– SLIDE 01 / TITLE –

Open Source FIDO tokens

Why?

Ivan Zorin · Independent Researcher · 26.05.2026

$ whoami

- System Engineer
- Opensoursini Contributini (IronOS, HydraFW, etc.)
- Independent Researcher
- Conference Speaker (phdays, offzone, meetups)
- Community Enjoyer ❤
  - Free Software ideology
  - Right to Repair movement
  - Hackerspace culture

AGENDA

01 · FIDO token

02 · Mainstream features

03 · Alternatives

04 · ???

05 · PROFIT

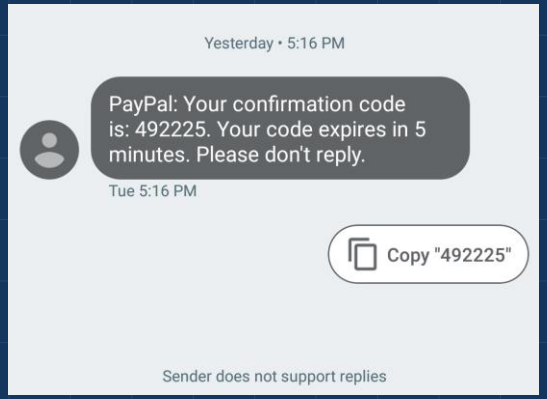
DISCLAIMERS

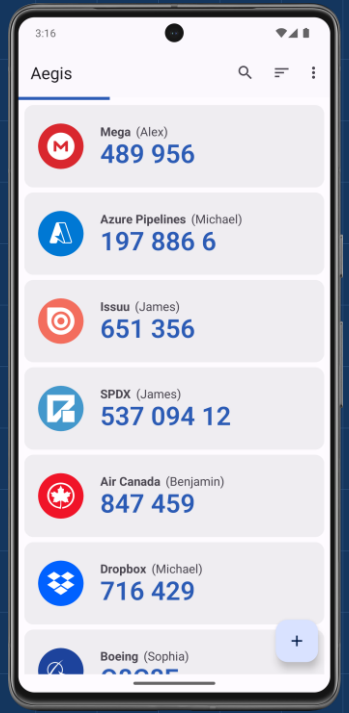
- ~~*100%*~~ *99%** AI/ML/LLM free
- *0/1/n-day* free
- *crypto* - cryptography
- *token* - 2FA device

FIDO token

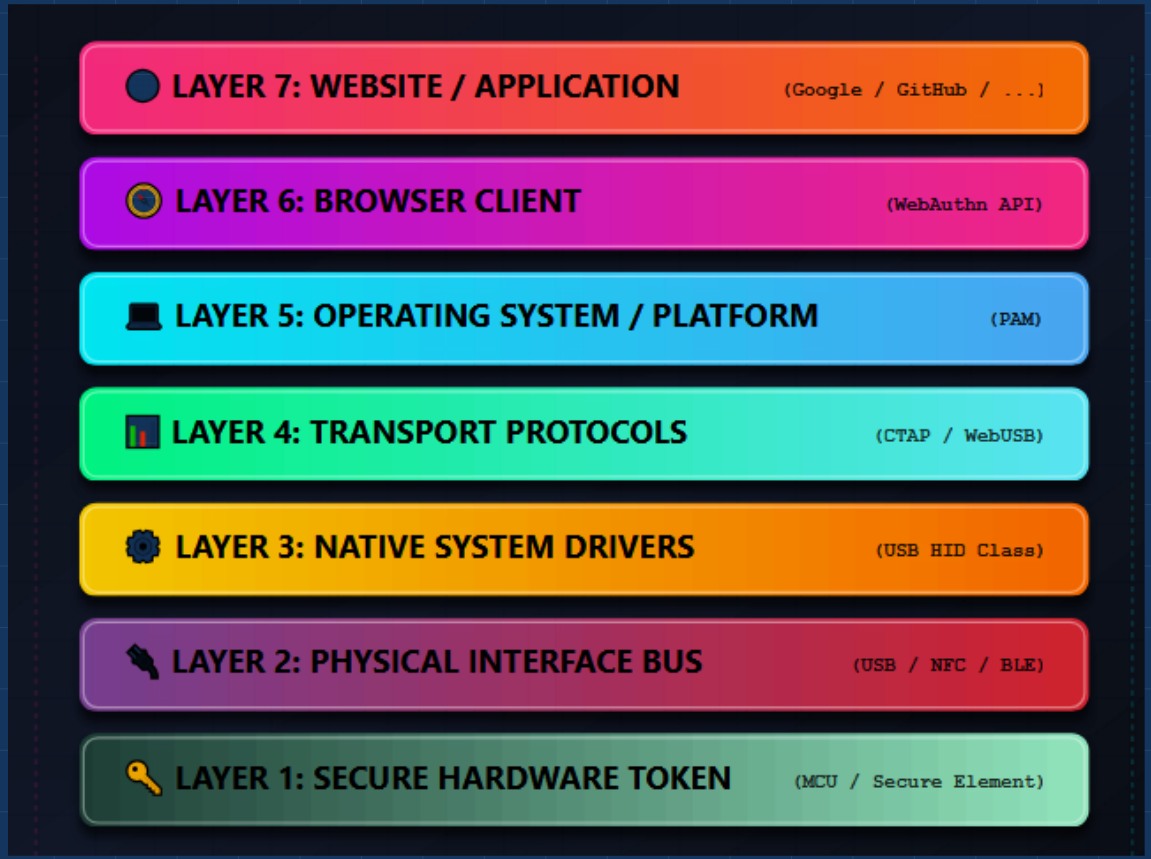
token = hardware
+ firmware
+ API/libraries
+ software tools
+ 3rd party software

FIDO token – why?





FIDO token – how?


LAYER 7: WEBSITE / APPLICATION
(Google / GitHub / ...)
LAYER 6: BROWSER CLIENT
(WebAuthn API)
LAYER 5: OPERATING SYSTEM / PLATFORM
(PAM)
LAYER 4: TRANSPORT PROTOCOLS
(CTAP / WebUSB)
LAYER 3: NATIVE SYSTEM DRIVERS
(USB HID Class)
LAYER 2: PHYSICAL INTERFACE BUS
(USB / NFC / BLE)
LAYER 1: SECURE HARDWARE TOKEN
(MCU / Secure Element)

FIDO token – how?

RESOURCE

TOKEN

FIDO token – how: init

RESOURCE TOKEN

<<< registration <<<

FIDO token – how: simplified

RESOURCE TOKEN

FIDO token – how: VERY simplified

RESOURCE TOKEN

>>> challenge >>>

FIDO token – how: VERY SIMPLIFIED

RESOURCE TOKEN

>>> challenge >>>

<<< response = HMAC(challenge + key)

FIDO token – how: VERY VERY simplified

RESOURCE TOKEN

>>> challenge >>>

<<< response = HMAC(challenge + key)

result = verify(response);

FIDO token – how: VERY VERY simplified

RESOURCE TOKEN

>>> challenge >>>

<<< response = HMAC(challenge + key)

result = verify(response);

FIDO token – how: VERY VERY simplified

RESOURCE TOKEN

>>> challenge >>>

<<< response = HMAC(challenge + key)

result = verify(response);

FIDO token – смотря какой vendor


Б И

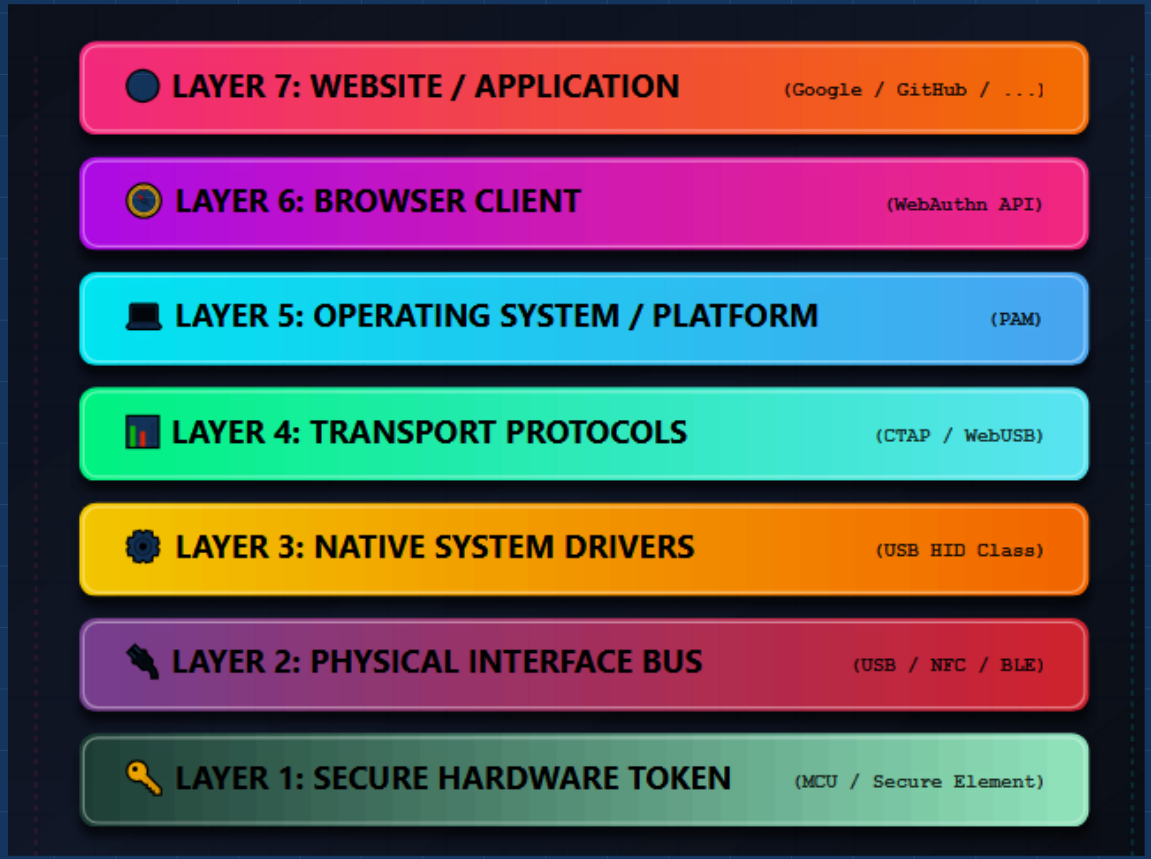
ЮБИК (~2007)

PROTOCOLS

- FIDO2 / passwordless
- FIDO / U2F
- OATH (TOTP / HOTP)
- GPG / PGP
- …

CRYPTO

- RSA
- ECC
- Hashes
- …


LAYER 7: WEBSITE / APPLICATION
(Google / GitHub / ...)
LAYER 6: BROWSER CLIENT
(WebAuthn API)
LAYER 5: OPERATING SYSTEM / PLATFORM
(PAM)
LAYER 4: TRANSPORT PROTOCOLS
(CTAP / WebUSB)
LAYER 3: NATIVE SYSTEM DRIVERS
(USB HID Class)
LAYER 2: PHYSICAL INTERFACE BUS
(USB / NFC / BLE)
LAYER 1: SECURE HARDWARE TOKEN
(MCU / Secure Element)

~~ЧЕРНОВИК:~~

~~Сделать подводку к не-секте~~

~~Open Source~~ Free Software

PRINCIPLES

- Freedom 0: to run the program as you wish
- Freedom 1: to study & to modify the program
- Freedom 2: to redistribute copies
- Freedom 3: to distribute copies of your modified versions

Open Source Hardware

PRINCIPLES

- Open Design
- Free Schematics [from NDA]
- Available Datasheets

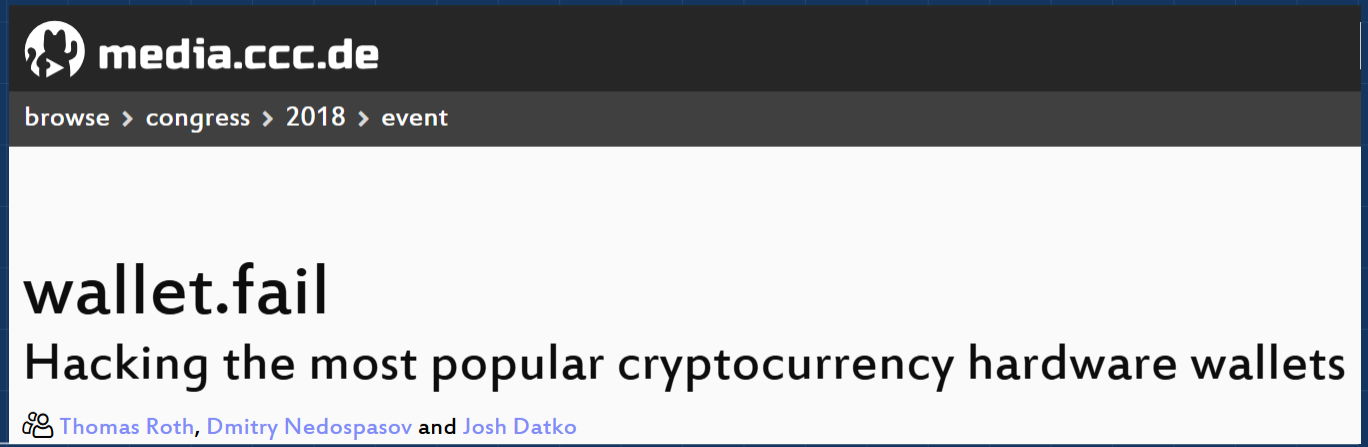
Yubikey alternatives

Yubikey alternatives – out of scope:

- Librem key
- Onlykey
- Trezor
- ~~Ledger~~

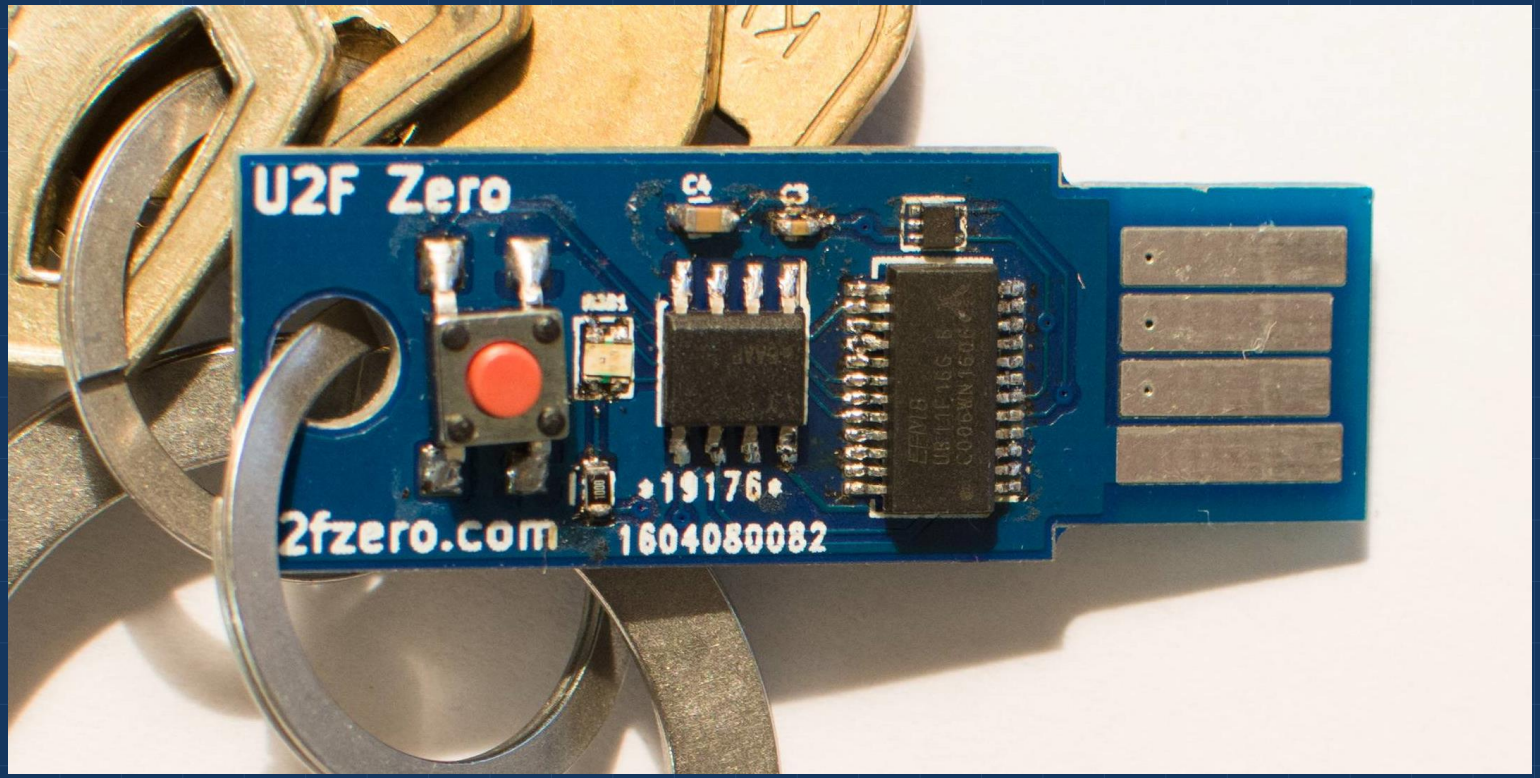
Yubikey alternatives - out of scope:

- Librem key
- Onlykey
- Trezor
- ~~Ledger~~



Yubikey alternatives:

- Solokeys
- OpenSK
- Nitrokey
- GNUk


U2F Zero
C1
C3
R3
•19176•
2fzero.com
1604080082

SMT Parts

You need these 8 surface mount parts per U2F token which can all be purchased from Digikey or Mouser (totals ~$3 per board):

- Microcontroller (E0)
  - Digikey: EFM8UB11F16G-C-QSOP24
  - Mouser: EFM8UB11F16G-C-QSOP24
- Secure element (A1):
  - Digikey: ATECC508A-SOIC8
  - Mouser: 556-ATECC508A-SSHDAB
- [RGB LED] for status indication (RGB1):
  - Digikey: Cree Inc. CLVBA-FKA
- 100 Ohm current limiting resistor 0805 (R1):
  - Digikey: Vishay Dale CRCW0805100RFKEA
- Zener diode for ESD protection (Z1):
  - Digikey: Toshiba DF5A5.6FU
- Push button for user input (SW1):
  - Digikey: E-Switch TL3305AF260QG
  - Mouser: E-Switch TL3305AF260QG
- 4.7 uF bypass capacitor (C4) and 0.1 uF bypass capacitor (C3). All 0805.

About the Team

SoloKeys

Laurel, MD · solokeys.com

At SoloKeys, we make open source hardware for secure applications.

Conor Patrick

conorpp

Nicolas Stalder

nickraystalder

nickray

Emanuele Cesena

0x0ece

0x0ece

Haden Patrick

Solokeys v1: ~2018

HARDWARE

- STM32 F432
- USB A/C, USB & NFC

FIRMWARE

- C / Bare metal
- Python (tooling)

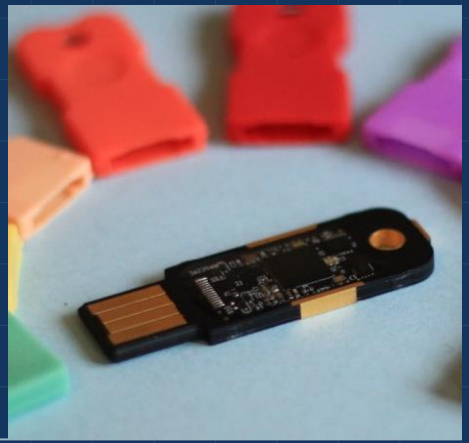
Solokeys v2: ~2021

HARDWARE

- NXP LPC55S69
- USB A/C, USB & NFC

FIRMWARE

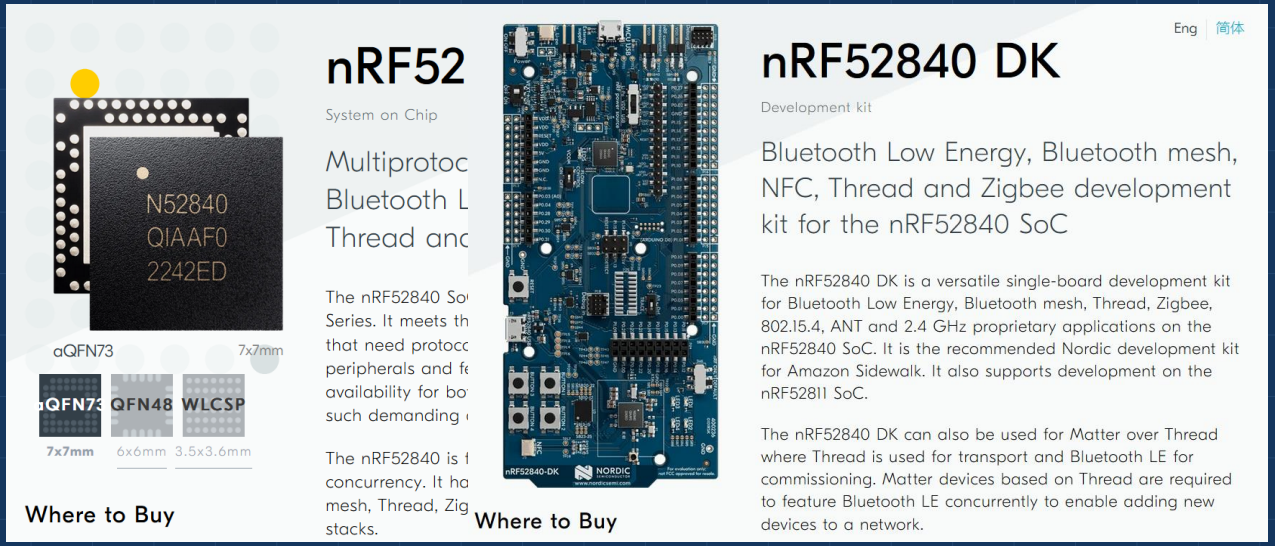
- Rust / Trussed
- Tooling



OpenSK

OpenSK (by Google)

OpenSK (by Google)



OpenSK (by Google)

nRF52

System on Chip

Multiprotoc
Bluetooth L
Thread anc

The nRF52840 So
Series. It meets th
that need protocc
peripherals and fe
availability for bo
such demanding

The nRF52840 is
concurrency. It ha
mesh, Thread, Zig
stacks.

Where to Buy

nRF528

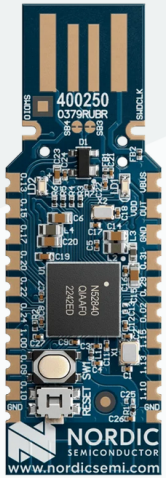
Development kit

Bluetooth Lo
NFC, Thread
kit for the nR

The nRF52840 DK is
for Bluetooth Low En
802.15.4, ANT and 2.
nRF52840 SoC. It is t
for Amazon Sidewalk
nRF52811 SoC.

The nRF52840 DK co
where Thread is usec
commissioning. Matt
to feature Bluetooth
devices to a network

Where to Buy



Eng | 简体

nRF52840 Dongle

nRF Connect for Desktop hardware

Designed for nRF Connect for Desktop

The nRF52840 Dongle is a small, low-cost USB dongle that supports Bluetooth 5.4, Bluetooth mesh, Thread, Zigbee, 802.15.4, ANT and 2.4 GHz proprietary protocols. The Dongle is the perfect target hardware for use with nRF Connect for Desktop as it is low-cost but still support all the short range wireless standards used with Nordic devices. The dongle has been designed to be used as a wireless HW device together with nRF Connect for Desktop. For other use cases please do note that there is no debug support on the Dongle, only support for programming the device and communicating through USB.

It is supported by most of the nRF Connect for Desktop apps and will automatically be programmed if needed. In addition custom applications can be compiled and downloaded to the

OpenSK (by Google)

Google Security Blog

The latest news and insights from Google on security and safety on the Internet

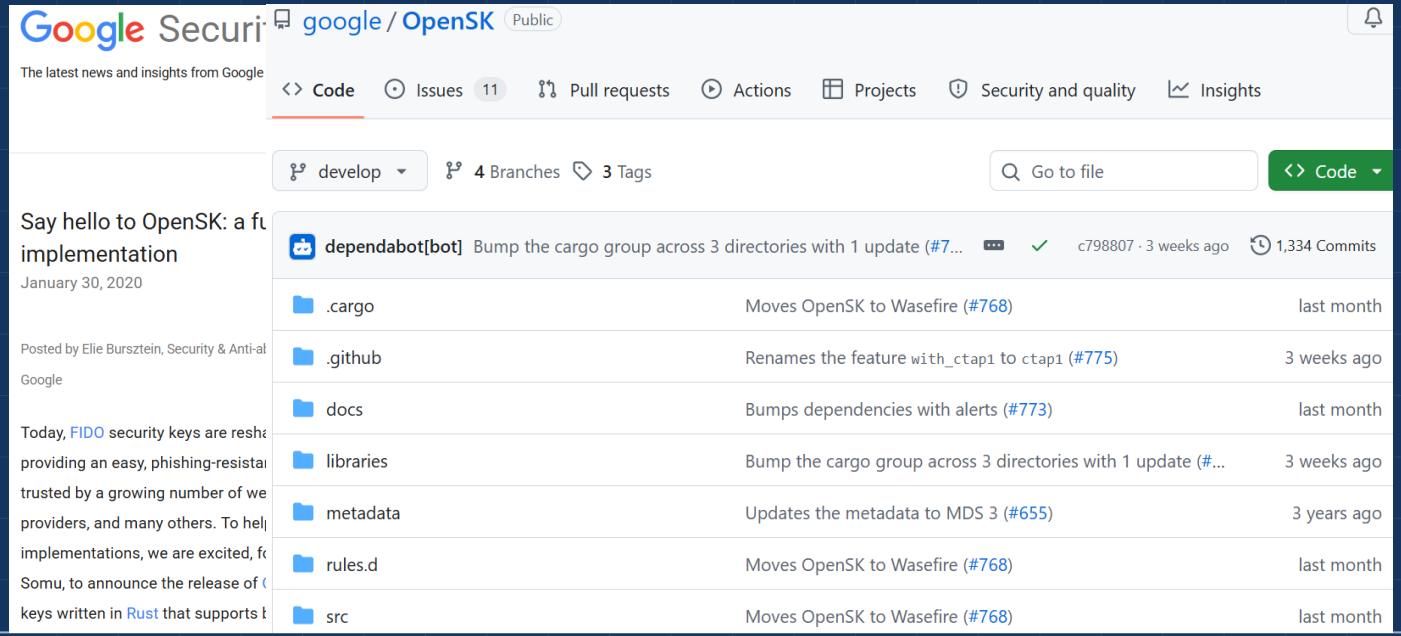
Say hello to OpenSK: a fully open-source security key implementation

January 30, 2020

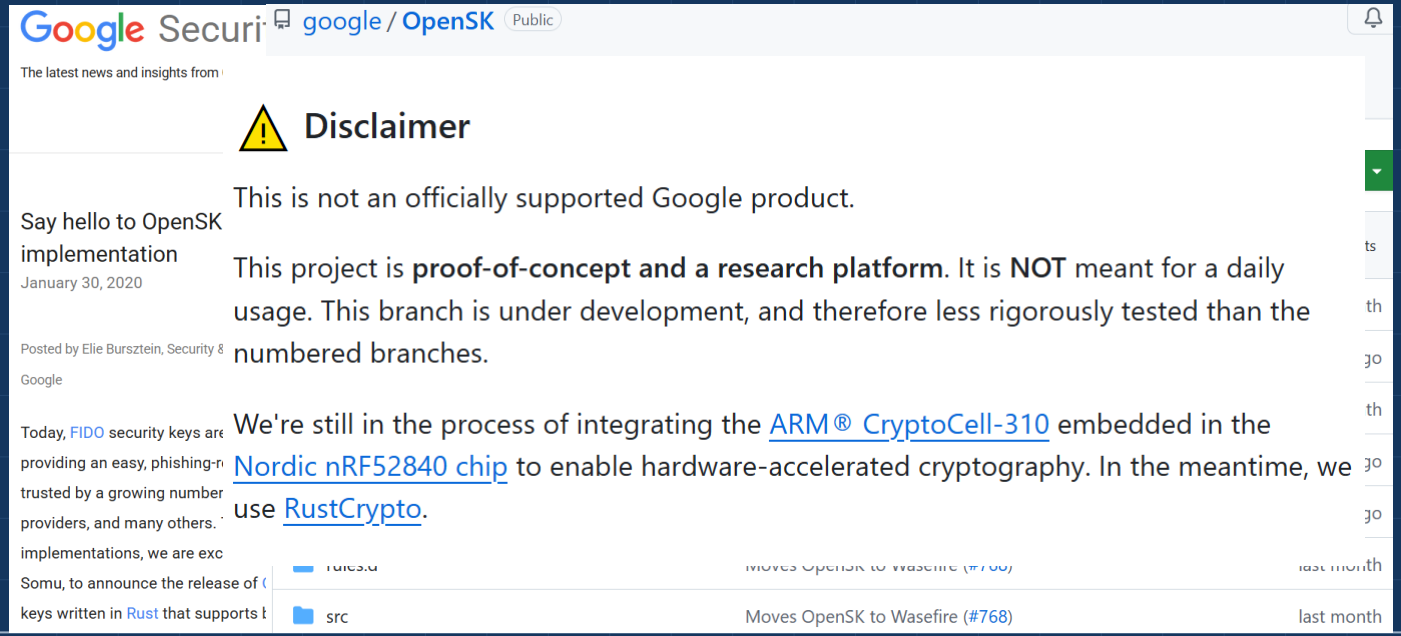
Posted by Elie Bursztein, Security & Anti-abuse Research Lead, and Jean-Michel Picod, Software Engineer, Google

Today, FIDO security keys are reshaping the way online accounts are protected by providing an easy, phishing-resistant form of two-factor authentication (2FA) that is trusted by a growing number of websites, including Google, social networks, cloud providers, and many others. To help advance and improve access to FIDO authenticator implementations, we are excited, following other open-source projects like Solo and Somu, to announce the release of OpenSK, an open-source implementation for security keys written in Rust that supports both FIDO U2F and FIDO2 standards.

OpenSK (by Google)



OpenSK ~~(by Google)~~



OpenSK: ~2020

HARDWARE

· Nordic nRF52840

FIRMWARE

· Rust / TockOS

Google

Google

Feitian-based

- Titan Security Key (~2018)
- Titan Security Key (~2023)

Nitrokey (est. 2009)

Nitrokey 3 (~2021):

HARDWARE

- NXP LPC55S69

FIRMWARE

- Rust / Trussed

GNUk (~2010)

GNUk – НЕ FIDO!

GNUk – OpenPGP Smart Card

GNUk – OpenPGP Smart Card


ВОТ ТАК, С ПОМОЩЬЮ НЕХИТРЫХ ПРИСПОСОБЛЕНИЙ БУХАНКУ БЕЛОГО (ИЛИ ЧЕРНОГО) ХЛЕБА МОЖНО ПРЕВРАТИТЬ В ТРОЛЛЕЙБУС...
НО ЗАЧЕМ?!
ХИЛЫЕ РУКИ.

GNUk

GNUk

life.augmented

STM32F103x8
STM32F103xB

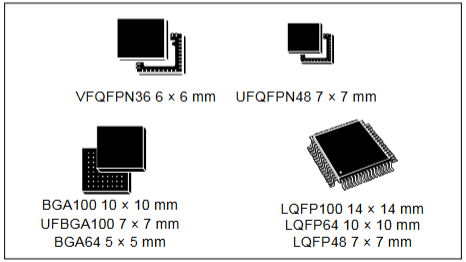
Medium-density performance line Arm®-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. interfaces

Datasheet - production data

Features

Includes ST state-of-the-art patented technology

- Arm® 32-bit Cortex®-M3 CPU core
  - 72 MHz maximum frequency, 1.25 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state memory access
  - Single-cycle multiplication and hardware division
- Memories
  - 64 or 128 Kbytes of Flash memory
  - 20 Kbytes of SRAM



- Debug mode:
  - Serial wire debug (SWD) and JTAG interfaces

GNUk

life.augmented

Medium-density performanc
64 or 128 KB Flash, USB, CAN,

Features

Includes ST state-of-the-art patented technology

- Arm® 32-bit Cortex®-M3 CPU core
  - 72 MHz maximum frequency, 1.25 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state memory access
  - Single-cycle multiplication and hardware division
- Memories
  - 64 or 128 Kbytes of Flash memory
  - 20 Kbytes of SRAM

GNUk

life.augmented

Medium-density performanc
64 or 128 KB Flash, USB, CAN,

Features

Includes ST state-of-the-art patented technology

- Arm® 32-bit Cortex®-M3 CPU core
  - 72 MHz maximum frequency, 1.25 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state memory access
  - Single-cycle multiplication and hardware division
- Memories
  - 64 or 128 Kbytes of Flash memory
  - 20 Kbytes of SRAM

GNUk

HARDWARE

· STM32 F103

FIRMWARE

· C (PolarSSL + ChibiOS / chopstx)

Hardware & Firmware Security

- Root of Trust
- Boot of Trust
- Zero Trust

Yubikey (Infineon-based)

PROS

CONS

Yubikey (Infineon-based)

PROS

- Security through obscurity
-
-

CONS

- Security through obscurity
-
-

Yubikey (Infineon-based)

PROS

- Security through obscurity
- Non-updatable
-

CONS

- Security through obscurity
- Non-updatable
-

Yubikey (Infineon-based)

PROS

- Security through obscurity
- Non-updatable
- Un-auditable / auditable as blackbox

CONS

- Security through obscurity
- Non-updatable
- Un-auditable [as whitebox]

Yubikey CVEs?

Yubikey CVEs?..

Yubikey CVEs:

HARDWARE

· CVE-2017-15361 / YSA-2017-01:
The "ROCA" Factorization Attack

· CVE-2024-45678 / YSA-2024-03:
The "EUCLEAK" Side-Channel Extraction

FIRMWARE / SOFTWARE

· CVE-2018-14780 & CVE-2018-14779 / YSA-2018-03:
libykpiv Buffer Overflows

· --- / YSA-2019-02:
FIPS "Reduced Randomness" Entropy Leak

· CVE-2020-15000 / YSA-2020-05:
OpenPGP application Resetting Code bug

· CVE-2025-29991 / YSA-2025-02:
FIDO PIN/UV Auth Protocol Two Out of Conformance Bug

Conclusions… and Trust Issues

- Firmware verification
- Transparent Source Bill of Materials chain
- Independent Audit & Assessment
- Emergent Vulnerability Management & hot fixing
- Legacy, Education & Business opportunities

I have a dream!..

- Open Schematics
- SB Bootloader w. tooling to enroll your own FW signing keys
- Open Source Firmware with reproducible builds
- Tooling to verify flashed firmware
- Ergonomic & protected case by industrial designers

Ideal workflow

- Buy blank device & verify zeroes
- Download & verify firmware sources
- Verify toolchain & SBOM (libraries / external components / etc.)
- Generate & enroll keys into FW bootloader
- Flash, sign & verify firmware
- Regular automagical attestation of hardware & firmware

?

QUESTIONS · COMMENTS · DISAGREEMENTS

github.com/ia/talks/tree/main/yoprst26