Tapas
Tap-based Authentication with a Smartphone

Daniel McCarney, David Barrera,
Jeremy Clark, Sonia Chiasson,
and Paul van Oorschot
Carleton University

Passwords are here to stay; at least for the short-term. We are interested in password management that requires no server-side changes.

Password Managers come in two flavours: with or without the protection of a master password. The former provides no protection against theft or loss; the latter offers weak protection against offline attacks and an unclear mental model of when the wallet is open and closed.

Tapas is a novel type of password manager based on pairing a **smartphone wallet** with a **browser-based manager** on a computer, such that both are required to log-in. To log-in, the user selects the account on the smartphone, taps it, and the associated credentials are securely transmitted to the browser. Tapas properties include:

- Passwords remain secure if either device is stolen
- Wallet is secure against offline/dictionary attacks
- Users do not have to remember even one password
- Phishing protection is provided by pinning passwords to URLs and HTTPS certificates
- In Tapas, the wallet is never left open

Protocol 1: Pairing the Manager and Wallet

User input: Upon a user choosing to set-up a new Wallet, the following protocol is initiated by the Manager.

Communication channel: a one-way untappable channel from the Manager to the Wallet.

1. The Manager generates an authentication key pair for itself $\langle pk_m, sk_m \rangle$ and sends its public key pk_m to the Wallet.
2. The Manager generates an authentication key pair for the Wallet $\langle pk_w, sk_w \rangle$ and sends the pair to the Wallet.
3. The Manager generates a secret key k for a symmetric key authenticated encryption scheme $\mathsf{Enc}_k()$.

Output: The Manager stores $\langle pk_m, pk_w, sk_m, k \rangle$ and erases sk_w. The Wallet store $\langle pk_m, pk_w, sk_w \rangle$.

Protocol 2: Storing a password

User input: Upon a user choosing to save a password, the following protocol is initiated by the Manager.

Communication channel: a mutually-authenticated secure channel with perfect forward secrecy between the Manager and the Wallet. The participants, respectively, identify themselves with pk_m and pk_w.

1. The Manager takes password p_i and site information s_i and computes $c_i = \mathsf{Enc}_k(p_i \| s_i)$.
2. The Manager sends $\langle c_i, s_i \rangle$ to the Wallet.
3. The Wallet prompts the user to create a tag t_i for referencing the site. It uses s_i to suggest a value for the tag.

Output: The Manager erases $\langle p_i, s_i, c_i \rangle$. The Wallet stores $\langle t_i, c_i \rangle$ and erases s_i.

Protocol 3: Retrieving a password

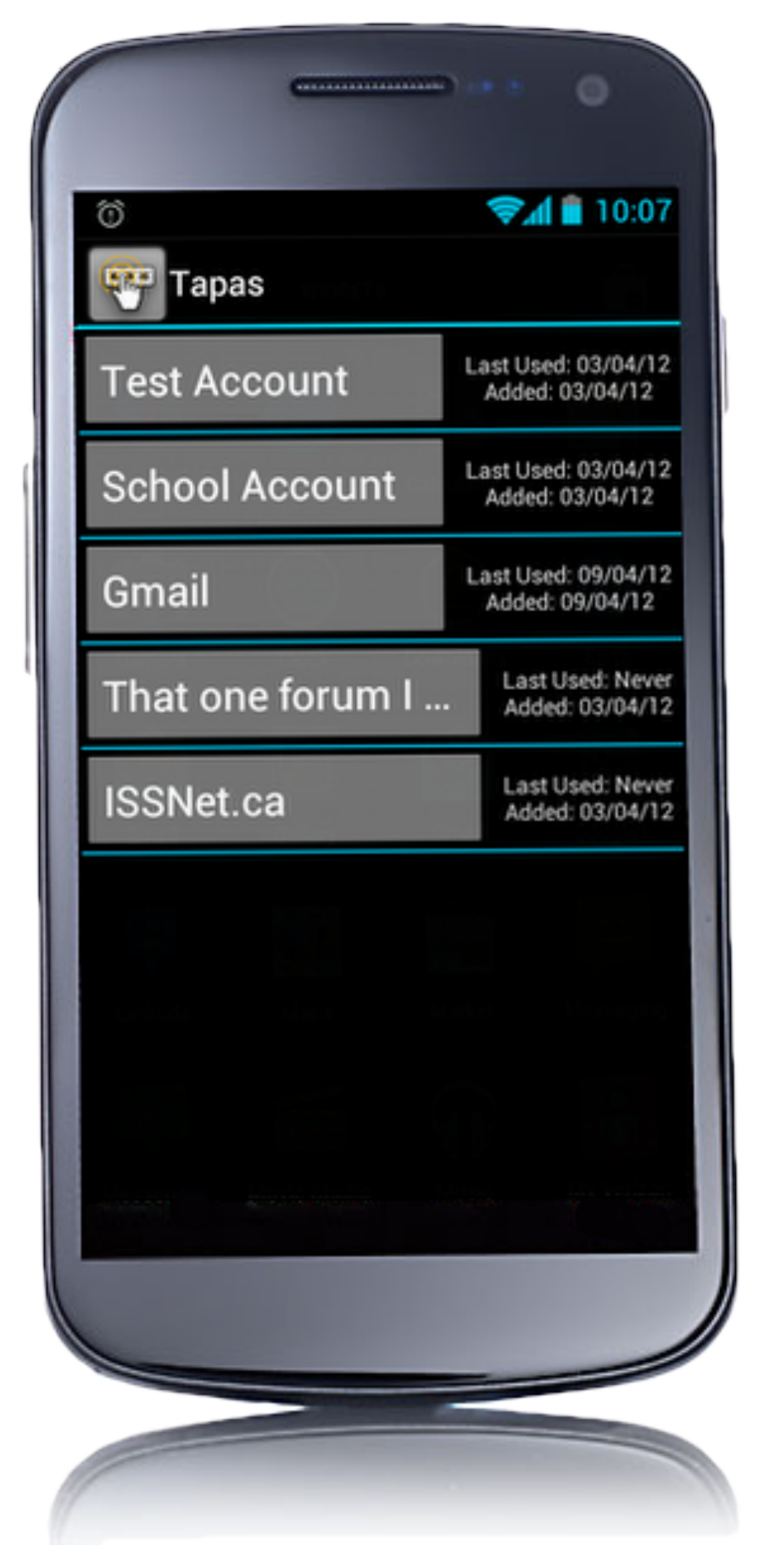
User input: Upon a user tapping the tag associated with a password, the following protocol is initiated by the Wallet.

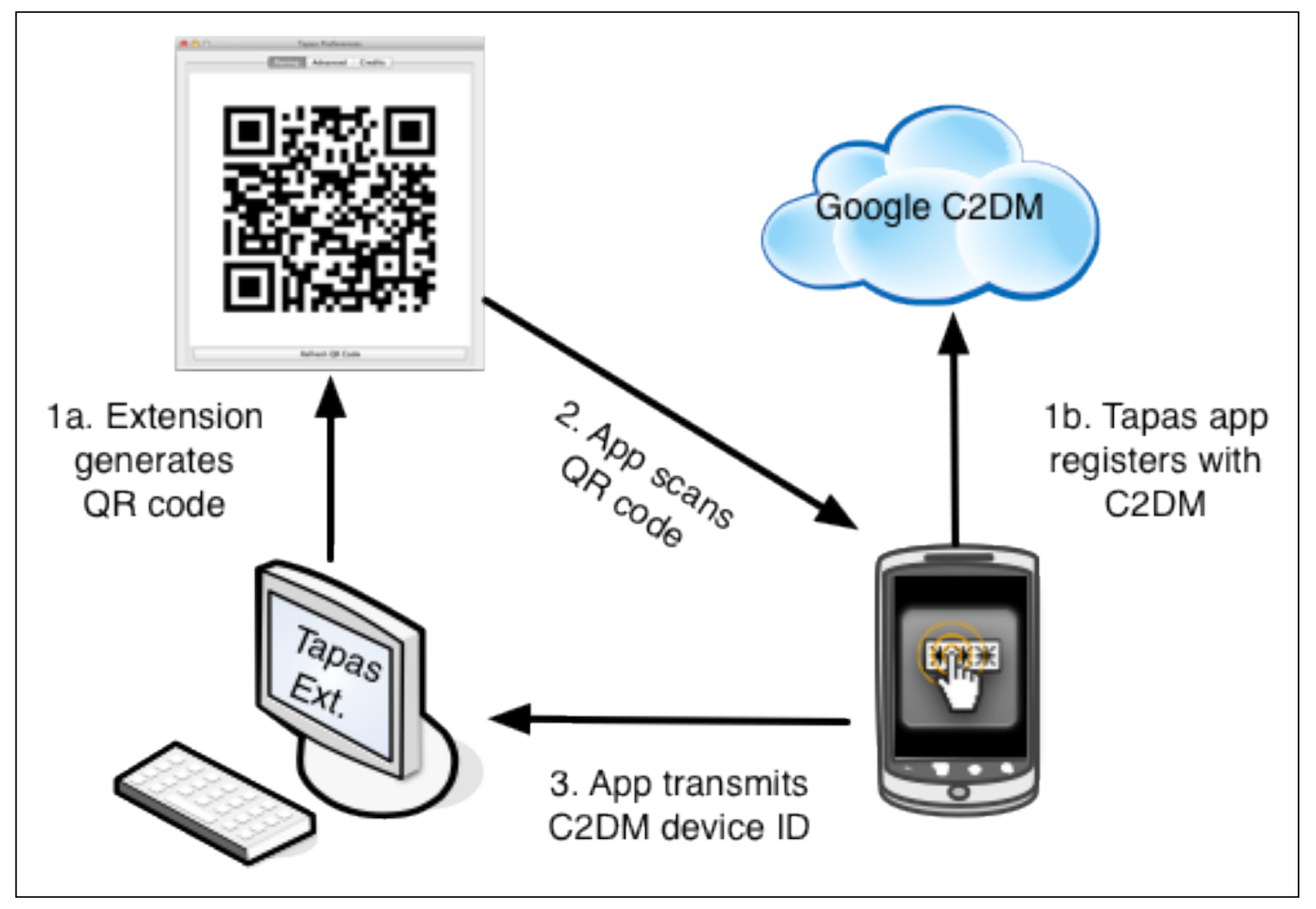
Communication channel: a mutually-authenticated secure channel with perfect forward secrecy between the Manager and the Wallet. The participants, respectively, identify themselves with pk_m and pk_w.

1. The Wallet retrieves the c_i value associated with the tapped t_i, and sends c_i to the Manager.
2. The Manager decrypts and authenticates c_i to retrieve s_i and p_i.
3. The Manager checks that s_i matches the site information for current site.
4. The Manager inputs p_i to the site.

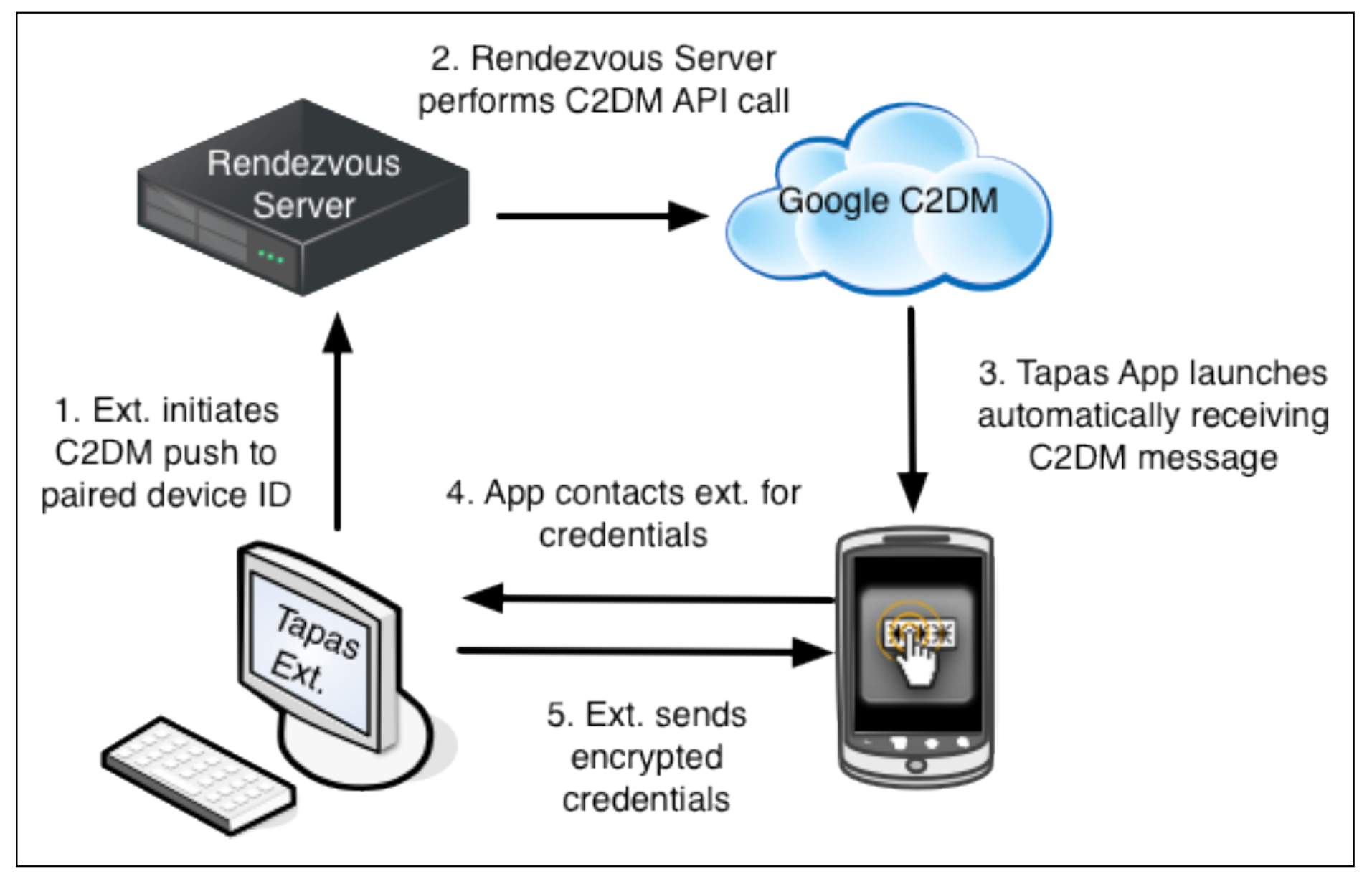
Output: The Manager erases $\langle p_i, s_i, c_i \rangle$.

Tapas is implemented with an **Android app** serving as the wallet and a **Firefox extension** serving as the manager.

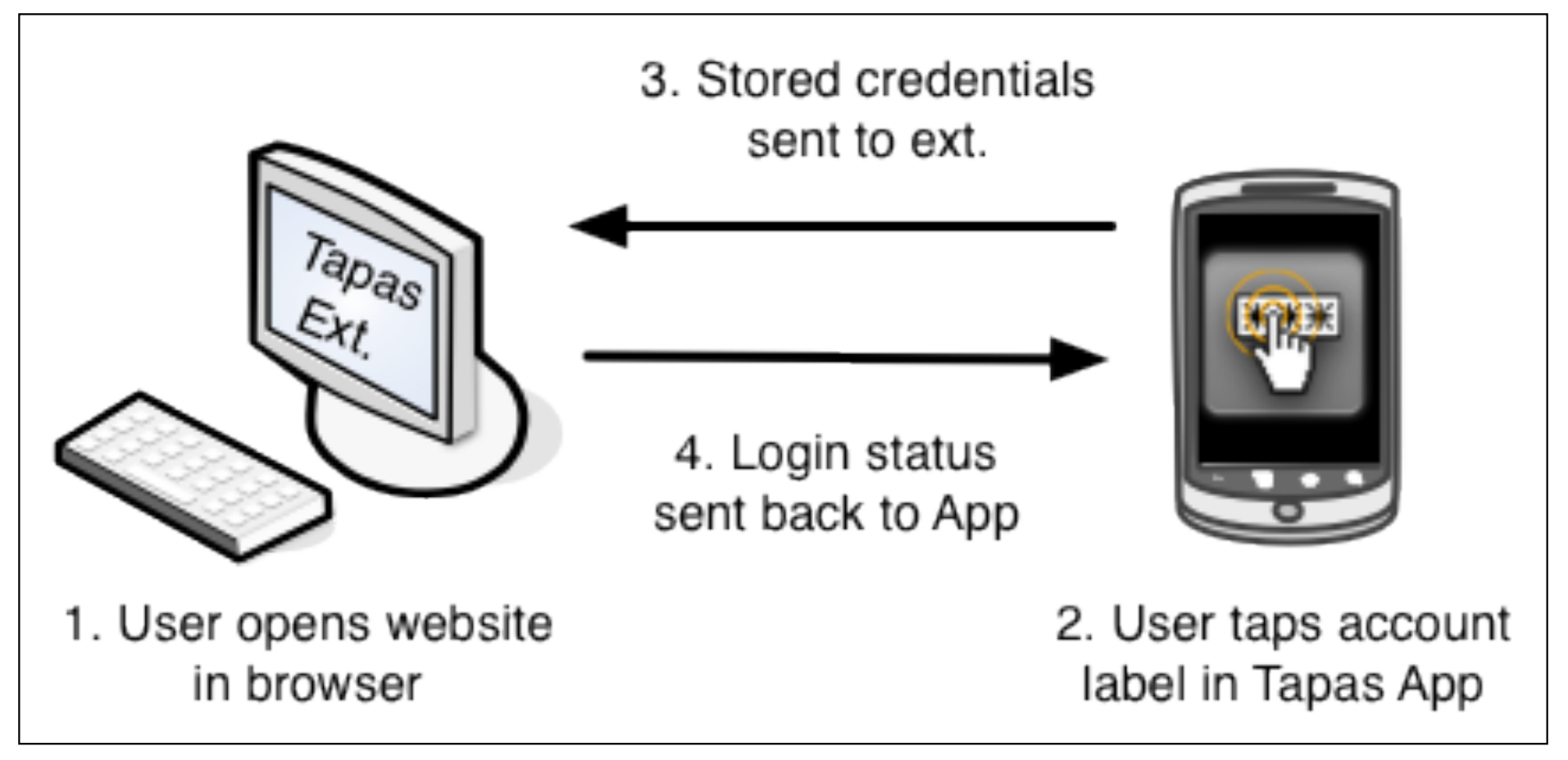



Google C2DM
1a. Extension generates QR code
2. App scans QR code
1b. Tapas app registers with C2DM
Tapas Ext.
3. App transmits C2DM device ID

Setup: The smartphone is paired with a Firefox extension, which displays a QR code. The smartphone simultaneously registers with Google's cloud-to-device messaging service for receiving push notifications.

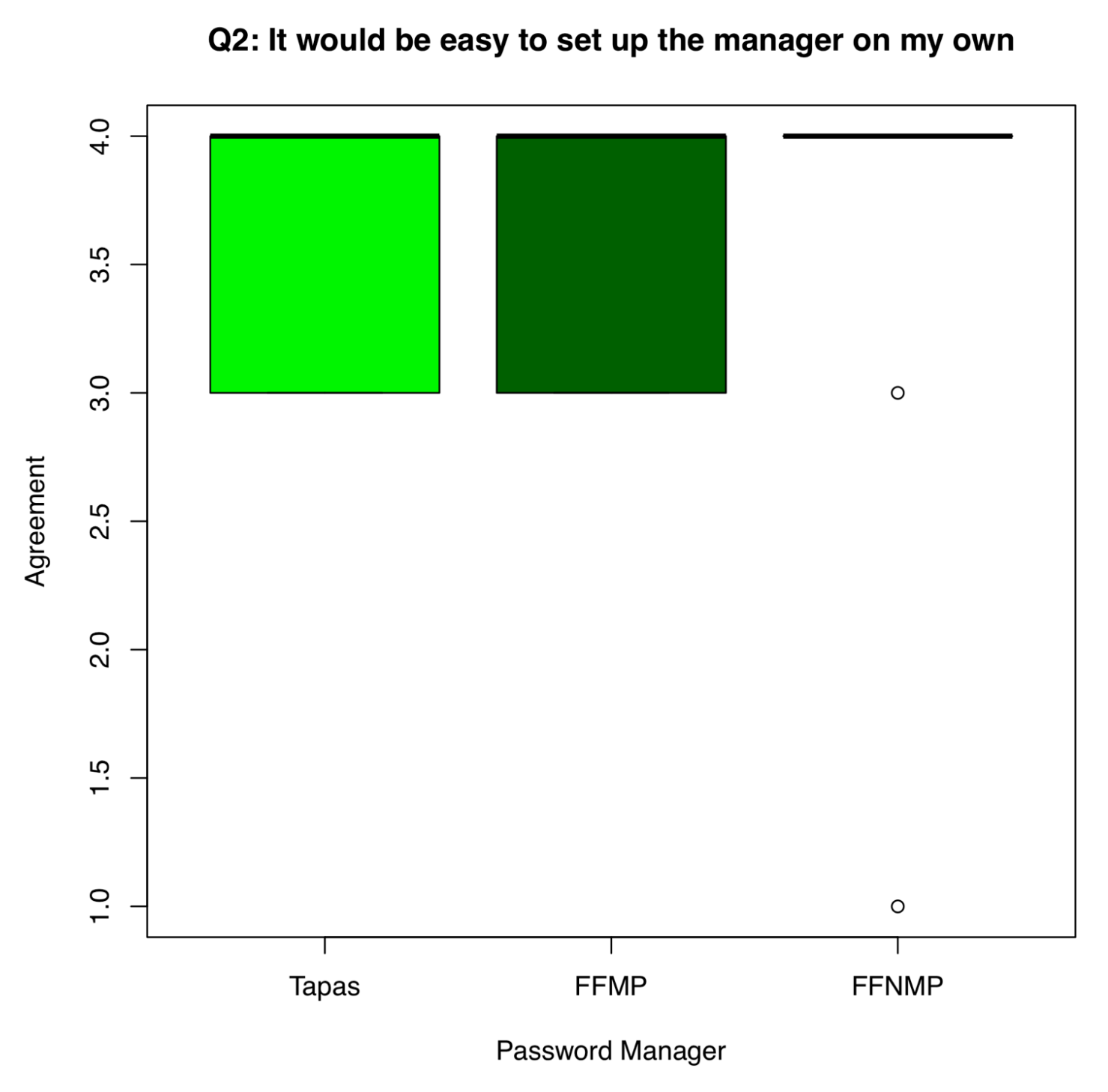

2. Rendezvous Server performs C2DM API call
Rendezvous Server
Google C2DM
1. Ext. initiates C2DM push to paired device ID
3. Tapas App launches automatically receiving C2DM message
4. App contacts ext. for credentials
Tapas Ext.
5. Ext. sends encrypted credentials

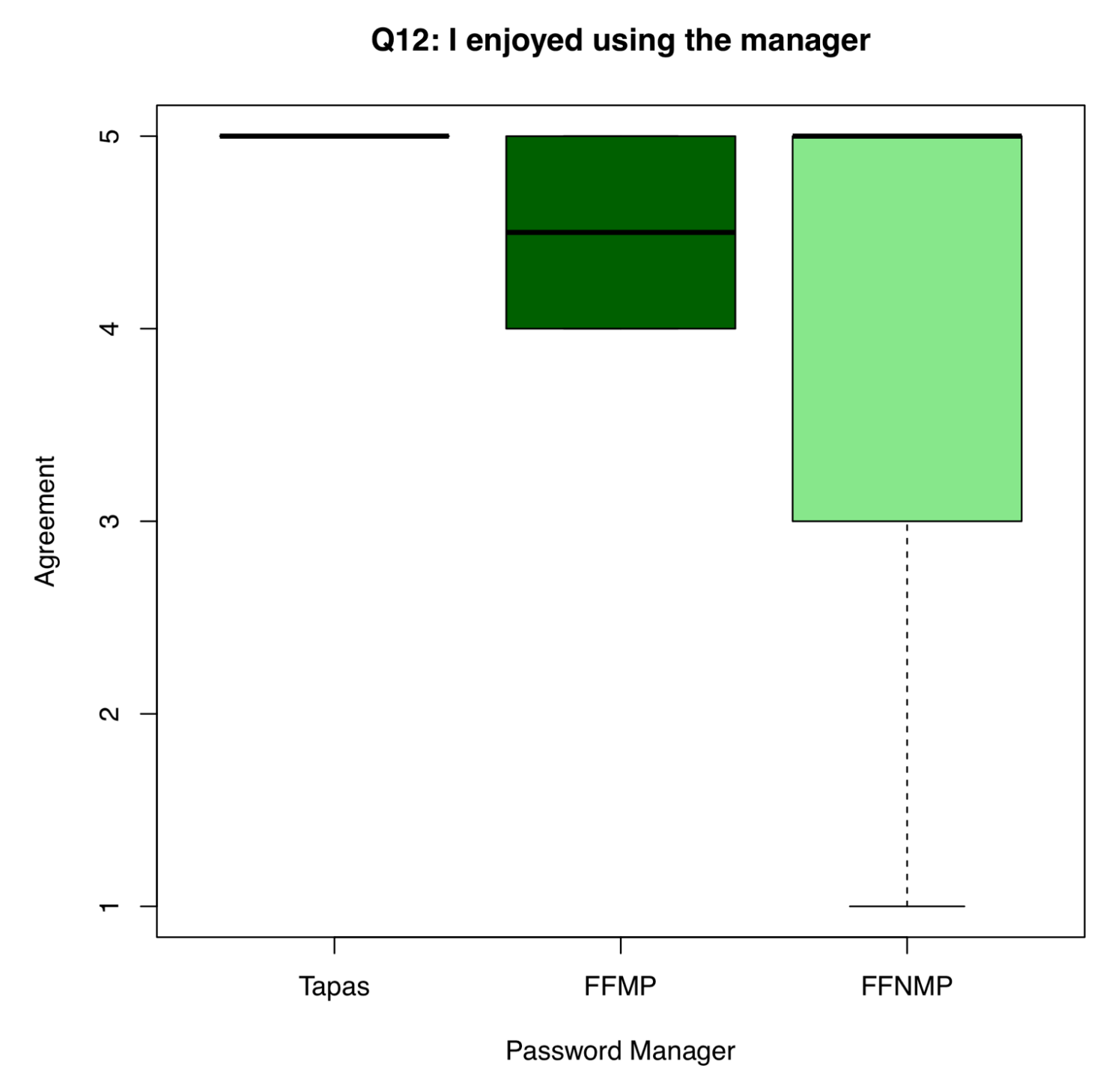
Password Save: The extension detects when a password is being entered, and prompts the user to save it with Tapas. The extension notifies the device (via a push notification) that a password needs to be stored. The smartphone retrieves the encrypted credentials from the extension.


3. Stored credentials sent to ext.
Tapas Ext.
4. Login status sent back to App
1. User opens website in browser
2. User taps account label in Tapas App

Login: The user visits the website and then taps on the corresponding account in the Tapas application on the smartphone. The stored credentials are transmitted to the extension and filled into the login form, after checking that the URL and SSL certificate chain matches those that were stored.

User Study: we conducted a between-subjects study with 30 participant. Each participant used either Tapas, the Firefox password manager in its default configuration (FFNMP), or Firefox with a master password (FFMP)


Q2: It would be easy to set up the manager on my own
Agreement
Tapas
FFMP
FFNMP
Password Manager


Q12: I enjoyed using the manager
Agreement
Tapas
FFMP
FFNMP
Password Manager