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

Scott Jordan¹, Yoshimichi Nakatsuka¹, Ercan Ozturk¹, Andrew Paverd², Gene Tsudik¹

¹ University of California, Irvine
² Microsoft
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

- Device Public Key
- Master Private Key
2. Visiting a website

Fresh Public Key

Cookie wrapper (, )
3. Proving data ownership

Private Key

Request, Proof

Request, Proof

Proof
Implementation

Consumer Device
(Native Application)

Trusted Consumer Device
(Solokey)
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

Device Public Key: 724 ms

Master Private Key: 332 ms
Evaluation: Visiting a website

- Fresh Public Key: 24.6 ms
- Cookie wrapper: 0.4 ms
- Wrapper generation: 0.4 ms
- Storage (annual): 22.61 MB

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

Storage (annual): 22.61 MB

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

Private Key

Request, Proof: 1357.4 ms

Verification: 1.5 ms

Request, Proof
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)