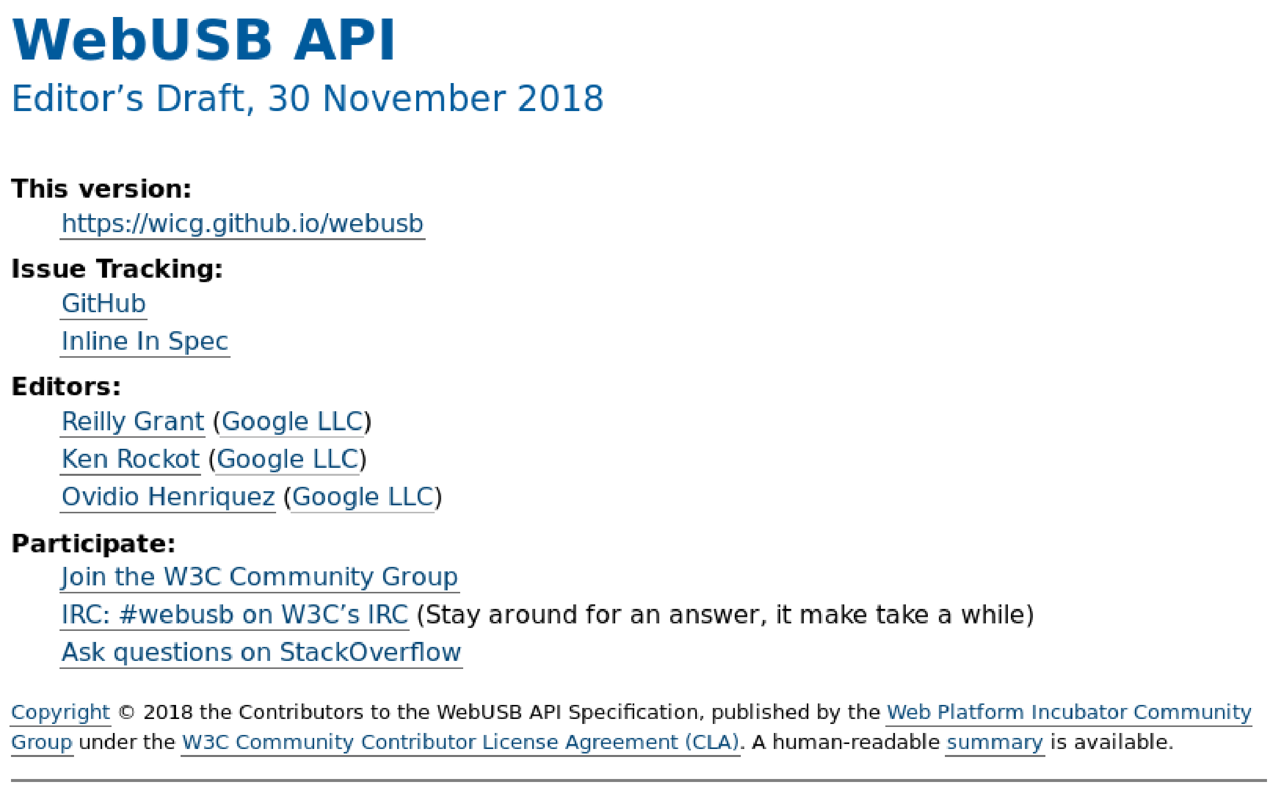
Securing the USB Interface

Jan Sebastian GOETTE
Waseda NSL
securehid@jaseg.net

Why do we need this?

- Organically grown ~~mess~~ ecosystem of protocols
- Created in another century
- Bad Ideas™ abound, then and now

Why do we need this?



WebUSB API

Editor’s Draft, 30 November 2018

This version:
https://wicg.github.io/webusb

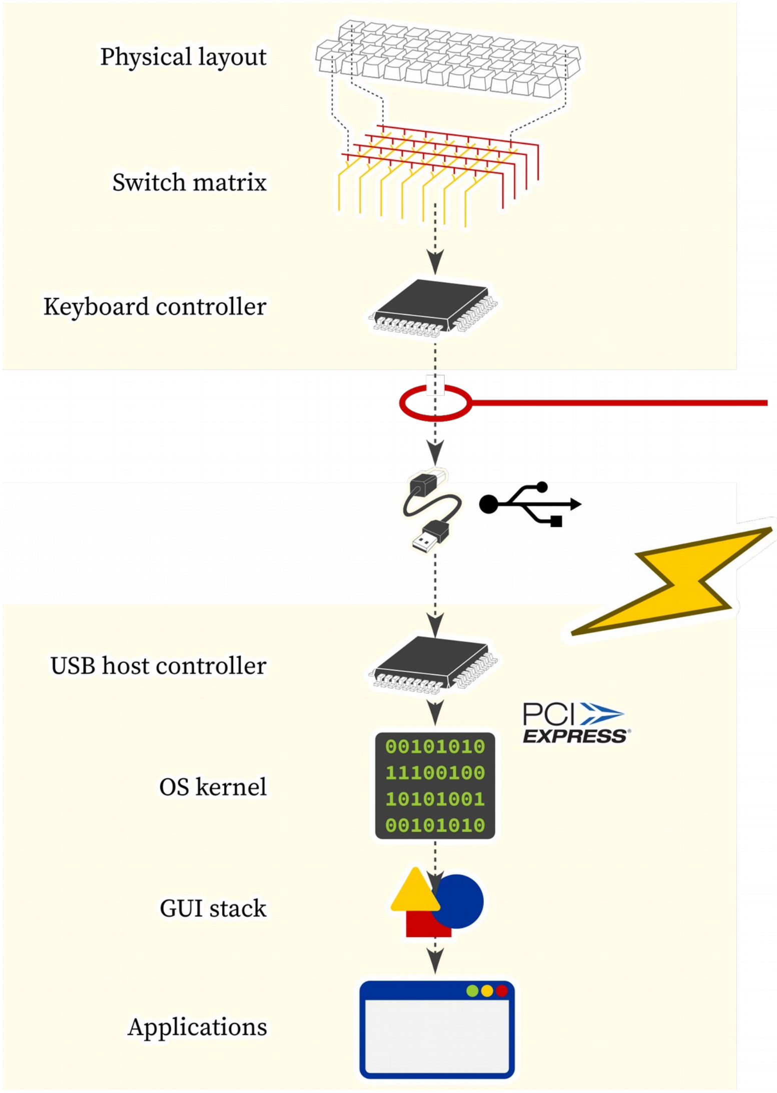
Issue Tracking:
GitHub
Inline In Spec

Editors:
Reilly Grant (Google LLC)
Ken Rockot (Google LLC)
Ovidio Henriquez (Google LLC)

Participate:
Join the W3C Community Group
IRC: #webusb on W3C’s IRC (Stay around for an answer, it make take a while)
Ask questions on StackOverflow

Copyright © 2018 the Contributors to the WebUSB API Specification, published by the Web Platform Incubator Community Group under the W3C Community Contributor License Agreement (CLA). A human-readable summary is available.

Conventional system using a USB keyboard



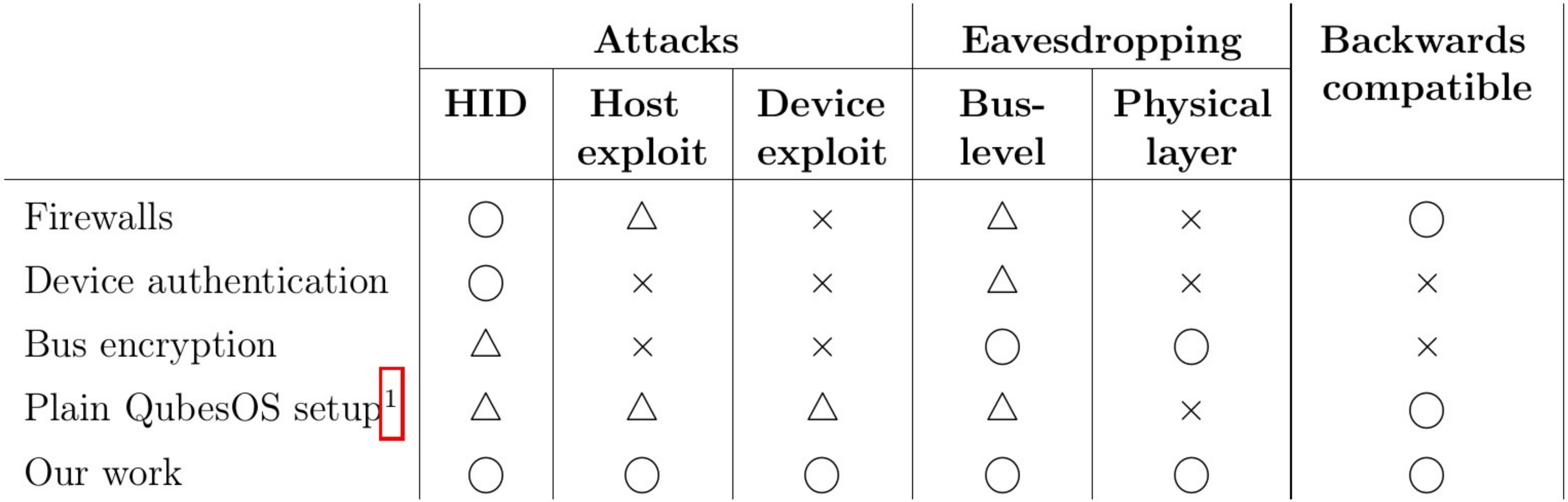
Regular USB Keyboard

Raw privileged input commands traverse trusted USB setup unprotected

Trusted USB setup may be attacked by malicious devices

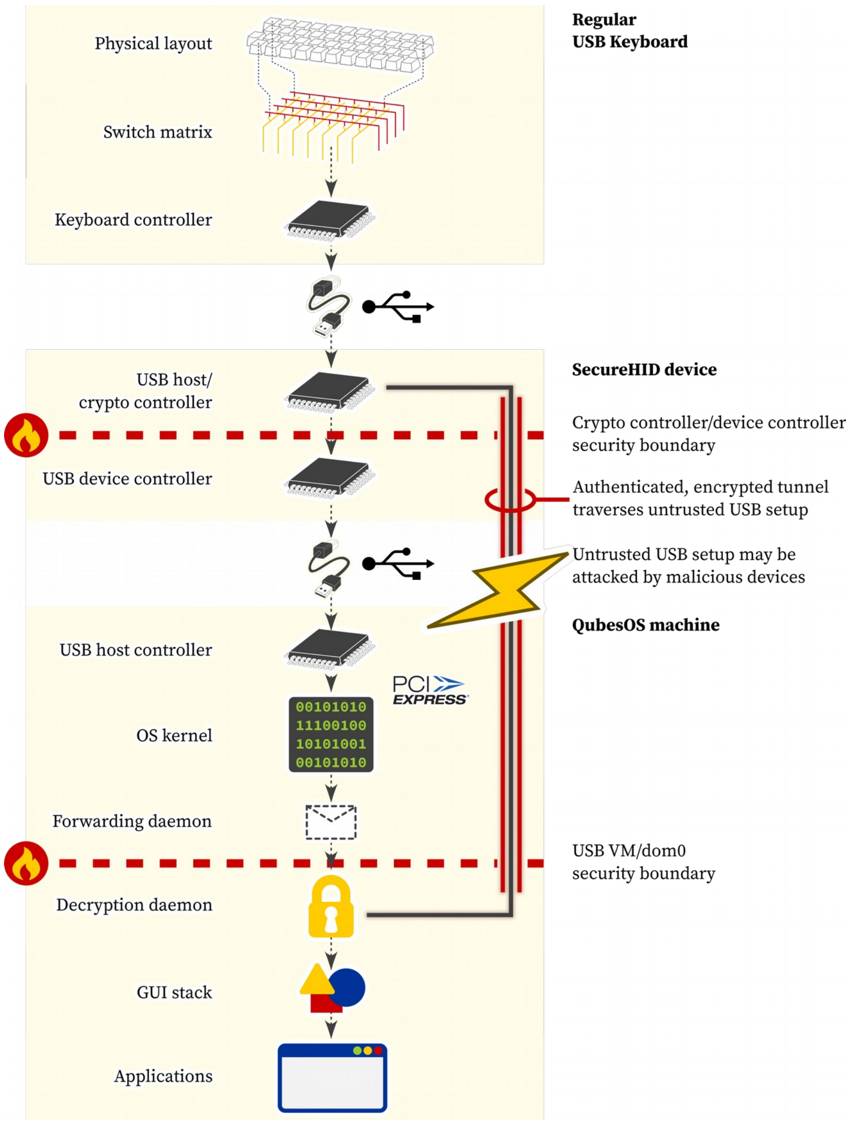
Standard Linux/Windows/Mac installation

The state of the art

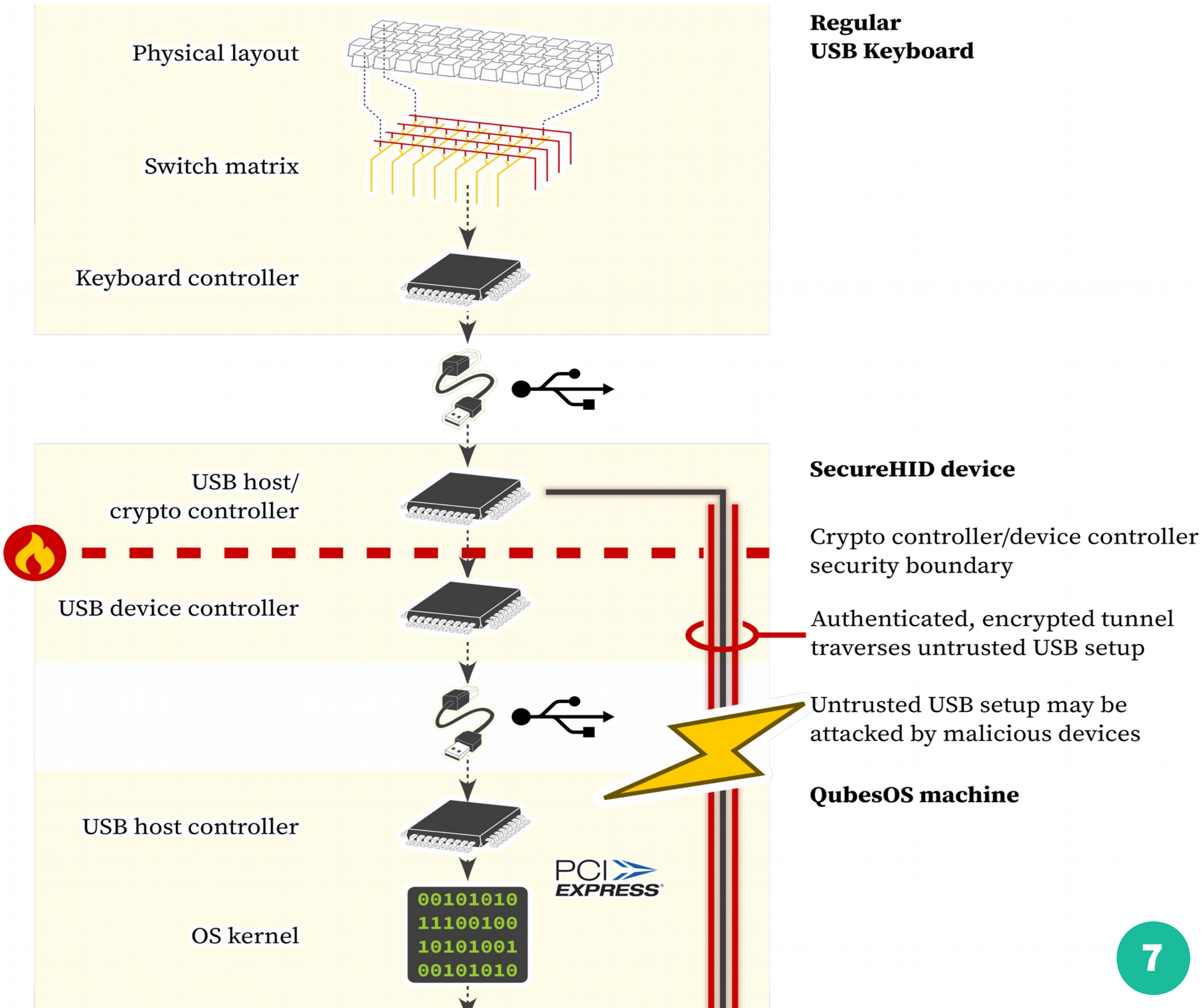


	Attacks			Eavesdropping		Backwards compatible
	HID	Host exploit	Device exploit	Bus-level	Physical layer	
Firewalls	○	△	×	△	×	○
Device authentication	○	×	×	△	×	×
Bus encryption	△	×	×	○	○	×
Plain QubesOS setup[1]	△	△	△	△	×	○
Our work	○	○	○	○	○	○

SecureHID/ QubesOS-based system

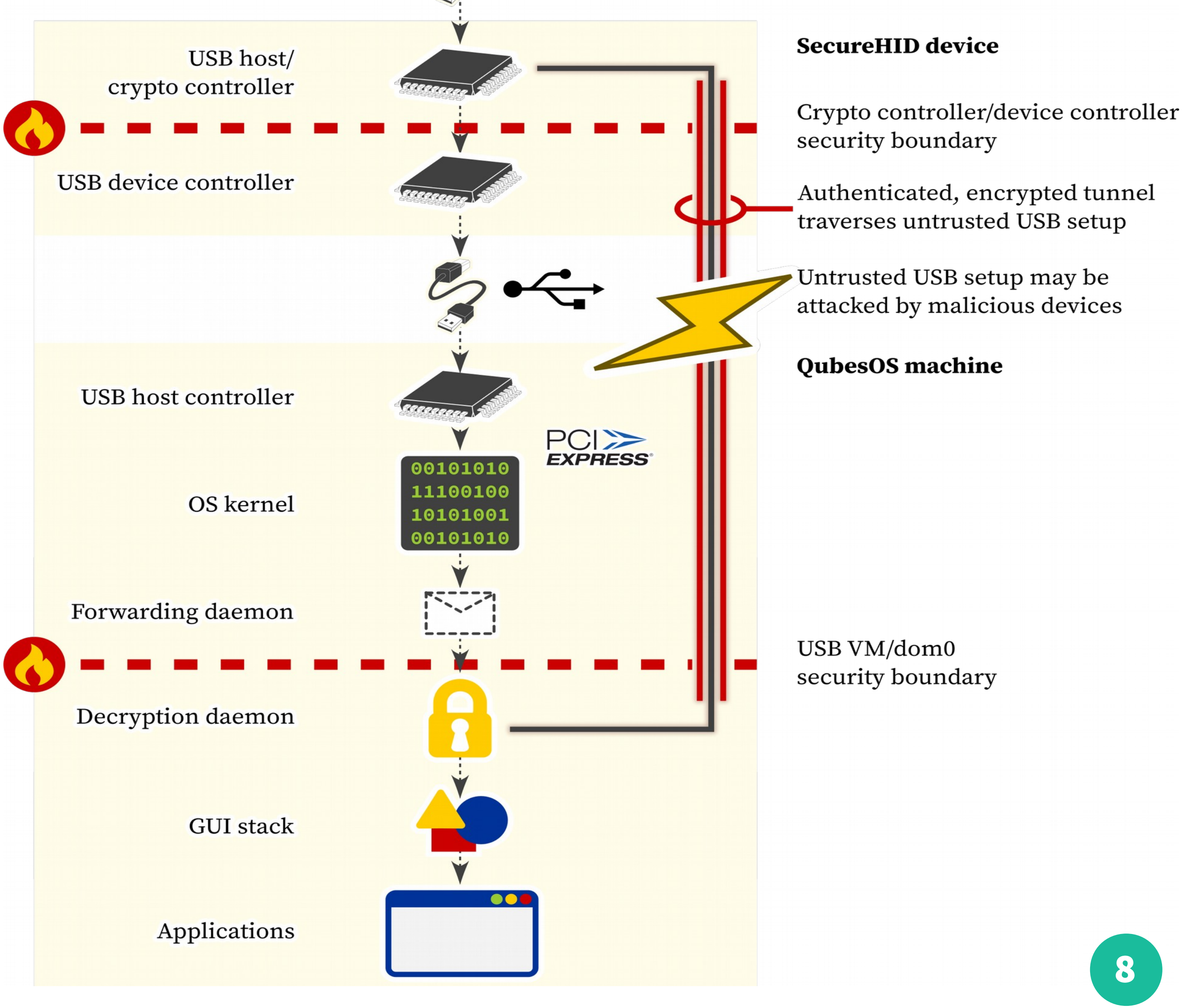


SecureHID/ QubesOS-based system: Keyboard side

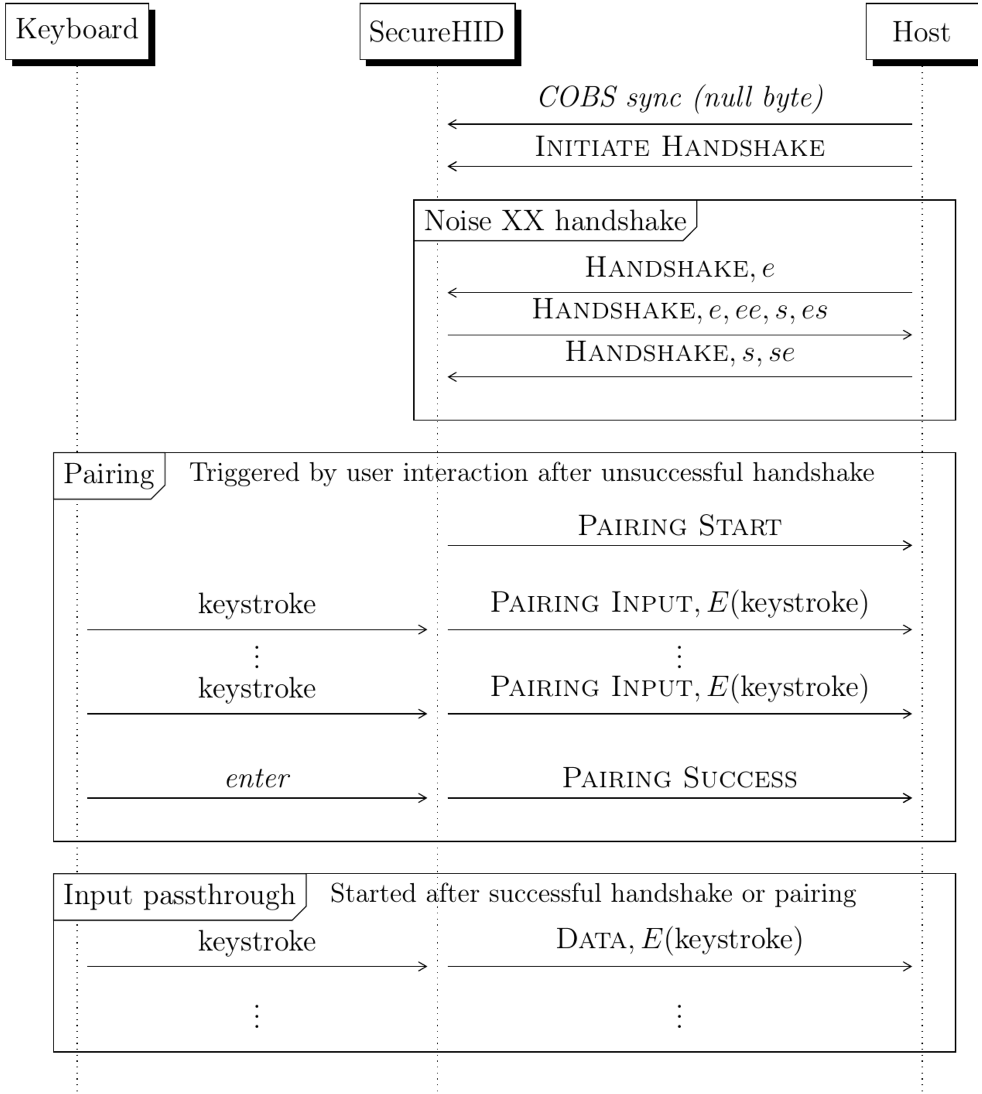


SecureHID/ QubesOS-based system:

Host side


SecureHID device
USB host/ crypto controller
Crypto controller/device controller security boundary
USB device controller
Authenticated, encrypted tunnel traverses untrusted USB setup
Untrusted USB setup may be attacked by malicious devices
QubesOS machine
USB host controller
PCI EXPRESS
OS kernel
00101010
11100100
10101001
00101010
Forwarding daemon
USB VM/dom0 security boundary
Decryption daemon
GUI stack
Applications

Handshake and pairing protocol



Keyboard

SecureHID

Host

COBS sync (null byte)

INITIATE HANDSHAKE

Noise XX handshake

HANDSHAKE, e

HANDSHAKE, e, ee, s, es

HANDSHAKE, s, se

Pairing

Triggered by user interaction after unsuccessful handshake

PAIRING START

keystroke

PAIRING INPUT, E(keystroke)

⋮

keystroke

PAIRING INPUT, E(keystroke)

enter

PAIRING SUCCESS

Input passthrough

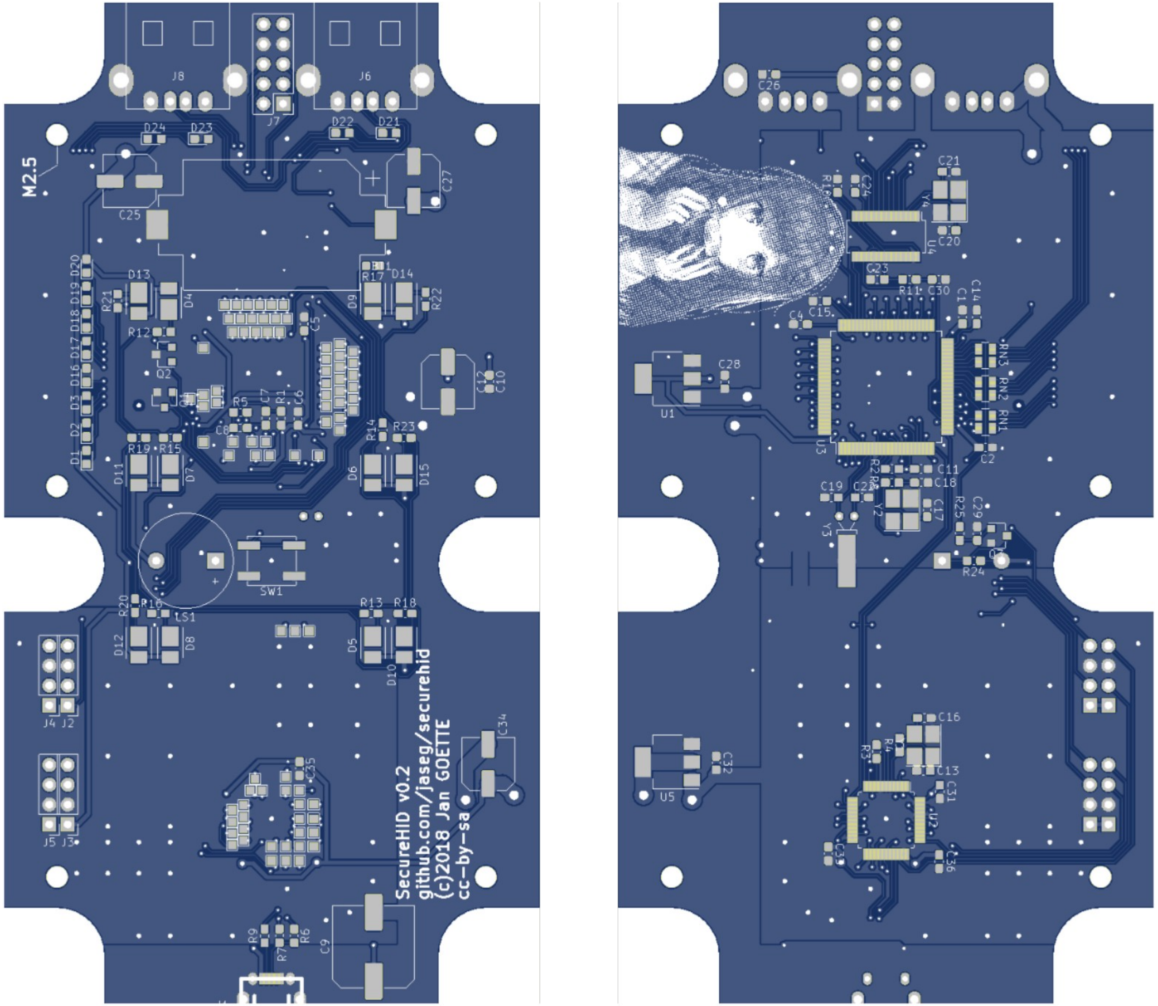
Started after successful handshake or pairing

keystroke

DATA, E(keystroke)

⋮

Physical realization



Questions?

Comments?

`securehid@jaseg.net`

Title slide picture by Gürkan Sengün on Wikimedia Commons