Identity Issues

TPAC 2012

Eric Rescorla

ekr@rtfm.com

Big topic on the list: Tying Identity to `getUserMedia()`

- Raised by Martin Thomson
- Good discussion by Cullen, Harald, Jim, Li, Oscar, Tim, Stefan, Jim

Background

- Identity is intended to allows authentication that isn't mediated by the calling site
- For high security, we want the site to be able to neither see nor touch media

  The API MUST provide a mechanism for the requesting JS to relinquish the ability to see or modify the media (e.g., via MediaStream.record()). Combined with secure authentication of the communicating peer, this allows a user to be sure that the calling site is not accessing or modifying their conversion.

  — `draft-ietf-rtcweb-security-arch-05`

General Idea: Grant Media Access to an Identity

- In basic gUM, user gives permission to *site* to access devices
- What we want is to give permission to *identity* to access devices
  - Site just manipulates handles to the streams
  - But can't access them

Topics

- How do we isolate the stream? And what can you do with an isolated stream?
- How does the user/browser learn the identity?
- How and when does the user give consent?
- When does the light go on and off?
- Do we need some (modest) new primitives for operating on isolated streams?

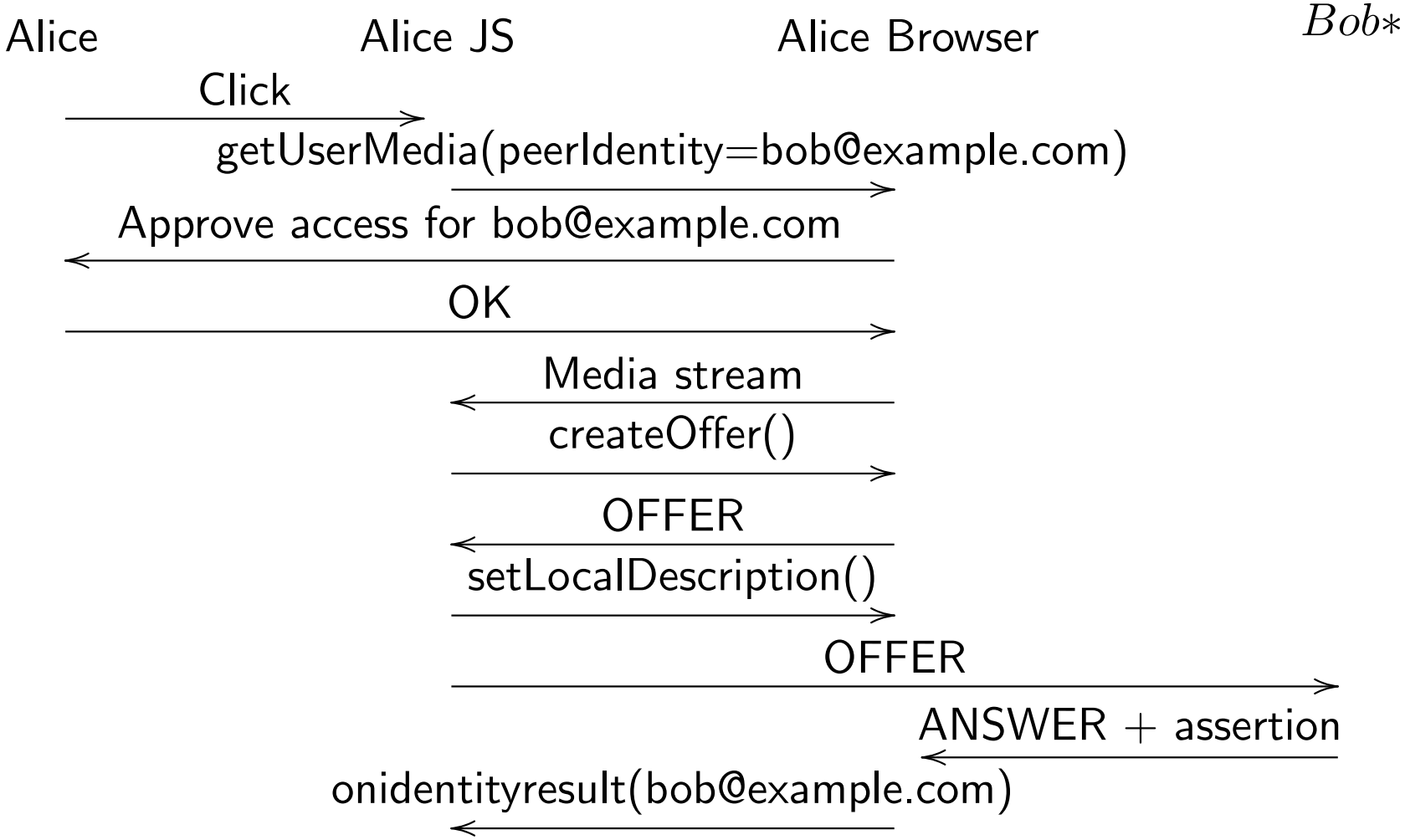
Stream Isolation

- How do you isolate a stream?
  - Tag it as being in a separate origin
  - Normal cross-origin protections apply
  - This is how cross-origin video streams work *now*
- What can you do with it?
  - Attach to a video/audio element
    - But that element can't be modified/read from content
  - `pc.addStream()`*

*Subject to permissions. See later slides

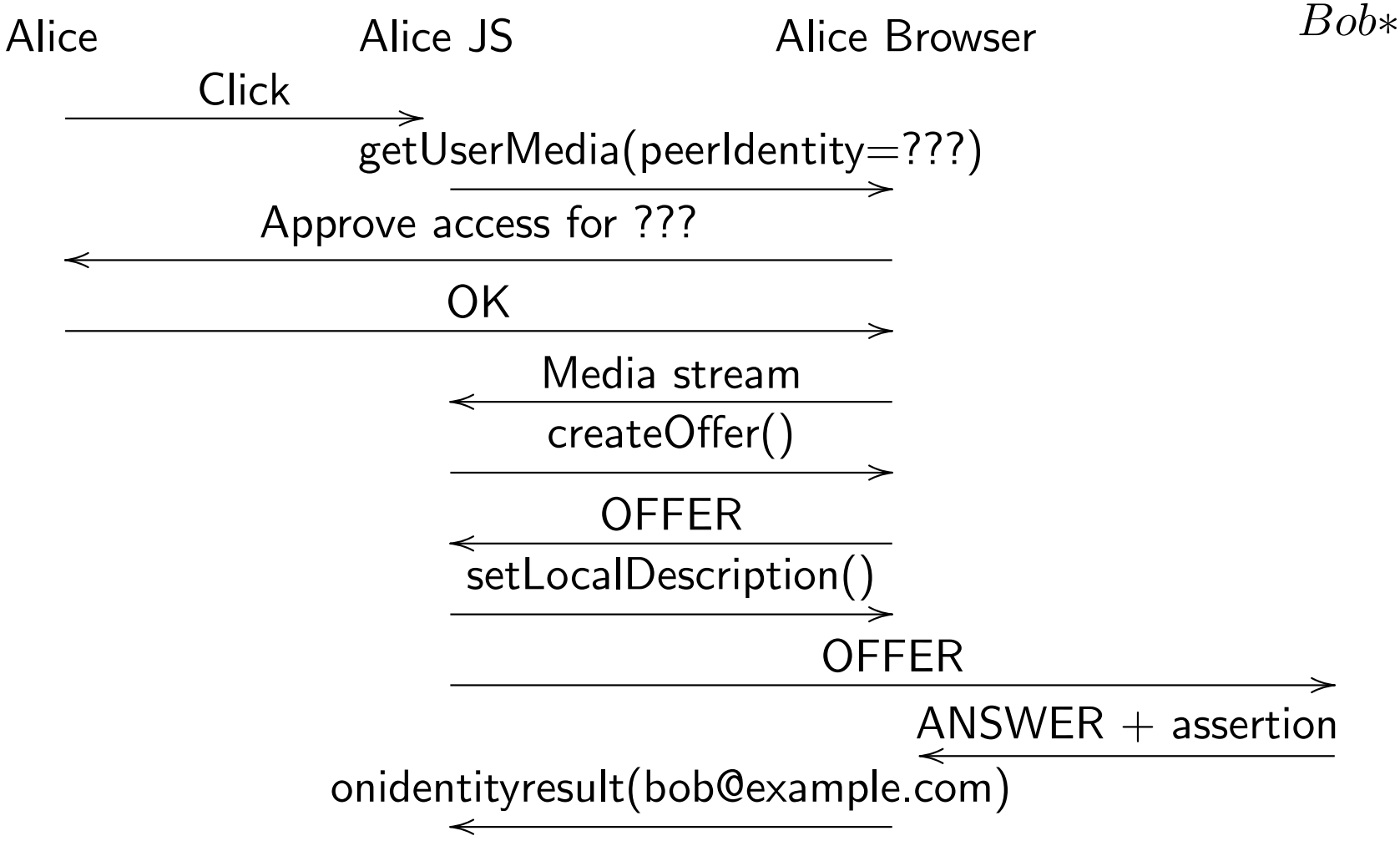
Proposal 1: Identity Provided to gUM (Thomson)

- Call
  `getUserMedia({peerIdentity:"smithee@example.com"},...)`
  - Browser prompts user with "do you want to share your camera with 'foo@example.com"'
  - (Or "Allan Smithee" if he is in your address book)
    * Not the site's address book
- PeerConnection enforces:
  - Keying must be via DTLS
  - MediaStream can only be connected to a PC with identity `smithee@example.com`


Alice
Alice JS
Alice Browser
$Bob*$
Click
getUserMedia(peerIdentity=bob@example.com)
Approve access for bob@example.com
OK
Media stream
createOffer()
OFFER
setLocalDescription()
OFFER
ANSWER + assertion
onidentityresult(bob@example.com)

Are you sure you know the other side's identity?

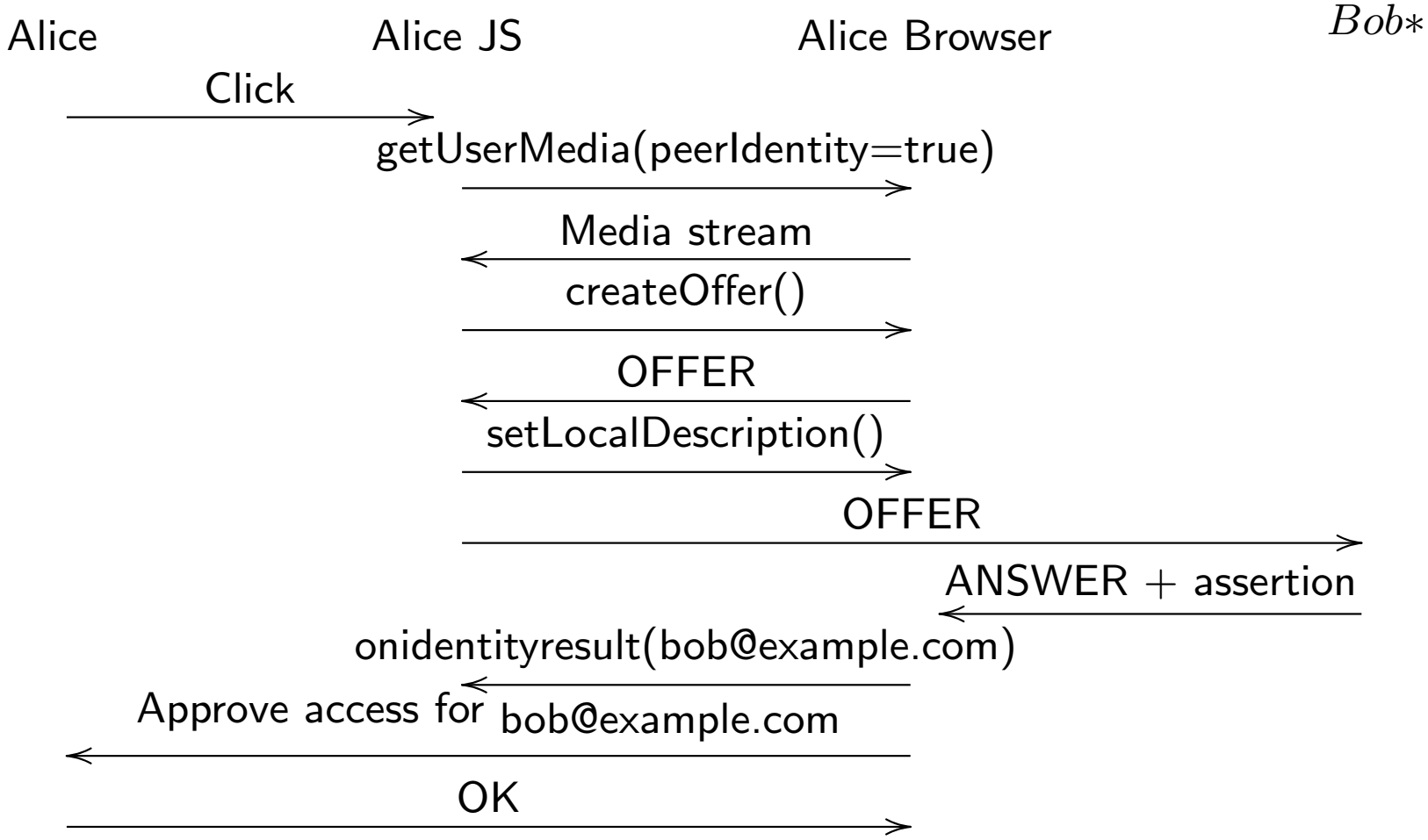
- Easy for the callee
- But for the caller not so much...
  - Forking
  - Calling an IVR with multiple operators
  - Other side has multiple IdPs (Facebook, G+...)


Alice
Alice JS
Alice Browser
$Bob*$
Click
getUserMedia(peerIdentity=???)
Approve access for ???
OK
Media stream
createOffer()
OFFER
setLocalDescription()
OFFER
ANSWER + assertion
onidentityresult(bob@example.com)

- What now?

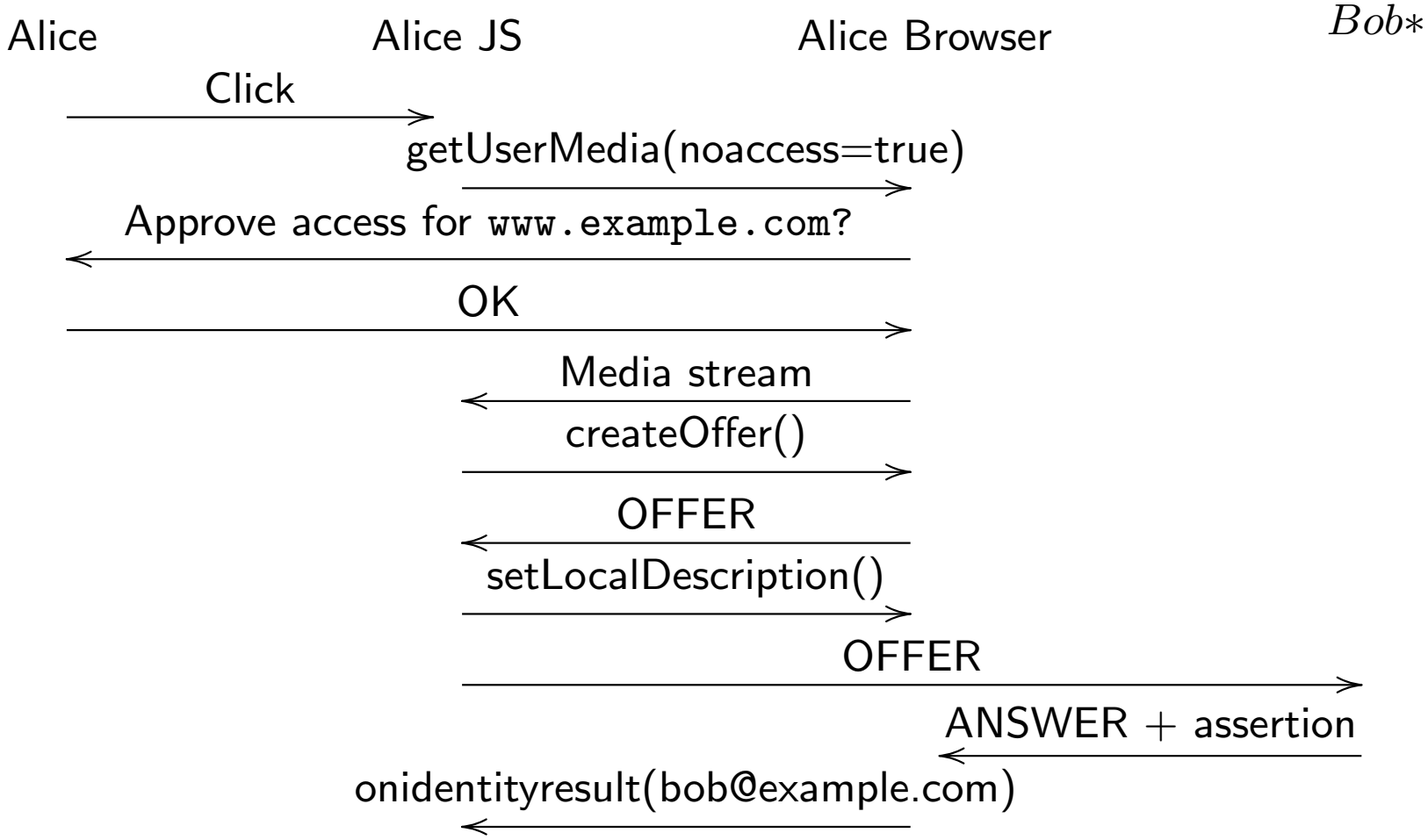
Proposal 2: Prompt user after call (Ohlsson)

- Call `getUserMedia({peerIdentity=true})`
  - Media stream acquired silently (or maybe site-specific dialog)
  - But tagged and access controlled as in Proposal 1
- Once PeerConnection is established, browser knows peer identity
  - User prompted with gUM permissions dialog based on peer identity


Alice
Alice JS
Alice Browser
$Bob*$
Click
getUserMedia(peerIdentity=true)
Media stream
createOffer()
OFFER
setLocalDescription()
OFFER
ANSWER + assertion
onidentityresult(bob@example.com)
Approve access for bob@example.com
OK

Proposal 3: Site permissions with identity display (EKR?)

- Call `getUserMedia({noaccess=true})`
  - Media permissions as usual
- Once PeerConnection is established, browser knows peer identity
  - Browser displays identity and a "no access" indicator in the UI


Alice
Alice JS
Alice Browser
$Bob*$
Click
getUserMedia(noaccess=true)
Approve access for www.example.com?
OK
Media stream
createOffer()
OFFER
setLocalDescription()
OFFER
ANSWER + assertion
onidentityresult(bob@example.com)

Checking of identity ("phishing"?)

- Proposals 1 and 2 require user to explicitly assent to identity
  - Proposal 1 does matching in the browser
  - Proposal 2 does matching in the JS (if anywhere)
- Proposal 3 does not require user to explicitly assent
  - Instead provides an indicator he can check
- Does forcing the user to check do anything useful?

Long-term consent

- I may not want to grant long-term consent to any site
- But may be willing to grant it to a peer identity or set of identities
- Any of these models can probably be made to work
- But easier with Proposal 1 since we know ACL status at gUM time

When does the green light go on?

- Preferably when permission is granted by the user
- ... and when self video can be displayed
- This makes Proposal 2 look less attractive

Proposal 4: Hybrid 1 and 3

- Call `getUserMedia(peerIdentity=bob@example.com)`
  - This acts like option 1
  - gUM permissions check done based on identity/origin pair
  - Media inaccessible to site
- Call `getUseMedia(peerIdentity=null)`
  - This acts like option 3
  - gUM permissions check done based on origin
  - Media inaccessible to site
- Call `getUserMedia()`
  - Current behavior

Statistics and Level meters

- Making a stream inaccessible means restricting access
- What about meta-information
  - Level meters?
  - Statistics?
- Proposed approach
  - Figure out what's trivially safe and allow it
  - If things look unsafe and not critical, hold off on them
    * Maybe provide in-chrome gadgetry later

What about that green light (and other in-use indicators)?

- Overall objective: user knows when his devices are being accessed or *accessible*

- Indicator should be on if site *could* access devices whether it is or not
- Example: what if we turn off light when stream muted
  - Site asks for permission; light goes on; and does a call
  - When call ends, mutes stream; light goes off
  - Waits until user is idle (and hopefully away) site unmutes; light goes on
    - But user doesn't notice

Proposed rule

- Light goes on when user gives permission
- Light stays on until MediaStream closed or otherwise unavailable
- Light goes off
- Any new access requires a new permission check
- For short-term permissions a permissions check means user interaction