HasTEE – Confidential Computing with Haskell

Abhiroop Sarkar
Chalmers University, Gothenburg

CHALMERS


CHALMERS

hbc Public archive

HBC - Chalmers Haskell-B compiler

M 7 2

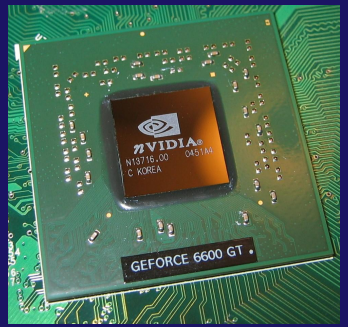
Why Functional Programming Matters

John Hughes, Institutionen för Datavetenskap,
Chalmers Tekniska Högskola,
41296 Göteborg,
SWEDEN. rjmh@cs.chalmers.se

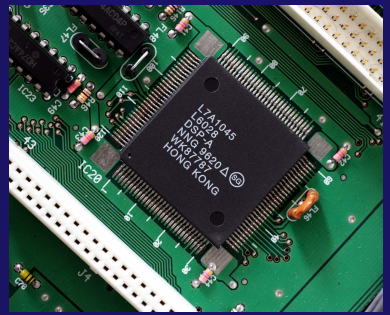
This paper dates from 1984, and circulated as a Chalmers memo for many years. Slightly revised versions appeared in 1989 and 1990 as [Hug90] and

EDSLs

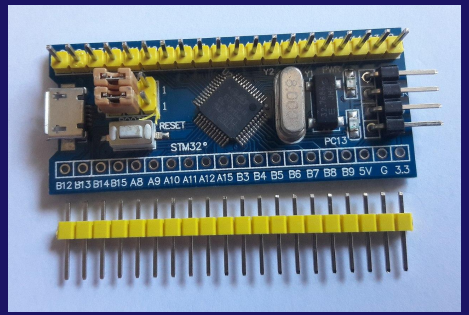
Obsidian



Feldspar



SynchronVM



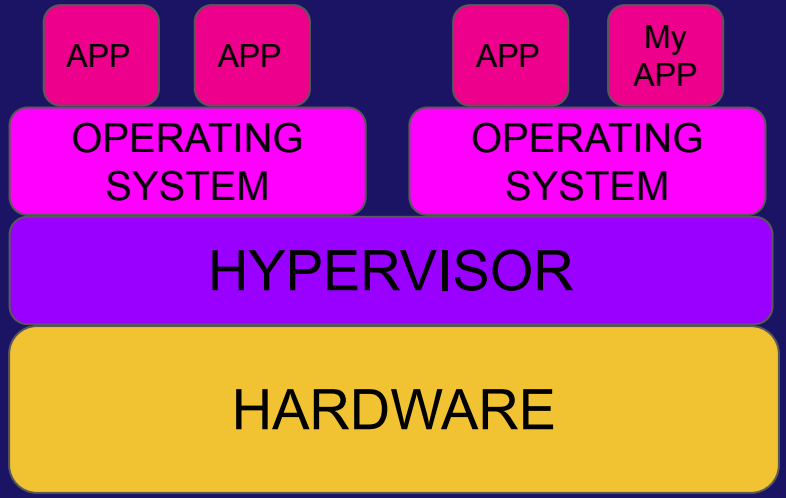
?

arm
TRUSTZONE

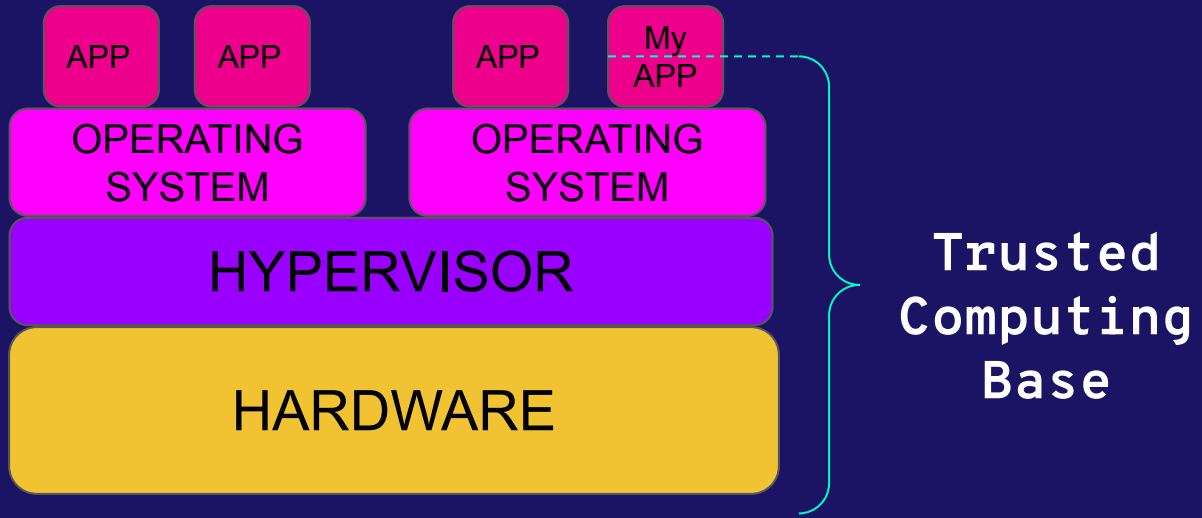
intel
SGX

Trusted Execution Environments

Cloud Deployments



Cloud Deployments



OS Vulnerabilities

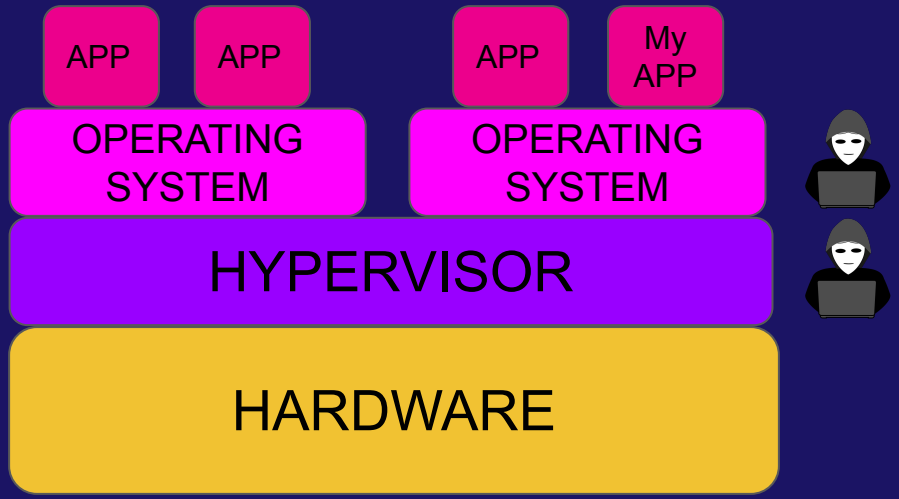
Vulnerability	Total	core	drivers	net	fs	sound
Missing pointer check	8	4	3	1	0	0
Missing permission check	17	3	1	2	11	0
Buffer overflow	15	3	1	5	4	2
Integer overflow	19	4	4	8	2	1
Uninitialized data	29	7	13	5	2	2
Null dereference	20	9	3	7	1	0
Divide by zero	4	2	0	0	1	1
Infinite loop	3	1	1	1	0	0
Data race / deadlock	8	5	1	1	1	0
Memory mismanagement	10	7	1	1	0	1
Miscellaneous	8	2	0	4	2	0
Total	141	47	28	35	24	7

Figure 2: Vulnerabilities (rows) vs. locations (columns).

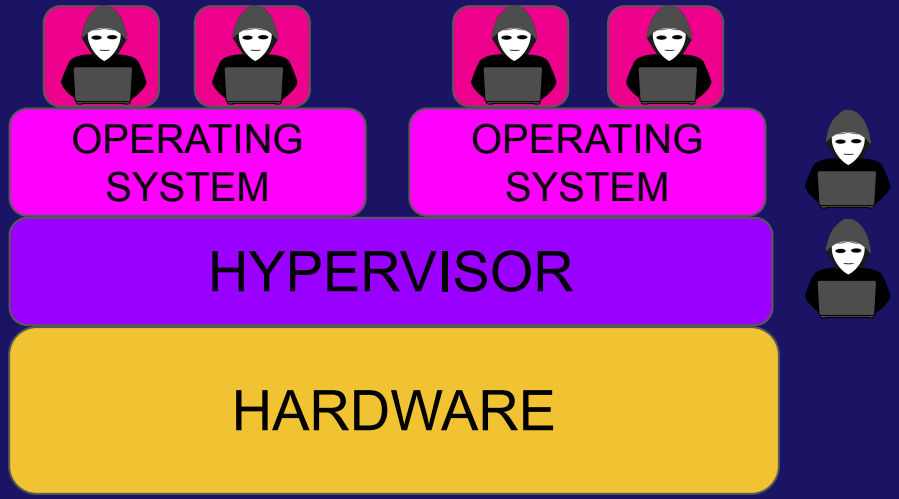
Linux kernel vulnerabilities: State-of-the-art defenses and open problems. Mao et al. In *Proceedings of the Second Asia-Pacific Workshop on Systems* (pp. 1-5).

Characterizing hypervisor vulnerabilities in cloud computing servers. Perez-Botero et al. In *Proceedings of the 2013 international workshop on Security in cloud computing*.

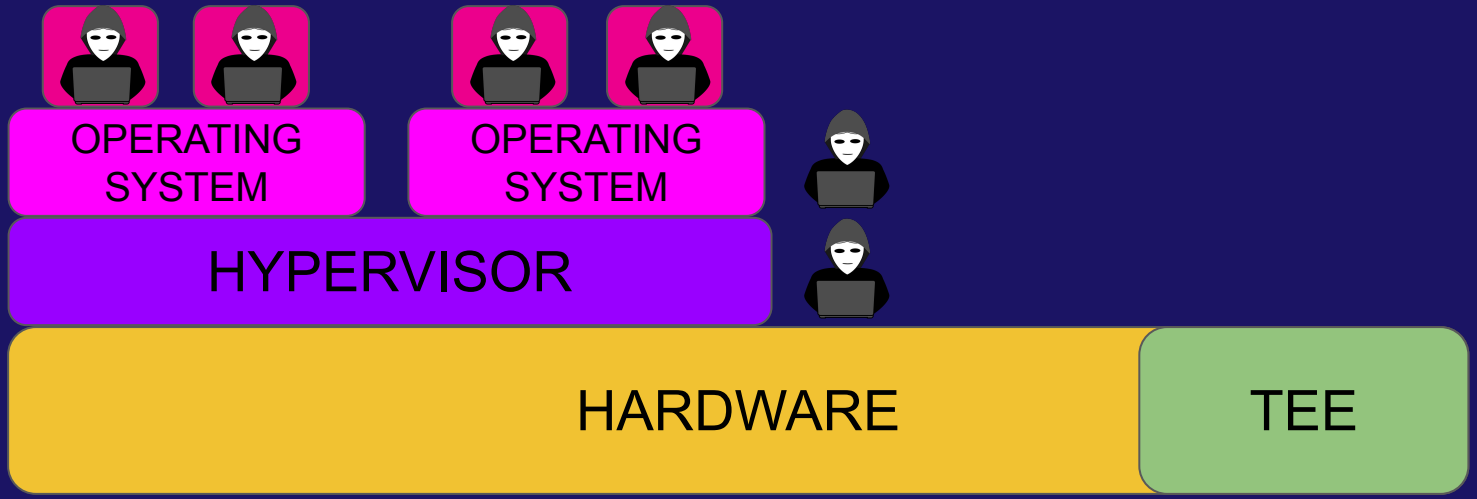
Cloud Deployments



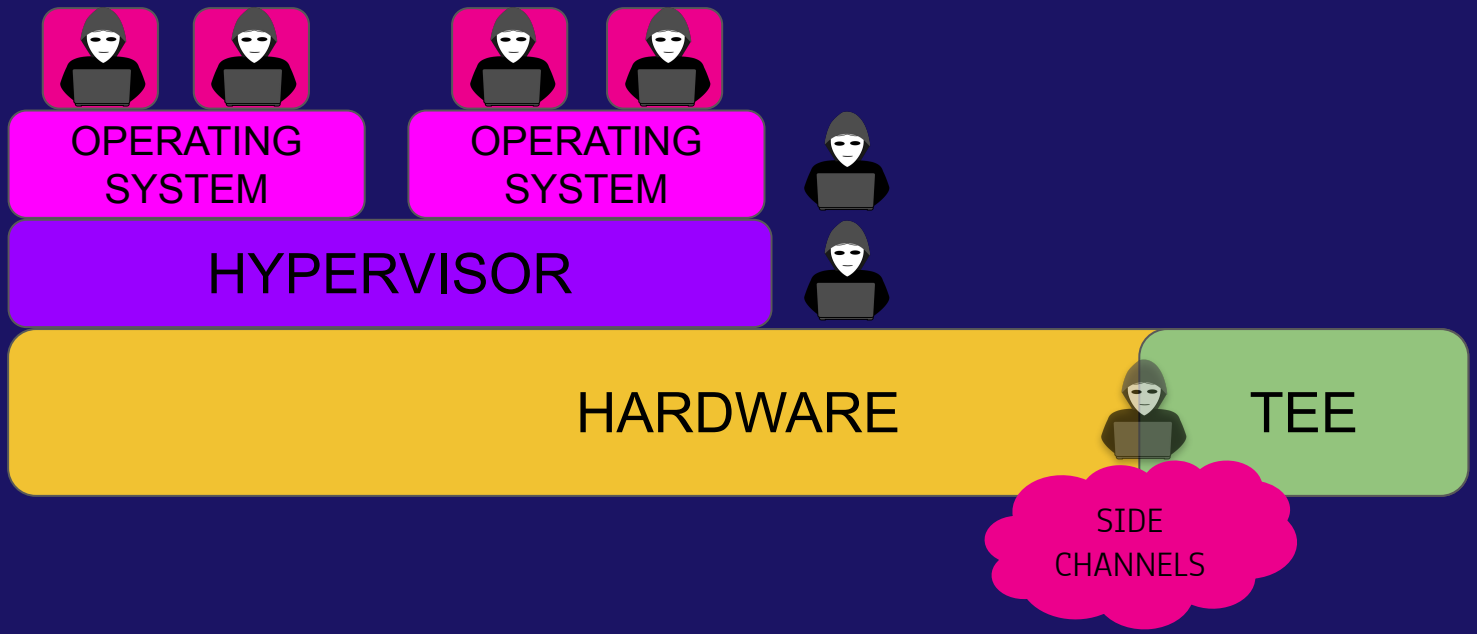
Cloud Deployments



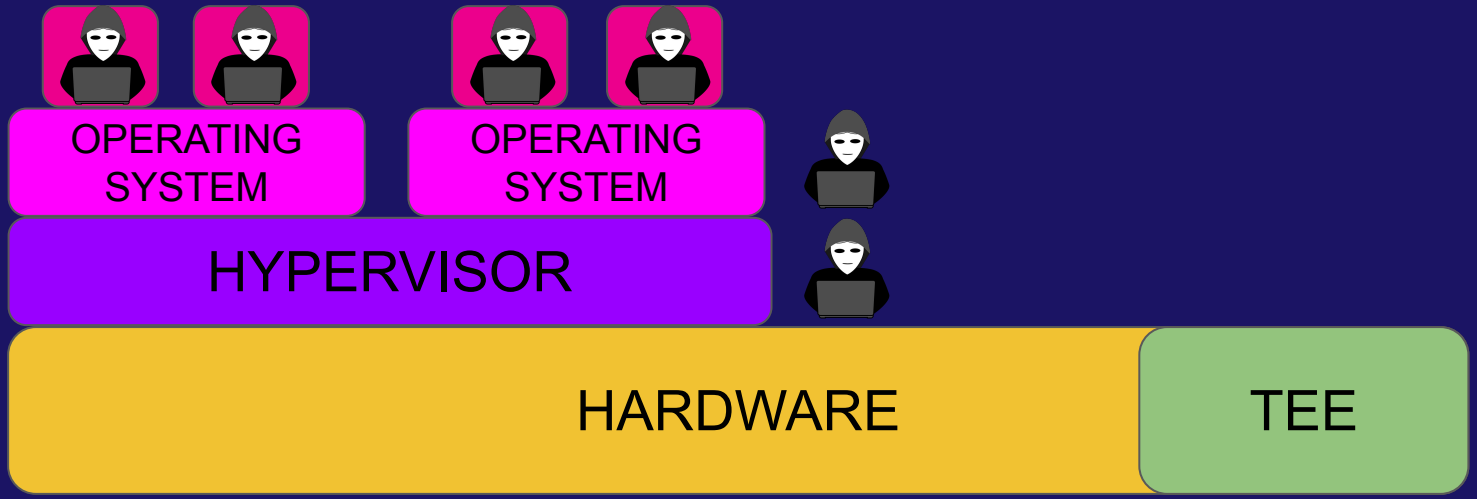
Trusted Execution Environment (TEE)



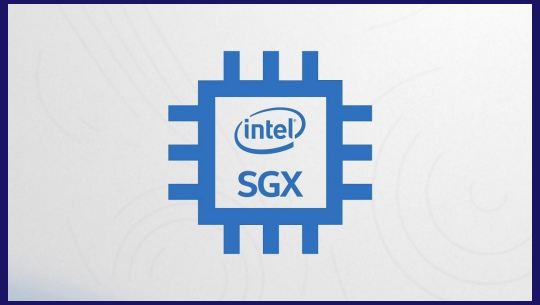
Trusted Execution Environment (TEE)



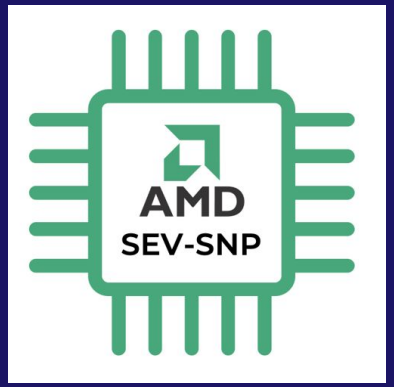
Trusted Execution Environment (TEE)



Trusted Execution Environment (TEE)



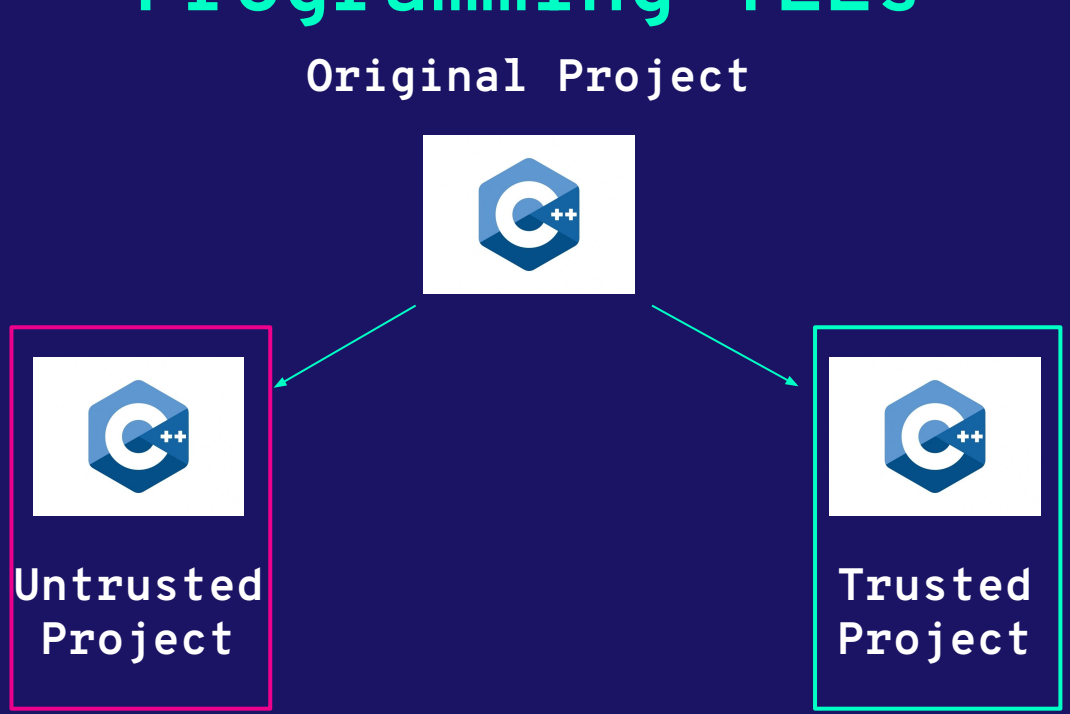
arm
TRUSTZONE



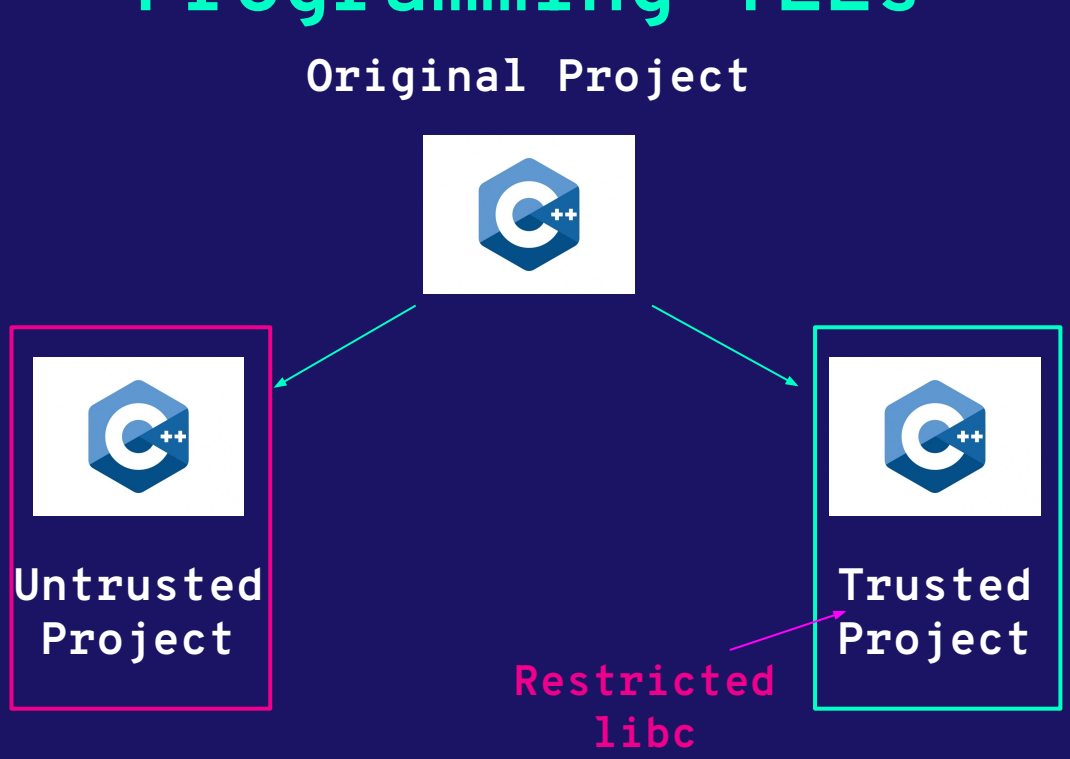
Physical Memory Protection

Programming TEEs

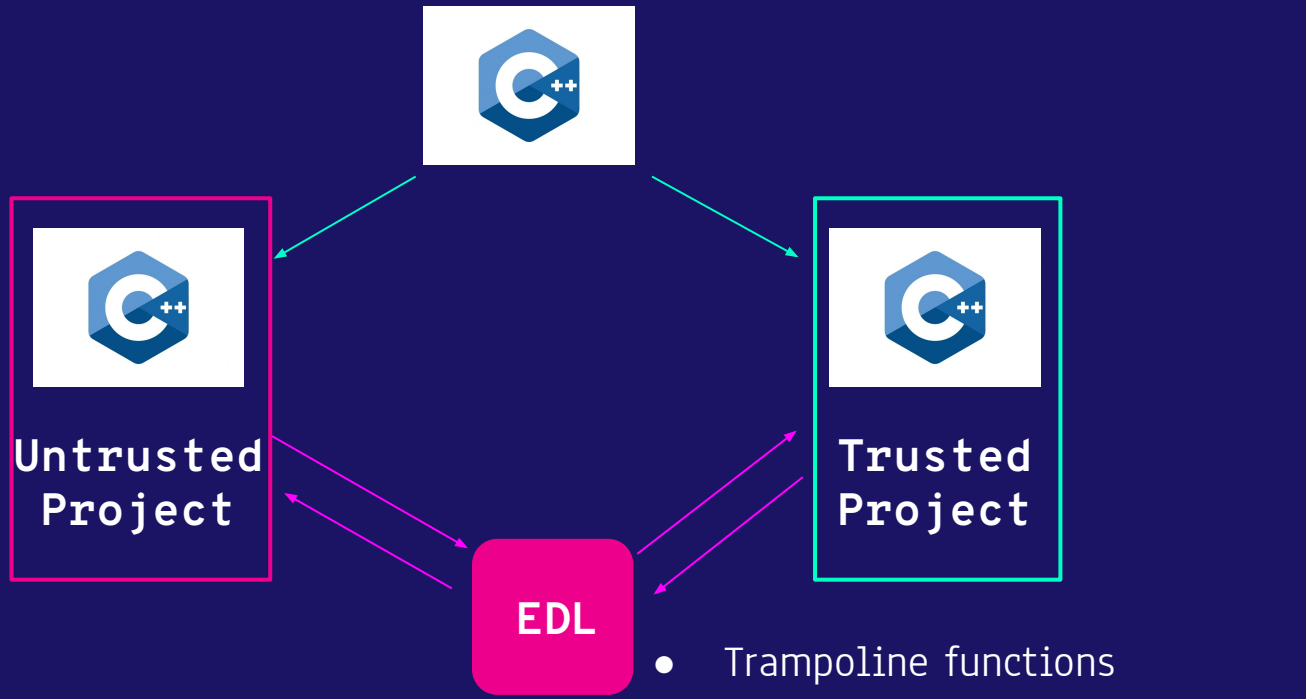
Programming TEEs



Programming TEEs



Programming TEEs



- Trampoline functions
- Arcane Makefiles
- Complex data copying protocol

Programming TEEs

You are passing a double pointer, it is, a pointer to pointer to char (`char **`).

While marshaling/unmarshaling pointers, the EDL Processor processes (copies and validates input and output) only the first level of indirection, it's up to the developer to handle the additional levels of indirection. Hence, for an array of pointers it will only copy the first array of pointers, not the pointed values, copying them is the developer's responsibility. 😢

Source: stackoverflow.com

Secure Program Partitioning

STEVE ZDANCEWIC, LANTIAN ZHENG, NATHANIEL NYSTROM, and ANDREW C. MYERS
Cornell University

Language Support for Secure Software Development with Enclaves

Aditya Oak, *TU Darmstadt*
Amir M. Ahmadian, *KTH Royal Institute of Technology*
Musard Balliu, *KTH Royal Institute of Technology*
Guido Salvaneschi, *University of St.Gallen*

First, **seamless integration** of enclave programming into software applications remains challenging. For example, Intel provides a C/C++ interface to the SGX enclave but no direct support is available for managed languages. As managed languages like Java and Scala are extensively used for developing distributed applications, developers need to either interface their programs with the C++ code executing in the enclave (e.g., using the Java Native Interface [12]) or compile

PtrSplit: Supporting General Pointers in Automatic Program Partitioning

Shen Liu
The Pennsylvania State University
University Park, PA
sxl463@cse.psu.edu

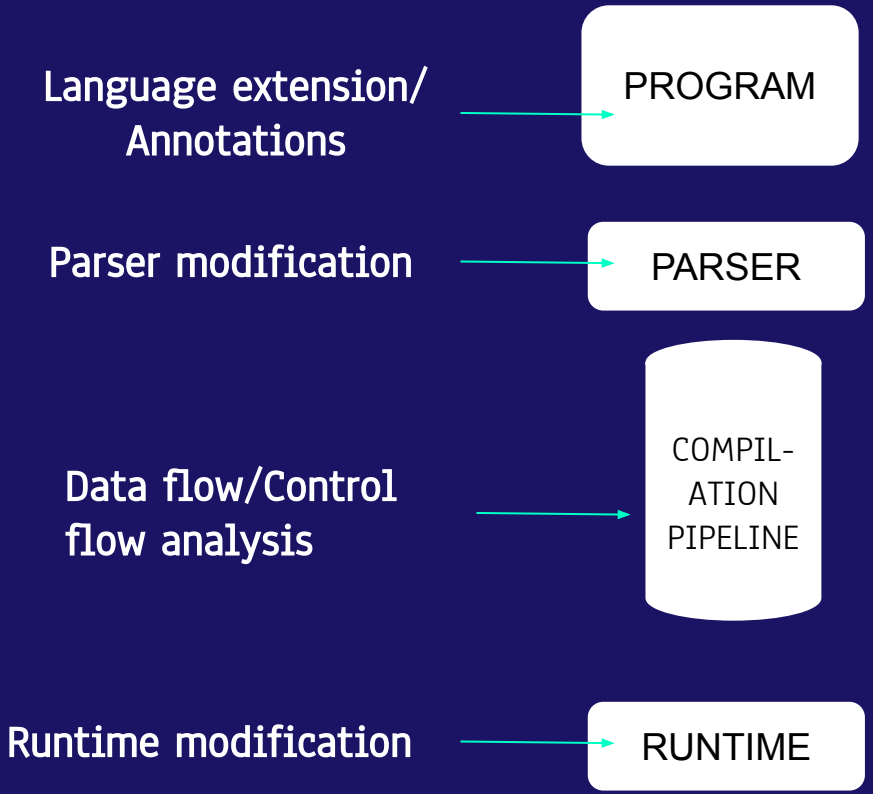
Gang Tan
The Pennsylvania State University
University Park, PA
gtan@

Trent Jaeger
The Pennsylvania State University
University Park, PA

Glamdring: Automatic Application Partitioning for Intel SGX

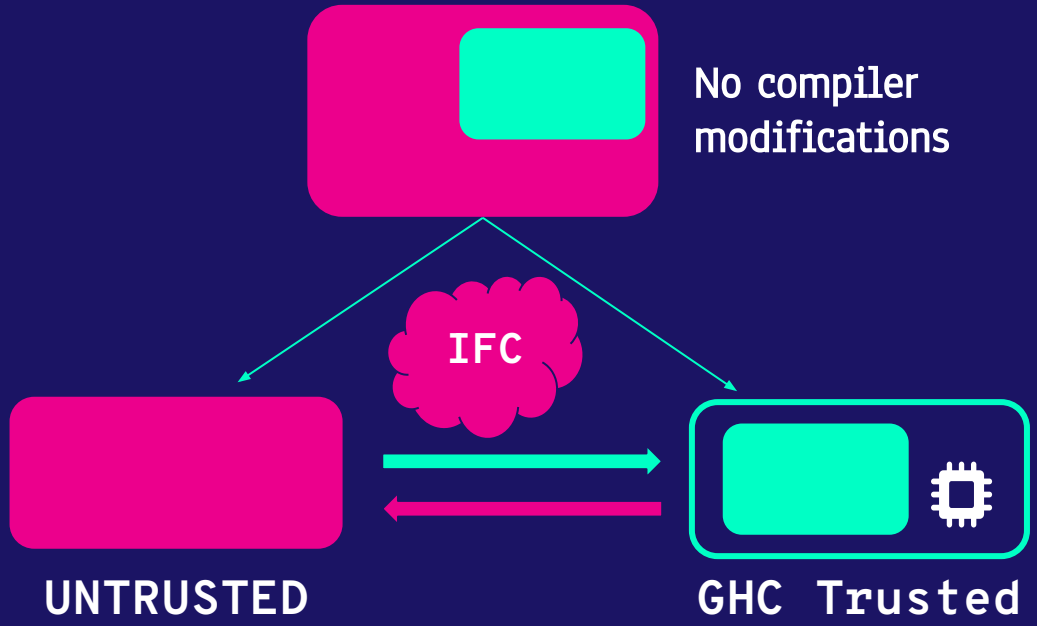
Joshua Lind, *Imperial College London*
Christian Priebe, *Imperial College London*
Divya Muthukumaran, *Imperial College London*
Dan O'Keeffe, *Imperial College London*
Pierre-Louis Aublin, *Imperial College London*
Florian Kelbert, *Imperial College London*
Tobias Reiher, *TU Dresden*
David Goltzsche, *TU Braunschweig*
David Eyers, *University of Otago*
Rüdiger Kapitza, *TU Braunschweig*
Christof Fetzer, *TU Dresden*
Peter Pietzuch, *Imperial College London*

PROGRAM PARTITIONING



HasTEE

HasTEE



Sarkar, A., Krook, R., Russo, A. and Claessen, K., 2023, August. HasTEE: Programming Trusted Execution Environments with Haskell. In *Proceedings of the 16th ACM SIGPLAN International Haskell Symposium* (pp. 72-88).

HasTEE Key Contributions

- Type-driven Partitioning with no compiler modifications
- Program TEEs in a high-level language - Haskell
- Enforce Information Flow Control on data within enclaves

MONAD

```
return :: a → m a

(>>=) :: m a → (a → m b) → m b
```

MONAD


```
(>>=) :: m a → (a → m b) → m b
```

MONAD


```
(>>=) :: m a → (a → m b) → m b
```

TAINT
TRACKING

MONAD

```
return :: a → m a
```

```
(>>=) :: m a → (a → m b) → m b
```

TAINT TRACKING

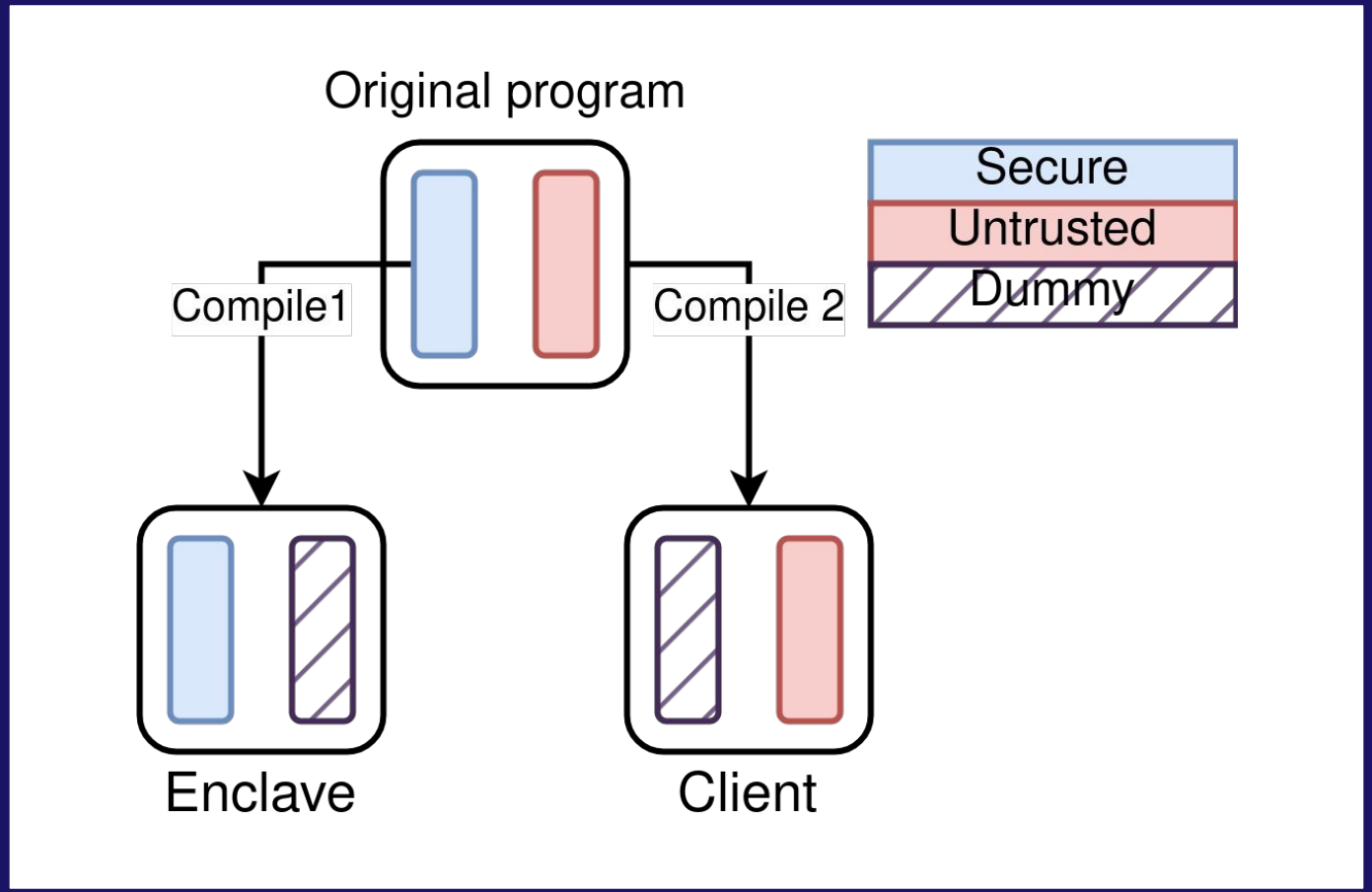
ALTERNATE SEMANTICS

Illustration : Password Checker

```
pwdChkr :: Enclave String -> String -> Enclave Bool
pwdChkr pwd guess = fmap (== guess) pwd

passwordChecker :: App Done
passwordChecker = do
  passwd <- inEnclaveConstant "secret"
  efunc <- inEnclave $ pwdChkr passwd
  runClient $ do -- Client code
    liftIO  $ putStrLn "Enter your password"
    userInput <- liftIO getLine
    res       <- gateway (efunc <@> userInput)
    liftIO $ putStrLn ("Login returned " ++ show res)

main = runApp passwordChecker
```

```
pwdChkr :: Enclave String -> String -> Enclave Bool
pwdChkr pwd guess = fmap (== guess) pwd

passwordChecker :: App Done
passwordChecker = do
  passwd <- inEnclaveConstant "secret"
  efunc <- inEnclave $ pwdChkr passwd
  runClient $ do -- Client code
    liftIO $ putStrLn "Enter your password"
    userInput <- liftIO getLine
    res       <- gateway (efunc <@> userInput)
    liftIO $ putStrLn ("Login returned " ++ show res)

main = runApp passwordChecker
```

COMPILED TWICE


Original program
Secure
Untrusted
Dummy
Compile1
Compile 2
Enclave
Client

Compilation 1

```
-- Enclave
pwdChkr :: Enclave String -> String -> Enclave Bool
pwdChkr pwd guess = fmap (== guess) pwd

passwordChecker :: App Done
passwordChecker = do
  passwd <- inEnclaveConstant "secret"
  efunc <- inEnclave $ pwdChkr passwd
  return DONE

-- wait for calls from Client
main = runApp passwordChecker
```

Compilation 2

```
-- Client
pwdChkr = -- gets optimised away

passwordChecker :: App Done
passwordChecker = do
  passwd <- return Dummy
  efunc <- inEnclave $ -- ignores pwdChkr body
  runClient $ do -- Client code
    liftIO  $ putStrLn "Enter your password"
    userInput <- liftIO getLine
    res       <- gateway (efunc <@> userInput)
    liftIO $ putStrLn ("Login returned " ++ show res)

-- drives the application
main = runApp passwordChecker
```

Compilation 1

```
-- Enclave
pwdChkr :: Enclave String -> String -> Enclave Bool
pwdChkr pwd guess = fmap (== guess) pwd

passwordChecker :: App Done
passwordChecker = do
  passwd <- inEnclaveConstant "secret"
  efunc <- inEnclave $ pwdChkr passwd
  return DONE

-- wait for calls from Client
main = runApp passwordChecker
```

Compilation 2

```
-- Client
pwdChkr = -- gets optimised away

passwordChecker :: App Done
passwordChecker = do
  passwd <- return Dummy
  efunc <- inEnclave $ -- ignores pwdChkr body
  runClient $ do -- Client code
    liftIO  $ putStrLn "Enter your password"
    userInput <- liftIO getLine
    res       <- gateway (efunc <@> userInput)
    liftIO $ putStrLn ("Login returned " ++ show res)

-- drives the application
main = runApp passwordChecker
```

Compilation 1

```
-- Enclave
pwdChkr :: Enclave String -> String -> Enclave Bool
pwdChkr pwd guess = fmap (== guess) pwd

passwordChecker :: App Done
passwordChecker = do
  passwd <- inEnclaveConstant "secret"
  efunc <- inEnclave $ pwdChkr passwd
  return DONE

-- wait for calls from Client
main = runApp passwordChecker
```

GHC Trusted

INTEL SGX

Information Flow Control

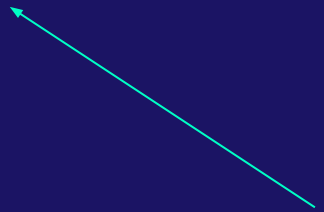
Information Flow Control

Information Flow Control

```
gateway :: (Binary a) => Secure (Enclave a) → Client a
```

Information Flow Control

```
gateway :: (Binary a) => Secure (Enclave a) → Client a
```



Lack of a Binary instance
prevents accidental leaks

Information Flow Control

Enclave monad restricted
using a RestrictedIO typeclass

```
gateway :: (Binary a) => Secure (Enclave a) → Client a
```

```
type RestrictedIO m = (RandomIO m, FileIO m, ..)

class FileIO m where
  readFile :: FilePath -> m String

class RandomIO m ...
```

Non-interference Proposition

p :: Enclave a -> App Done
p has no `gateway` operation

e1 :: Enclave a

p e1

side ≈ effect

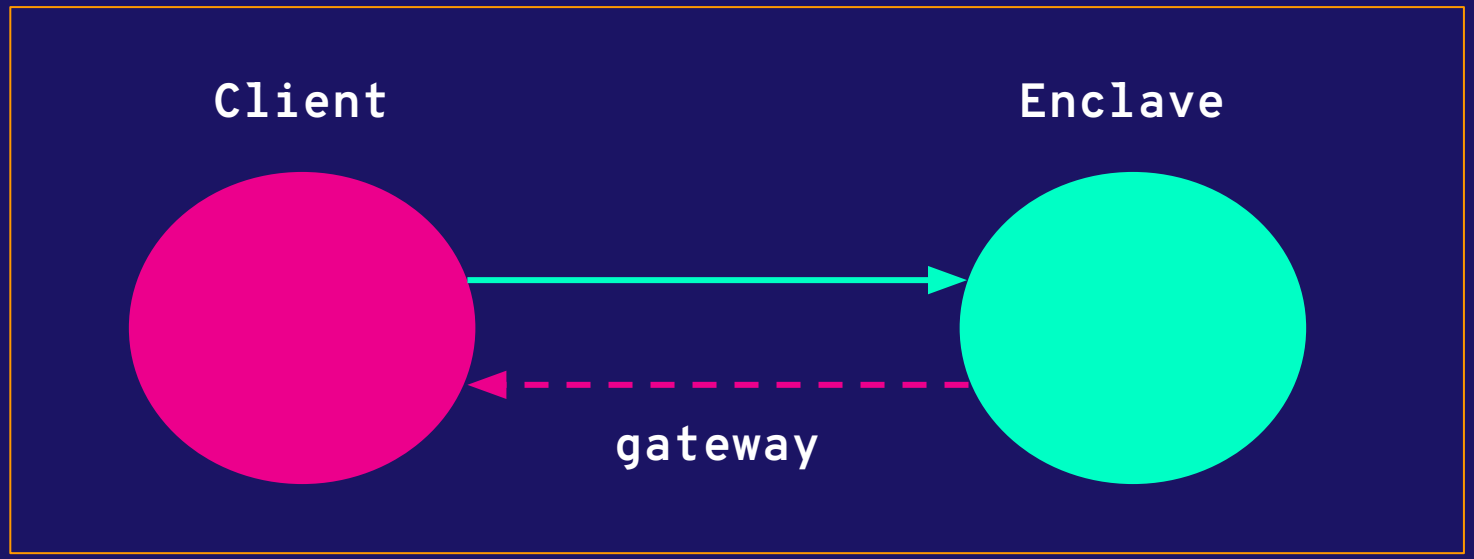
e2 :: Enclave a

p e2

Code Auditing

```
78    res <- gateway rem_srv
79    case res of
80      Nothing -> do
81        liftIO $ threadDelay 1000000
82        retrygateway rem_srv
83      Just a -> return a
84
85  computeGradient :: API
86                  -> P.PrvKey
87                  -> Config
88                  -> Matrix Double
89                  -> Vector Int
90                  -> Client (Vector CT)
91  computeGradient api prv cfg x y = do
92    let m = ncols x
93    prod  <- dotprodHEC api pubK (weights cfg) x
94    yPred <- V.mapM (sigmoid_taylor_expandC api pubK) prod
95    yEnc  <- V.mapM ((\e -> liftIO $ encrypt pubK e) . int2Double) y
96    prod' <- do
97      op <- subPHE yPred yEnc
98      dotprodHEC api pubK op (M.transpose x)
99    gateway $ recryptMany api <@> V.map (\c -> homoMul pubK c (1 / (fromIntegral m))) prod'
100   where
101     pubK = pubKey cfg
102     subPHE xs ys = gateway $ recryptMany api <@> V.zipWith (\v1 v2 -> homoSub pubK v1 v2) xs ys
103
104 -- | Dot product that reencrypts the CT between any two operations
105 dotprodHEC :: API
106            -> P.PubKey
107            -> V.Vector CT
108            -> Matrix Double
109            -> Client (V.Vector CT)
110 dotprodHEC api pubk w x = do
111   -- let zero = go2I 0.0
112   enc_zero <- liftIO $ encrypt pubk 0.0
113   vec <- flip Prelude.mapM [1..i] $ \i' -> do
114     let dots = V.zipWith (\d cipher -> homoMul pubk cipher d) (getRow i' x') w
115     dots' <- gateway $ recryptMany api <@> dots
116     V.foldM (\c c' -> gateway $ recrypt api <@> homoAdd pubk c c') enc_zero dots'
117   return $ V.fromList vec
118    where
119      x' = transpose x
120      i  = nrows x'
121
122 sigmoid_taylor_expandC :: API -> P.PubKey -> CT -> Client CT
123 sigmoid_taylor_expandC api pubK cipher = do
124   let val = 0.5
125   enc_val <- liftIO $ encrypt pubK val --go2I val
126   first <- gateway $ recrypt api <@> homoMul pubK cipher 0.25
127   gateway $ recrypt api <@> homoAdd pubK enc_val first
128
129 updateModel :: API -> Config -> Vector CT -> Client Config
130 updateModel api cfg grad = do
131     let lr = learningRate cfg / sqrt (1 + fromIntegral (iterN cfg))
132     gr <- do
133       let op = V.map (\w -> homoMul pubK w (alpha cfg)) (weights cfg)
134       op' <- gateway $ recryptMany api <@> op
135       addP grad op'
136     nw <- do
```

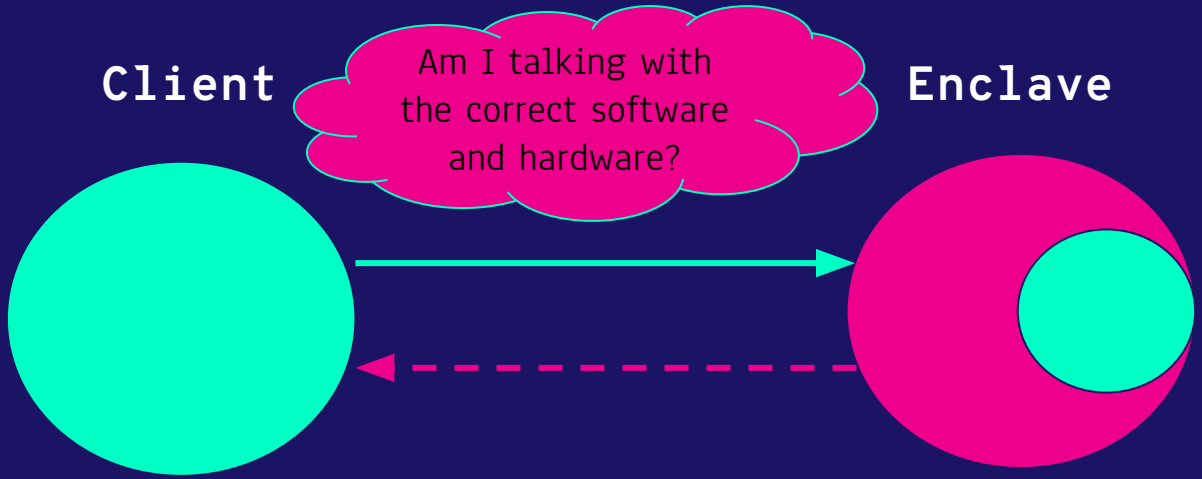
Information Flow Control



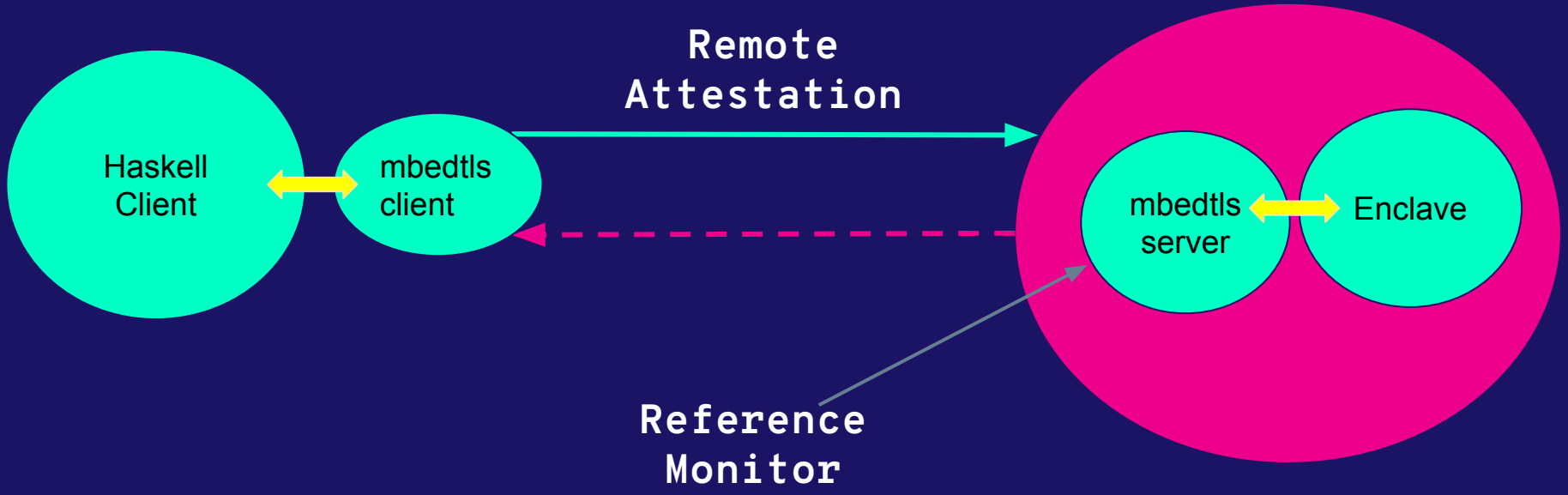
Single Machine

Cloud Computing

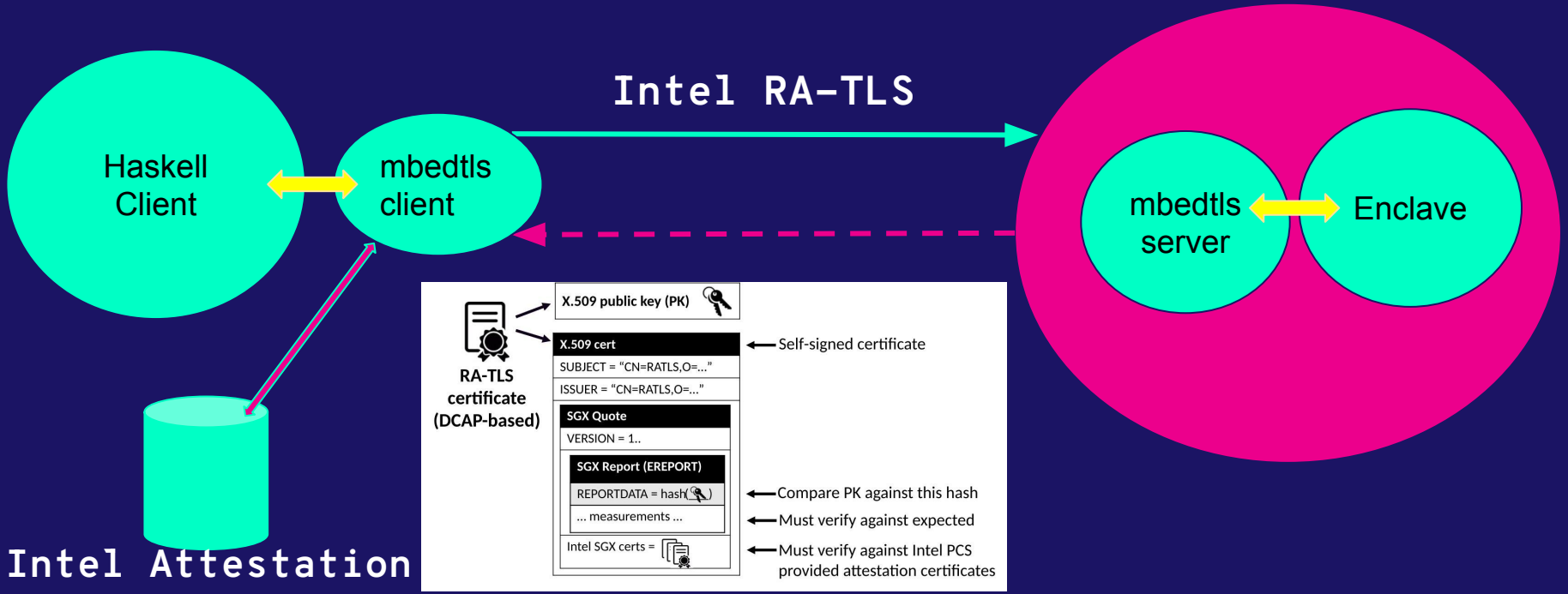
Cloud Computing



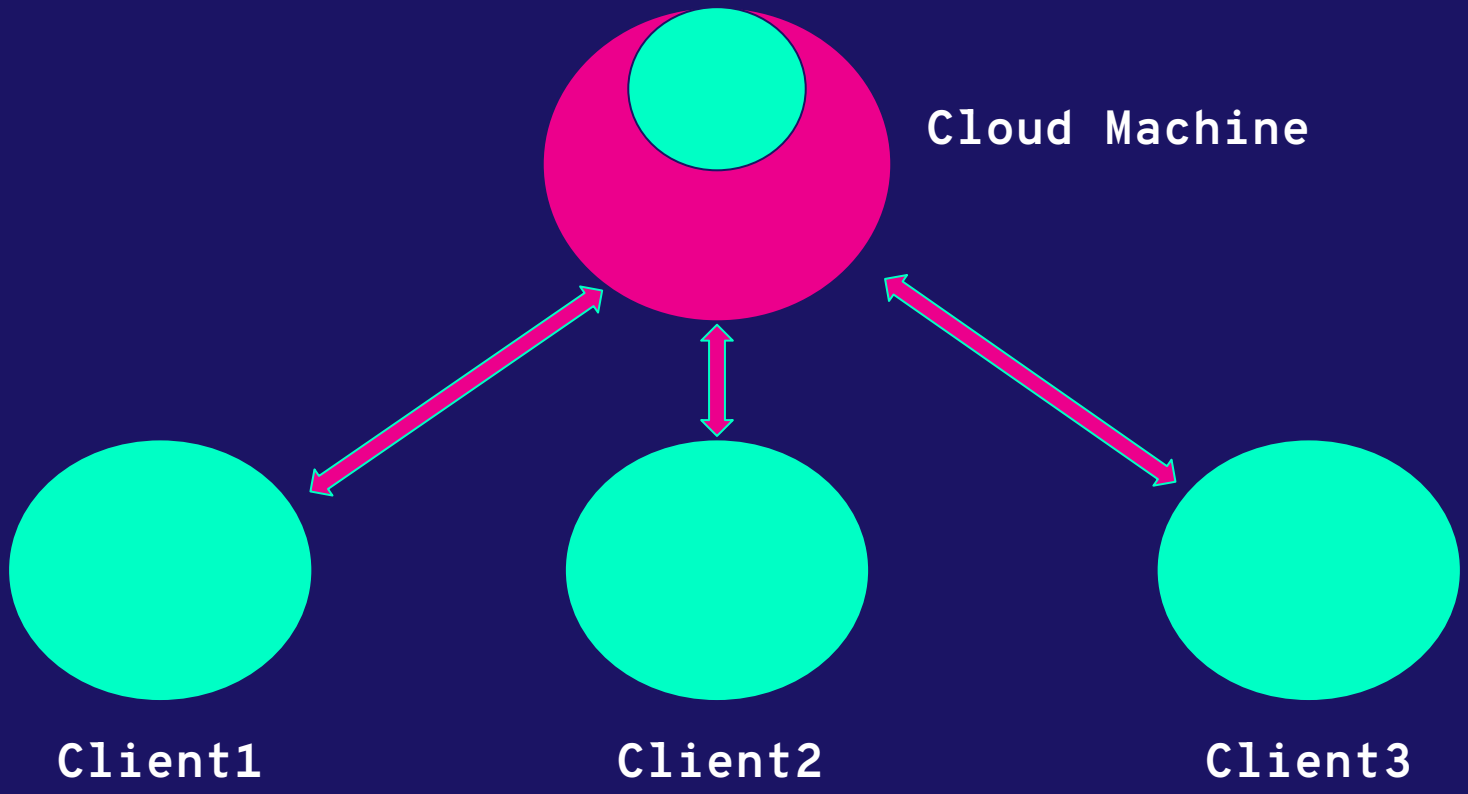
Cloud Computing



Cloud Computing

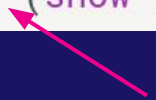


Cloud Computing



Information Flow Control

```
pwdChecker :: Enclave String -> String -> Enclave Bool
pwdChecker pwd guess = do
  p <- pwd
  if (p == guess)
  then return True
  else return False


...
...
res <- gateway.. call pwdChecker..
putStrLn (show res)
```



Violates non-interference
Public output influenced by secret

Information Flow Control

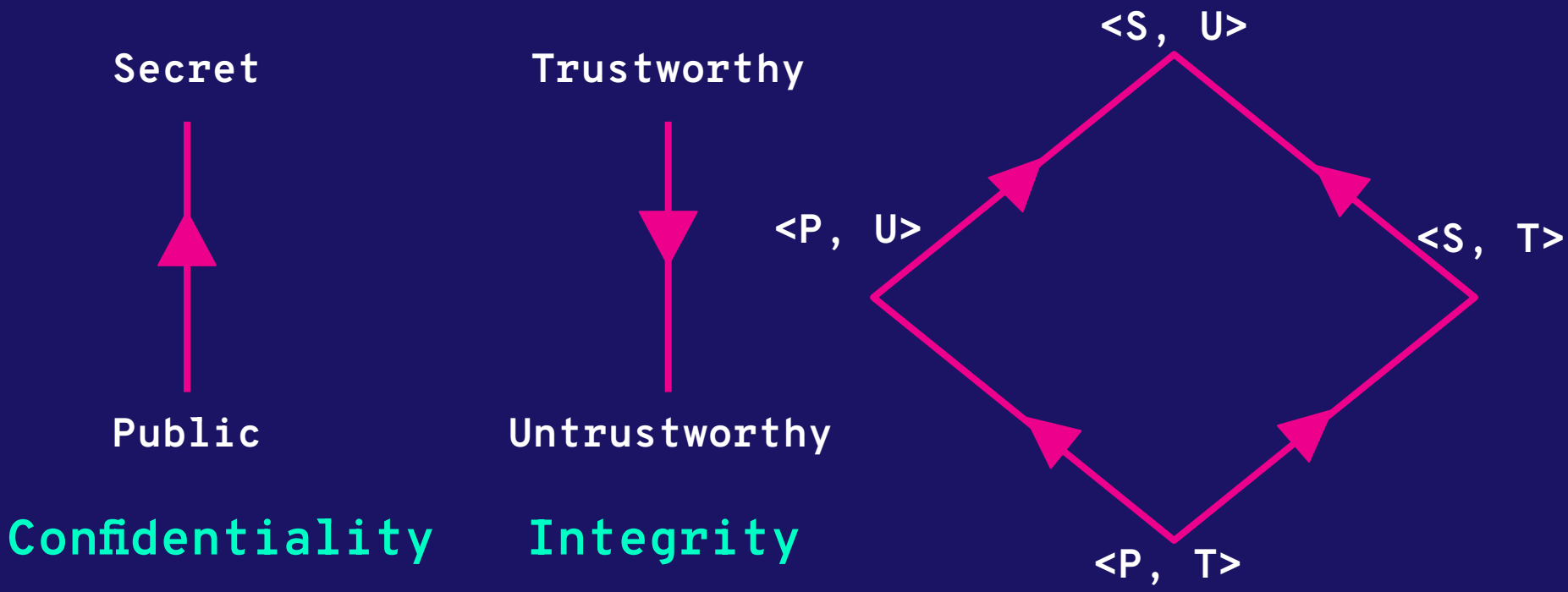
```
pwdChecker :: Enclave String -> String -> Enclave Bool
pwdChecker pwd guess = do
  p <- pwd
  if (p == guess)
  then return True
  else return False


...
...
res <- gateway.. call pwdChecker..
putStrLn (show res)
```

No way to distinguish data source

Information Flow Control

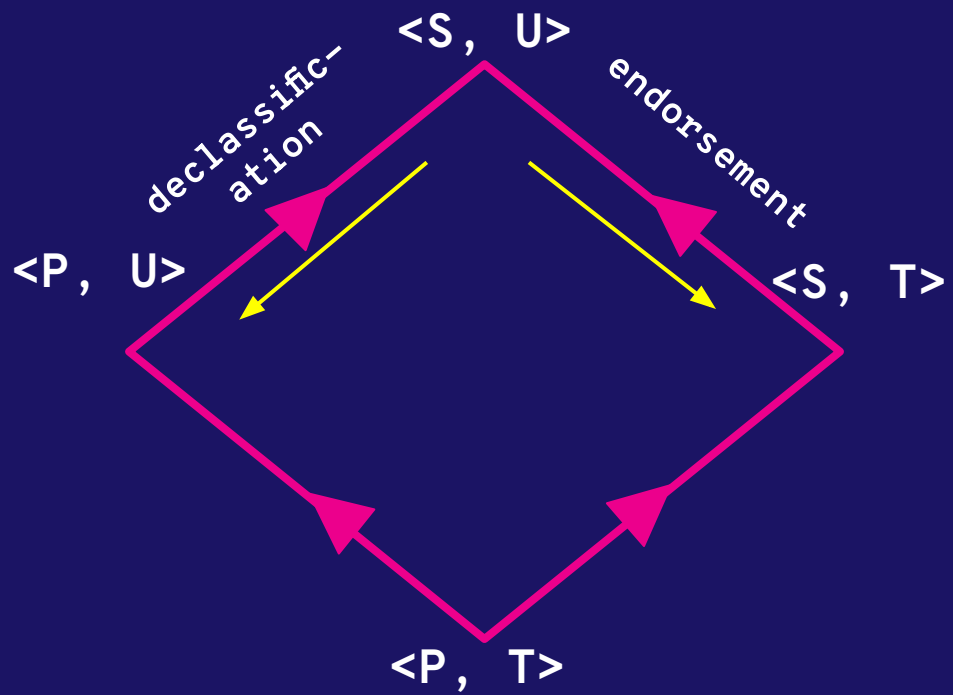
```
class Label l where
  lub       :: l -> l -> l
  glb       :: l -> l -> l
  canFlowTo :: l -> l -> Bool

Enclave l a -- parameterised on Label l
Labeled l a -- parameterised on Label l
```

Information Flow Control

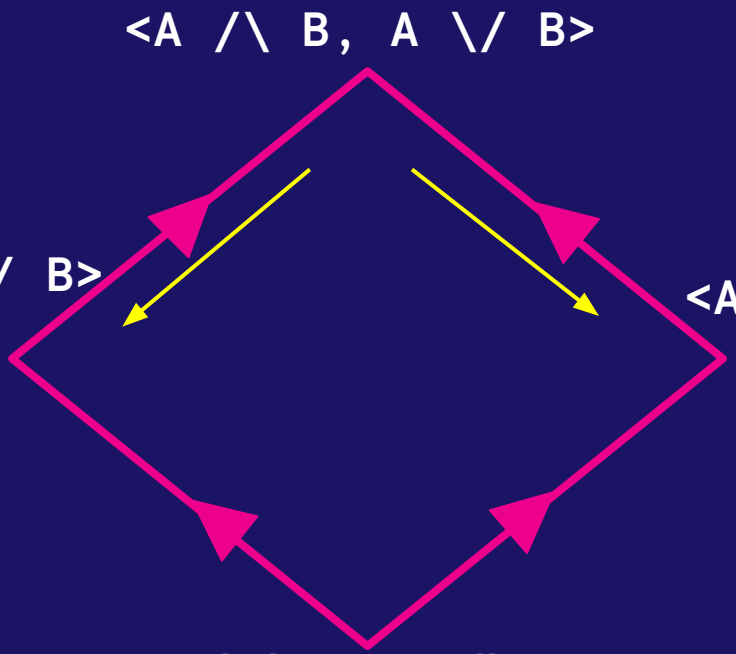


Denning, Dorothy E. "A lattice model of secure information flow." *Communications of the ACM* 19.5 (1976): 236-243.
Biba, K.J. Integrity considerations for secure computer systems. Technical Report. April 1977.

Information Flow Control



Disjunction Category Labels



A and B are principals

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*

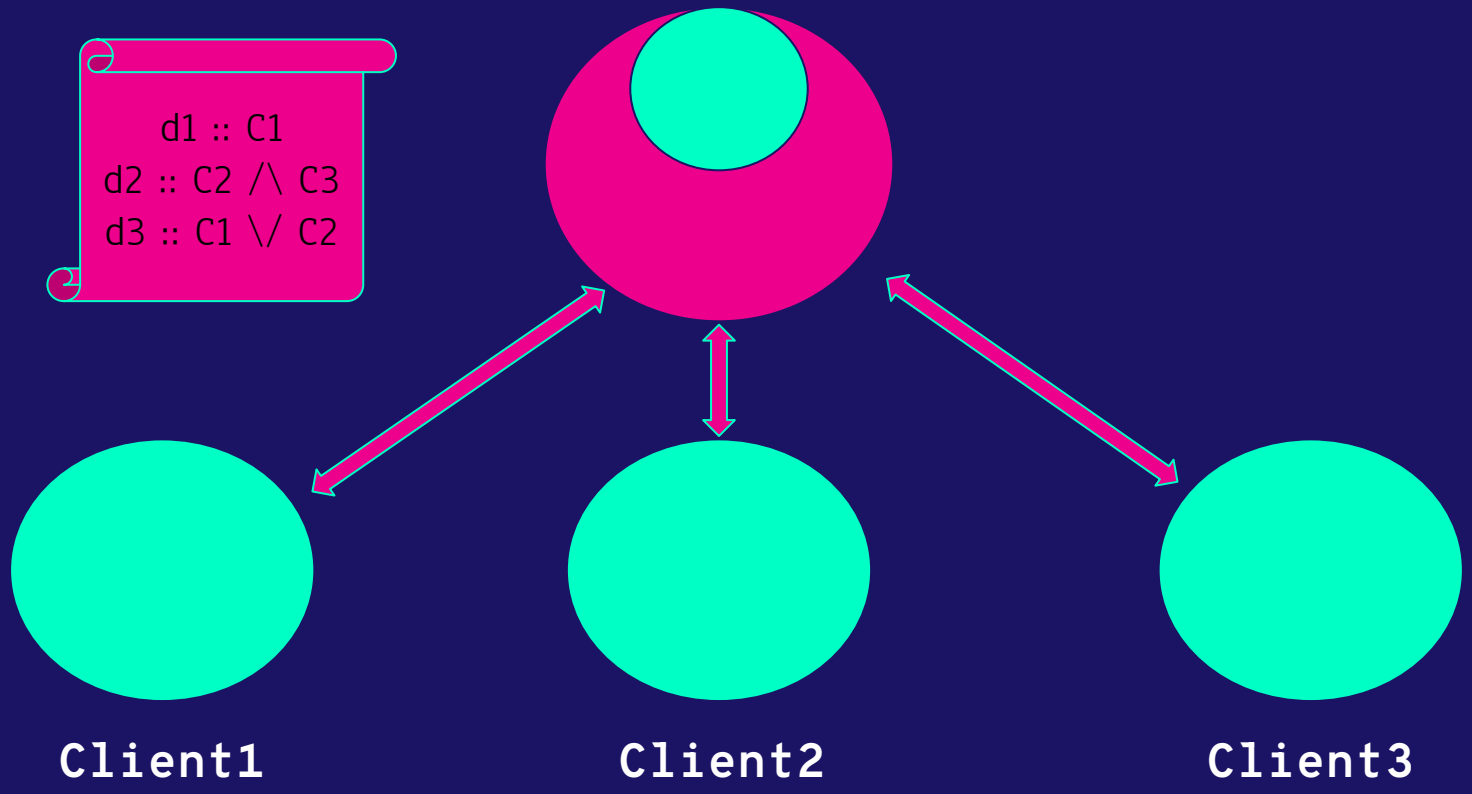
Information Flow Control

```
taint     :: Label l => l -> Enclave l ()
label     :: Label l => l -> a -> Enclave l (Labeled l a)
unlabel   :: Label l => Labeled l a -> Enclave l a
labelOf   :: Label l => Labeled l a -> l
toLabeled :: Label l => l -> Enclave l a -> Enclave l (Labeled l a)
```

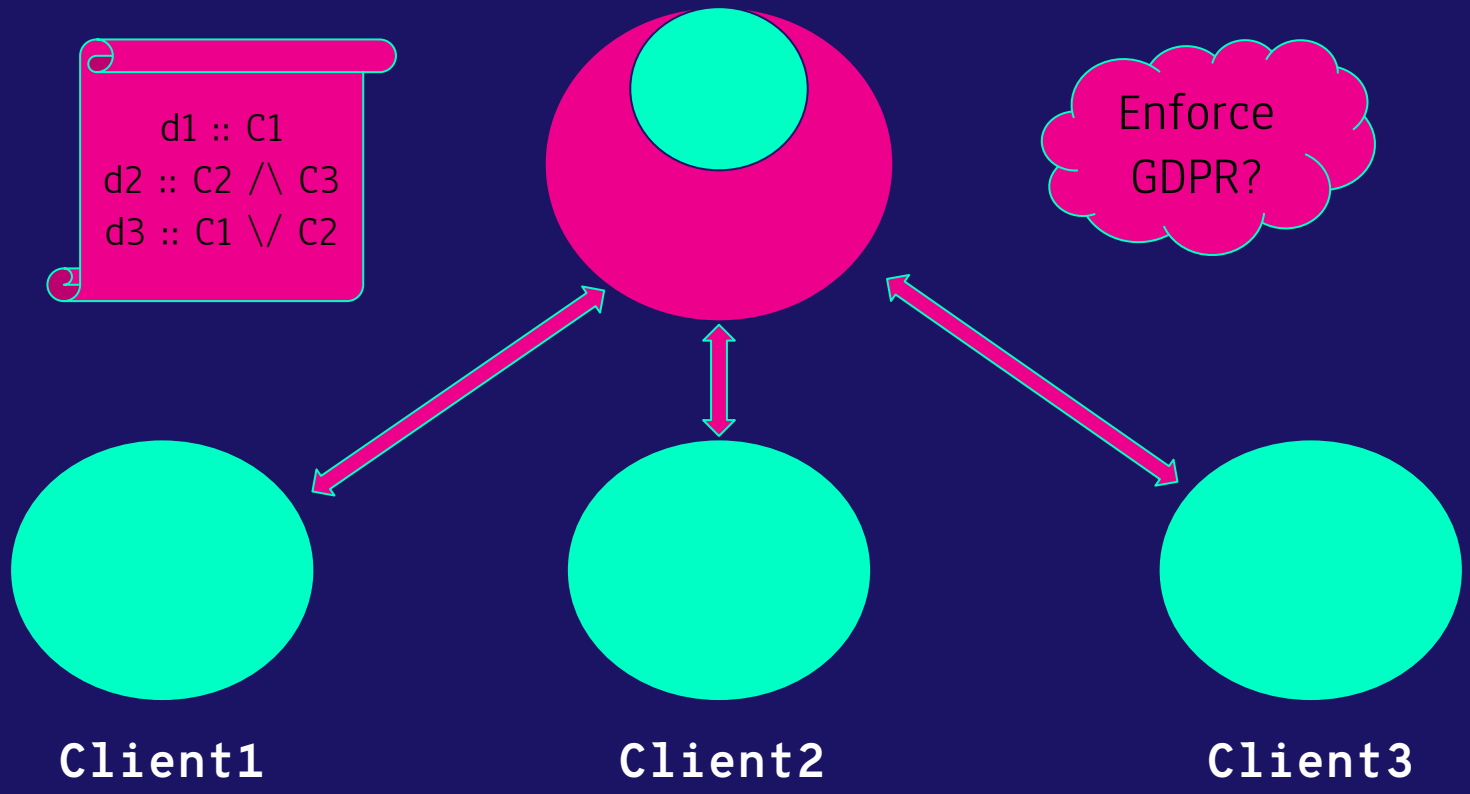
Supports declassification and endorsement via privileges (capabilities)

DYNAMIC CHECKS for dynamically changing policies

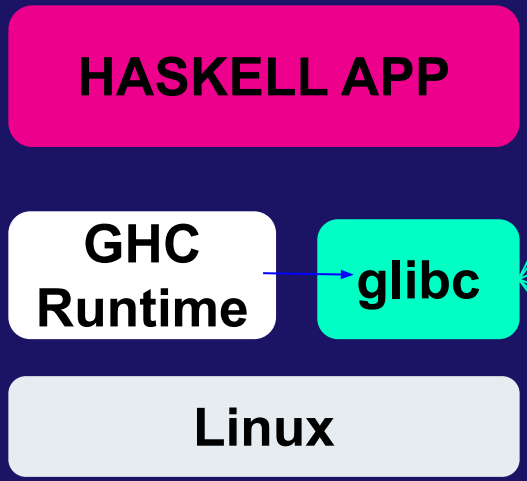
Data Clean Room



Data Clean Room

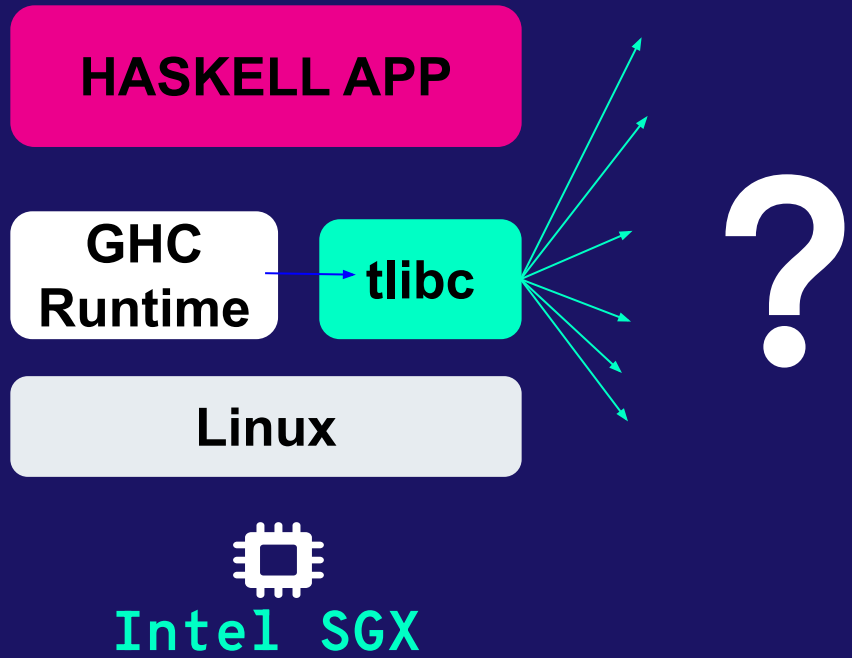


IMPLEMENTATION



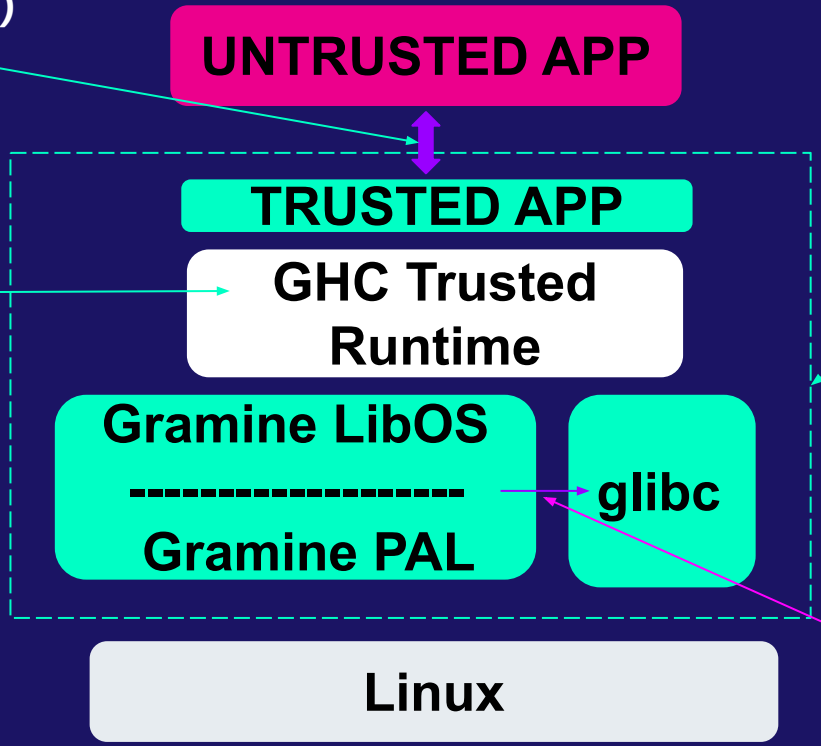
- mmap
- madvise
- getrusage
- poll
- select
- clock apis
- pthread apis
- libm apis
- …

IMPLEMENTATION



IMPLEMENTATION

IPC (local)
mbedTLS (remote)



Patched mmap,
select, etc

Trusted
Code Base ~
100K LOC

40 system
calls

PERFORMANCE

Memory	RSS	Virtual Size	Disk Swap
At rest	19,132 KB	287,920 KB	0 KB
Peak	20,796 KB	290,032KB	0 KB

PERFORMANCE

Memory	RSS	Virtual Size	Disk Swap
At rest	19,132 KB	287,920 KB	0 KB
Peak	20,796 KB	290,032KB	0 KB

Enclave Page Cache
size = 93MB

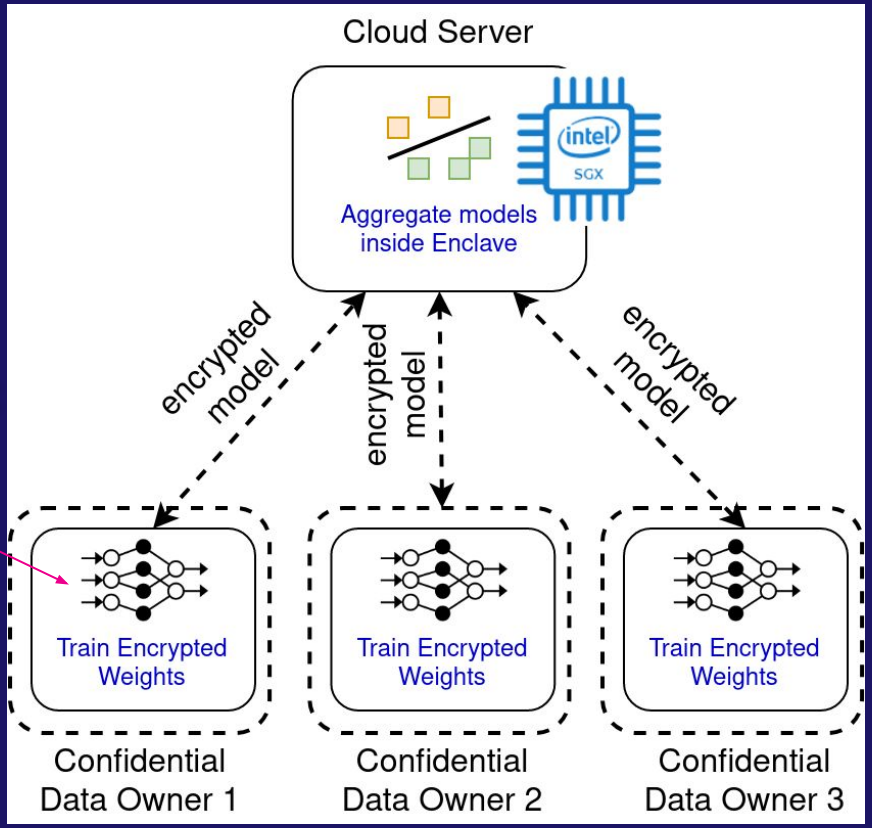
PERFORMANCE

Memory	RSS	Virtual Size	Disk Swap
At rest	19,132 KB	287,920 KB	0 KB
Peak	20,796 KB	290,032KB	0 KB

LATENCY ~ 60 ms
vs
0.6 ms in native SDK

Zero Trust Federated Learning

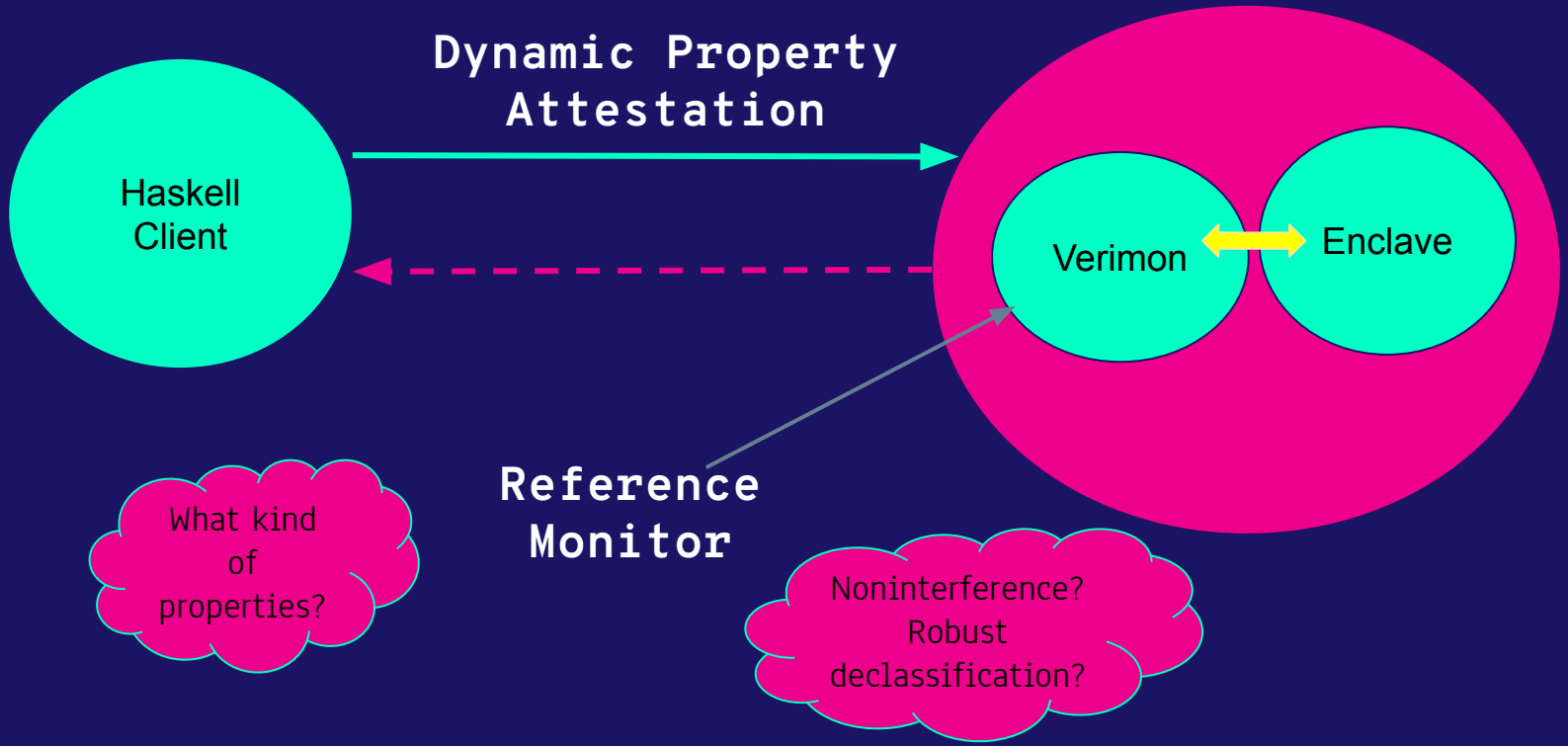
Uses homomorphic encryption for training



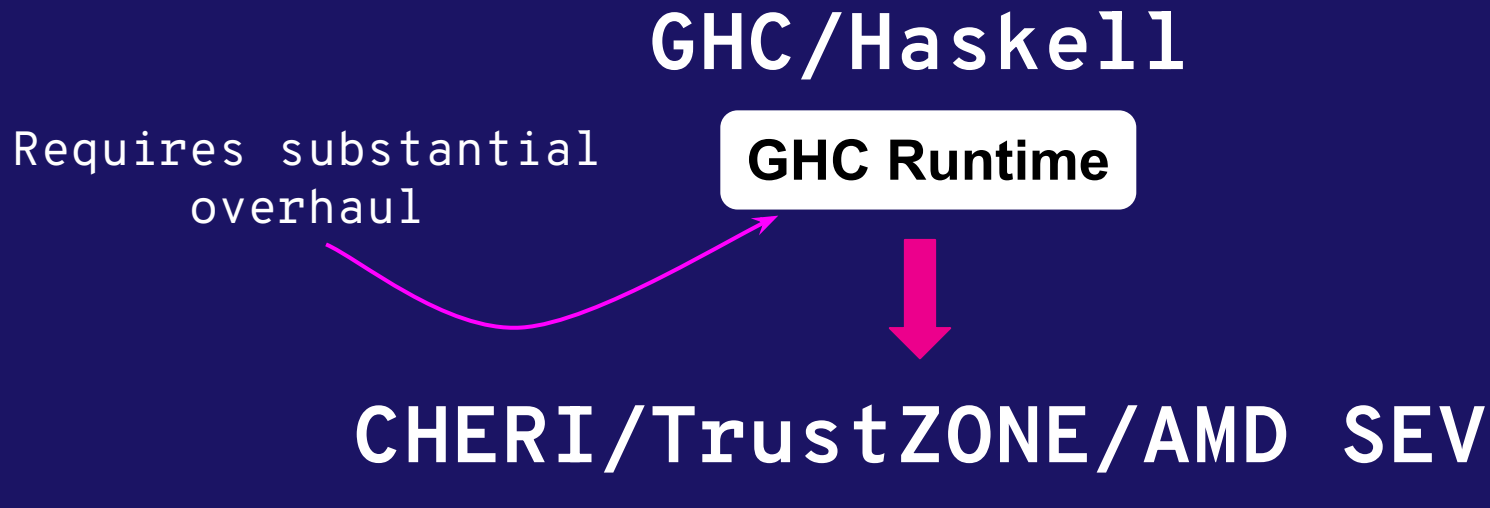
Applications

- Privacy-preserving Federated Learning
- Encrypted Password Wallet
- Data Clean Room with Differential Privacy

FUTURE WORK


Dynamic Property Attestation
Haskell Client
Verimon
Enclave
Reference Monitor
What kind of properties?
Noninterference? Robust declassification?

FUTURE WORK


GHC/Haskell
Requires substantial overhaul
GHC Runtime
CHERI/TrustZONE/AMD SEV

THANKS !

https://github.com/Abhiroop/HasTEE