U2F (universal 2nd factor)

how security keys work

Michael Stapelberg

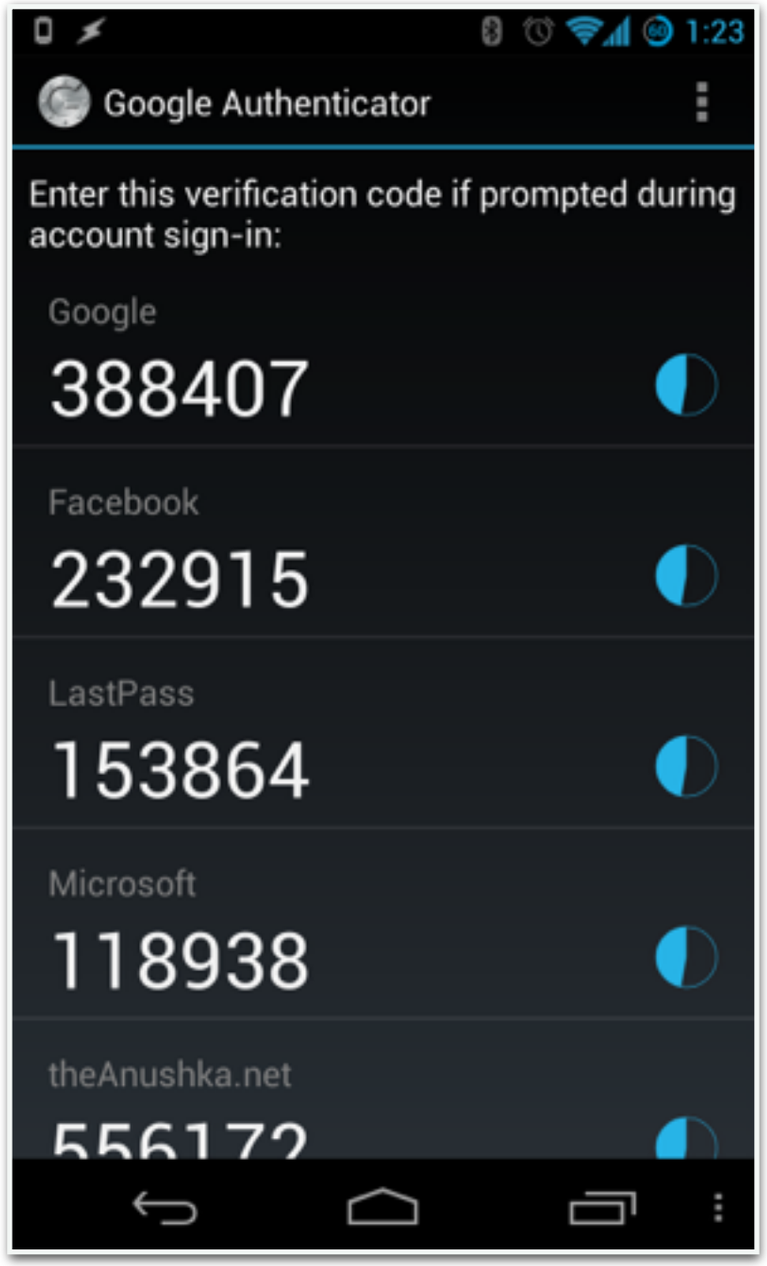
2014-10-30, NoName e.V.

Contents

- motivation for 2nd factor auth
- security keys from a user perspective
- yubico’s security key products
- the protocol in detail
- questions?

2nd factor auth

- user + pass = things you **know**
- username often (semi-)public
- passwords often weak, stolen, phished, …
- second factor: a thing you **have**
  - OTP (One Time Password)

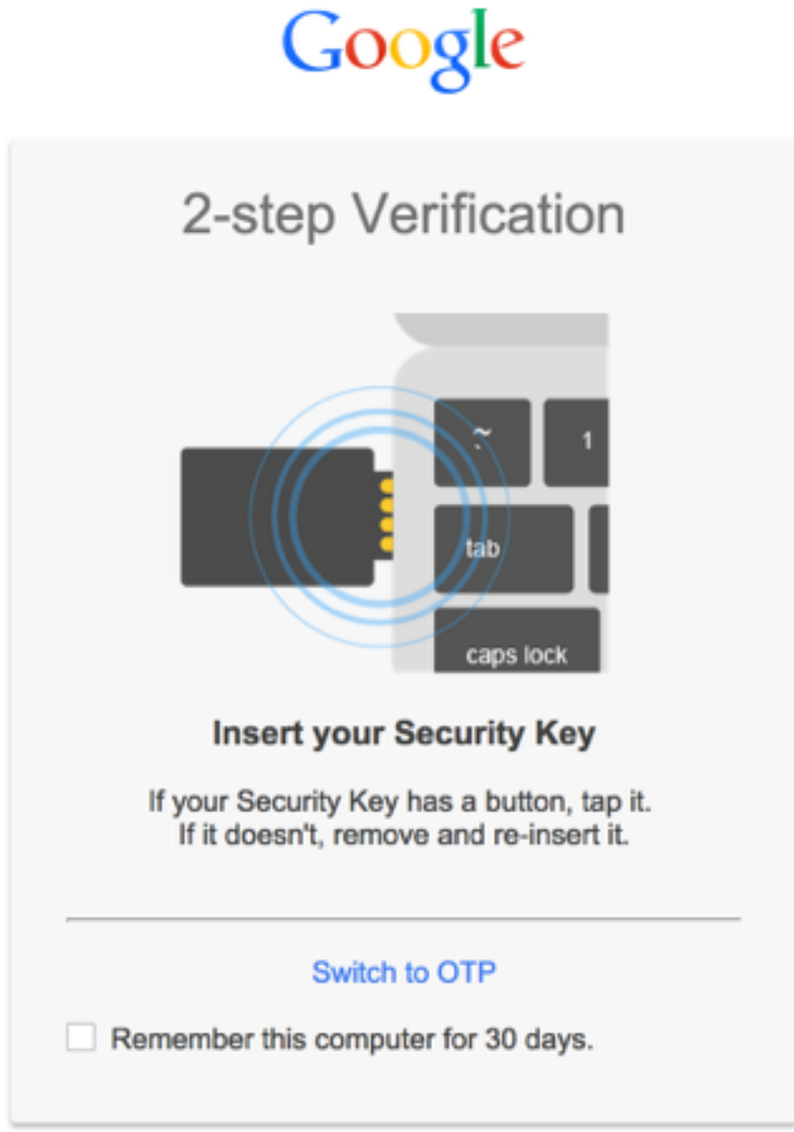


security keys for users

- when opting in, you register one or more security keys

- logging into google requires plugging in and touching the security key

- on phones: NFC instead of plugging in



Yubico’s security keys

18 USD

U2F

50 USD

OTP
U2F
NFC
SmartCard

60 USD

OTP
U2F

SmartCard

protocol in detail

- server, browser (or app), security key
- hashes: SHA256, signatures: ECDSA (on P-256)
- raw data: base64, structured data: JSON
- this presentation assumes you are familiar with the properties of hashes and signatures

registration: server

GET /user/zekjur/register

challenge	<32 bytes pseudo random>
version	U2F_V2
appId	http://sigrid.zekjur.net/ssh

registration: browser

challenge

<32 bytes pseudo random>

version

http://sigrid.zekjur.net/

appId

http://sigrid.zekjur.net/ssh

registration: security key

challenge <32 bytes PRNG>

origin http://sigrid.zekjur.net/

appId http://sigrid.zekjur.net/ssh

\x05

user pub key

len(key handle)

key handle

len(att cert)

attestation cert

signature

registration: security key

\x05

user pub key

len(key handle)

key handle

len(att cert)

attestation cert

signature

\x00

hash(appid)

hash(challenge, origin)

key handle

user pub key

registration: browser

POST /user/zekjur/register

clientData

challenge, origin, typ

registrationData

(from security key)

registration: server

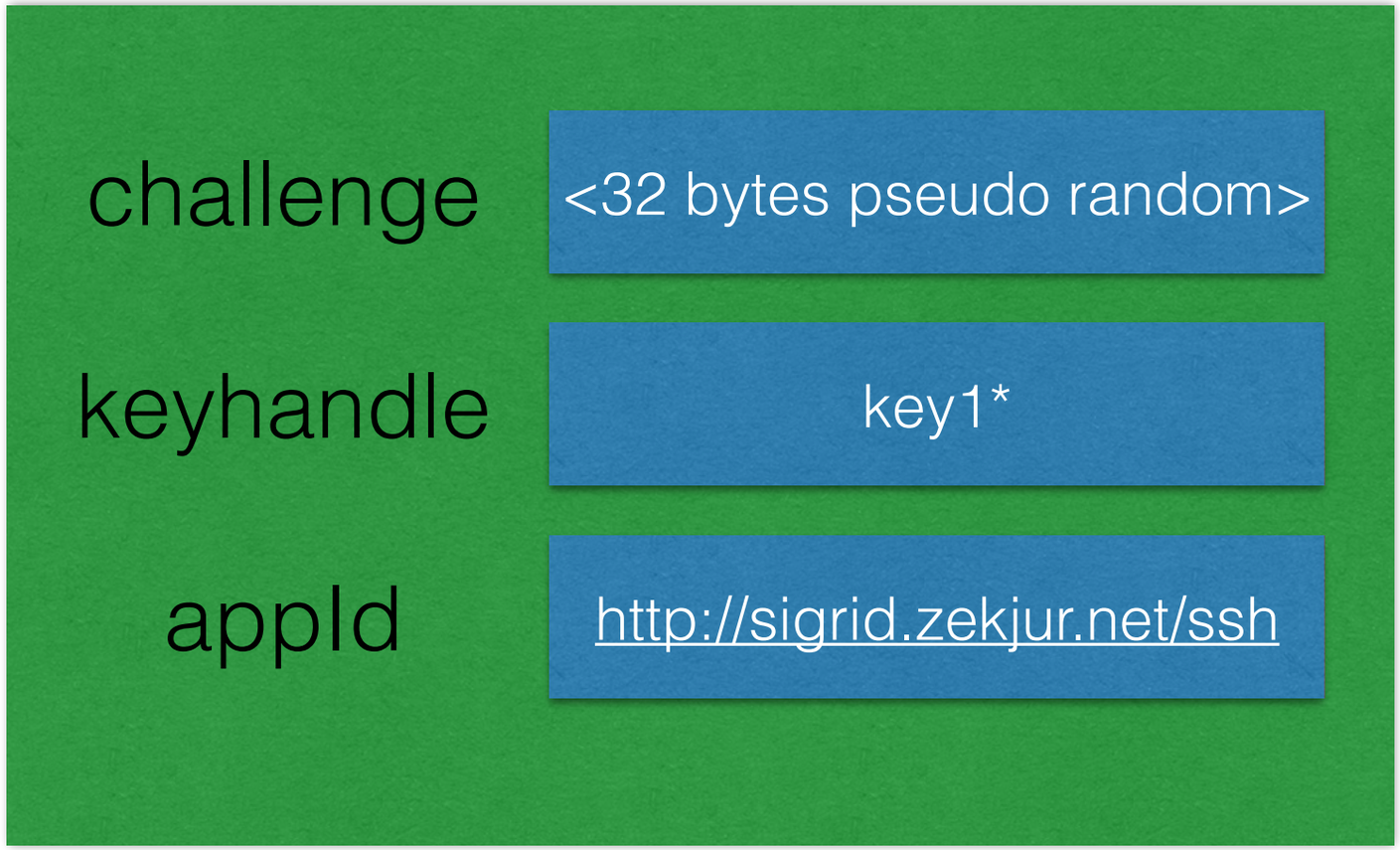
- check that attestation certificate is trusted
  - data was generated by a trusted security key
- verify signature
  - no attacker modified the appid or origin
- save user pub key and key handle

auth: server

GET /user/zekjur/login

challenge	<32 bytes pseudo random>
keyhandle	key1*
appId	http://sigrid.zekjur.net/ssh

auth: browser



auth: security key

challenge

<32 bytes pseudo

appId

http://sigrid.zekjur.net/ssh

keyhandle

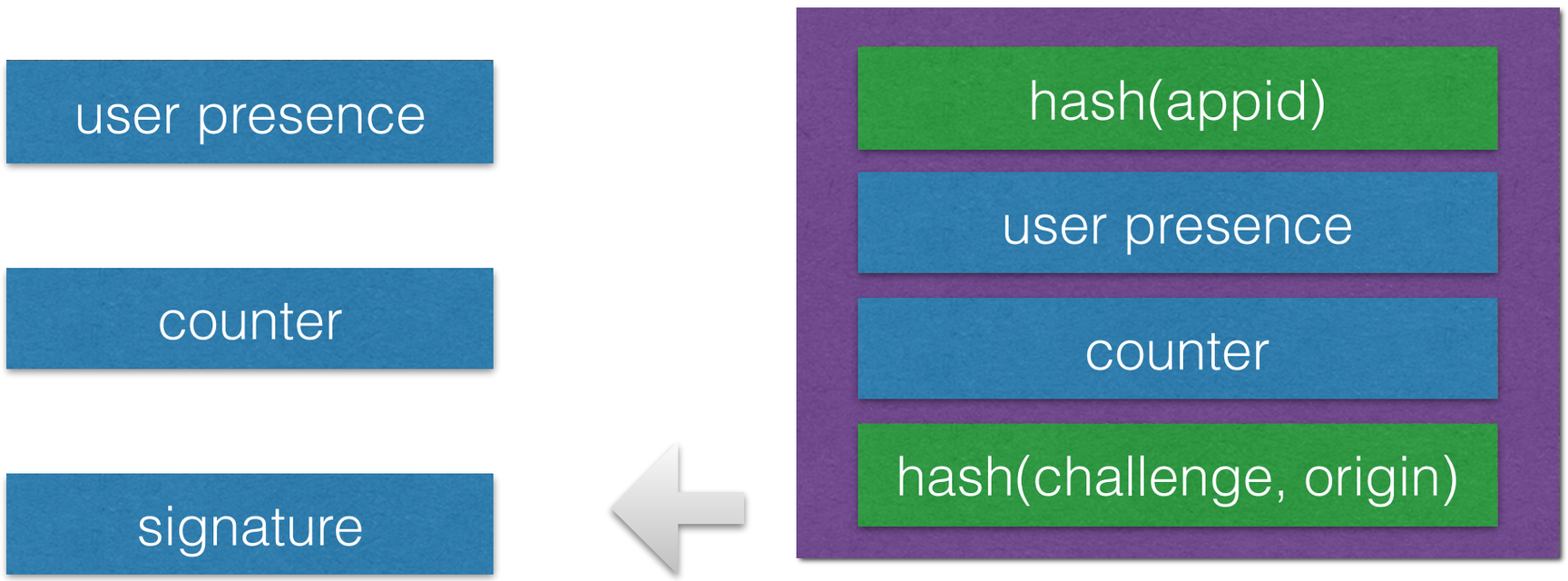
key1*

user presence

counter

signature

auth: security key



auth: browser

POST /user/zekjur/login

clientData	challenge, origin, typ
signatureData	(from security key)
keyHandle	key1*

auth: server

- verify signature (using **saved** pubkey!)
  - no attacker modified the appid or origin
    - definitely talking to the same security key
- verify user presence bit
- verify that counter is increasing

summary

- unless there is a wide-spread problem (think Debian OpenSSL)

- we can verify that the user **has** the registered (!) security key

- phishing becomes **a lot harder**, virtually impossible to do without the user being able to notice it (and it now requires malware)

questions?

- https://www.yubico.com/
- http://googleonlinesecurity.blogspot.ch/2014/10/strengthening-2-step-verification-with.html
- http://fidoalliance.org/specifications/download/
- https://github.com/yubico/?query=u2f