Security:
Trusted execution environments
Ryoan

Emmett Witchel

CS380L

Big tech has a poor track record for trust

- Administrators have a lot of control and sometimes misbehave
- Cloud providers have competing interests
- Data is valuable and there are buyers
- Cloud providers are a high value target for attacks

But public clouds are useful

- Provide rapid, elastic access to resources
- Handle administration
- Ensure resources are available reliably
- Have large machines and accelerators like GPUs

The market agrees; public clouds made $105 billion last year

Objective: make cloud computing an option for users with sensitive data

Requirements:

- Do not trust the cloud provider
  - History tells us the cloud provider is not trustworthy
- Performance must be reasonable
  - Users can always buy their own machines

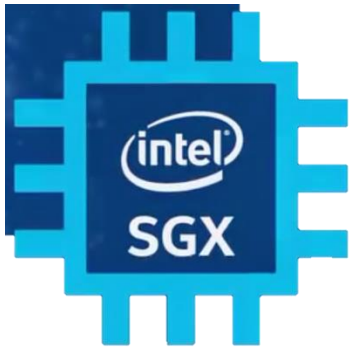
Trusted Execution Environments

- Support common/important use cases
  - Niche applications have niche appeal

System design

Trusted Execution Environments (TEEs)

- Hardware isolation mechanism that cannot be bypassed by software
  - Necessary since the cloud provider controls the OS and Hypervisor
- Existing CPU TEEs: Intel SGX, Arm TrustZone, RISC-V Keystone
- Proposed GPU TEEs: Graviton [Volos et. al, OSDI`18], HIX [Jang et. al, ASPLOS`19]



arm
TRUSTZONE

TEEs are a performant mechanism for keeping secrets from the cloud provider

- Memory is isolated from all external code
  - I.e., only code inside a TEE can access or modify its state
- TEEs operate at near-native speeds
- Trusted attestation prevents hardware spoofing

TEEs are not the silver bullet

- Micro-architectural side channels
- Memory limits

Hardware oversights

- Users must vet TEE code
  - TEE code can misbehave and leak secrets
- TEE guarantees end at the device boundary
  - Workloads with accelerators must compose TEEs

Fundamental design issues

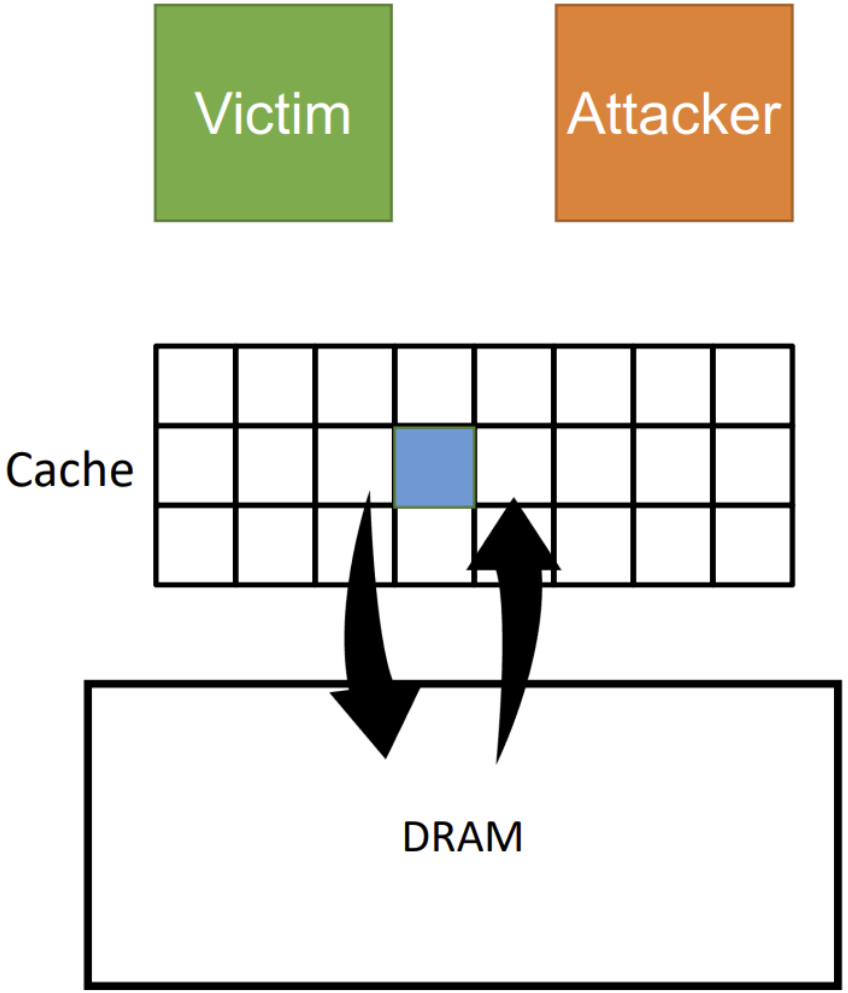
Our contribution: Augment TEE security with systems designed to protect applications

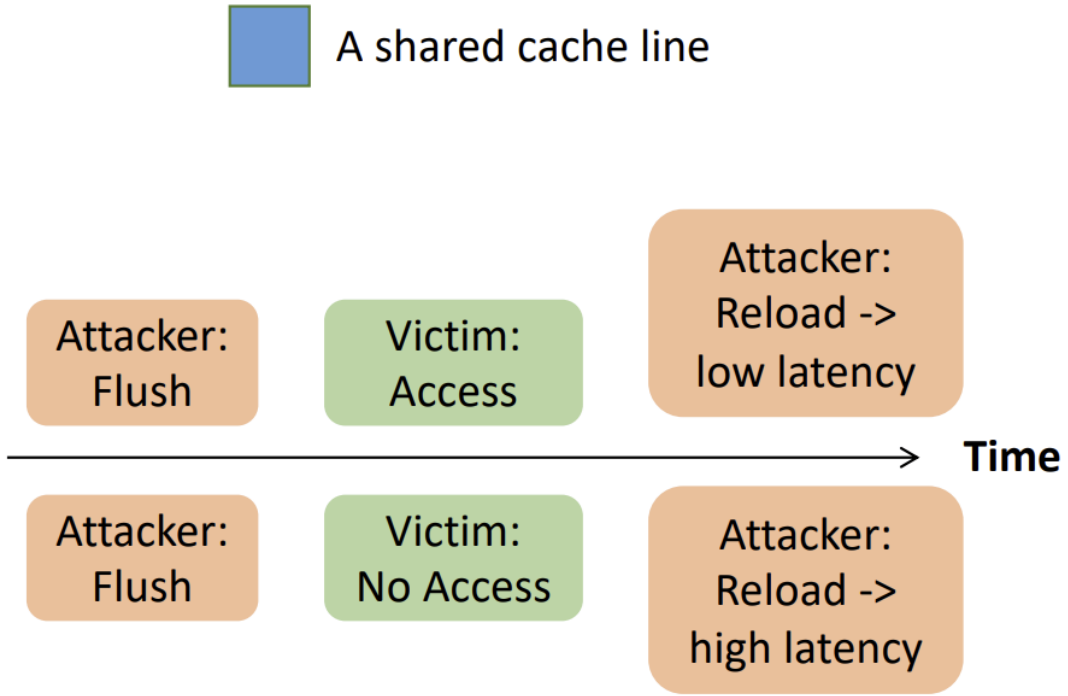
Ryoan	• Users must vet TEE code - TEE code can misbehave and leak secrets	***Applications are often proprietary***
Telekine	• TEE guarantees end at the device boundary - Workloads with accelerators must compose TEEs	***Communication exposes new timing channels***

Micro-architectural side-channel attacks

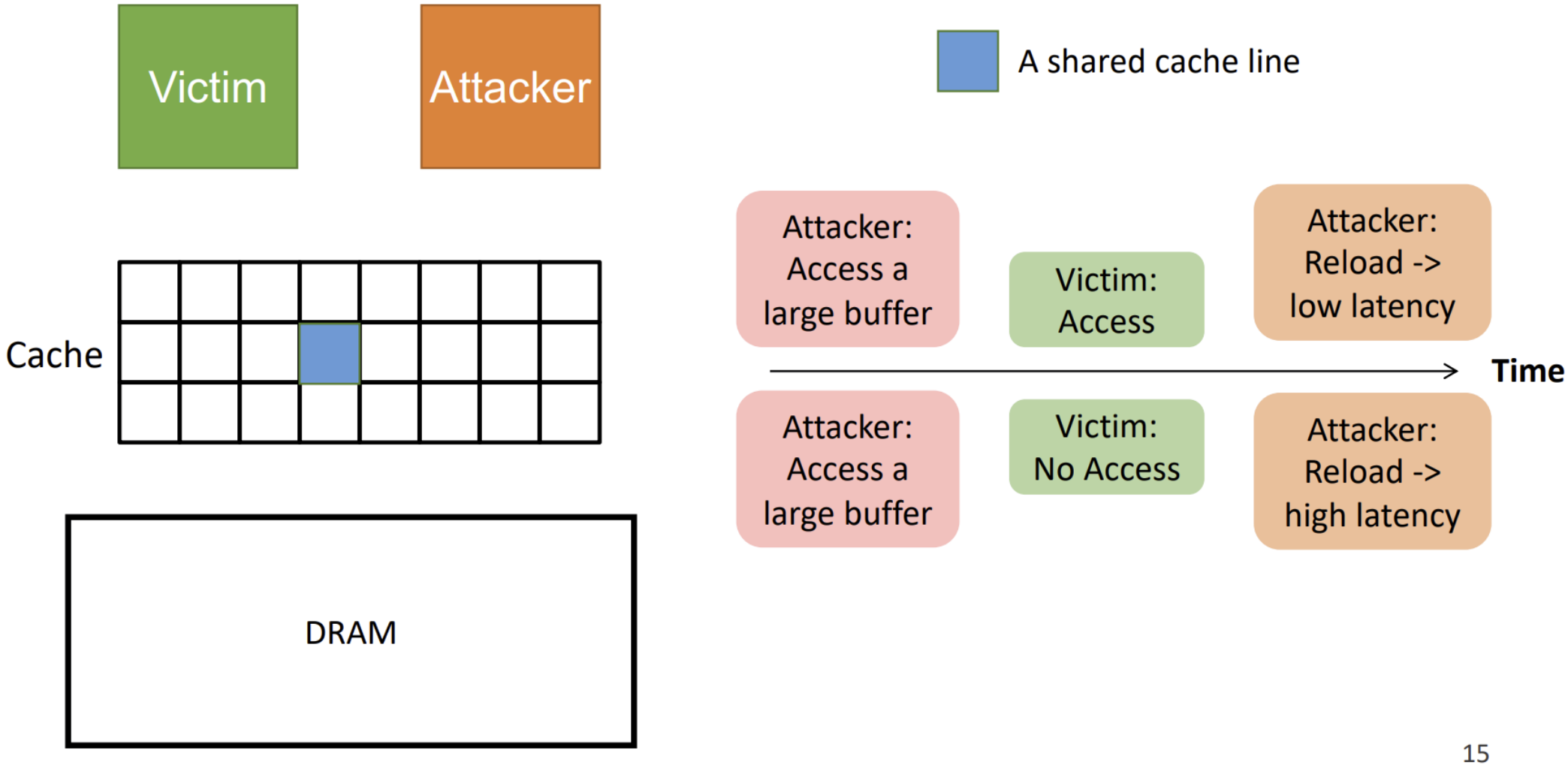
- Micro-architectural side-channel attacks refer to a side-channel attack that exploits information leakage from the hardware infrastructure itself.
  - The attacks can be found in a large scope of devices - servers, workstations, laptops, smart-phones, etc.
- A side-channel attack is any attack based on extra information that can be gathered because of the fundamental way a computer protocol or algorithm is implemented (e.g., time, power consumption, sound), rather than flaws in the design of the protocol or algorithm itself.

Flush+Reload





Attack Strategy #2: Evict+Reload



- TEEs allow you to run trusted code on untrusted infrastructure
  - Give an example of where a TEE would be useful to a computation
  - What security guarantee does Ryoan provide?

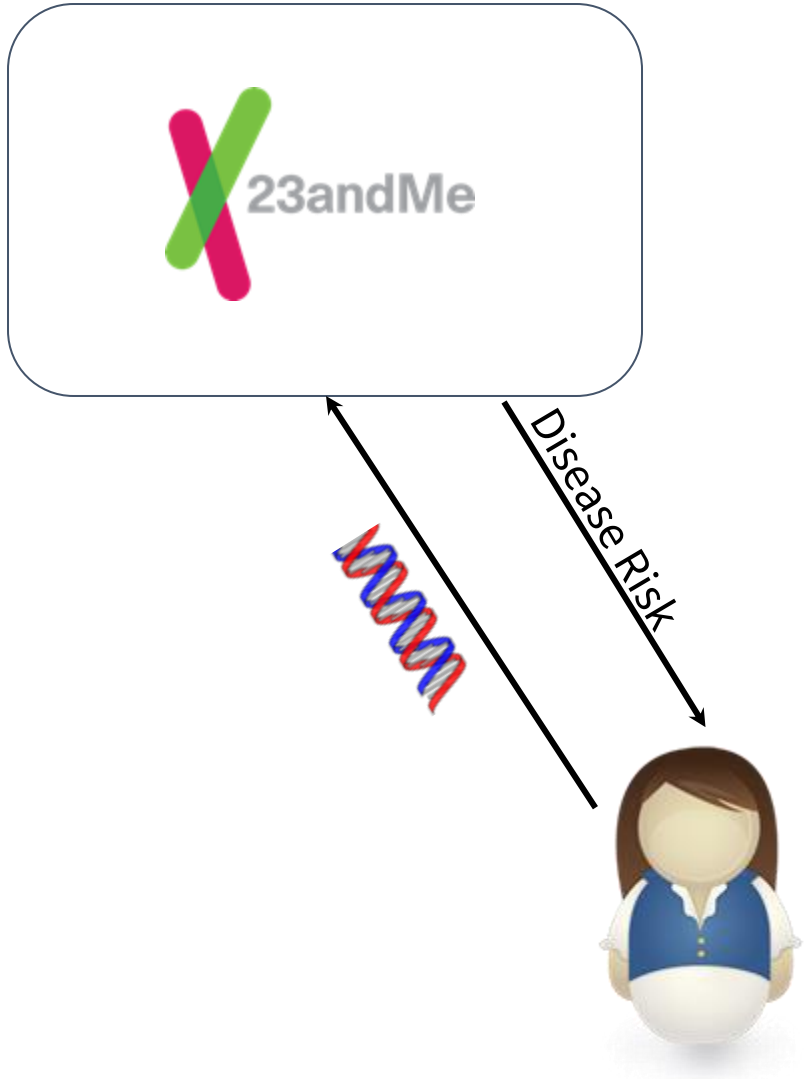
The University of Texas at Austin

Ryoan: A Distributed sandbox for Untrusted Computation on Secret Data

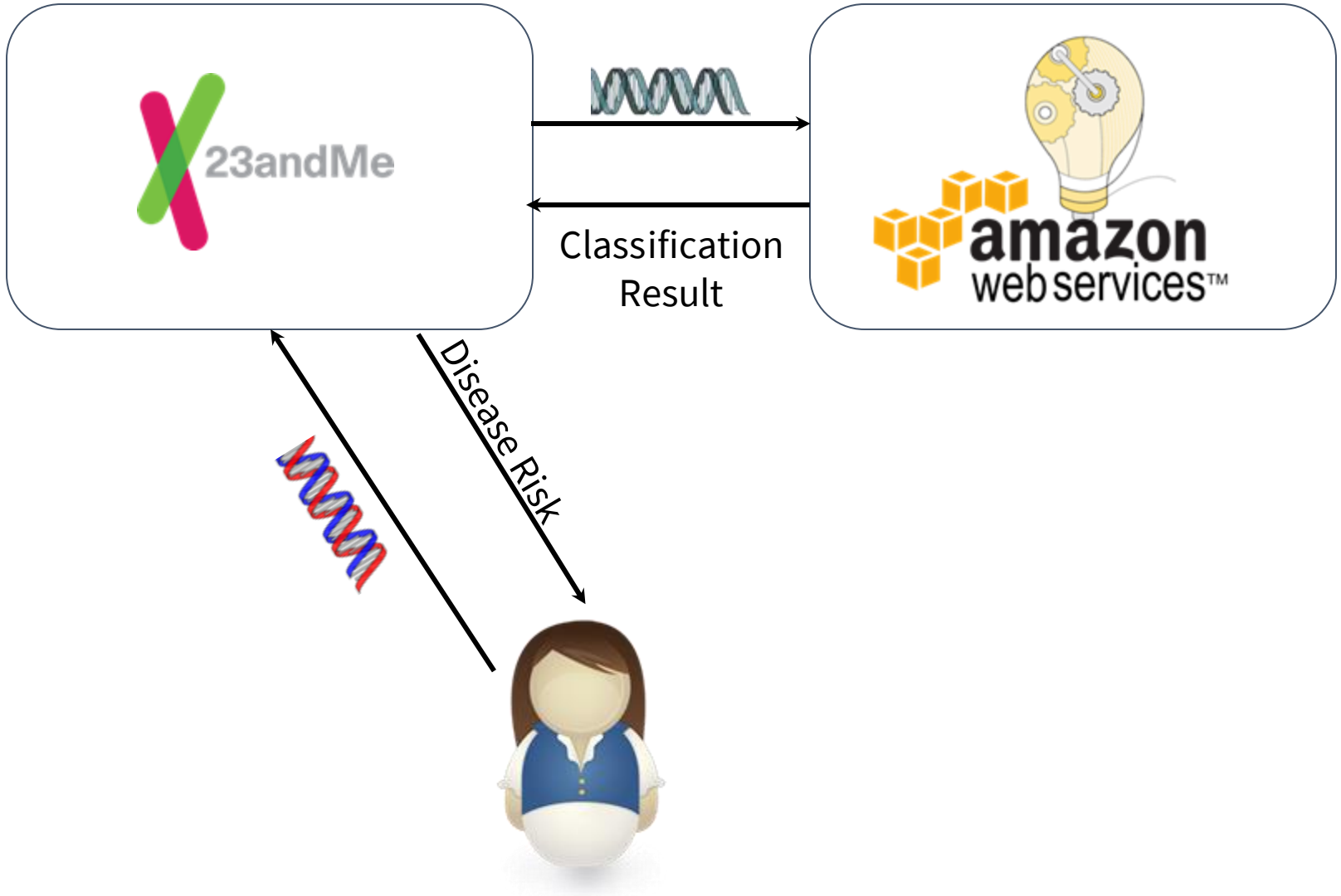
OSDI 2016 (Best Paper)

Tyler Hunt, Zhiting Zhu, Yuanzhong Xu,
Simon Peter, Emmett Witchel

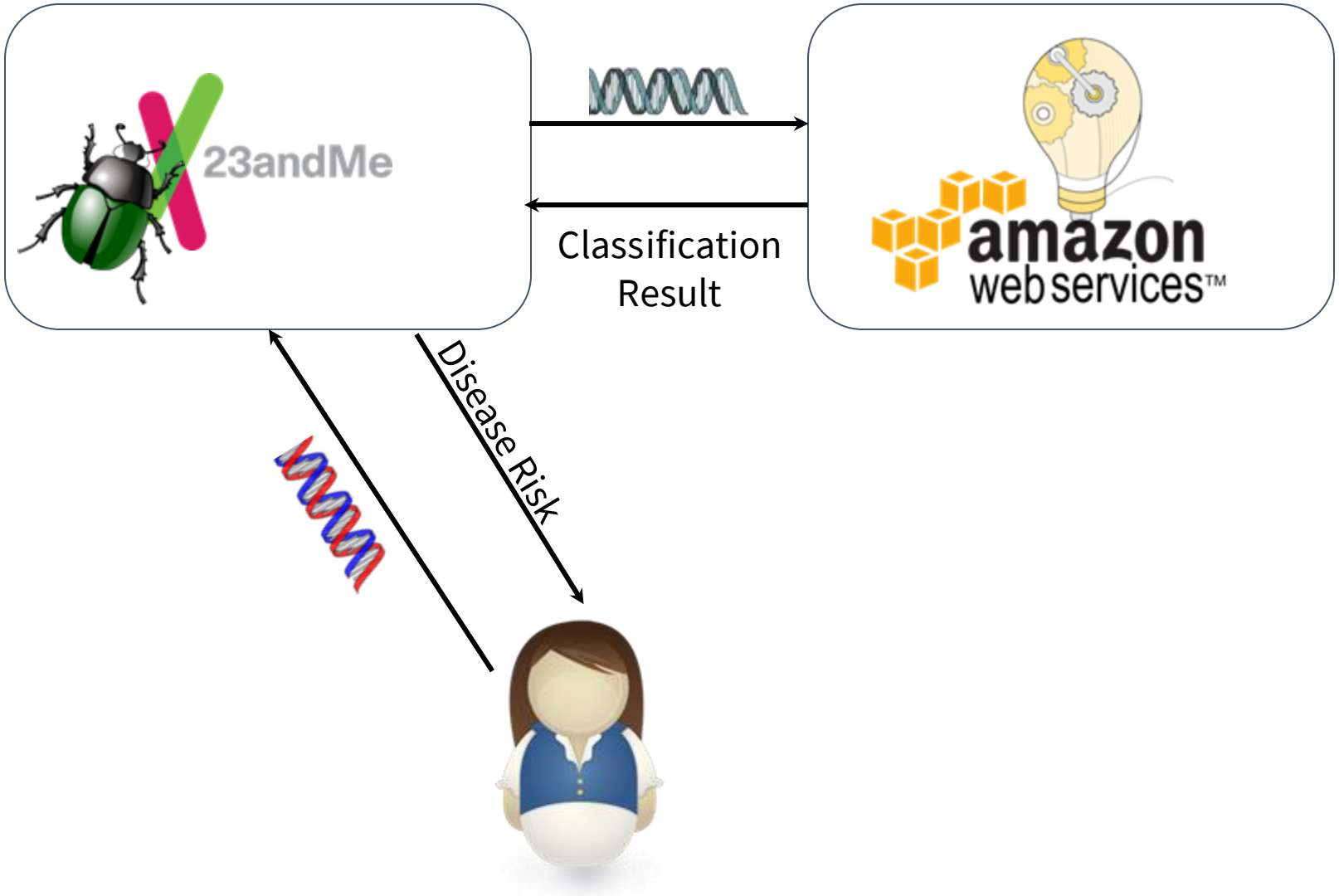
Disease risk assessment: Trust issues



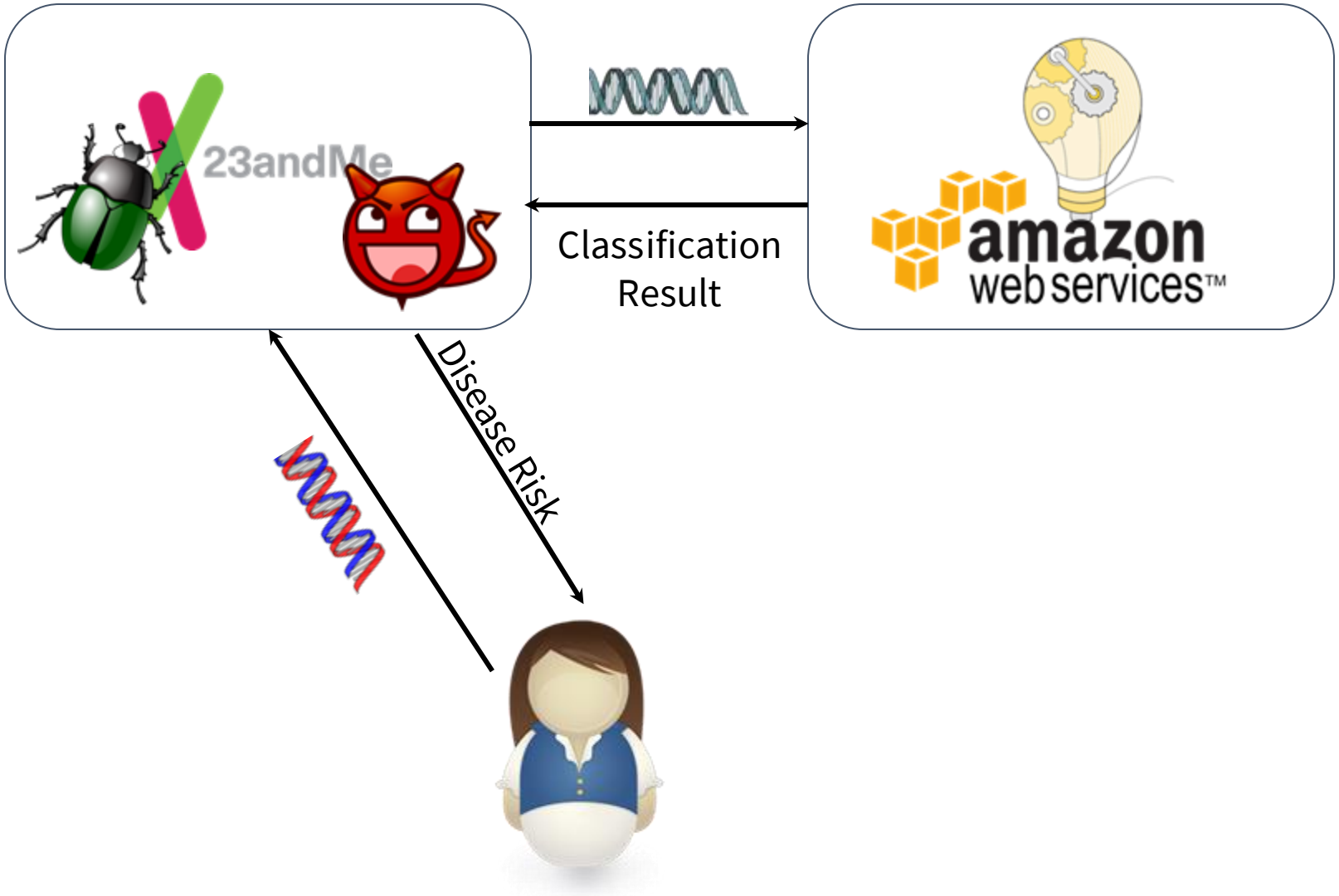
Disease risk assessment: Trust issues



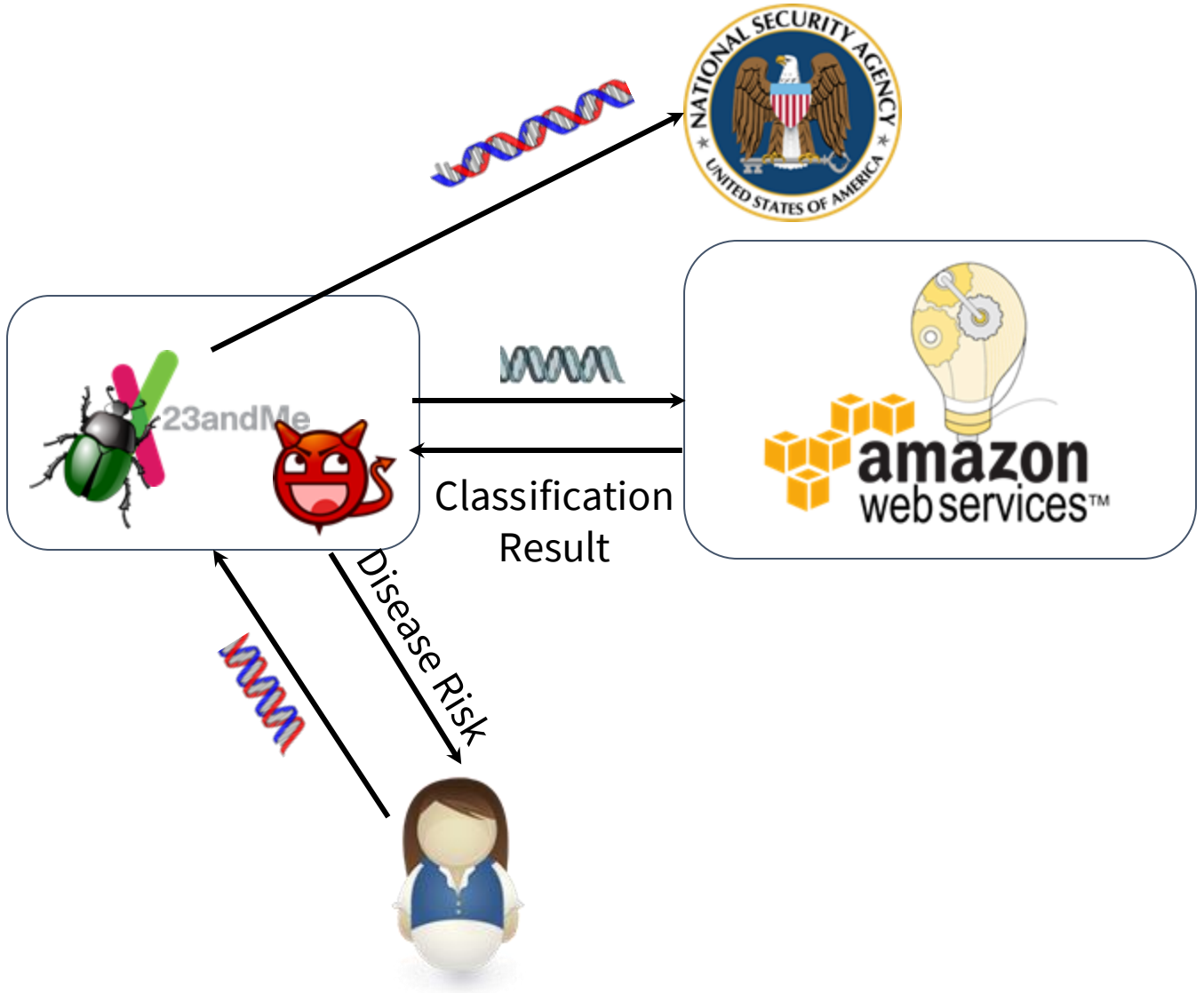
Disease risk assessment: Trust issues



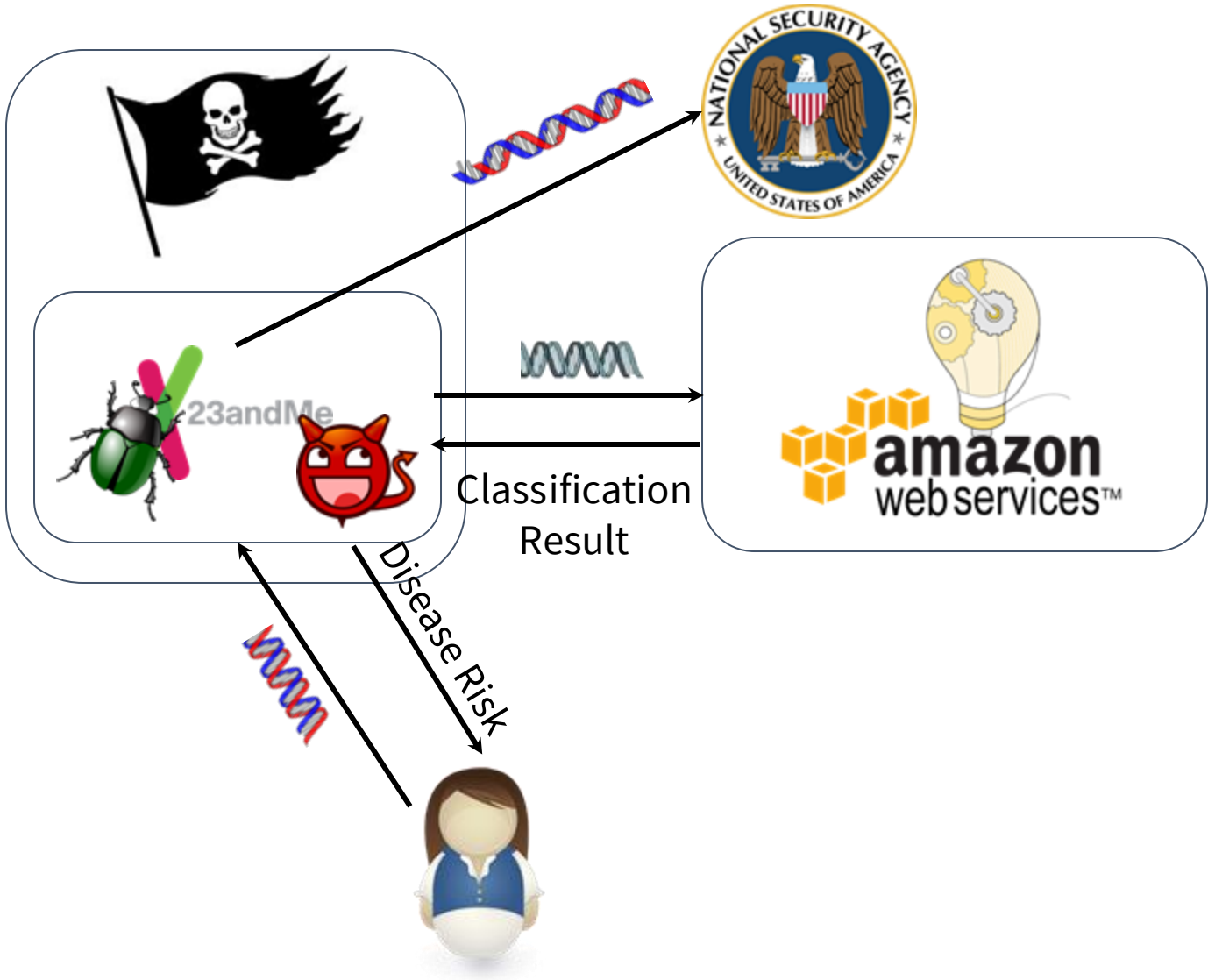
Disease risk assessment: Trust issues



Disease risk assessment: Trust issues



Disease risk assessment: Trust issues



Talk outline

Introduction

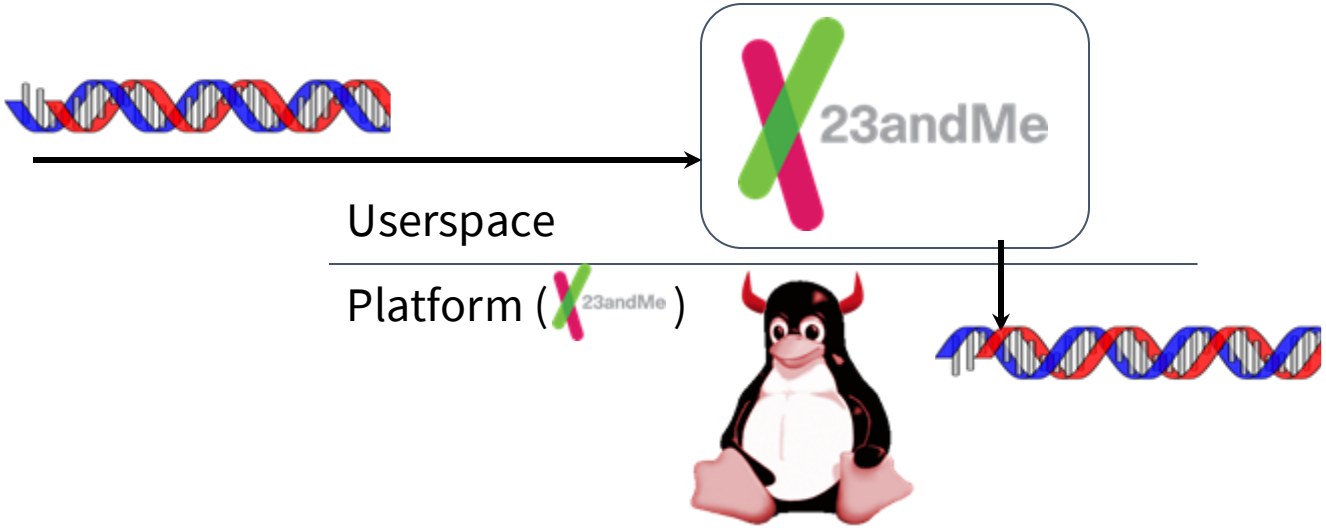
Controlling untrusted modules

Covert and side channels

Evaluation

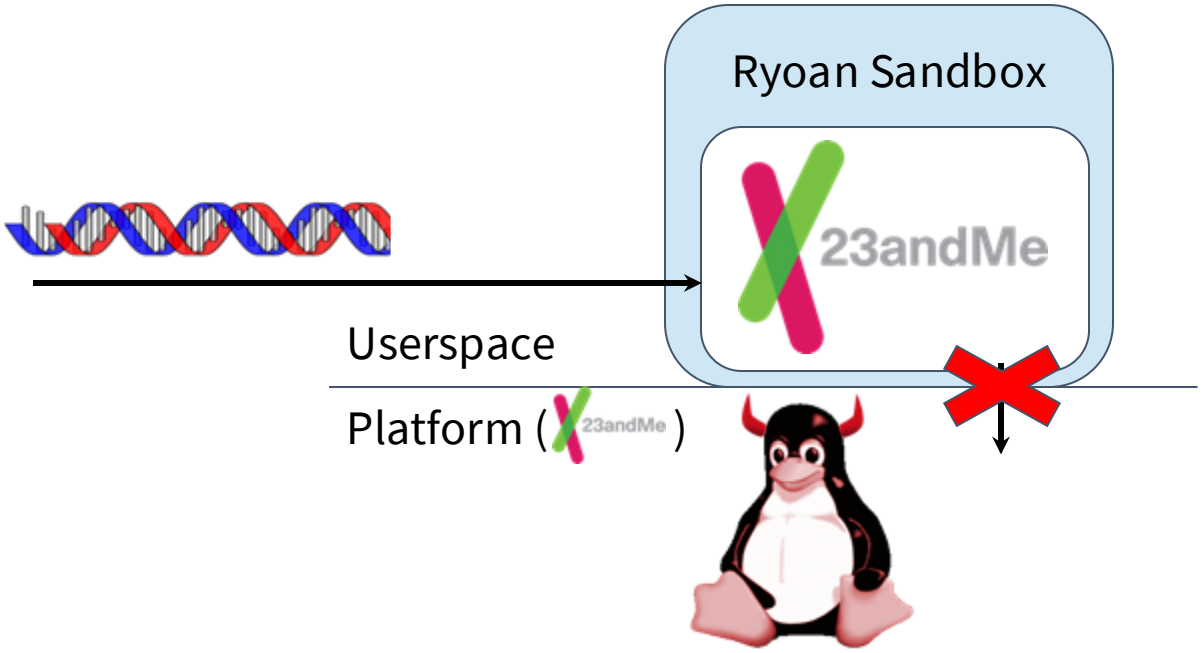
Ryoan's goals

- Provide user data secrecy
  - Without trusting the application
  - Without trusting the platform (OS, Hypervisor)
- Support cooperation between service providers



Ryoan's goals

- Provide user data secrecy
  - Without trusting the application
  - Without trusting the platform (OS, Hypervisor)
- Support cooperation between service providers



Threat model

Users

- Don't trust service providers for secrecy
- Don't trust platforms for secrecy

Service Providers

- Control platforms
- Don't trust other service provides for secrecy

Everyone

- Trusts Ryoan
- Trusts Intel SGX


- User


- User Data


- Untrusted Code


- Untrusted Platform


- Ryoan


- SGX

Threat model

Users

- ◎ Don't trust service providers for secrecy
- ◎ Don't trust platforms for secrecy

Service Providers

- ◎ Control platforms
- ◎ Don't trust other service provides for secrecy

Everyone

- ◎ Trusts Ryoan
- ◎ Trusts Intel SGX

- User

- User Data

- Untrusted Code

- Untrusted Platform

- Ryoan

- SGX

Threat model

Users

- Don't trust service providers for secrecy
- Don't trust platforms for secrecy

Service Providers

- Control platforms
- Don't trust other service provides for secrecy

Everyone

- Trusts Ryoan
- Trusts Intel SGX

 - User

 - User Data

 - Untrusted Code

 - Untrusted Platform

 - Ryoan

 - SGX

Threat model

Users

- Don't trust service providers for secrecy
- Don't trust platforms for secrecy

Service Providers

- Control platforms
- Don't trust other service provides for secrecy

Everyone

- Trusts Ryoan
- Trusts Intel SGX

 - User

 - User Data

 - Untrusted Code

 - Untrusted Platform

 - Ryoan

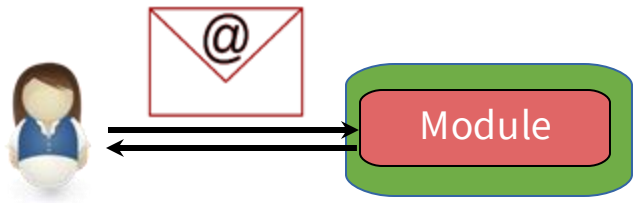
 - SGX

Ryoan uses TEEs for isolation and sandboxing for confinement

- TEE isolation protects secrets from privileged software
  - Cloud provider cannot use control of the machine to read secrets out of memory
- Sandbox confines the application to prevent it from violating isolation
  - Application does not have to be open source to be confined

Ryoan restricts programming model to make confinement easier

- Confinement in general is hard [Lampson`73]
- Modules must be request oriented
  - One request → one result
- Modules must have with a well-defined unit of work
  - e.g, An email, or A photo
- These restrictions allow Ryoan to support applications with a simple read-once, write-once IO pattern



Ryoan's world

Modules

- NaCl x86 binaries from service providers
- Application logic

Module

Platforms

- More service providers' code
- Host computation

Sandboxes

- Trusted code
- Confine modules
- Based on Google's Native Client (NaCl)

Ryoan's world

Modules

- NaCl x86 binaries from service providers
- Application logic

Module

Platforms

- More service providers' code
- Host computation

Sandboxes

- Trusted code
- Confine modules
- Based on Google's Native Client (NaCl)

Ryoan's world

Modules

- NaCl x86 binaries from service providers
- Application logic

Platforms

- More service providers' code
- Host computation

Sandboxes

- Trusted code
- Confine modules
- Based on Google's Native Client (NaCl)

Module

Ryoan's world

Modules

- NaCl x86 binaries from service providers
- Application logic

Module

Platforms

- More service providers' code
- Host computation

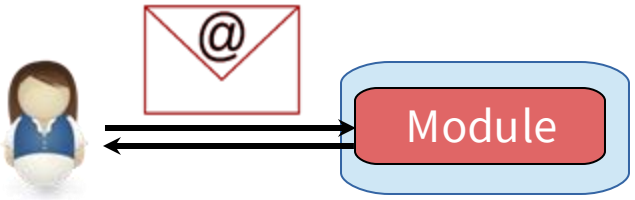
Sandboxes

- Trusted code
- Confine modules
- Based on Google's Native Client (NaCl)

Ryoan applications

Modules

- Request oriented
- Well defined unit of work
  - One request→one result
  - e.g, 1 email, 1 photo



Composable

- Modules can be connected to build services

Talk outline

Introduction

Controlling untrusted modules

Covert and side channels

Evaluation

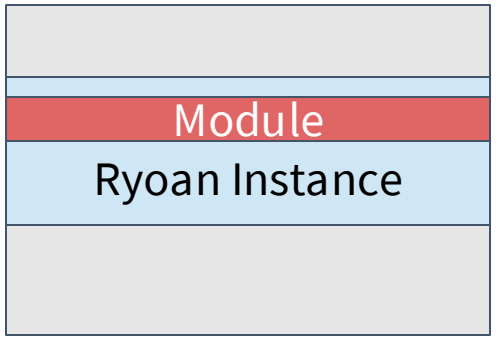
Intel SGX in 2 minutes (or less)

◎ Provides Enclaves
- Regions of a process's virtual address space

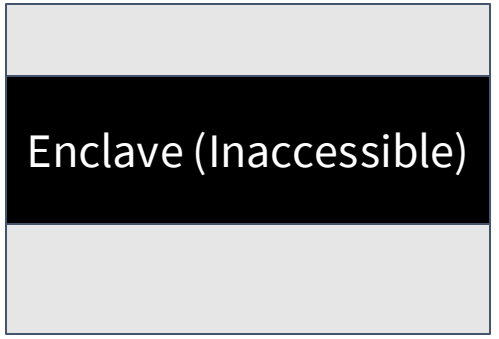
◎ Enclaves
- Can only be accessed by enclave code
- Still have access to the rest of memory

◎ Attestations
- Hardware signed hashes of initial code and data

Enclave Code's View

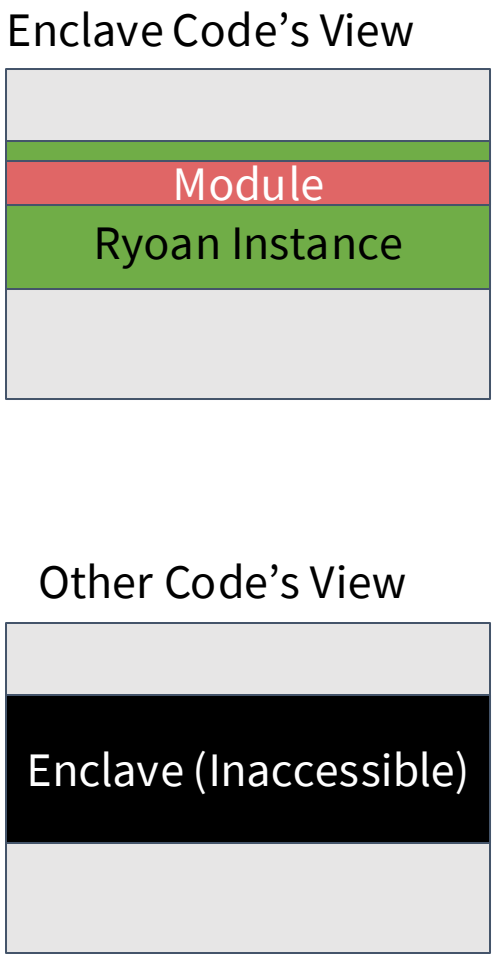


Other Code's View



TEE of choice: Intel SGX

- TEEs provided by SGX are called Enclaves
  - Regions of a process's virtual address space
- Enclaves
  - Can only be accessed by enclave code
  - Still have access to the rest of memory
- Attestations
  - Hardware signed hashes of initial code and data



Chain of trust

- SGX provides unforgeable attestation of the sandbox

- Statements Ryoan makes about the module can now be trusted

Ryoan's view of SGX

- SGX gives you:
  - ***Trusted*** computation on secret data

- Ryoan uses SGX to give you:
  - *Guarantees on **Untrusted*** computation

Confining untrusted code

Problem:

- Platform can read secrets out of memory

Confining untrusted code

Problem:

- Platform can read secrets out of memory

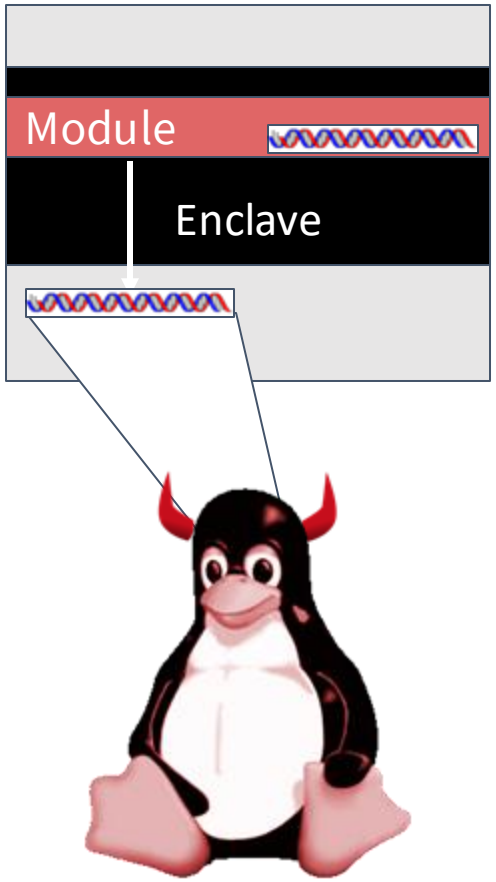
Solution:

- Execute module inside of an enclave

Confining untrusted code

Problem:

- Module can copy secrets to non-enclave memory



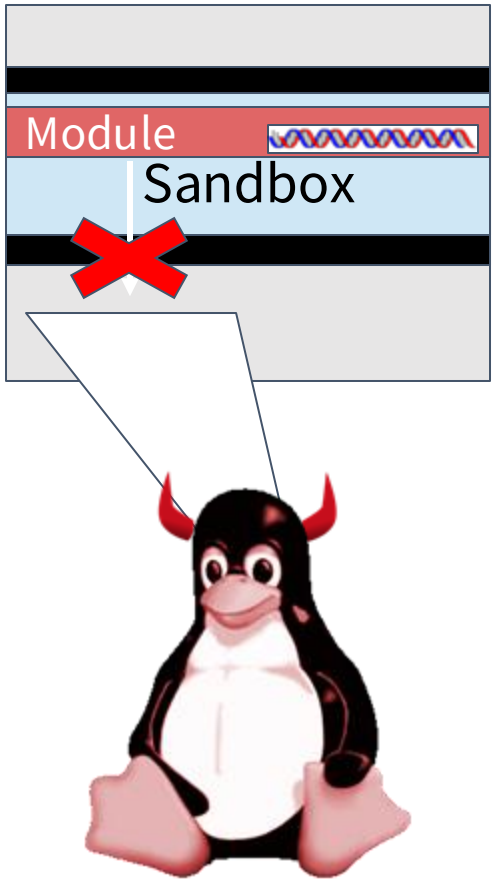
Confining untrusted code

Problem:

- Module can copy secrets to non-enclave memory

Solution:

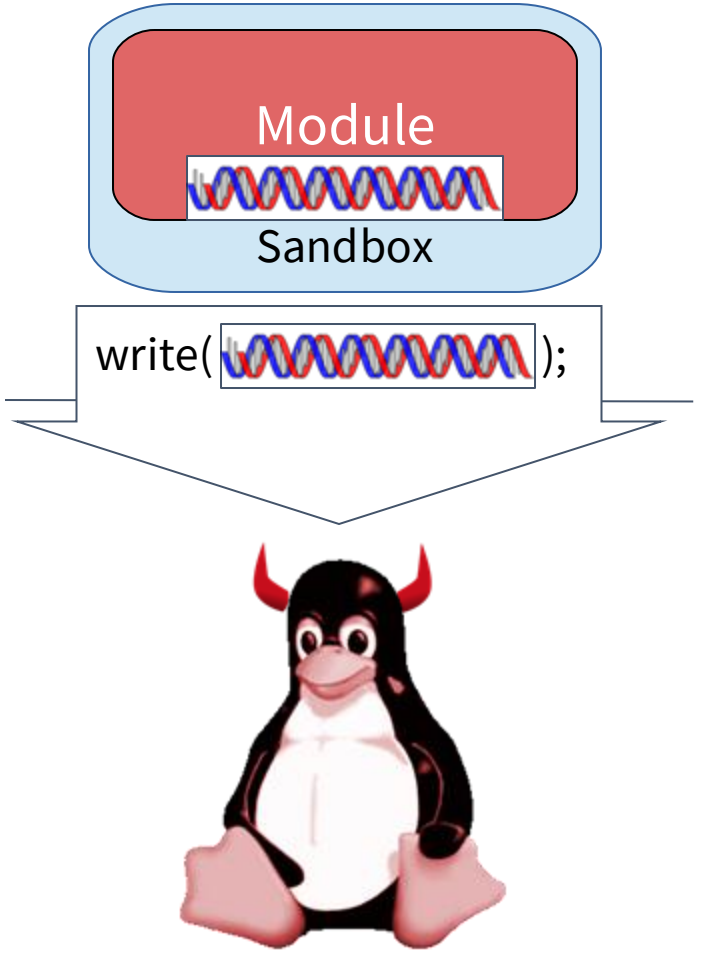
- Restrict accessible memory with a sandbox
  - Property of NaCl



Confining untrusted code

Problem:

◎ Modules can use system calls to write out user data



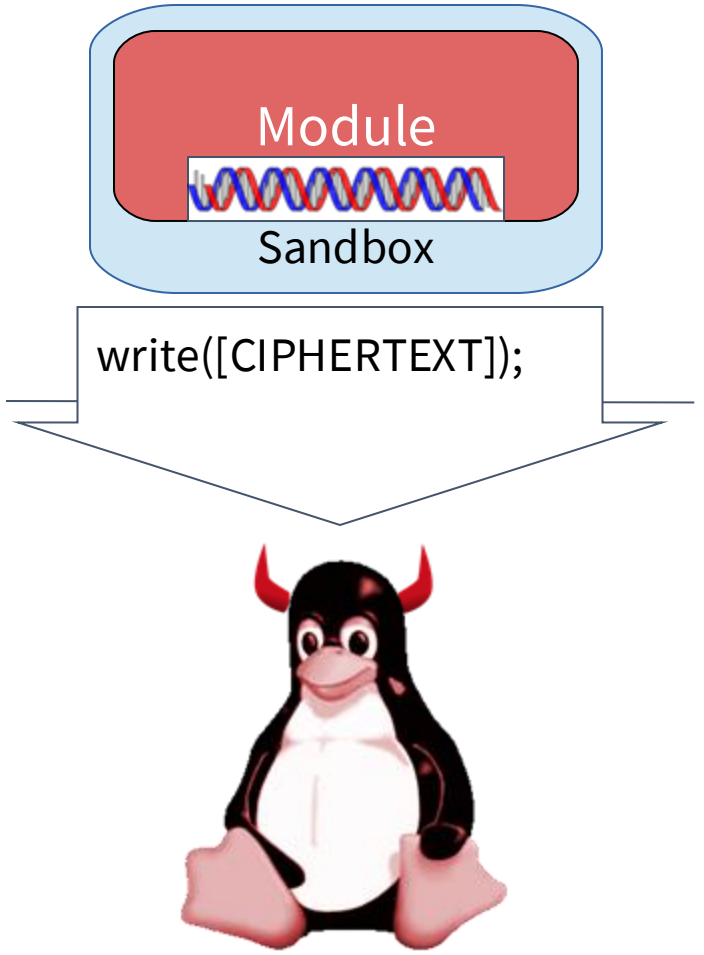
Confining untrusted code

Problem:

- Modules can use system calls to write out user data

Solution:

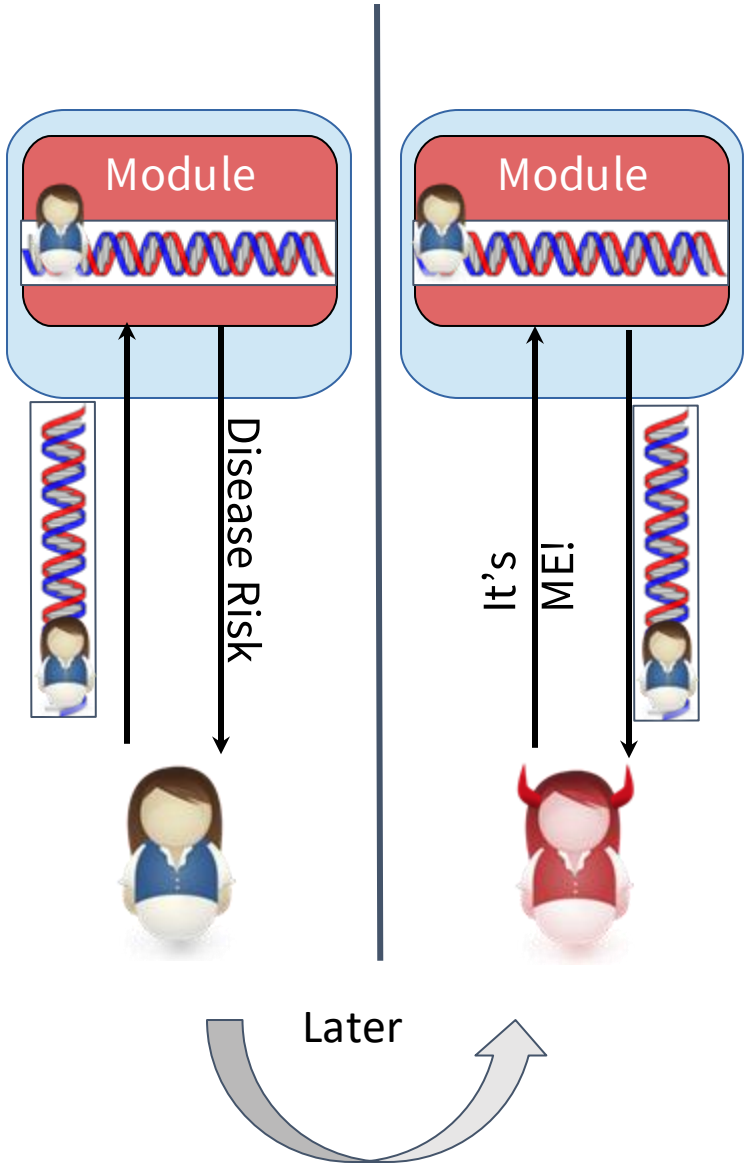
- NaCl modules call sandbox to access system calls
- Enforce encryption



Confining untrusted code

Problem:

- Modules can collude with users to steal data



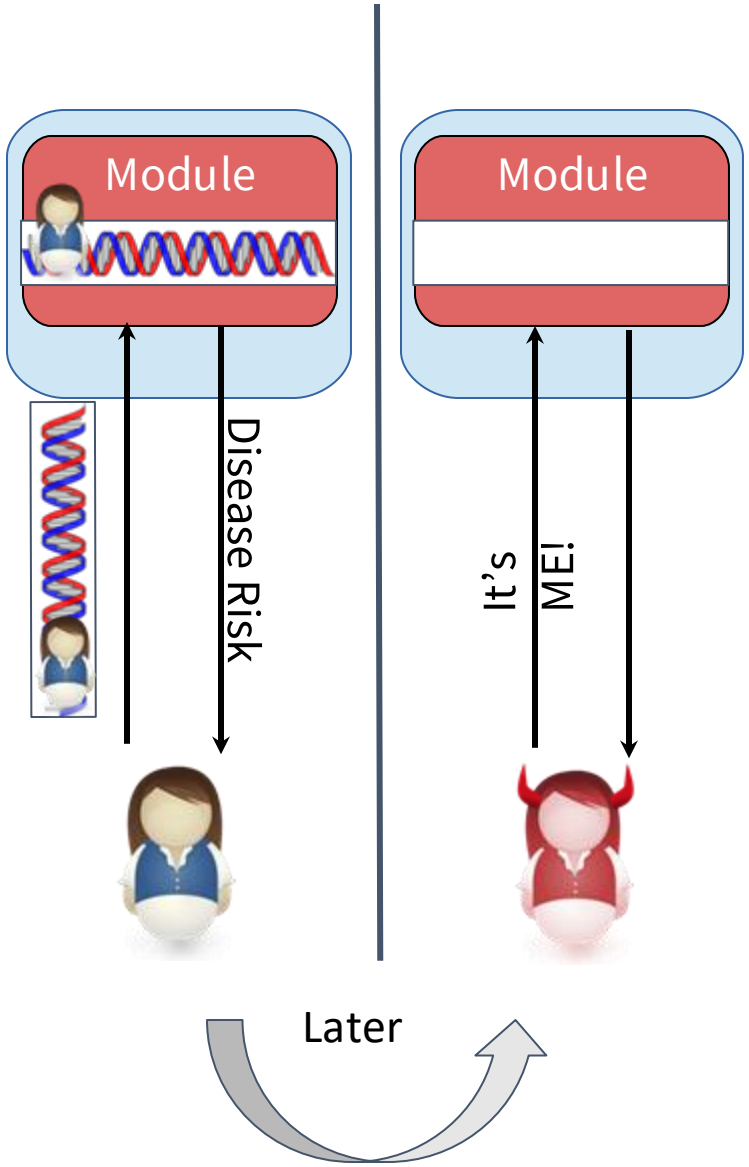
Confining untrusted code

Problem:

- Modules can collude with users to steal data

Solution:

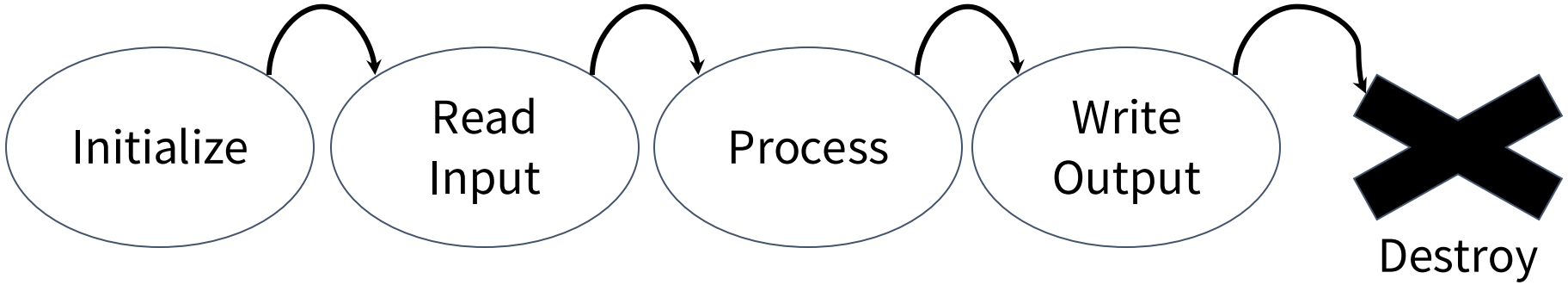
- Don’t let modules keep state between requests



Modules cannot keep state

- Module life cycle imposed by Ryoan
  - Read, process, write, destroy

- Sandbox enforces one request per module execution
  - Represent a complete unit of work
  - Only contain content from one user



Talk outline

Introduction

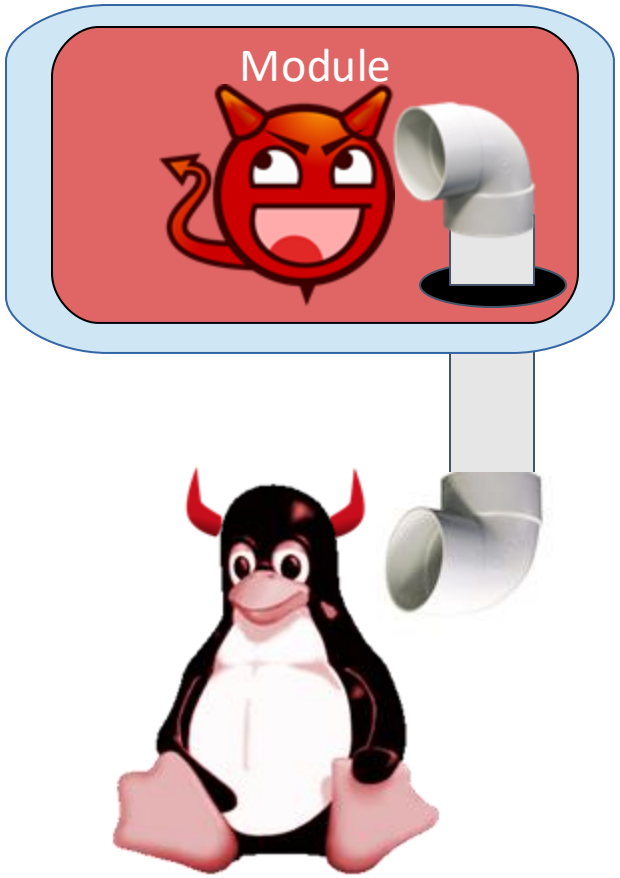
Controlling untrusted modules

Covert and side channels

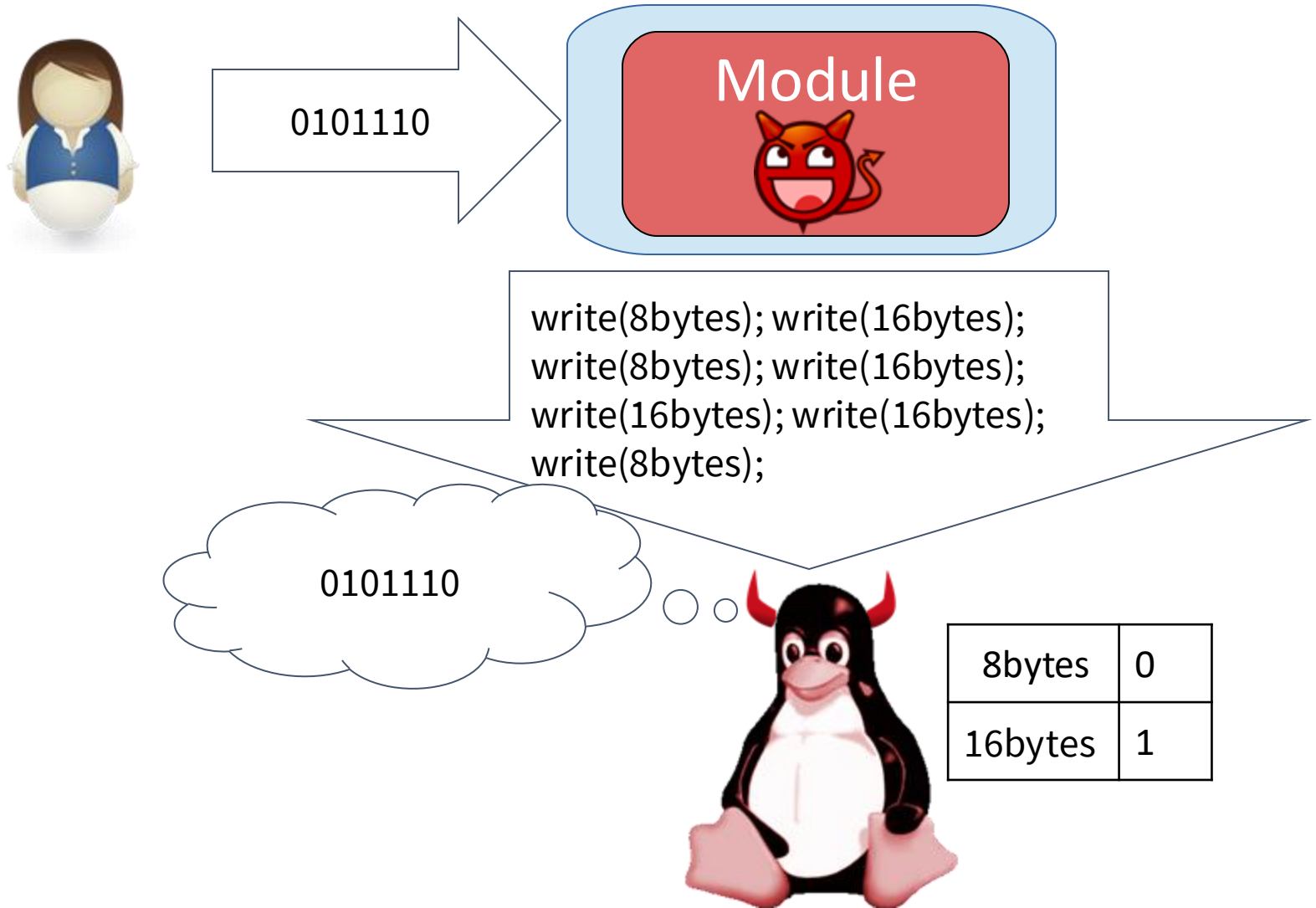
Evaluation

Covert and side channels

- Output, via some externally visible property of execution
- Ryoan: Software covert channels
  - System calls
  - Execution time
- Hardware covert channels:
  - Hardware vendor's responsibility

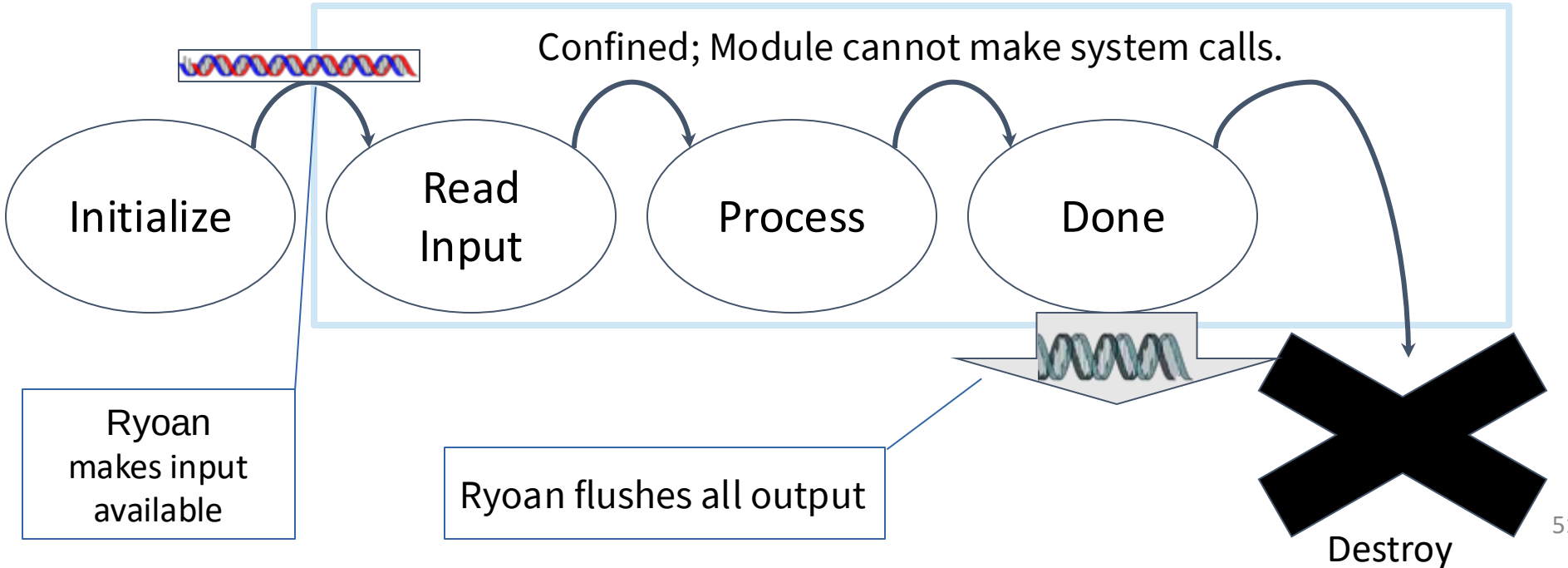


System call covert channel



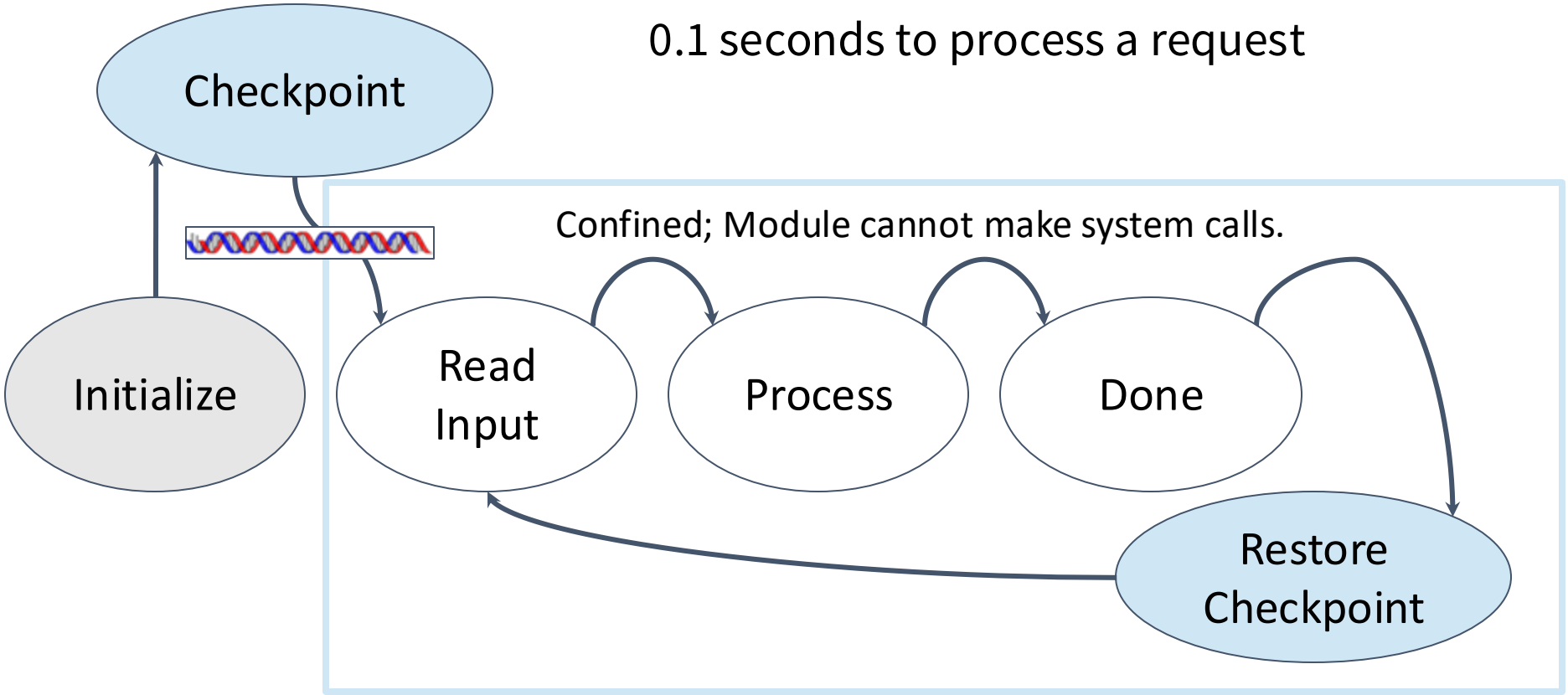
Eliminating system call channel

- Remove modules ability to make system calls
- Ryoan performs all data input and output independent of the content



Initialization is expensive

ClamAV (virus scanner):
25.0 seconds to initialize
0.1 seconds to process a request



Confined compatibility API

Dynamic Memory

- Modules can call mmap for “new” memory
- Return memory from a pre-allocated pool.

Replaced system calls:
mmap

In-memory file API

- File system operations in memory
- Examples:
  - Temp files
  - Preexisting files

Replaced system calls:
open, close, read, write, stat, lseek, unlink, mkdir, rmdir, getdents

Confined compatibility API

Dynamic Memory

- Modules can call mmap for “new” memory
- Return memory from a pre-allocated pool.

Replaced system calls:
mmap

In-memory file API

- File system operations in memory
- Examples:
  - Temp files
  - Preexisting files

Replaced system calls:
open, close, read, write, stat, lseek, unlink, mkdir, rmdir, getdents

Confined compatibility API

Dynamic Memory

- Modules can call mmap for “new” memory
- Return memory from a pre-allocated pool.

Replaced system calls:
mmap

In-memory file API

- File system operations in memory
- Examples:
  - Temp files
  - Preexisting files

Replaced system calls:
open, close, read, write, stat, lseek, unlink, mkdir, rmdir, getdents

Talk outline

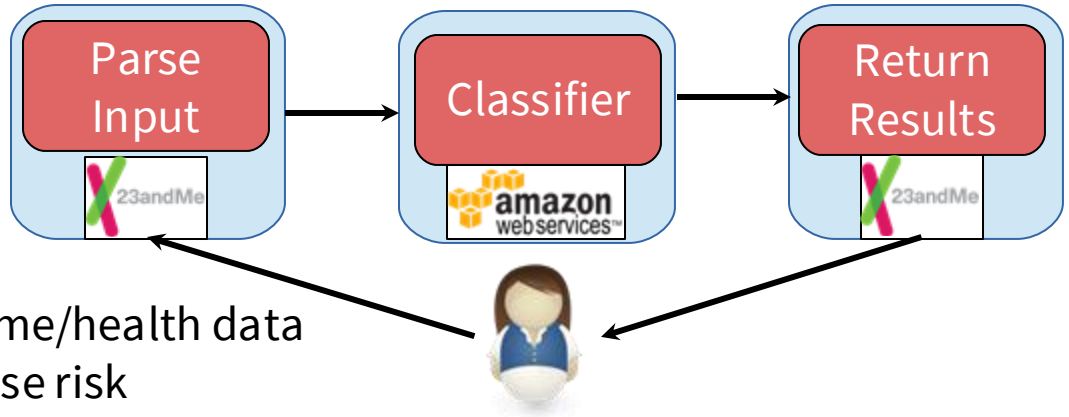
Introduction

Controlling untrusted modules

Covert channels

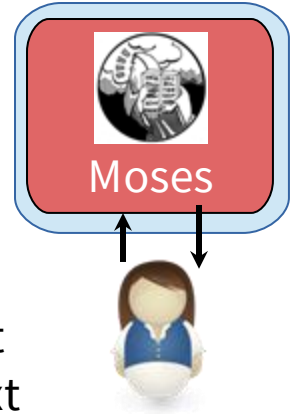
Evaluation

Health



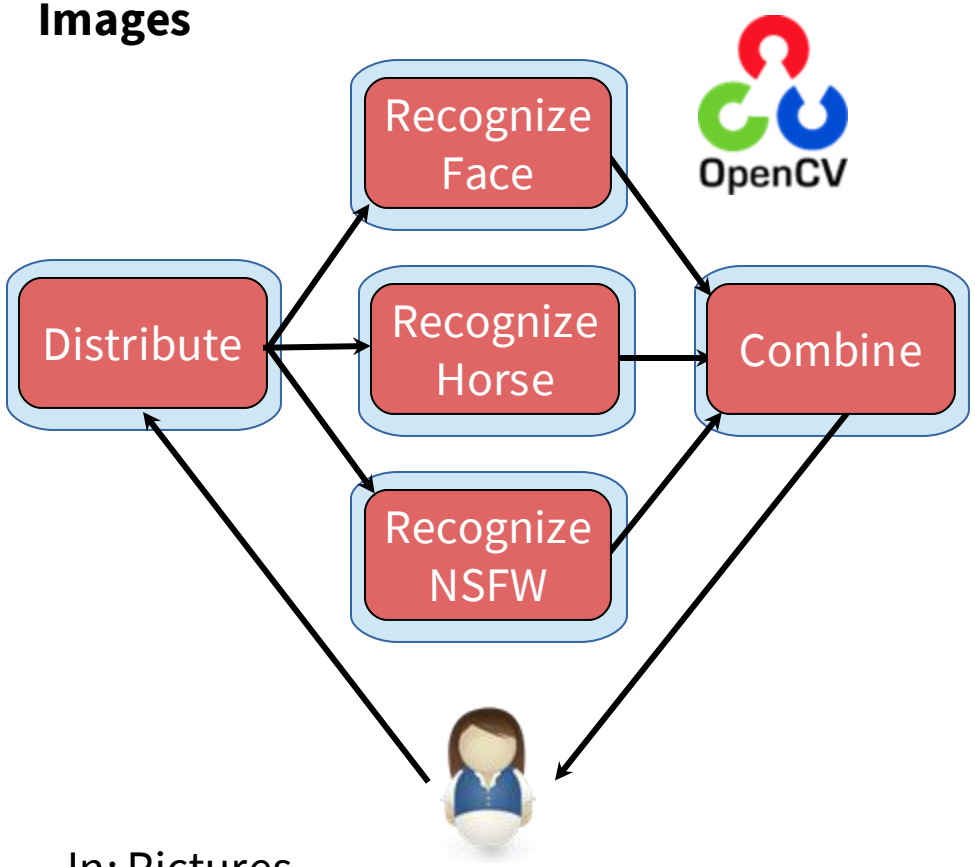
In: Genome/health data
Out: Disease risk

Translation



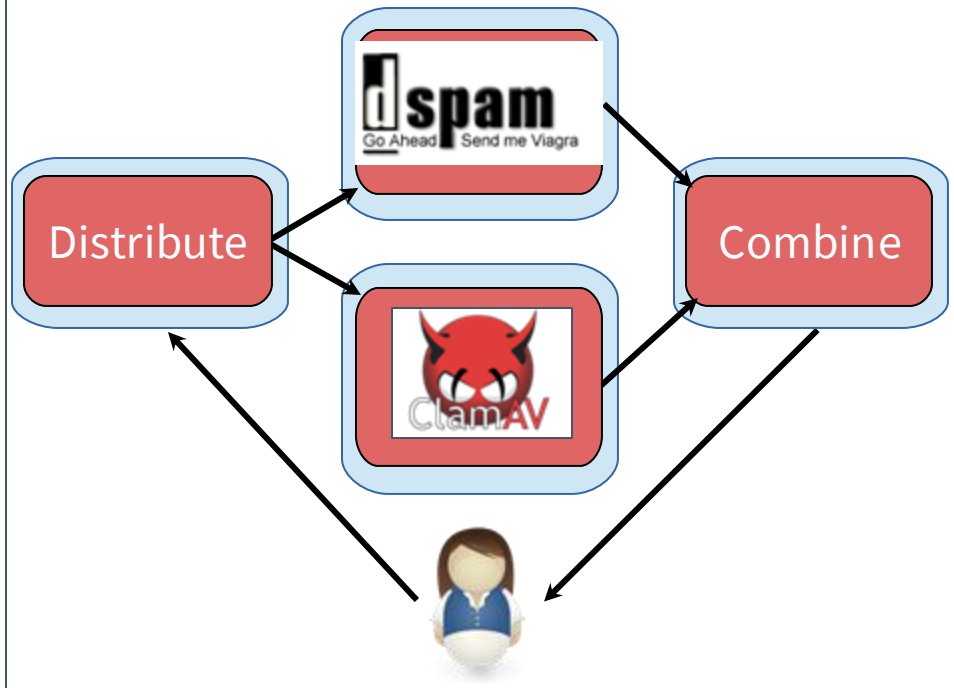
In: French text
Out: English text

Images



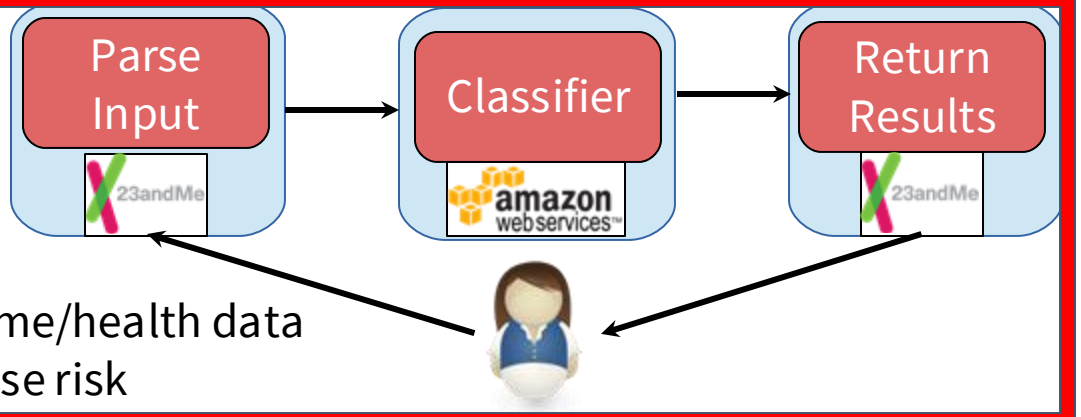
In: Pictures
Out: Array of objects

Email



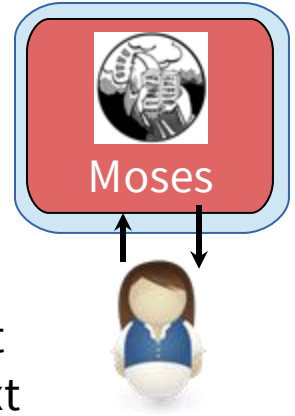
In: Emails
Out: Spam & virus status

Health



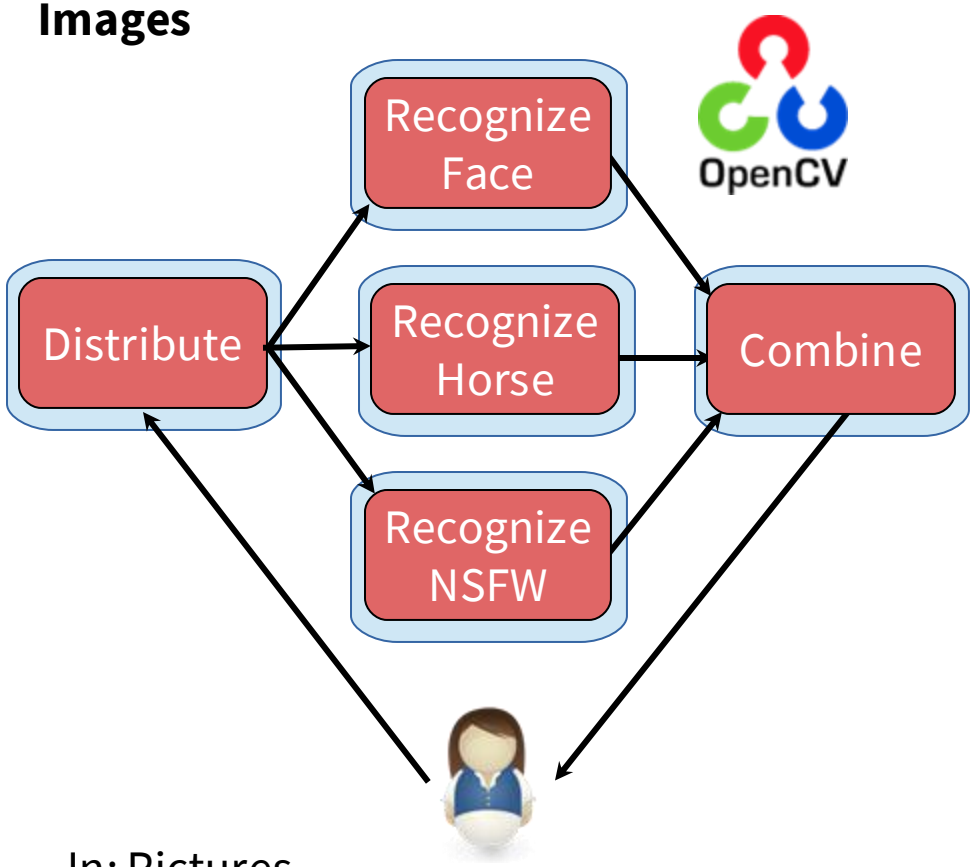
In: Genome/health data
Out: Disease risk

Translation



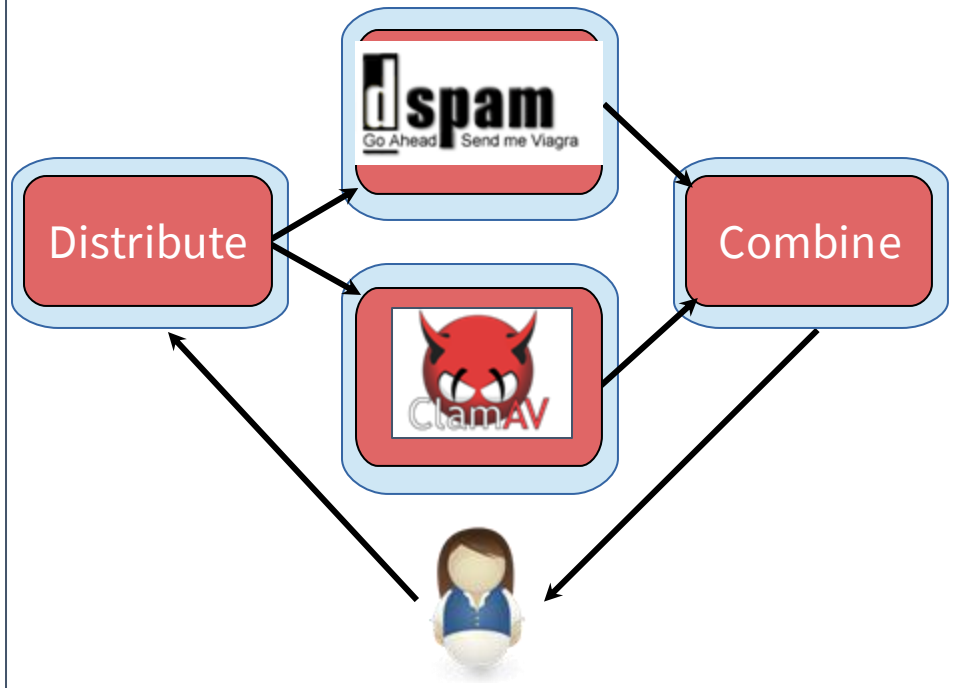
In: French text
Out: English text

Images



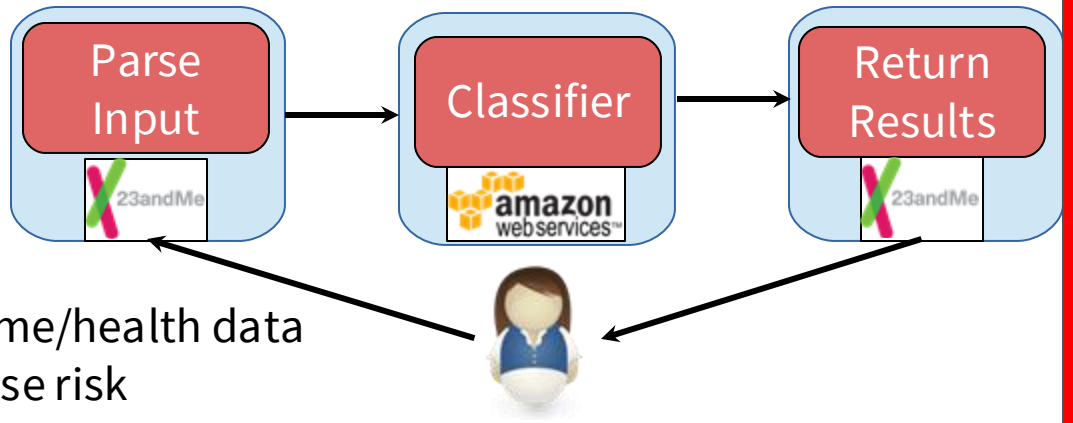
In: Pictures
Out: Array of objects

Email



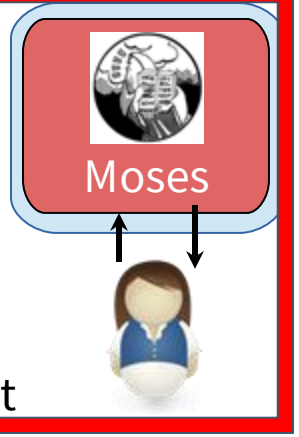
In: Emails
Out: Spam & virus status

Health



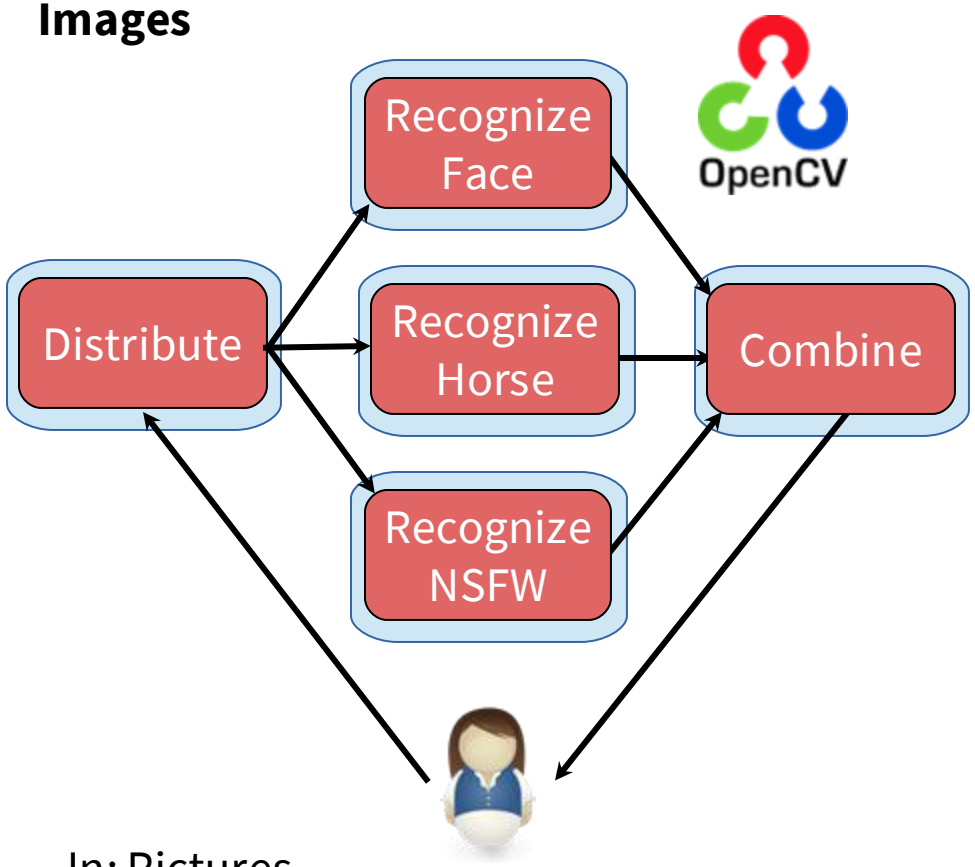
In: Genome/health data
Out: Disease risk

Translation



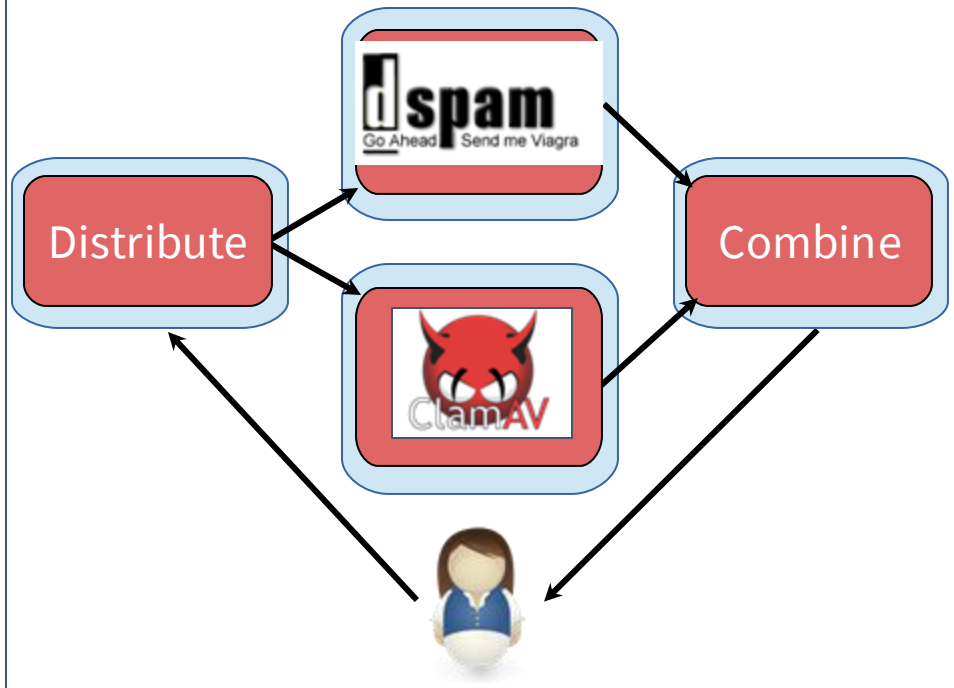
In: French text
Out: English text

Images



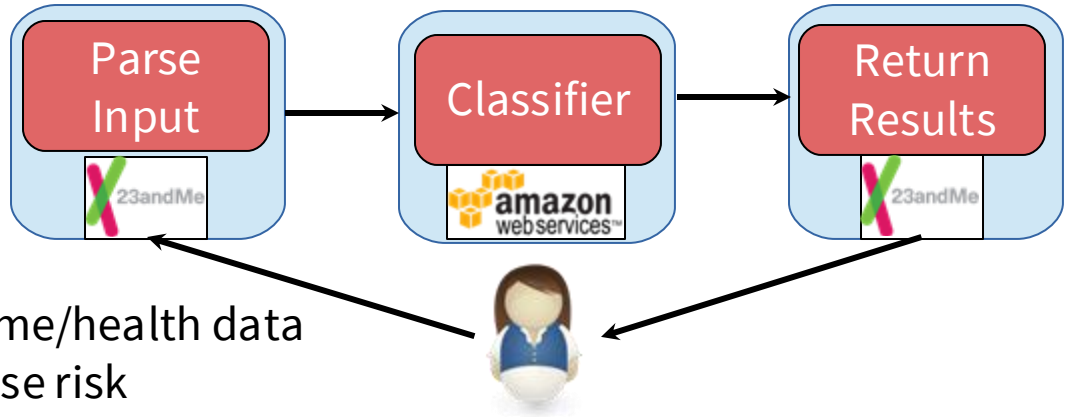
In: Pictures
Out: Array of objects

Email



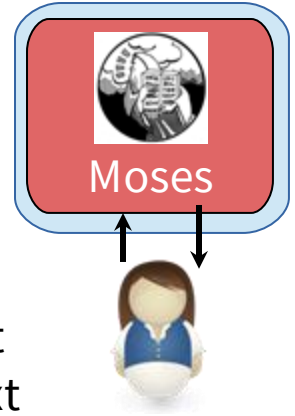
In: Emails
Out: Spam & virus status

Health



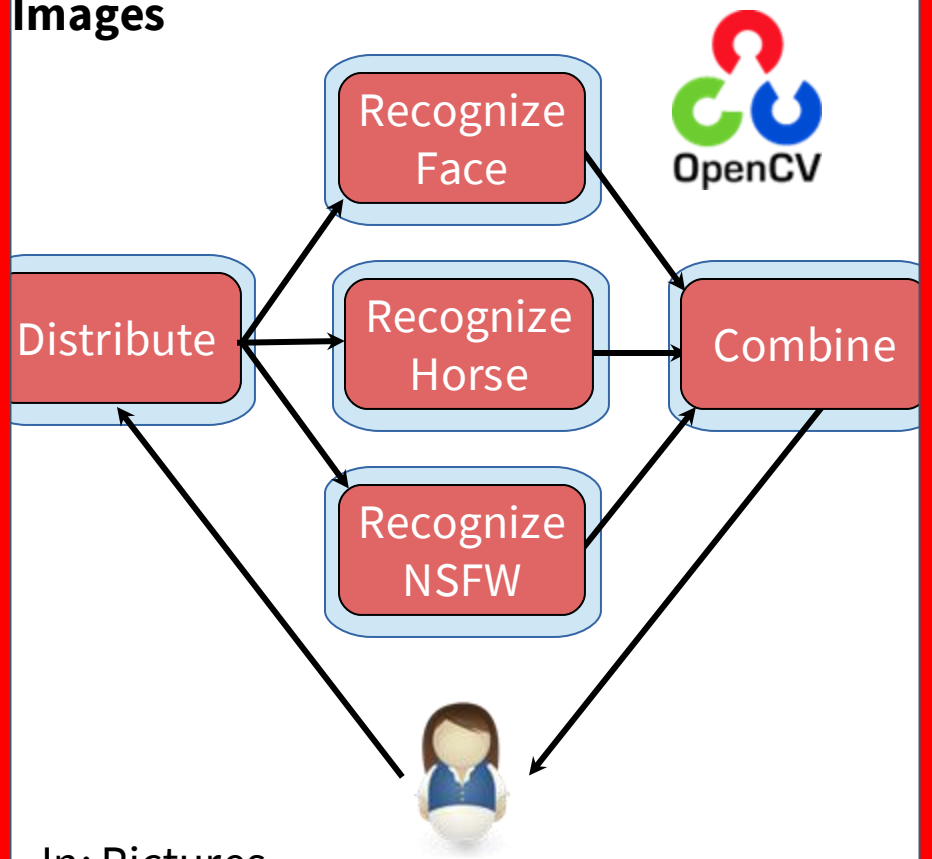
In: Genome/health data
Out: Disease risk

Translation



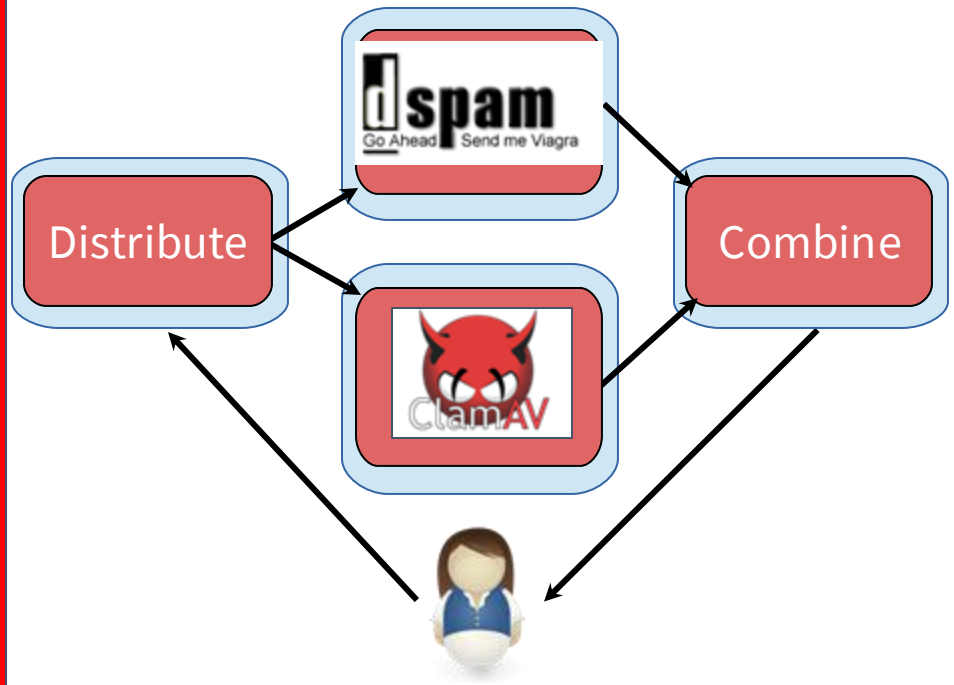
In: French text
Out: English text

Images



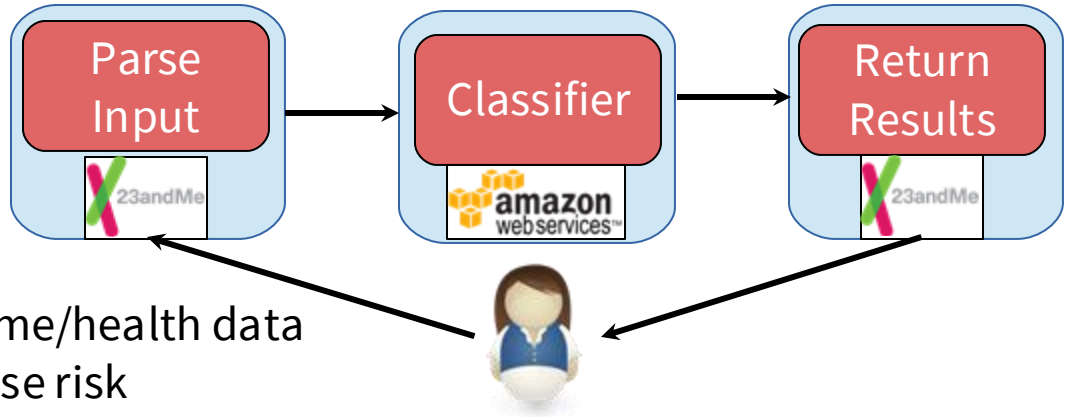
In: Pictures
Out: Array of objects

Email



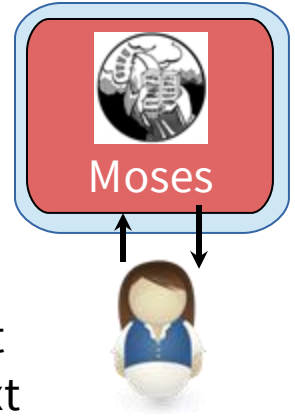
In: Emails
Out: Spam & virus status

Health



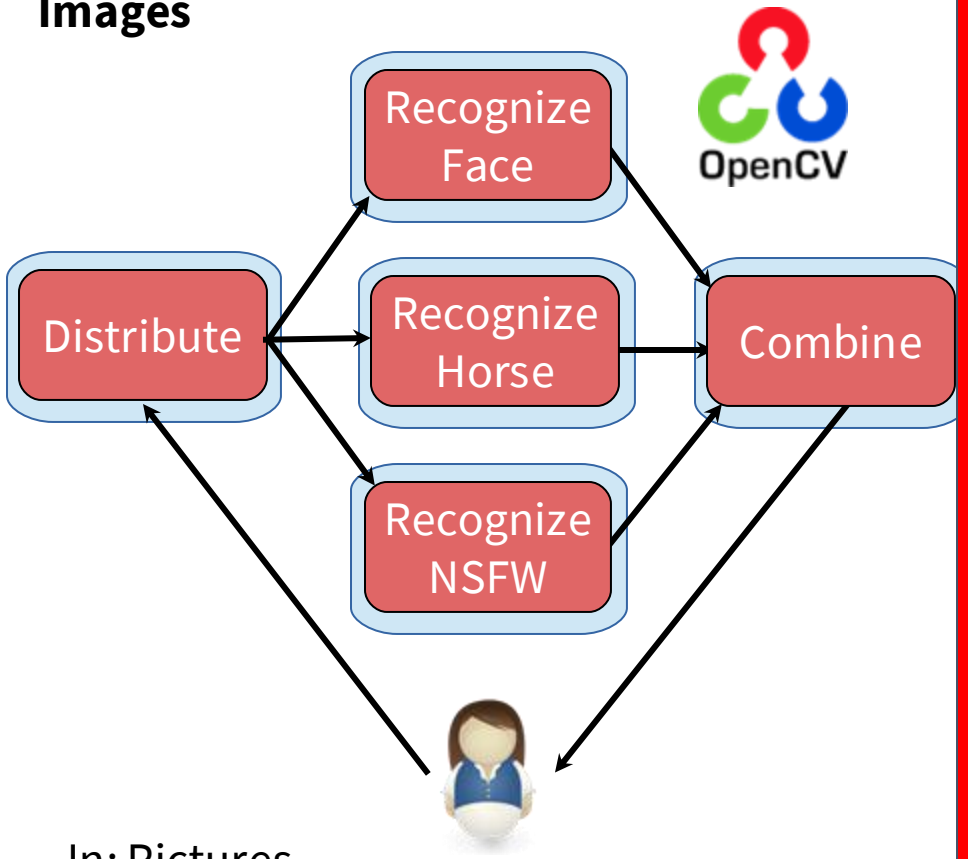
In: Genome/health data
Out: Disease risk

Translation



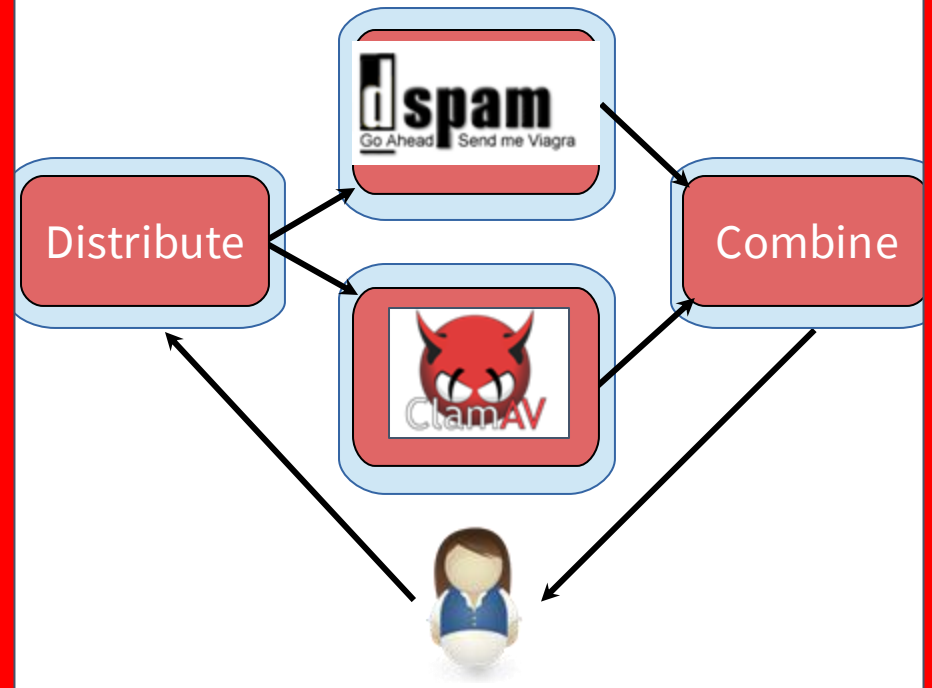
In: French text
Out: English text

Images



In: Pictures
Out: Array of objects

Email

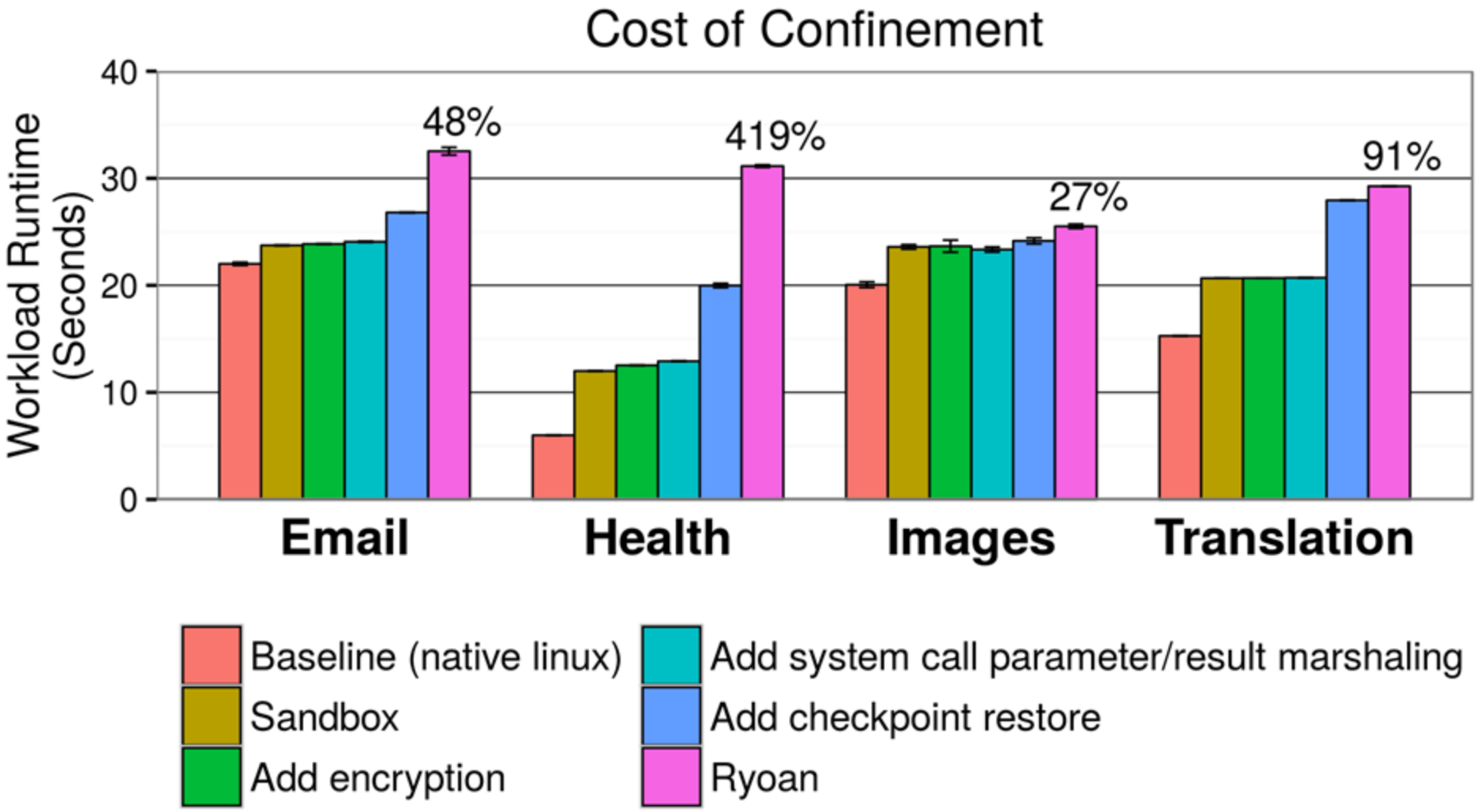


In: Emails
Out: Spam & virus status

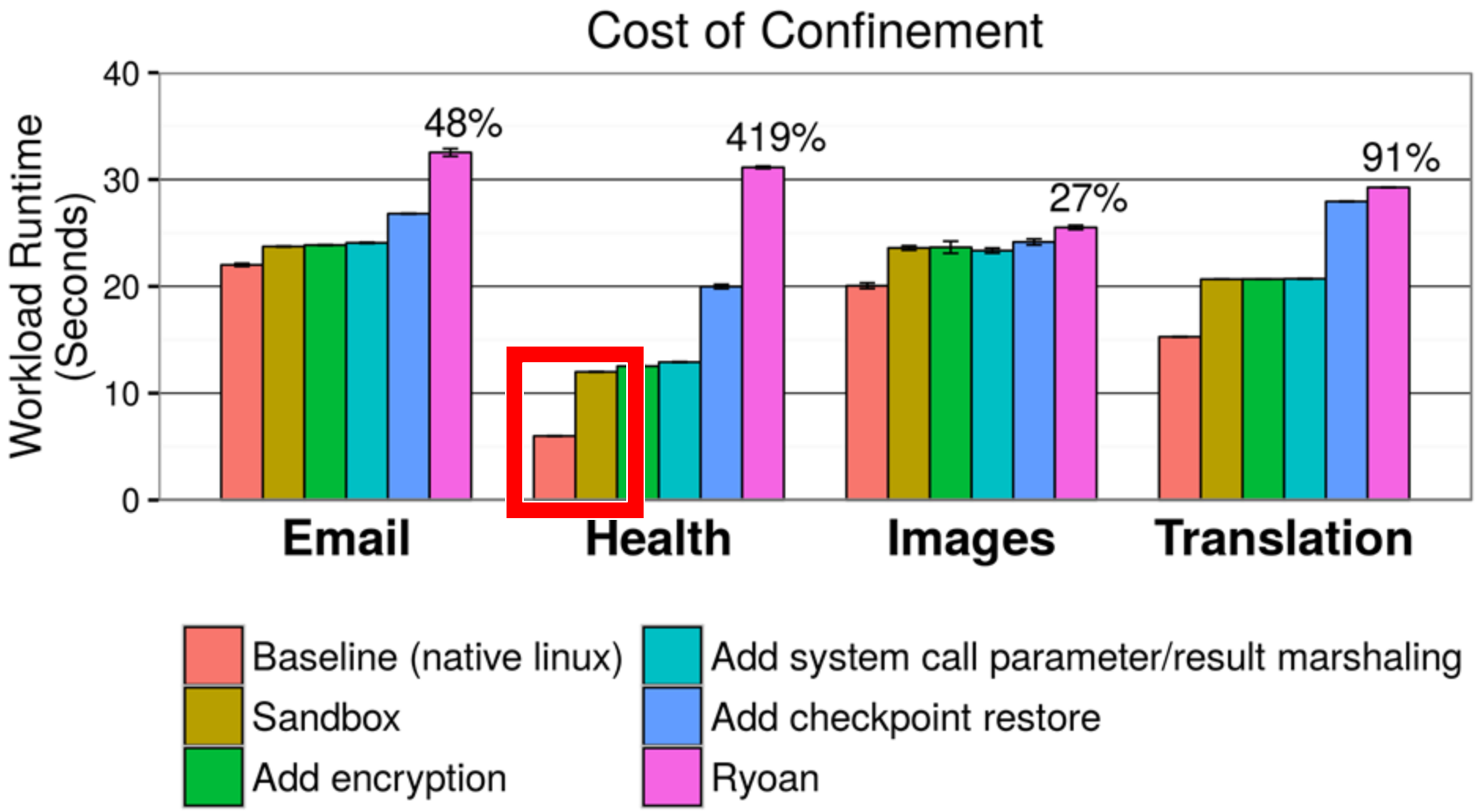
Evaluation

- Implementation requires SGX v2 instructions (spec: Fall 2014, coming soon)
  - Dynamic memory allocation/protection

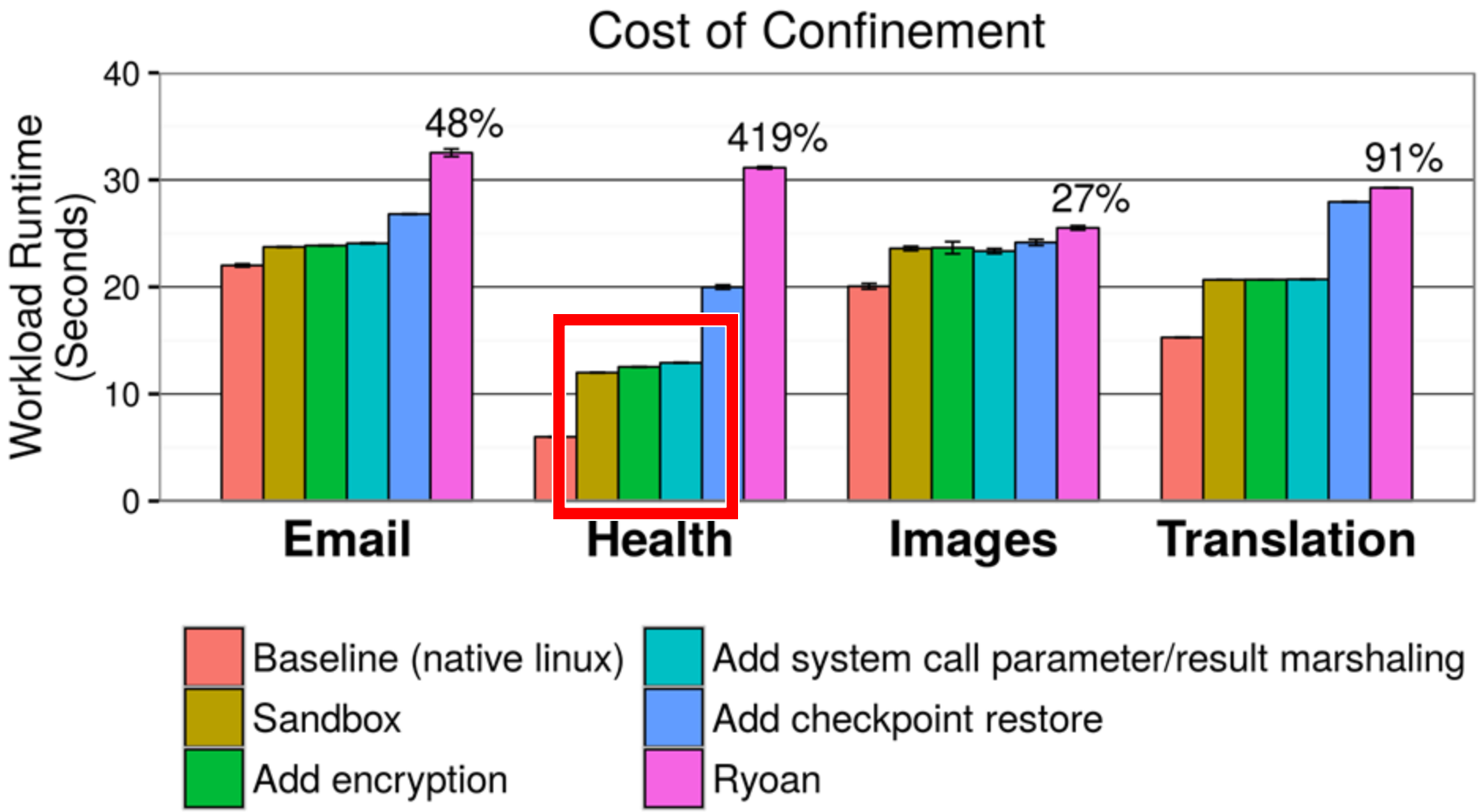
- SGX performance model
  - Measured SGX v1 latencies on our hardware
  - Estimated SGX v2 latencies (sensitivity study in paper)
  - Flush TLB on all system calls, page faults, and interrupts



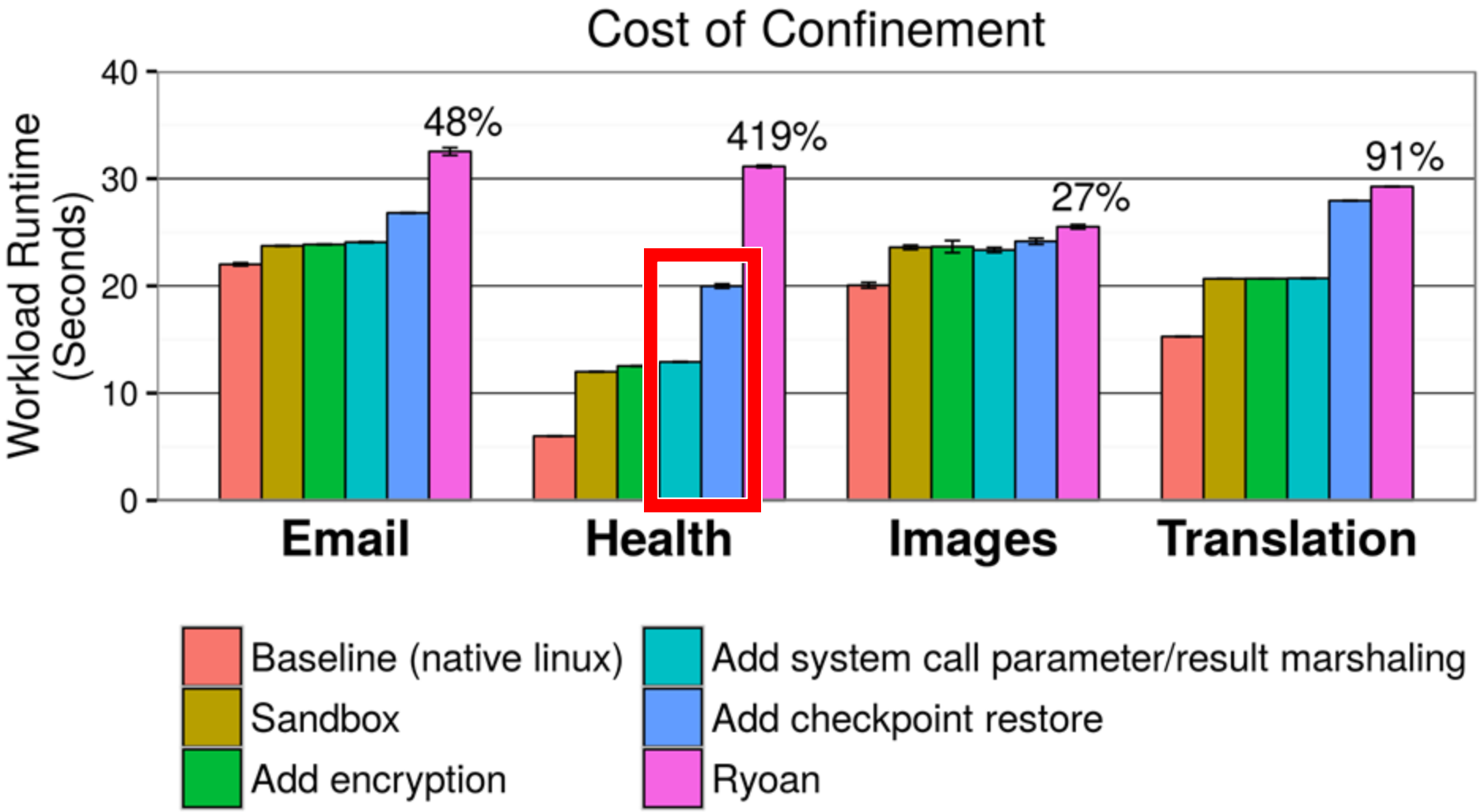
Health	20,000 1.4KB Boolean vectors from different users
Translation	30 short paragraphs, sizes 25-300B, 4.1KB total
Images	12 images, sizes 17KB-613KB
Email	250 emails, 30% with 103KB-12MB attachment



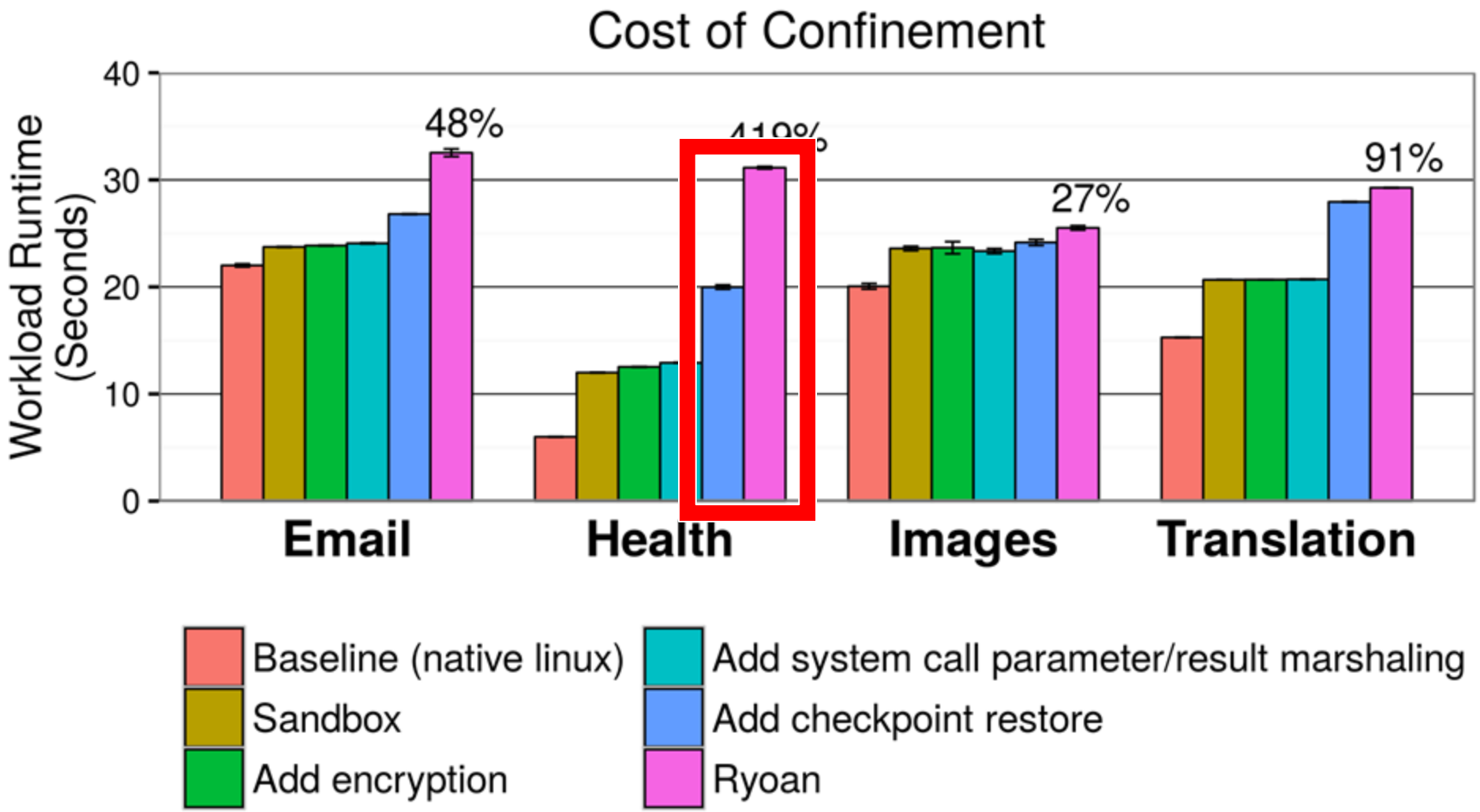
Health	20,000 1.4KB Boolean vectors from different users
Translation	30 short paragraphs, sizes 25-300B, 4.1KB total
Images	12 images, sizes 17KB-613KB
Email	250 emails, 30% with 103KB-12MB attachment



Health	20,000 1.4KB Boolean vectors from different users
Translation	30 short paragraphs, sizes 25-300B, 4.1KB total
Images	12 images, sizes 17KB-613KB
Email	250 emails, 30% with 103KB-12MB attachment



Health	20,000 1.4KB Boolean vectors from different users
Translation	30 short paragraphs, sizes 25-300B, 4.1KB total
Images	12 images, sizes 17KB-613KB
Email	250 emails, 30% with 103KB-12MB attachment



Health	20,000 1.4KB Boolean vectors from different users
Translation	30 short paragraphs, sizes 25-300B, 4.1KB total
Images	12 images, sizes 17KB-613KB
Email	250 emails, 30% with 103KB-12MB attachment

Ryoan summary

- Allows untrusted code to operate on secret data on untrusted platforms

- Sandbox with SGX
  - Eliminates explicit channels
- Module can’t call platform
  - Eliminates covert channels
- Mostly backwards compatible
  - Sandbox code implements system calls