VICEROY: GDPR-/CCPA-compliant Verifiable Accountless Consumer Requests

Scott Jordan[1], Yoshimichi Nakatsuka[1],
Ercan Ozturk[1], Andrew Paverd[2], Gene Tsudik[1]

[1] University of California, Irvine
[2] Microsoft

Viceroy butterfly
https://unsplash.com/@jcotten

Data Protection Regulations

- GDPR (General Data Protection Regulation)
  - *data subjects* in the EU/EEA
- CCPA (California Consumer Privacy Act)
  - *consumers* who are California residents
- …
- Grant consumers legal rights over their data:
  - Access
  - Correct
  - Delete

Verifiable Consumer Request (VCR)

- Request from a **consumer** to a **service provider** (e.g., website) to access/modify/delete personal data
- Website must **verify** authenticity of request
  - Otherwise, there are privacy consequences
- Verification is straightforward when consumer has an account
  - Ask the consumer to log in etc.
- But what about consumers without accounts?
  - Data protection regulations still apply

How are “Accountless” consumers currently verified?

Government-issued ID

Signed statement

Credit card number

Phone interview

Ad-hoc, Insecure, Privacy-invasive

Introducing VICEROY

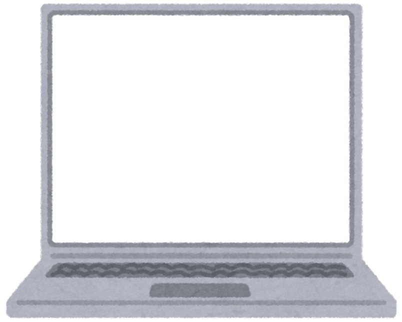
A framework enabling **accountless** consumers to request their data in a **secure** and **privacy preserving** manner.

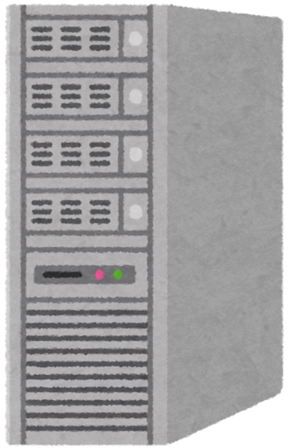
Specifically, VICEROY…

- allows consumers to generate VCRs without relying on symmetric tokens,
- allows website operators to efficiently and securely verify VCRs,
- can be integrated into existing websites with minimal changes.

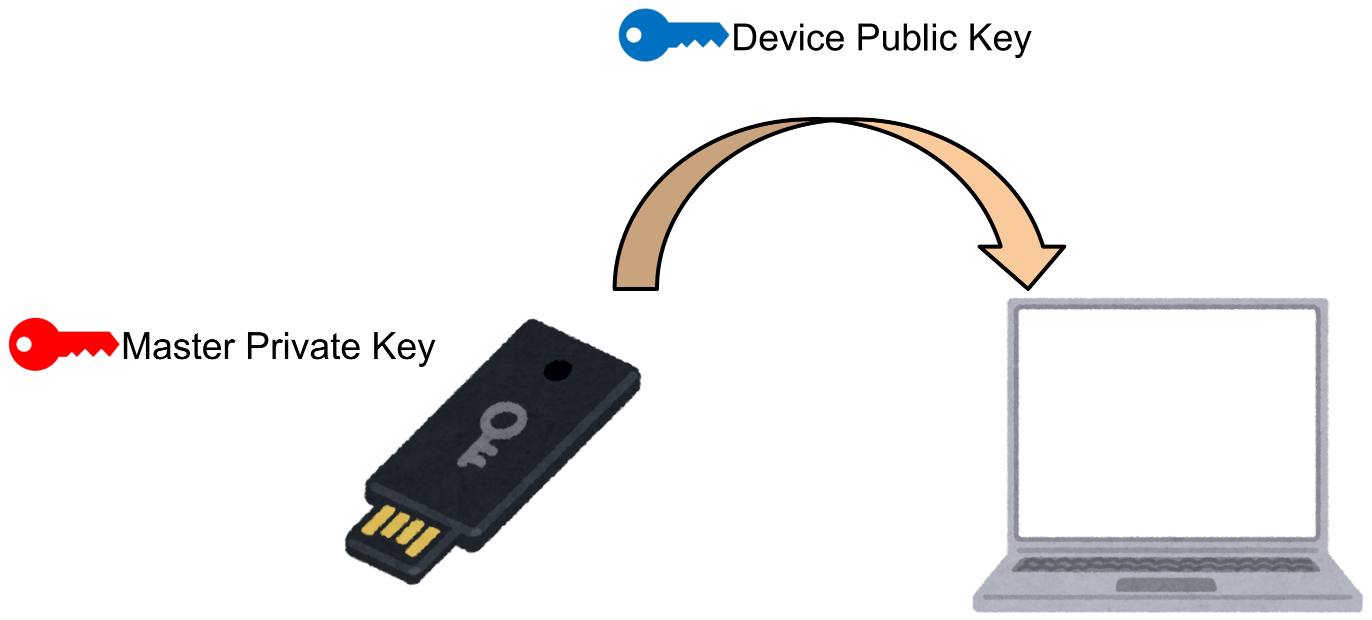
Overview of VICEROY

Trusted Client Device

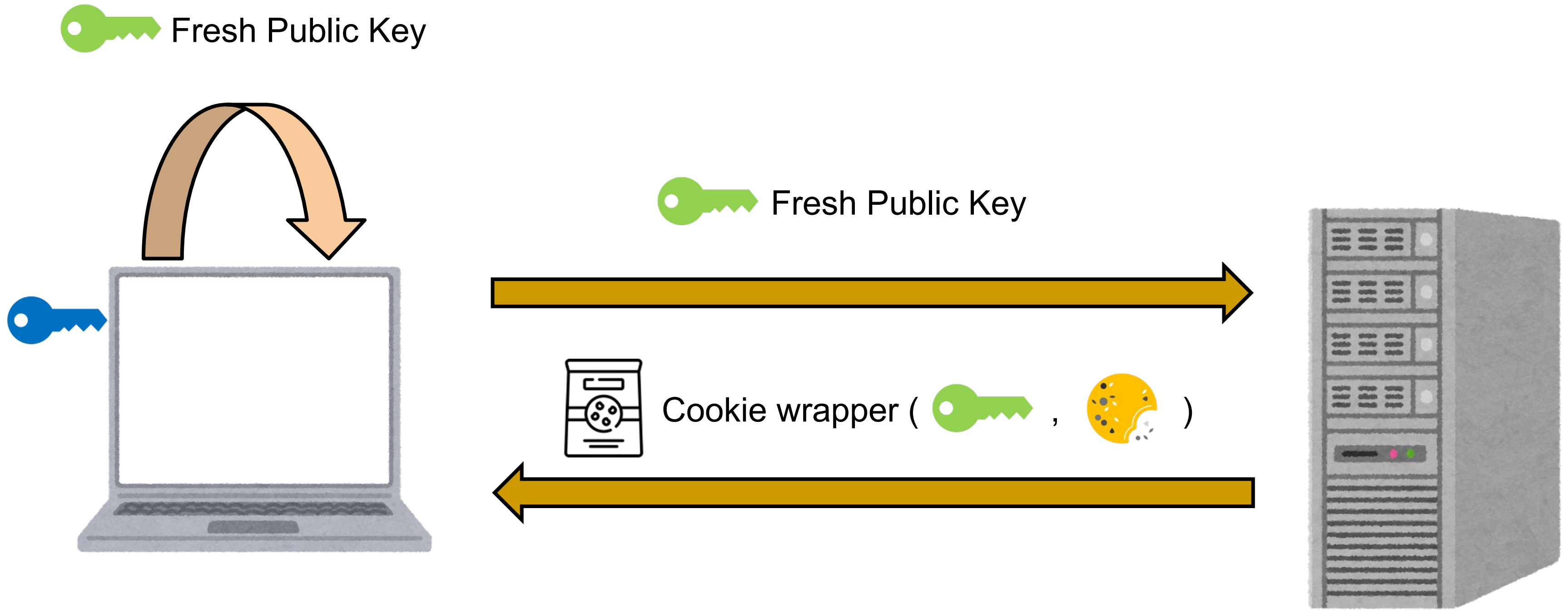

Client Device


Server

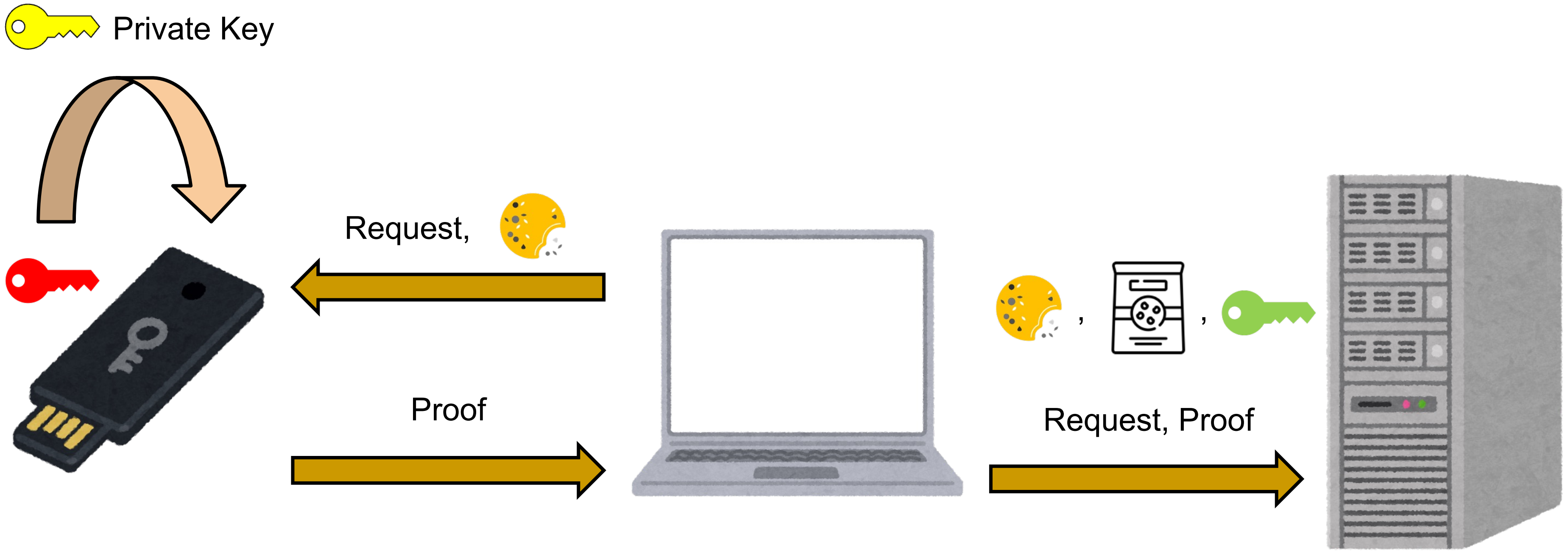
1. Setup phase



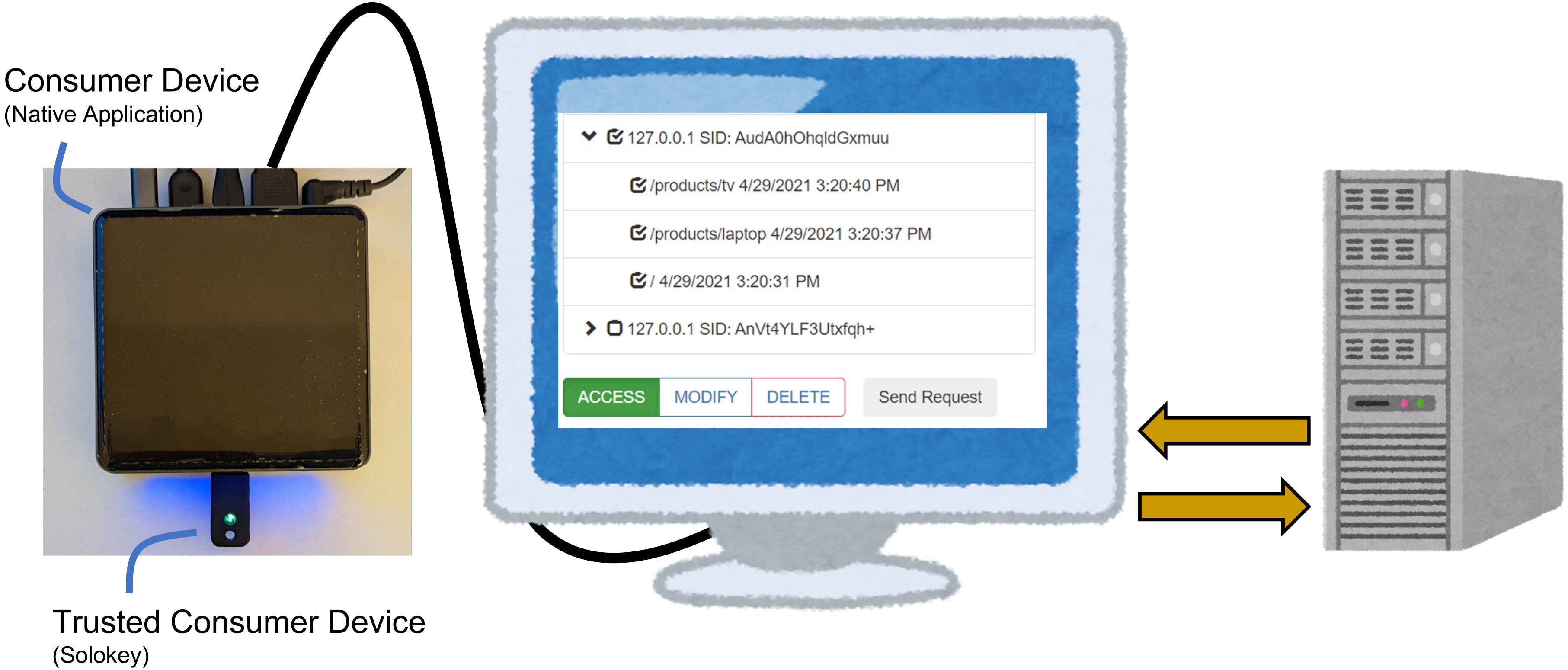
2. Visiting a website



3. Proving data ownership



Implementation



127.0.0.1 SID: AudA0hOhqldGxmuu
/products/tv 4/29/2021 3:20:40 PM
/products/laptop 4/29/2021 3:20:37 PM
/ 4/29/2021 3:20:31 PM
127.0.0.1 SID: AnVt4YLF3Utxfqh+

Trusted Consumer Device: Solokey

- FIDO2 security key
- Open source firmware & bootloader
  - Hardware schematics too :)
- Specs
  - Arm Cortex-M4 MCU (80 MHz)
  - 64 kB RAM
  - 256 kB flash memory
  - Random Number Generator
  - Physical button
  - Multiple interfaces (USB-A, USB-C, NFC)
- Solokey Hacker: Unlocked bootloader

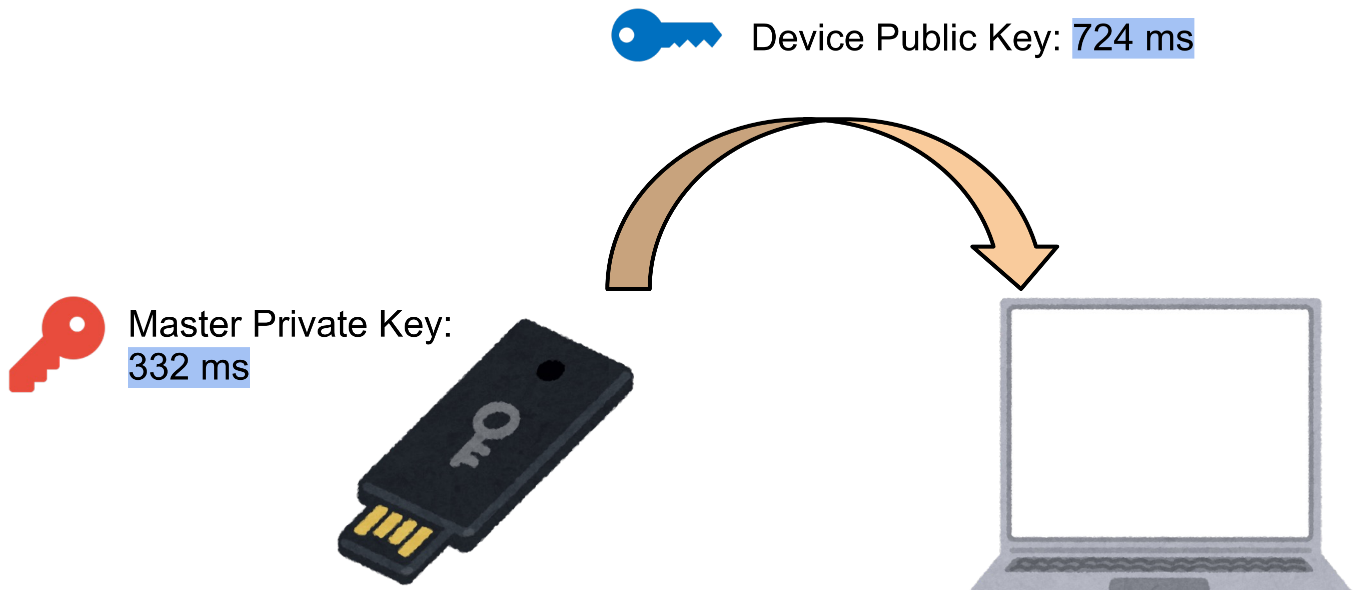
Using Solokey: Challenges

- Documentation
  - Very detailed, but distributed across different websites (Github docs, Readme, Official docs)
  - Some missing details
    - What to do if Solokey becomes unresponsive?
    - What if the official serial monitor doesn’t work?
- Limited resources
  - 64 kB RAM, 256 kB ROM
  - Can we add custom code/data?
- Low CPU frequency
  - 80 MHz
  - What would the eval numbers be like?

Using Solokey: Solutions

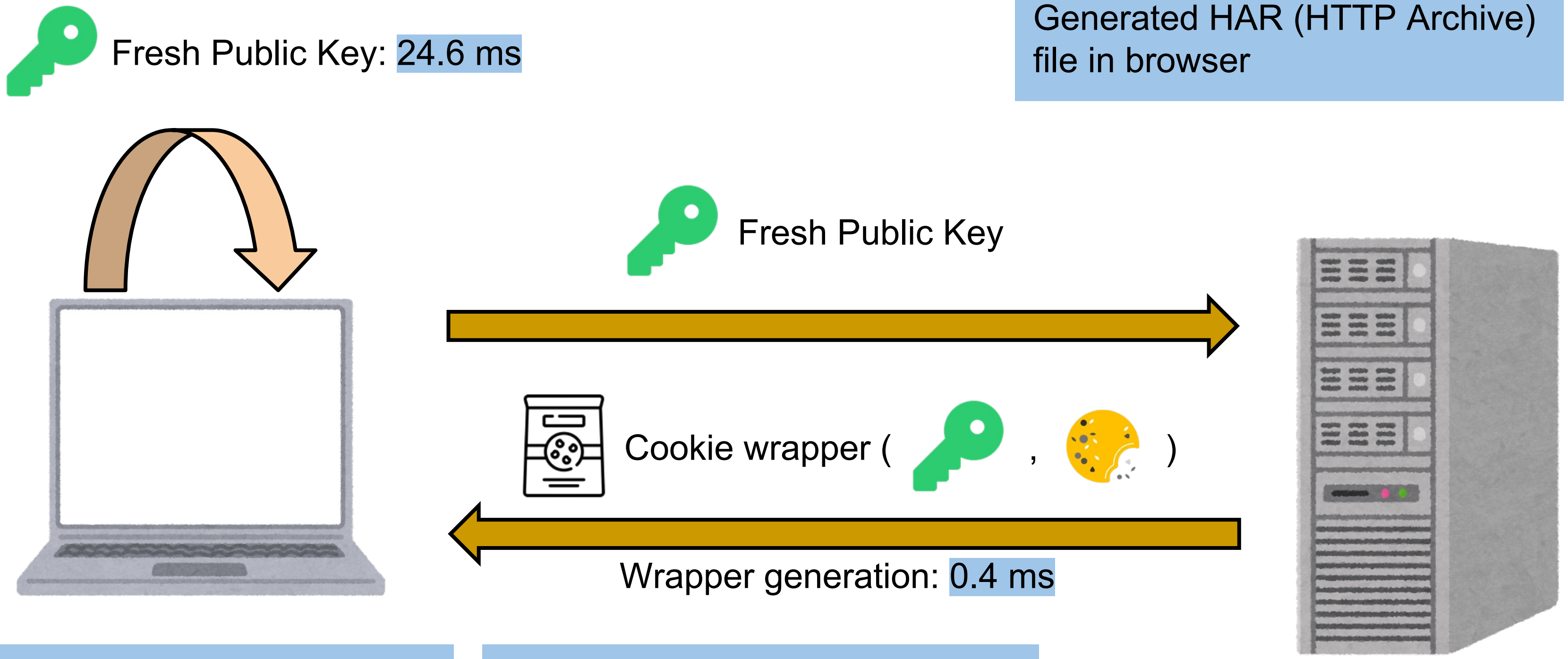
- Documentation: Details in one README.
    - How do I…
        - build my code for Solokey? → Follow our detailed steps
        - add my code? → Follow our examples
        - write code to communicate with Solokey? → See our sample code
        - revive an unresponsive Solokey? → Follow these instructions
        - debug Solokey without using default serial monitor? → Use `minicom`
- Limited resources
    - New code may need to go on a diet
    - Use existing code (e.g., master key pair generation, storage)
- Low CPU frequency

Evaluation: Setup



Evaluation: Visiting a website

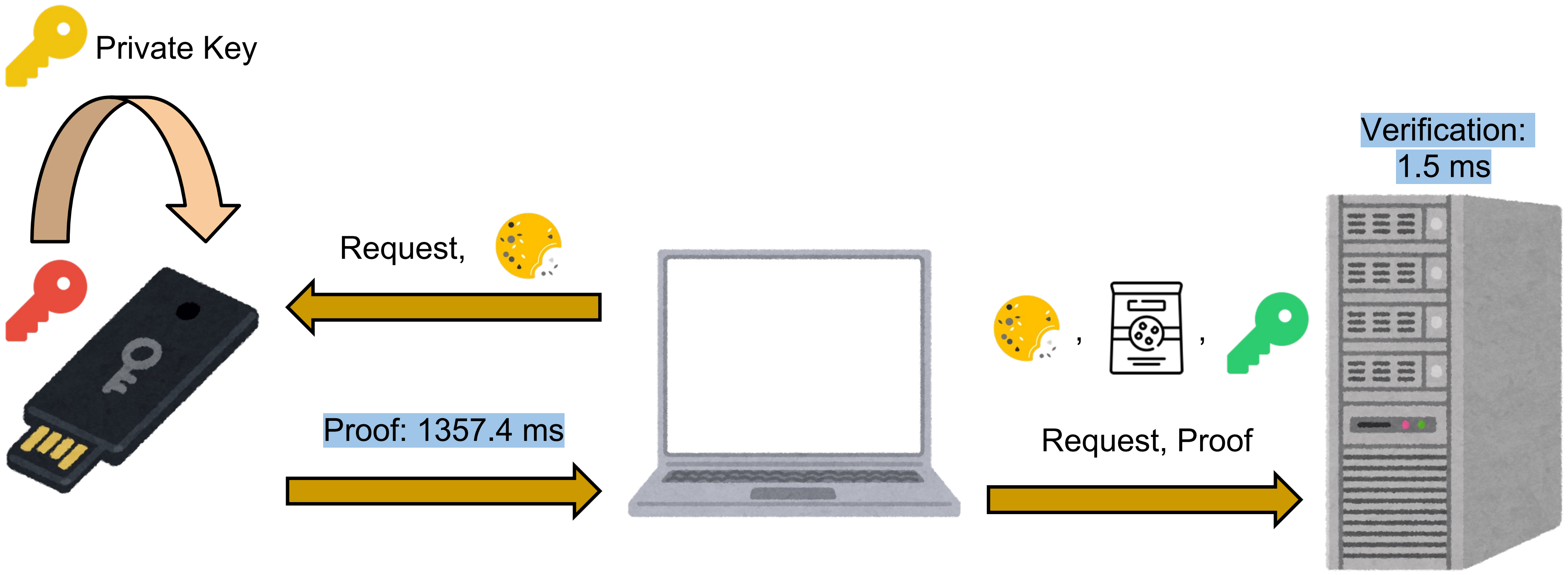
Measuring storage:
Generated HAR (HTTP Archive) file in browser



Storage (annual): 22.61 MB[1]

[1] 163 web page visits per day * 0.38 kB per visit (Crichton et al. *How Do Home Computer Users Browse the Web?* https://dl.acm.org/doi/abs/10.1145/3473343)

Evaluation: Proving data ownership



Code availability

- Official VICEROY Github repo
  - Chrome extension (Consumer device UI)
  - Native application (Consumer device)
  - Server (VICEROY APIs)
  - Solokey firmware (Trusted Consumer device)
  - VICEROY protocol specification via Tamarin Prover
- Contact:
  - Yoshimichi Nakatsuka (nakatsuy[at]uci.edu)