MFA:
Multifactor Authentication

2FA vs MFA

- **2FA** Two-Factor authentication
  - Something you **know** (password)
  - Something you **have** (smartphone or security key)

- search "`google titan`"
- search "`yubico security key`"

- We will consider 2FA and **MFA** synonyms

Second factors

- **Smartphone**
  - OTP SMS
  - OTP Authenticator App
  - Push notifications
- **SecurityKey** (USB/NFC/Bluetooth)

- SecurityKey **much more secure** than other methods
- **All enormously** more secure than password only

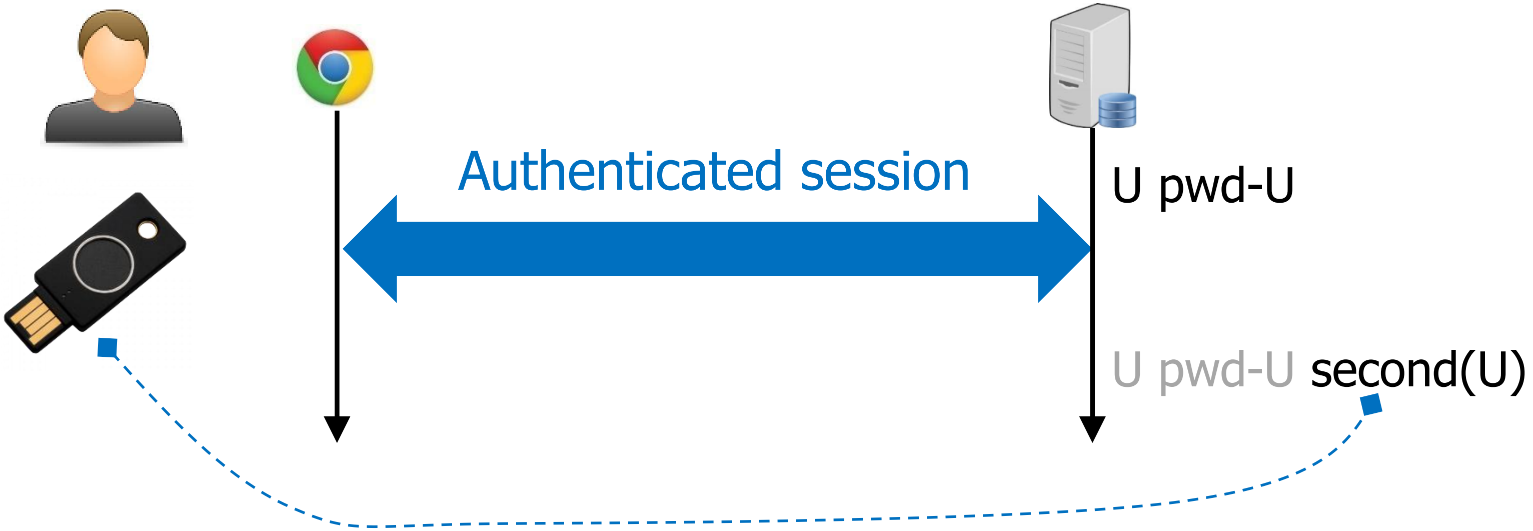
Our Focus

- **Web app** (BASIC/FORM over HTTPS)
- **Different** organizations
- Extremely relevant in practice

- Very useful also for
  - Authentication for "legacy" protocols (SMTP, POP, SSH, ...)
  - Same organization

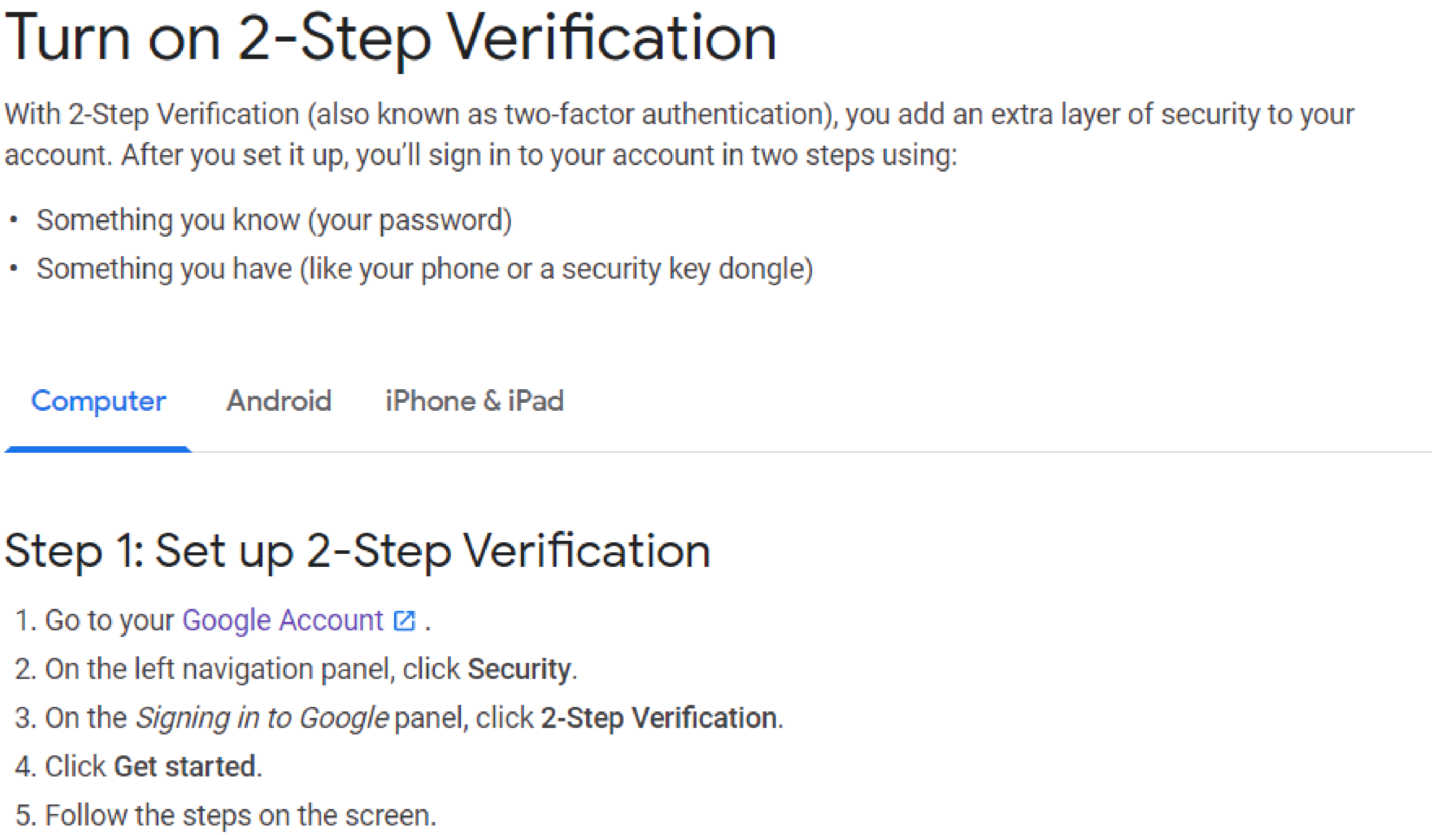
  - Much less used (very difficult to deploy on legacy applications)

Step 1: Linking 2nd factor (once)



- User selects 2FA in account options (Service must support 2FA)
- Message exchange depends on Service and 2FA technique

Linking Example (I): Google



Turn on 2-Step Verification

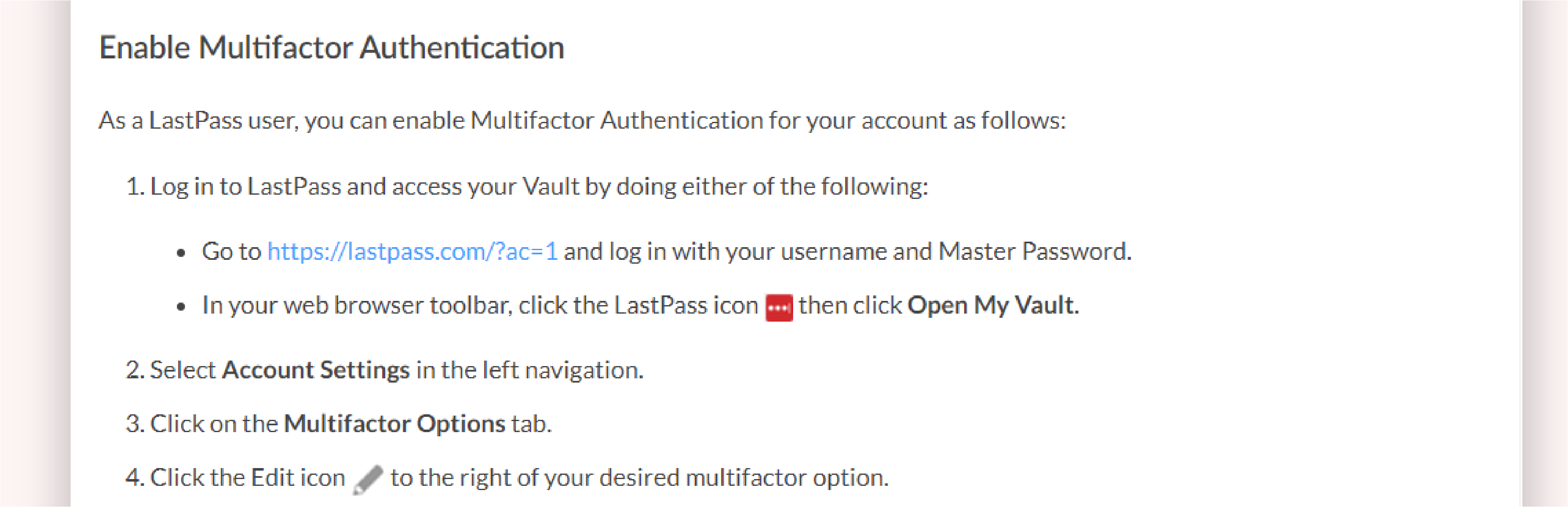
With 2-Step Verification (also known as two-factor authentication), you add an extra layer of security to your account. After you set it up, you'll sign in to your account in two steps using:

- Something you know (your password)
- Something you have (like your phone or a security key dongle)

Step 1: Set up 2-Step Verification

1. Go to your Google Account .
2. On the left navigation panel, click **Security**.
3. On the *Signing in to Google* panel, click **2-Step Verification**.
4. Click **Get started**.
5. Follow the steps on the screen.

Linking Example (II): Lastpass

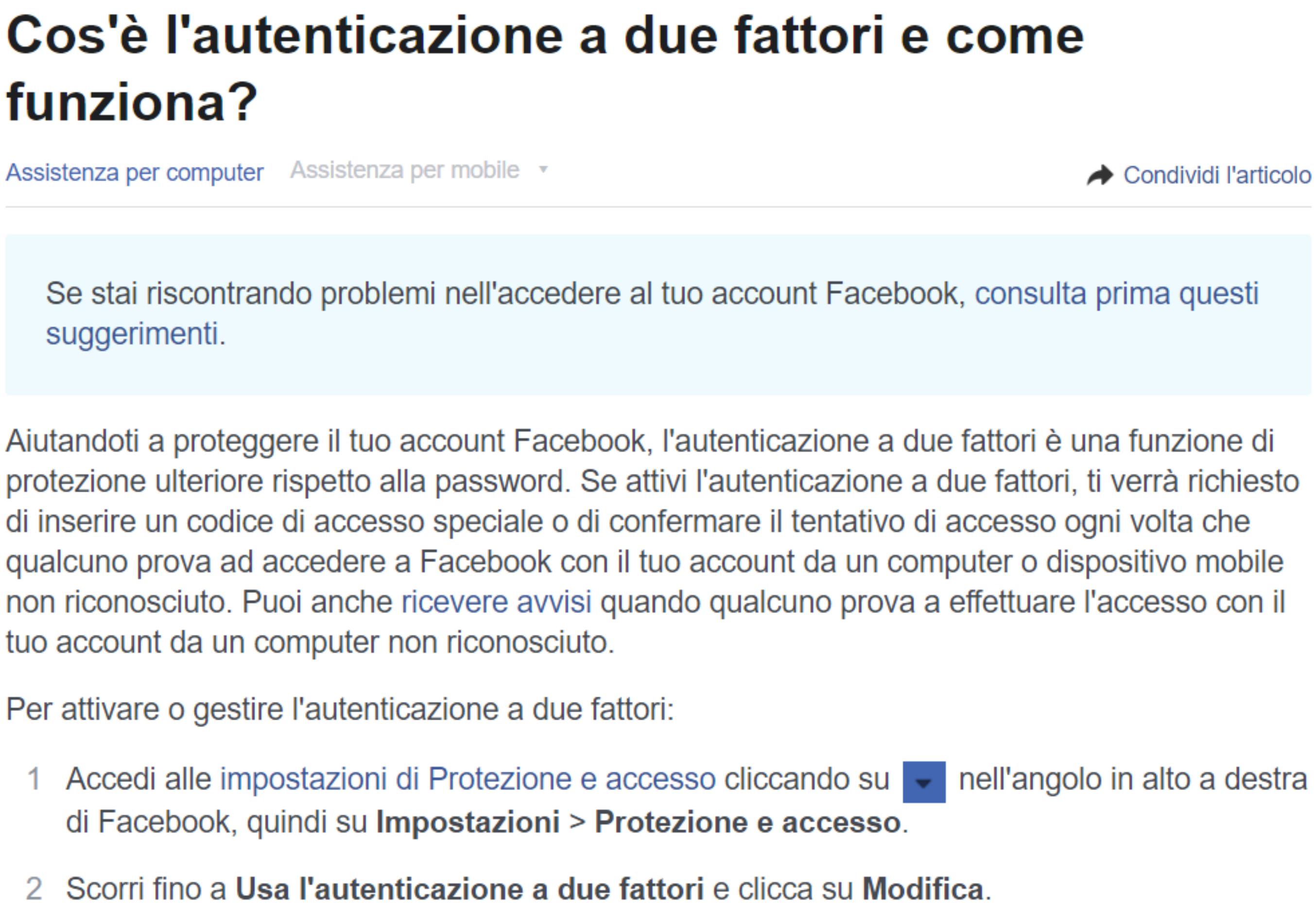


Enable Multifactor Authentication

As a LastPass user, you can enable Multifactor Authentication for your account as follows:

1. Log in to LastPass and access your Vault by doing either of the following:
   - Go to https://lastpass.com/?ac=1 and log in with your username and Master Password.
   - In your web browser toolbar, click the LastPass icon ••• then click **Open My Vault.**
2. Select **Account Settings** in the left navigation.
3. Click on the **Multifactor Options** tab.
4. Click the Edit icon ✎ to the right of your desired multifactor option.

Linking Example (III): Facebook



Cos'è l'autenticazione a due fattori e come funziona?

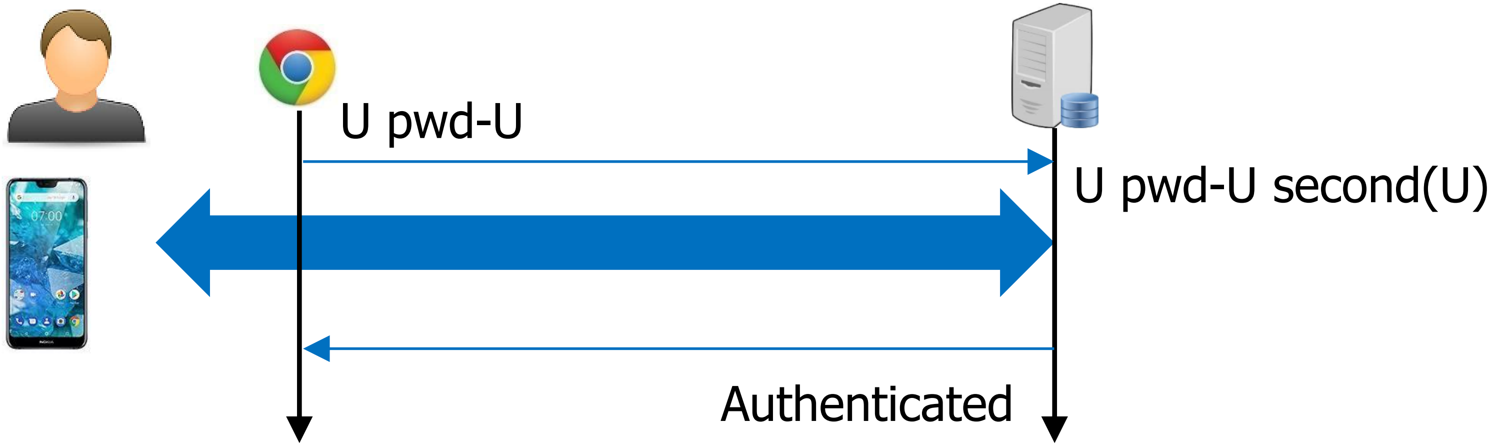
Se stai riscontrando problemi nell'accedere al tuo account Facebook, consulta prima questi suggerimenti.

Aiutandoti a proteggere il tuo account Facebook, l'autenticazione a due fattori è una funzione di protezione ulteriore rispetto alla password. Se attivi l'autenticazione a due fattori, ti verrà richiesto di inserire un codice di accesso speciale o di confermare il tentativo di accesso ogni volta che qualcuno prova ad accedere a Facebook con il tuo account da un computer o dispositivo mobile non riconosciuto. Puoi anche ricevere avvisi quando qualcuno prova a effettuare l'accesso con il tuo account da un computer non riconosciuto.

Per attivare o gestire l'autenticazione a due fattori:

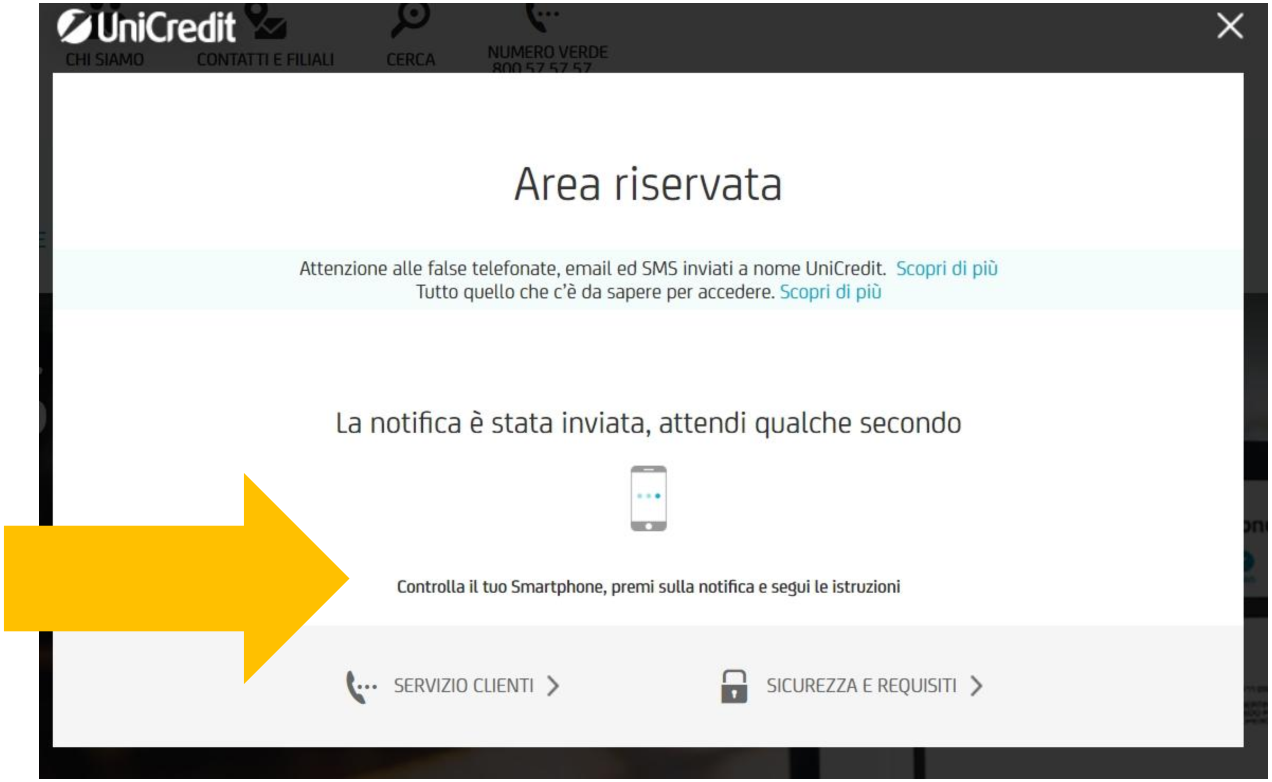
1. Accedi alle impostazioni di Protezione e accesso cliccando su ▾ nell'angolo in alto a destra di Facebook, quindi su **Impostazioni** > **Protezione e accesso**.
2. Scorri fino a **Usa l'autenticazione a due fattori** e clicca su **Modifica**.

Step 2:
Login (every time)



- Message exchange depends on Service and 2FA technique

Login Example (I): Unicredit



Area riservata

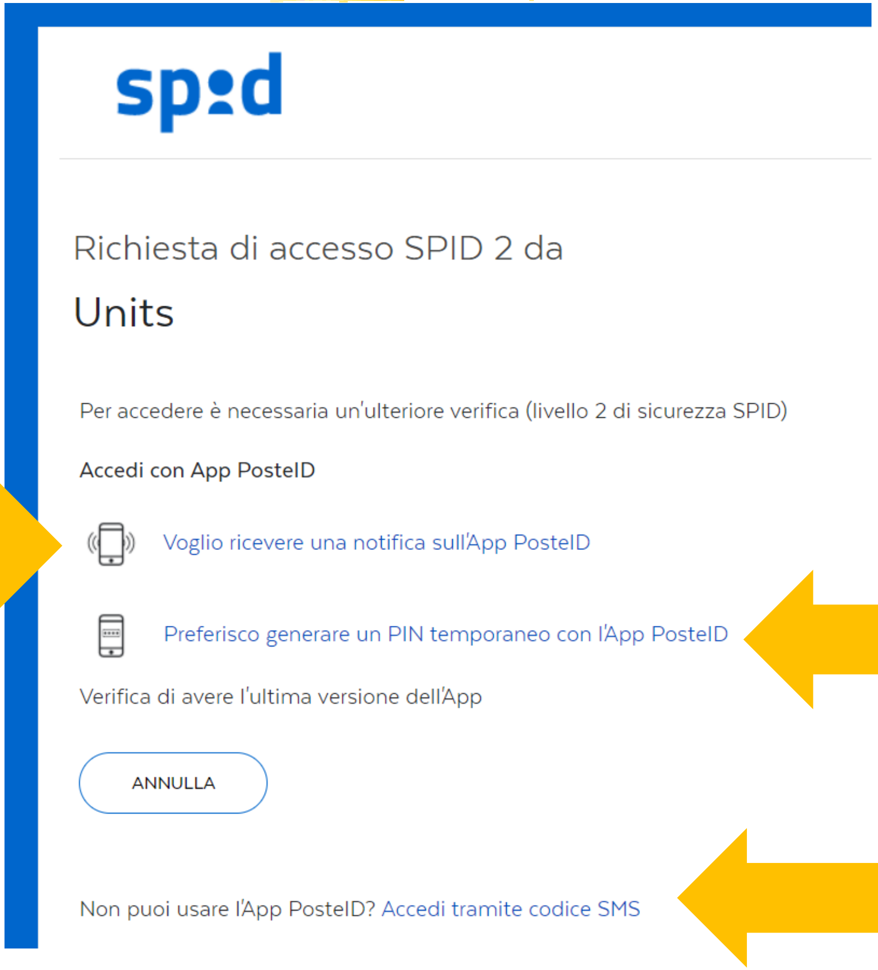
Attenzione alle false telefonate, email ed SMS inviati a nome UniCredit. Scopri di più
Tutto quello che c'è da sapere per accedere. Scopri di più

La notifica è stata inviata, attendi qualche secondo

Controlla il tuo Smartphone, premi sulla notifica e segui le istruzioni

Push Notification

Login Example (II): SPID-PosteID



spid

Richiesta di accesso SPID 2 da

Units

Per accedere è necessaria un'ulteriore verifica (livello 2 di sicurezza SPID)

Accedi con App PosteID

Voglio ricevere una notifica sull'App PosteID

Preferisco generare un PIN temporaneo con l'App PosteID

Verifica di avere l'ultima versione dell'App

Non puoi usare l'App PosteID? Accedi tramite codice SMS

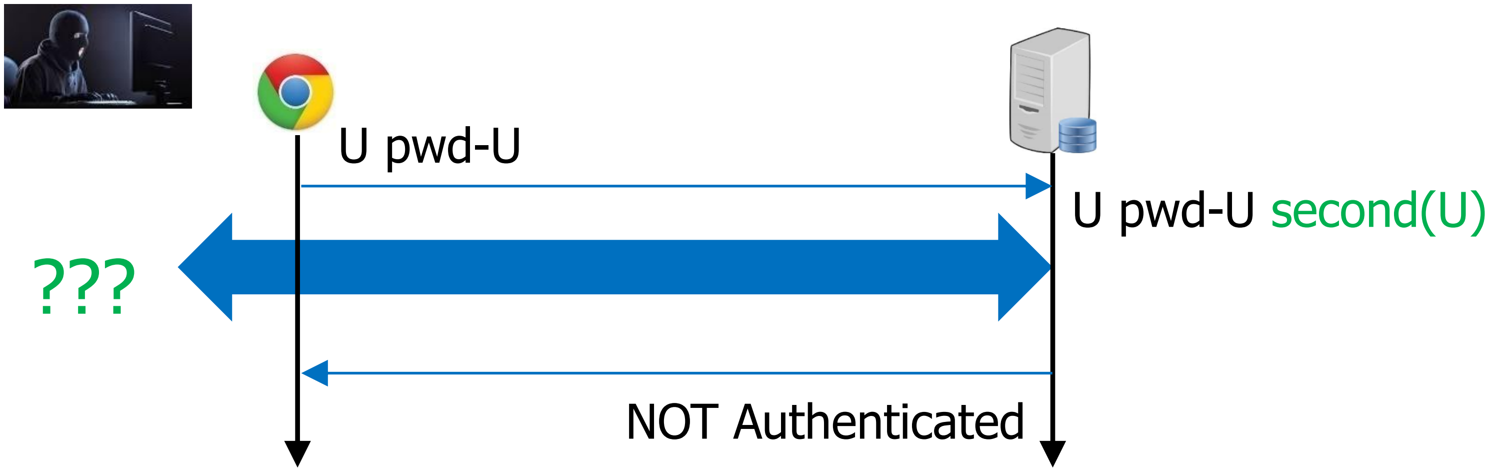
Push Notification

OTP AuthenticatorApp

OTP SMS

Threat Model: Stolen Password (I)

- Adversary has <U,P>



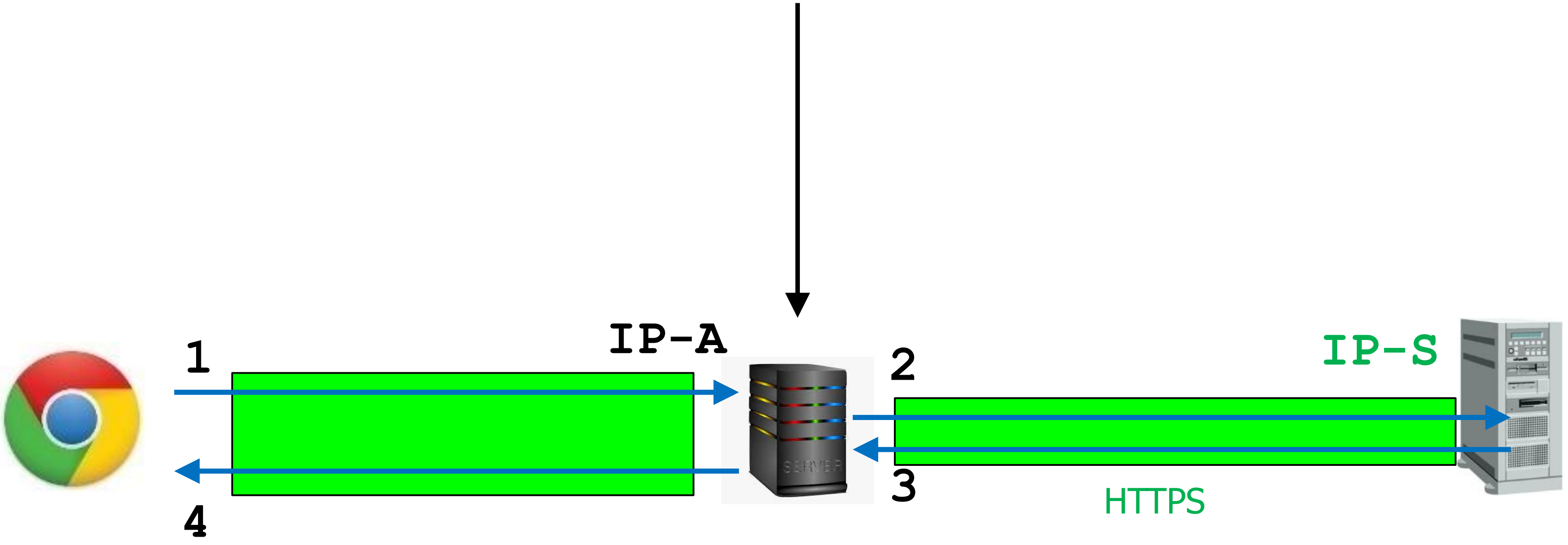
Threat Model: Stolen Password (II)

❑ **Smartphone**	
❑ OTP SMS	Solved
❑ OTP Authenticator App	Solved
❑ Push notifications	Solved
❑ **SecurityKey** (USB/NFC/Bluetooth)	Solved

Evil Proxy

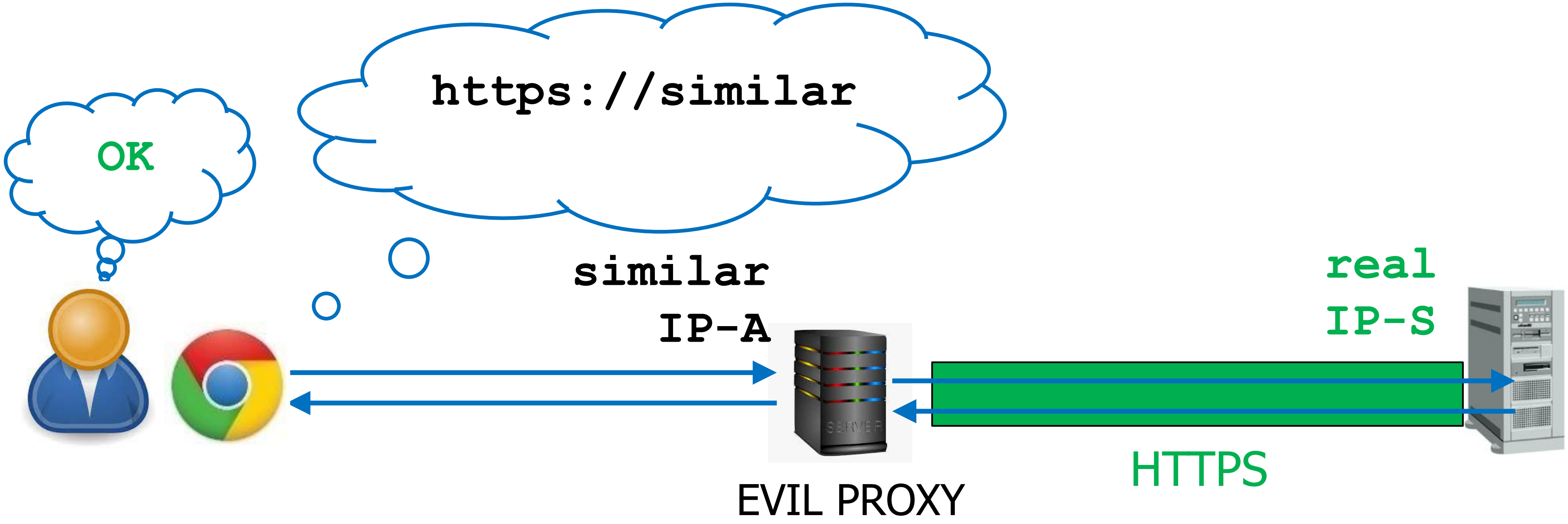
Proxy **specialized** for **AitM**

- ❑ Presents to C **all** resources of target website **without any local copy**
- ❑ Can target **many different** websites at the same time
- ❑ **Configuration** specifies what to **modify** and what to **log**



Threat Model: "Real-time Phishing" (I)

- User does **not** detect that is accessing the **wrong** URL



Threat Model: "Real-time Phishing" (II)

□ **Smartphone**	
□ OTP SMS	Not Solved
□ OTP Authenticator App	Not Solved
□ Push notifications	Not Solved
□ **SecurityKey** (USB/NFC/Bluetooth)	Solved

Keep in mind

MFA is extremely important

- Very effective for very realistic threat model
- Enabling 2FA is a very high priority defensive investment

MFA is essential … Use of anything beyond the password significantly increases the costs for attackers, which is why **the rate of compromise of accounts using any type of MFA is less than 0.1% of the general population**.

Alex Weinert, Director of Identity Security Microsoft
November 2020

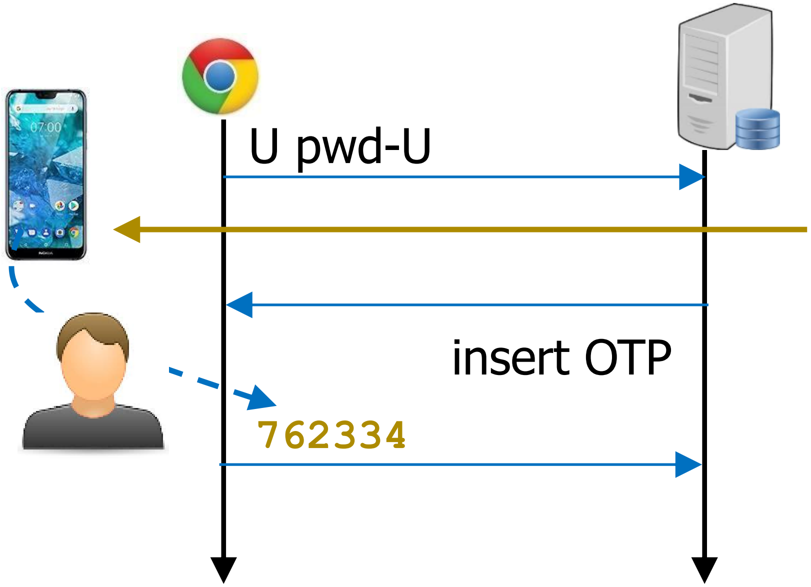
One-Time Passwords (OTP)

Second factor **(REMIND)**

- **Smartphone**
  - OTP SMS
  - OTP Authenticator App
  - Push notifications
- **SecurityKey** (USB/NFC/Bluetooth)

- SecurityKey **much more secure** than other methods
- **All enormously** more secure than password only

OTP SMS



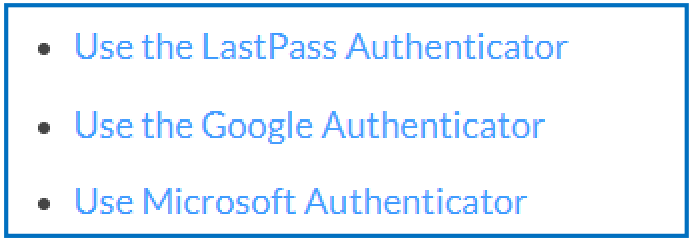
One-Time Password (**OTP**):

- 6-digit code with uniform distribution in [0,999999]
- Valid only for **this** login attempt

OTP AuthApp: Linking

- **Generic** "Authenticator App" installed on User smartphone

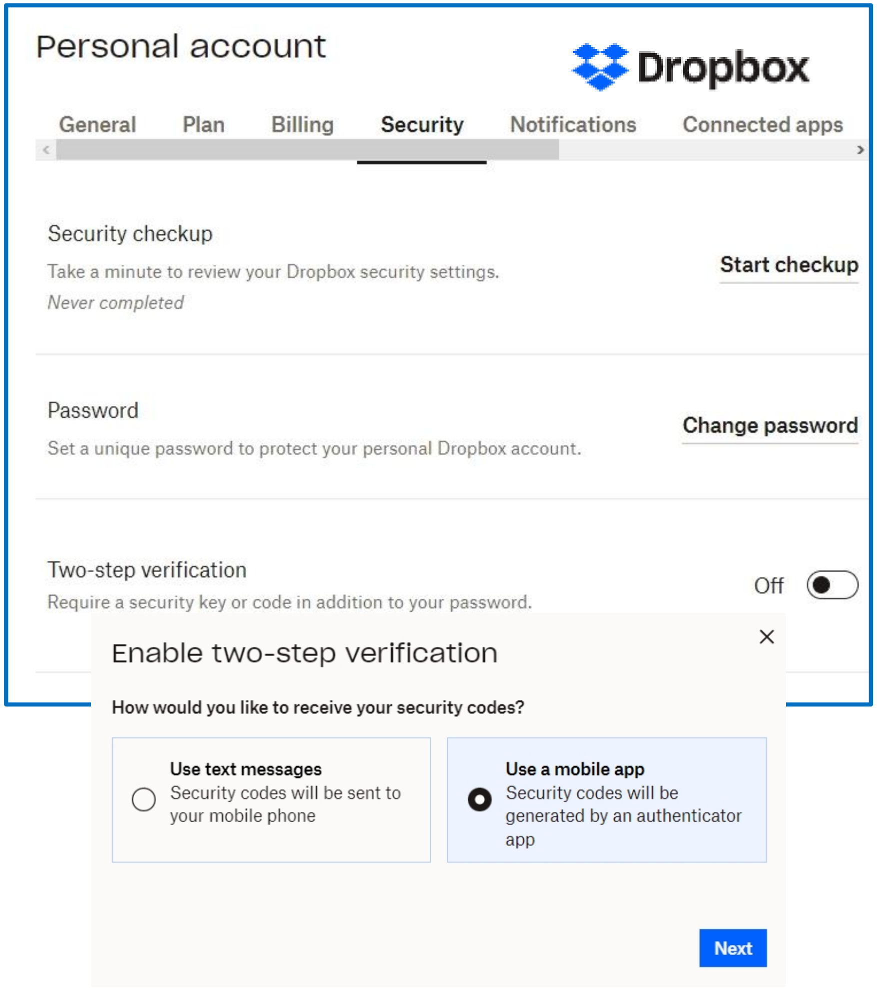
- User:
  1. Enable 2FA
  2. Choose AuthApp and Install on smartphone (if not installed already)



- Use the LastPass Authenticator
- Use the Google Authenticator
- Use Microsoft Authenticator

3. Link account at Service with AuthApp on smartphone

Example (outline)



Personal account

Dropbox

General | Plan | Billing | **Security** | Notifications | Connected apps

Security checkup

Take a minute to review your Dropbox security settings.

Never completed

Start checkup

Password

Set a unique password to protect your personal Dropbox account.

Change password

Two-step verification

Require a security key or code in addition to your password.

Off

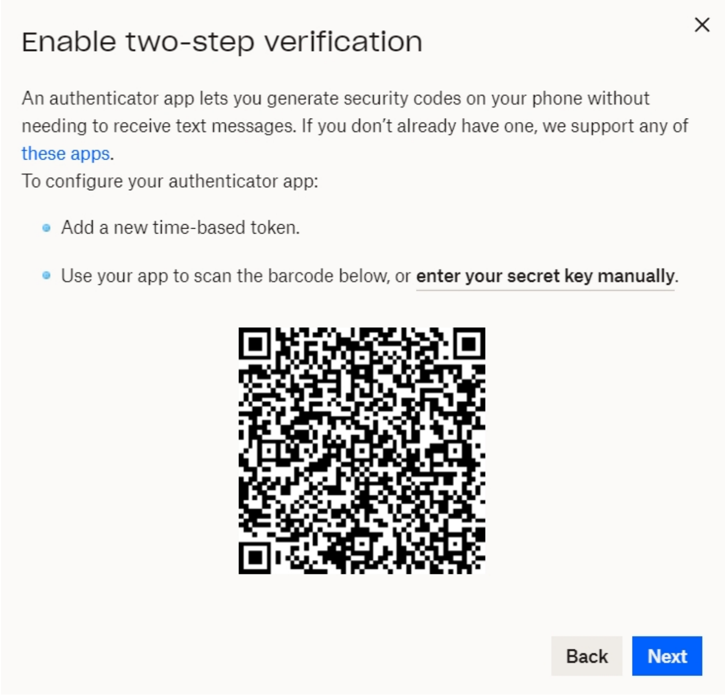
Enable two-step verification

How would you like to receive your security codes?

○ **Use text messages** Security codes will be sent to your mobile phone

● **Use a mobile app** Security codes will be generated by an authenticator app

Next



Enable two-step verification

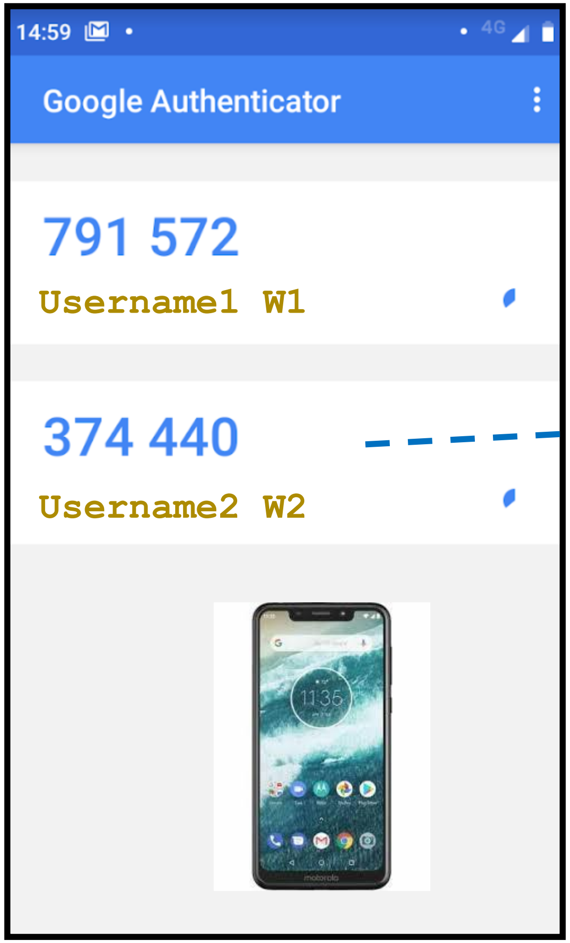
An authenticator app lets you generate security codes on your phone without needing to receive text messages. If you don't already have one, we support any of these apps.

To configure your authenticator app:

- Add a new time-based token.
- Use your app to scan the barcode below, or **enter your secret key manually**.

Back | Next

OTP AuthApp: Login



14:59

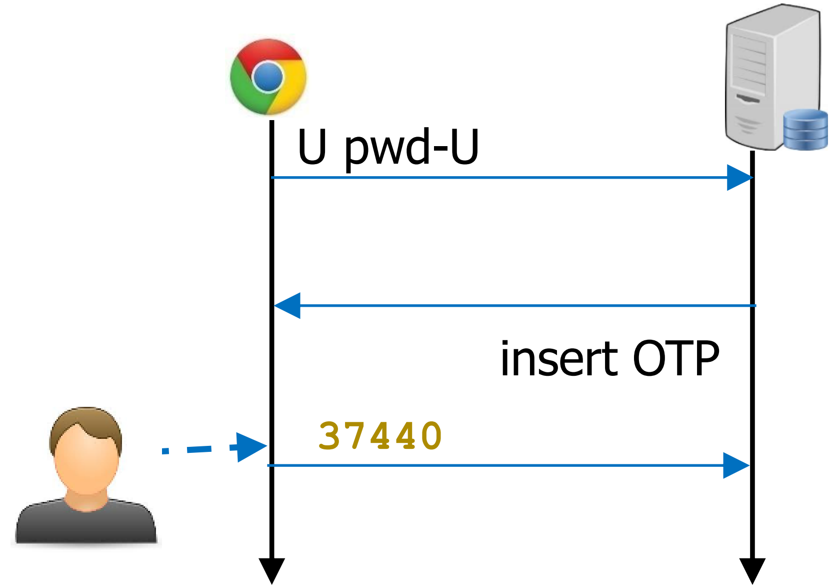
Google Authenticator

791 572

Username1 W1

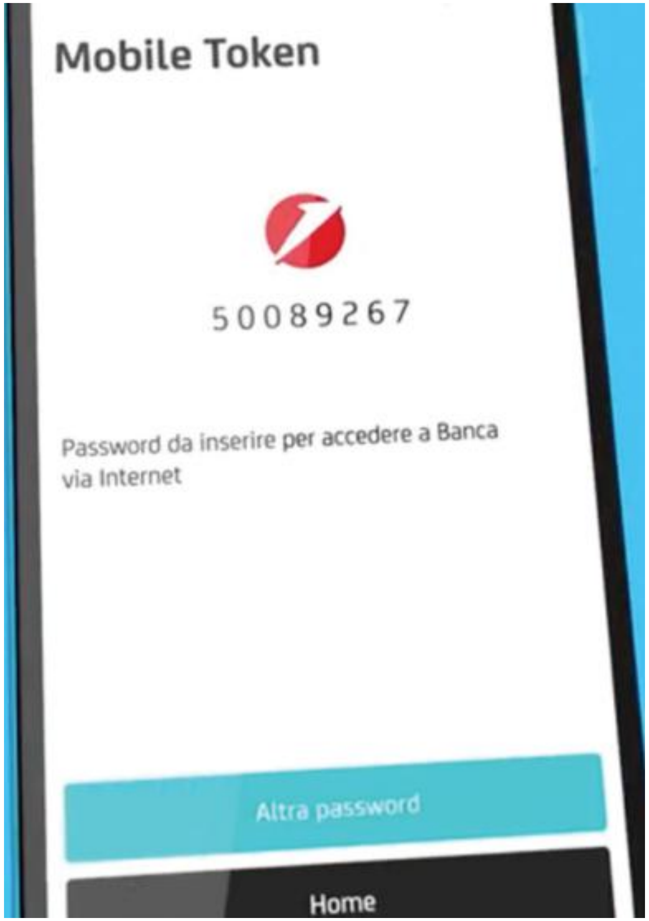
374 440

Username2 W2



Remark

- A Service might use only "its own" Authenticator App



Mobile Token

50089267

Password da inserire per accedere a Banca via Internet

- Conceptually identical

OTP AuthApp: Implementation

AuthApp Linking: Requirement (I)

U1, K1

Dropbox

U1, K1

- Private key K1
  - Generated by service
  - Securely sent to AuthApp upon activation

AuthApp Linking: Requirement (II)

U1, K1
U2, K2
U3, K3

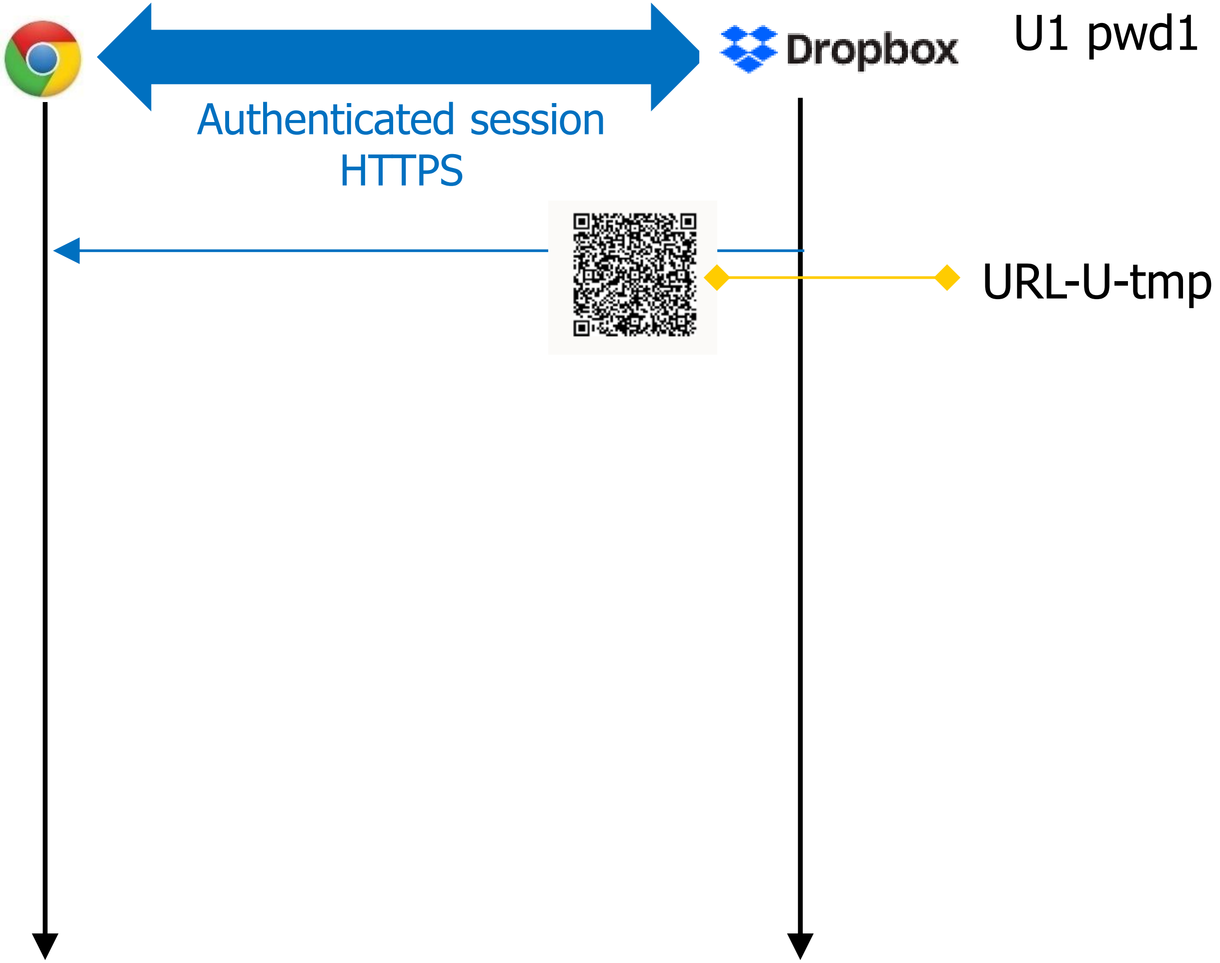
AuthApp usually linked to **several** services

U1, K1 Dropbox

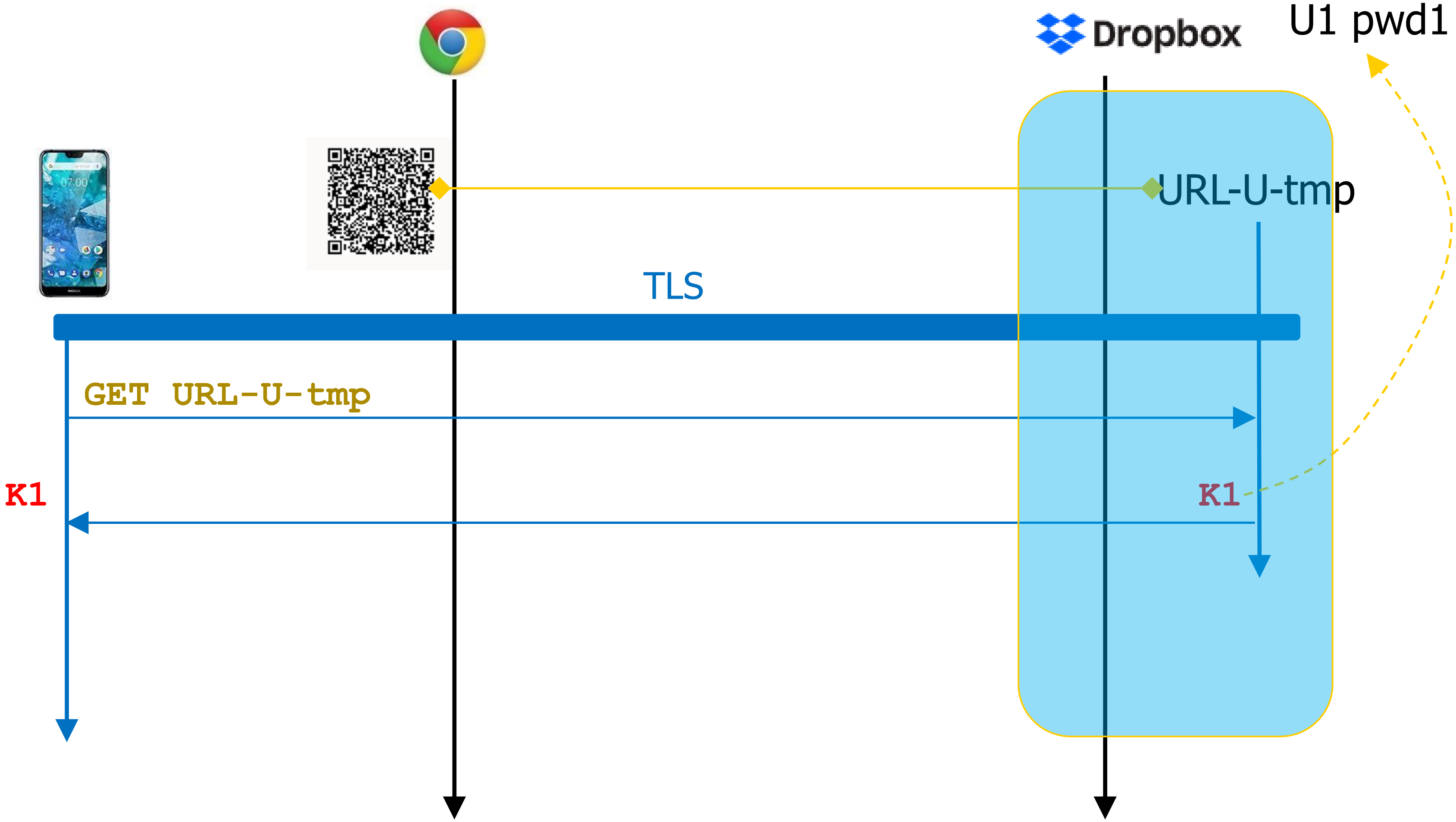
U2, K2

U3, K3 LastPass•••|

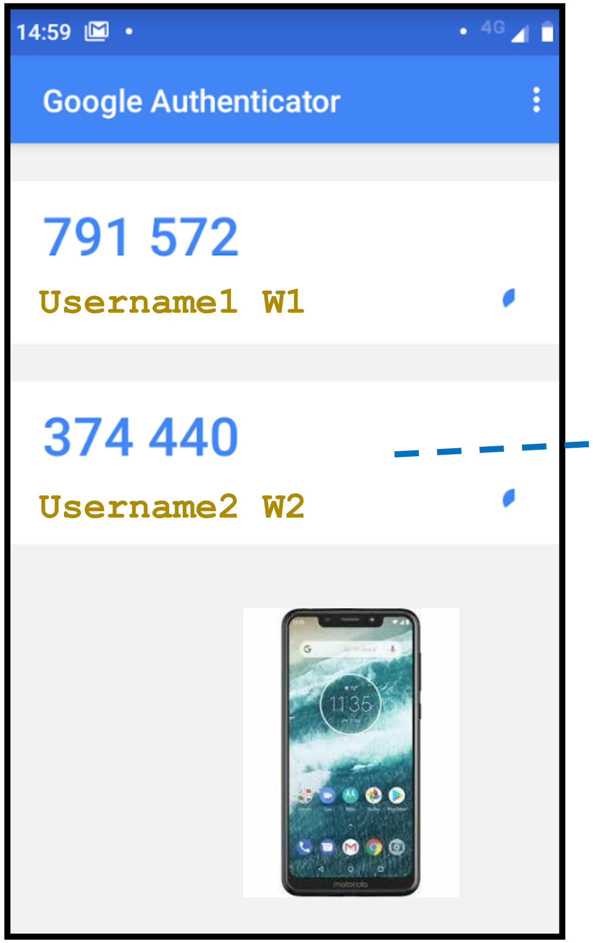
AuthApp Linking (I)



AuthApp Linking (II)



AuthApp Login: Requirement



14:59

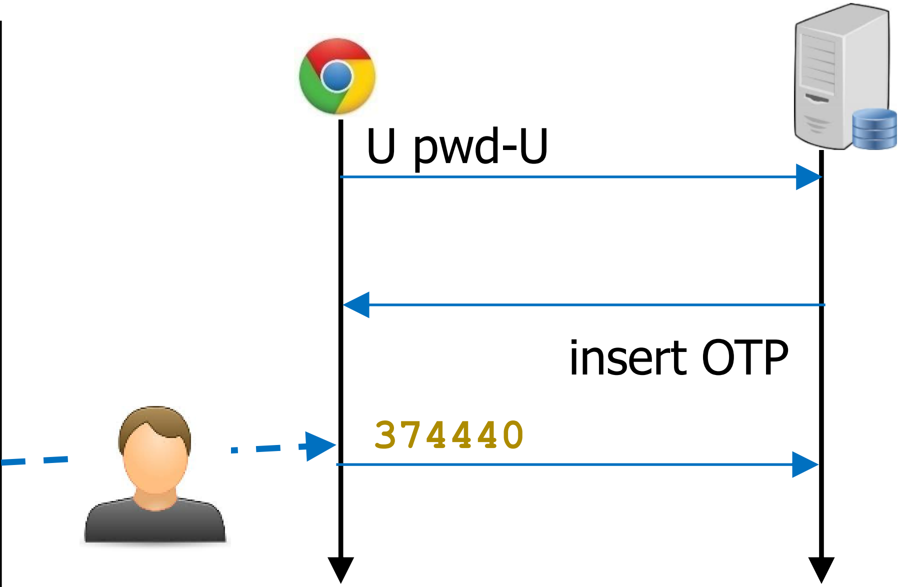
Google Authenticator

791 572

Username1 W1

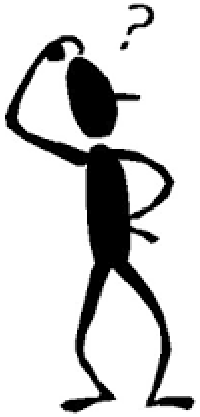
374 440

Username2 W2



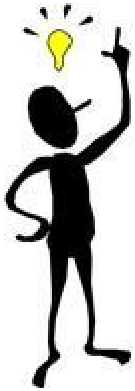
No communication Service - AuthApp

how can they agree?

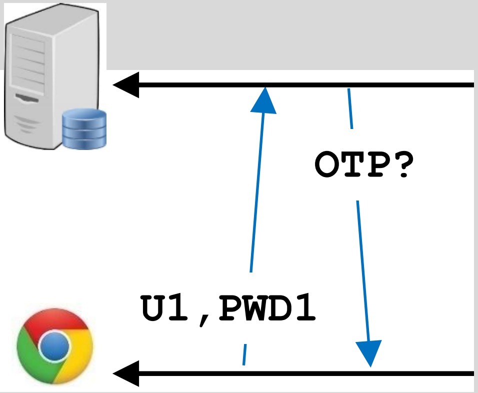


TOTP (Time-Based OTP): Basic Idea

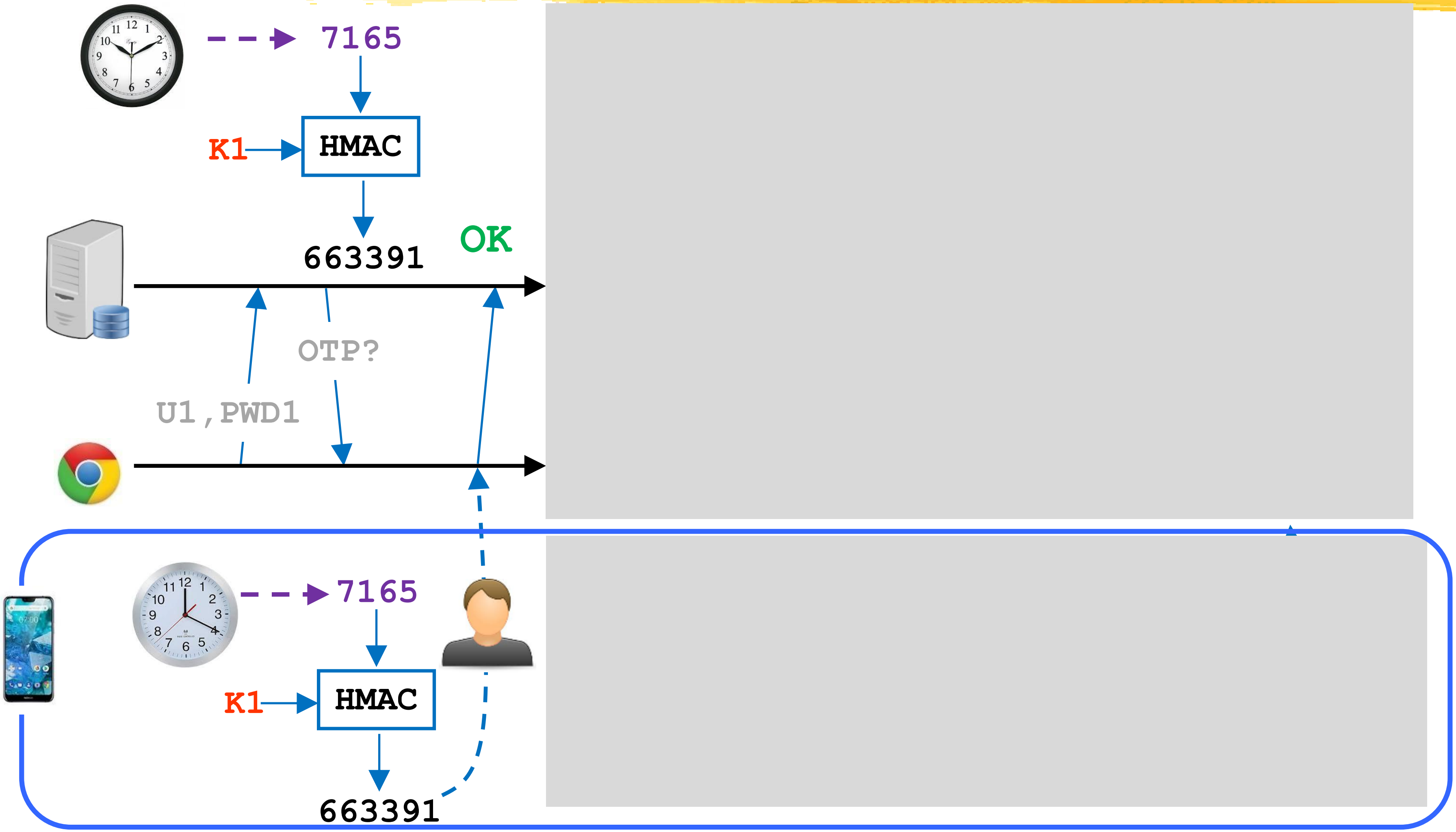
- OTP(t) = ENCRYPT$_K$(t - T0) *// T0 conventional zero time*
- Read the clock and encrypt the result



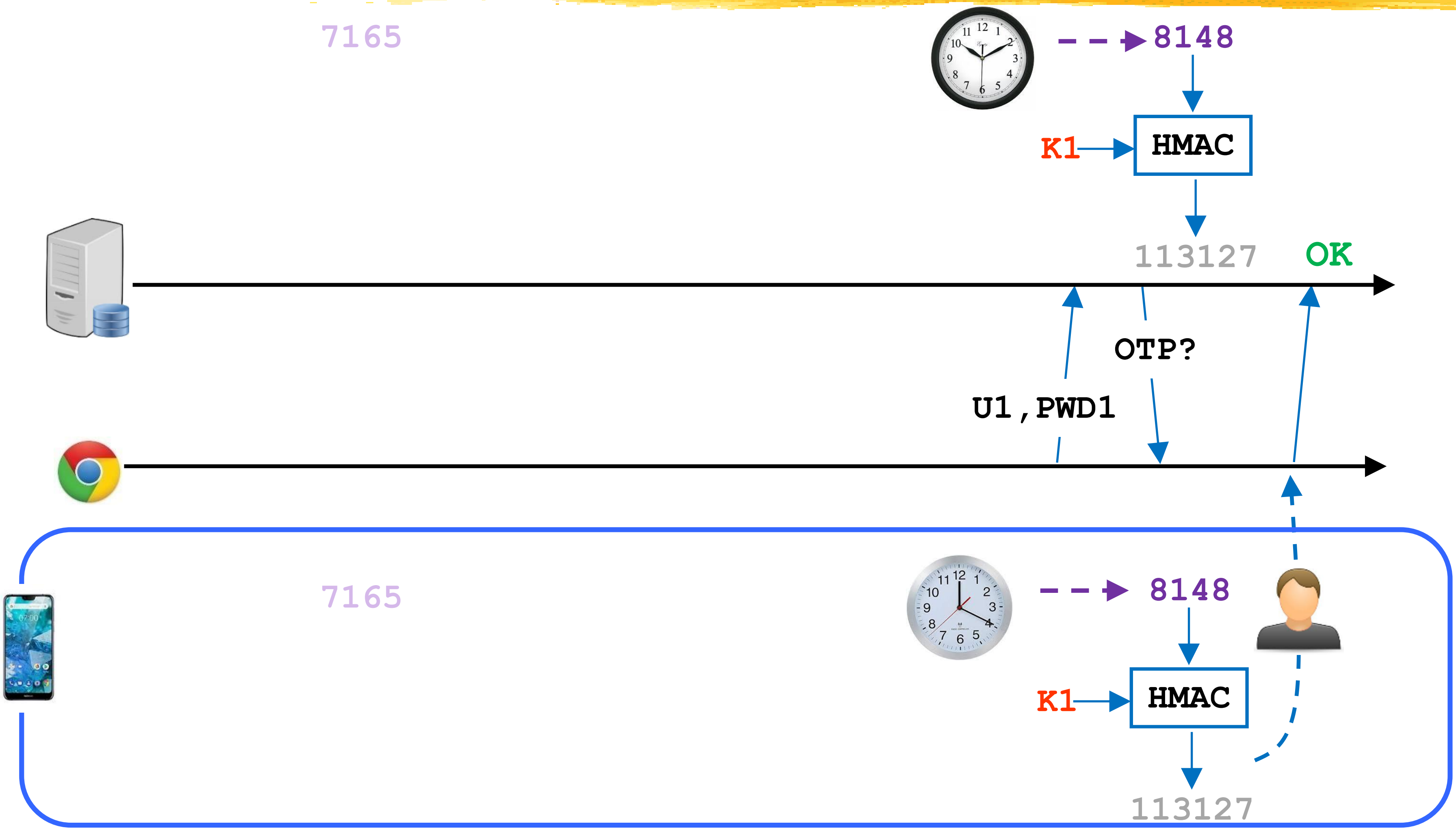
Example (I)



Example (II)



Example (III)


7165
8148
K1
HMAC
113127
OK
OTP?
U1,PWD1
7165
8148
K1
HMAC
113127

TOTP (Time-Based OTP): Some details

- Clocks cannot be perfectly synchronized
- Messages have latency

- OTP(**t**) = ENCRYPT$_K$(**t** - T0) **cannot work in practice**

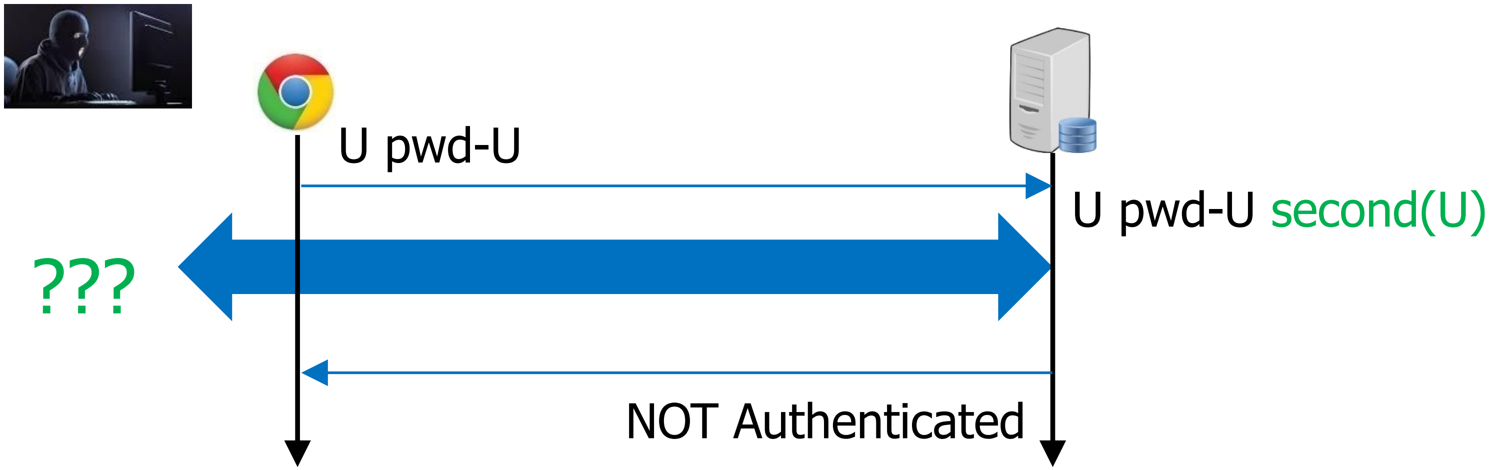
- OTP(**t**) = ENCRYPT$_K$ (**t** - T0 **/ DX**) // DX = 30 s
- One-time password changes every 30 seconds

- Actual algorithm more complex
  https://en.wikipedia.org/wiki/Time-based_one-time_password

OTP Attacks

Threat Model: Stolen Password

- ❑ Adversary has <U,P>
- ❑ **Solved!**
- ❑ **Always keep in mind**



U pwd-U

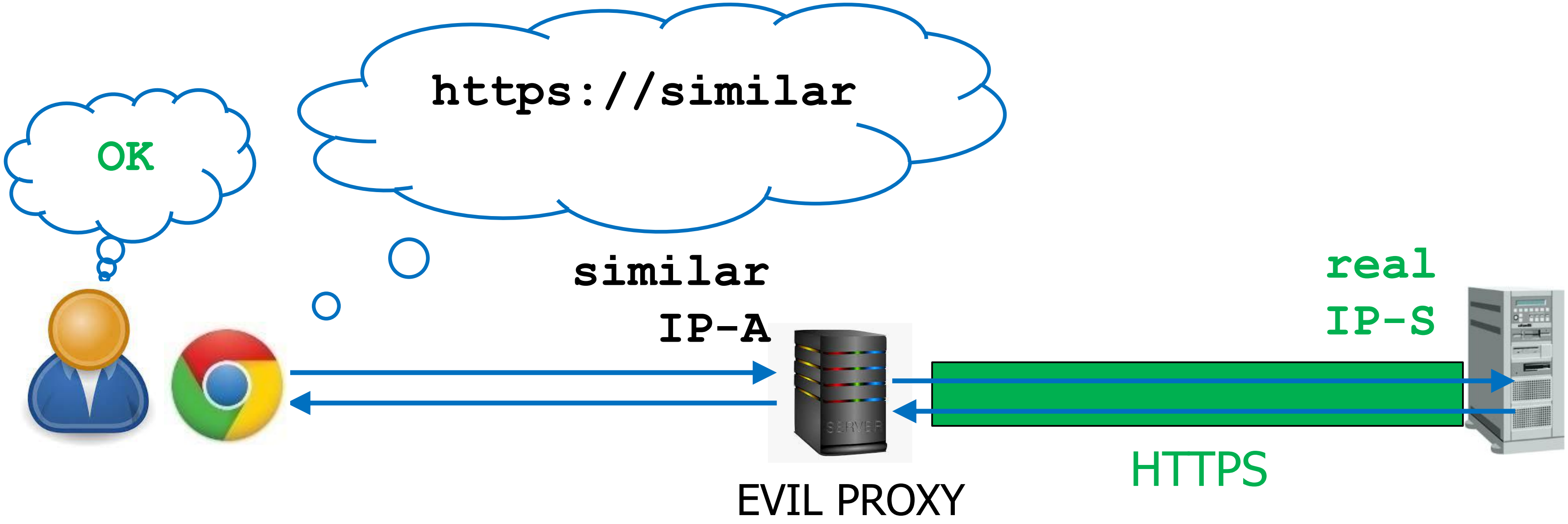
U pwd-U second(U)

???

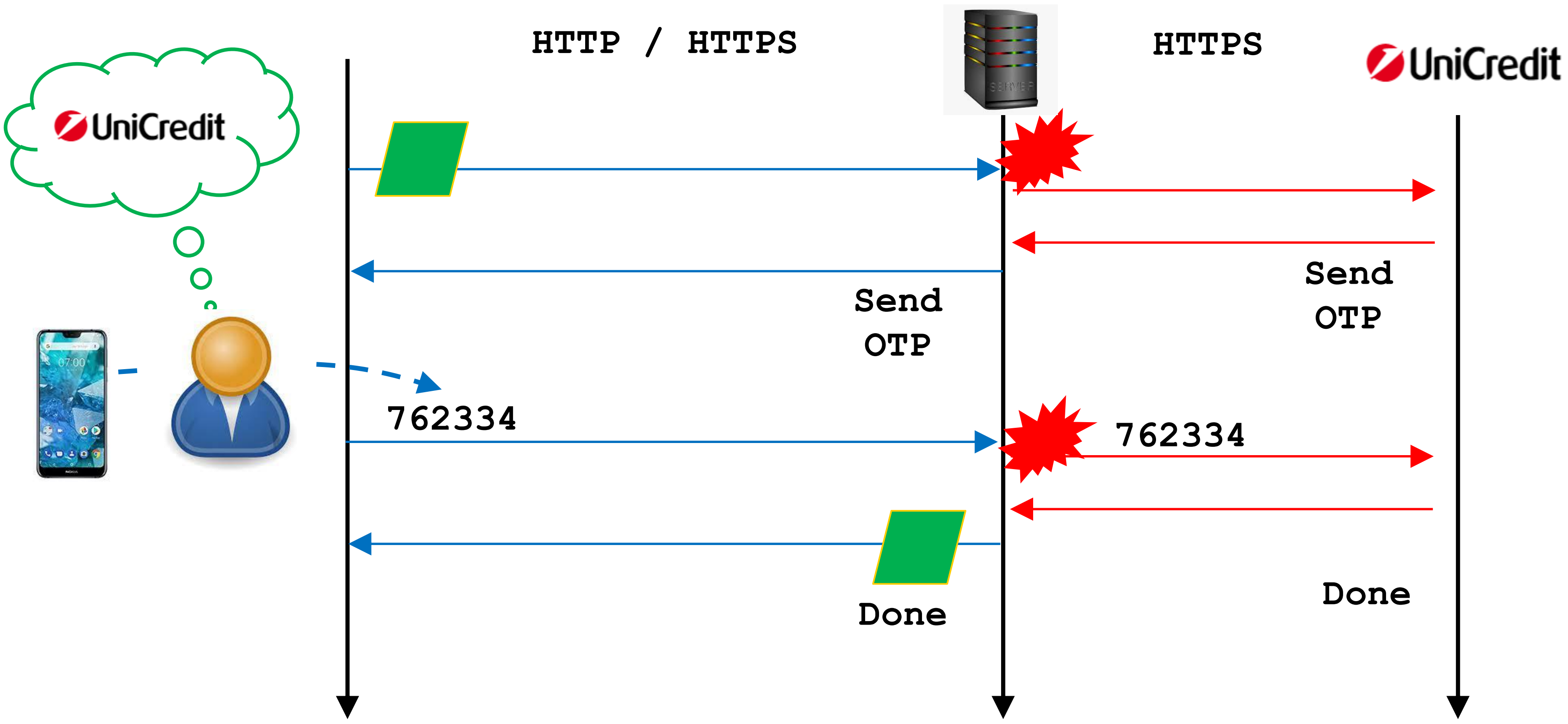
NOT Authenticated

Threat Model: "Real-time Phishing" (REMIND)

- User does **not** detect that is accessing the **wrong** URL

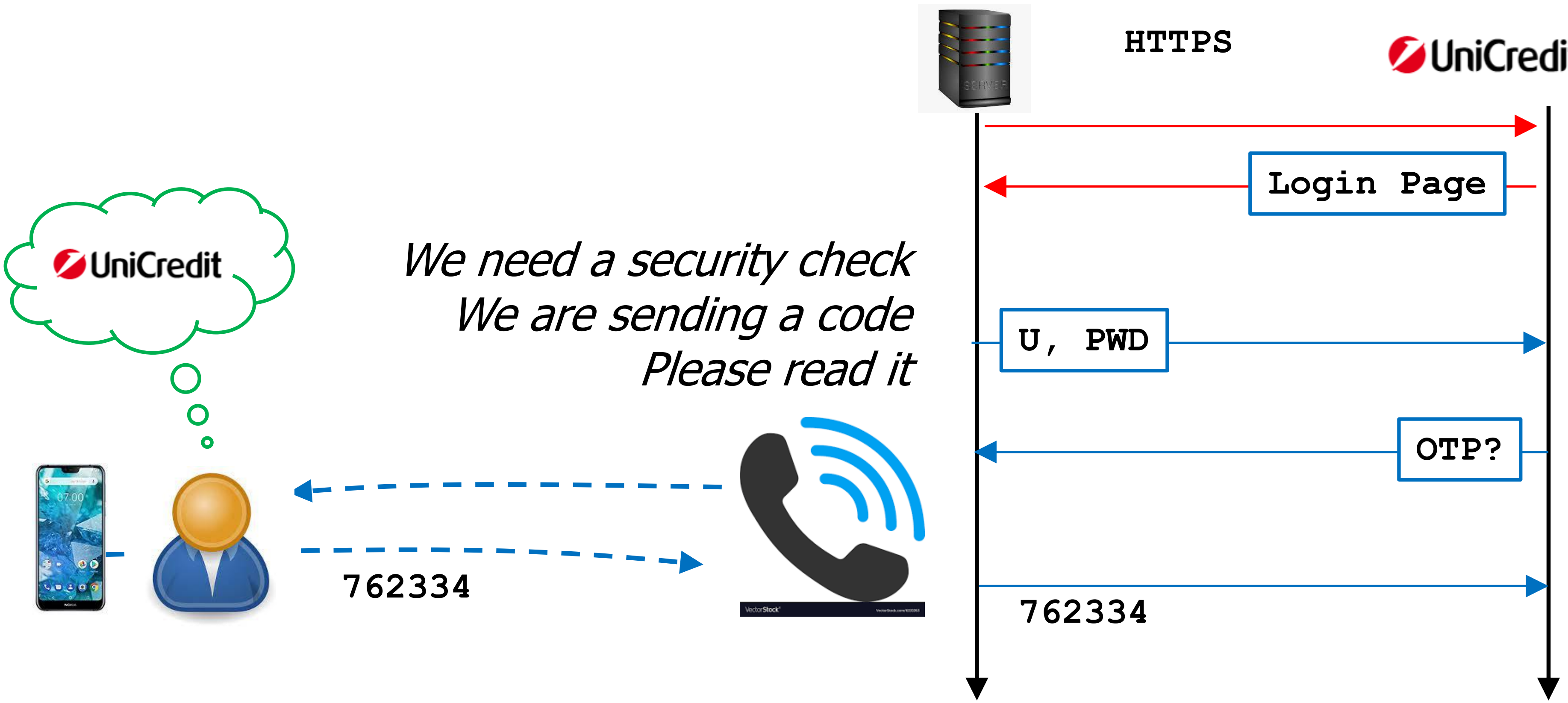


Threat Model: "Real-time phishing"



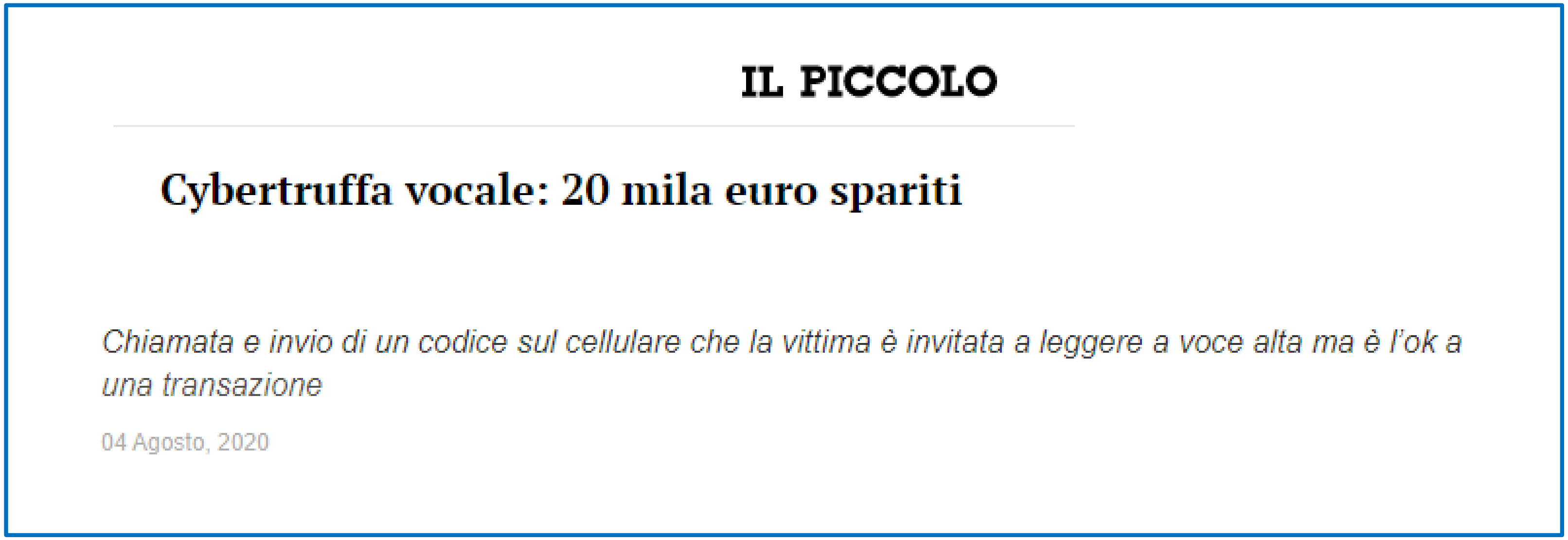
NOT Solved!

Other Threat Model: Vishing ("voice" phishing)



We need a security check
We are sending a code
Please read it

Unfortunately, it works...



IL PICCOLO

Cybertruffa vocale: 20 mila euro spariti

Chiamata e invio di un codice sul cellulare che la vittima è invitata a leggere a voce alta ma è l'ok a una transazione

04 Agosto, 2020

Search also "vishing" on Companion website

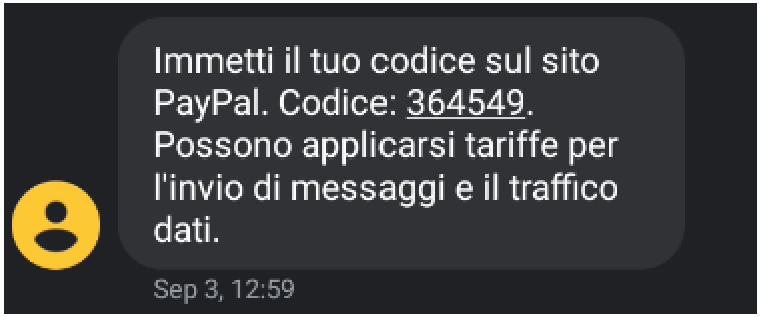
Keep in mind

- OTP does **not** solve phishing

- OTP makes phishing much **more costly** to attackers
- ...but it is still a real danger

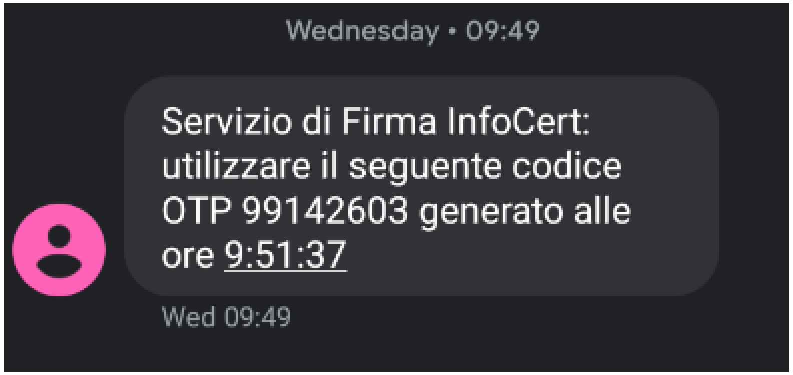
- Who am I giving this OTP to?
- For doing what?

Not very informative...



Immetti il tuo codice sul sito PayPal. Codice: 364549. Possono applicarsi tariffe per l'invio di messaggi e il traffico dati.

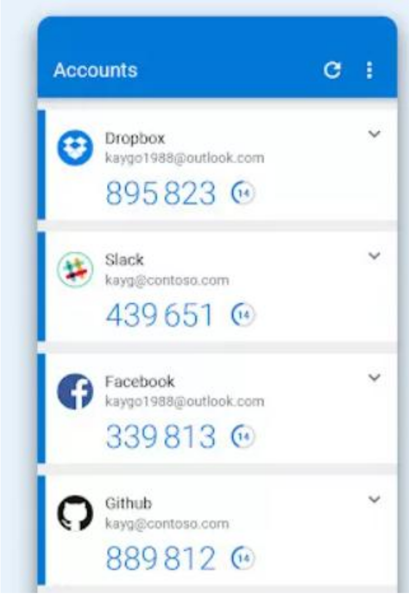
Sep 3, 12:59



Wednesday • 09:49

Servizio di Firma InfoCert: utilizzare il seguente codice OTP 99142603 generato alle ore 9:51:37

Wed 09:49



Accounts

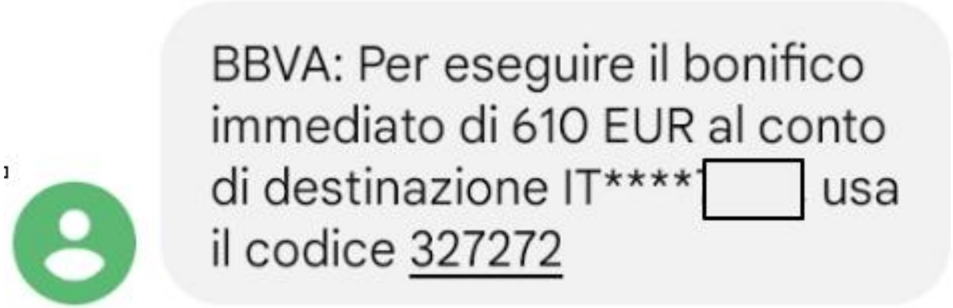
Dropbox
kaygo1988@outlook.com
895 823

Slack
kayg@contoso.com
439 651

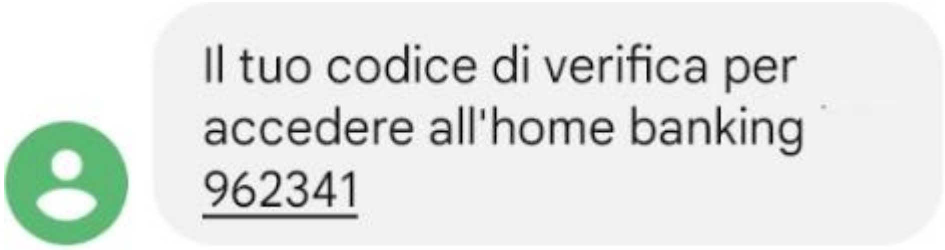
Facebook
kaygo1988@outlook.com
339 813

Github
kayg@contoso.com
889 812

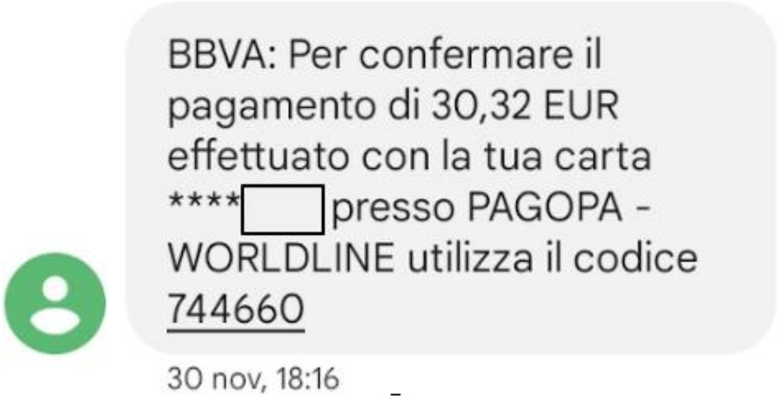
Much better



BBVA: Per eseguire il bonifico immediato di 610 EUR al conto di destinazione IT****[redacted] usa il codice 327272



Il tuo codice di verifica per accedere all'home banking 962341



BBVA: Per confermare il pagamento di 30,32 EUR effettuato con la tua carta ****[redacted]presso PAGOPA - WORLDLINE utilizza il codice 744660

30 nov, 18:16

Other Threat Models: OTP SMS

- Attacker knows password

  +

- Malware on smartphone
  - Read, forward, delete SMS
- SIM swap (fraudulent SIM change)
  - Phone number fraudulently taken by another SIM
- SMS routing attacks
  - SS7 phone protocol weakness: SMS sent to Attacker

- **Realistic** (search "MFA Attacks" on Companion website)
- **Not solved**

Other Threat Models: OTP AuthApp

- Attacker knows password

+

- Malware on smartphone
  - Read, forward, delete SMS
- SIM swap (fraudulent SIM change)
  - Phone number fraudulently taken by another SIM
- SMS routing attacks
  - SS7 phone protocol weakness: SMS sent to Attacker

AuthApp does not grant any "screenshot rights" to any other app

OTP do not travel across phone network

- **Realistic** (search "MFA Attacks" on Companion website)
- **Solved**

SMS vs AuthApp

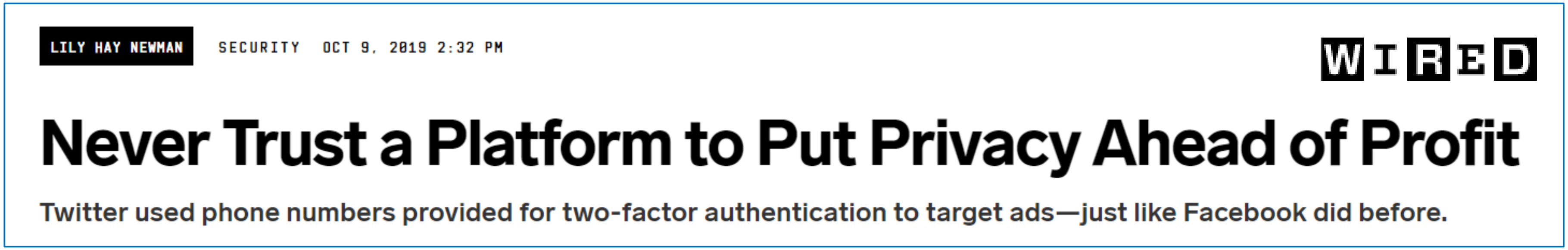
... it’s time to start your **move away from the SMS** and voice Multi-Factor Authentication (MFA) mechanisms.

... It bears repeating, however, that **MFA is essential** – we are discussing **which** MFA method to use, not **whether** to use MFA.....

Alex Weinert, Director of Identity Security Microsoft
November 2020

OTP: Privacy Implications

- OTP-SMS:
  - Service must know User **phone number**
  - Service **might abuse** this information (e.g. as an identifier for linking identities across different marketing databases)



LILY HAY NEWMAN SECURITY OCT 9, 2019 2:32 PM

WIRED

Never Trust a Platform to Put Privacy Ahead of Profit

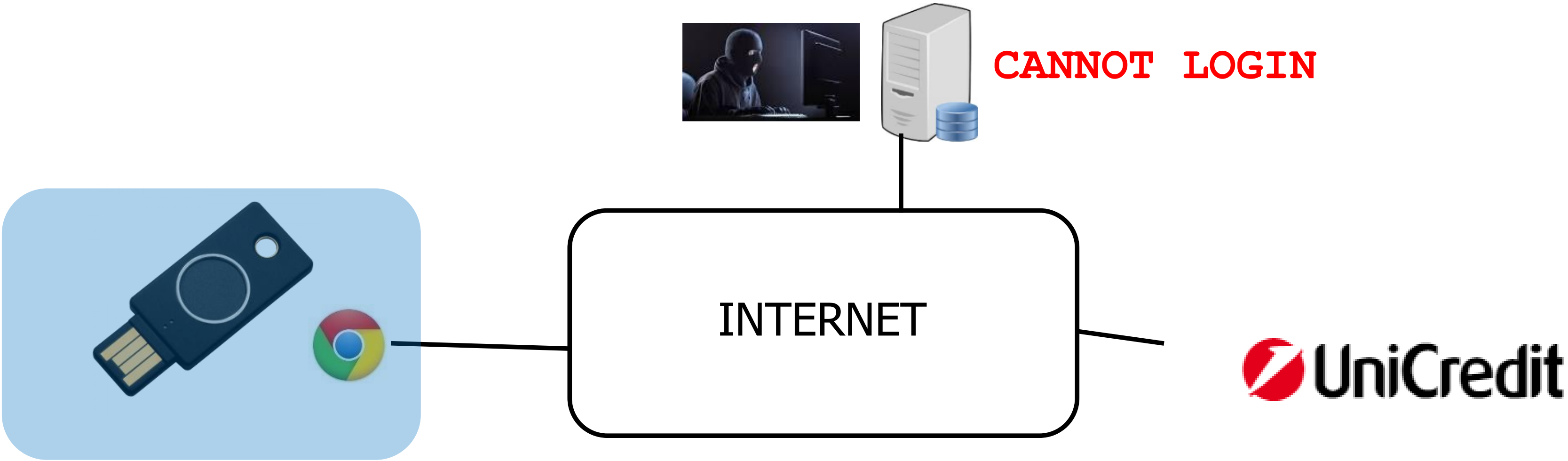
Twitter used phone numbers provided for two-factor authentication to target ads—just like Facebook did before.

- OTP-AuthApp:
  - Service need not know User phone number

Security Keys

Security Key

SecKey must be close to the Browser
(USB / Bluetooth / NFC)



CANNOT LOGIN

INTERNET

Security Key Linking: Requirement

DNS NAME

W1, U1, Kpriv-1, Kpub-1
W2, U2, Kpriv-2, Kpub-2
W3, U3, Kpriv-3, Kpub-3

W1	U1, Kpub1	Dropbox
W2	U2, Kpub2	f
W3	U3, Kpub3	LastPass

One SecKey associated with **many** services

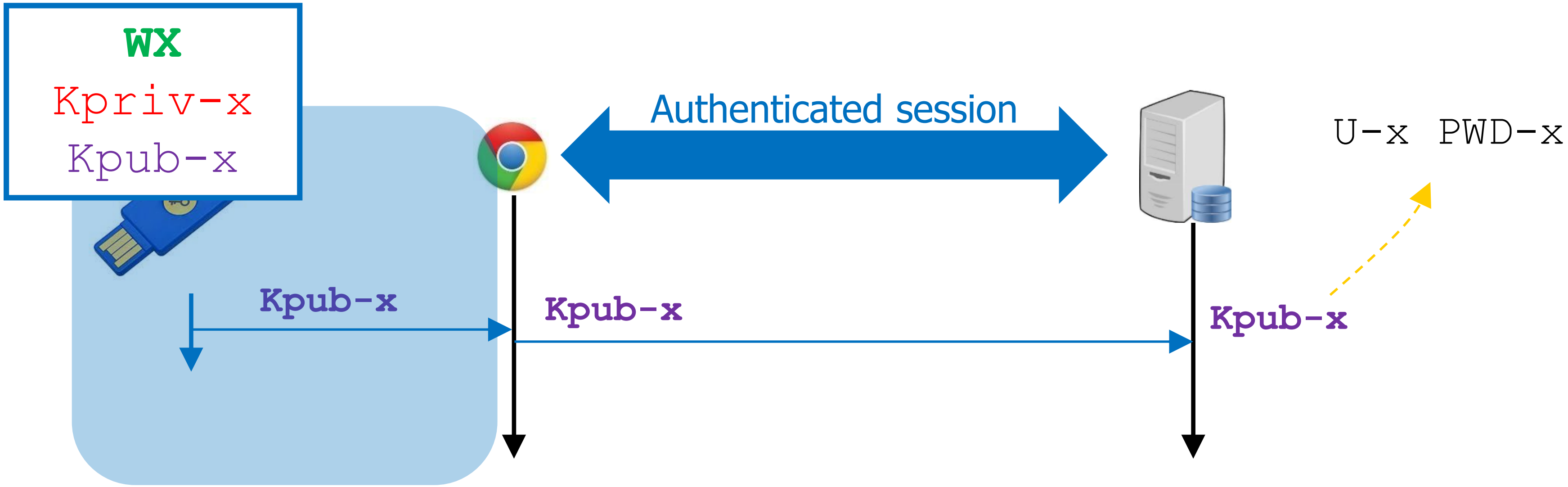
Remark

- ❑Username omitted from next slides for ease of description
  - ❑ **One** username and keypair for each service

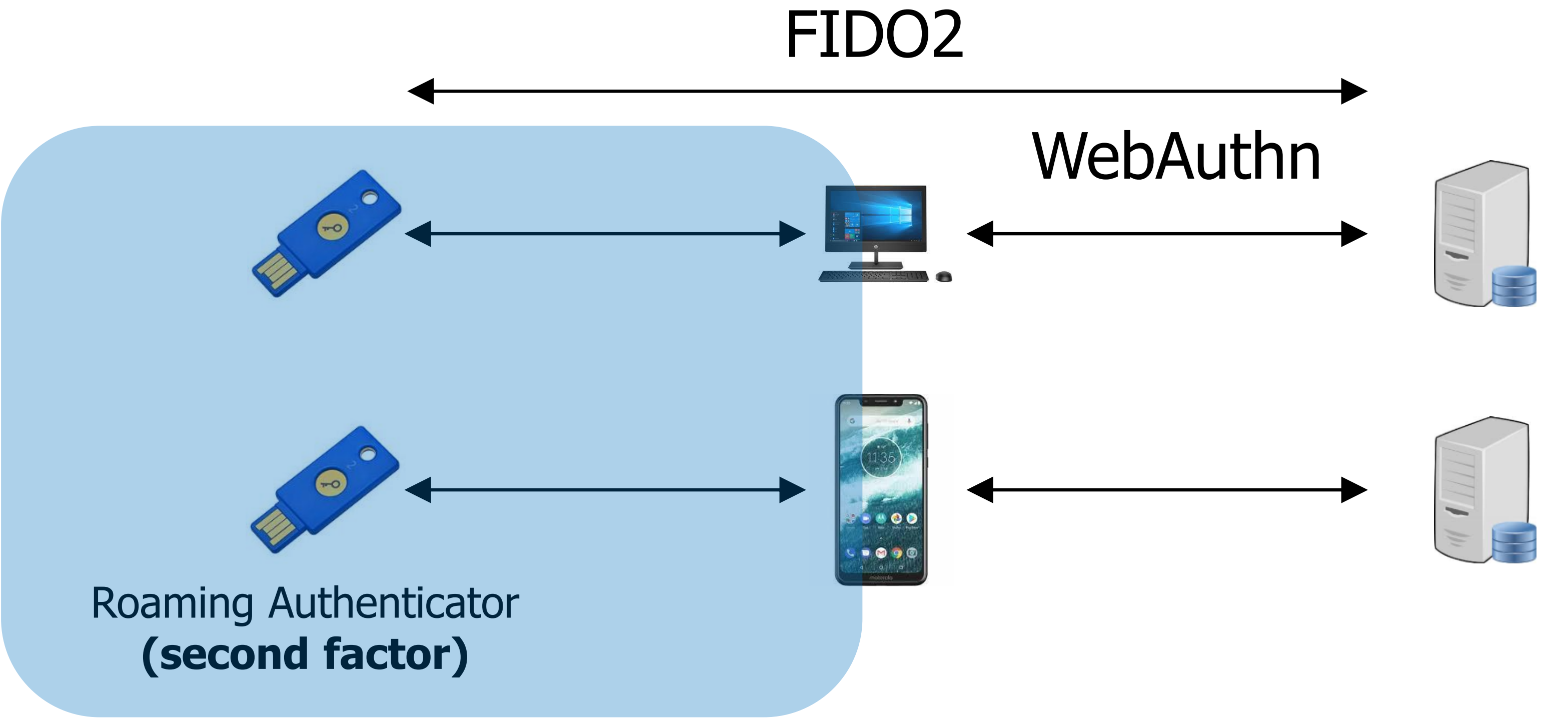
W1, ~~U1,~~ Kpriv-1, Kpub-1
W2, ~~U2,~~ Kpriv-2, Kpub-2
W3, ~~U3,~~ Kpriv-3, Kpub-3

Security Key Linking: Implementation

1. User authenticates to WX with U-x, PWD-x
2. SecKey generates <Kpriv-x, Kpub-x> to be used **only** with WX
3. SecKey sends Kpub-x to WX securely
4. **WX associates** Kpub-x with U-x



Open Standards (very complex...)



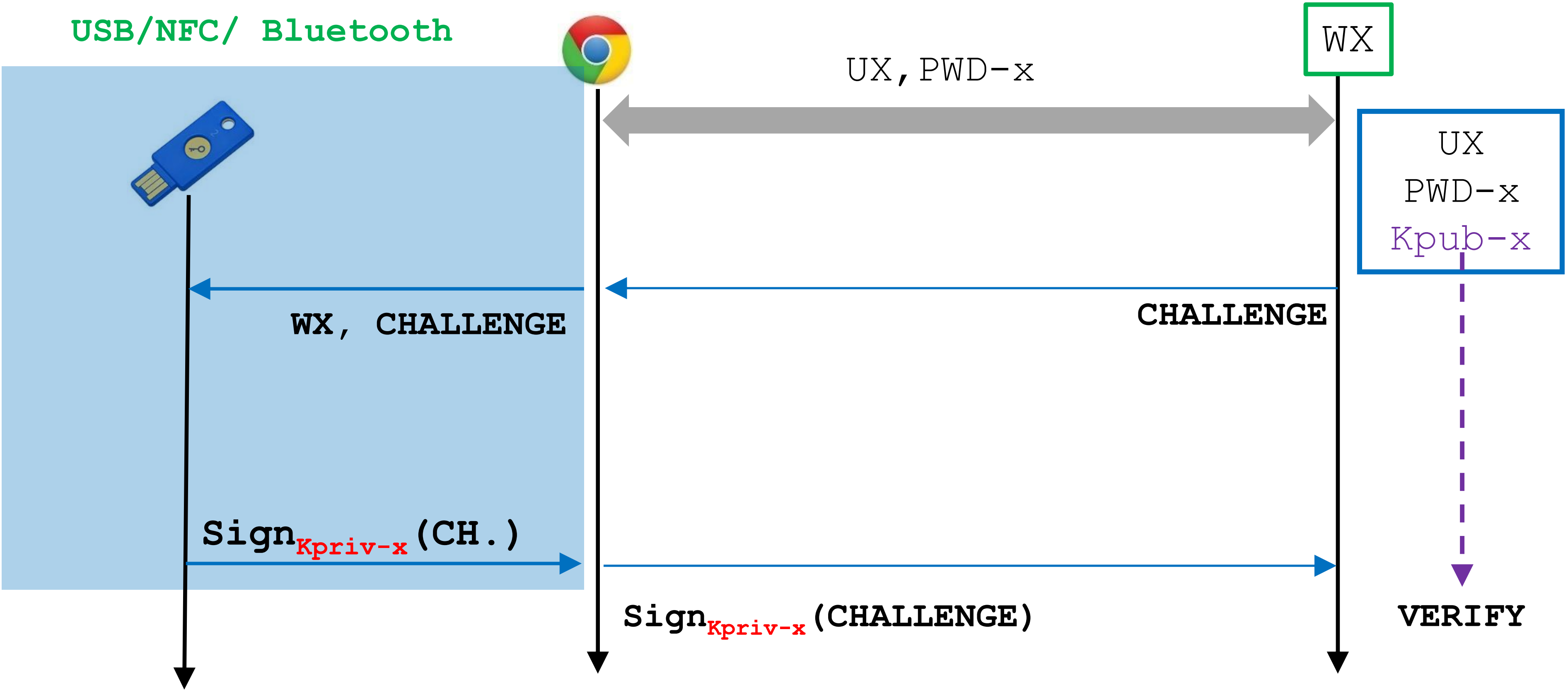
FIDO2

WebAuthn

Roaming Authenticator
(second factor)

No direct communication
Service ↔ Second factor

Security Key: Login



USB/NFC/ Bluetooth

WX

UX,PWD-x

UX
PWD-x
Kpub-x

WX, CHALLENGE

CHALLENGE

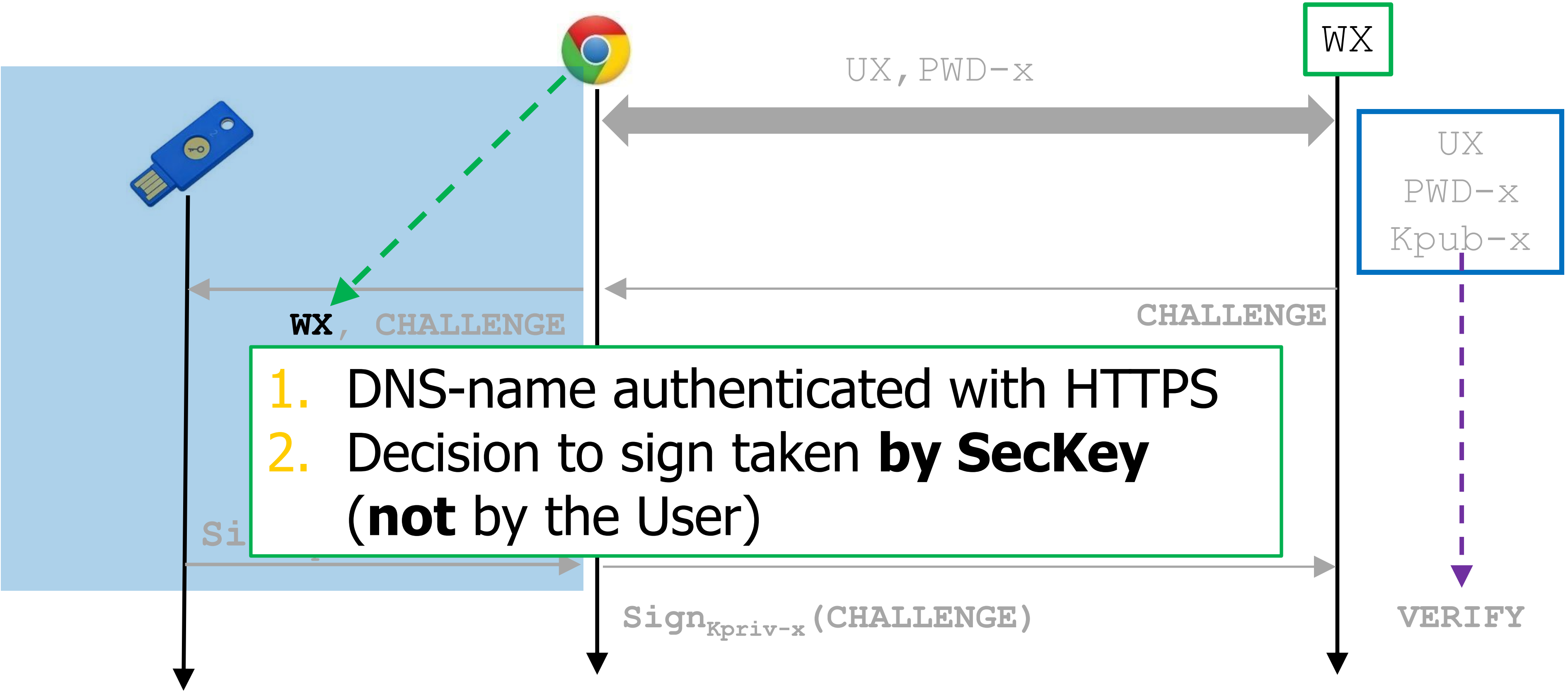
$\text{Sign}_{\text{Kpriv-x}}$(CH.)

$\text{Sign}_{\text{Kpriv-x}}$(CHALLENGE)

VERIFY

Real flow more complex

Key facts

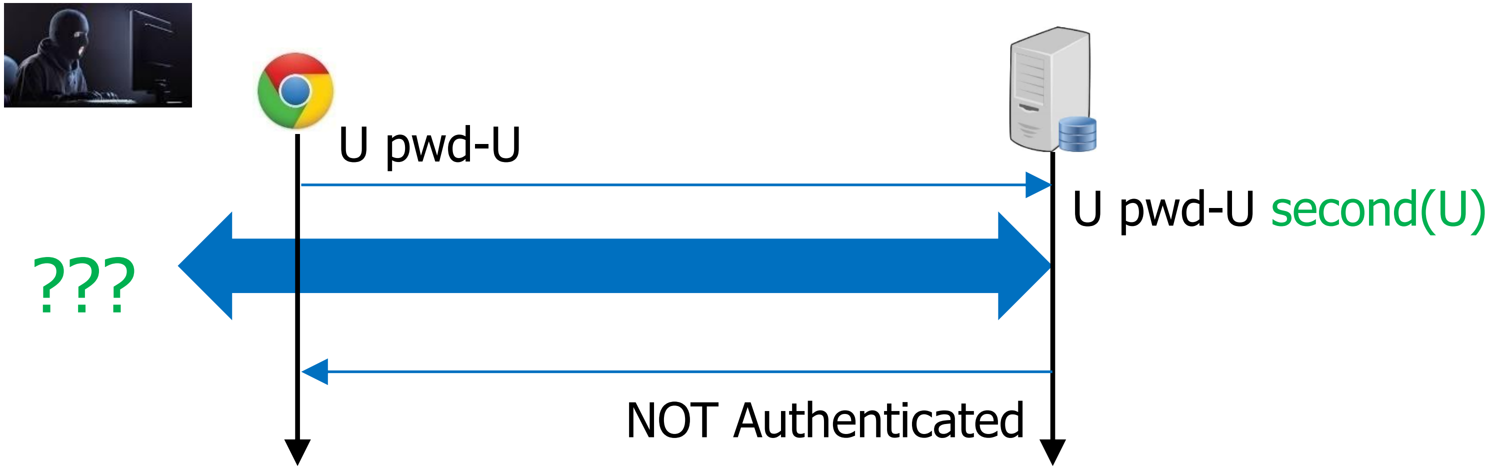


1. DNS-name authenticated with HTTPS
2. Decision to sign taken **by SecKey** (**not** by the User)

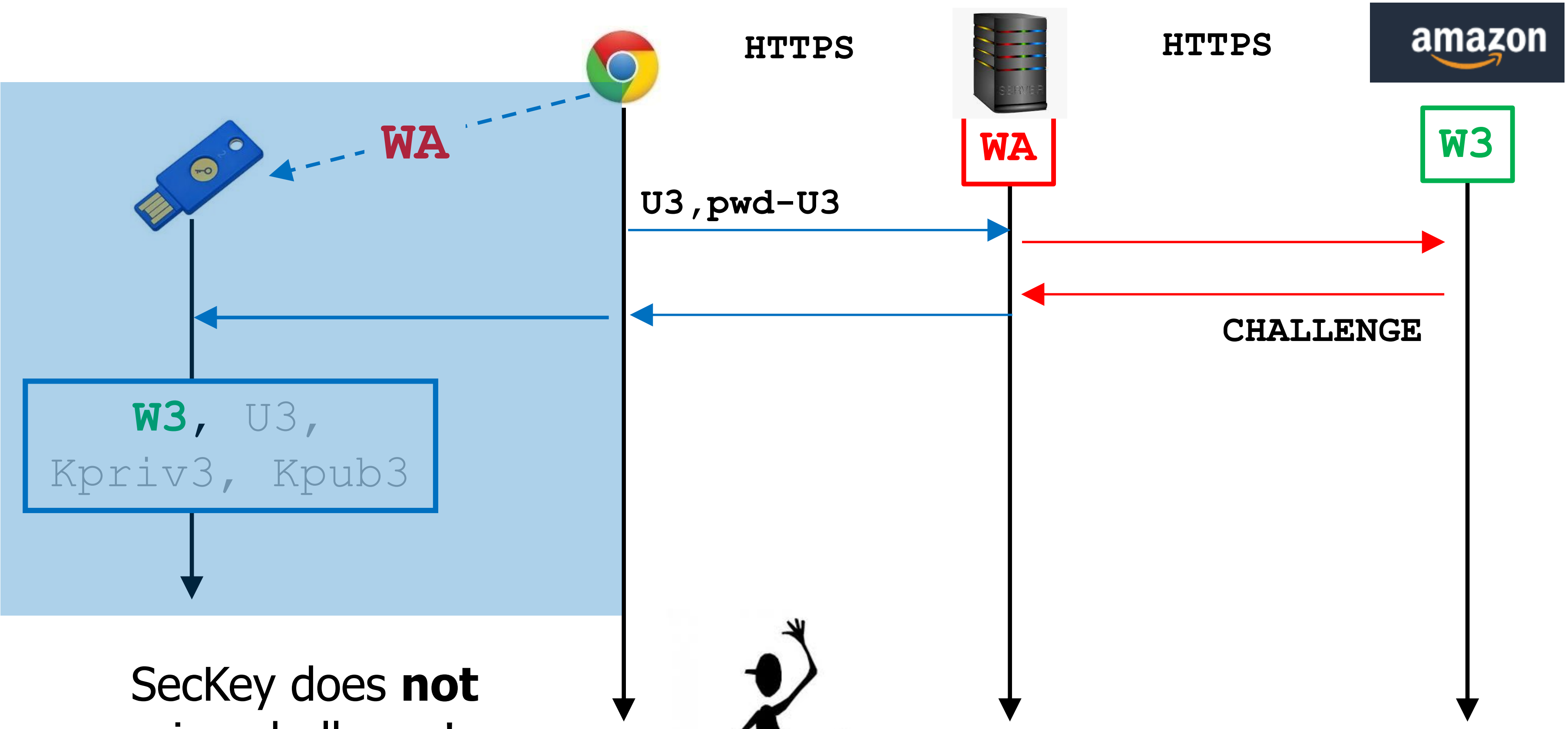
Security Key: Attacks

Threat Model: Stolen Password

- □ Adversary has <U,P>
- □ **Solved!**
- □ **Always keep in mind**

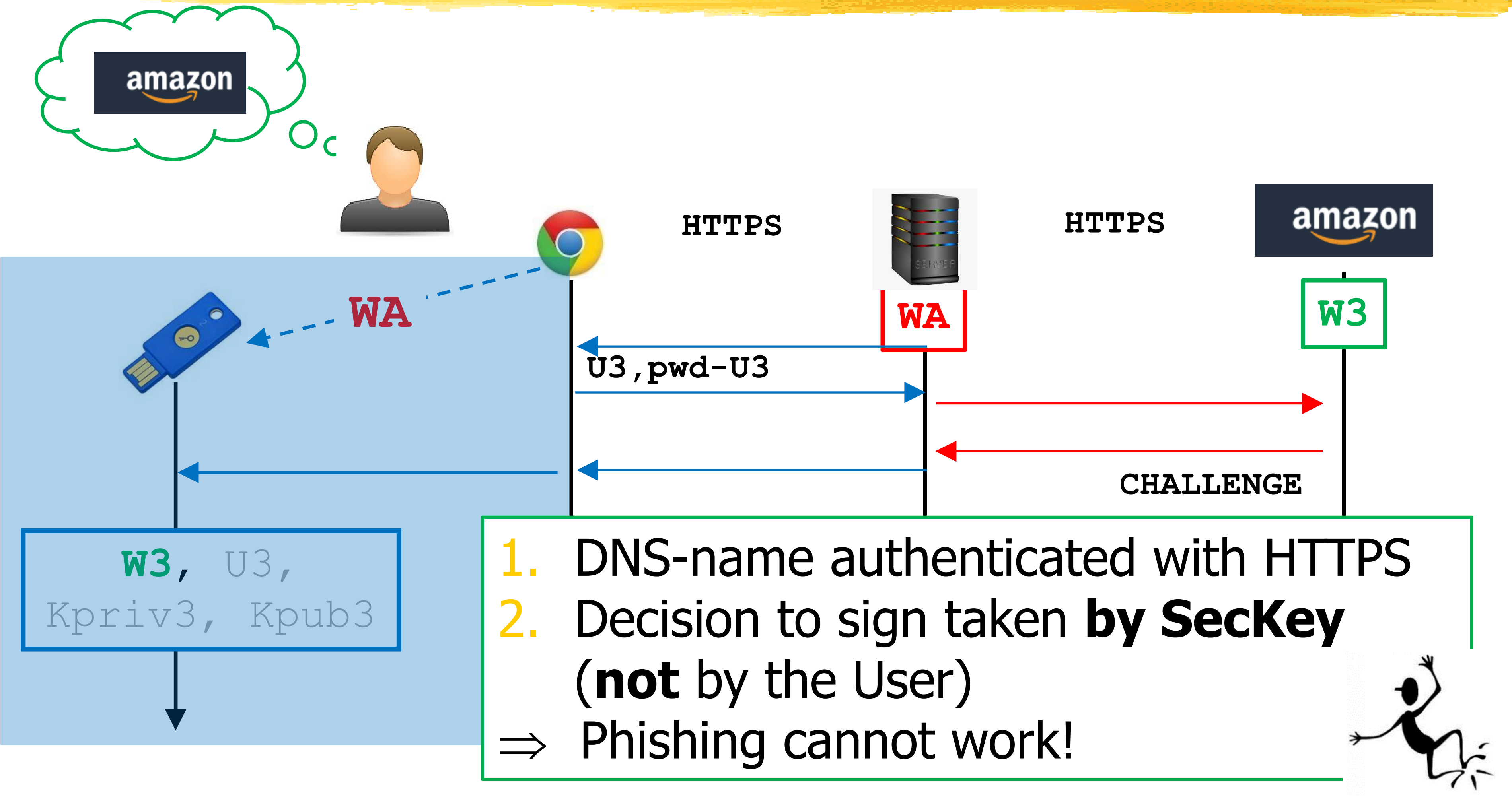


Real-time Phishing Solved! (I)



SecKey does **not** sign challenge!
(**DNS name mismatch**)

Real-time Phishing Solved! (II)



1. DNS-name authenticated with HTTPS
2. Decision to sign taken **by SecKey** (**not** by the User)

⇒ Phishing cannot work!

Unsolved Threat models (out of scope)

1. Attacker has **valid certificate for S name**
   - Browser cannot discriminate between real and fake service

2. Attacker has **malware on Browser device**
   - Malware can alter/forge Browser / SecKey traffic

Remark

MANY (omitted) complex details for coping with crucial requirements

- Attacker has physical access to SecKey (loss, stealing, brief access)
  - Cloning must be very difficult
  - Extracting set of service names must be very difficult

- Attacker may be the Manufacturer
  - If and when we realize it, certain SecKeys can no longer be trusted; Service must be able to know who the Manufacturer and product id are

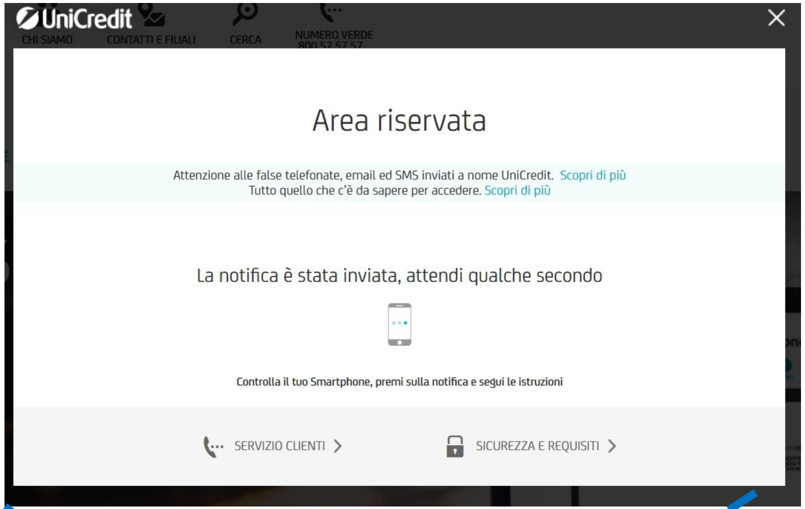
- Sets of Services might collude to link the respective user identities
  - Service cannot identify which specific SecKey it is interacting with

Push notifications

Second factor (REMIND)

- **Smartphone**
  - OTP SMS
  - OTP Authenticator App
  - Push notifications
- **SecurityKey** (USB/NFC/Bluetooth)

Smartphone
Push Notification (I)

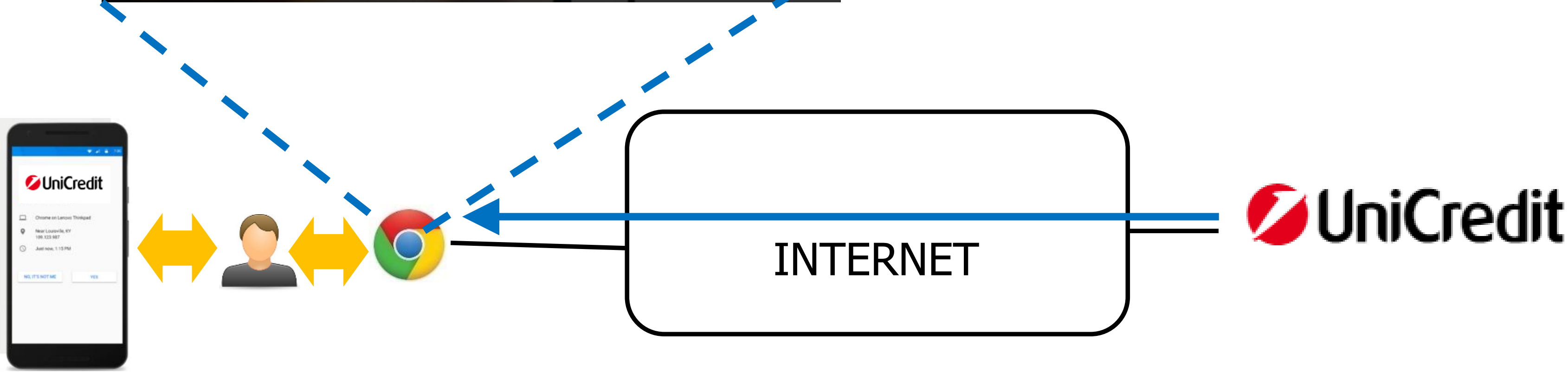


Area riservata

Attenzione alle false telefonate, email ed SMS inviati a nome UniCredit. Scopri di più
Tutto quello che c'è da sapere per accedere. Scopri di più

La notifica è stata inviata, attendi qualche secondo

Controlla il tuo Smartphone, premi sulla notifica e segui le istruzioni

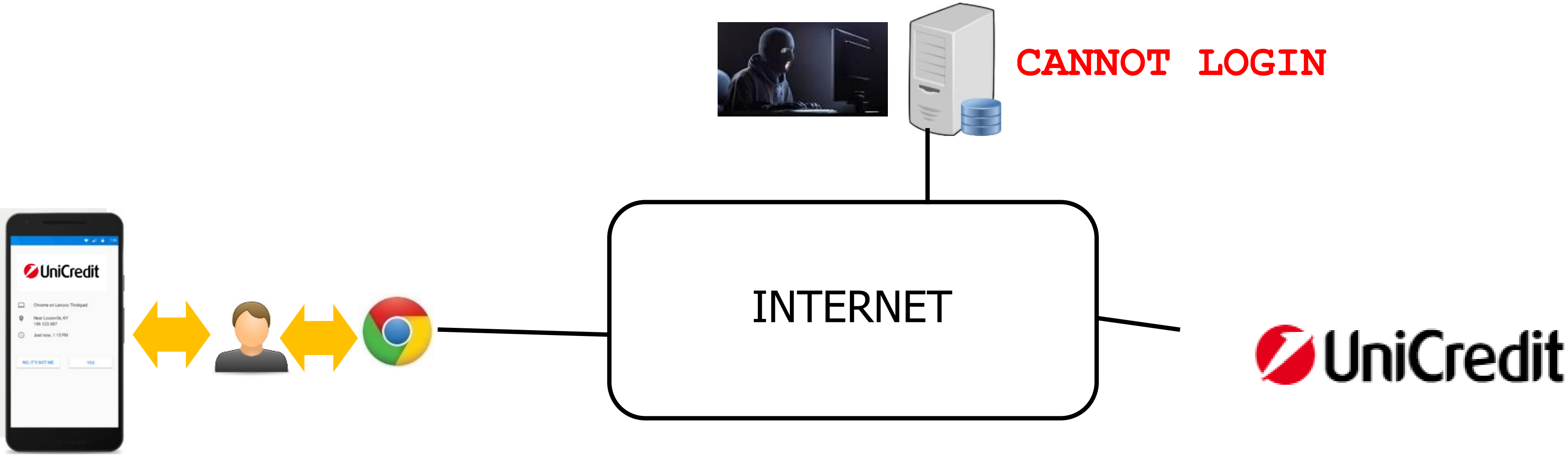


Smartphone
Push Notification (II)

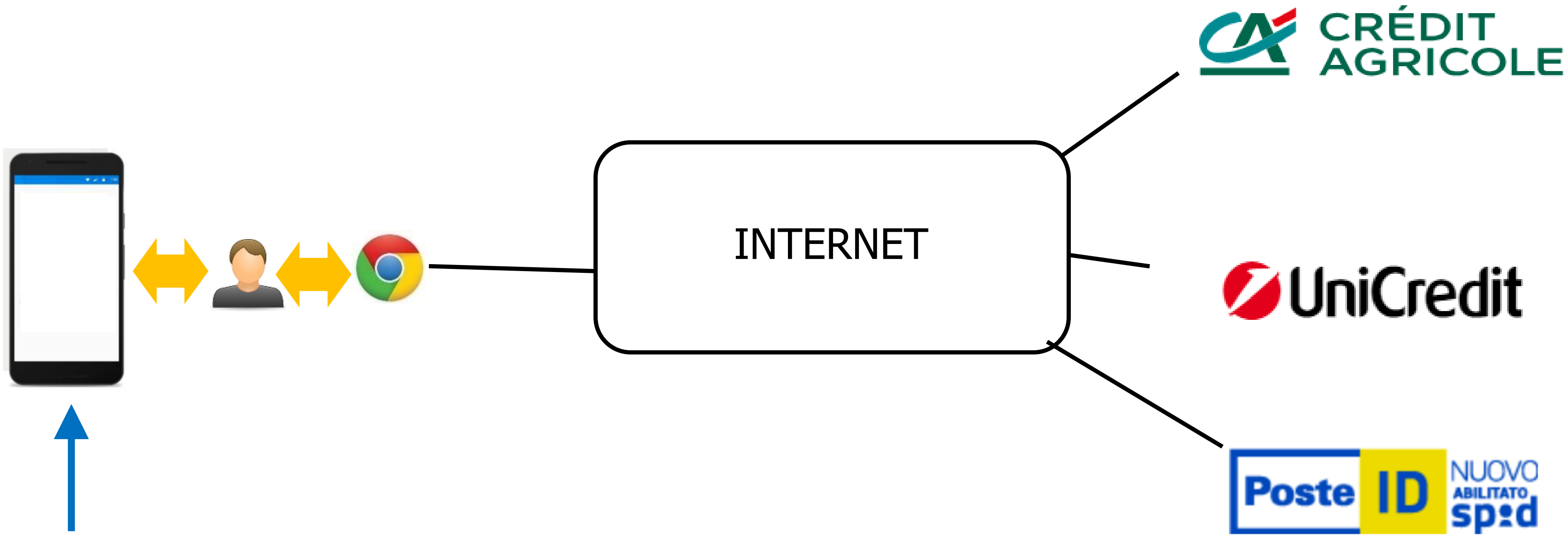
~~SecKey must be close to the Browser~~

~~(USB / Bluetooth / NFC)~~

Smartphone must be close to the User

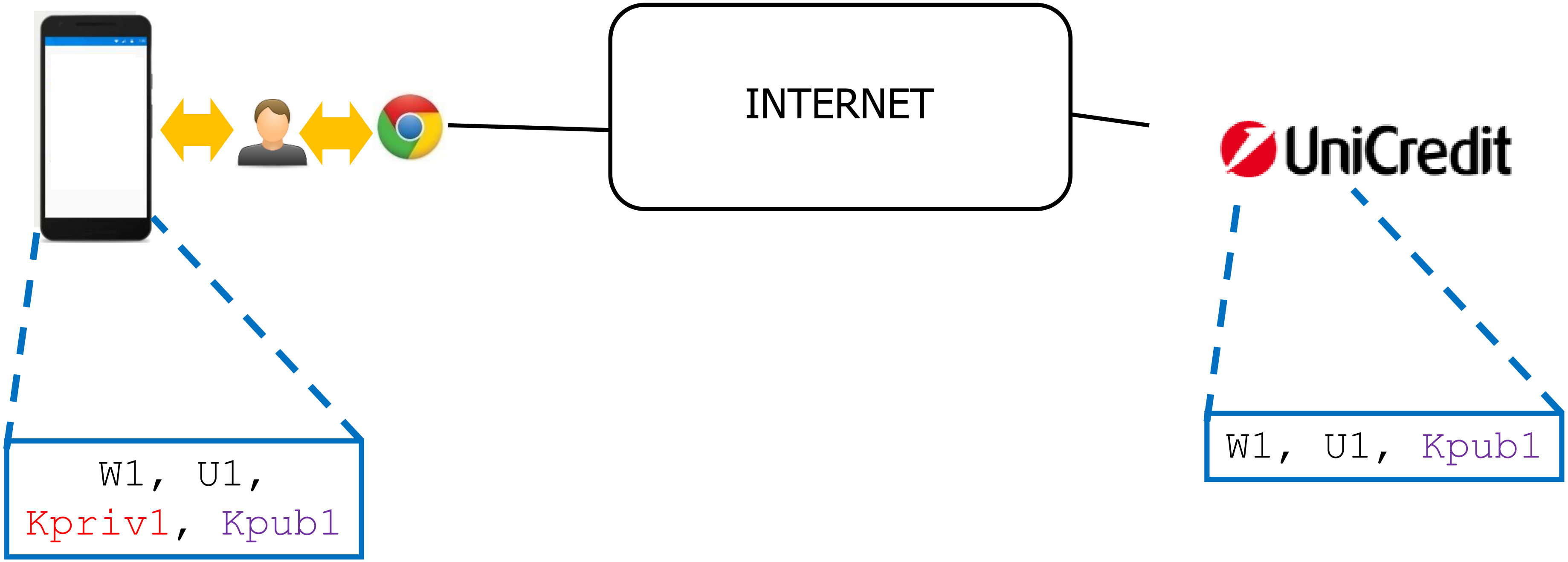


Smartphone Push Notification (III)



Every service has **its own** App where it sends notifications

Linking Requirement (Implementation omitted)



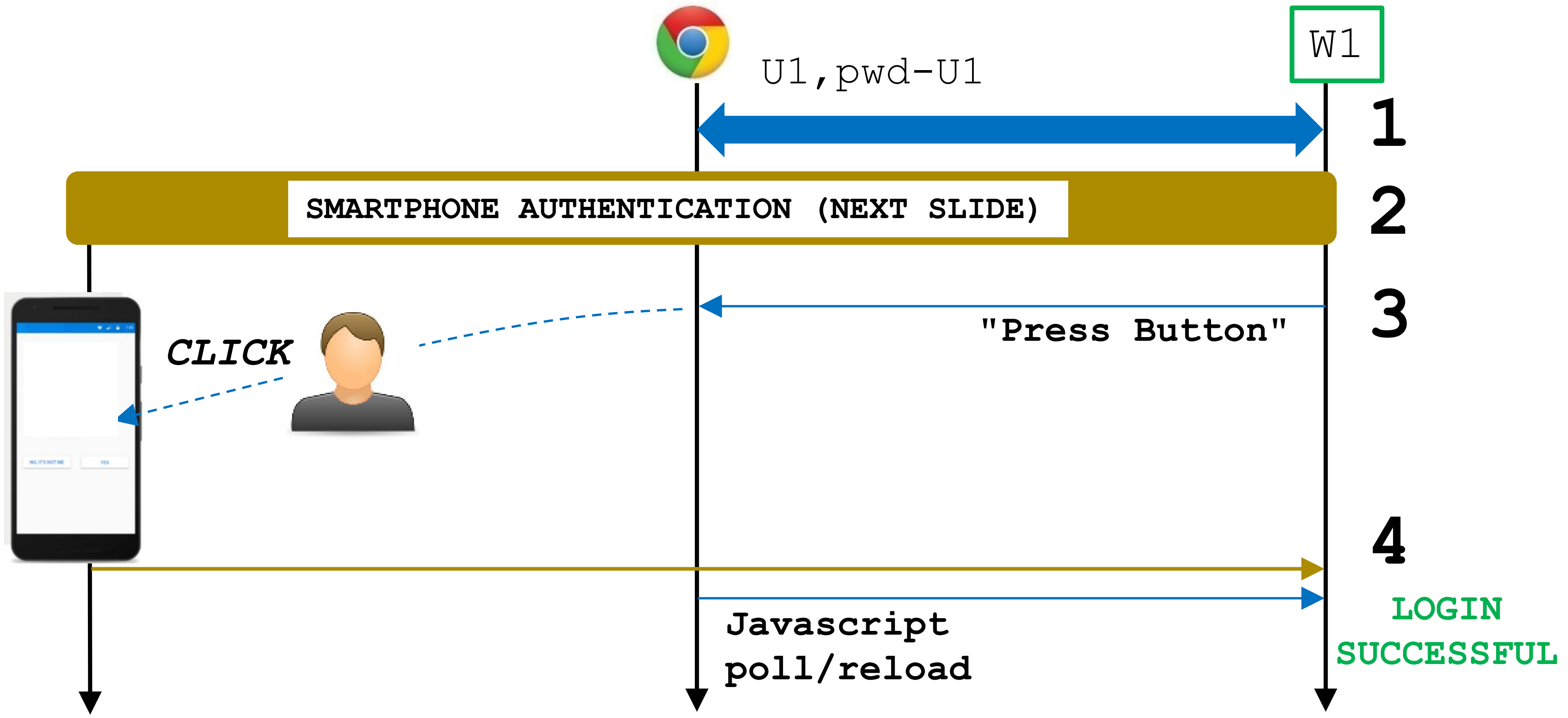
INTERNET

UniCredit

W1, U1,
Kpriv1, Kpub1

W1, U1, Kpub1

Login (Outline) (I)



W1

U1,pwd-U1

1

SMARTPHONE AUTHENTICATION (NEXT SLIDE)

2

"Press Button"

3

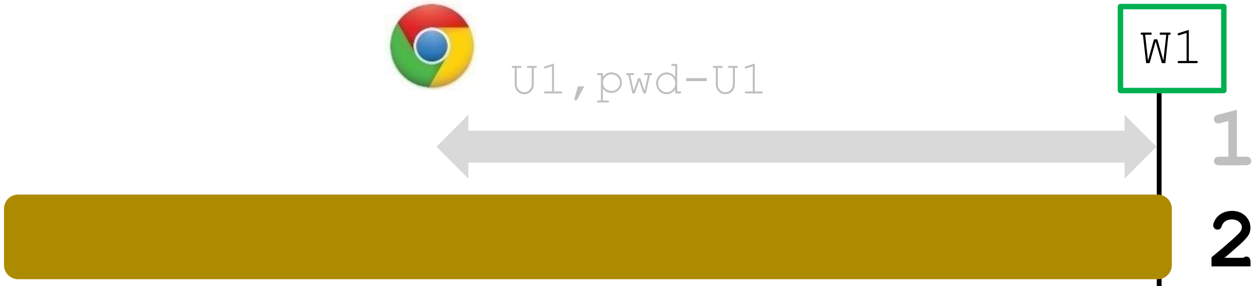
CLICK

4

LOGIN SUCCESSFUL

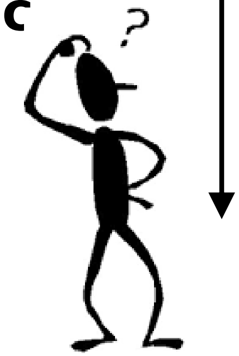
Javascript poll/reload

Key Problem



- ❑ **"push"** ⇒
  Service should connect to **Smartphone**

- ❑ What is its IP-address?
- ❑ It is usually **private** and **dynamic**



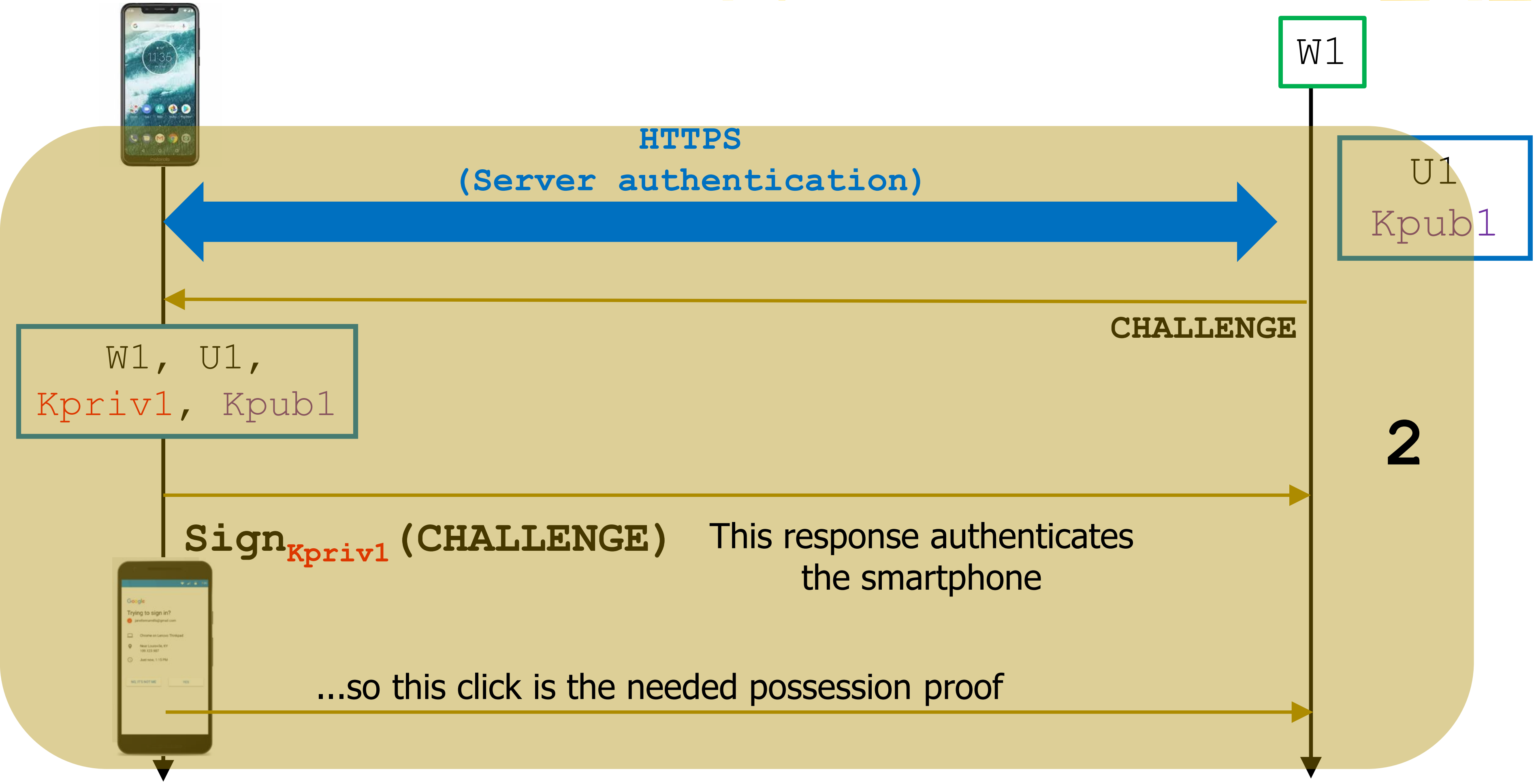
Solution (Outline)

- Service should connect to Smartphone

- What is its IP-address?
- It is usually **private** and **dynamic**
- Service sends notifications to a **cloud service**
- Every smartphone:
  - Continuously **polls** that cloud service
  - Connects as a TCP client and checks whether there is any notification

- Alternative implementation:
  - User launches smartphone app that acts as a client and connects to the service
  - Next slide assumes this pattern

Login (Outline) (II)



W1

U1

Kpub1

HTTPS

(Server authentication)

CHALLENGE

W1, U1,

Kpriv1, Kpub1

2

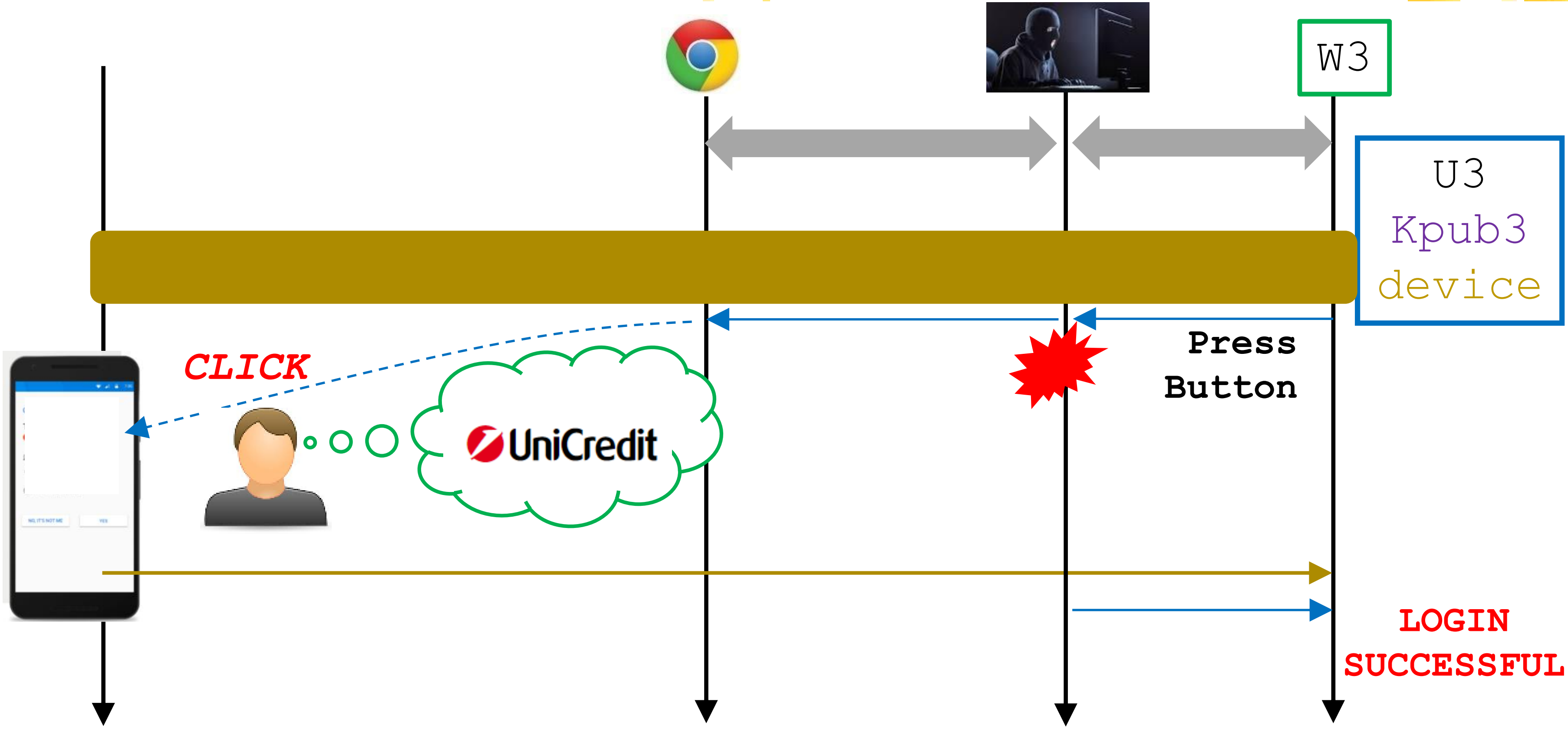
Sign$_{Kpriv1}$ (CHALLENGE)

This response authenticates the smartphone

...so this click is the needed possession proof

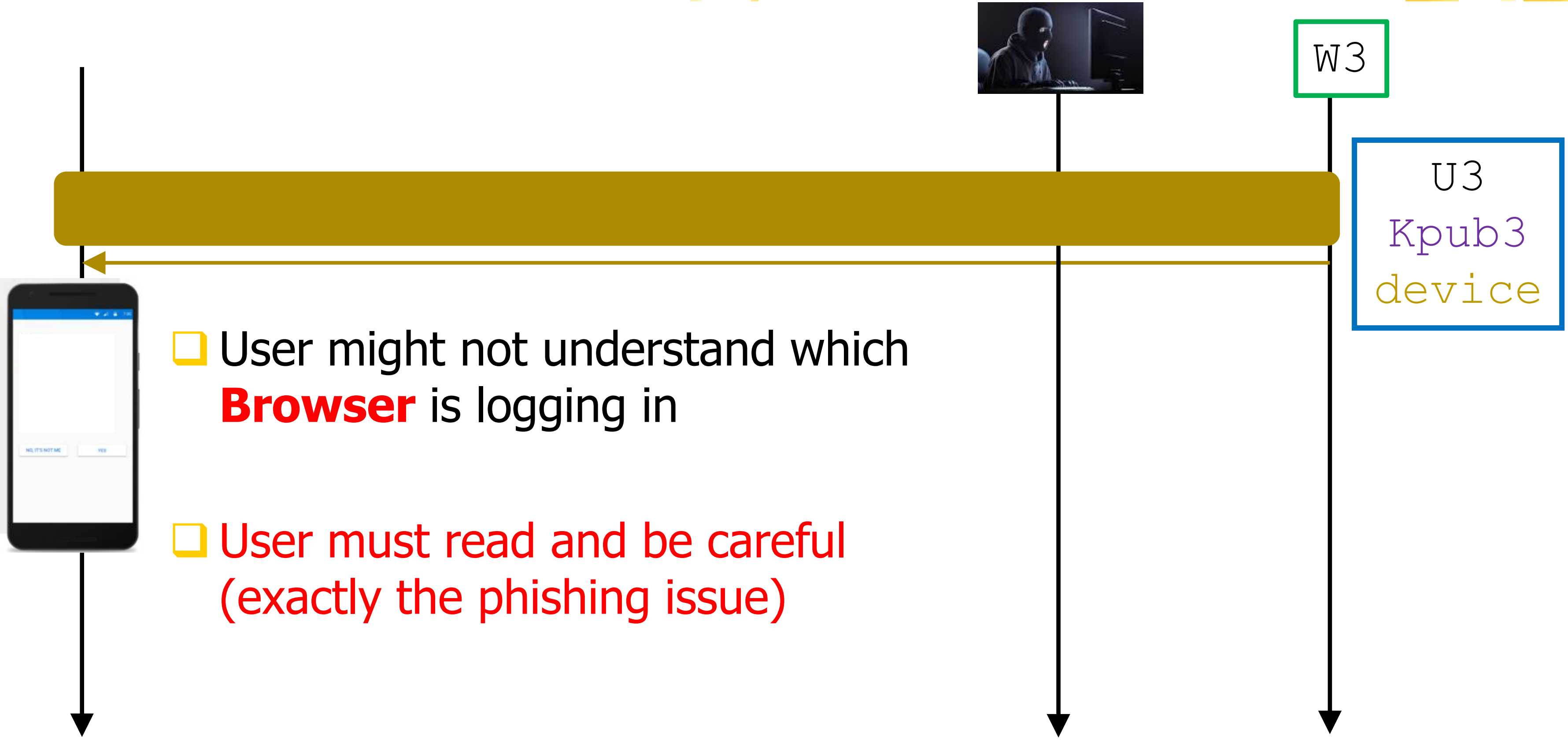
Push notifications Attacks

Threat Model: "Real-time phishing"



NOT Solved!

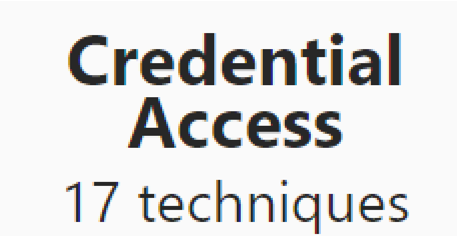
Keep in mind:
Not Phishing-Resistant



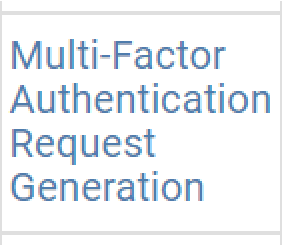
- ❑ User might not understand which **Browser** is logging in

- ❑ User must read and be careful (exactly the phishing issue)

MFA Bombing


Credential Access

17 techniques


Multi-Factor Authentication Request Generation

- Adversaries may **continuously** repeat login attempts in order to **bombard** users with MFA push notifications, SMS messages, and phone calls, potentially resulting in **the user finally accepting the authentication request** in response to "MFA fatigue."

- Unbelievable but it may indeed work...

MFA:
Summary of Limitations

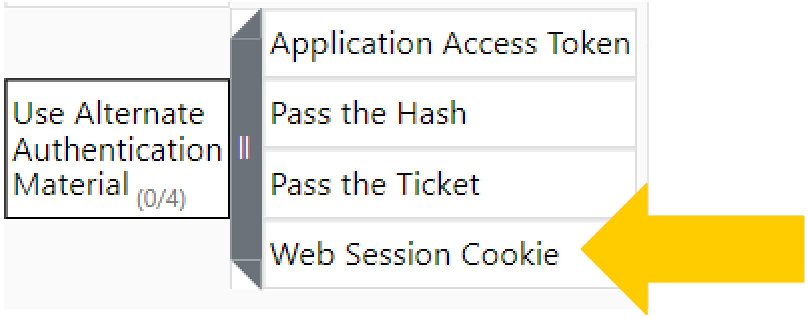
Does 2FA protect AFTER authentication?

- 2FA is checked **only** at the beginning of a session
  - Webapp login
  - Workstation logon (if it were deployed in such a setting)

- 2FA does **not** defend against attacks **during** an authenticated session
  - Stealing of authentication cookie
  - Pass-the-hash / Pass-the-ticket

Use Alternate Authentication Material

Lateral Movement
9 techniques



Use Alternate Authentication Material (0/4)

Application Access Token

Pass the Hash

Pass the Ticket

Web Session Cookie

- Alternate authentication material is **legitimately generated** by systems **after** a user or application successfully authenticates by providing a valid identity and the required authentication factor(s).

- By **stealing** alternate authentication material, adversaries are able to bypass system access controls and authenticate to systems **without knowing the plaintext password** or **any additional authentication factors**.

Summary of Limitations (I)

- **Everything but SecKey**
  - Phishing / Voice Phishing
  - Who am I authorizing?
  - For doing what?

- OTP AuthApp better than OTP – SMS
  - Malware
  - SIM swap
  - SMS routing

Summary of Limitations (II)

- **Everything**
  - Attacks **after** initial authentication

    (discussed in the Appendix)
  - Recovery procedures for **loss** of second factor
    - Be careful of your **email password**

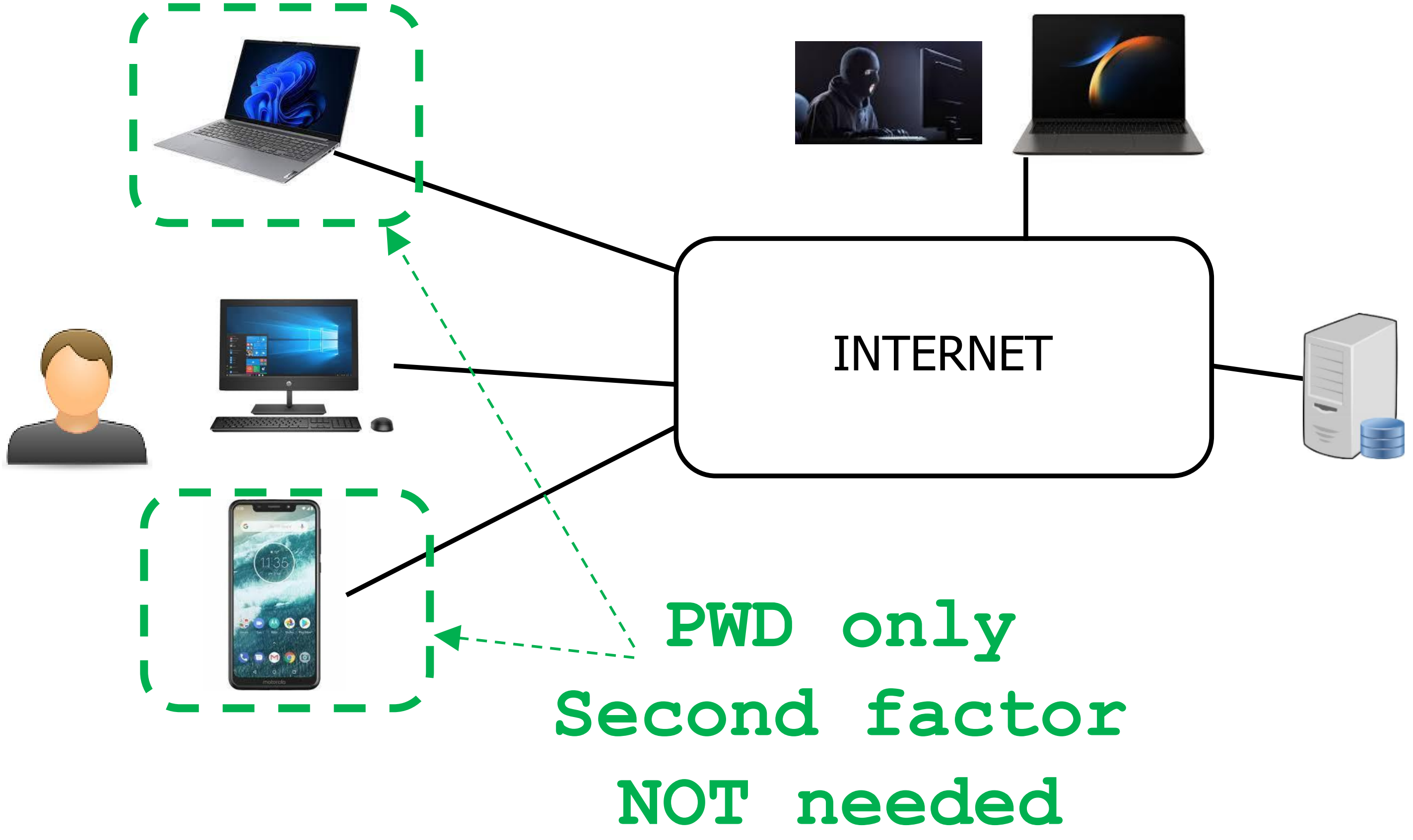
Passwordless Login (Passkey)

Warning: Terminology

- "Special" handling of **2FA@S** from **certain devices**
  - Only password (no 2FA)
  - Not even password (!)

- **Terminology not uniform** across vendors / consortia
- **Myriad** of different scenarios:
  - We will give a "simplified and general" description
  - Mapping to specific cases not easy
  - Search "passwordless" on companion website
- Key terms:
  - Trusted device
  - Passwordless device (or login)
  - **Passkey**

Trusted Device: Example


INTERNET

PWD only

Second factor

NOT needed

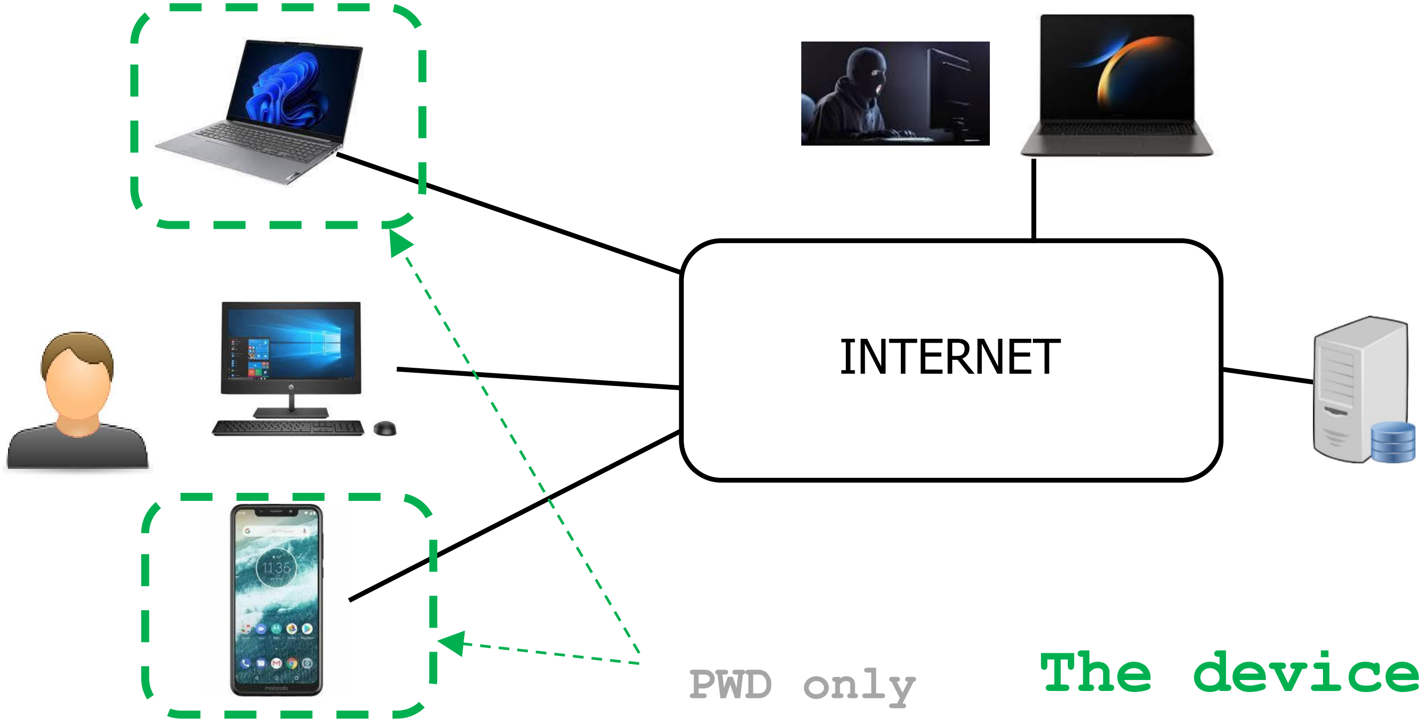
Trusted Device: Functionality

- Scenario:
  - User has activated **2FA** on Service S

- User may declare a certain device D-K **trusted**:
  - User authenticates from D-K with **password only**

- Much easier to use

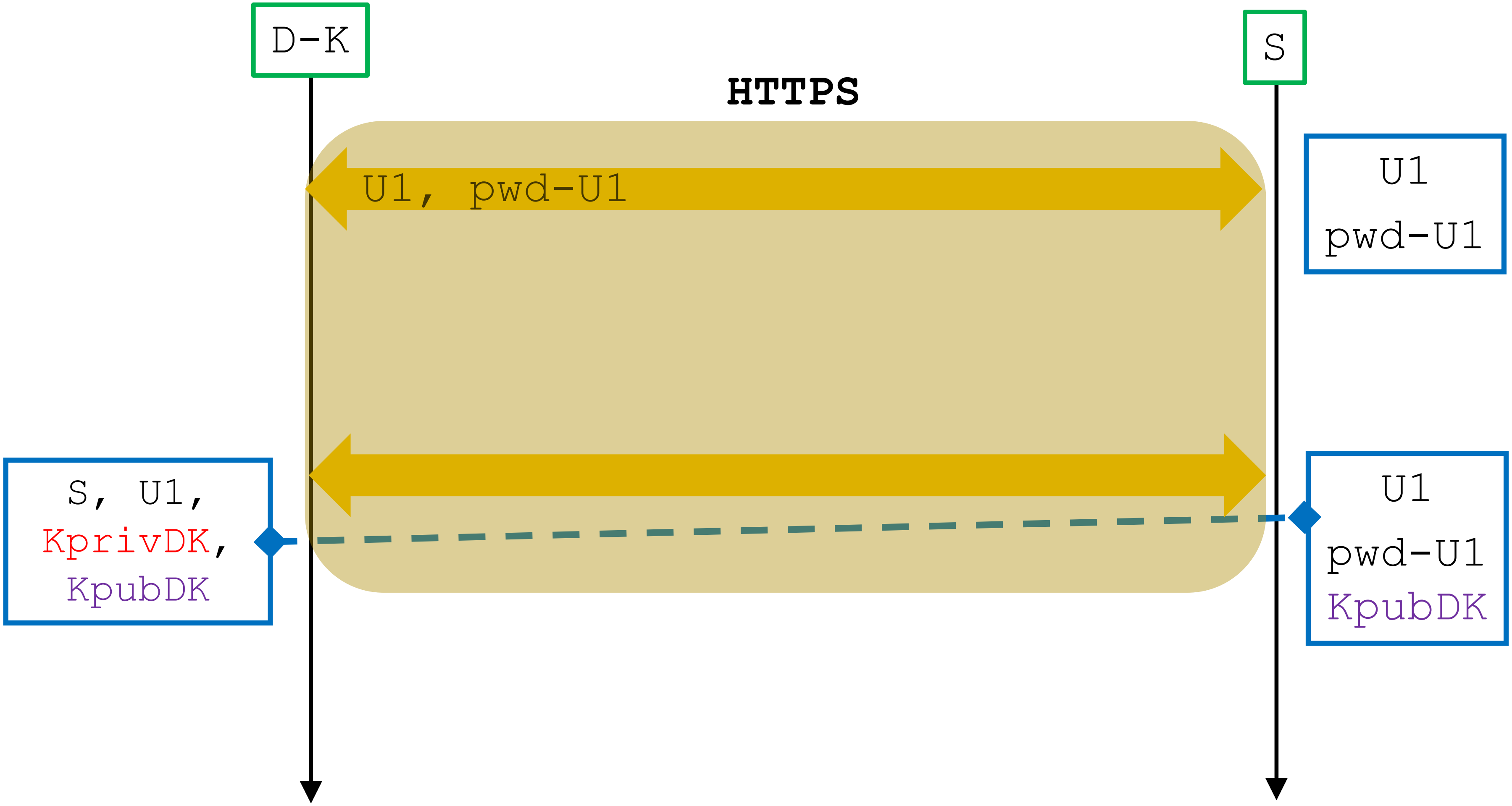
Trusted Device ≈ 2nd Factor



PWD only
Second factor
NOT needed

The device itself
is
the second factor

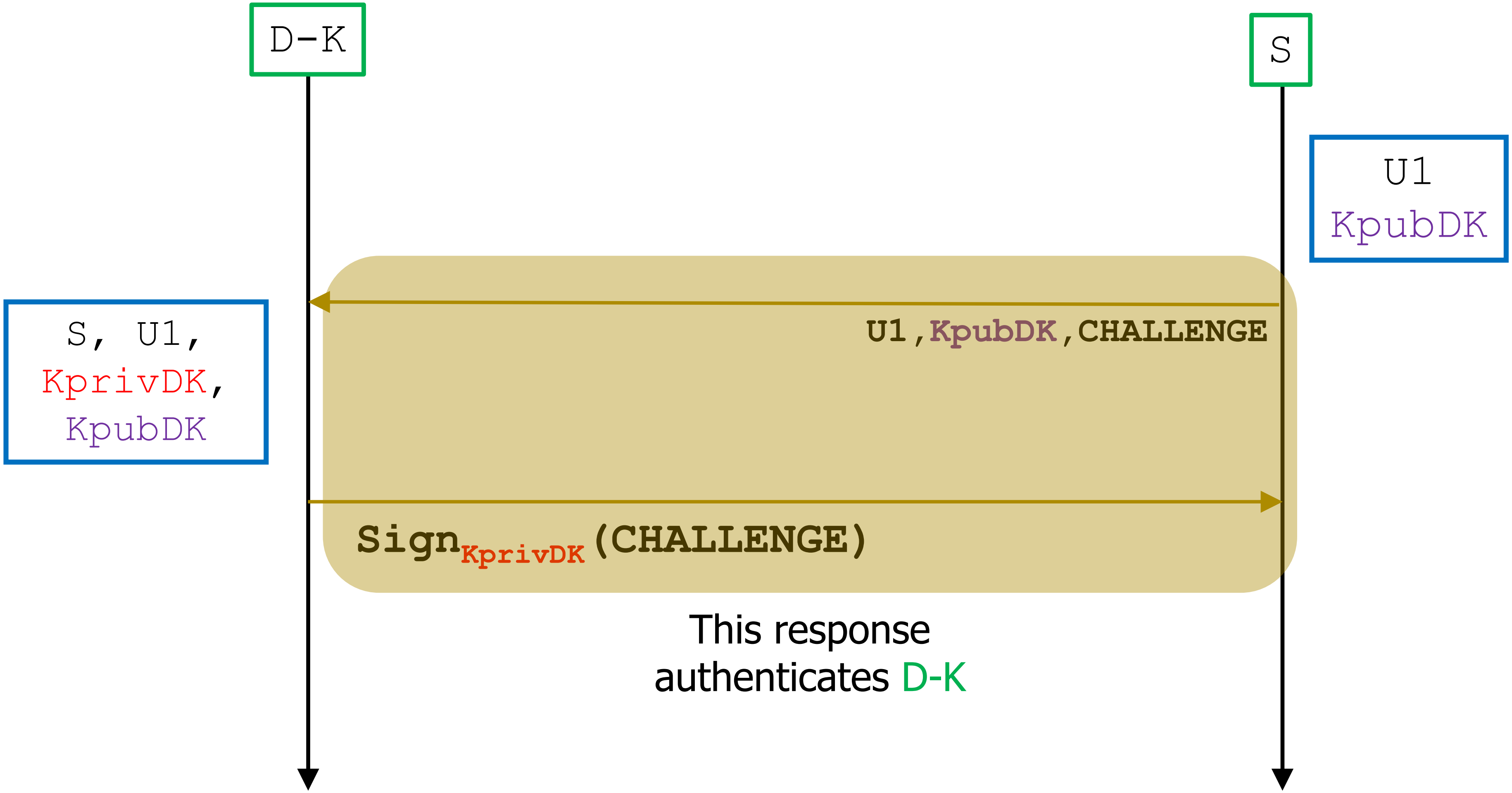
Trusted Device Establishment (Outline)



Remark 1

- User may declare device D-K **trusted**:
- This can **only** happen while User is logged in from D-K with password **and 2nd factor**
- Obvious

Trusted Device Authentication (Outline)



This response authenticates D-K

Remark 2

- User may declare device D-K **trusted**:
  - User authenticates from D-K with **password only**

- S can decide **autonomously** to occasionally **require second factor** anyway
  - D-K connects from an anomalous geographic location
  - D-K was declared trusted long time ago
  - ...

Hhmmm...

- User may declare device D-K **trusted**:
  - User authenticates from D-K with **password only**

- Attacker knows U, PWD-U@S
- Attacker **steals** trusted device D-K
  - Or **physical access** for some time

⬇

- Attacker can access S

Remark 3

- User may declare device D-K **trusted**:
  - User authenticates from D-K with **password only**

- U must:
  - **Protect** access to D-K with password **different** from PWD-U@S (as soon as D-K is declared trusted)
  - **Revoke** trusted status of D-K asap (in case D-K is lost / stolen)

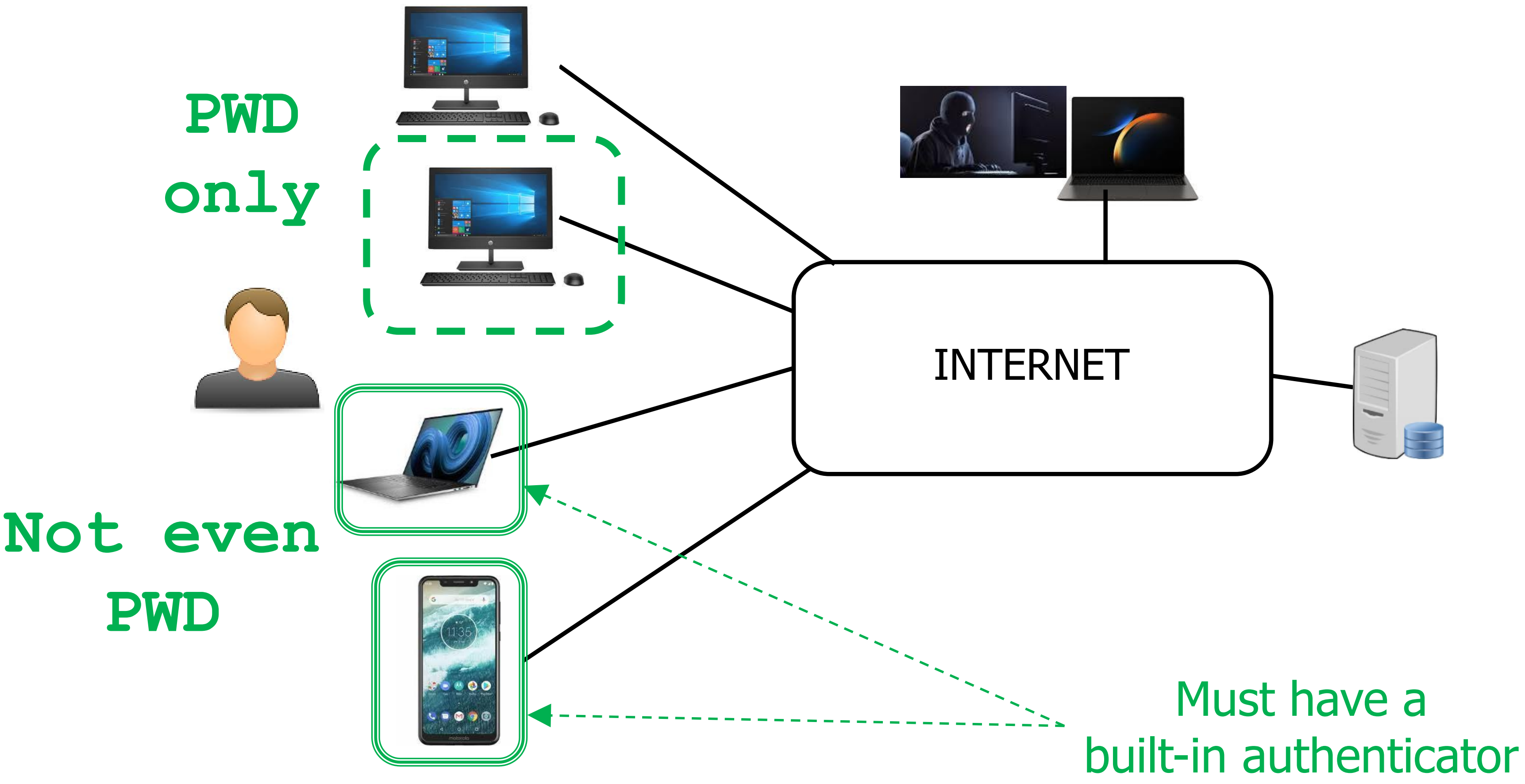
Passwordless Device: Functionality

- User has declared D-K **trusted**:
  - User authenticates from D-K with password only

- ...and **passwordless**:
  - User authenticates from D-K **without any password** (!)

- Requirement: D-K must have a "**built-in authenticator**"
  - Fingerprint reader
  - Face recognition

Passwordless Device: Example

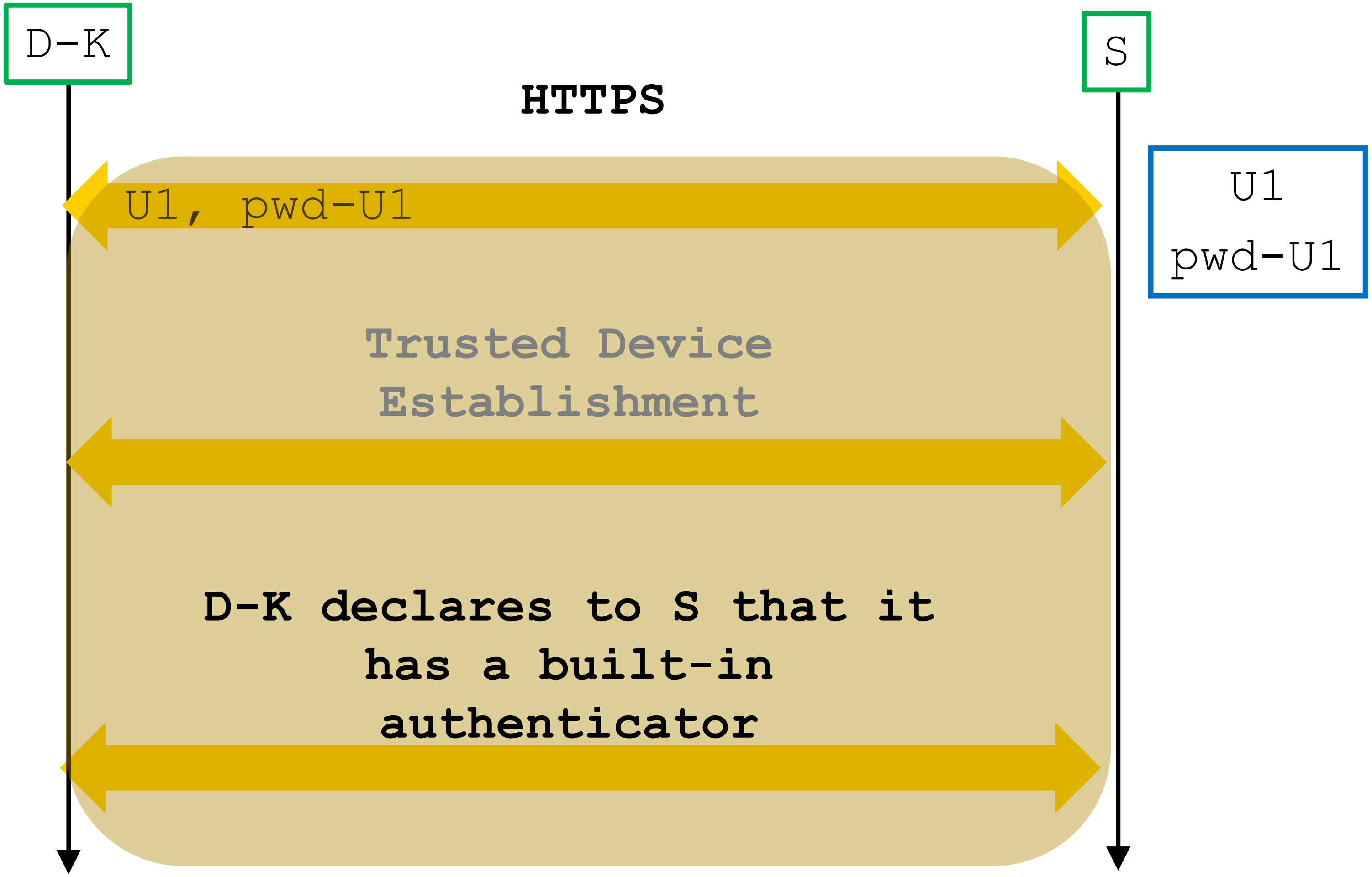


Remark

- User declare D-K **trusted** and **passwordless**:
  - User connects from D-K **without any password (!)**

- Common scenario when:
  - S = bank, SPID-enabled app
  - D-K = smartphone

Passwordless Device Establishment (Outline)



D-K

S

HTTPS

U1, pwd-U1

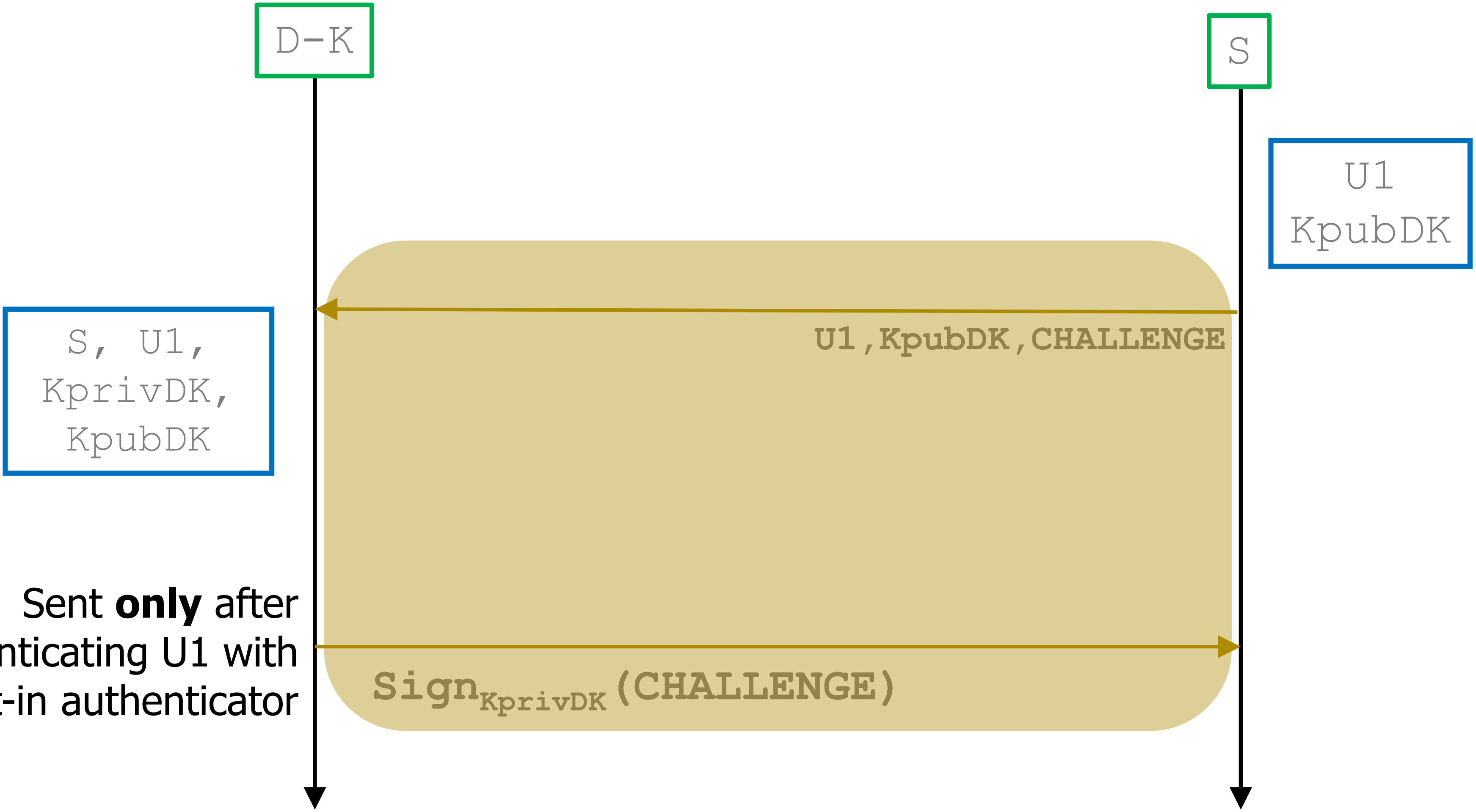
U1

pwd-U1

Trusted Device Establishment

D-K declares to S that it has a built-in authenticator

Passwordless Device Login (Outline)



Sent **only** after authenticating U1 with built-in authenticator

Passkeys in a nutshell

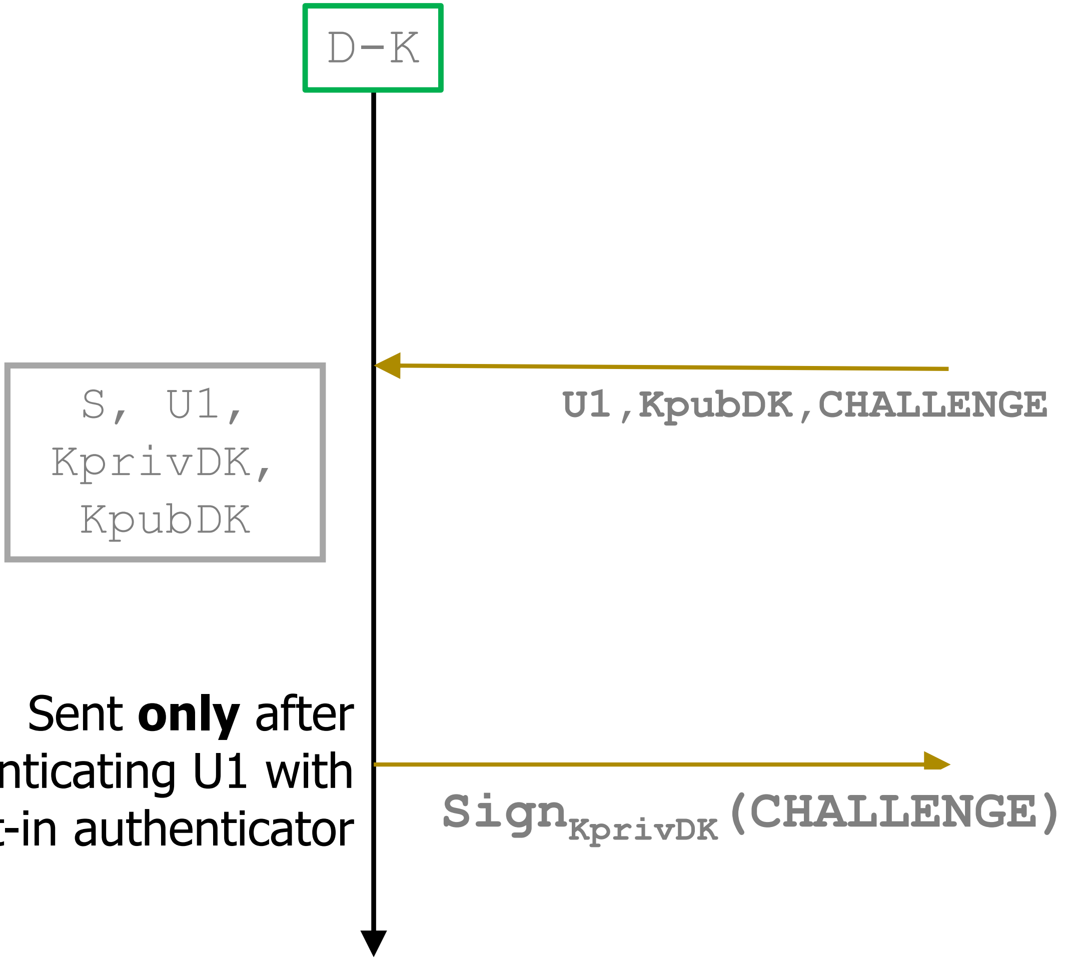
- Prevalent terminology:
  - D-K is trusted and passwordless for service S

  ≡

  - "**Passkey** for S stored on D-K"

- Passkey ≈ KPRIV-DK
- In certain cases, passkey can be migrated to other (trusted) devices

Hhmmm...



D-K

S, U1, KprivDK, KpubDK

U1,KpubDK,CHALLENGE

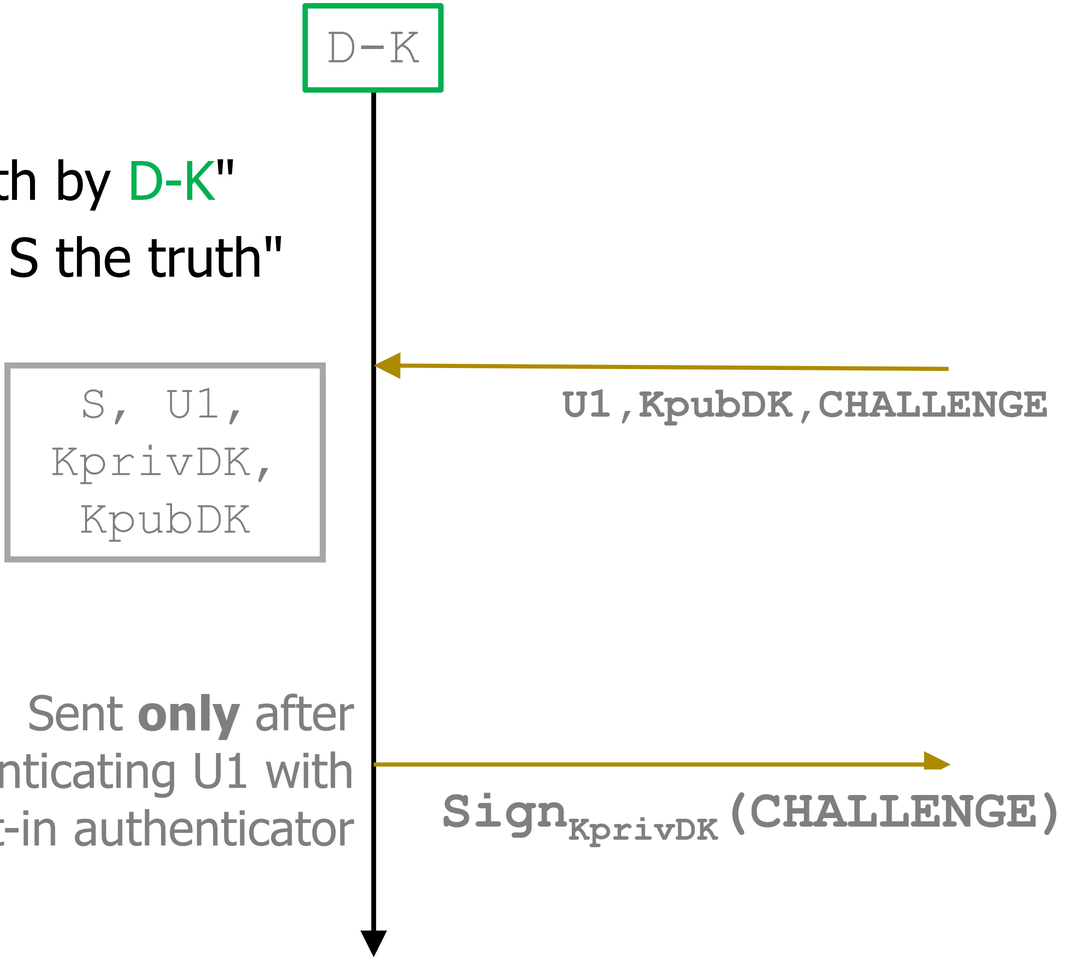
Sent **only** after authenticating U1 with built-in authenticator

Sign$_{KprivDK}$(CHALLENGE)

How can I be guaranteed?

TRUSTED device!

- User point of view:
  1. "I **trust** built-in auth by D-K"
  2. "I **trust** D-K to tell S the truth"



Sent **only** after authenticating U1 with built-in authenticator

Summary

- Trusted device
  - Proves knowledge of private key to S
- Passwordless device
  - Trusted device with biometric authenticator

- **Login** from trusted device
  1. Device proves knowledge of private key to S
  2. User proves knowledge of password to S
- **Login** from passwordless device
  1. Device proves knowledge of private key to S
  2. User proves biometric property **to device**

Passwordless is "more secure"!

- **Login** from trusted device
  2. User proves knowledge of password to S
- **Login** from passwordless device
  2. User proves biometric property **to device**

- Passwords are **phishable**
- **No** risk of disclosing password from passwordless device!

Passkey vs Long term Cookie

- IF Adversary can read information on D-K
- THEN They are "≈equivalent"

- Cookies:
  - Used after authentication
  - Transmitted in every request
  - Stored on Service
- Passkeys:
  - Used for authentication
  - Never leave the device