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

23andMe

Classification Result

Disease Risk

Amazon Web Services

Patient
Disease risk assessment: Trust issues
Disease risk assessment: Trust issues
Disease risk assessment: Trust issues
Disease risk assessment: Trust issues

Disease Risk Classification Result

23andMe

Classification Result

Disease Risk

NATIONAL SECURITY AGENCY
UNITED STATES OF AMERICA

amazon web services™
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 providers 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 providers 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 providers 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 providers 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

Sandbox
- 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

**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
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

Module
write([CIPHERTEXT]);

Sandbox
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

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

0101110

<table>
<thead>
<tr>
<th>8bytes</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>16bytes</td>
<td>1</td>
</tr>
</tbody>
</table>
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.

In-memory file API
- File system operations in memory
- Examples:
  - Temp files
  - Preexisting files

Replaced system calls:
- `mmap`

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.

In-memory file API
- File system operations in memory
- Examples:
  - Temp files
  - Preexisting files

Replaced system calls:
- mmap

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.

**In-memory file API**
- File system operations in memory
- Examples:
  - Temp files
  - Preexisting files

Replaced system calls: mmap

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

Parser Input
Classifier
Return Results

Translation

In: French text
Out: English text

Moses

Images

In: Pictures
Out: Array of objects

Recognize Face
Recognize Horse
Recognize NSFW

Combine
Distribute

Email

In: Emails
Out: Spam & virus status

Distribute
Recognize

Combine

OpenCV

Spam

ClamAV
Health

In: Genome/health data
Out: Disease risk

Parse Input → Classifier → Return Results

Translation

In: French text
Out: English text

Moses

Email

In: Emails
Out: Spam & virus status

Spam & virus status

Email → Combine → Distribute

Images

In: Pictures
Out: Array of objects

Recognize Face → Combine → Distribute

Recognize Horse → Combine → Distribute

Recognize NSFW → Combine → Distribute

OpenCV

Distribute

Recognize NSFW

Recognize Horse

Recognize Face

In: Pictures
Out: Array of objects
**Health**

- **In:** Genome/health data
- **Out:** Disease risk

**Translation**

- **In:** French text
- **Out:** English text

**Images**

- **In:** Pictures
- **Out:** Array of objects

- **Recognize Horse**
- **Recognize Face**
- **Recognize NSFW**

**Email**

- **In:** Emails
- **Out:** Spam & virus status

- **Distribute**
- **Combine**

**Distribute**

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pictures</td>
<td>Array of objects</td>
</tr>
</tbody>
</table>

**Combine**

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emails</td>
<td>Spam &amp; virus status</td>
</tr>
</tbody>
</table>
Health

- Parse Input
- Classifier
- Return Results

- In: Genome/health data
- Out: Disease risk

Translation

- Moses

- In: French text
- Out: English text

Images

- Recognize Face
- Recognize Horse
- Recognize NSFW

- Distribute
- Combine

- In: Pictures
- Out: Array of objects

Email

- Distribute
- Combine

- 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
Cost of Confinement

<table>
<thead>
<tr>
<th>Workload Runtime (Seconds)</th>
<th>Email</th>
<th>Health</th>
<th>Images</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (native linux)</td>
<td>🟥</td>
<td>🫖</td>
<td>🟩</td>
<td>🟪</td>
</tr>
<tr>
<td>Add system call parameter/result marshaling</td>
<td>↑</td>
<td>↑</td>
<td>↓</td>
<td>↑</td>
</tr>
<tr>
<td>Sandbox</td>
<td>🟨</td>
<td>🟪</td>
<td>🟩</td>
<td>🟪</td>
</tr>
<tr>
<td>Add checkpoint restore</td>
<td>🟪</td>
<td>🟩</td>
<td>🟪</td>
<td>🟩</td>
</tr>
<tr>
<td>Add encryption</td>
<td>🟩</td>
<td>🟪</td>
<td>🟩</td>
<td>🟪</td>
</tr>
<tr>
<td>Ryoan</td>
<td>🟪</td>
<td>🟩</td>
<td>🟪</td>
<td>🟪</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Health</td>
</tr>
<tr>
<td>20,000 1.4KB Boolean vectors from different users</td>
</tr>
<tr>
<td>Translation</td>
</tr>
<tr>
<td>30 short paragraphs, sizes 25-300B, 4.1KB total</td>
</tr>
<tr>
<td>Images</td>
</tr>
<tr>
<td>12 images, sizes 17KB-613KB</td>
</tr>
<tr>
<td>Email</td>
</tr>
<tr>
<td>250 emails, 30% with 103KB-12MB attachment</td>
</tr>
</tbody>
</table>
Cost of Confinement

**Workload Runtime (Seconds)**

- **Email**: 48%
- **Health**: 419%
- **Images**: 27%
- **Translation**: 91%

**Baseline** (native linux)
- Add system call parameter/result marshaling
- Add checkpoint restore
- Add encryption
- Ryoan

**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
Cost of Confinement

<table>
<thead>
<tr>
<th>Workload</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>20,000 1.4KB Boolean vectors from different users</td>
</tr>
<tr>
<td>Health</td>
<td>48%</td>
</tr>
<tr>
<td>Images</td>
<td>419%</td>
</tr>
<tr>
<td>Translation</td>
<td>27%</td>
</tr>
<tr>
<td></td>
<td>91%</td>
</tr>
</tbody>
</table>

- Baseline (native linux)
- Add system call parameter/result marshaling
- Sandbox
- Add checkpoint restore
- Add encryption
- Ryoan

<table>
<thead>
<tr>
<th>Health</th>
<th>20,000 1.4KB Boolean vectors from different users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td>30 short paragraphs, sizes 25-300B, 4.1KB total</td>
</tr>
<tr>
<td>Images</td>
<td>12 images, sizes 17KB-613KB</td>
</tr>
<tr>
<td>Email</td>
<td>250 emails, 30% with 103KB-12MB attachment</td>
</tr>
</tbody>
</table>
Cost of Confinement

<table>
<thead>
<tr>
<th>Workload Runtime (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Email</strong></td>
</tr>
<tr>
<td><strong>Health</strong></td>
</tr>
<tr>
<td><strong>Images</strong></td>
</tr>
<tr>
<td><strong>Translation</strong></td>
</tr>
</tbody>
</table>

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

Legend:
- Baseline (native linux)
- Sandbox
- Add encryption
- Add system call parameter/result marshaling
- Add checkpoint restore
- Ryoan

Bar Chart:
- 48% increase in workload runtime for Health
- 410% increase in workload runtime for Health
- 27% increase in workload runtime for Images
- 91% increase in workload runtime for Translation
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
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

---

![Diagram showing the process of input and output sizes](image-url)