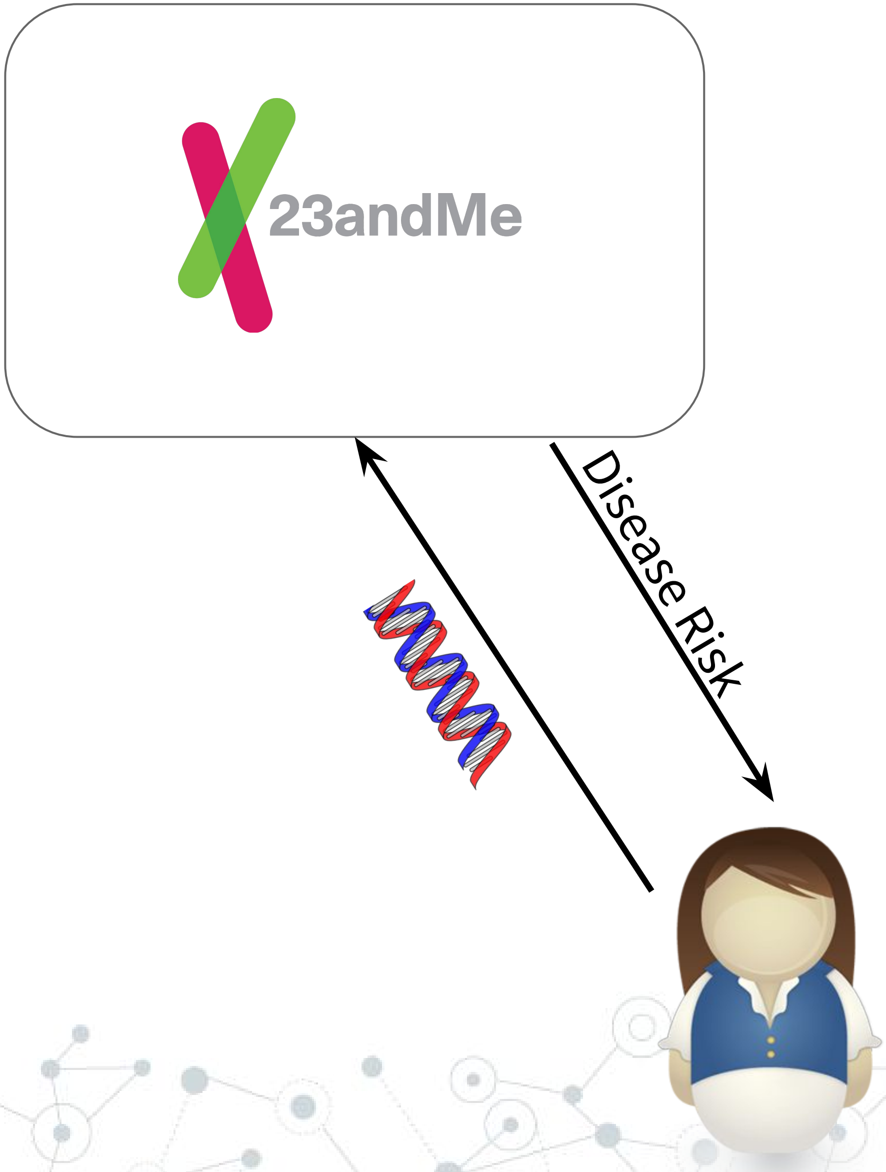
TEXAS
The University of Texas at Austin

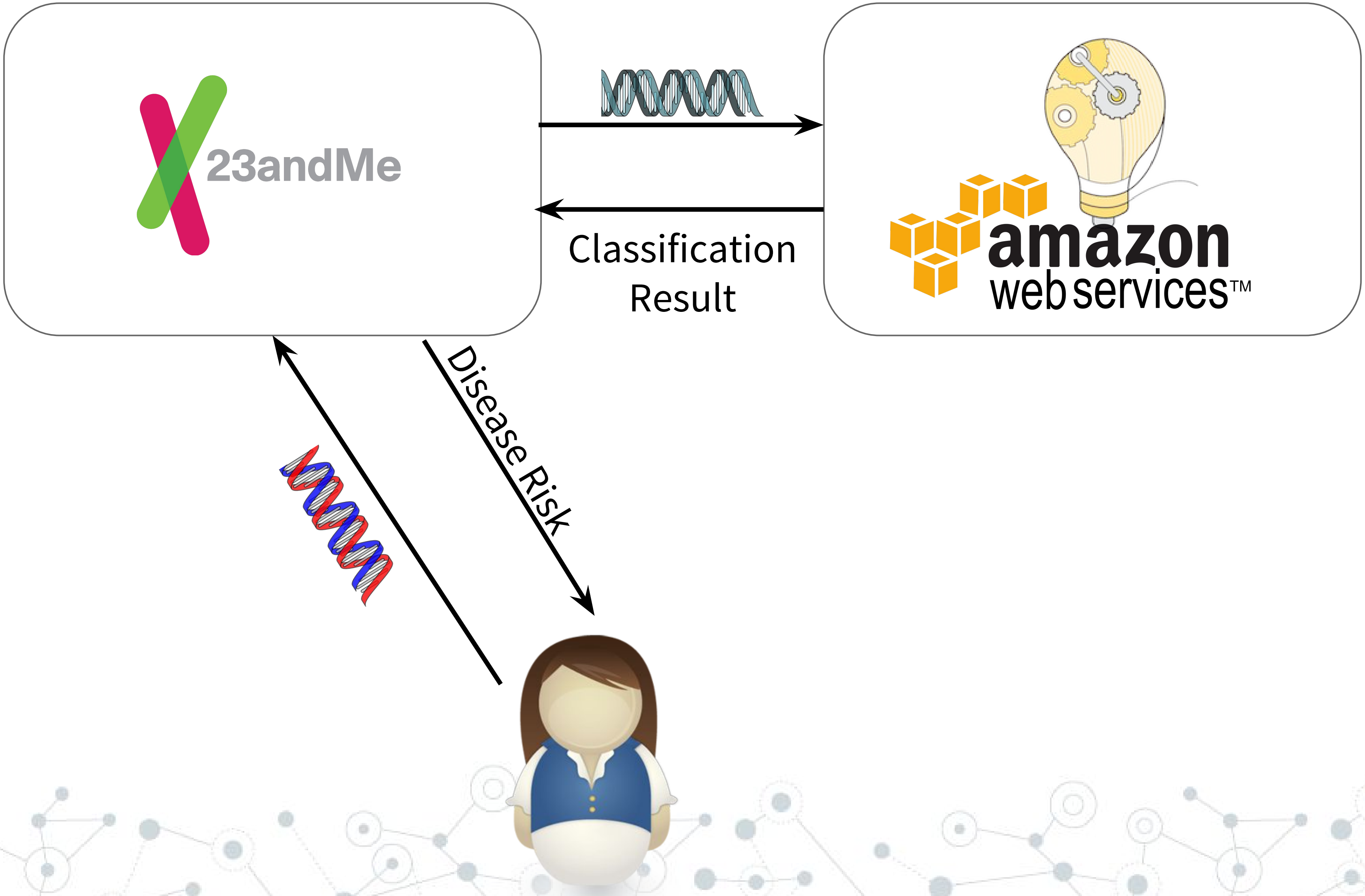
Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data

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

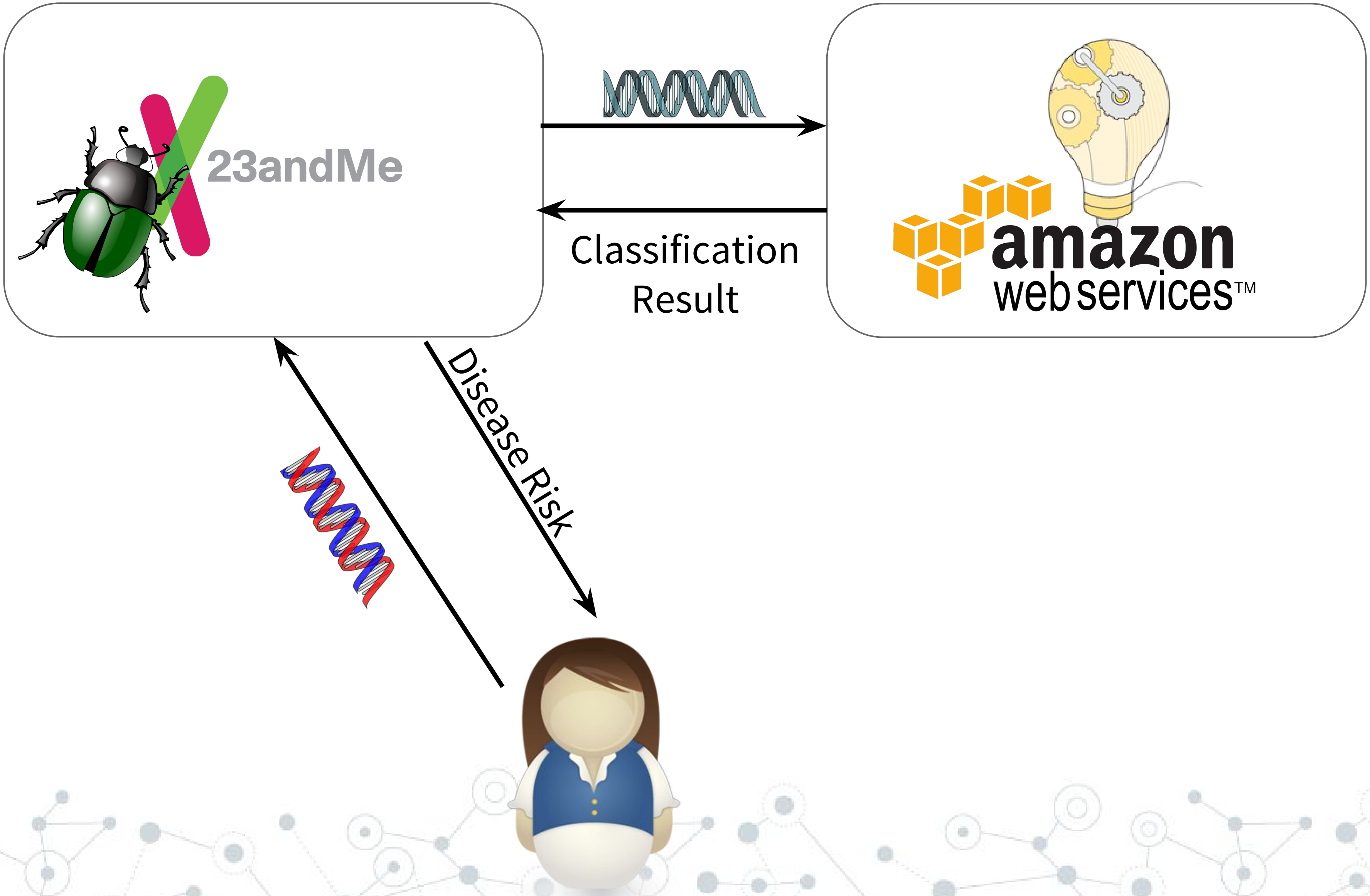
Disease risk assessment: Trust issues



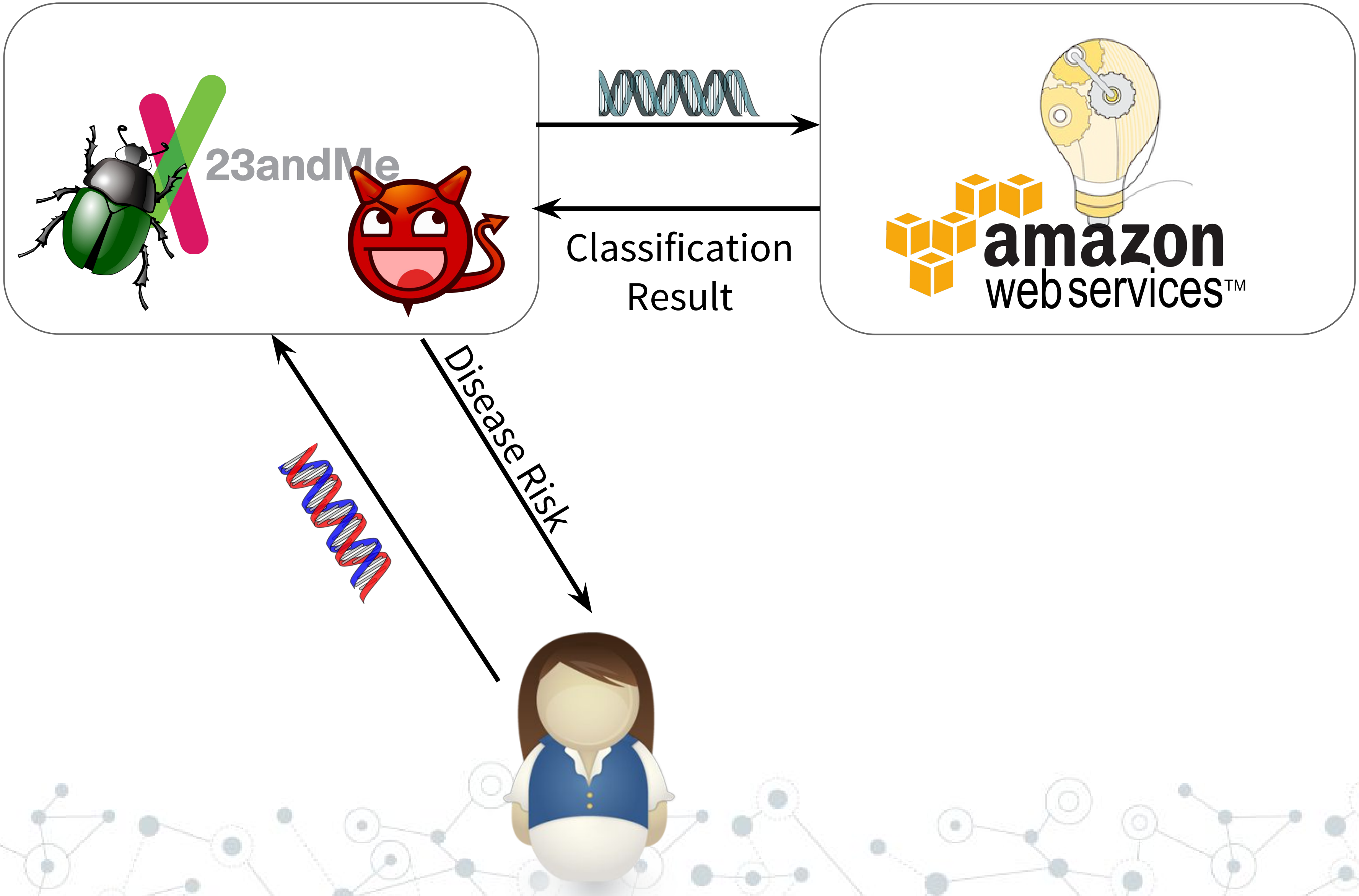
Disease risk assessment: Trust issues



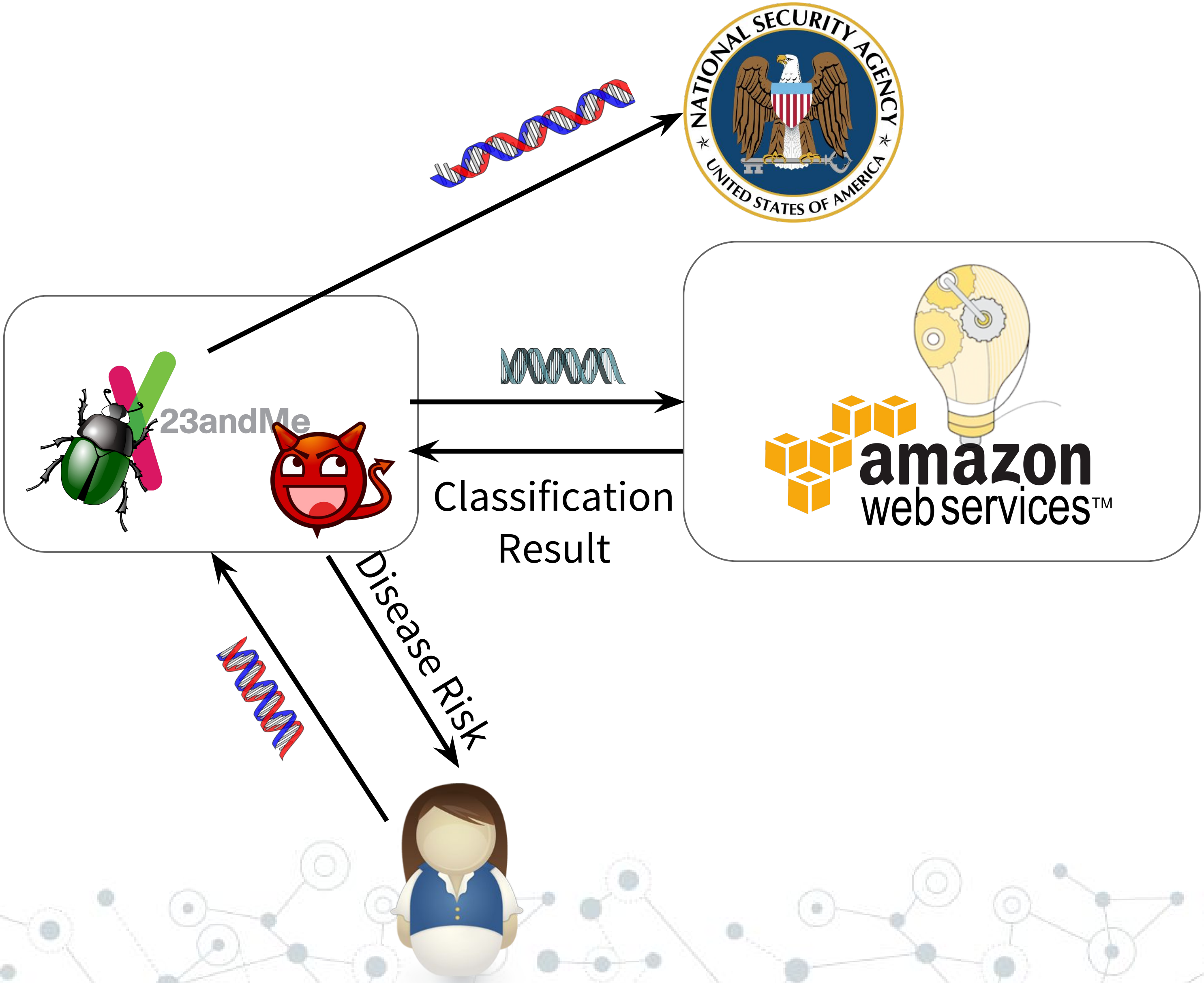
Disease risk assessment: Trust issues



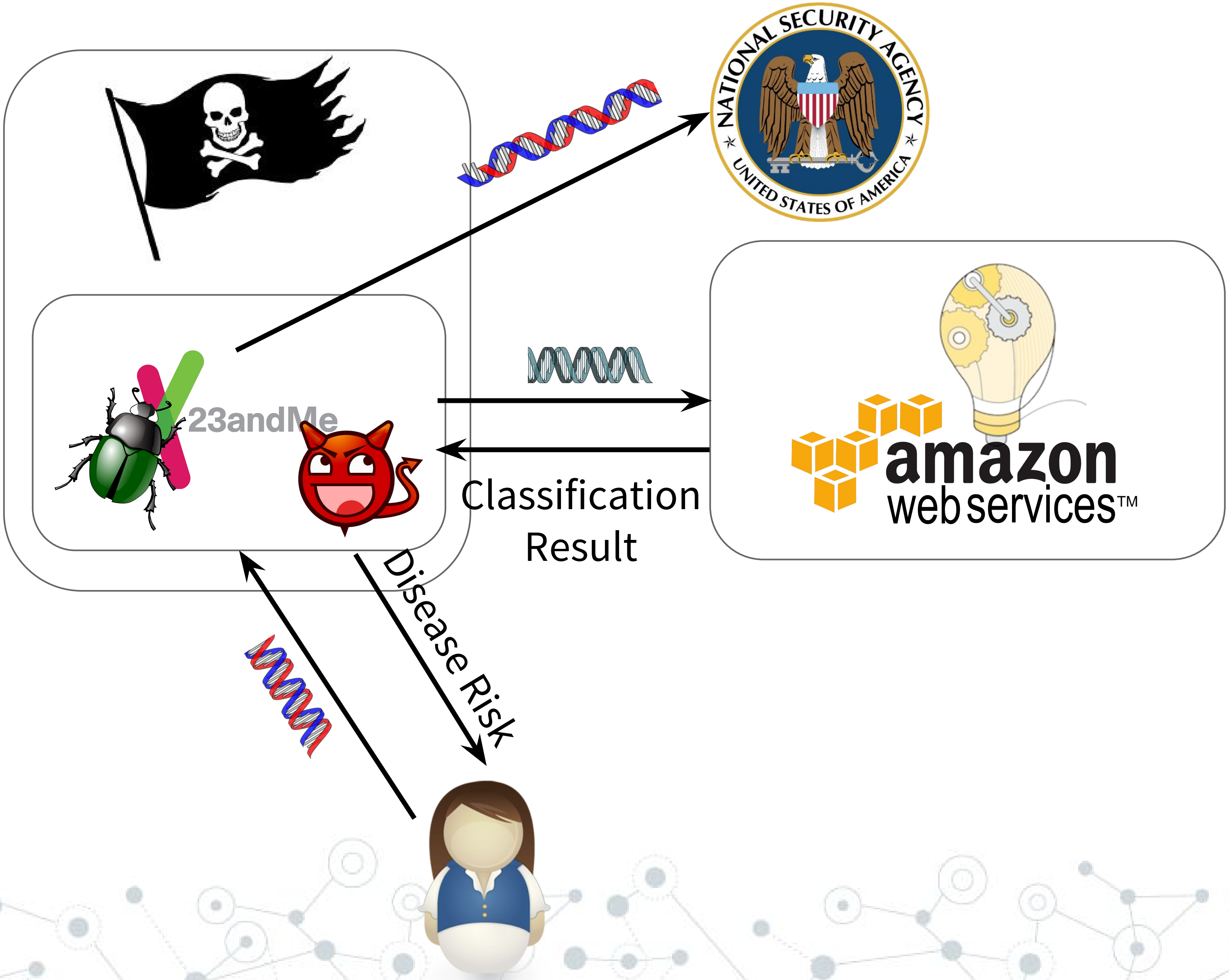
Disease risk assessment: Trust issues



Disease risk assessment: Trust issues



Disease risk assessment: Trust issues



Talk outline

Introduction

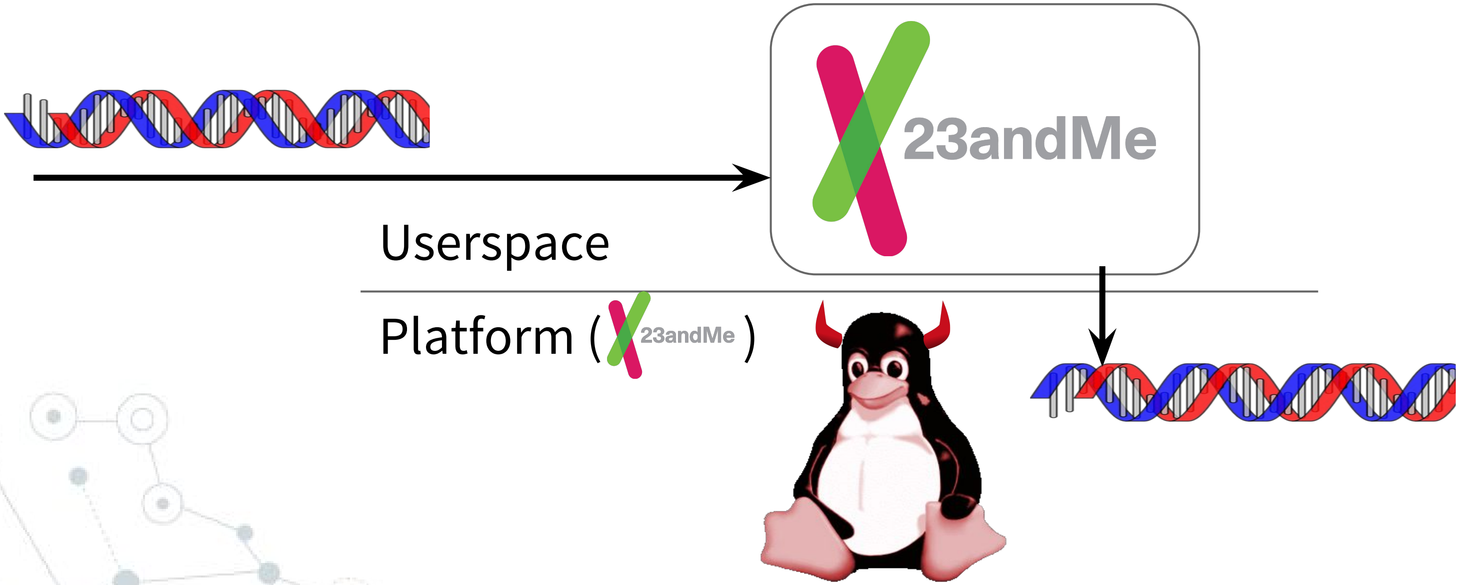
Controlling untrusted modules

Covert and side channels

Evaluation

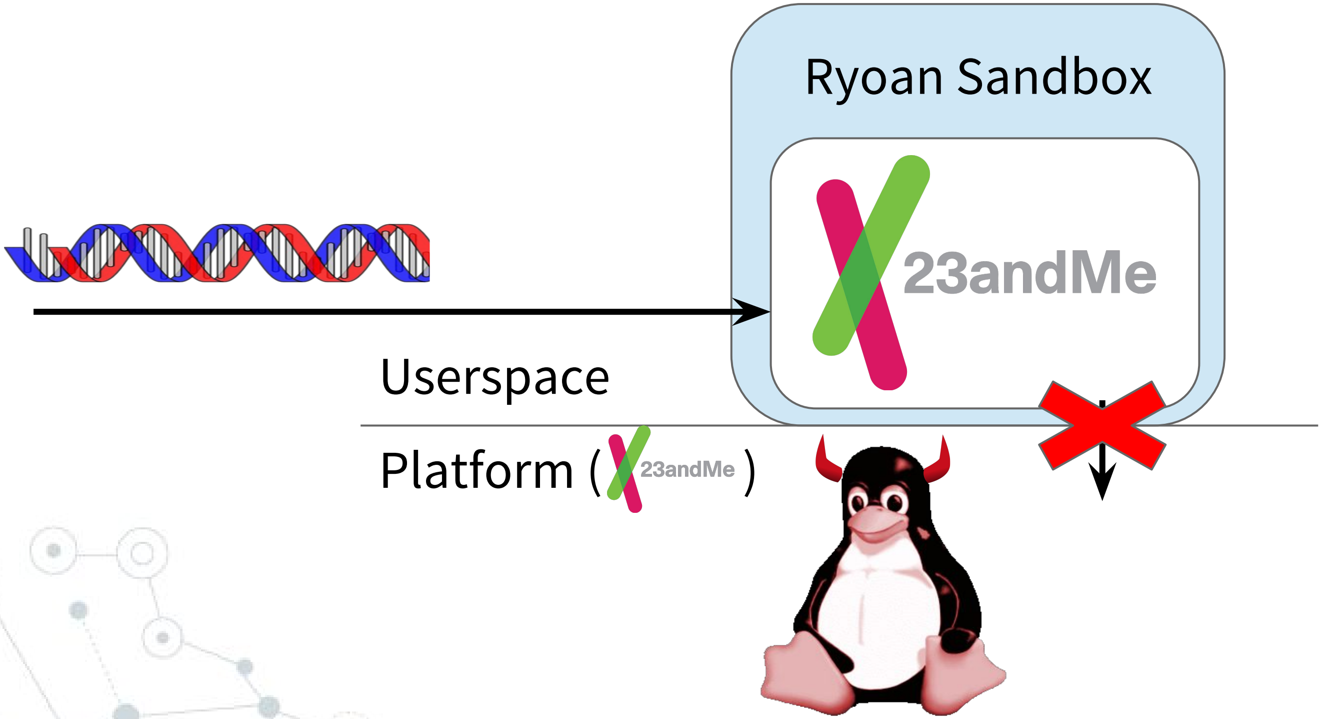
Ryoan's goals

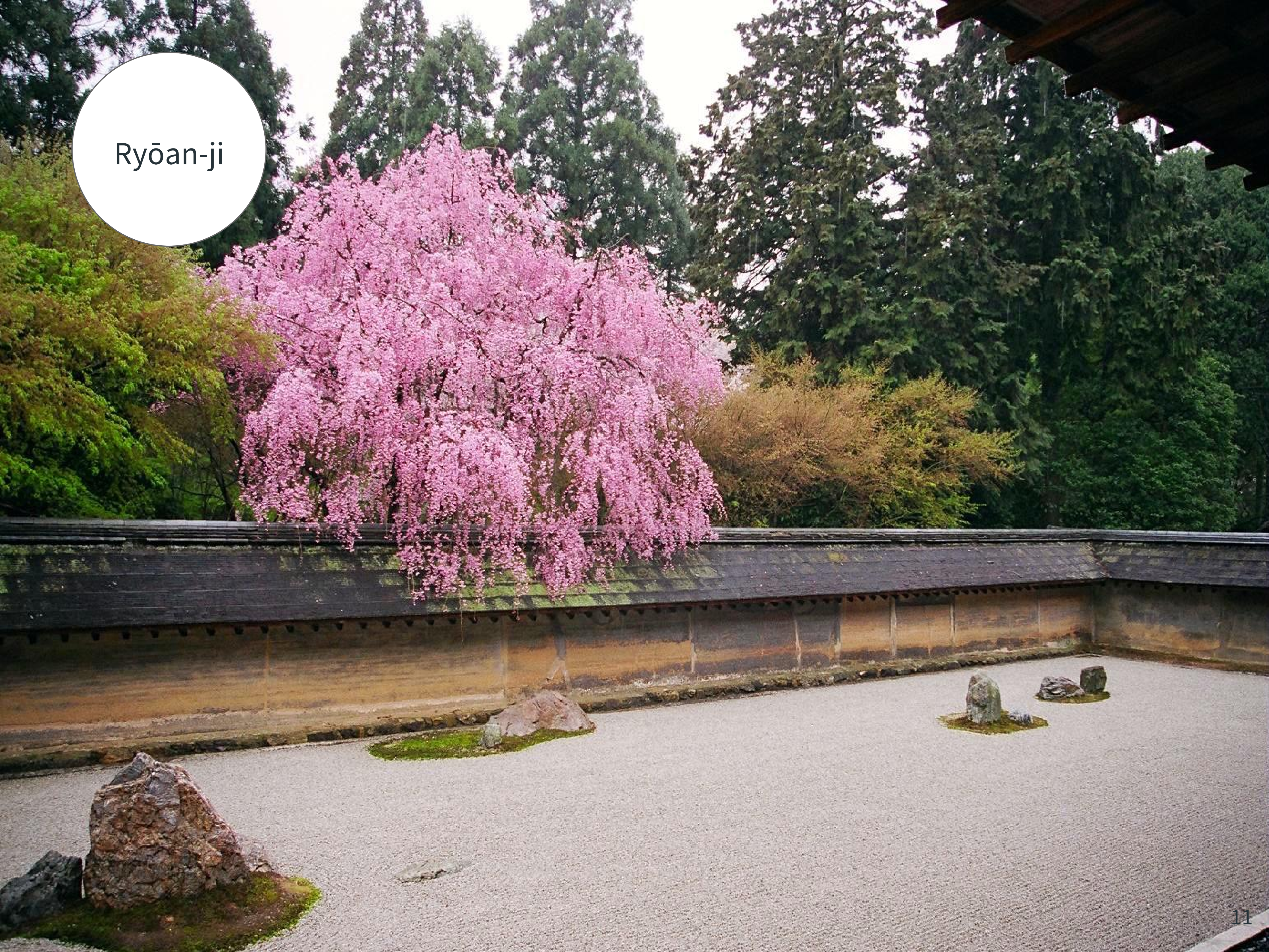
- Provide user data secrecy
  - Without trusting the application
  - Without trusting the platform (OS, Hypervisor)
- Support cooperation between service providers



Ryoan's goals

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





Ryōan-ji

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

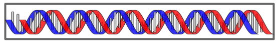
Service Providers

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

Everyone

- Trusts Ryoan
- Trusts Intel SGX

- User



- User Data

🟥 - Untrusted Code

- Untrusted Platform

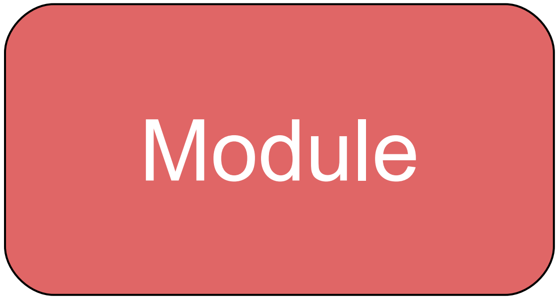
🟦 - Ryoan

⬛ - SGX

Ryoan’s world

Modules

- NaCl x86 binaries from service providers
- Application logic

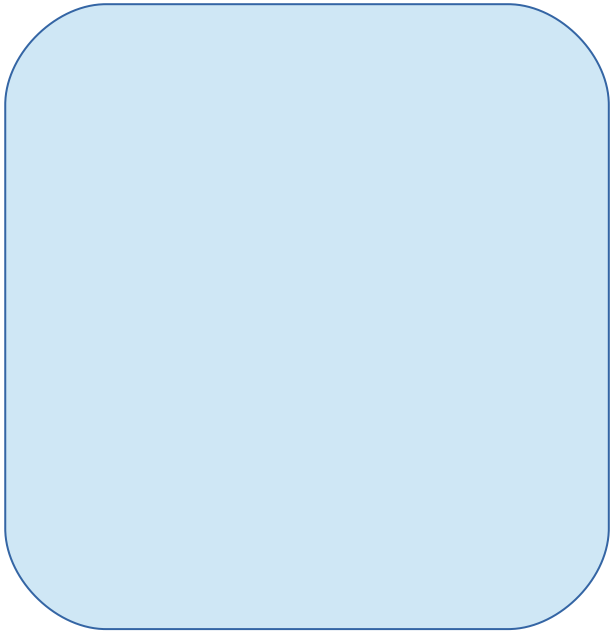


Platforms

- More service providers’ code
- Host computation

Sandboxes

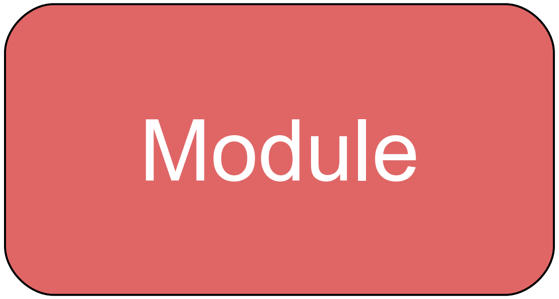
- Trusted code
- Confine modules
- Based on Google’s Native Client (NaCl)



Ryoan’s world

Modules

- NaCl x86 binaries from service providers
- Application logic

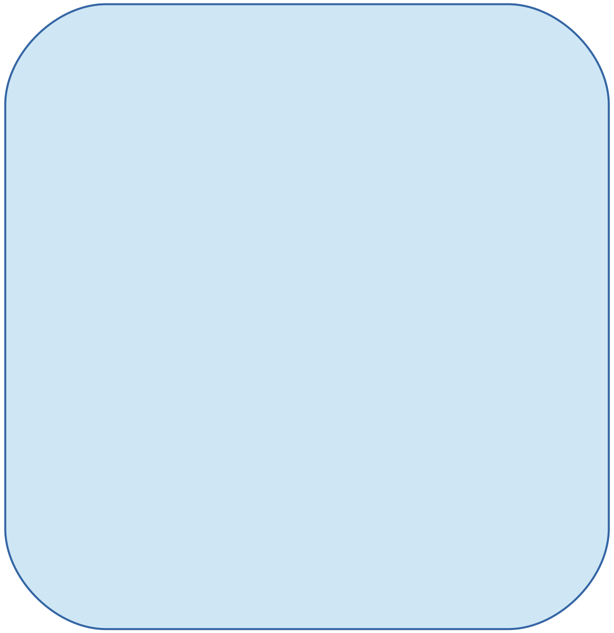


Platforms

- More service providers’ code
- Host computation

Sandboxes

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



Ryoan's world

Modules

- NaCl x86 binaries from service providers
- Application logic

Module

Platforms

- More service providers' code
- Host computation

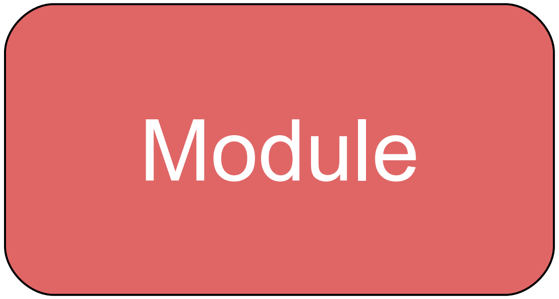
Sandboxes

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

Ryoan’s world

Modules

- NaCl x86 binaries from service providers
- Application logic

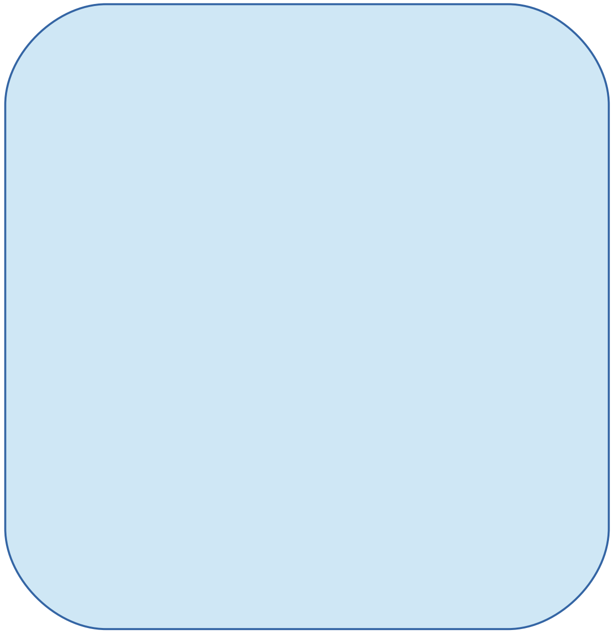

Module

Platforms

- More service providers’ code
- Host computation

Sandboxes

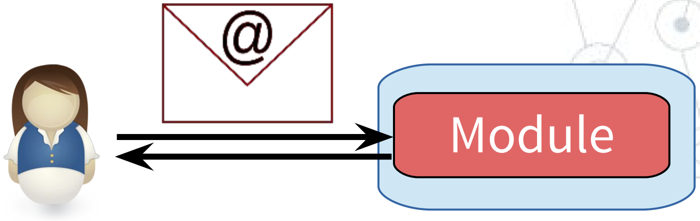
- Trusted code
- Confine modules
- Based on Google’s Native Client (NaCl)



Ryoan 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

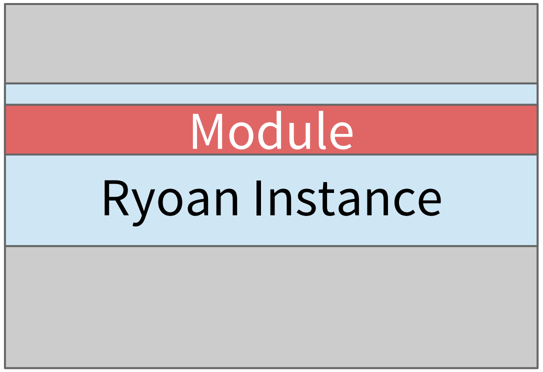
Enclaves

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

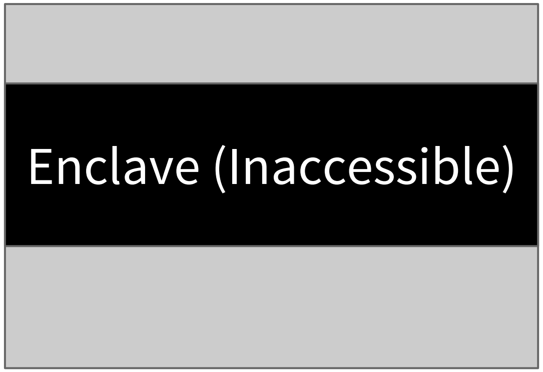
Attestations

- Hardware signed hashes of initial code and data

Enclave Code’s View

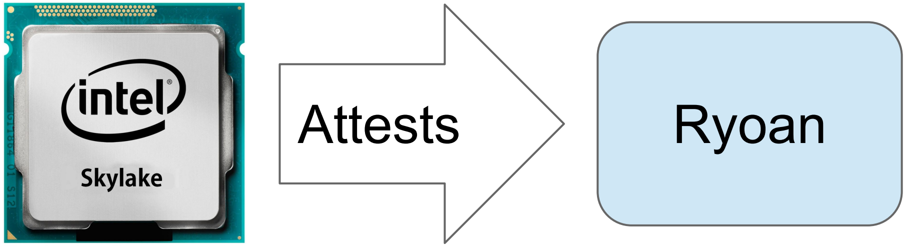


Other Code’s View

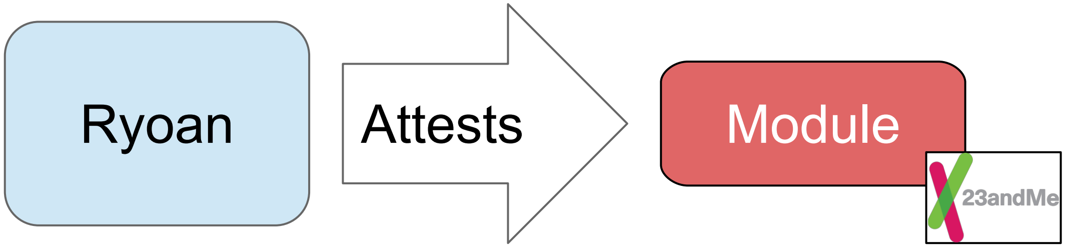


Chain of trust

- 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

Solution:

- Execute module inside of an enclave

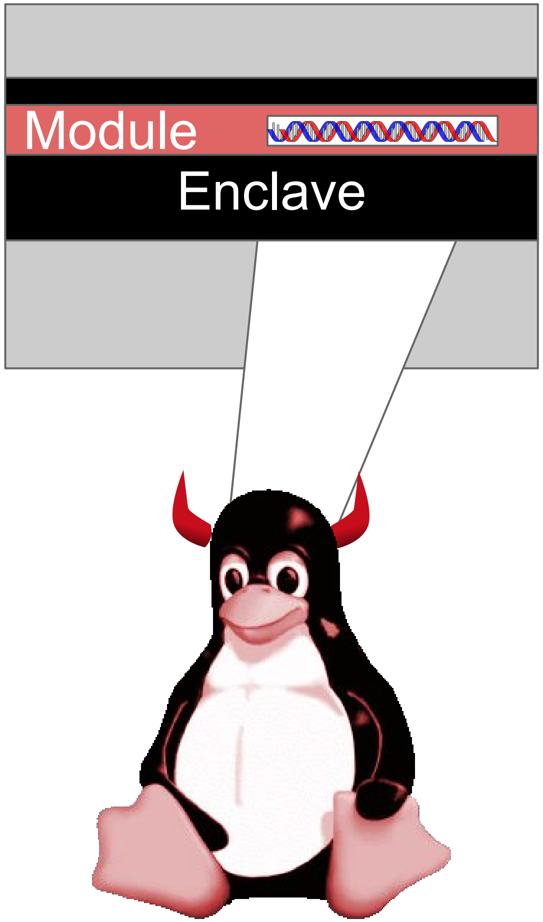
Confining untrusted code

Problem:

- Platform can read secrets out of memory

Solution:

- Execute module inside of an enclave



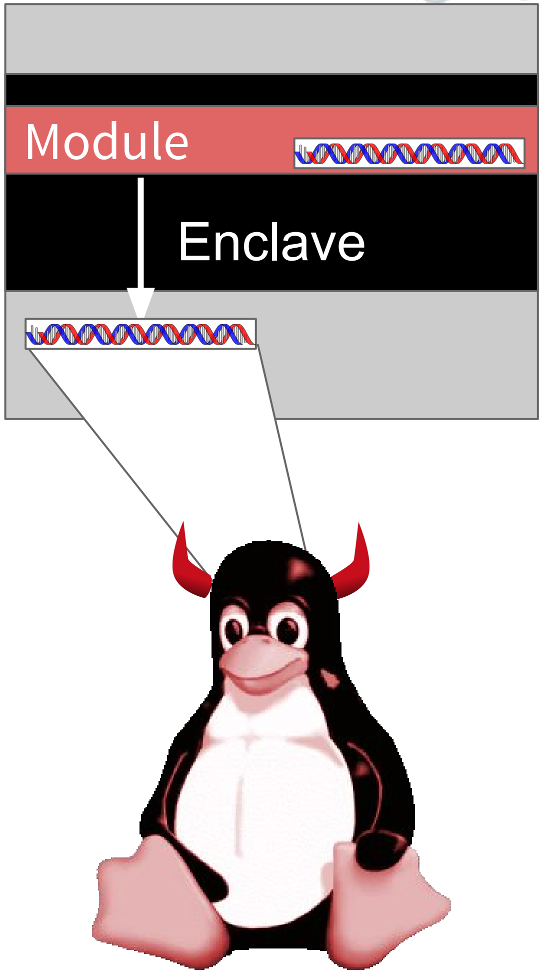
Confining untrusted code

Problem:

- Module can copy secrets to non-enclave memory

Solution:

- Restrict accessible memory with a sandbox
  - Property of NaCl



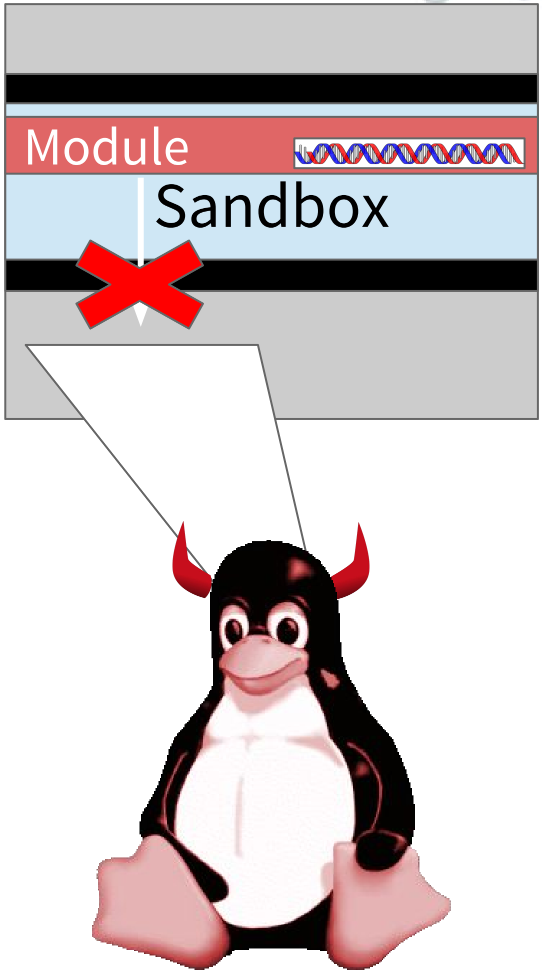
Confining untrusted code

Problem:

- Module can copy secrets to non-enclave memory

Solution:

- Restrict accessible memory with a sandbox
  - Property of NaCl



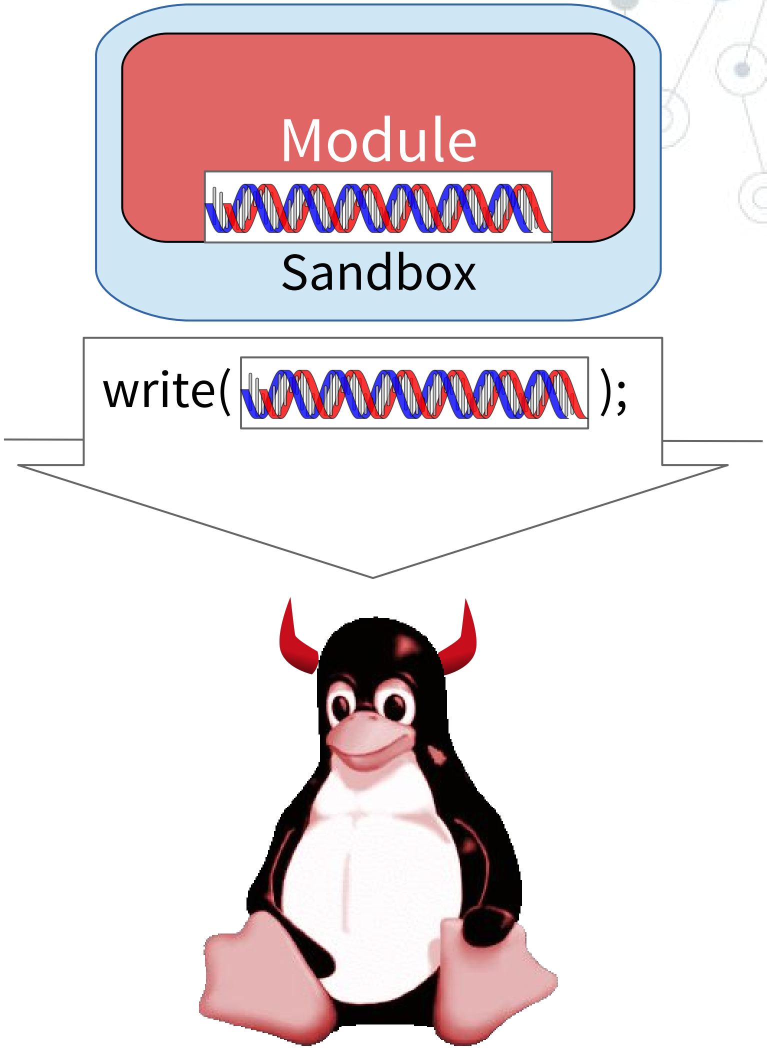
Confining untrusted code

Problem:

- Modules can use system calls to write out user data

Solution:

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



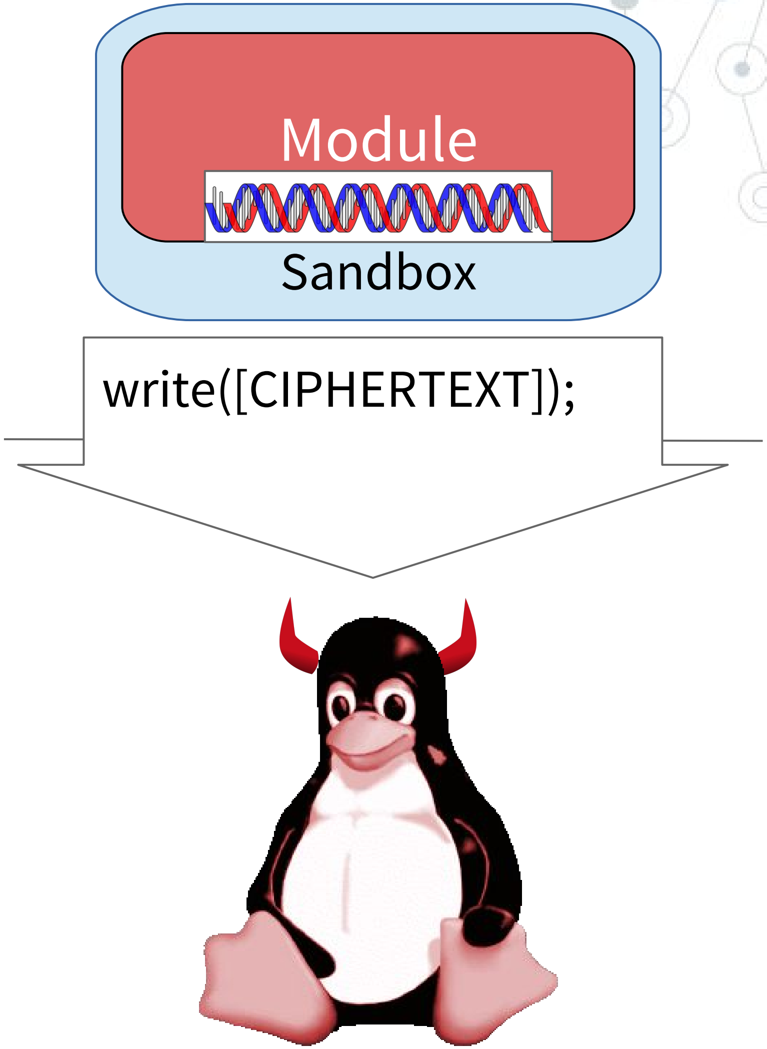
Confining untrusted code

Problem:

- Modules can use system calls to write out user data

Solution:

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



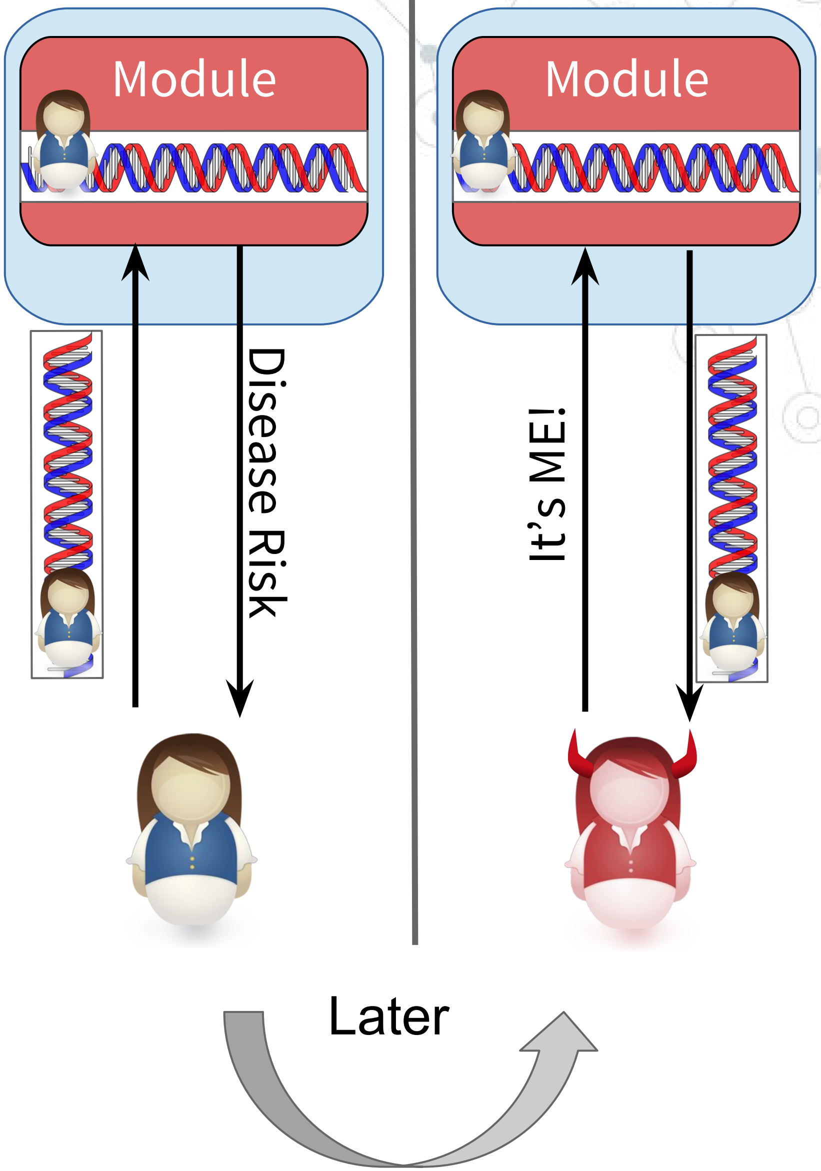
Confining untrusted code

Problem:

- Modules can collude with users to steal data

Solution:

- Don't let modules keep state between requests



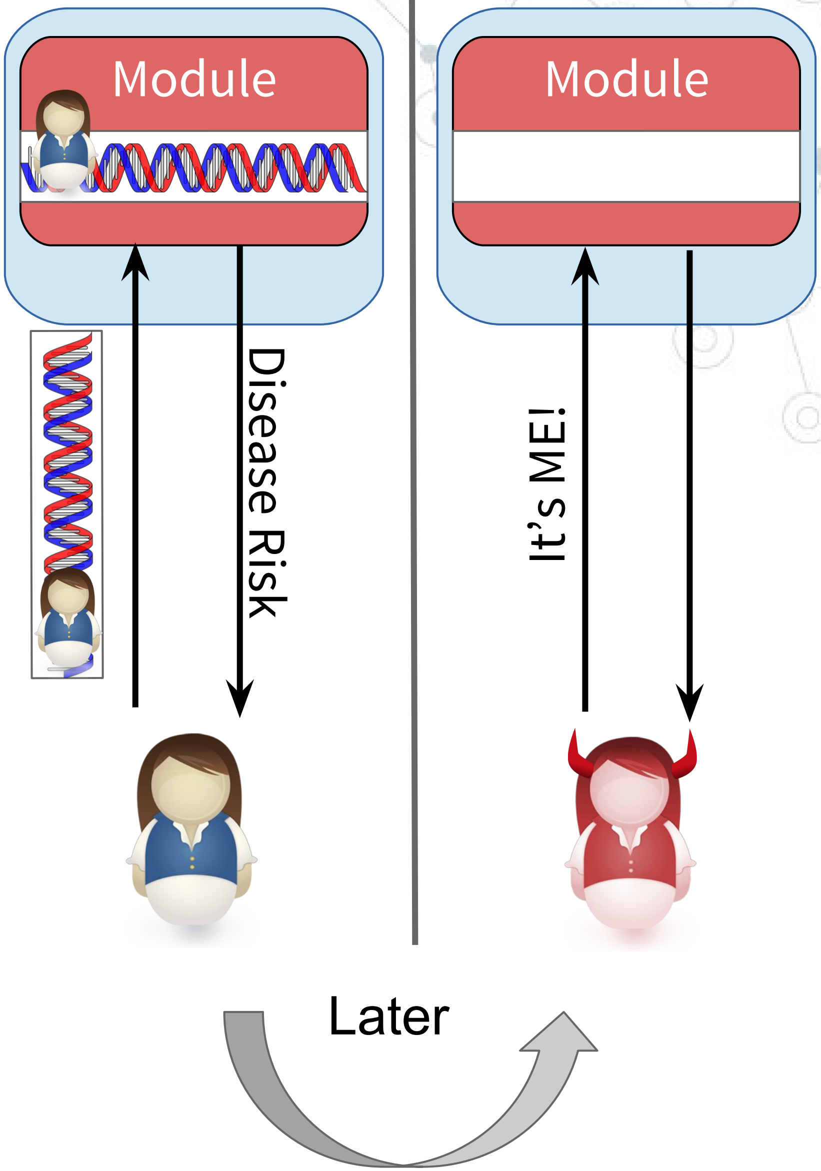
Confining untrusted code

Problem:

- Modules can collude with users to steal data

Solution:

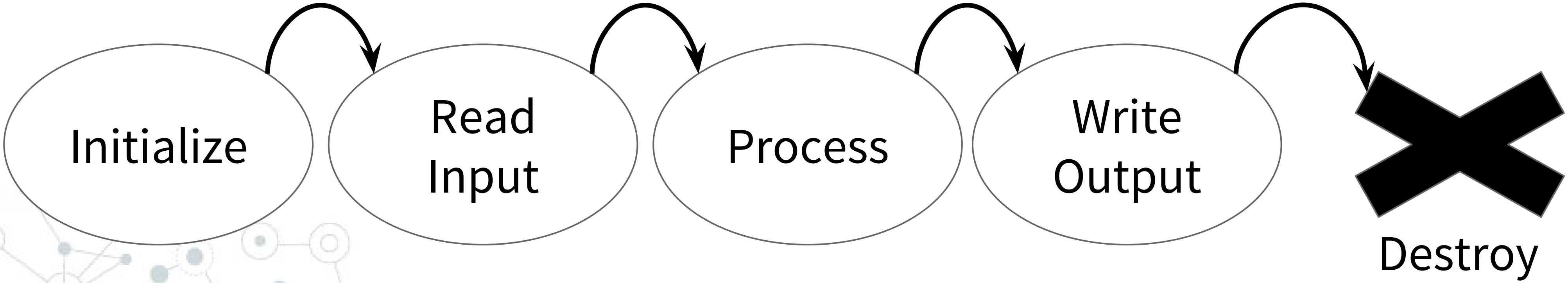
- Don't let modules keep state between requests



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

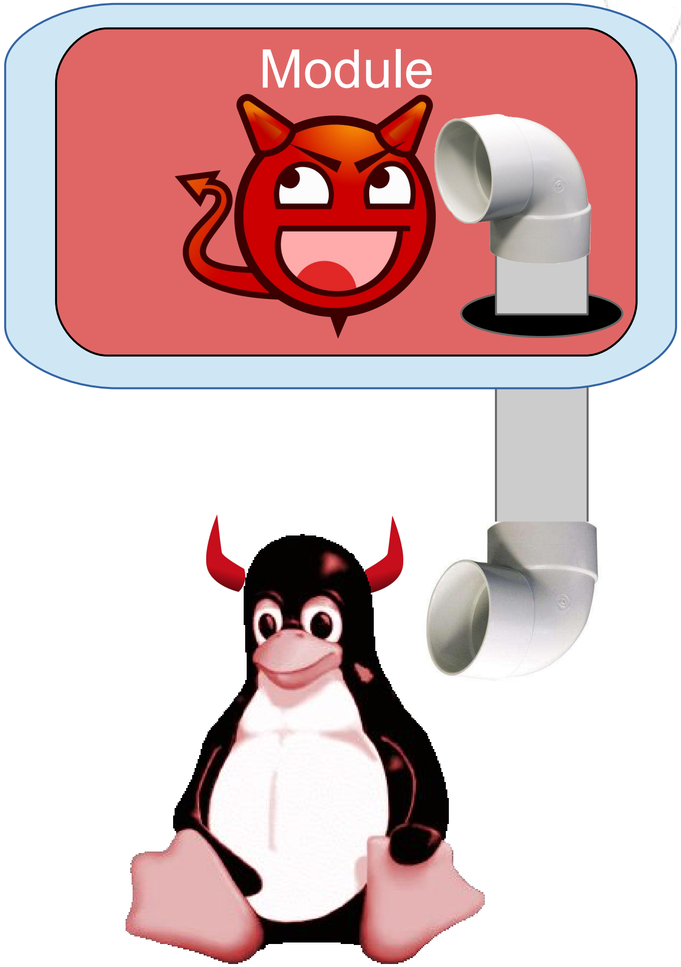
Controlling untrusted modules

Covert and side channels

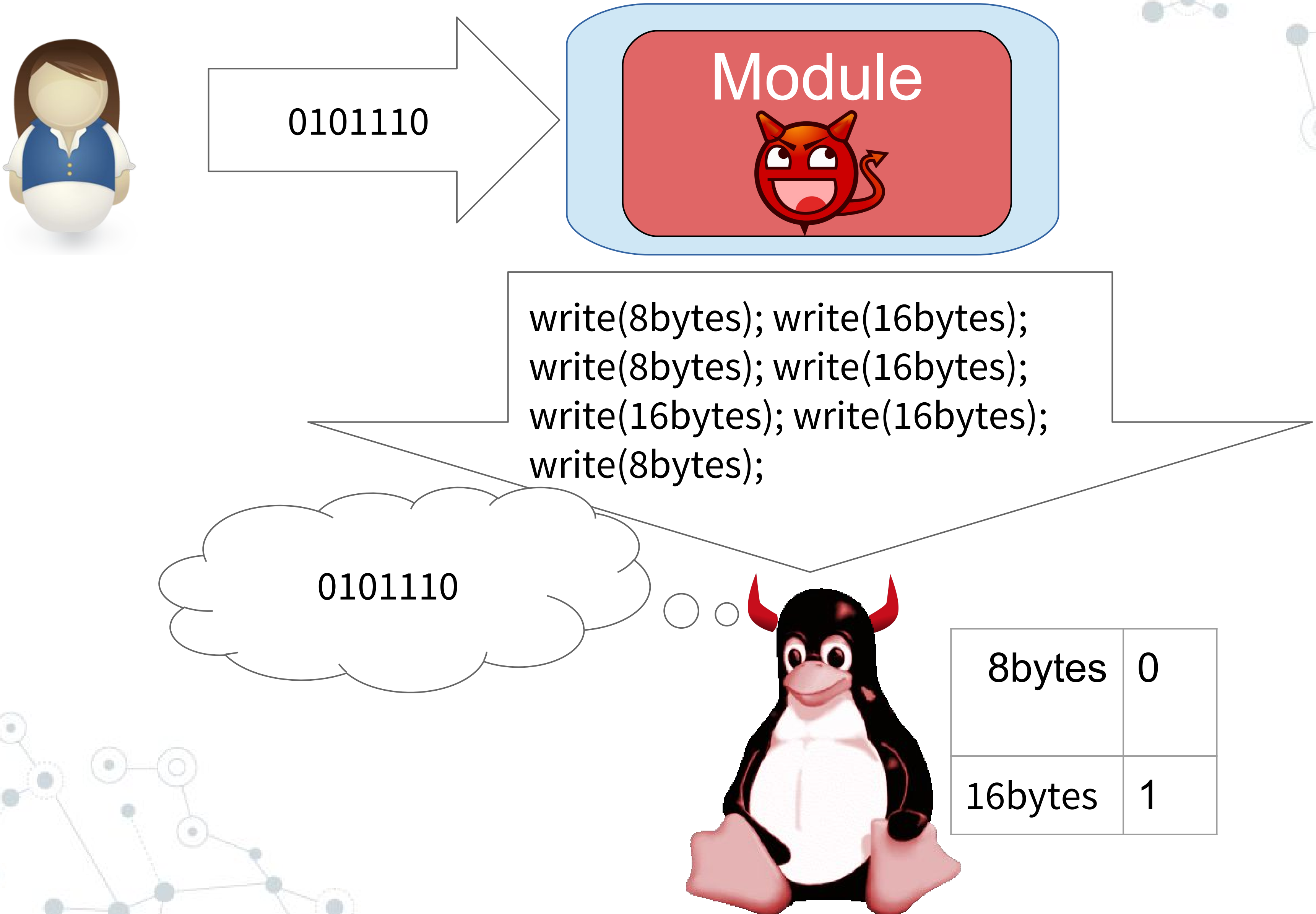
Evaluation

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



0101110

Module

```
write(8bytes); write(16bytes);
write(8bytes); write(16bytes);
write(16bytes); write(16bytes);
write(8bytes);
```

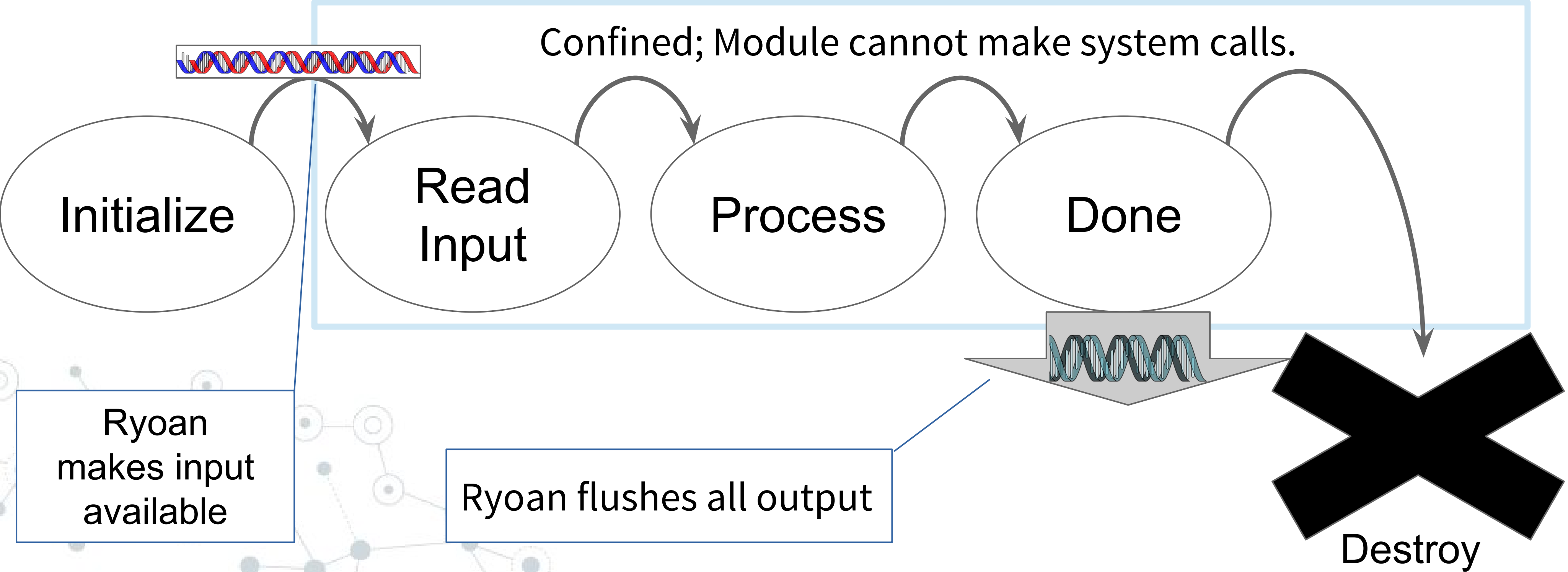
0101110

8bytes	0
16bytes	1

Eliminating system call channel

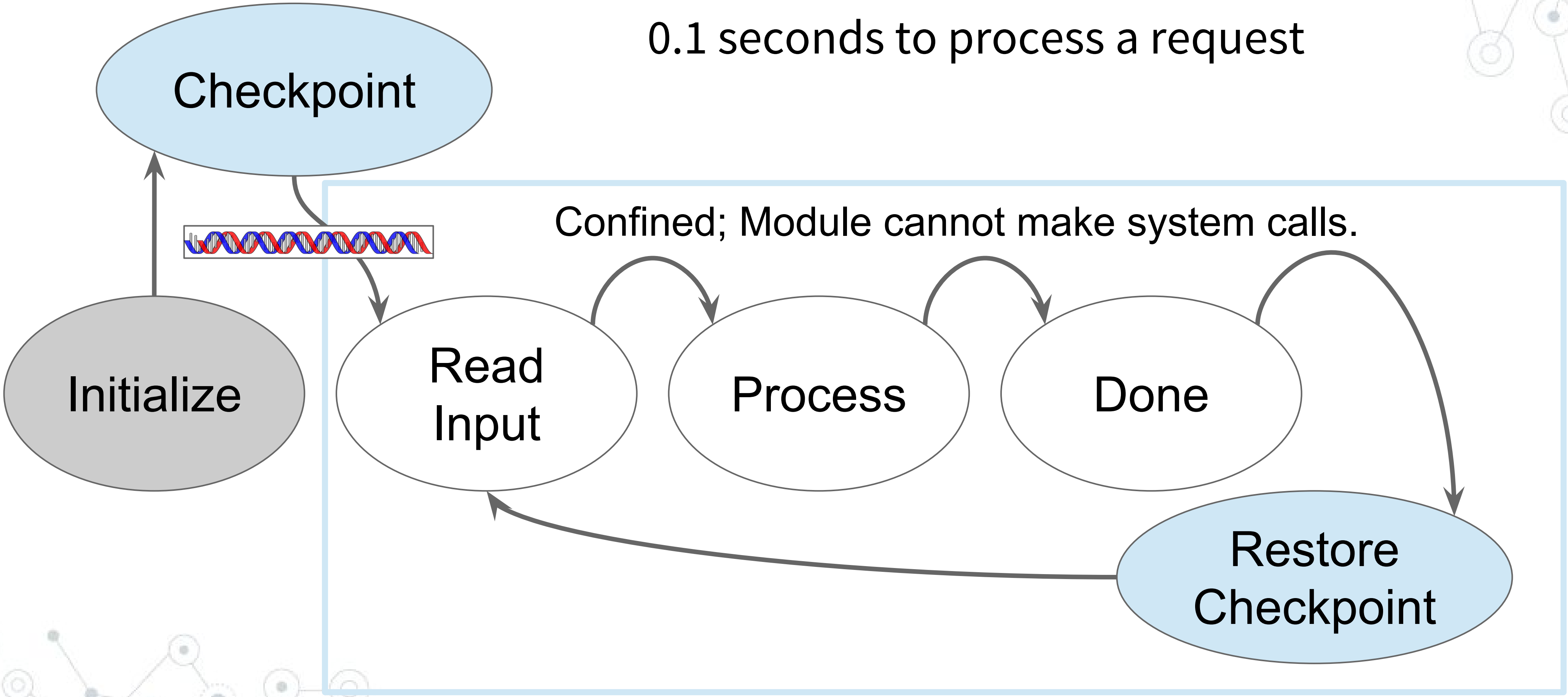
- Remove modules ability to make system calls

- Ryoan performs all data input and output independent of the content


Confined; Module cannot make system calls.
Initialize
Read Input
Process
Done
Destroy
Ryoan makes input available
Ryoan flushes all output

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

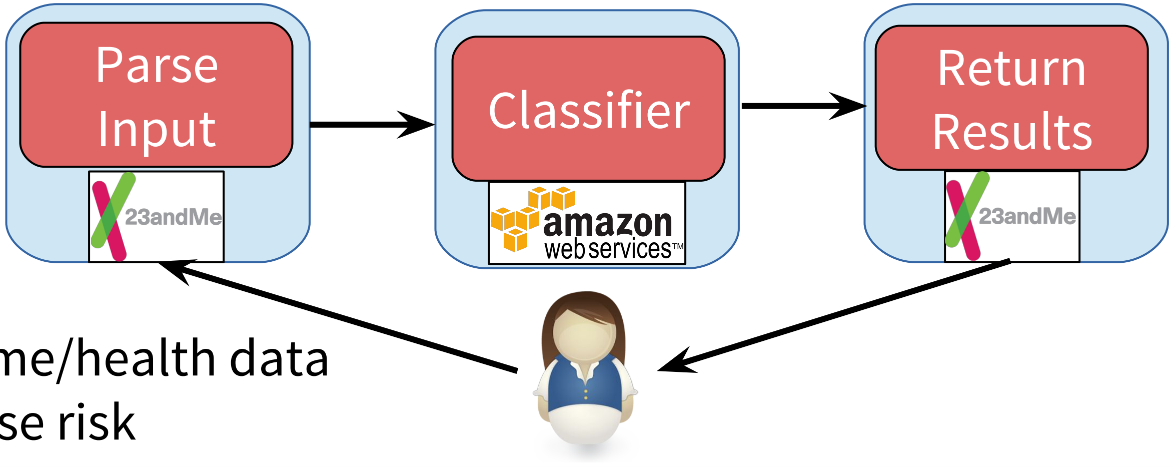
Introduction

Controlling untrusted modules

Covert channels

Evaluation

Health



Parse Input

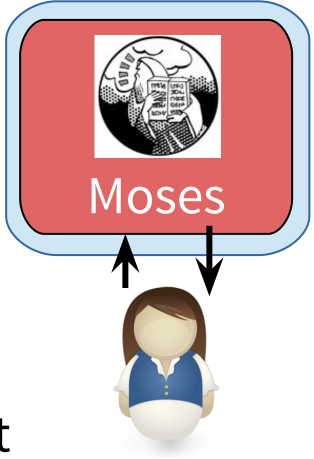
Classifier

Return Results

In: Genome/health data

Out: Disease risk

Translation

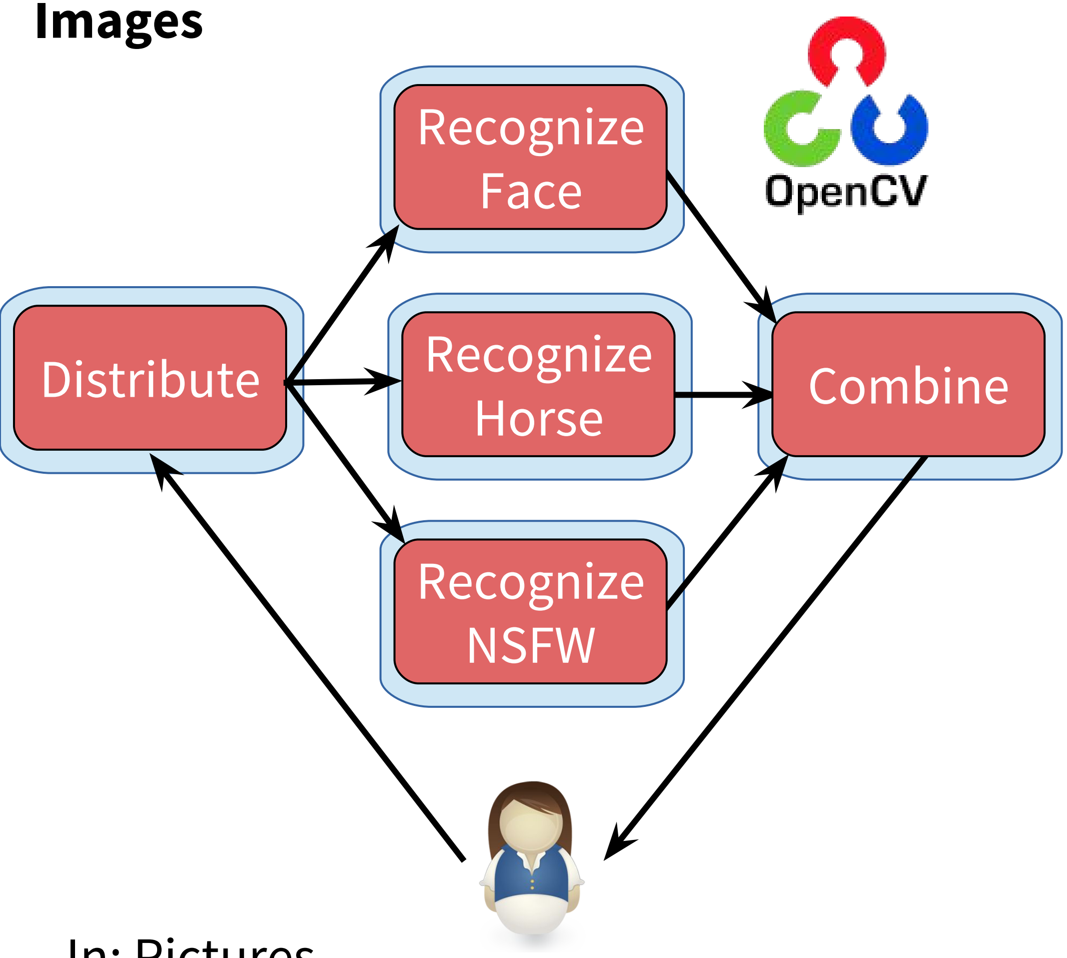


Moses

In: French text

Out: English text

Images



Distribute

Recognize Face

Recognize Horse

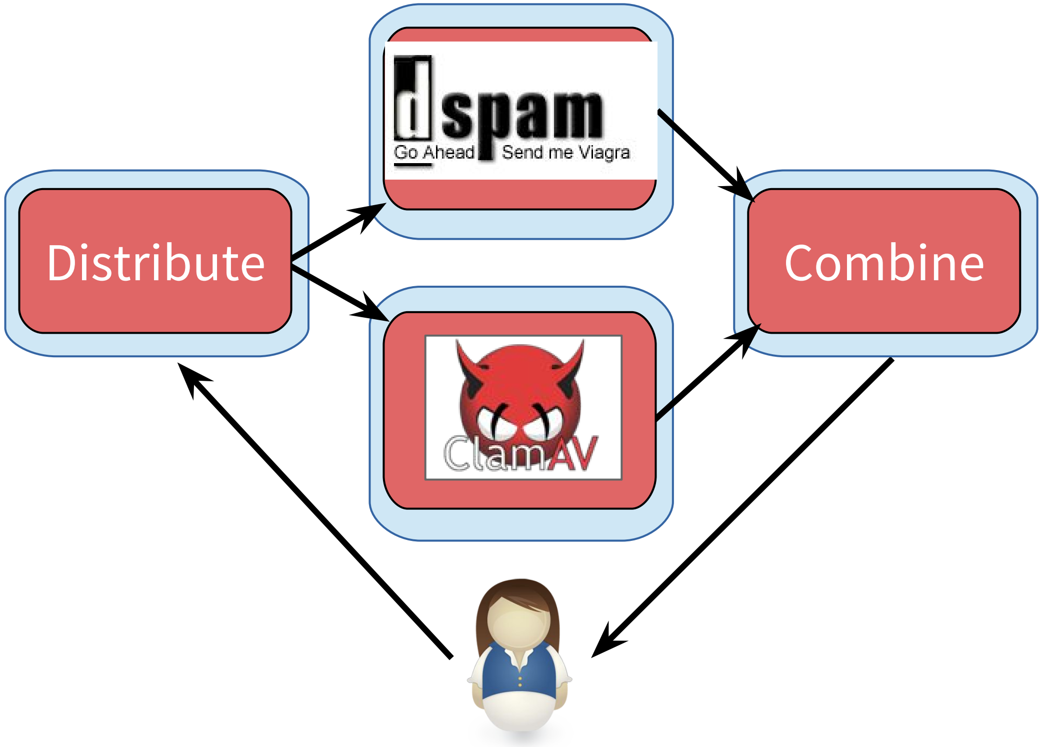
Recognize NSFW

Combine

In: Pictures

Out: Array of objects

Email



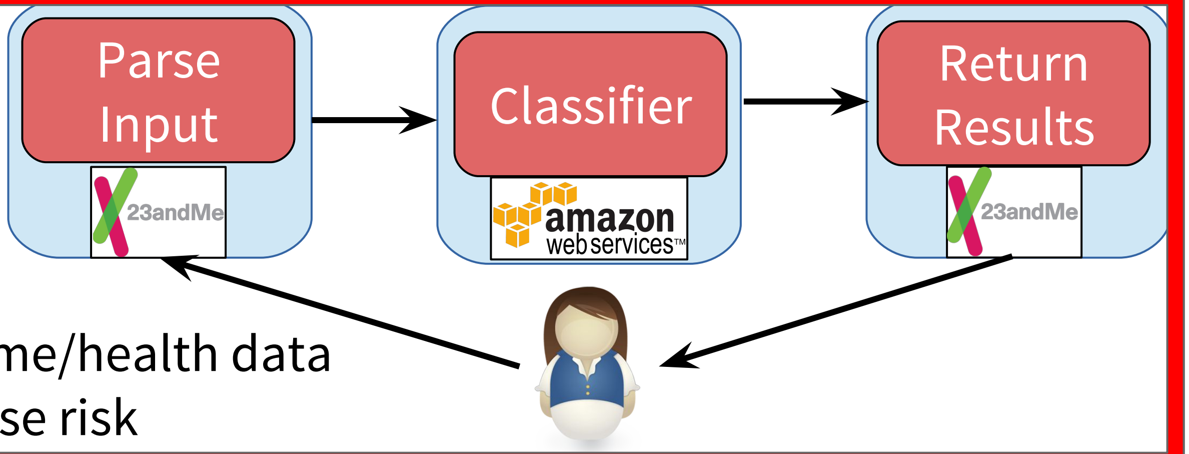
Distribute

Combine

In: Emails

Out: Spam & virus status

Health

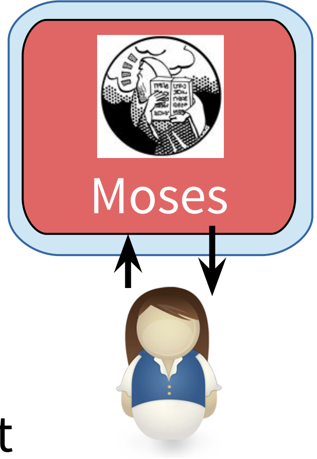

Parse Input

Classifier

Return Results

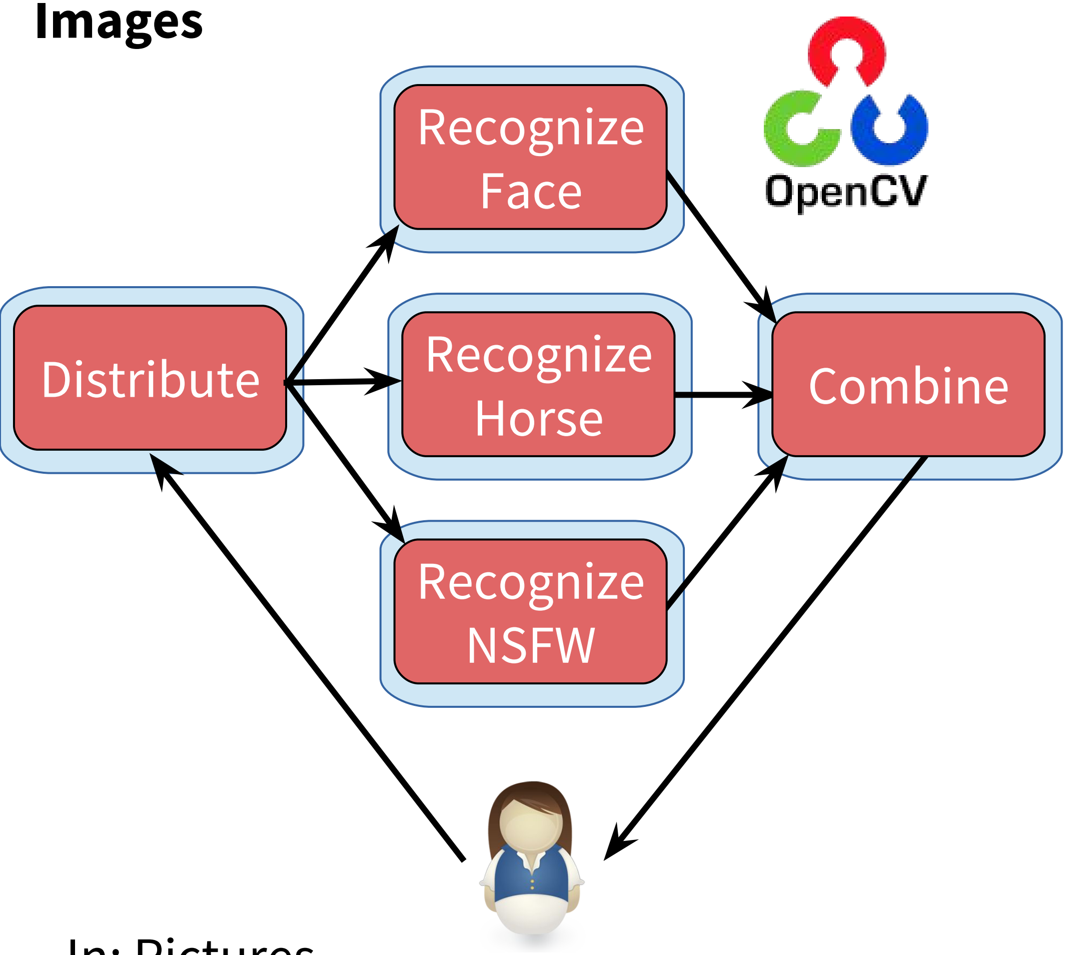
In: Genome/health data
Out: Disease risk

Translation


Moses

In: French text
Out: English text

Images


Distribute

Recognize Face

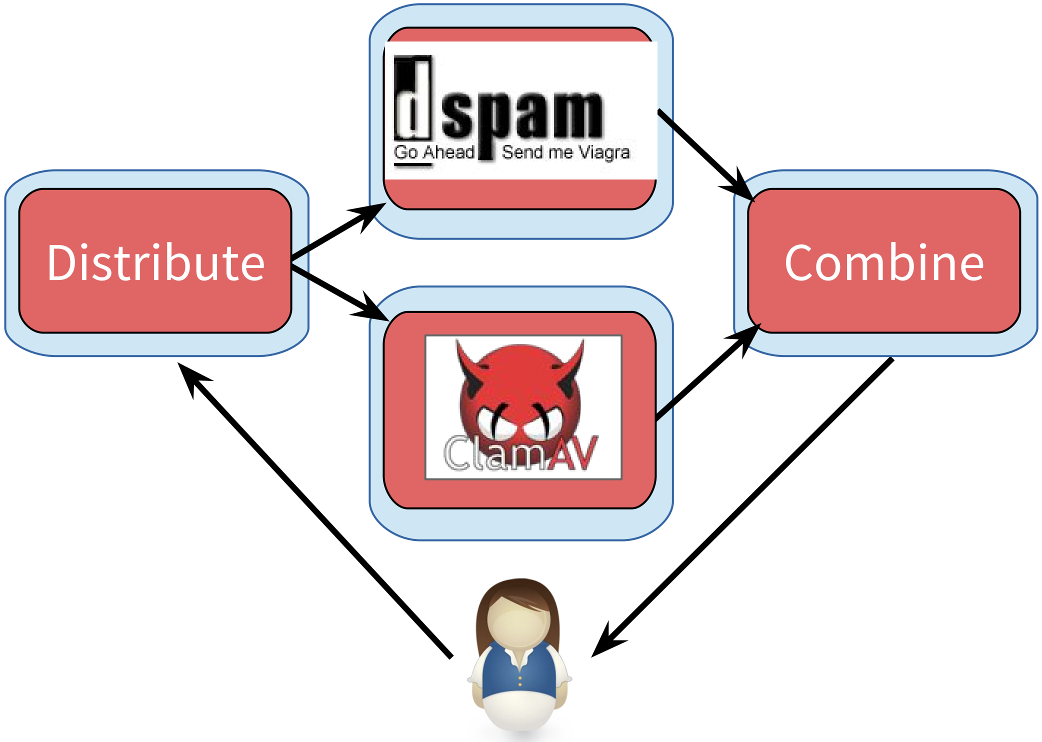
Recognize Horse

Recognize NSFW

Combine

In: Pictures
Out: Array of objects

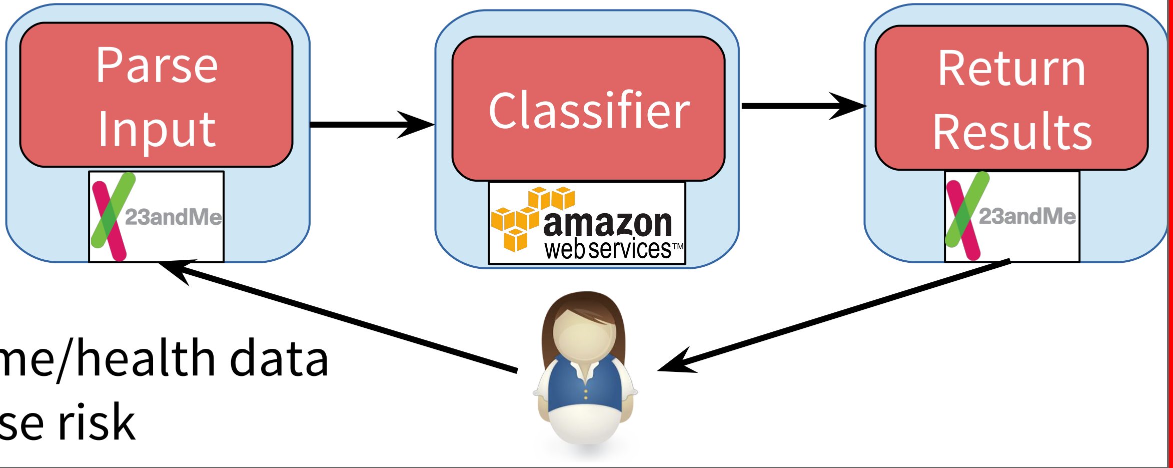
Email


Distribute

Combine

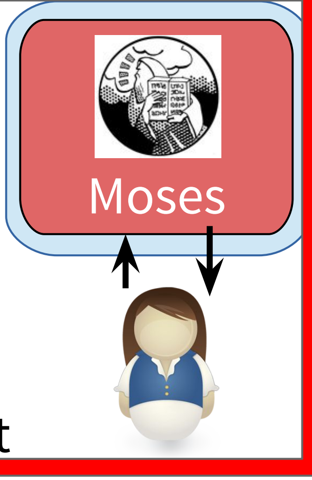
In: Emails
Out: Spam & virus status

Health



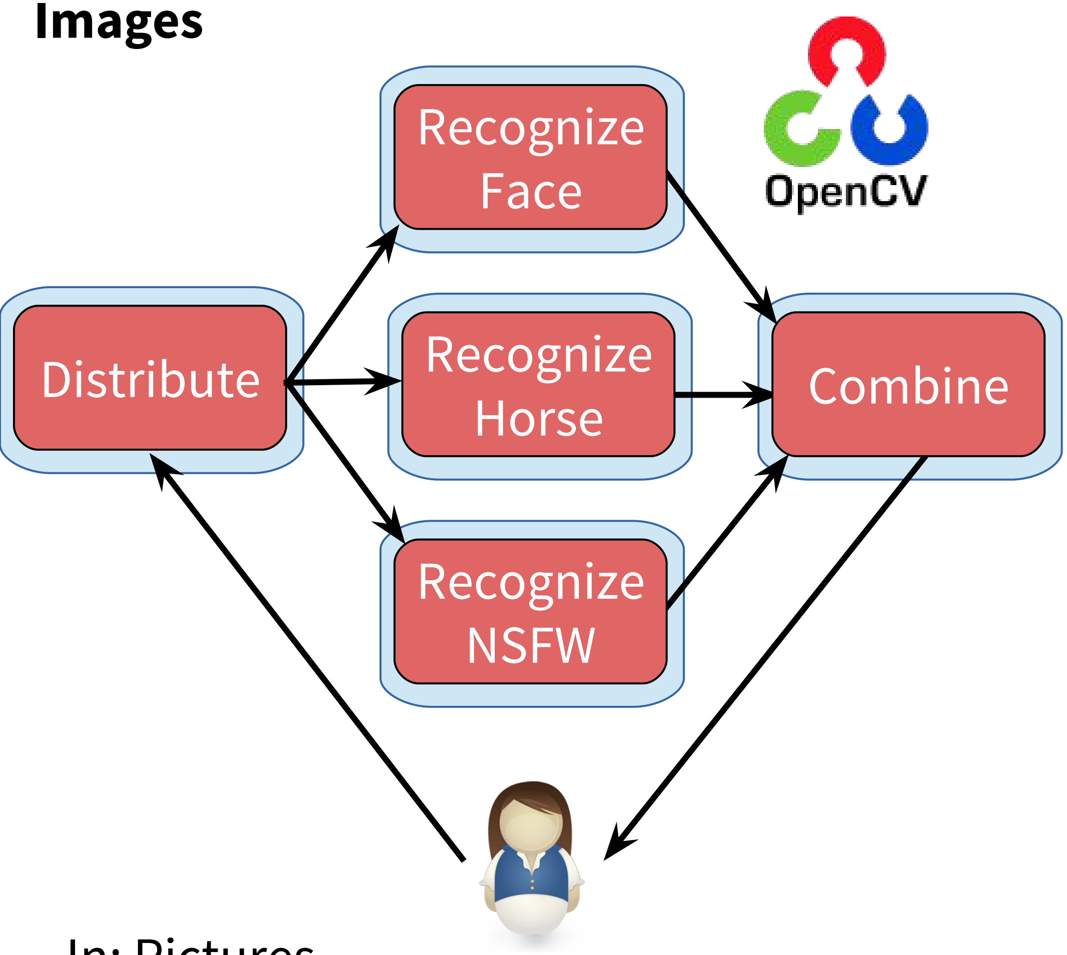
In: Genome/health data
Out: Disease risk

Translation



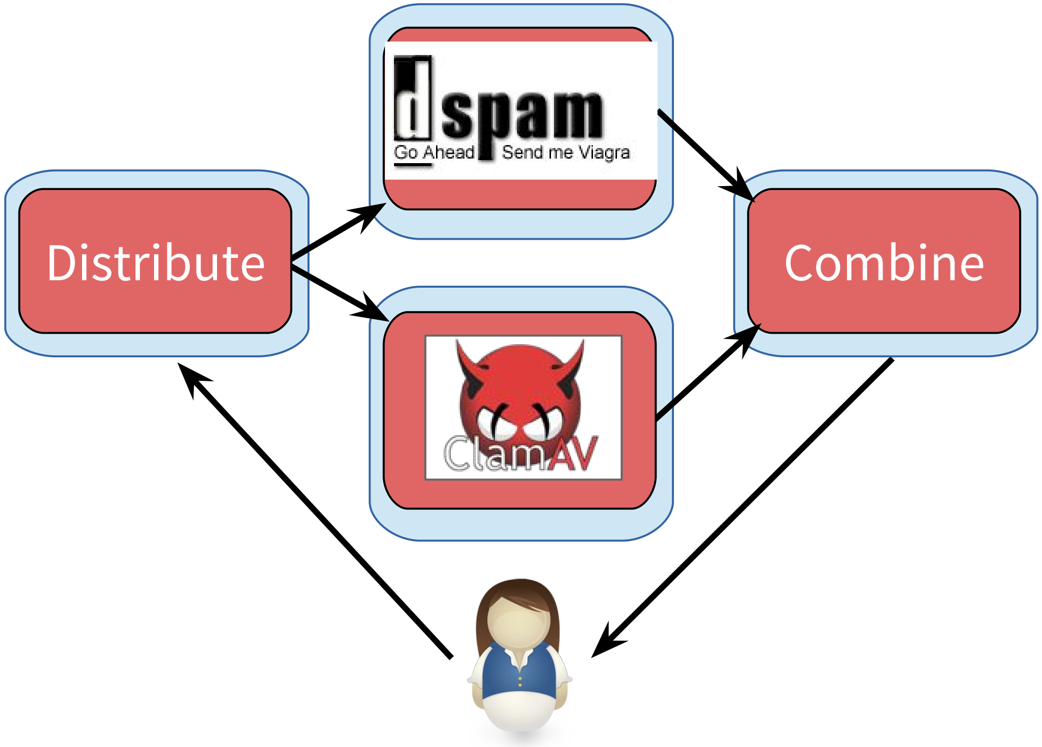
In: French text
Out: English text

Images



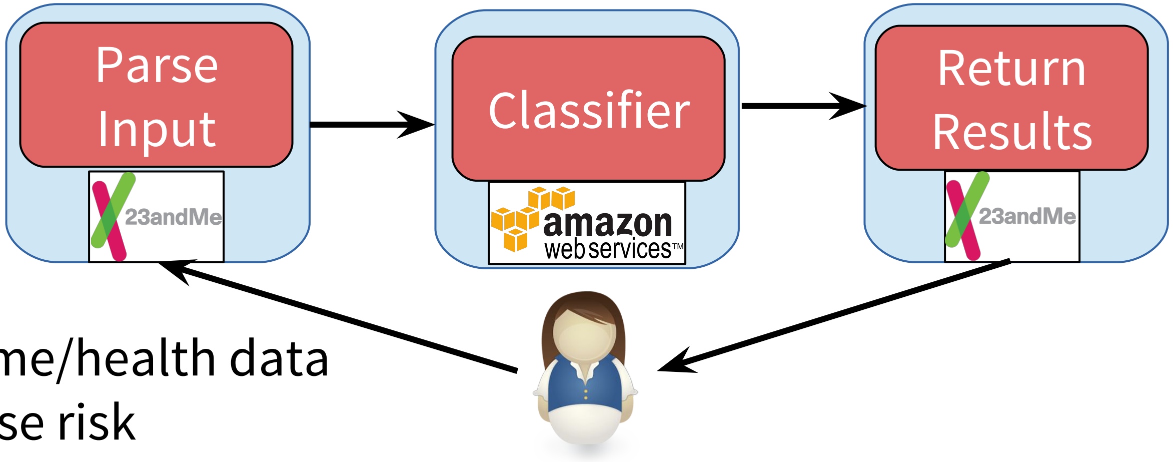
In: Pictures
Out: Array of objects

Email



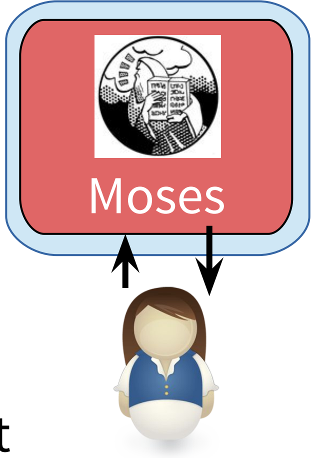
In: Emails
Out: Spam & virus status

Health



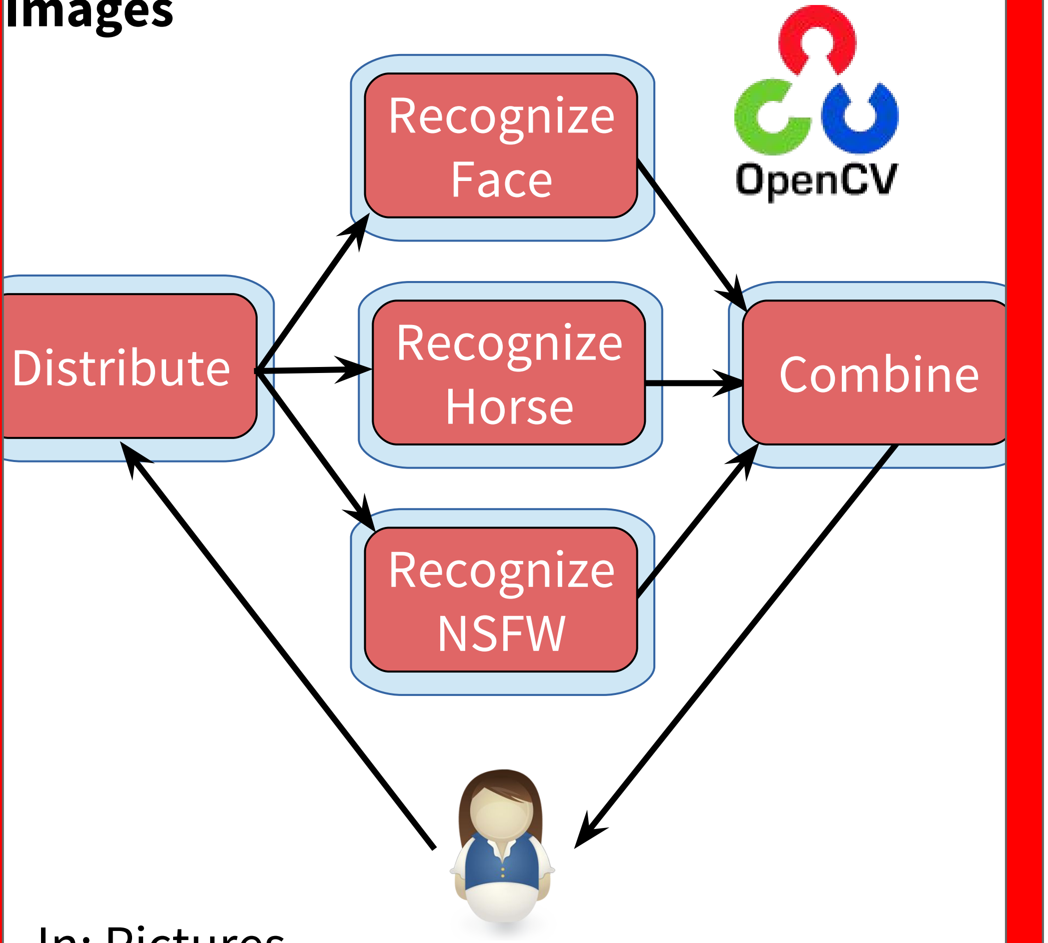
In: Genome/health data
Out: Disease risk

Translation



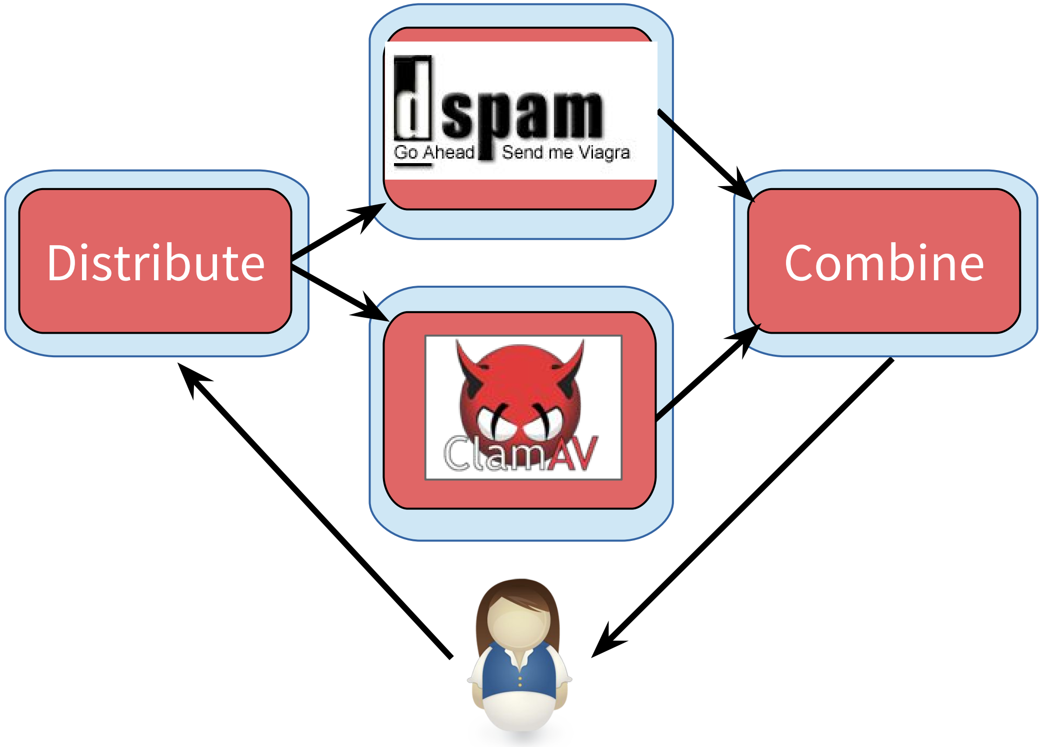
In: French text
Out: English text

Images



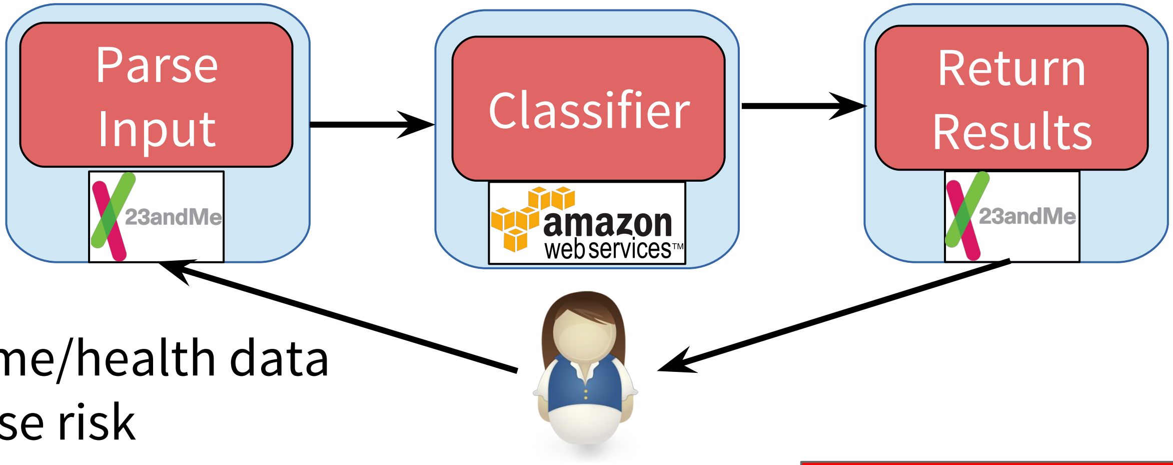
In: Pictures
Out: Array of objects

Email



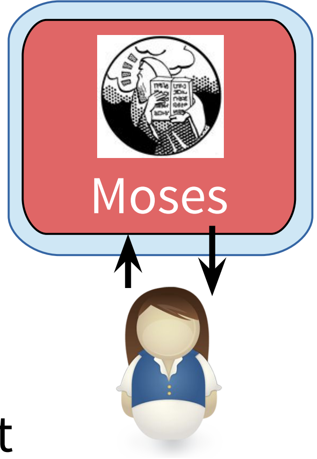
In: Emails
Out: Spam & virus status

Health



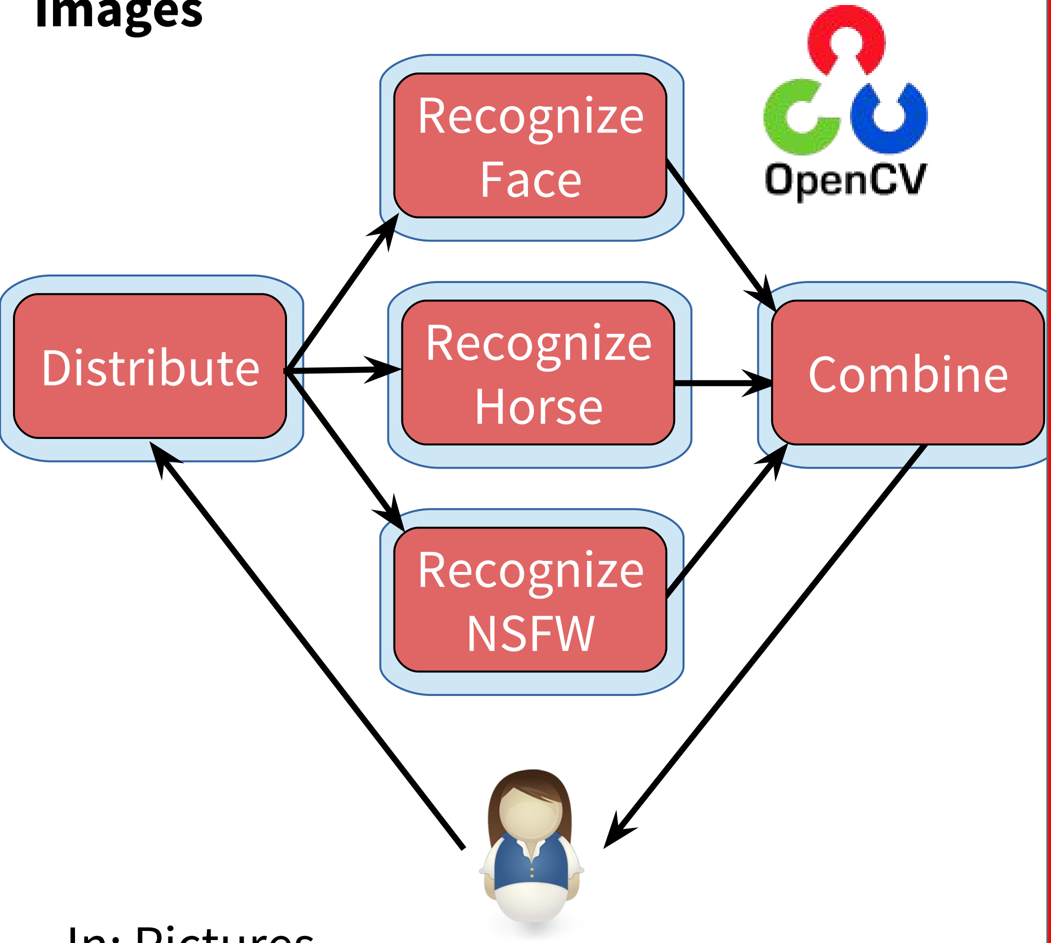
In: Genome/health data
Out: Disease risk

Translation



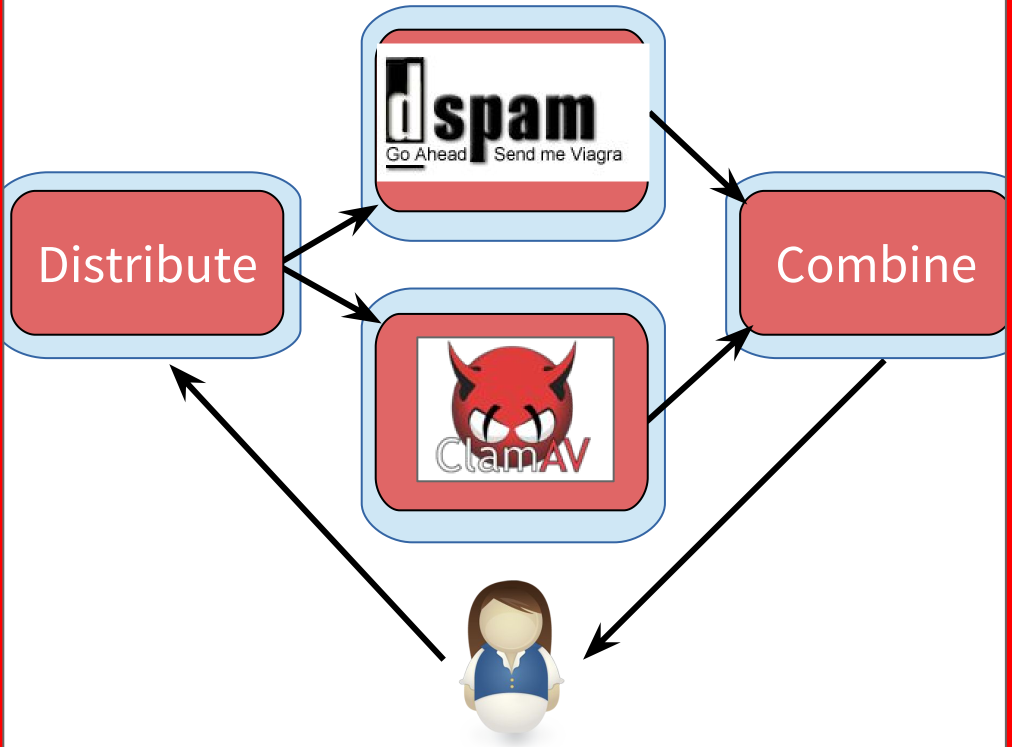
In: French text
Out: English text

Images



In: Pictures
Out: Array of objects

Email

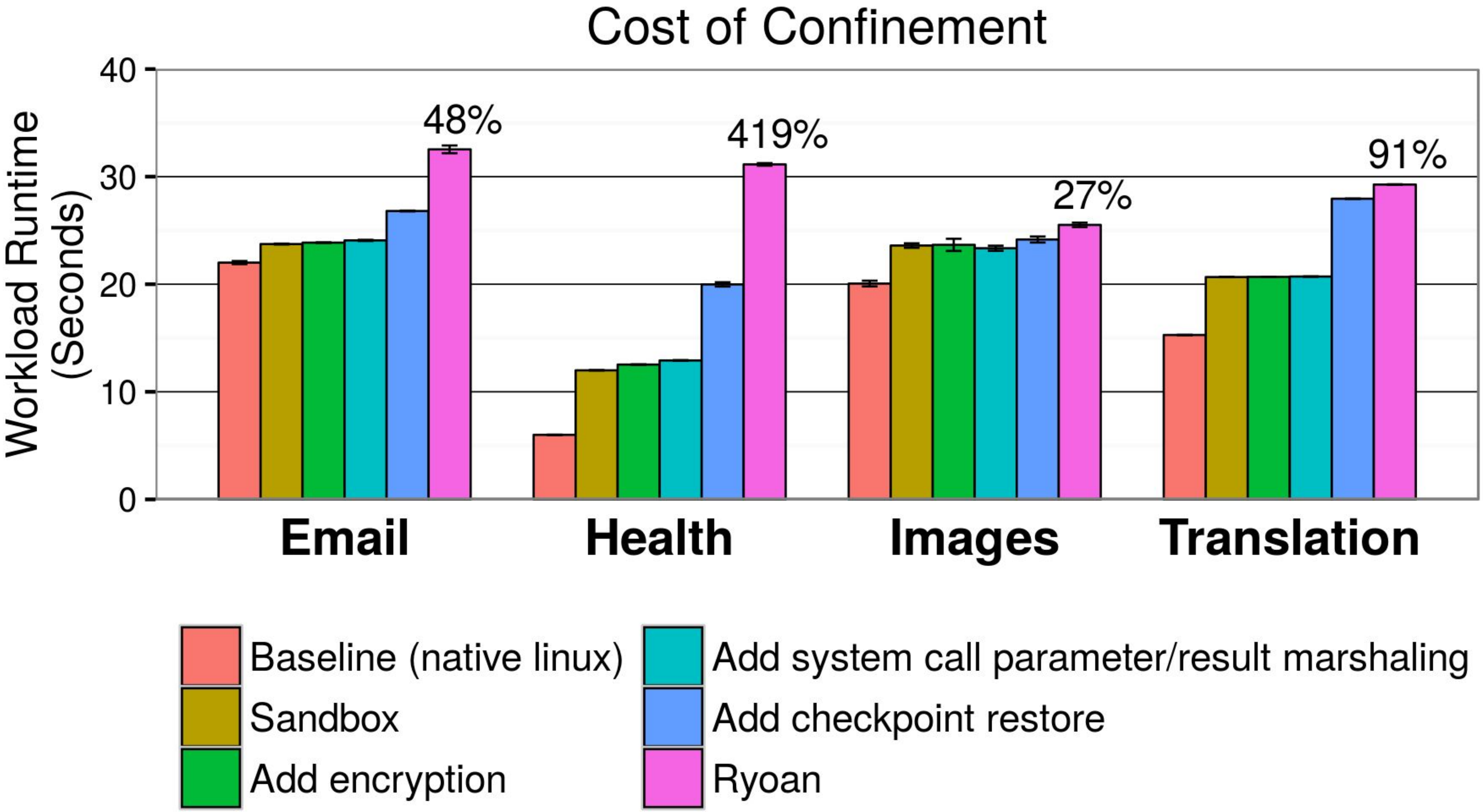


In: Emails
Out: Spam & virus status

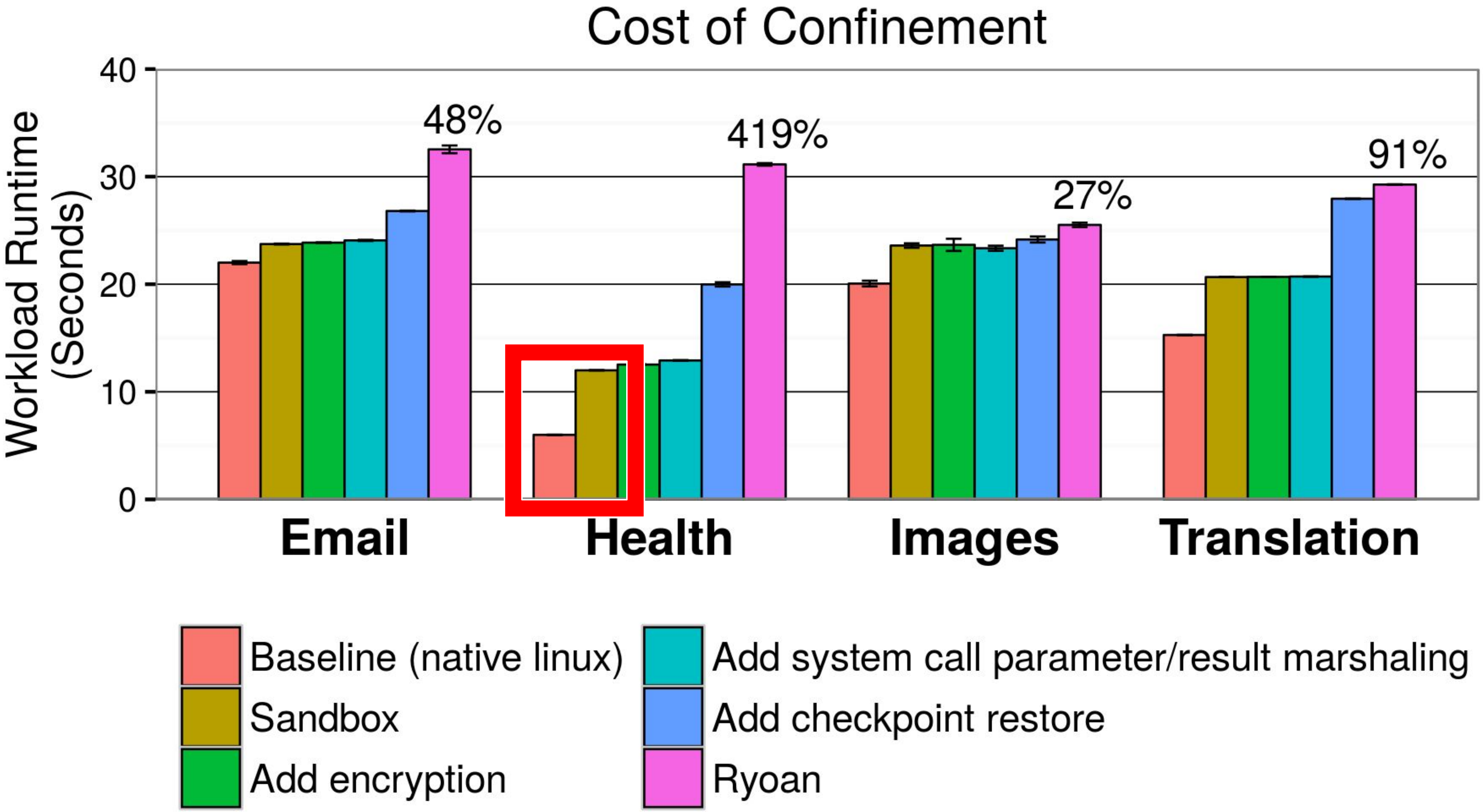
Evaluation

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

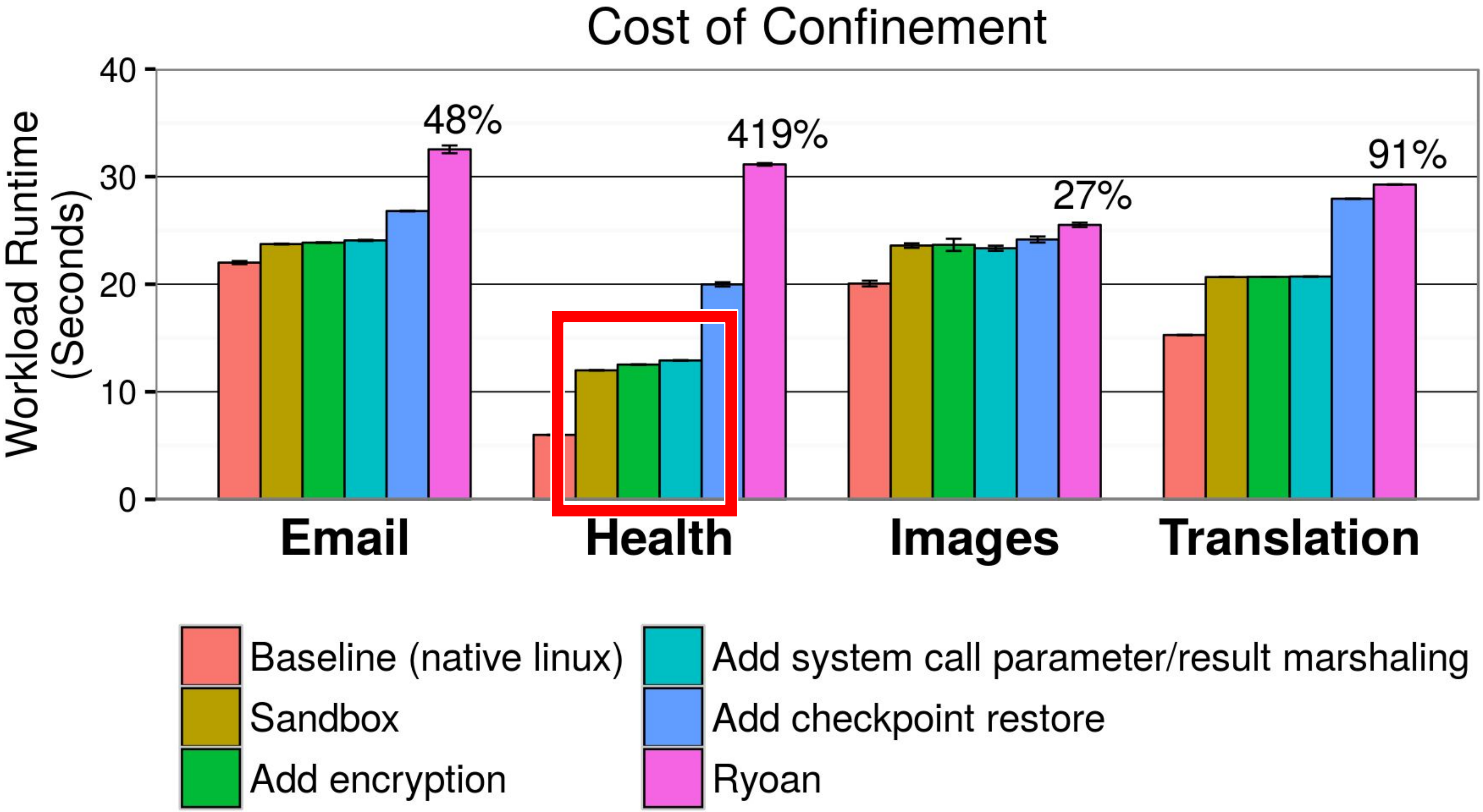
- SGX performance model
  - Measured SGX v1 latencies on our hardware
  - Estimated SGX v2 latencies (sensitivity study in paper)
  - Flush TLB on all system calls, page faults, and interrupts



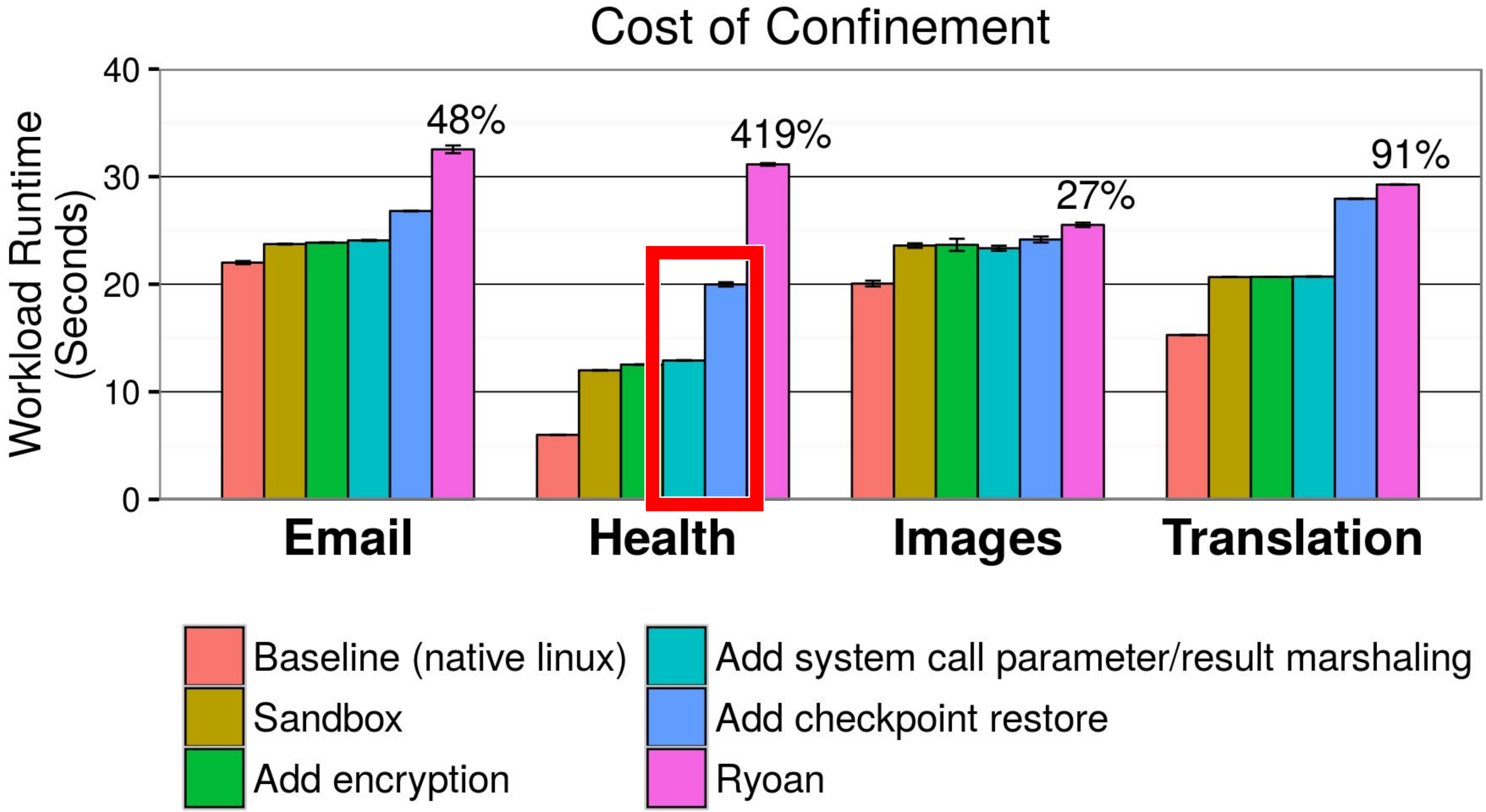
Health	20,000 1.4KB Boolean vectors from different users
Translation	30 short paragraphs, sizes 25-300B, 4.1KB total
Images	12 images, sizes 17KB-613KB
Email	250 emails, 30% with 103KB-12MB attachment



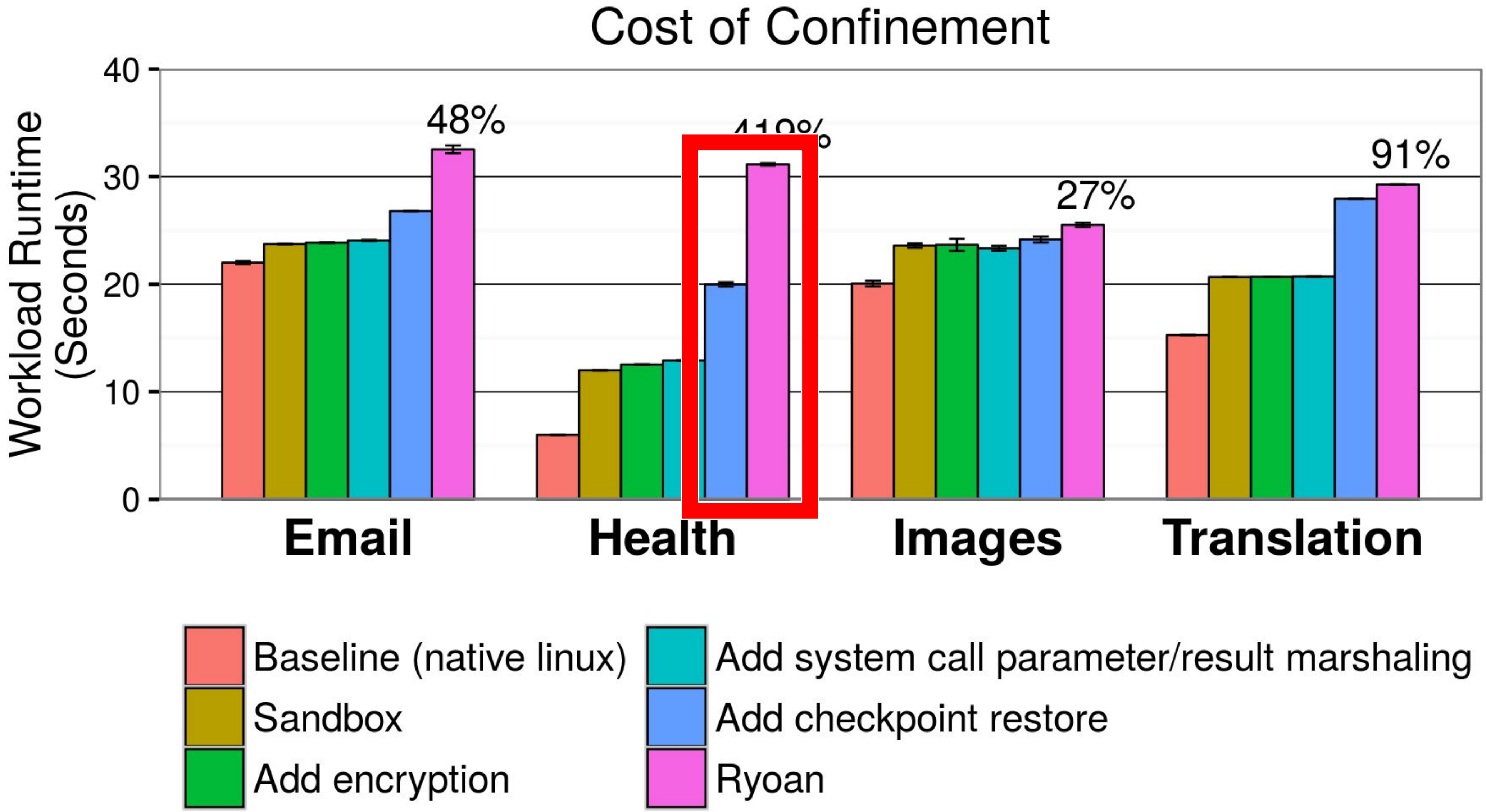
Health	20,000 1.4KB Boolean vectors from different users
Translation	30 short paragraphs, sizes 25-300B, 4.1KB total
Images	12 images, sizes 17KB-613KB
Email	250 emails, 30% with 103KB-12MB attachment



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

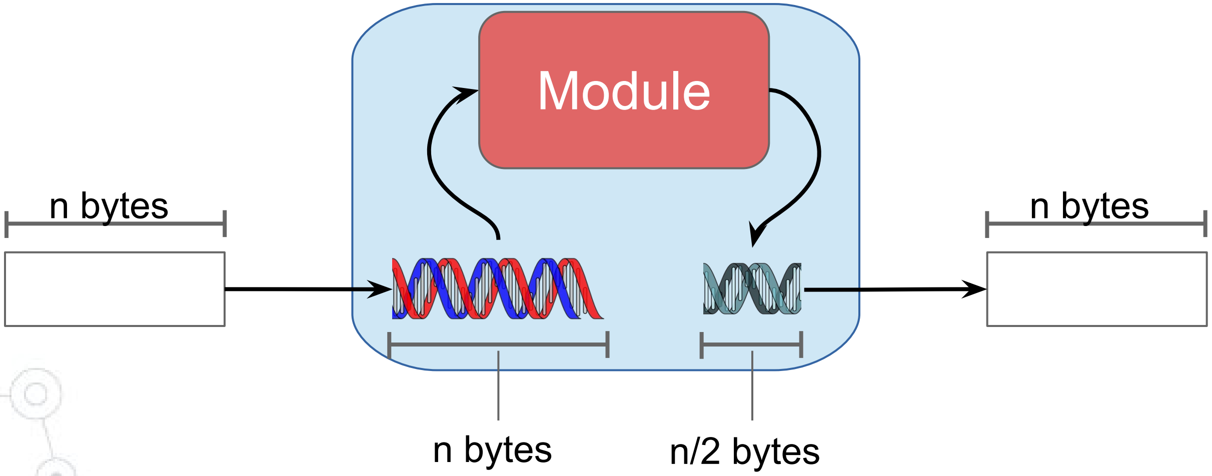
- Allows untrusted code to operate on secret data on untrusted platforms

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

(Backup Slides Follow)

Output Size

- Output Size is a (configurable) fixed function of input size.
  - Output is padded or truncated by Ryoan
  - Always predefined in the specification
  - Examples (n bytes of input)
    - Virus Scanner output: n bytes + 1 bit
    - Machine Translation output: 2n bytes


Module
n bytes
n bytes
n bytes
n/2 bytes