yubico

Universal 2nd Factor

2015-02-01
FOSDEM Security devroom
Simon Josefsson
simon@yubico.com

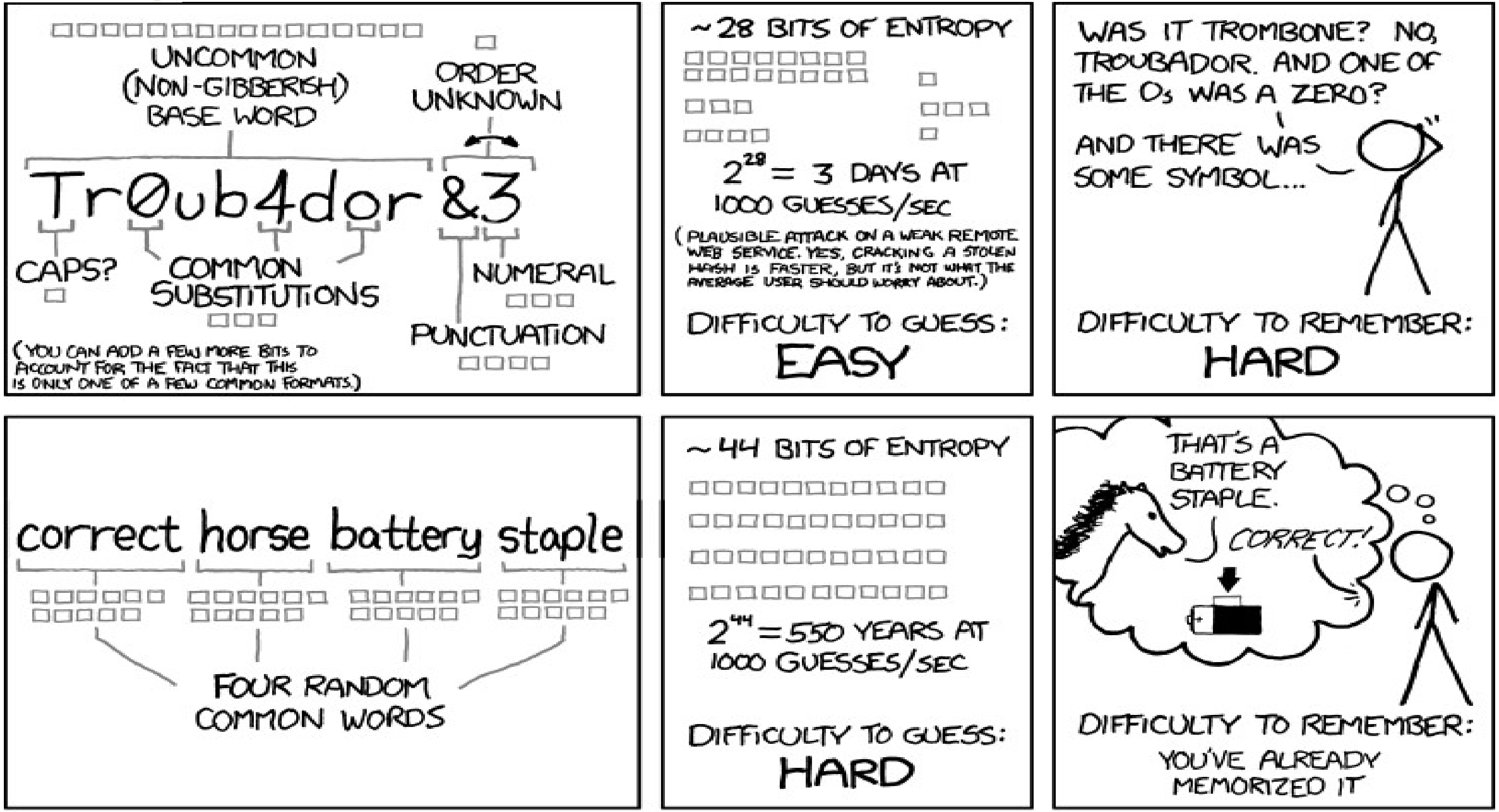
yubico

yubico
yubico

What is U2F?

yubico

xkcd

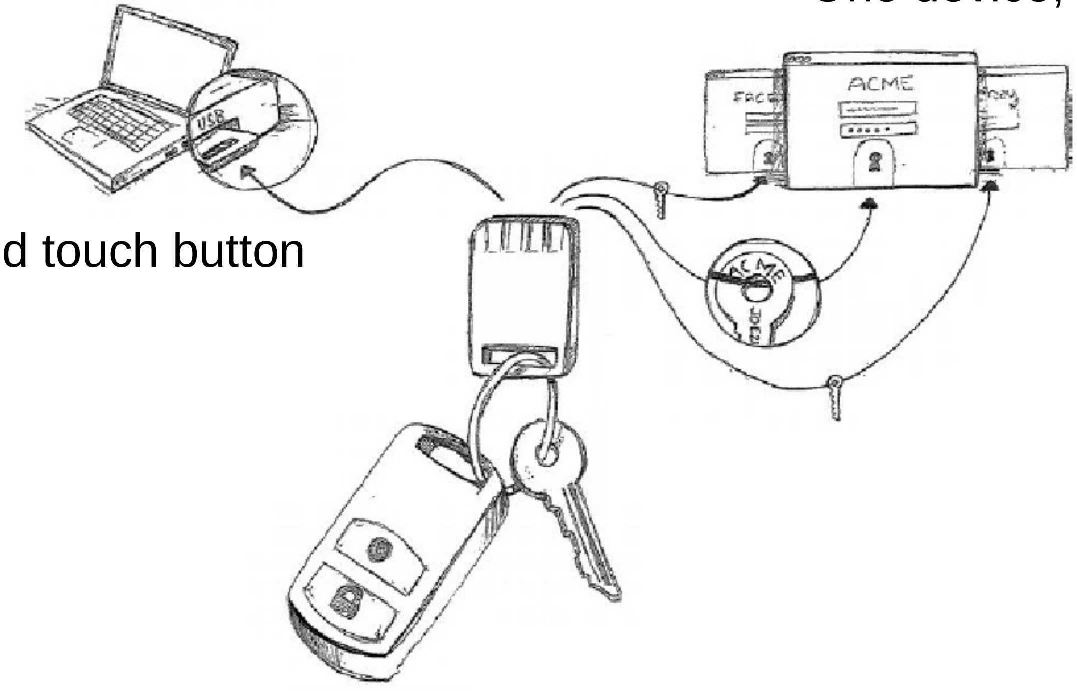

UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor &3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

co

The U2F solution

One device, many services



Easy: Insert and touch button

Safe: Unphishable Security

Pre-History of U2F: Gnubby

Yubico designed a precursor to U2F with Google and NXP. Deployed to Google staff around the world.

To reach mass market, standardization and multiple vendors are needed. During 2012 the FIDO Alliance started working on U2F.

Bank of America.

IdentityX

Google

SAMSUNG

PayPal™

DISCOVER

QUALCOMM

BlackBerry.

ARM

fido™
alliance

oberthur
TECHNOLOGIES
THE M COMPANY

lenovo. FOR THOSE WHO DO.™

yubico
Trust the Net.

Synaptics

Over 150 members

NXP

Microsoft

CrucialTec

MasterCard

Nok Nok
LABS

VISA

Alibaba Group

RSA

What is this U2F protocol?

Core idea: Standard public key cryptography

- User's device mints new key pair, ”registers” public key and key-handle with server
- Key handle contain data to restore private key on device
- Server provides key-handle and asks user's device to sign data to verify the user
- One device, many services - “**Bring Your Own Authenticator**”

Design considerations

- **Privacy:** Site-specific keys, no unique device ID
- **Security**: No phishing or man-in-the-middle, no soft private keys
- **Trust**: User decides what authenticator to use
- **Pragmatics**: Affordable today
- **Usability**: No delays, fast crypto on device, no driver installs

yubico

Think:

Driverless smartcard for the modern consumer web, plus privacy

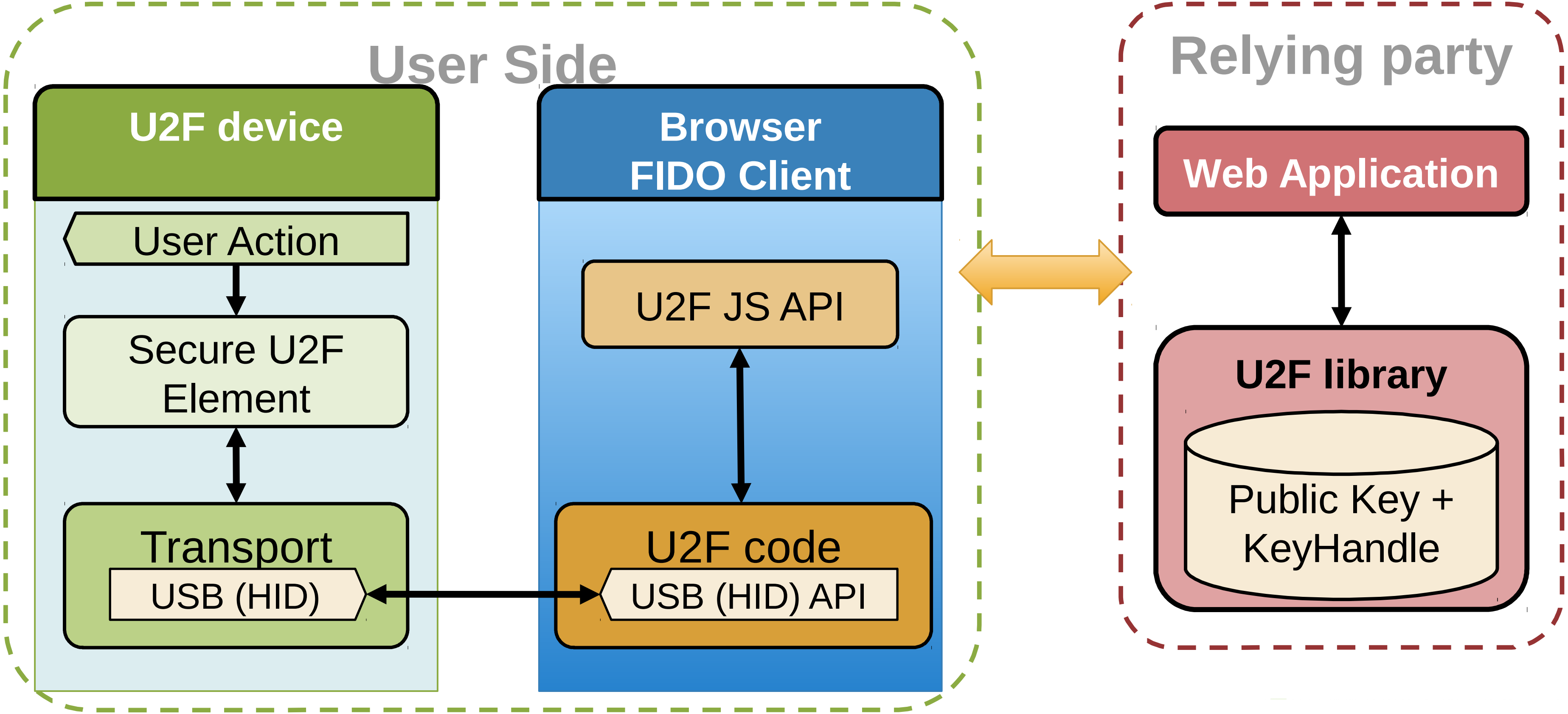
USB today, the world tomorrow

Bluetooth® SMART

NFC

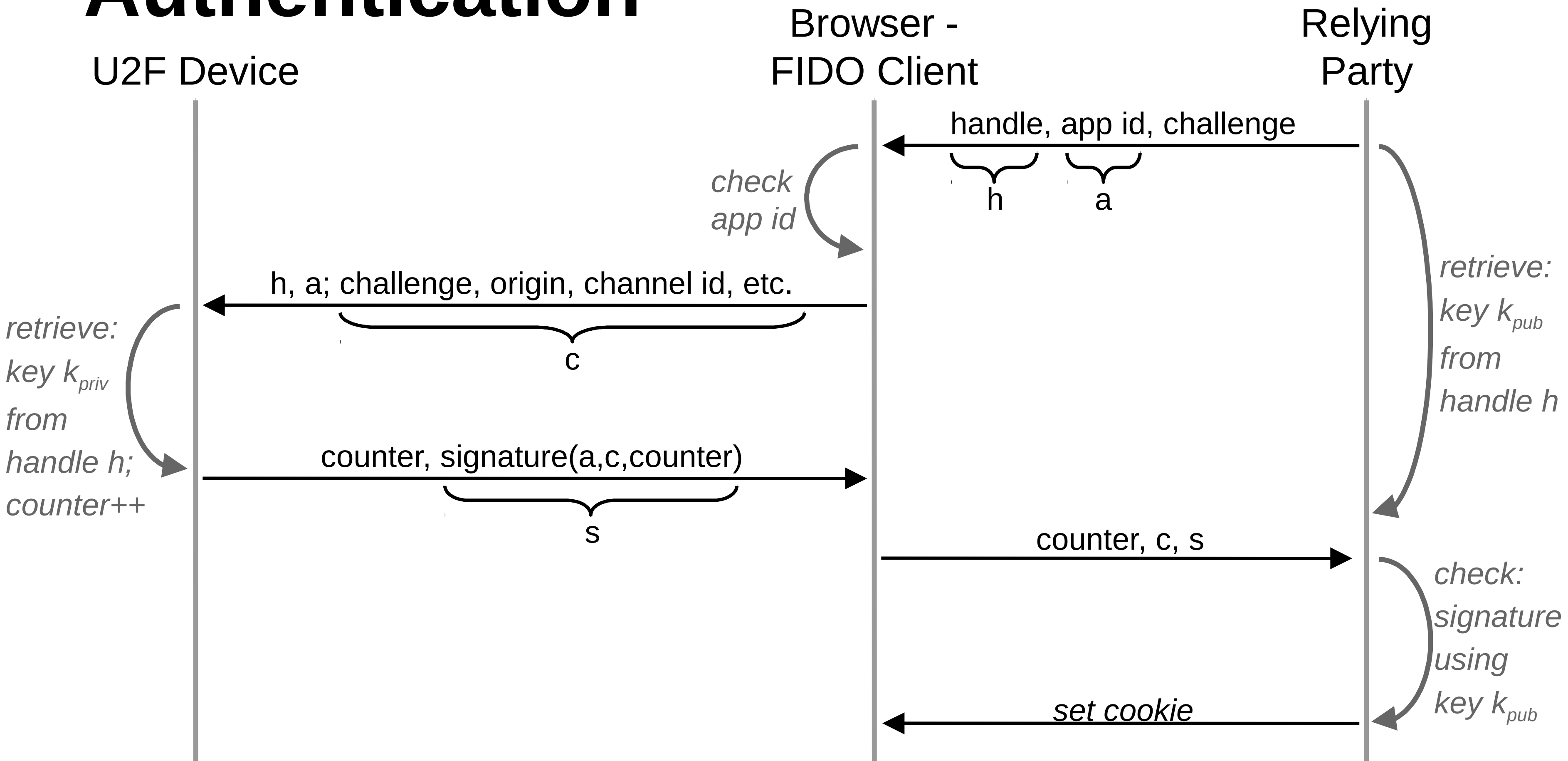
Hardware separation important! Software in complex hosts too fragile → keys stolen on 0day vuln.

U2F entities



Demo

Authentication



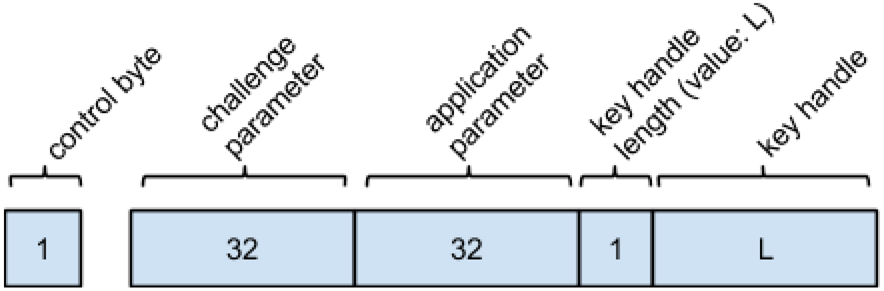
U2F Device
Browser - FIDO Client
Relying Party
handle, app id, challenge
h
a
check app id
retrieve: key k_{pub} from handle h
h, a; challenge, origin, channel id, etc.
c
retrieve: key k_{priv} from handle h; counter++
counter, signature(a,c,counter)
s
counter, c, s
check: signature using key k_{pub}
set cookie

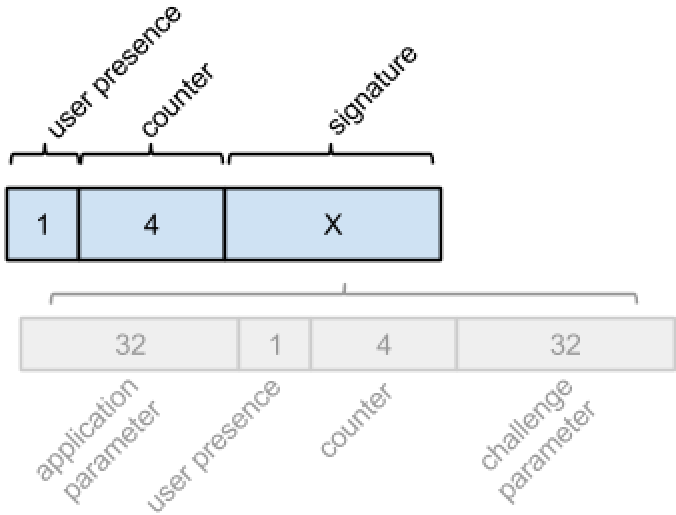
U2F Authentication JSON blobs

Server sends: { "keyHandle": "yQ_cxLOEDDrQ1rGesE249-QYNjGoNWpY2QRSQzE9p0qQZNk2i3Z6ioYAAumOZnJQhuQDJ2VVtOcUD85kYRdjuQ", "version": "U2F_V2", "challenge": "cDftdgcY3SOYMaKPq6JFt0nmpFACTZuJ5EbRr-VTnxA", "appId": "http:\/\/example.org" }

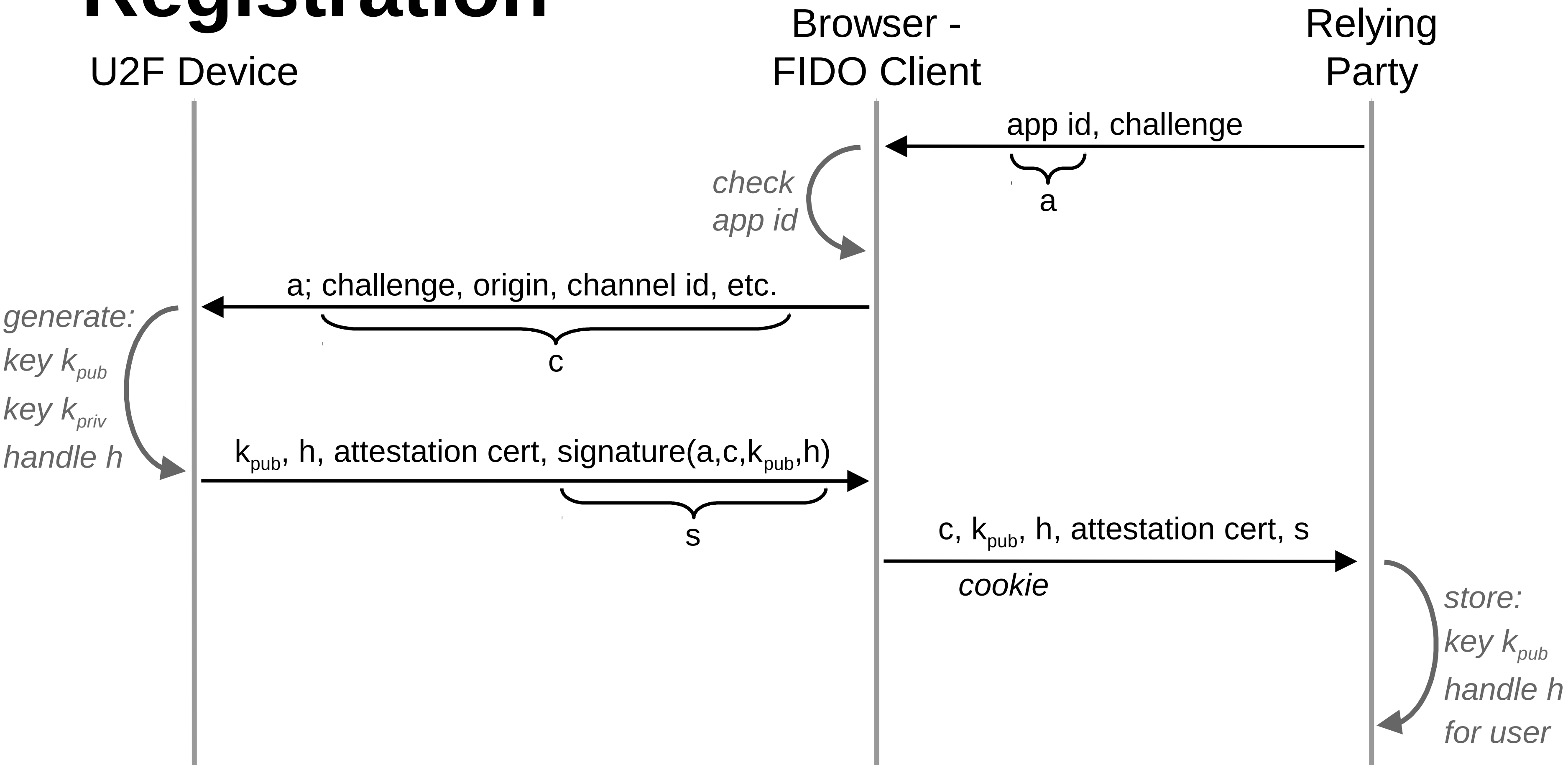
Client responds: { "signatureData": "AQAAADMwRgIhAKCAGKKDcZe1Rt4HdOnD2JkF5yU711AxjngH_-dW9-e5AiEAylw5kzYKRg2rSl0JU1zsJibF3MIWtOCXGv1h4KazCys=", "clientData": "eyAiY2hhbGxlbmdlIjogImNEZnRkZ2NZM1NPWU1hS1BxNkpGdDBubXBGQUNUWnVKNUViUnItVlRueEEiLCAib3JpZ2luIjogImh0dHA6XC9cL2V4YW1wbGUub3JnIiwgInR5cCI6ICJuYXZpZ2F0b3IuaWQuZ2V0QXNzZXJ0aW9uIiB9", "keyHandle": "yQ_cxLOEDDrQ1rGesE249-QYNjGoNWpY2QRSQzE9p0qQZNk2i3Z6ioYAAumOZnJQhuQDJ2VVtOcUD85kYRdjuQ" }

USB HID Authenticate





Registration



U2F Device

Browser - FIDO Client

Relying Party

app id, challenge

a

check app id

a; challenge, origin, channel id, etc.

c

generate:
key k_{pub}
key k_{priv}
handle h

k_{pub}, h, attestation cert, signature(a,c,k_{pub},h)

s

c, k_{pub}, h, attestation cert, s

cookie

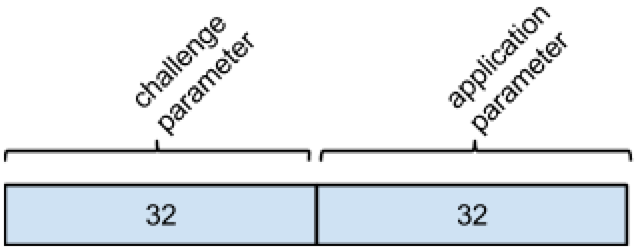
store:
key k_{pub}
handle h
for user

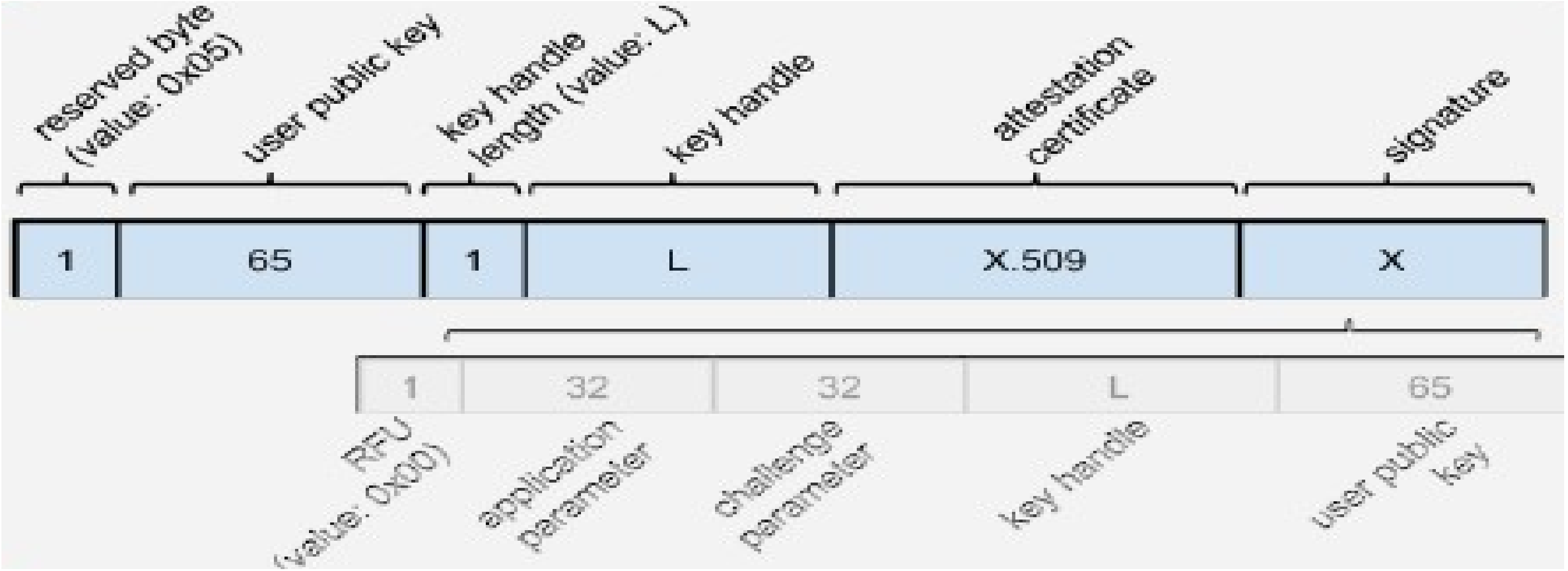
U2F Register JSON blobs

Server sends: { "challenge": "oVXT29EiA16cFFIQCzwPp-waGiMahI2WIevJXcFQCVc", "version": "U2F_V2", "appId": "http:\/\/example.org" }

Client responds: { "registrationData": "BQQ91soQ8zQlX-yBzGJtOWMvKbWPkIsOqA_1psdwK7fid03vAXcDreXFFgcYEaxI5dUyWcs3jiw67Z_D0KxZMTP2QMkP3MSzhAw60NaxnrBNuPfkGDYxqDVqWNkEUkMxPadKkGTZNot2eoqGAALpjmZyUIbkAydlVbTnFA_OZGEXY7kwg...W_AMRED0ExAGowC0YQMvgbqWGZiZAiBUt00SBB1TTtFfbwr4Lp1daS5L6gqMQxtiHIrHjZwFKw==", "clientData": "eyAiY2hhbGxlbmdlIjogIm9WWFQyOUVpQTE2Y0ZGSVFDendQcC13YUdpTWFoSTJXSWV2SlhjRlFDVmMiLCAib3JpZ2luIjogImh0dHA6XC9cL2V4YW1wbGUub3JnIiwgInR5cCI6ICJuYXZpZ2F0b3IuaWQuZmluaXNoRW5yb2xsbWVudCIgfQ==" }

Registration: USB HID





Application and Facet ID's

Application

A set of functionality provided by a common entity (the application owner), and perceived by the user as belonging together. For example, *PayPal* is an application that allows users to pay for stuff.

Facets

An (application) facet is how an application is implemented on various platforms. For example, the application PayPal may have an Android app, an iOS app, and a Web app. These are all facets of the PayPal application.

Facet ID

A platform-specific identifier (URI) for an application facet. Simplest case: facet id and application id is the same.

- For the Web, the Facet ID is the web origin, written as a URI without a path (e.g.https://login.paypal.com).

- For Android, the Facet ID is the URI android:apk-key-hash:<hash-of-apk-signing-cert>.

- For iOS, the Facet ID is the URI ios:bundle-id:<ios-bundle-id-of-app>.

What if I want to support U2F?

- Server/Browser: Call Javascript APIs
  - Send key handle in HTML/JavaScript to browser
- Server: Implement registration flow
  - Decide how to handle attestation certificates
  - Verify registration response
  - Store public key, key handle with user account
- Server: Implement login flow
  - Check username/password, look up key handle
  - Verify authentication response (origin, signature, counter, …)
- Relying Party: Check your account recovery flow

So many keys...

- Authentication public/private key
  - Unique for every RP
  - Generated during U2F Registration
  - Public key sent to RP during Registration
  - Key handle can be used to derive private key
    - Unlimited number of RPs on small device
  - Hard coded to ECDSA using NIST P.256 curve

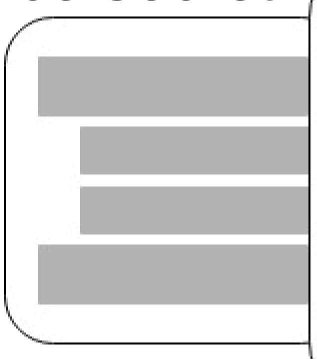
yubico

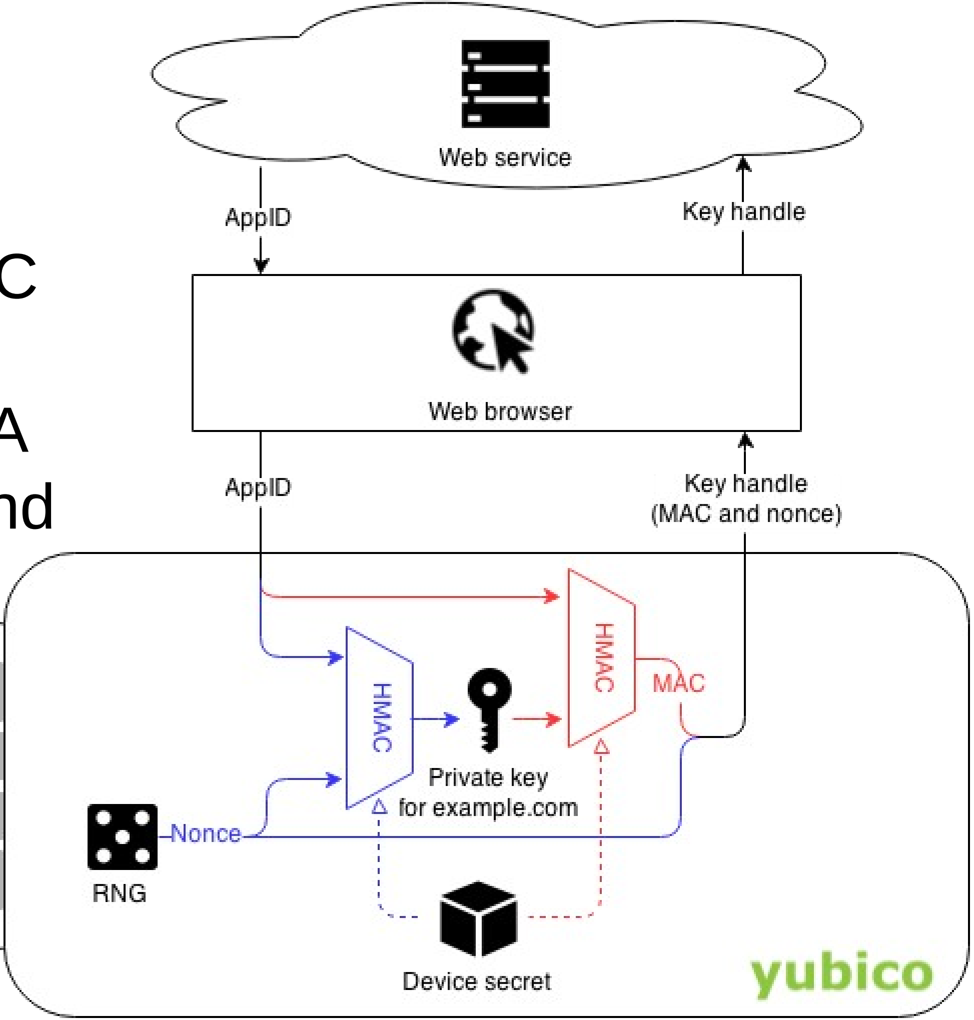
So many keys...

- Device-unique symmetric secret
  - Unwrap/derive per-RP ECDSA key from key handle
  - Unique random key for every device
  - Yubico derives private key using HMAC-SHA256

Yubico’s U2F KeyHandle

- Key handle is nonce+MAC instead of encrypted
- Device can derive ECDSA private key from nonce and symmetric device secret
- MAC detects invalid
  key handle or malicious RP



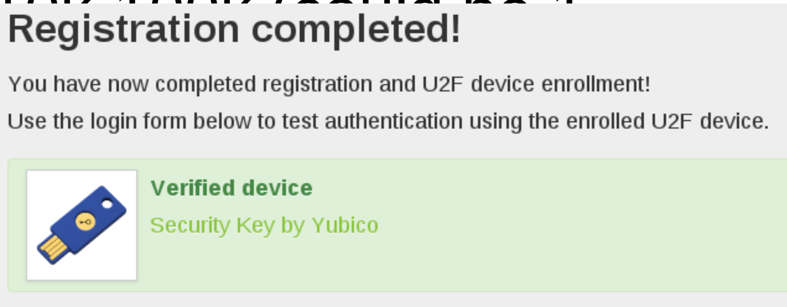


So many keys...

- ECDSA attestation key (unique per batch)
  - Linked with device attestation certificate
  - Signs U2F Registration blobs

U2F attestation

- Proves what U2F device the user used
- X.509 Certificate with batch-unique key
- Why batch-unique and not device-unique?
    - Privacy: device-unique key permits conspiring RPs to link a physical key to particular user
    - Common batch size could be 10k-100k (could be 1 breaking the privacy aspects)



Registration completed!

You have now completed registration and U2F device enrollment!

Use the login form below to test authentication using the enrolled U2F device.

Verified device

Security Key by Yubico

Yubico U2F software

Our idea is to publish host and server libraries in common languages as FOSS code

- C: libu2f-host & libu2f-server
- Java: java-u2flib-server
- PHP: php-u2flib-server
- Python: python-u2flib-host & python-u2flib-server

yubico

U2F C Libraries

- github.com/Yubico/libu2f-{server,host}
- Portable C99 few dependencies (json, OpenSSL, HIDAPI)
- server: Generate U2F challenges and verify responses
- host: Parse challenges and talk USB to get responses
- Command line tool

Resources

Libraries, Plugins, Sample Code, Documentation	developers.yubico.com/U2F
U2F Protocol Specification	fidoalliance.org/specifications
Yubico U2F Demo Server - Test your U2F device here!	demo.yubico.com/u2f

Thank you!

yubico

y