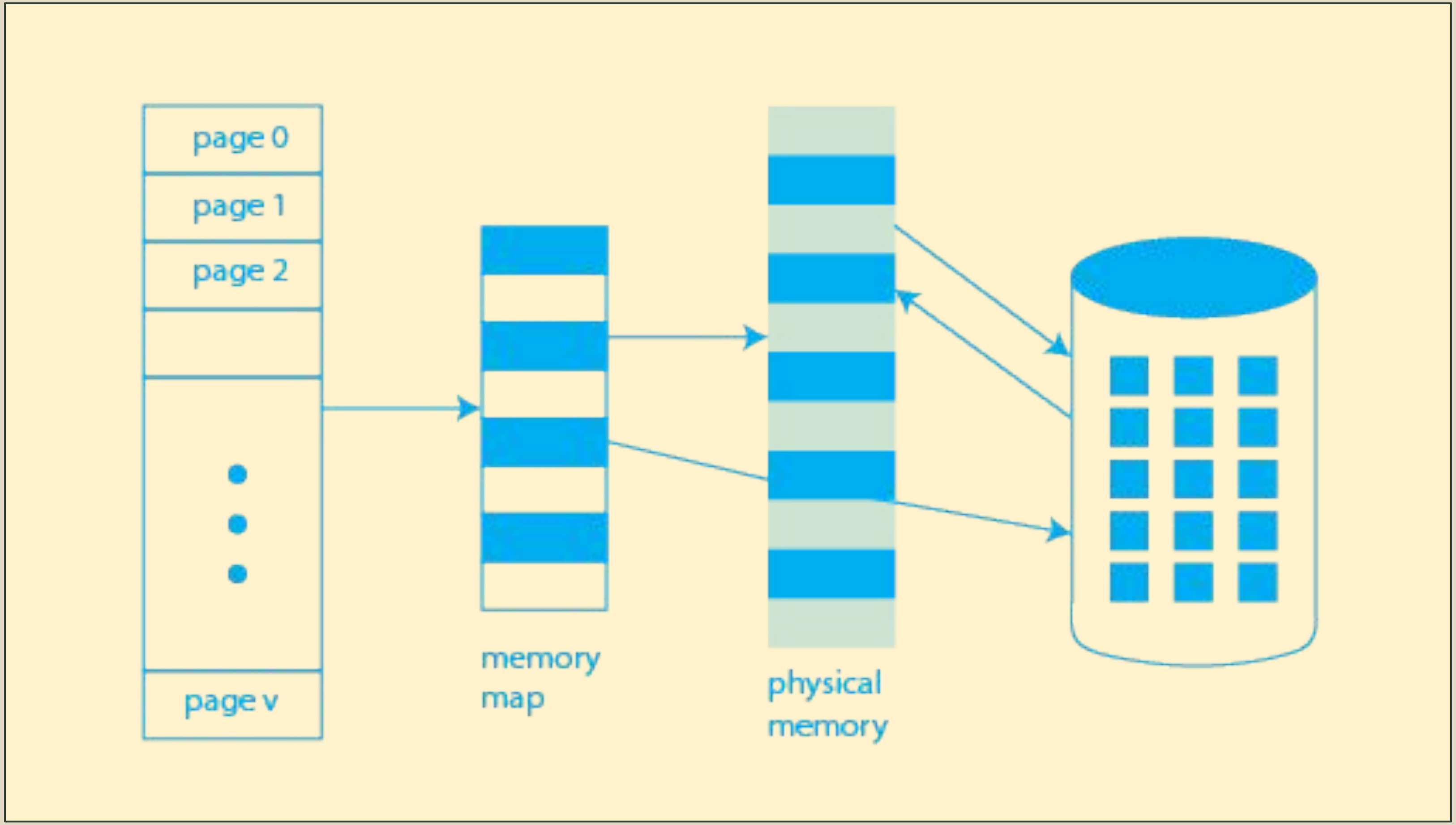
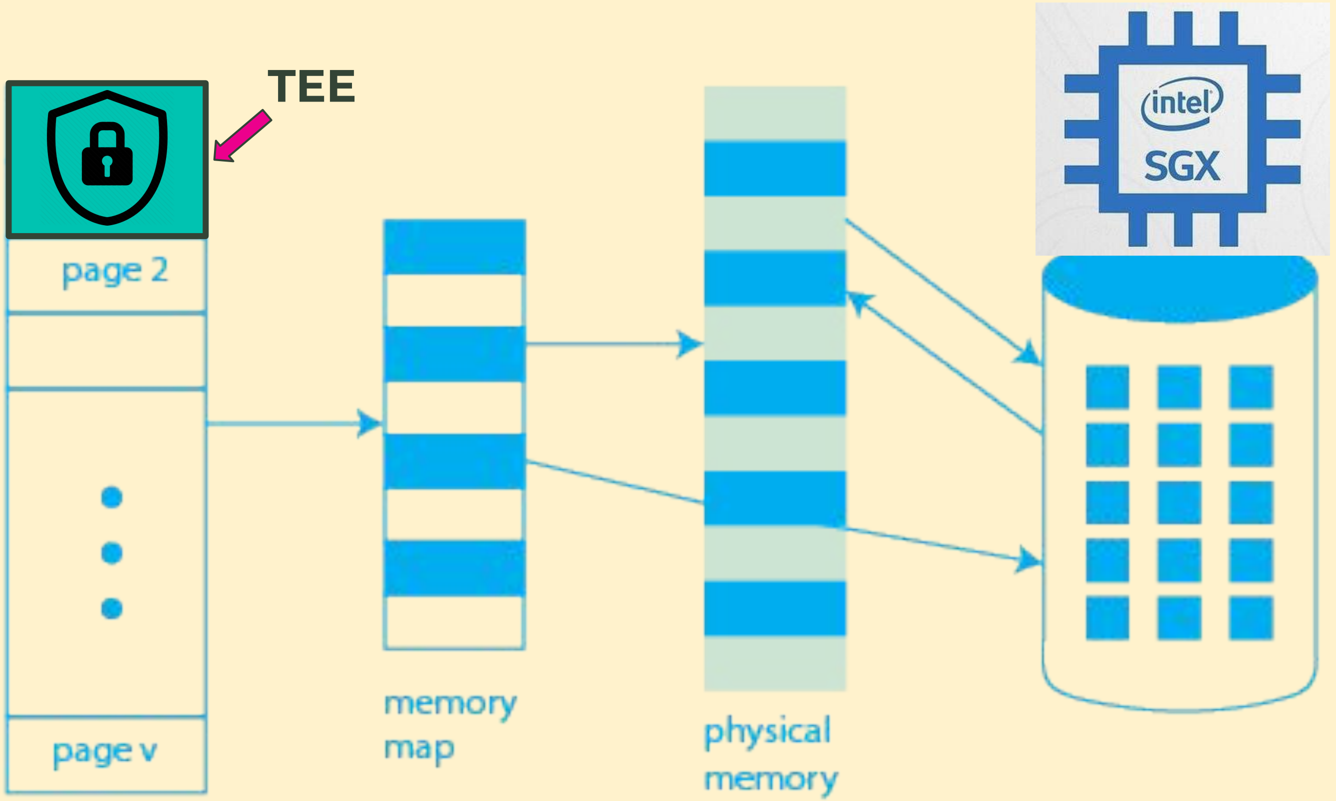
Functional Programming for Securing Cloud and Embedded Environments

Extra Slides

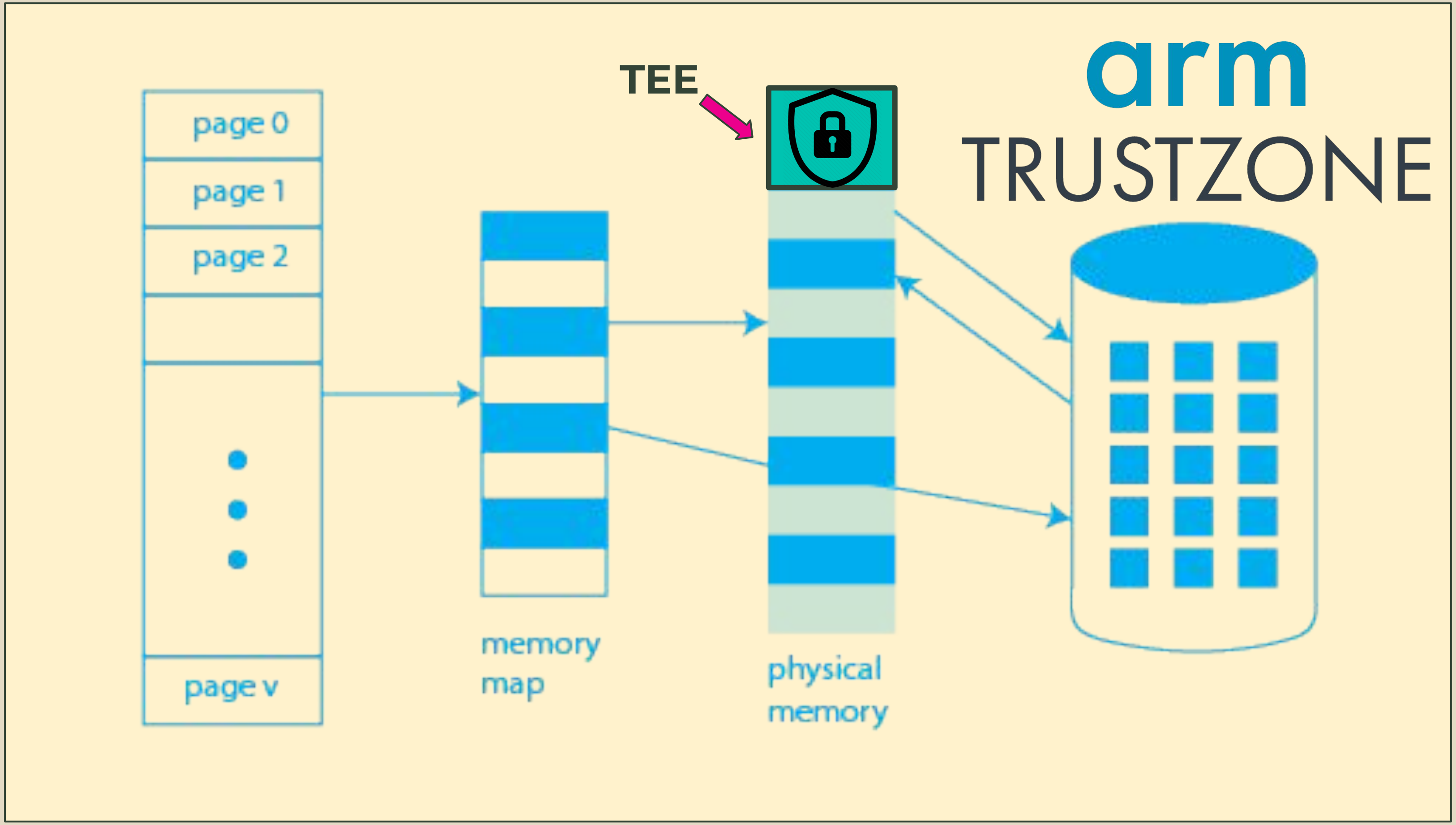
TEE Implementations



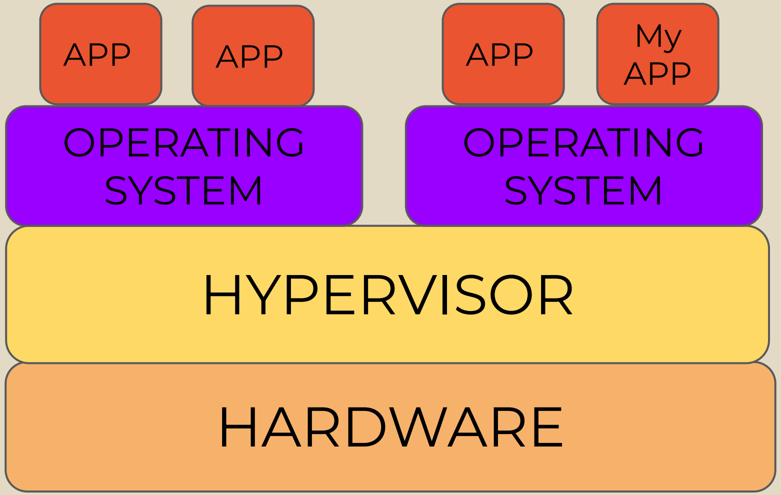
TEE Implementations



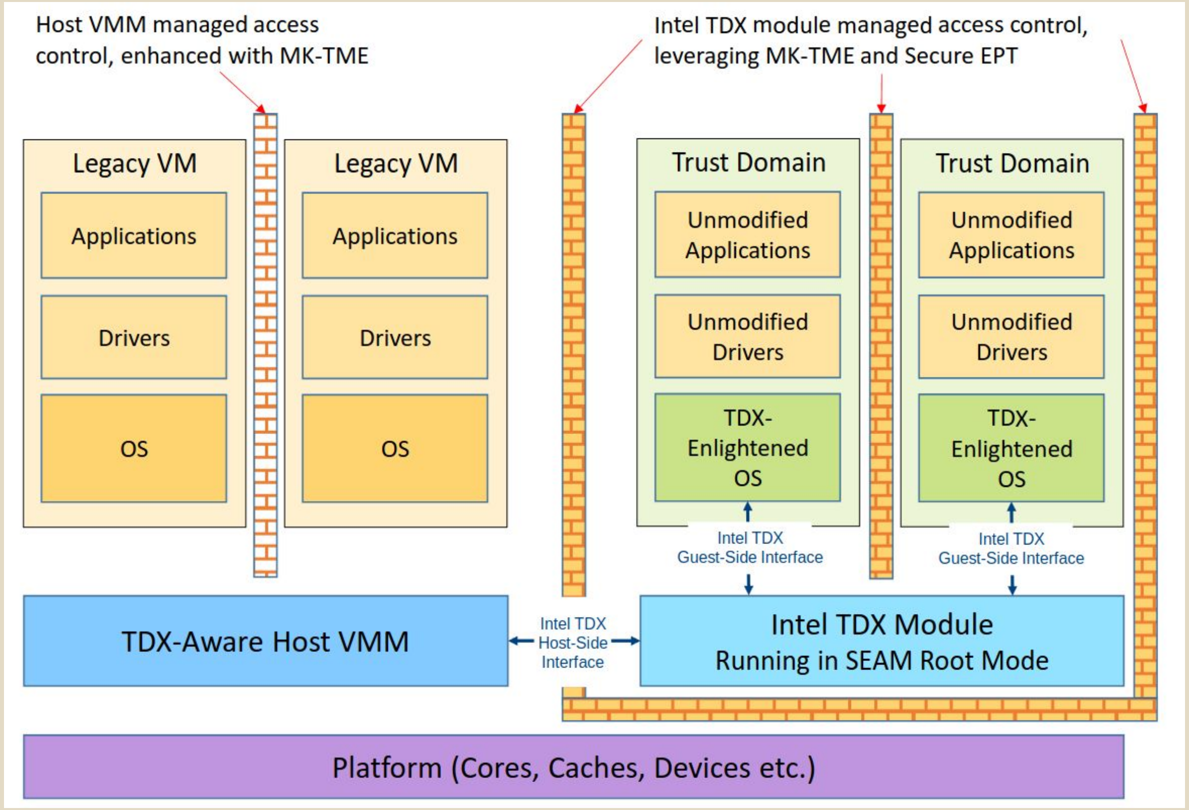
TEE Implementations



TEE Implementations

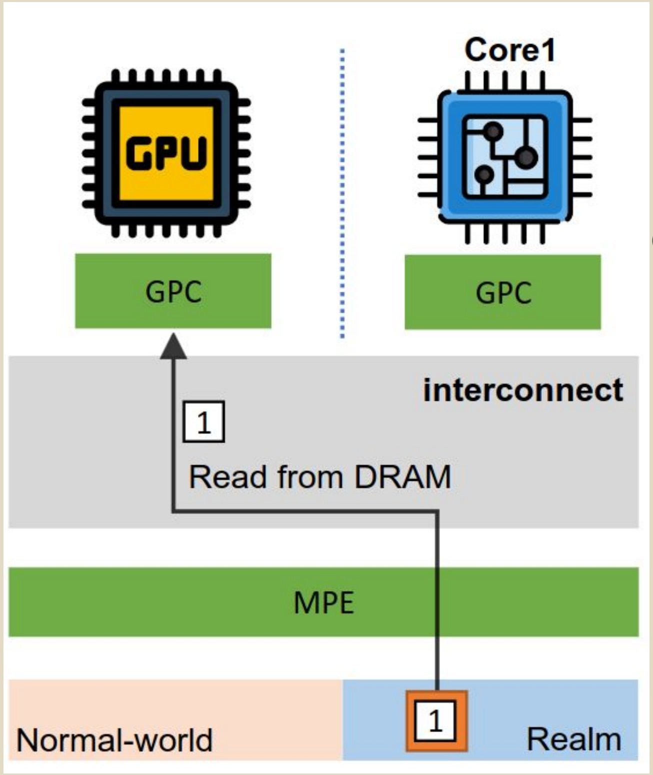


TEE Implementations


Host VMM managed access control, enhanced with MK-TME
Intel TDX module managed access control, leveraging MK-TME and Secure EPT
Legacy VM
Applications
Drivers
OS
Legacy VM
Applications
Drivers
OS
Trust Domain
Unmodified Applications
Unmodified Drivers
TDX-Enlightened OS
Intel TDX Guest-Side Interface
Trust Domain
Unmodified Applications
Unmodified Drivers
TDX-Enlightened OS
Intel TDX Guest-Side Interface
TDX-Aware Host VMM
Intel TDX Host-Side Interface
Intel TDX Module Running in SEAM Root Mode
Platform (Cores, Caches, Devices etc.)

Source: Intel TDX 1.0 spec

TEE Implementations

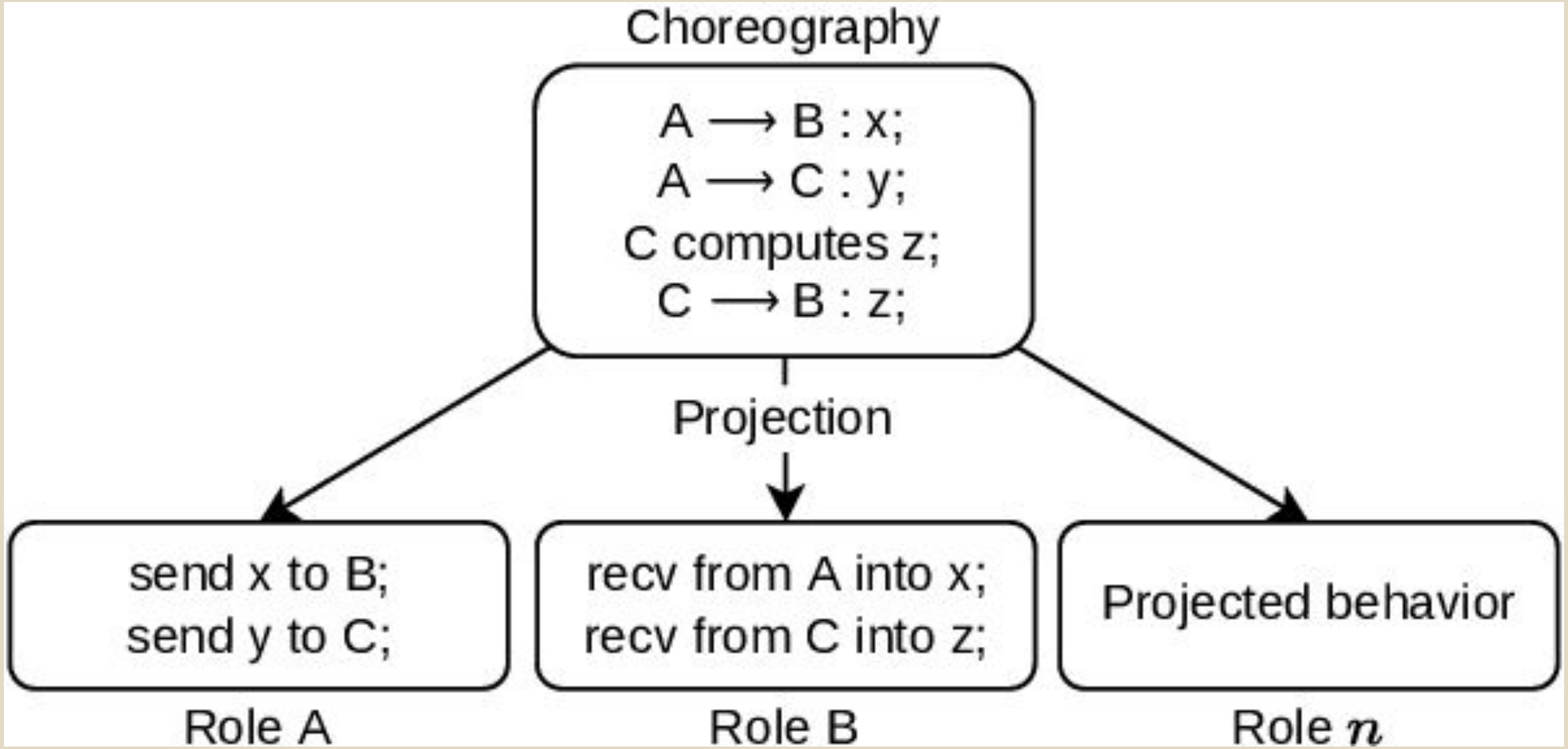


GPT accessible by monitor only

GPCs for address translation and bus transactions

Source: ACAI. Sridhara et al. Usenix Sec 2024

Choreographies



Information Flow

```
label :: (Label l) => l → a → Enclave l p (Labeled l a)
```

$$\Gamma \vdash \ldots$$

Information Flow

```
label :: (Label l) => l → a → Enclave l p (Labeled l a)
```

$$L \sqsubseteq L_{cur}\,?$$

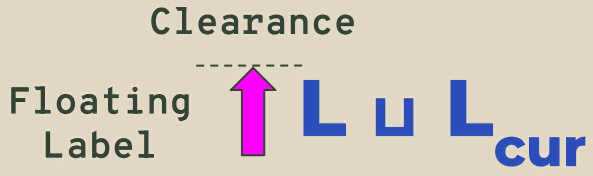
$$\begin{tikzcd} L_{cur} & \textcircled{Γ} \arrow[l] \vdash \dots \end{tikzcd}$$

Information Flow

label :: (Label l) => l → a → Enclave l p (Labeled l a)

unlabel :: (Label l) => Labeled l a → Enclave l p a

$$L \sqcup L_{cur} \sqsubseteq C_{cur} ?$$



$\Gamma \vdash \ldots$

Information Flow

unlabel

$$\langle \text{Alice} \land \text{Bob}, \text{Alice} \land \text{Bob} \rangle \sqsubseteq \langle \text{Alice}, \text{Alice} \rangle$$

Stefan, D., Russo, A., Mazières, D., & Mitchell, J. C. (2012). Disjunction Category Labels. In *Information Security Technology for Applications: 16th Nordic Conference on Secure IT Systems, NordSec 2011*

Information Flow

unlabel

$$\langle \text{Alice} \land \text{Bob}, \text{Alice} \land \text{Bob} \rangle \sqsubseteq \langle \text{Alice}, \text{Alice} \rangle$$

$$\frac{C_2 \Rightarrow C_1 \qquad\qquad I_1 \Rightarrow I_2}{\langle C_1, I_1 \rangle \sqsubseteq \langle C_2, I_2 \rangle}$$

Stefan, D., Russo, A., Mazières, D., & Mitchell, J. C. (2012). Disjunction Category Labels. In *Information Security Technology for Applications: 16th Nordic Conference on Secure IT Systems, NordSec 2011*

Information Flow

unlabel

$$\frac{\boxed{\text{Alice} \Rightarrow \text{Alice} \land \text{Bob}} \qquad \text{Alice} \land \text{Bob} \Rightarrow \text{Alice}}{\langle \text{Alice} \land \text{Bob}, \text{Alice} \land \text{Bob} \rangle \not\sqsubseteq \langle \text{Alice}, \text{Alice} \rangle}$$

Stefan, D., Russo, A., Mazières, D., & Mitchell, J. C. (2012). Disjunction Category Labels. In *Information Security Technology for Applications: 16th Nordic Conference on Secure IT Systems, NordSec 2011*

Declassification

```
unlabelP :: (PrivDesc l p) => Priv p → Labeled l a → Enclave l p a
```

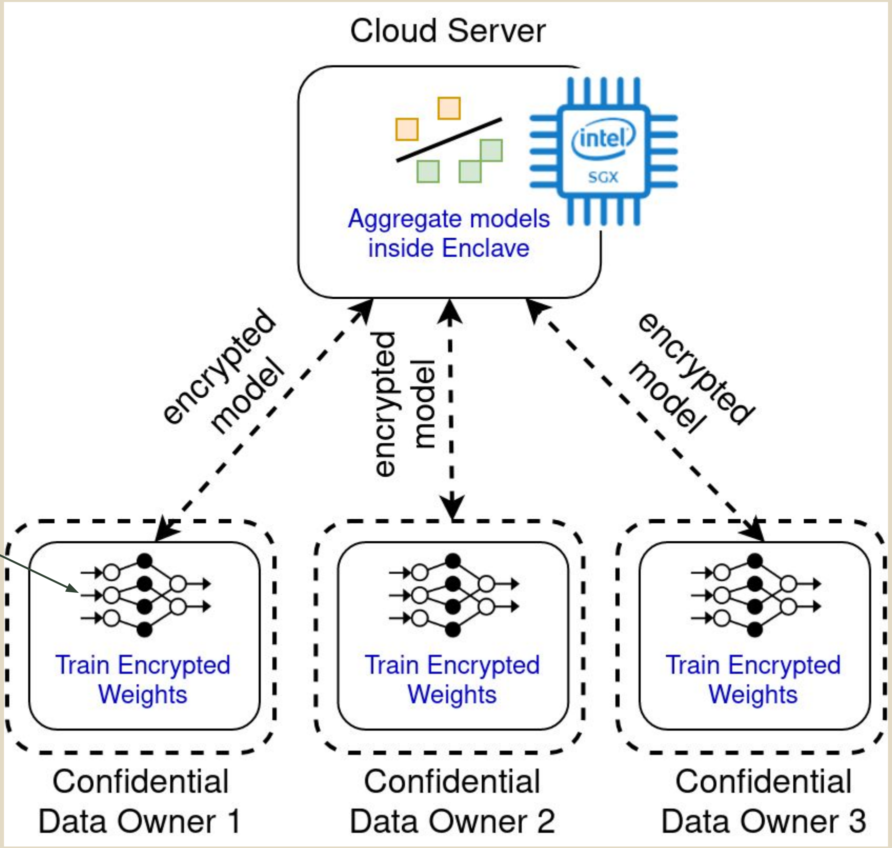
Privilege or capability

$$\frac{\text{Alice} \wedge \text{Bob} \Rightarrow \text{Alice} \wedge \text{Bob} \qquad \text{Alice} \wedge \text{Bob} \Rightarrow \text{Alice}}{\langle \text{Alice} \wedge \text{Bob}, \text{Alice} \wedge \text{Bob} \rangle \sqsubseteq_{P} \langle \text{Alice}, \text{Alice} \rangle}$$

Stefan, D., Russo, A., Mazières, D., & Mitchell, J. C. (2012). Disjunction Category Labels. In *Information Security Technology for Applications: 16th Nordic Conference on Secure IT Systems, NordSec 2011*

Zero Trust Federated Learning

Uses homomorphic encryption for training



Synchron

```
-- note frequencies
g = usec 2551
a = usec 2273
b = usec 2025
c = usec 1911
d = usec 1703
e = usec 1517

-- note duration
hn = msec 1000 -- half note
qn = msec  500 -- quarter note
```

```
twinkle : List Int
twinkle   = [ g,  g,  d,  d,  e,  e,  d.... ]

durations : List Int
durations = [qn, qn, qn, qn, qn, qn, hn.... ]
```

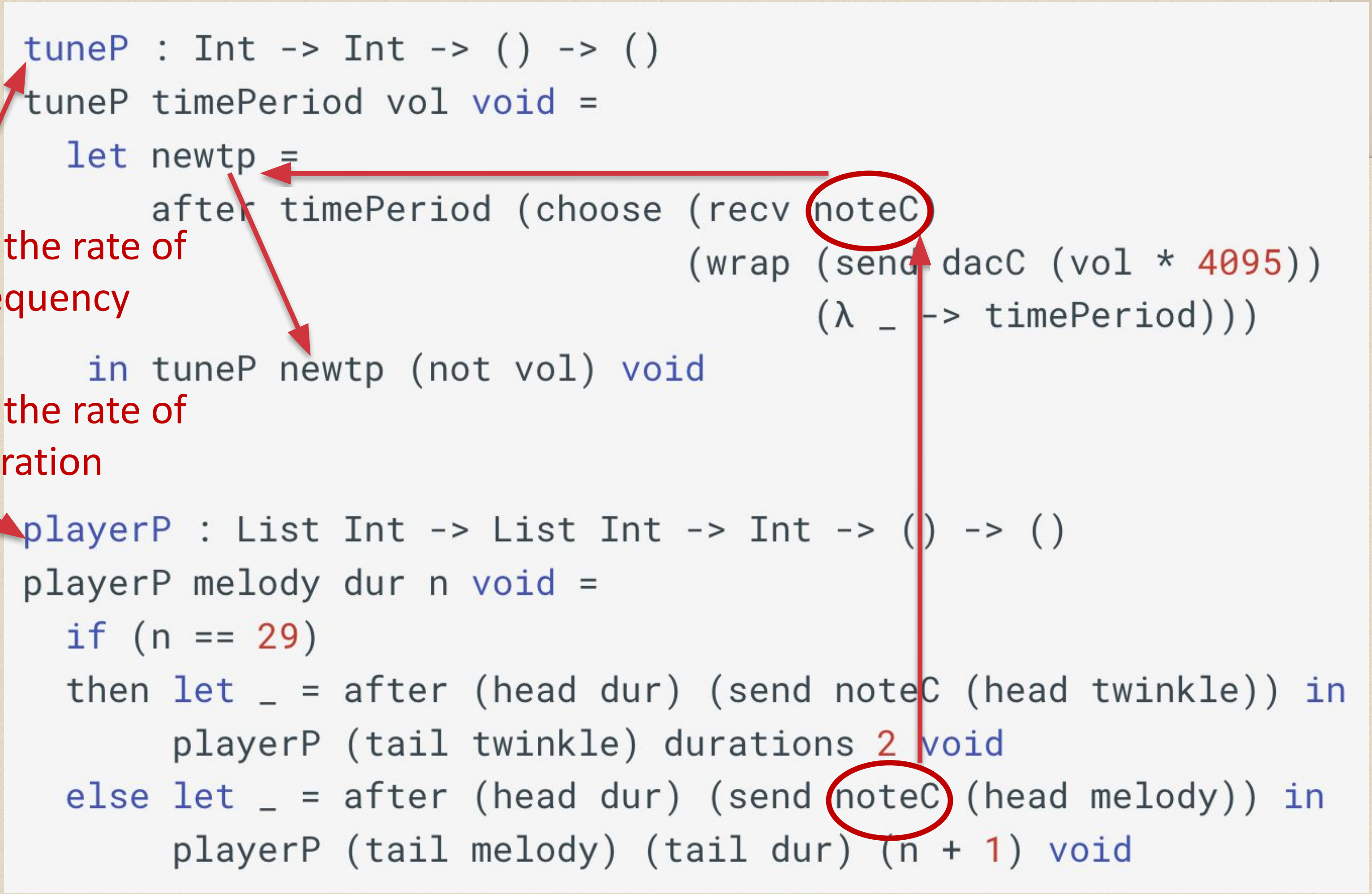
```
dacC : Channel Int
dacC = channel ()

noteC : Channel Int
noteC = channel ()
```

```
after t ev = syncT t 0 ev
```

```
tuneP : Int -> Int -> () -> ()
tuneP timePeriod vol void =
  let newtp =
      after timePeriod (choose (recv noteC)
                               (wrap (send dacC (vol * 4095))
                                     (λ _ -> timePeriod)))
   in tuneP newtp (not vol) void


playerP : List Int -> List Int -> Int -> () -> ()
playerP melody dur n void =
  if (n == 29)
  then let _ = after (head dur) (send noteC (head twinkle)) in
       playerP (tail twinkle) durations 2 void
  else let _ = after (head dur) (send noteC (head melody)) in
       playerP (tail melody) (tail dur) (n + 1) void
```

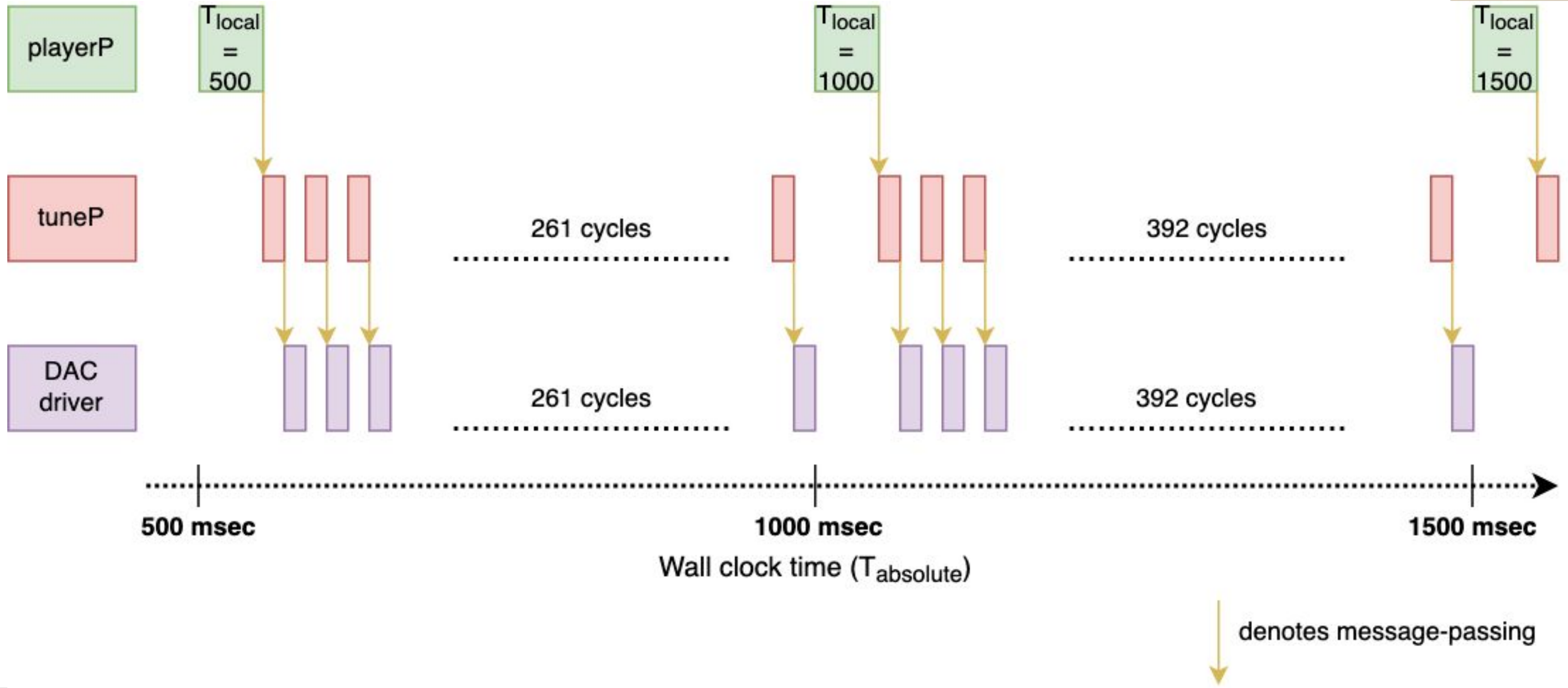


```
tuneP : Int -> Int -> () -> ()
tuneP timePeriod vol void =
  let newtp =
        after timePeriod (choose (recv noteC)
                                 (wrap (send dacC (vol * 4095))
                                       (λ _ -> timePeriod)))
   in tuneP newtp (not vol) void

playerP : List Int -> List Int -> Int -> () -> ()
playerP melody dur n void =
  if (n == 29)
  then let _ = after (head dur) (send noteC (head twinkle)) in
       playerP (tail twinkle) durations 2 void
  else let _ = after (head dur) (send noteC (head melody)) in
       playerP (tail melody) (tail dur) (n + 1) void
```

Runs at the rate of note frequency

Runs at the rate of note duration


playerP
tuneP
DAC driver
T_{local} = 500
T_{local} = 1000
T_{local} = 1500
261 cycles
392 cycles
261 cycles
392 cycles
500 msec
1000 msec
1500 msec
Wall clock time ($T_{absolute}$)
denotes message-passing