Transferring Digital Credentials with HTTP

draft-rescorla-tigress-http
https://github.com/ekr/draft-rescorla-tigress-http/tree/main

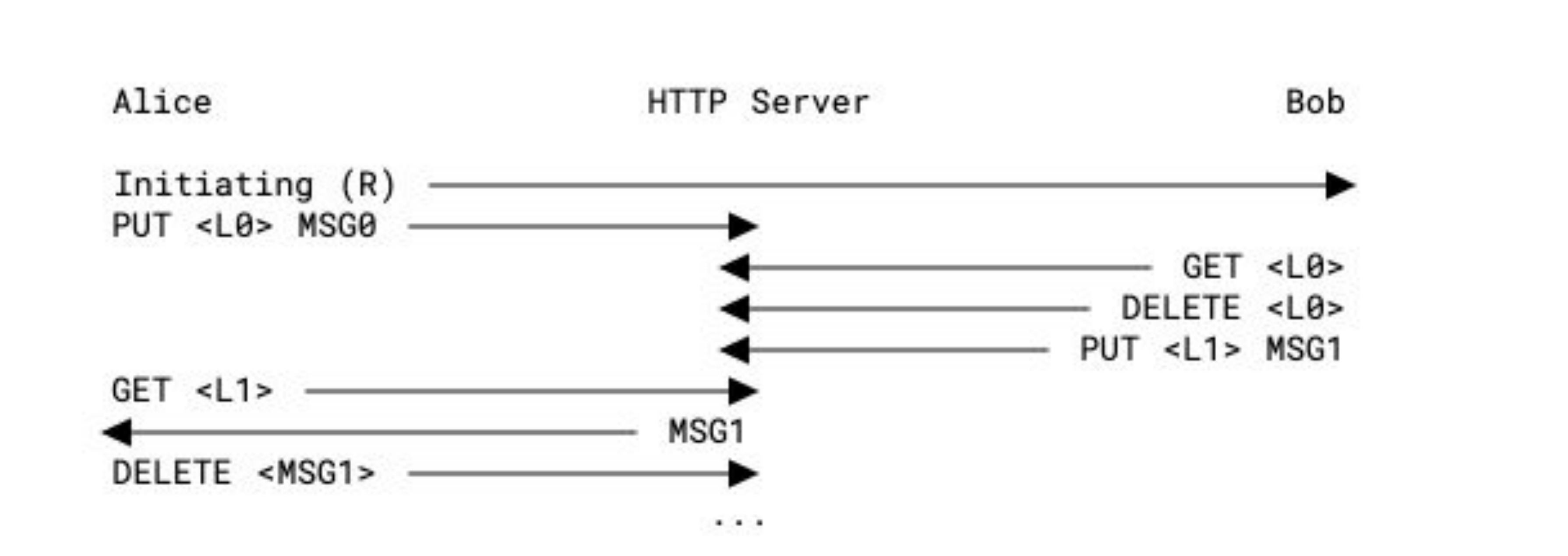
Eric Rescorla
Brad Lassey

IETF 118

Overall Idea

- Use a standard HTTP server as a dropbox
- Standard HTTP verbs (PUT, GET, DELETE)
- Each message has its own secret URL which controls access

Setup phase



Alice

HTTP Server

Bob

Initiating (R)

PUT <L0> MSG0

GET <L0>

DELETE <L0>

PUT <L1> MSG1

GET <L1>

MSG1

DELETE <MSG1>

...

HTTP Usage

- These are all standard HTTP verbs (PUT/GET/DELETE)
- Only requirement from the server is that it not allow enumeration of resources
    - This is a standard configuration

Secret URLs

- Initial message from the sender has a random value R
  - URL 0 is derived from R
- Subsequent messages contain random nonces
  - URL N+1 is derived the transcript of messages 0..N
- This means you need to see all previous messages to see a message’s address
- Messages are also encrypted with keys derived the same way

Notifications

- This doesn’t provide notifications of message delivery
  - You can still poll
- This is a straightforward extension but needs more than just HTTP
- Should be able to do something like WebPush
- This is orthogonal in any case

Security Properties

- An attacker can’t read the first message without knowing the shared secret in the initial invitation
- Once the first message is deleted by the receiver, the attacker can’t see the address for the second message
  - Even if it knows the shared secret
  - And hence can neither receive nor send messages
- There is a race condition here
  - This is inevitable unless there is a separate secret that isn’t known by the attacker

First-to-read Protection with Cookies

- A belt and suspenders mechanism for only allowing two parties to access
  - That doesn’t depend on deletion
- Can be satisfied by server restricting access to any given secret URL set to two session cookies.
- This goes beyond standard HTTP and should probably be optional in the spec.

Questions?