikimedia Commons

illuminated

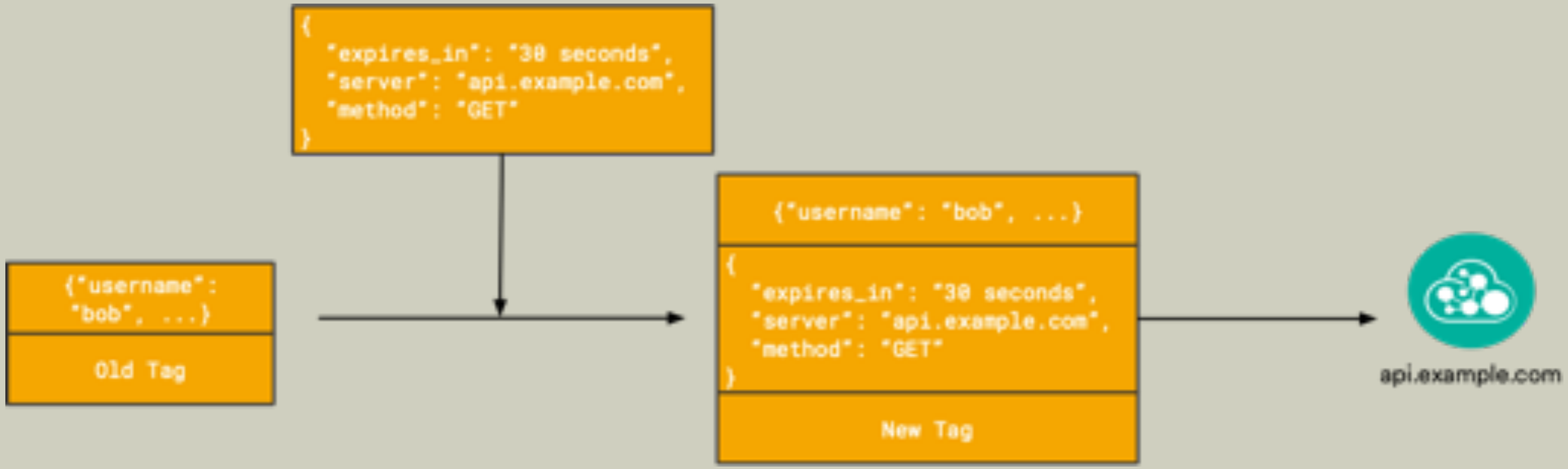
> “Cryptography is a tool for turning lots of different problems into key management problems.”
>
> — Lea Kissner

Contextual caveats

Least privilege with less effort

illuminated

Contextual caveats



Contextual caveats are added to a copy of the token, but the client still retains access to the original, unrestricted, macaroon.

How caveats work

illuminated

Pseudorandom Functions

$$\mathrm{PRF} : K \times M \to T$$

where $K = \{0,1\}^k, M = \{0,1\}^*, T = \{0,1\}^t$.

If you don’t know the key, k, then $\mathrm{PRF}(k, \cdot)$ looks indistinguishable from a random function $f : M \to T$.

Macaroon PRF

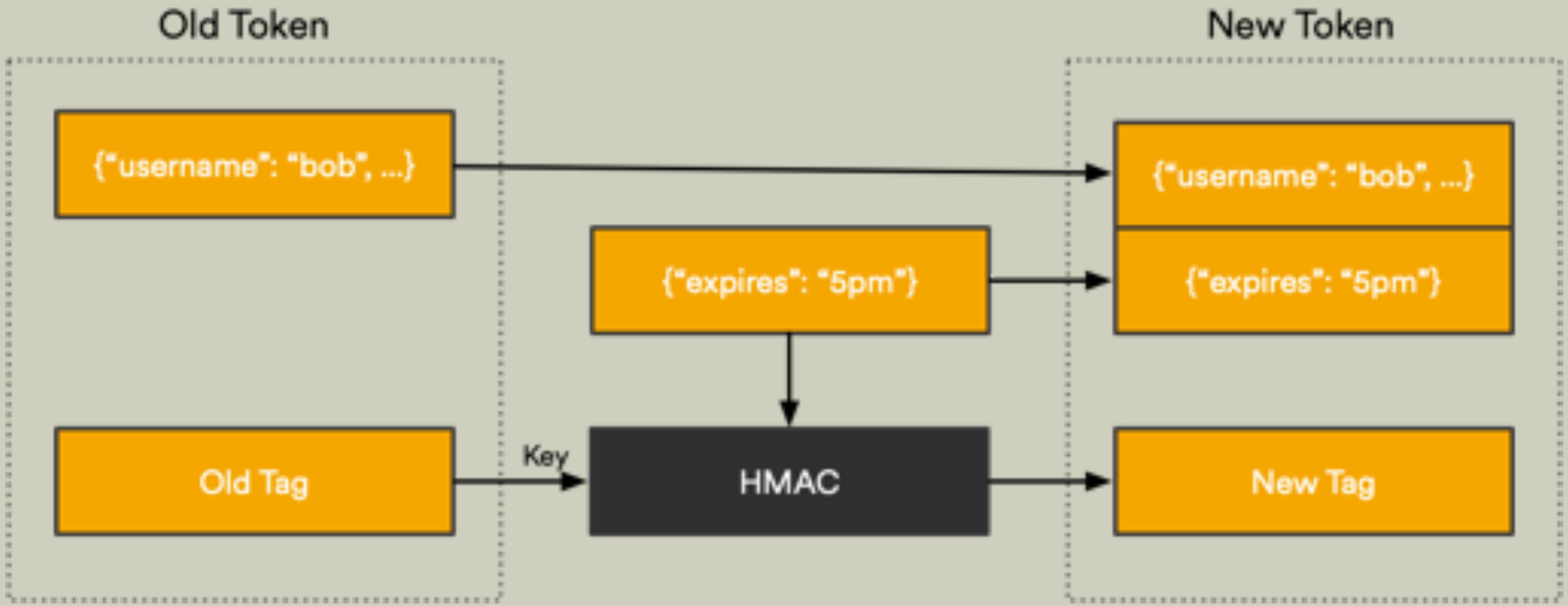
For HMAC:

$$T = K$$

$$\text{HMAC} : K \times M \to K$$

Use old tag as the key to compute a new tag.

Appending a caveat



Just-in-time contextual caveats

```
function restrict(macaroon, newCaveat) {
  var old = decode(macaroon);
  var newTag = hmac(old.tag, newCaveat);
  return encode(old.claims,
    old.caveats + newCaveat, newTag);
}
```

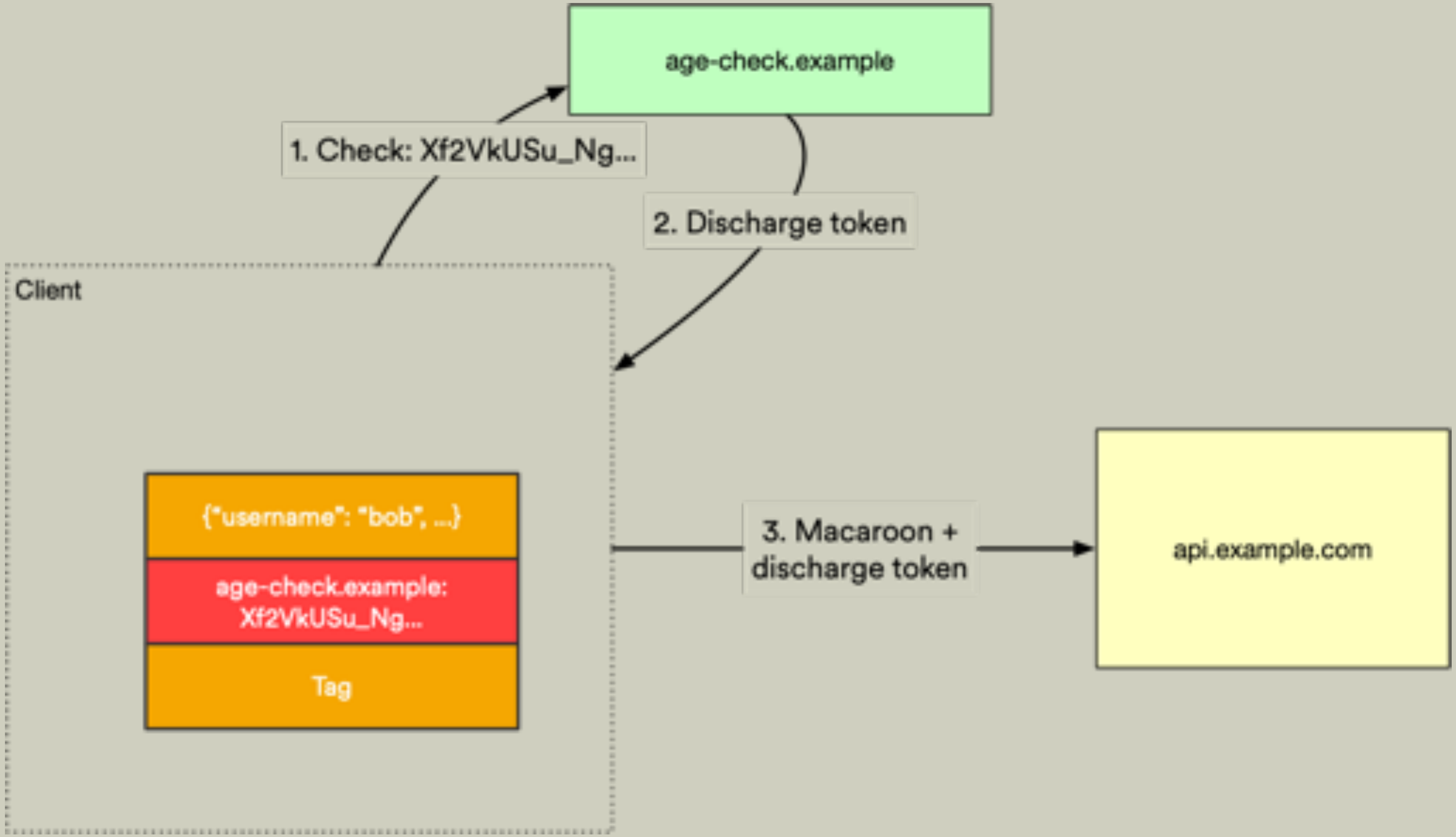
```
fetch('https://api.example', {
    headers: {
      'Authorization': 'Bearer ' +
        restrict(macaroon, {exp: now()+30})
    }
});
```

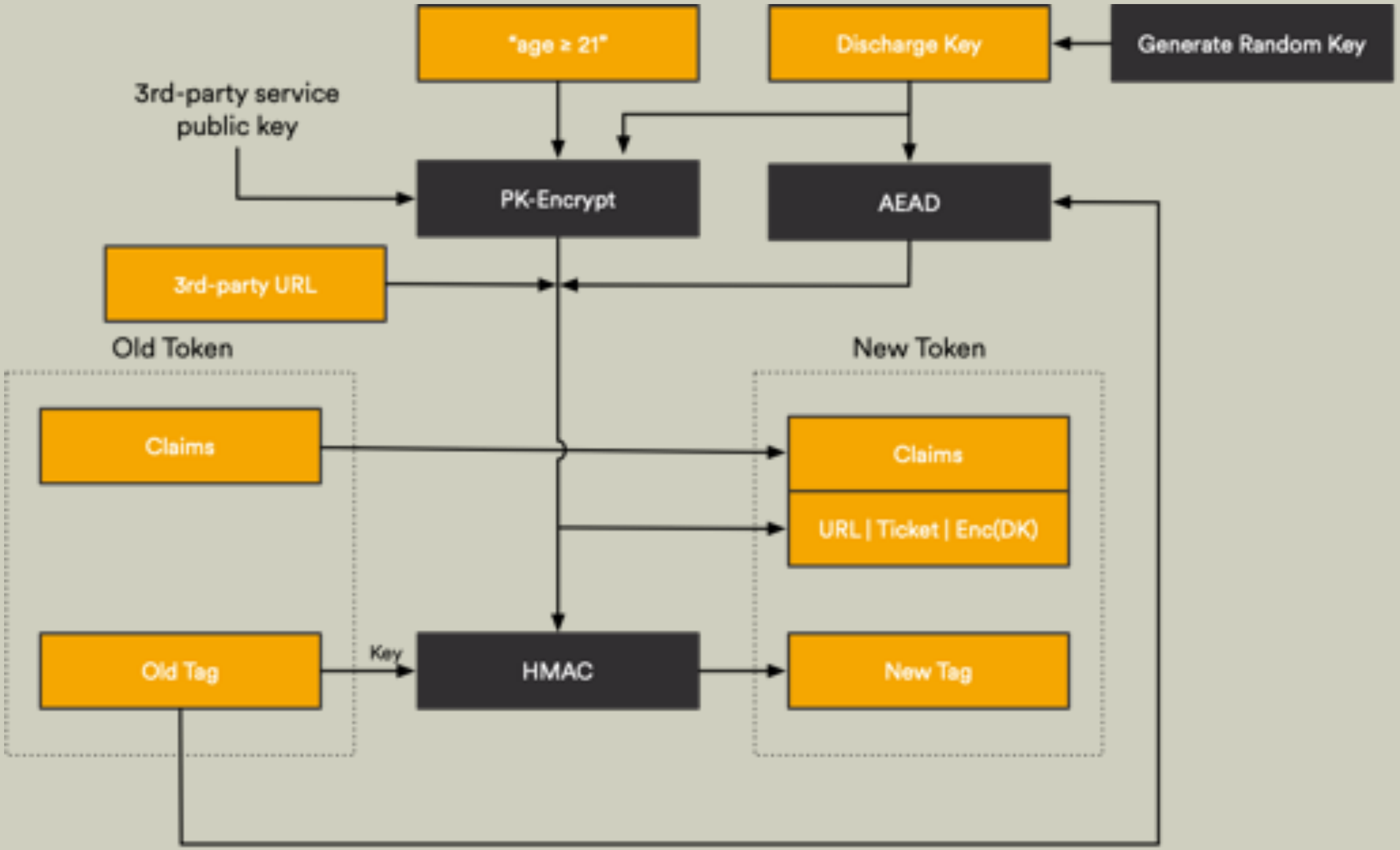
Verifying a Macaroon

```
var { claims, caveats, providedTag } =
    decode(macaroon);

var computedTag = hmac(key, claims);
for (var caveat in caveats) {
    computedTag = hmac(computedTag, caveat);
}

var sigOk = macEquals(providedTag,
computedTag);
return sigOk && checkAll(caveats, request);
```

3rd-party caveats

illuminated




"age ≥ 21"
Discharge Key
Generate Random Key
3rd-party service public key
PK-Encrypt
AEAD
3rd-party URL
Old Token
New Token
Claims
Claims
URL | Ticket | Enc(DK)
Old Tag
Key
HMAC
New Tag

What can you do with 3rd-party caveats?

- Require login via SSO
- Add revocation checks
- Implement anti-CSRF tokens
- Authorize payments & other transactions

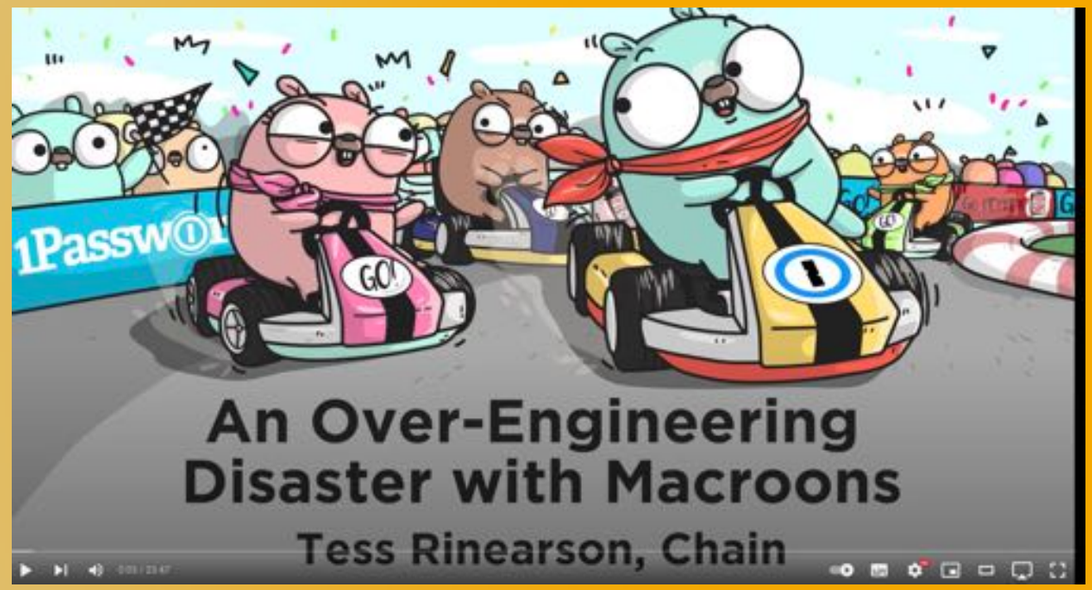
A word of caution

Easy to over-use 3rd-party caveats

Can make the system fragile

Case study: Tess Rinearson, Chain, GopherCon 2018[1]

Self-DoS via 3rd-party caveat that was discharged by an underpowered Rails app

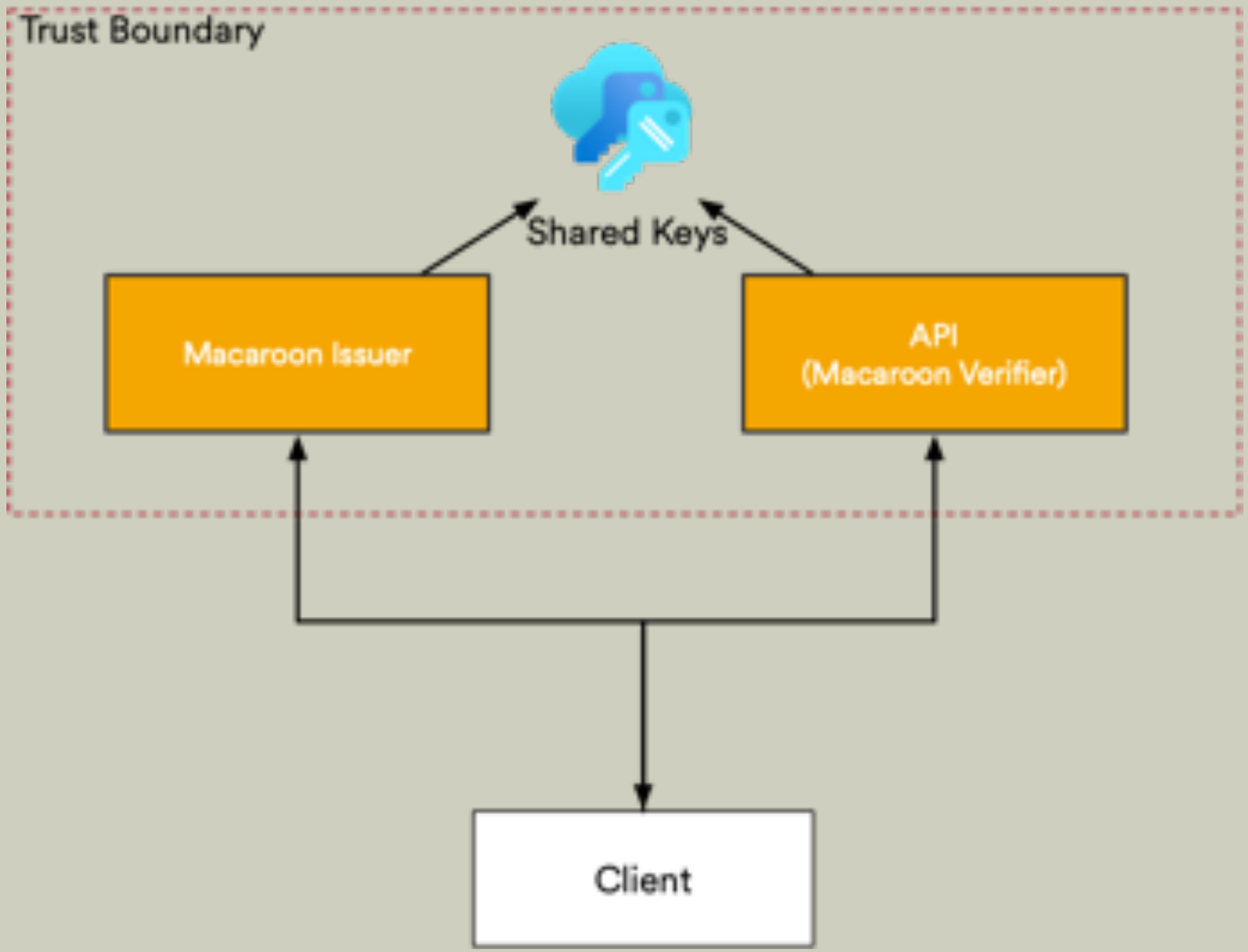


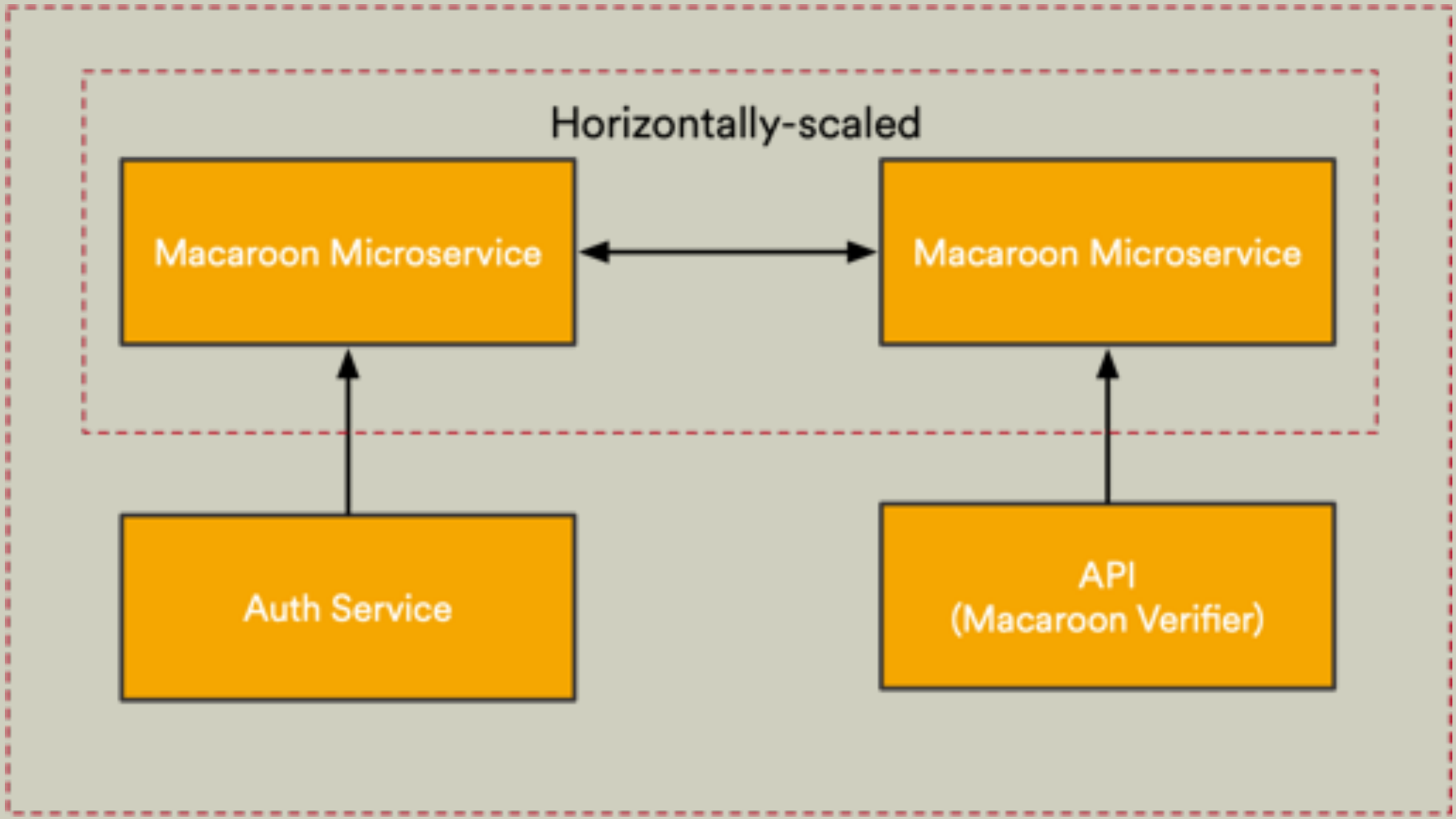
1. https://www.youtube.com/watch?v=MZFv62qz8RU

Deployment topics

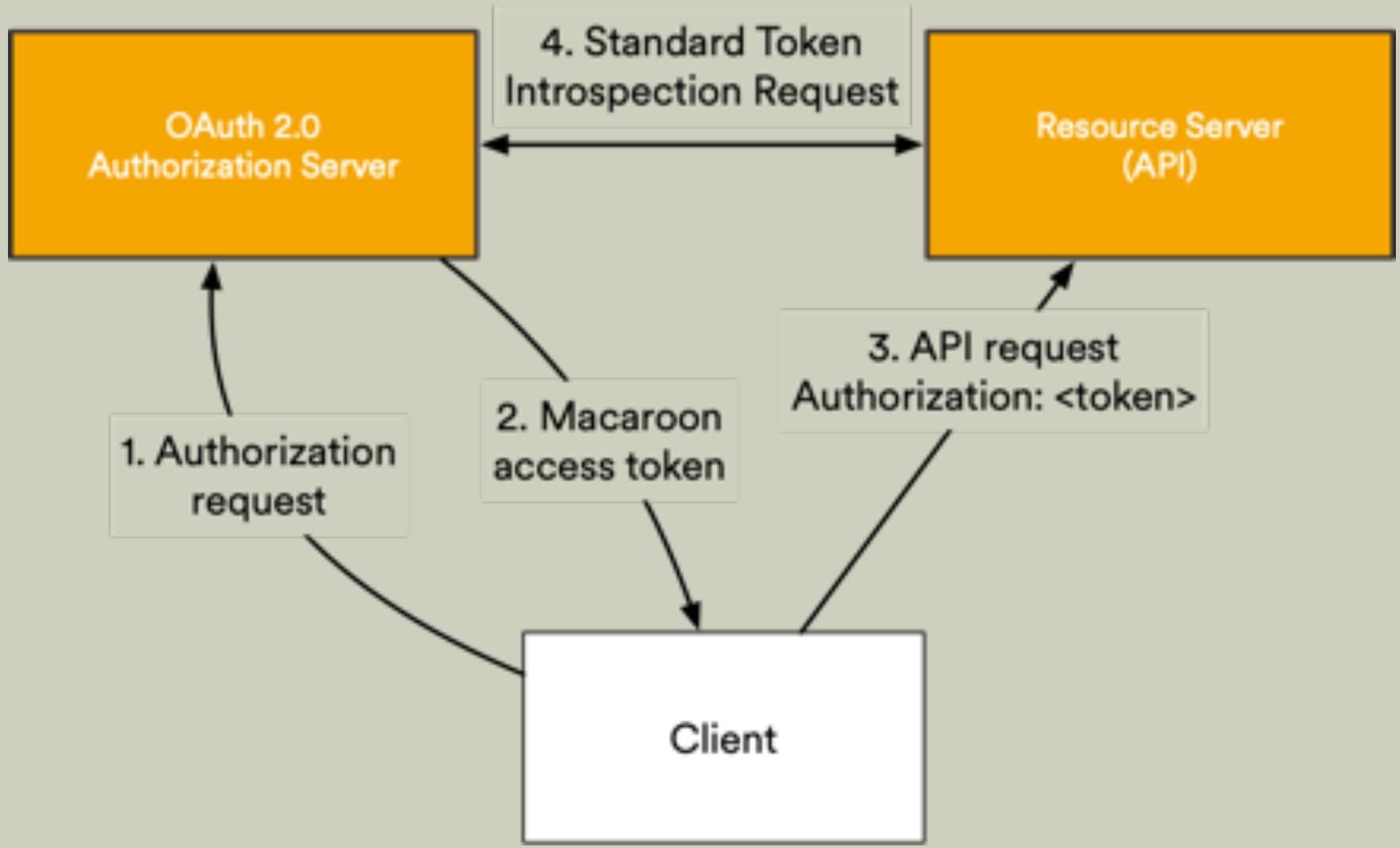
illuminated

Trust model


Trust Boundary
Shared Keys
Macaroon Issuer
API
(Macaroon Verifier)
Client



Macaroons and OAuth



illuminated

Macaroons and OAuth

Supports standard OAuth/JWT fields as caveats:

```
scope, aud, exp, nbf, cnf
```

Token introspection respects caveats—example: expiry time is minimum of original expiry and any `exp` caveats

Handles discharge macaroons too

Scan for more details

illuminated

Public Key Macaroons?

florentines

illuminated

Why *shouldn't* you use Macaroons?

No standard

De facto “standard” libmacaroons quirky, not widely liked

Not “boring”: burn innovation tokens?

Further reading:

https://research.google/pubs/macaroons-cookies-with-contextual-caveats-for-decentralized-authorization-in-the-cloud/

https://fly.io/blog/macaroons-escalated-quickly/

https://github.com/superfly/macaroon/blob/main/macaroon-thought.md

https://neilmadden.blog/2020/09/09/macaroon-access-tokens-for-oauth-part-2-transactional-auth/

illuminated