



Protecting Web Passwords from Rogue Servers using TEEs

*Klaudia Krawiecka, Arseny Kurnikov, Andrew Paverd,
Mohammad Mannan, N. Asokan*

Password database breaches

560 million more passwords
were exposed -- was yours?

Hundreds of millions of passwords just surfaced online. Here's how to keep your identity safe.

Ad closed by Google

Security

As if **ransomware like WannaCry** wasn't enough to keep you up at night, there's another password breach to worry about.

May 16, 2017 3:08 PM PDT

It's also cause for concern. Although "online safety" feels increasingly like an oxymoron these days, there are still steps you can take to protect yourself when breaches like this occur.

by **Rick Broida**

May 16, 2017 3:08 PM PDT

@cheapskateblog

<https://www.cnet.com/how-to/protect-yourself-from-the-latest-database-breach/>

Phishing

August 25, 2017

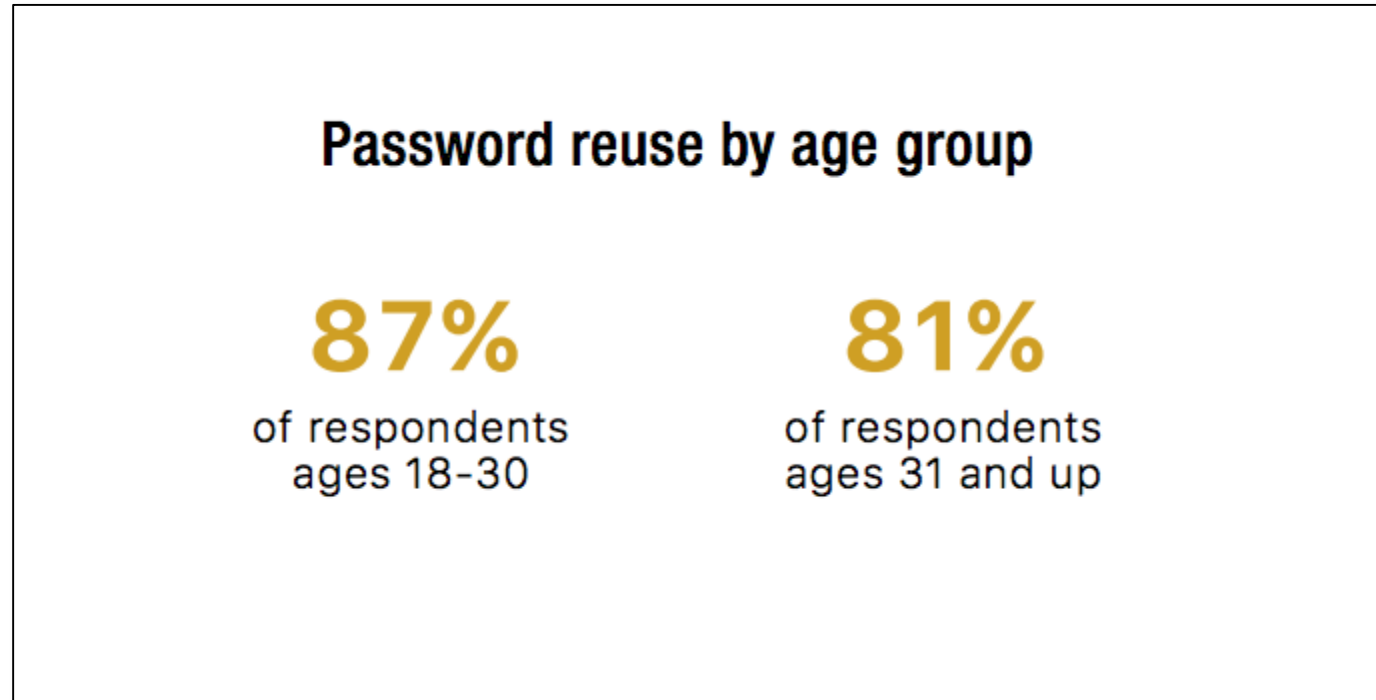
Recent phishing attacks reportedly capitalize on Office 365 security holes

Researchers from cloud security company **Avanan** have reported finding two ways that phishers are evading **Microsoft** Office 365 Security protections: one using "hexidecimal escape characters" to conceal coding and links, and the other by compromising SharePoint files.

The first method involves emails with an HTML attachment that contain a small excerpt of JavaScript that is obscured in hexadecimal **escape characters**. "Therefore, no links are visible, but when opened, it presents a locally-generated phishing page with login instructions," the company explains in an Aug. 24 **blog post**.

In one recent example, Avanan came across a phishing email, purportedly sent by PayPal, that displayed a fraudulent login page, asking for account information such as name, address, phone number and password. By entering the information and clicking a submit button, the recipient unknowingly transmits his or her information to the cybercriminals.

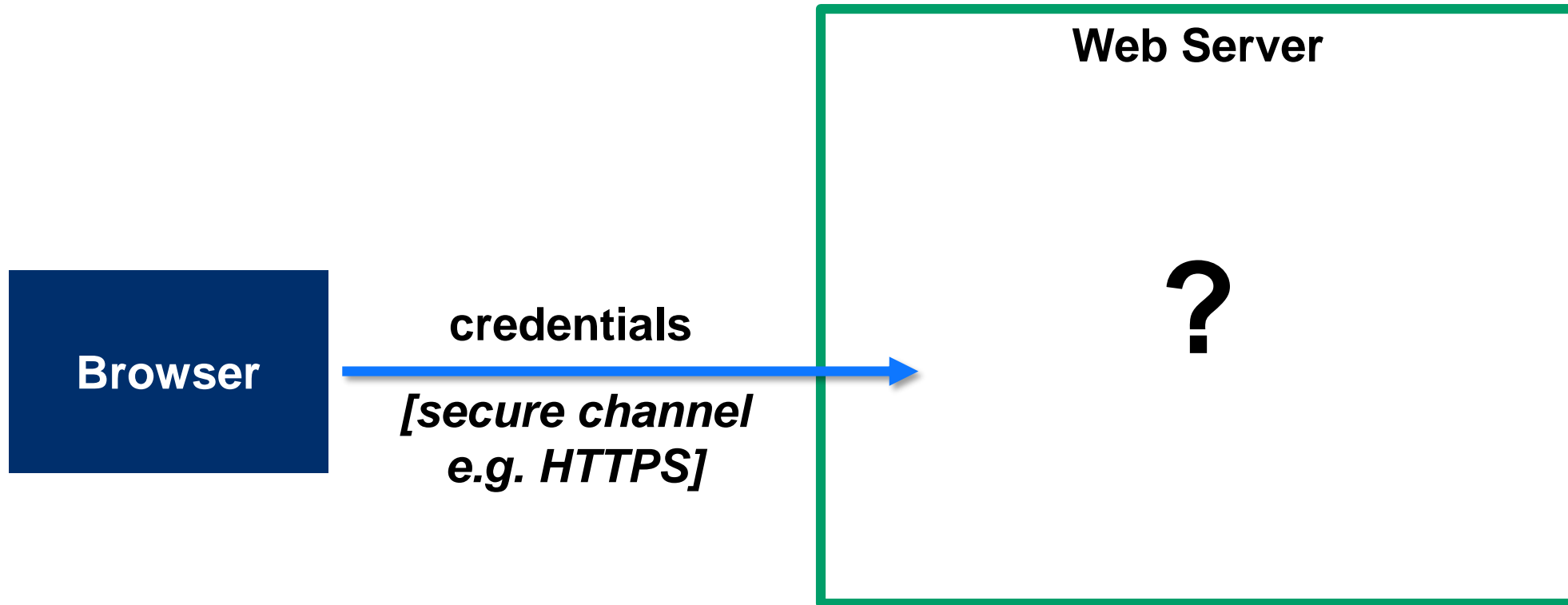
Password reuse



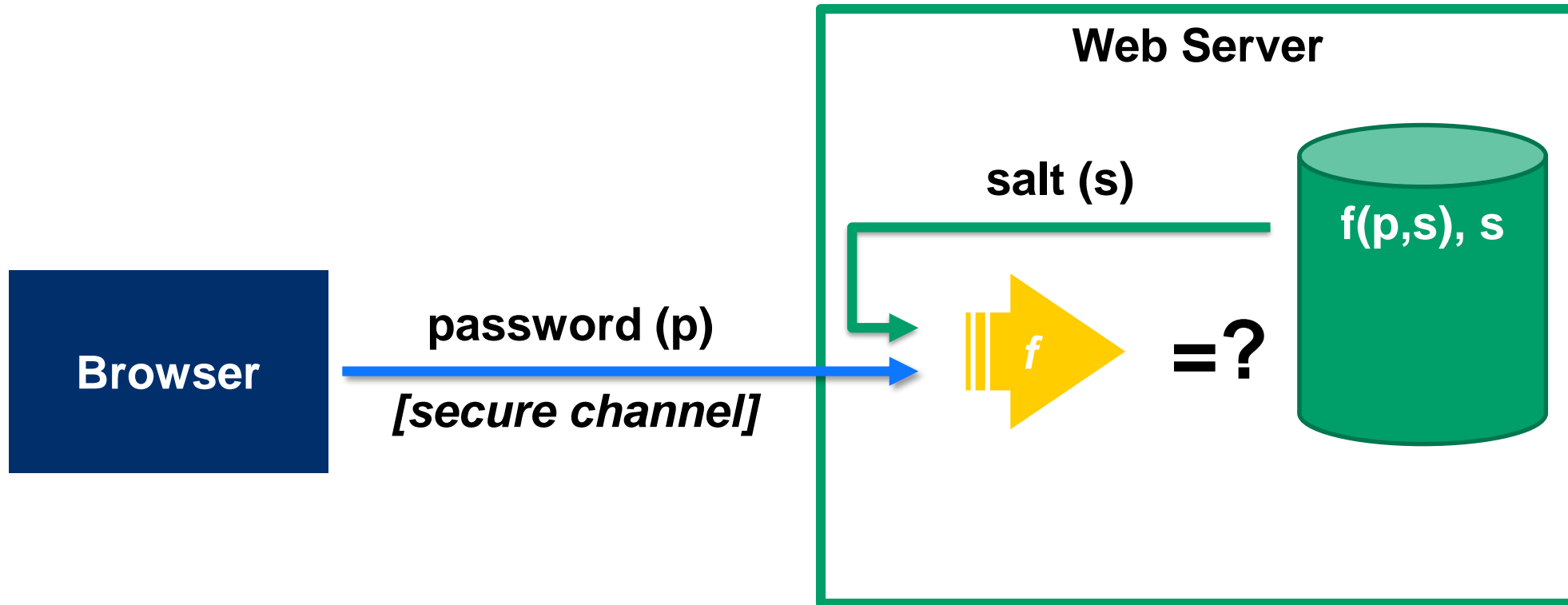
*survey of 1,000 smartphone users in 2017

<https://keepersecurity.com/assets/pdf/Keeper-Mobile-Survey-Infographic.pdf>

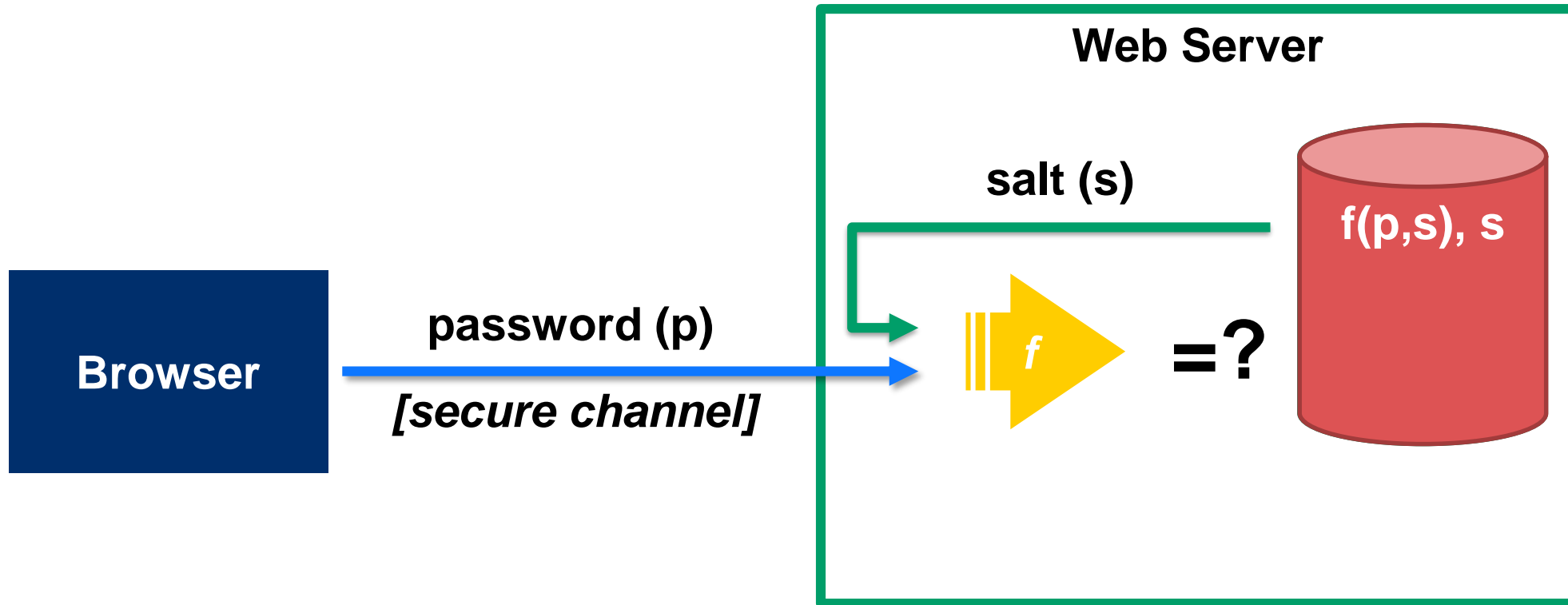
What happens on the server?



Storing passwords

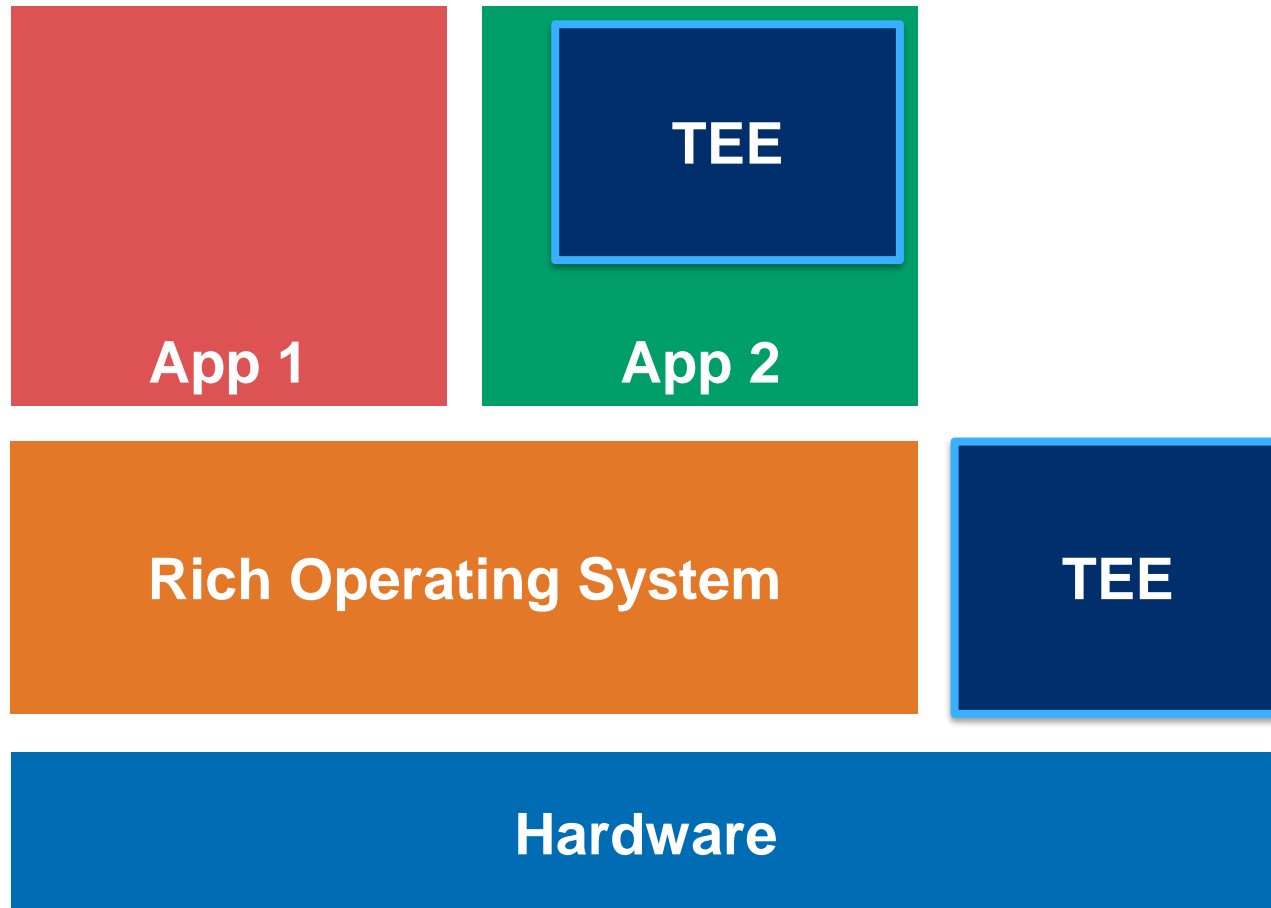


Password database breach



Offline guessing: guess passwords, apply $f()$, compare to leaked database.

Trusted Execution Environments



TEE features

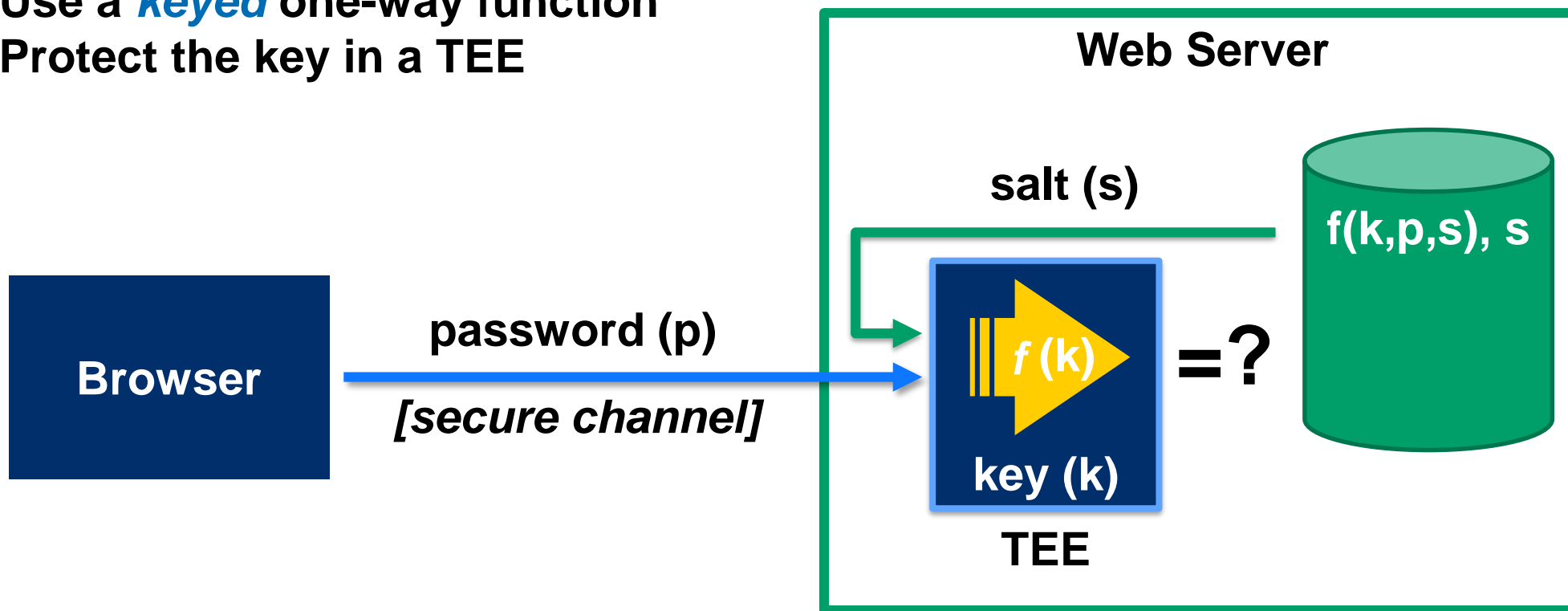
- Isolated execution
- Sealed storage
- Remote attestation

Available hardware TEEs

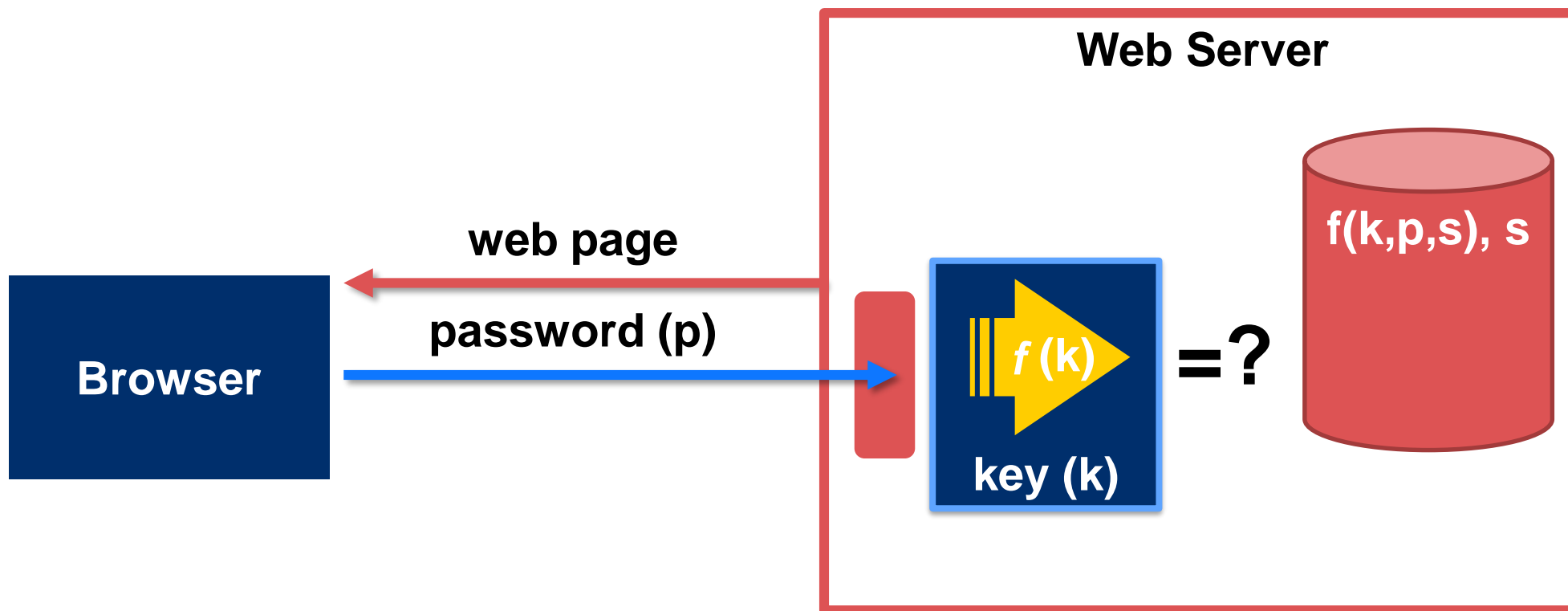
- ARM TrustZone
- Intel SGX

Storing passwords *securely*

- Use a *keyed* one-way function
- Protect the key in a TEE



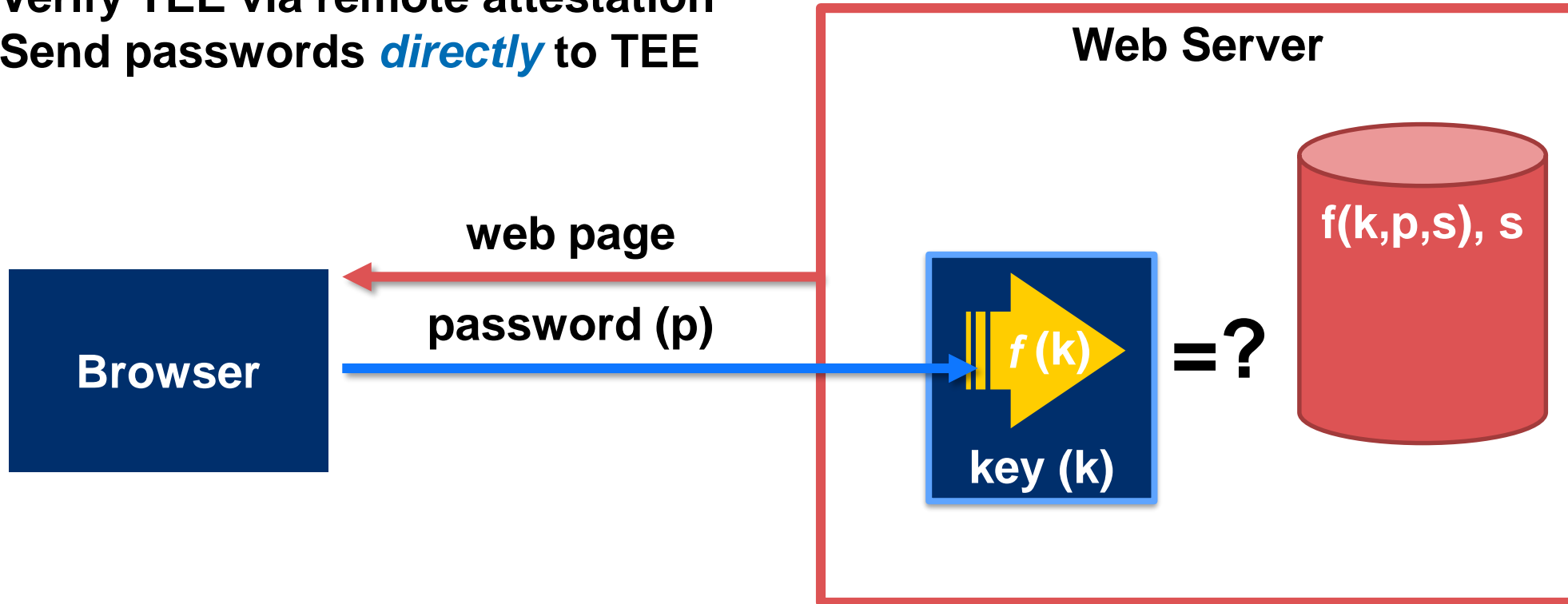
Compromised server (or Phishing)



Man-in-the-Middle: intercept passwords in transit.

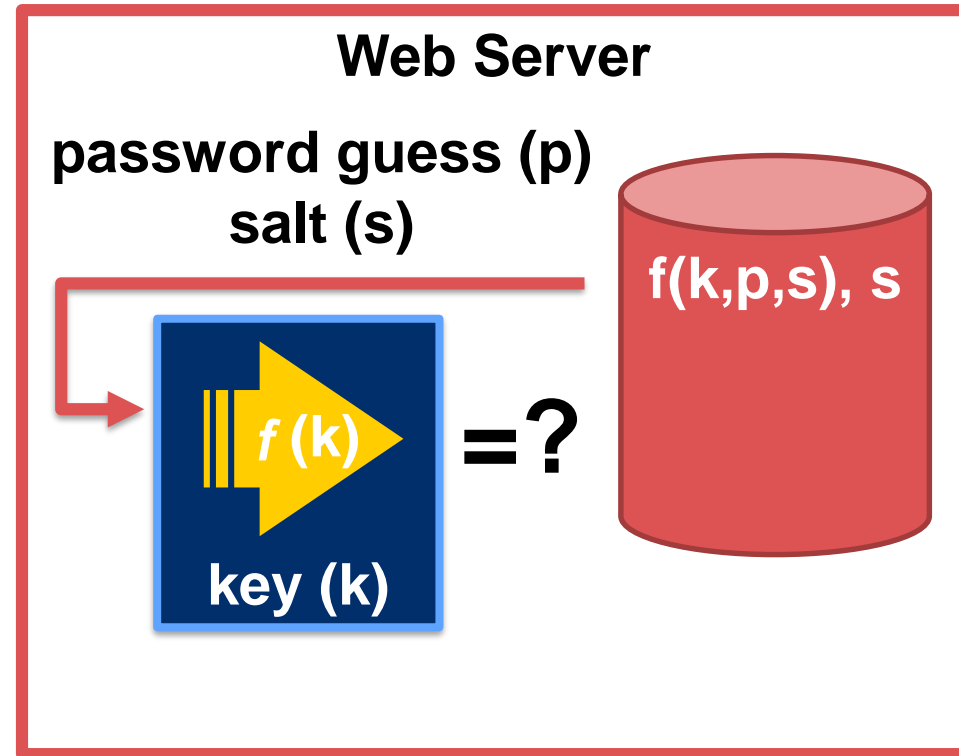
Transferring passwords *securely*

- Verify TEE via remote attestation
- Send passwords *directly* to TEE



Actively malicious server

Browser

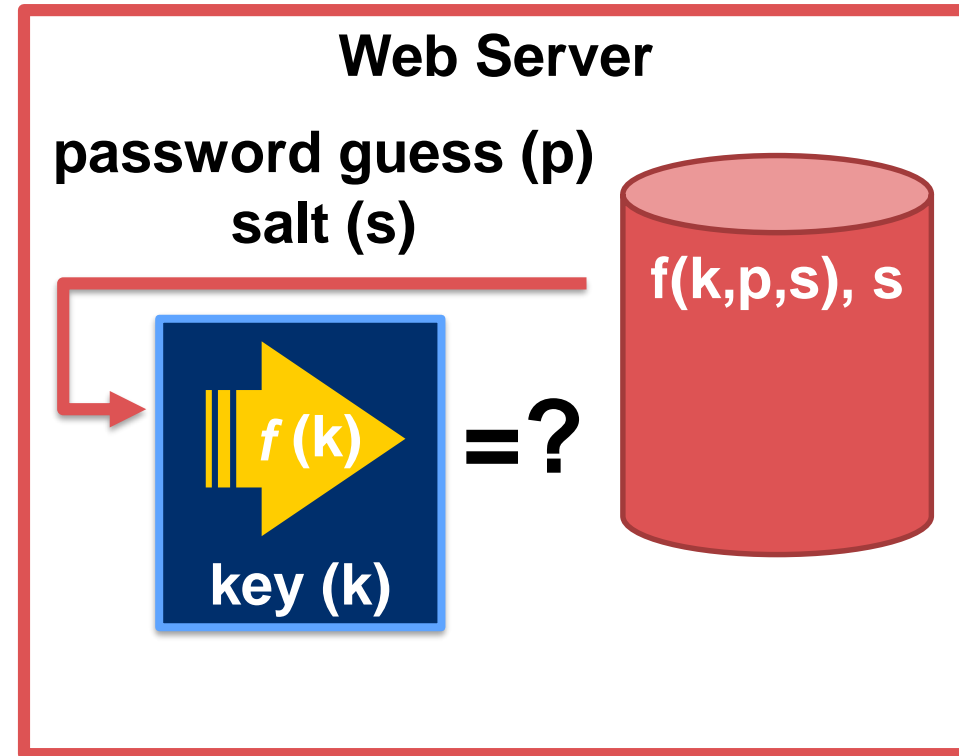


Online guessing: guess passwords, send to TEE, compare to database.

Processing passwords *securely*

- *Rate-limiting* in the TEE
(but can't use User ID)

Browser



Problem definition

Adversary capabilities

- Access passwords database
- Modify web content
- Access server-client communication
- Execute server-side code
- Launch phishing attacks

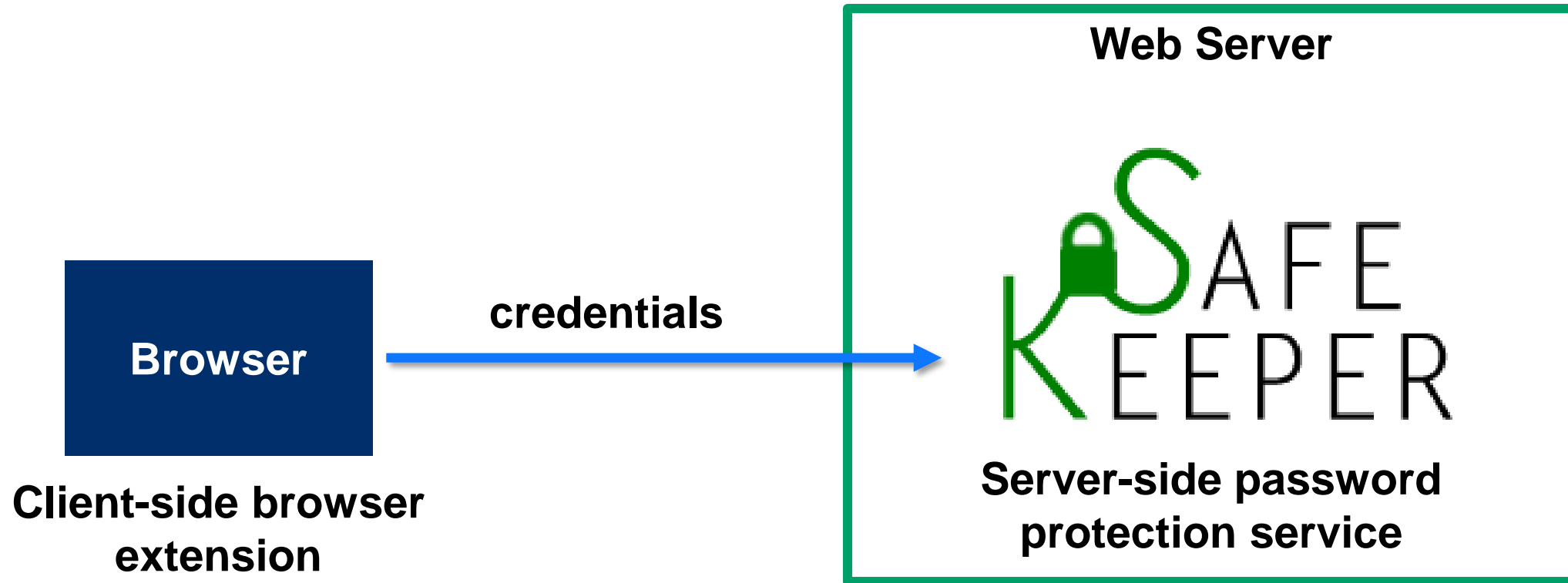
Requirements

- Password protection
 - a) Passwords can only be obtained through guessing
 - b) Offline guessing must be computationally infeasible
 - c) Online guessing must be throttled
- User awareness

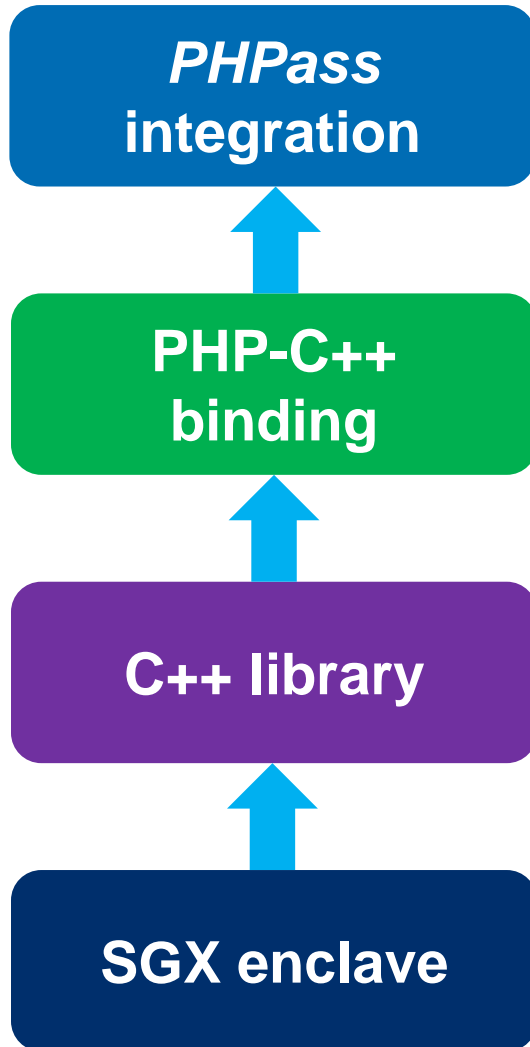
Design goals

- Minimal performance overhead
- Minimal software changes
- Ease of upgrade
- Backup and recovery

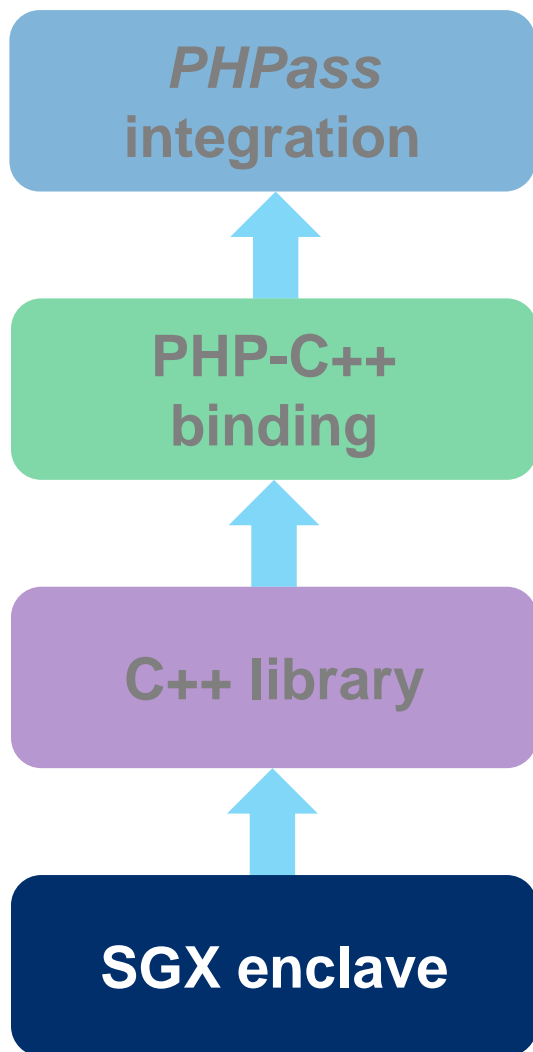
SafeKeeper



SafeKeeper – server side



SafeKeeper – server side



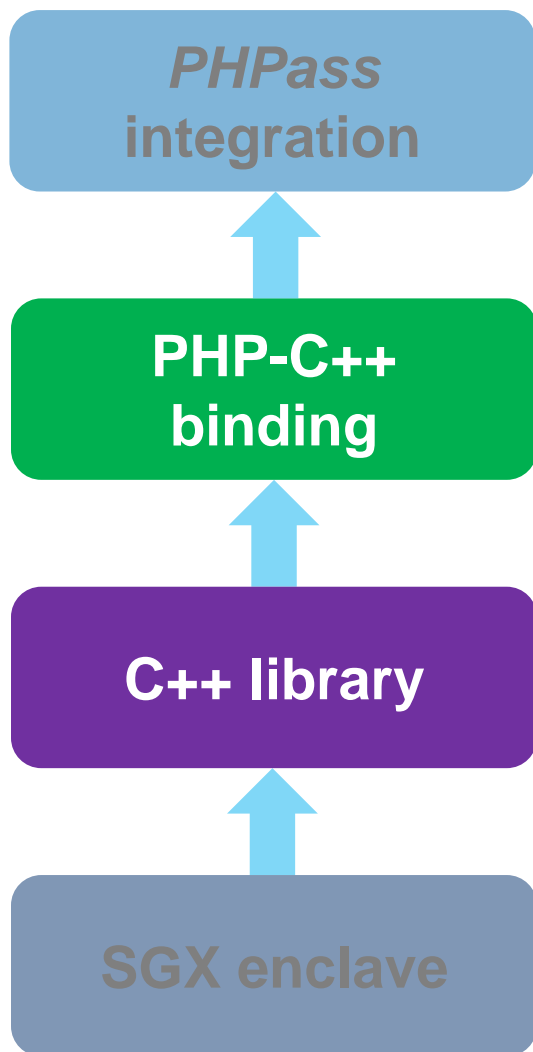
Password processing

- Key generated in enclave
- CMAC from `sgx_tcrypto` library
 - 128 bit key
 - AES-NI hardware acceleration

Rate limiting

- Per-user rate limiting based on *salt*
- In-TEE map of salts and attempts
 - Uses SGX trusted time

SafeKeeper – server side



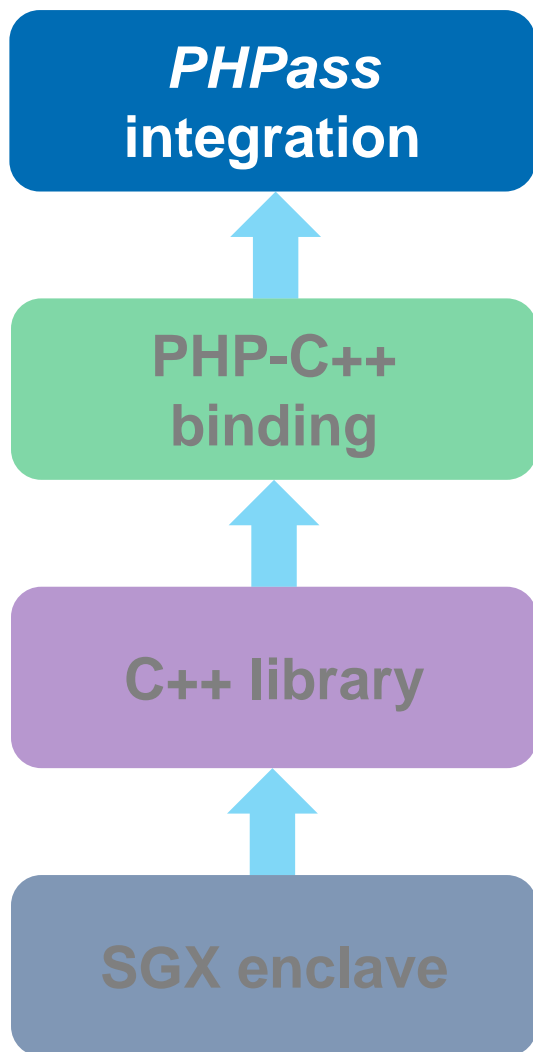
PHP-CPP

- C++ library for writing PHP extensions
<http://www.php-cpp.com/>

C++ Library

- Enclave initialization
- Sealed data storage/retrieval

SafeKeeper – server side



PHPass library

- Used by WordPress, Joomla, etc.
- Default: multi-round MD5 (!)

Enhanced to use our **SGX enclave**

WordPress using SafeKeeper

The screenshot displays the WordPress dashboard for a site named "Restaurant World Tou...". The interface includes a top navigation bar with "Upgrade to Pro", "New Post", and a user profile for "Dave". A left sidebar contains navigation links for Dashboard, Home, Comments I've Made, Site Stats, Akismet Stats, My Blogs, Blogs I Follow, Store, Posts, Media, Links, Pages, Comments, and Feedbacks. The main content area is titled "Dashboard" and features a "Right Now" widget with the following statistics:

CONTENT	DISCUSSION
8 Posts	9 Comments
1 Page	9 Approved
5 Categories	0 Pending
52 Tags	0 Spam

Below the statistics, it indicates the theme is "Hemingway with 7 Widgets" and that "Akismet has protected your site from 786 spam comments already. There's nothing in your spam queue at the moment." A "STORAGE SPACE" widget shows "3,072MB Space Allowed" and "0.08MB (0%) Space Used". To the right is the "QuickPress" form, which includes a title field, an "Add Media" button, a content area, a "Tags" field, and "Save Draft", "Reset", and "Publish" buttons. At the bottom right, the "Recent Drafts" section shows "There are no drafts at the moment".

“WordPress was used by more than 27.5% of the top 10 million websites as of February 2017”

https://w3techs.com/technologies/overview/content_management/all/

Performance

Scalability (PHPass)

- Unmodified: 446 (± 10) passwords/second
- SafeKeeper: 1653 (± 70) passwords/second

Scalability (Enclave only)

- 101,337 (± 4186) passwords/second

Memory Requirements

- 110 MB for in-enclave map of 1 million users

Setup: Intel Core i5 6500 3.2 GHz, 8 GB RAM, Ubuntu 16.04

Deployability

Software changes

- Drop-in replacement for hash function in PHPass
- Fewer than 10 lines of PHPass code changed

Upgrade path

- Can transparently upgrade existing databases without user input

Backup and recovery

- Can be distributed across multiple enclaves for scalability and failure tolerance

Problem definition

Adversary capabilities

- Access passwords database
- Modify web content
- Access server-client communication
- Execute server-side code
- Launch phishing attacks

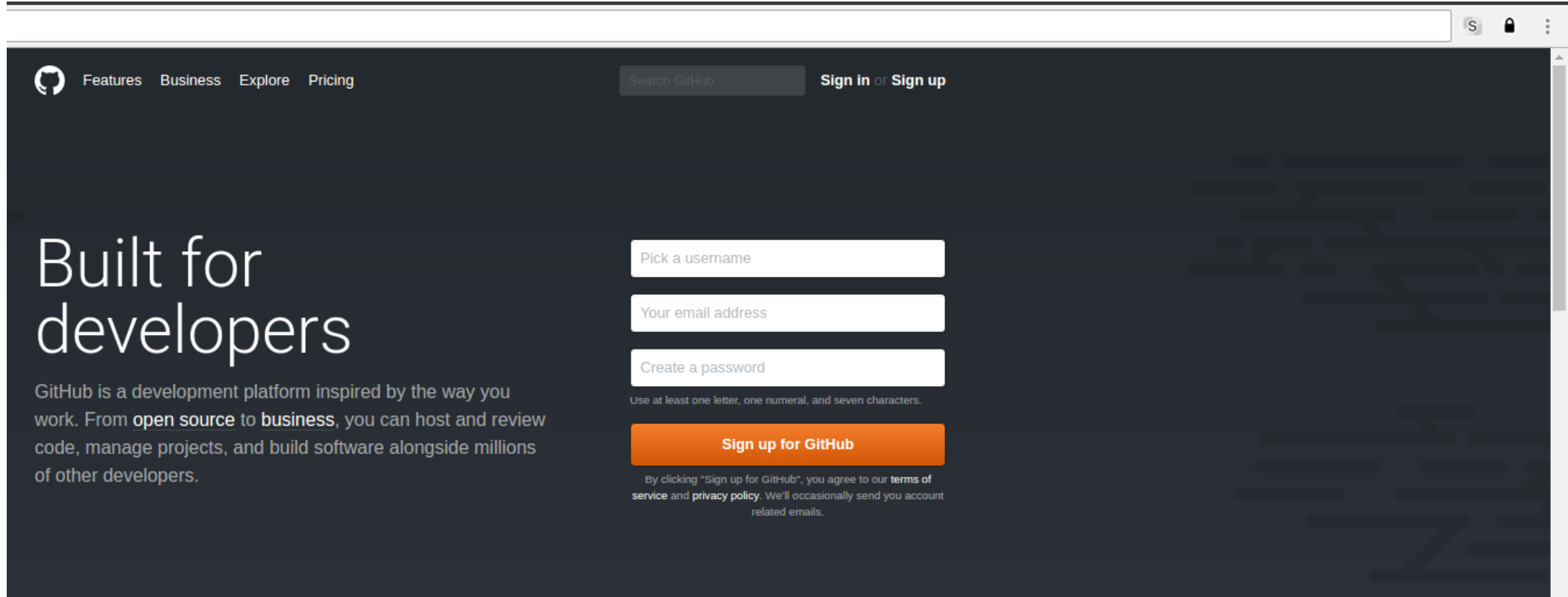
Requirements

- Password protection
 - a) Passwords can only be obtained through guessing
 - b) Offline guessing must be computationally infeasible
 - c) Online guessing must be throttled
- User awareness

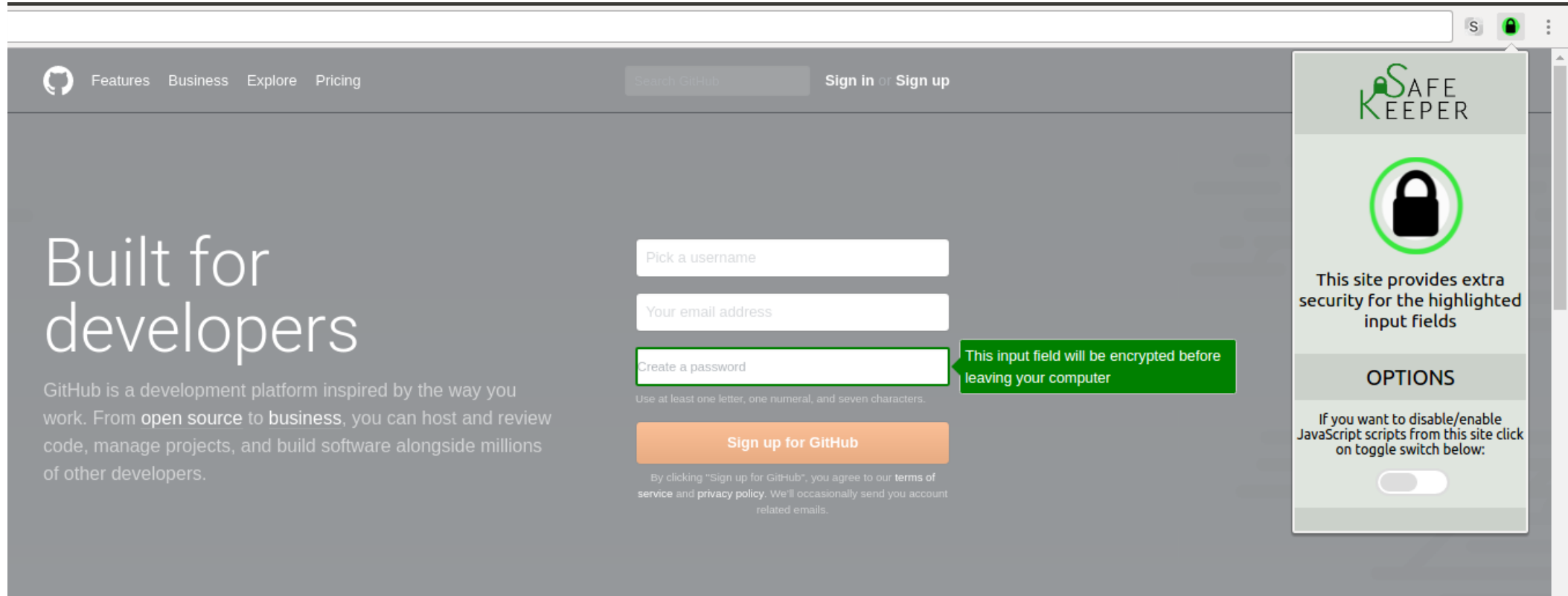
Design goals

- Minimal performance overhead
- Minimal software changes
- Ease of upgrade
- Backup and recovery

SafeKeeper Browser Add-on



SafeKeeper Browser Add-on



Can users use this effectively?

User study

86-participant on-site user study

Participants recruited using:

- Social media
- Email lists

Broad range of disciplines:

- Computer science
- MBA
- Design
- Consumer psychology
- ...

Age	
18 – 28 years	81%
29 – 38 years	19%

Highest qualification	
High school	9%
Bachelors	41%
Masters	34%
PhD	2%
Not specified	14%

Gender	
Male	72%
Female	28%

User study

“Does this website use SafeKeeper to protect your password?”

Main study

- 64 participants

Follow-up study

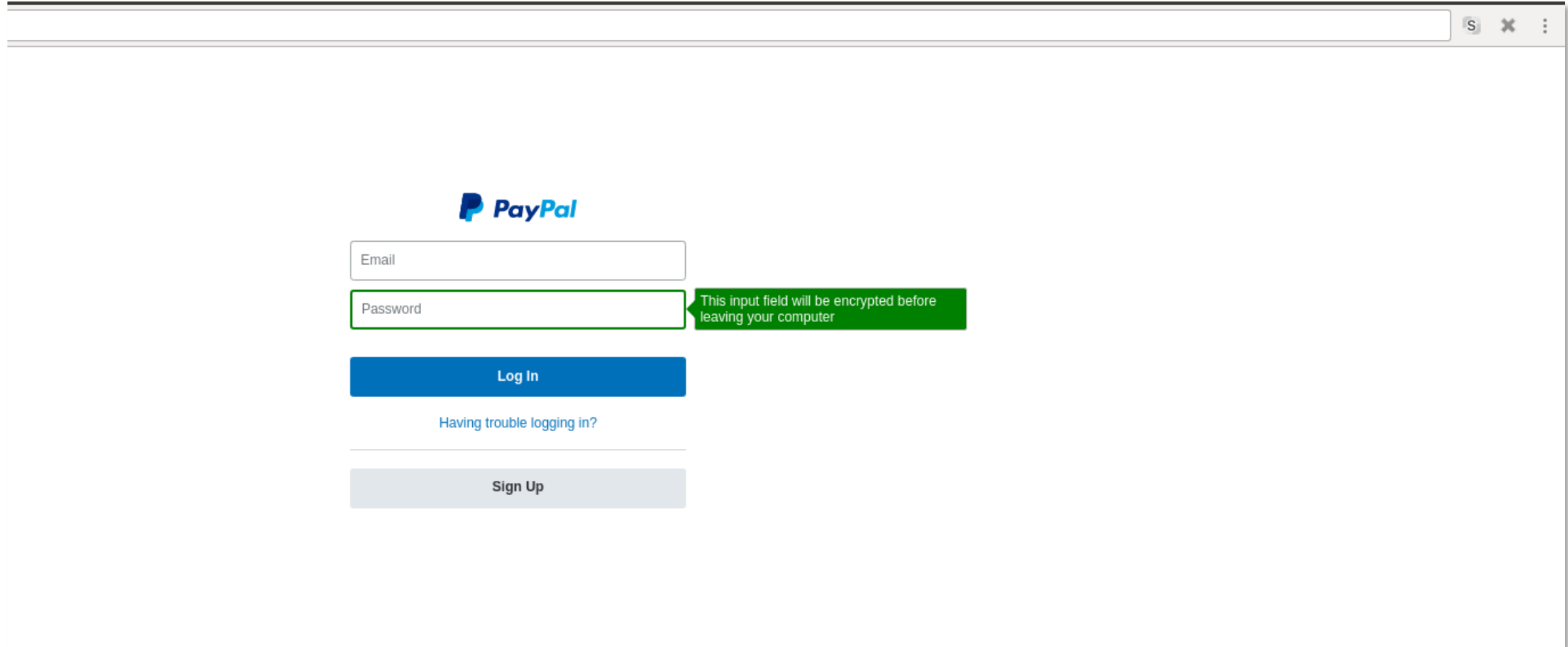
- 20 participants, 2 months later

Control group

- 22 participants, no instructions

#	Protected	Type of spoofing
6	No	None
7	No	Password field highlighted
3	No	Password field highlighted after time delay
4	Yes	None
5	Yes	Other fields highlighted

Attempted UI spoofing



Attempted UI Spoofing

The image shows a browser window displaying a Foursquare login page. The header is a solid blue bar. On the left, there is a search bar with the text "Espoo, FI" and a location pin icon. To the right of the search bar is a magnifying glass icon. Further right is a yellow "Sign Up" button. Below the search bar is a horizontal menu with several categories: "ending", "Food", "Coffee", "Nightlife", "Fun", and "Shopping", each with a corresponding icon. The main content area is a white box with the title "Log in to Foursquare". To the right of the title is a blue button with the Facebook logo and the text "Login with Facebook". Below the title are two input fields: "Email or Phone Number:" and "Password:". To the right of the password field is a link "(Forgot password?)". At the bottom of the white box is a green "Log in" button followed by the text "or Sign up for Foursquare".

Attempted UI Spoofing

The image shows a browser window displaying a Foursquare login page. The page has a blue header with a search bar containing "Espoo, FI" and a "Sign Up" button. Below the header is a navigation bar with icons for "ending", "Food", "Coffee", "Nightlife", "Fun", and "Shopping". The main content area features a white login form titled "Log in to Foursquare". The form includes a "Login with Facebook" button, an "Email or Phone Number:" label, a text input field, a "Password:" label, a "(Forgot password?)" link, another text input field, and a "Log in" button followed by "or Sign up for Foursquare". A green callout box with a white border points to the email input field, containing the text: "This input field will be encrypted before leaving your computer".

ending Food Coffee Nightlife Fun Shopping

Log in to Foursquare [Login with Facebook](#)

Email or Phone Number:

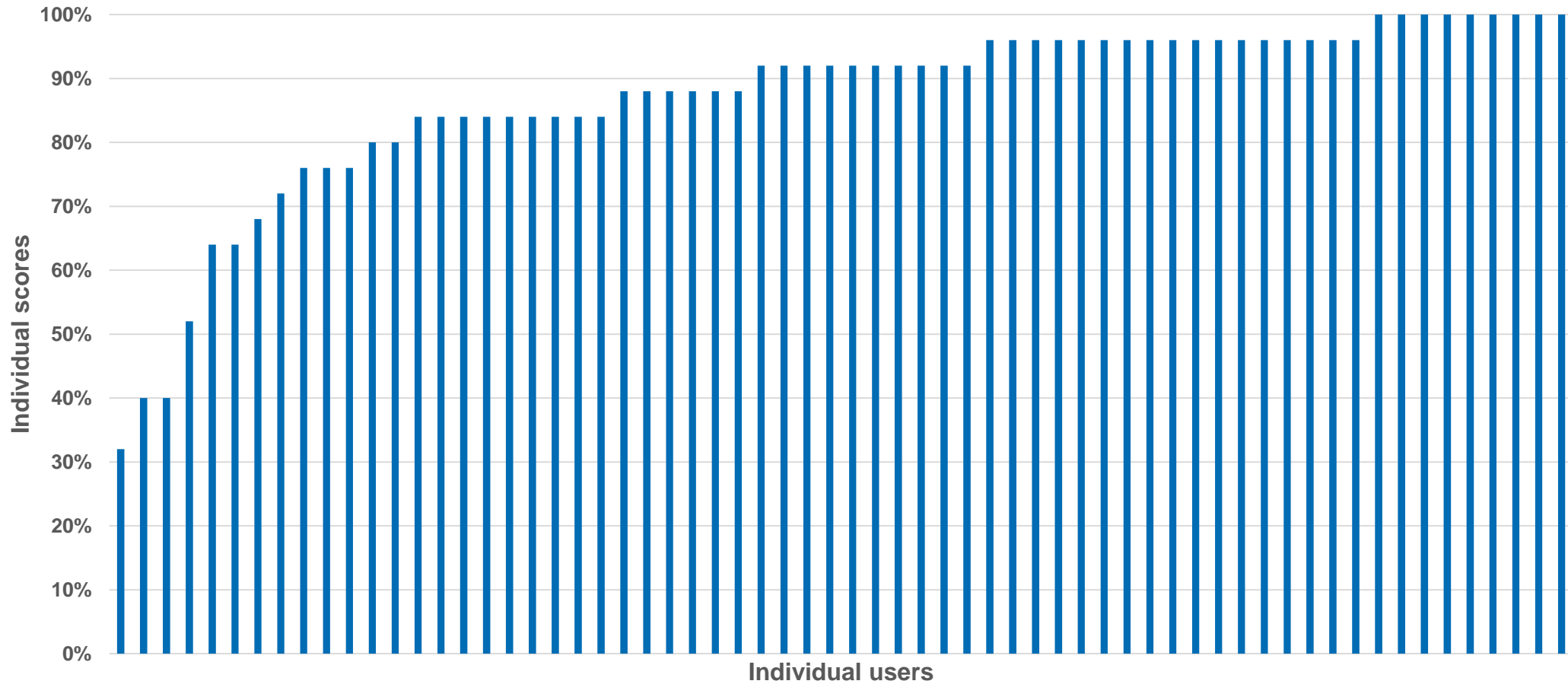
Password: (Forgot password?)

Log in or Sign up for Foursquare

This input field will be encrypted before leaving your computer

Main study results

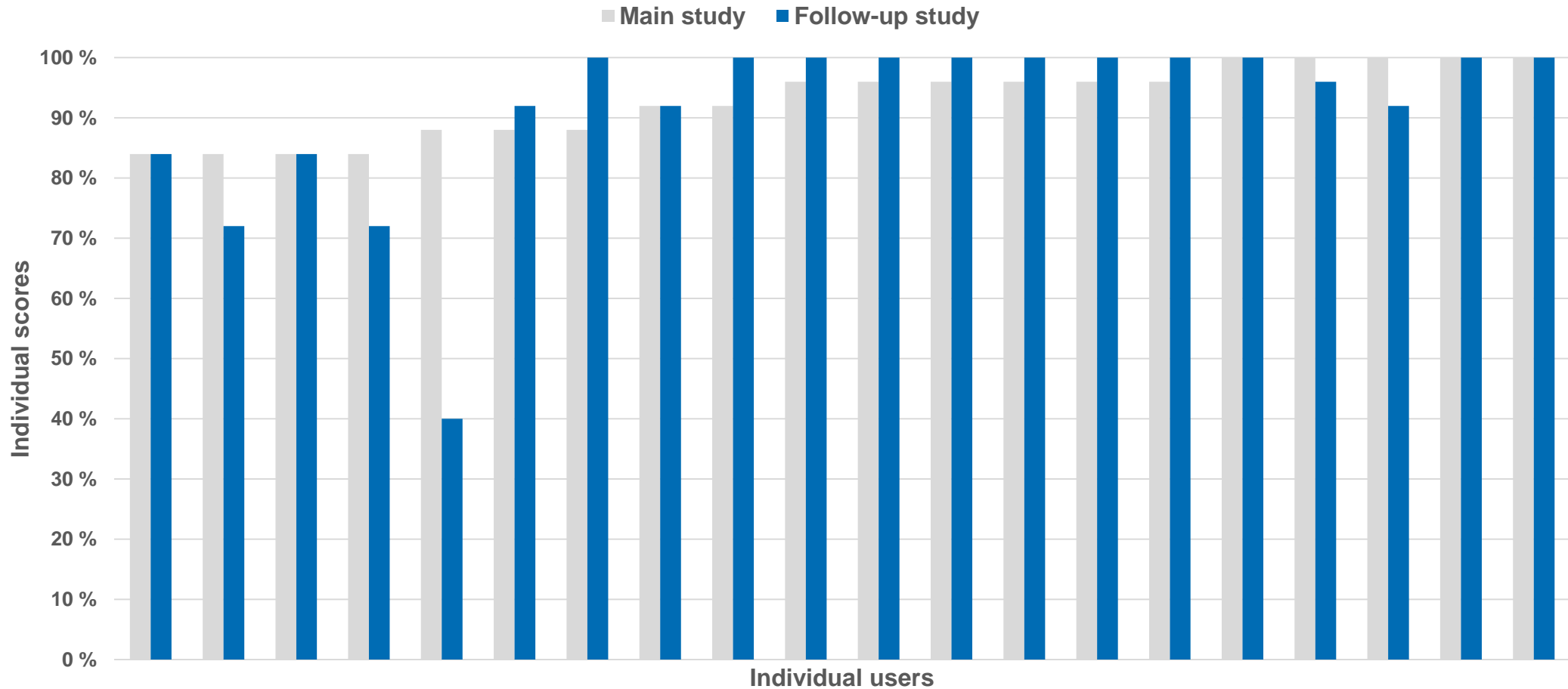
Average effectiveness: 87%



Follow-up study results

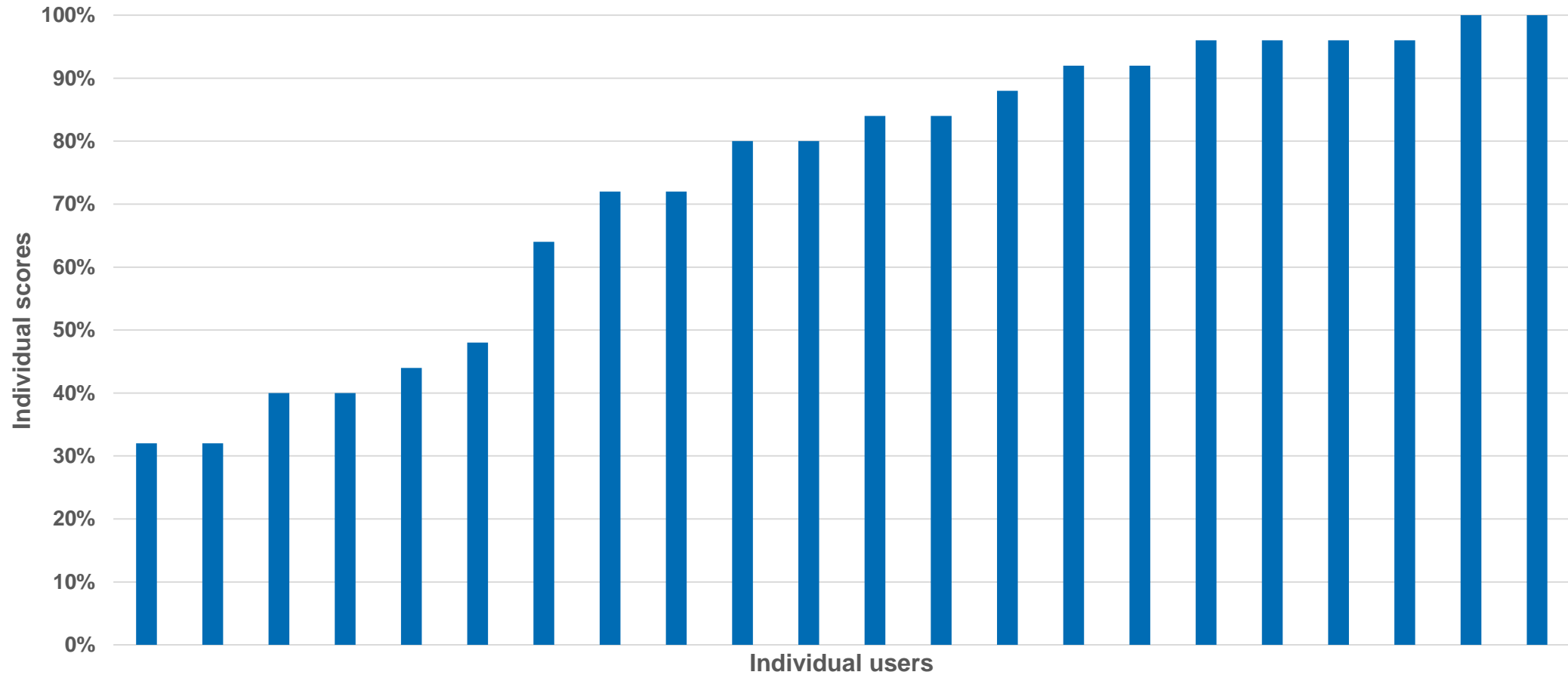
Main study effectiveness: 93%

Follow-up study effectiveness: 91%

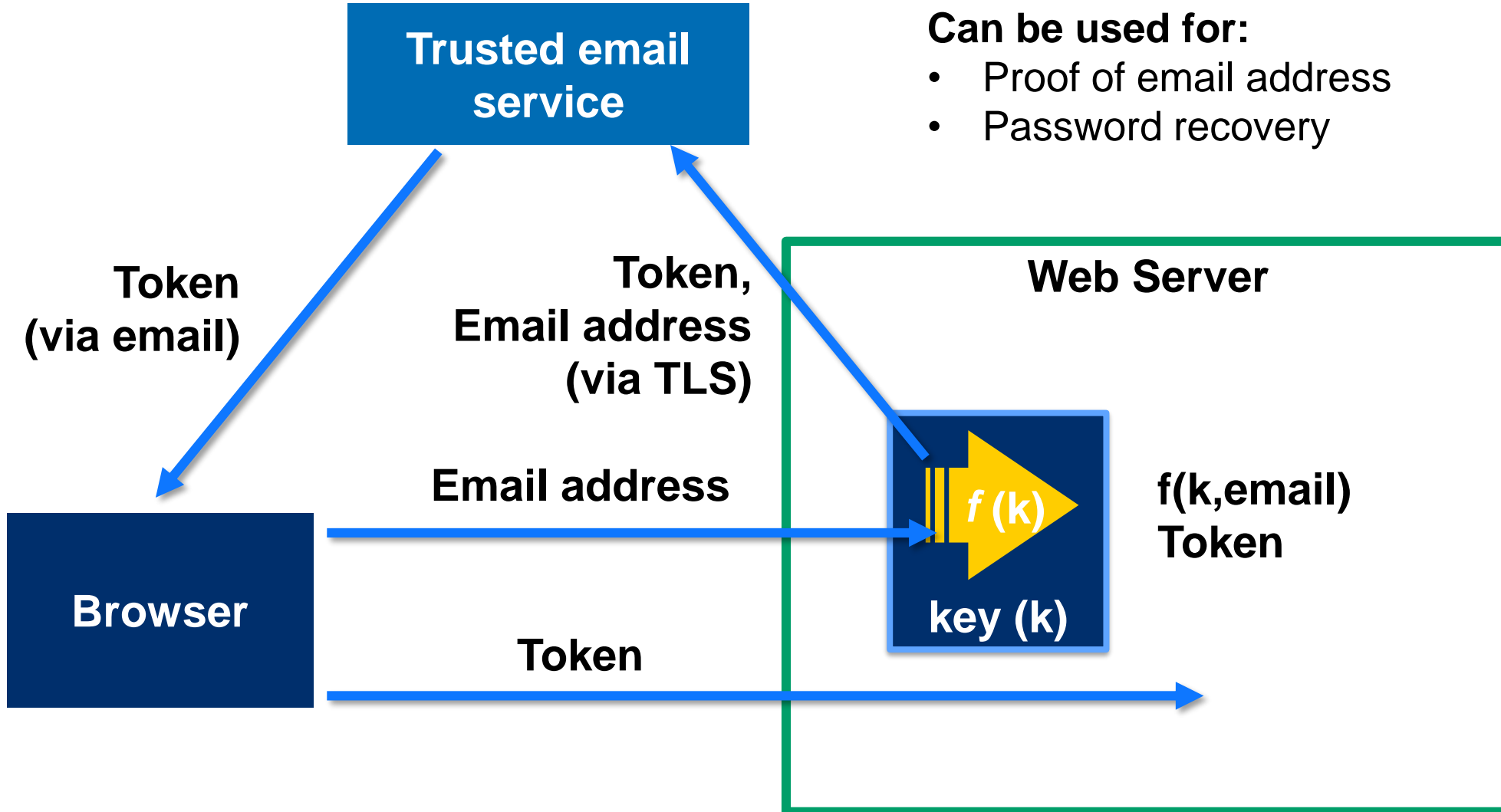


Control group results

Control group effectiveness: 74%



Future work: Protecting email addresses



Conclusions

- TEEs can help to protect password databases
- Can be integrated into existing systems
- Can achieve web-scale performance
- Can protect real users
- Potential for future work

<https://ssg.aalto.fi/projects/passwords/>

