# Paper Review Form (1000 words maximum)
# *cs940*: Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data [1]

February 26, 2018

## Paper Summary

Ryoan is an untrusted code containment system for secure remote distributed processing of sensitive data. Taking advantage of existing containment capabilities of Intel's hardware Software Guard Extensions (SGX) and Google's Native Client (NaCl), Ryoan provides additional protections by further sandboxing untrusted code running on NaCl within SGX enclaves. Ryoan enforces label-based taint tracking, data obliviousness in transmission, and additional confinements on NaCl modules to prevent both direct and side-channel leakage of sensitive data. Ryoan instances can be optimised for either one-time or checkpoint-based lifecycles based on workload specifics. Taking into account a range of realistic use cases, a functional prototype was implemented and tested, achieving reasonable performance.

## Pros and Cons

I believe that the paper has the following positive features:

- Ryoan's two-way protection between untrusted operating systems and untrusted code provides somewhat stronger assurances on the code's perspective than typical hypervisors, while simultaneously producing reduced overhead by leveraging hardware-assisted sandboxing instead of virtualisation.
- The prototype's use of public keys as users' tags provides convenient authentication of labels without creating additional overhead by storing the public keys separately.
- Building on a new hardware feature (SGX) with limited consumer-grade hardware support at the time, the authors performed very comprehensive benchmarks with limited assumptions and simulations implemented.

I believe that the paper has the following negative features:

- The usefulness of Ryoan's auditing trails are unclear, due to their lack of insight into the untrusted code's correctness. This may well be an unnecessary overhead between checkpoint-based lifecycles.

- Ryoan's maximum 1GB in-memory file system is inefficient and constraining. With the eradication of forward-able information, it can be very difficult to separate linearly-dependent data into segments fitting the limit.
- In the case of 23andMe, a realistic user of the service cannot generate their data label easily without relying on 23andMe to delegate the label's computations, as in-browser computations and storage can be easily compromised and non-persistent; this significantly reduces the security assurances provided to the user.

## The Problem/Motivation

A wide range of remote data-processing services deal with sensitive user data, such as in image processing, tax preparation, and health analysis. It is impossible for the user of a service to trust or control the actions performed by untrusted remote code. Existing solutions leveraging enclave-protected computation [2, 3] could not prevent intentional or unintentional side-channel information leakage. With the presence of untrusted code (not modelled by previous systems), it is significantly easier for data to be leaked via covert channels such as modulated syscalls. Alternative hardware and TPM-based solutions [4] suffered from limited execution capabilities. Therefore, additional techniques were required to confine untrusted code.

## The Solution/Approach

To achieve full confinement of untrusted code, Ryoan is built on two layers of existing security architectures: SGX, which provides hardware-enforced module memory protection and unforgeable authentication of initial code; and NaCl, which provides memory access and syscall safety as a standard sandbox. The additional protections provided by Ryoan (which operates as an extension to NaCl) are imposed due to side-channel security, performance, and compatibility considerations. Modules running within Ryoan are arranged in directed acyclic graphs (DAGs) to enable a chain of authenticated initialisations, as well as to provide an audit trail and enforce one-time data processing. Label-based taint tracking is used to ensure mandatory confinement when untrusted code gains access to user data. Results of computations are dynamically uniformised to prevent side-channel modulation.

Individually, each module will always be reinitialised or reset by Ryoan to a fresh checkpoint between executions. The module's interactions with the OS and other modules are strictly sanitised by Ryoan. Barring some hardware limitations in fault handling and possible timing channel leaks, these measures protect any data-derived information from being leaked by the untrusted code. Unconfined module-customisable initialisation and POSIX in-memory file system APIs provide compatibility and optimise optimisations when executing existing code.

## Evaluation

At the time of study, Ryoan's required SGX capabilities were not present in a majority of consumer-grade computing devices. Therefore the authors had to simulate some of Ryoan's functionalities with QEMU, operating on a Skylake laptop with only partial support for SGX 2.0 capabilities. A wide range of realistic computation workloads inspired from use-cases considered by the authors were tested on this platform. Among these workloads, Ryoan mostly exhibited a reasonable level of overhead (27%-91%) when compared with unsecured execution, and even less overhead when compared with existing sandbox solutions offering lesser levels of protection. However, it was also noted that Ryoan will perform significantly worse on high page fault or high setup cost workloads, as demonstrated through the gene analysis simulation experiment.

It was also demonstrated that SGX itself incurred low levels of overhead, and that the choice between reinitialisation and checkpoint restoration should be made based on the workload's characteristics to maximise performance.

## Your Opinion

From an engineering perspective, Ryoan was an excellent solution put together by leveraging security properties of both software and hardware mechanisms. A comprehensive protection from untrusted code could not have been achieved without all components. However, as the authors themselves mentioned, untrusted code confinement remains very challenging problem. While Ryoan certainly represented the state-of-art protection at the time of publication, various limitations in hardware and software can still result in side-channel information leakage, such as non-memory fault handling and timing channel modulations. GhostRider [5] leveraging a specialised hardware-software solution is significantly more resistant to these side-channel attacks, but incur high levels of performance cost the authors have deemed unacceptable.

Not long after the publication of this paper, Google deprecated NaCl and replaced it with WebAssembly [6], meaning that this solution needs substantial updates to stay current, in the face of its rapidly shifting backbones.

## Questions for the Authors

- Given the recently prototyped Spectre attacks against SGX enclaves [7], what effects do these attacks have on Ryoan's security assurances?
- As Google is moving away from NaCl towards WebAssembly, what changes in Ryoan have you made or can be made to adapt for the lack of future NaCl support?

# References


[1] T. Hunt, Z. Zhu, Y. Xu, S. Peter, and E. Witchel, "Ryoan: A distributed sandbox for untrusted computation on secret data." in *OSDI*, 2016, pp. 533–549.

[2] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. Stillwell *et al.*, "Scone: Secure linux containers with intel sgx." in *OSDI*, vol. 16, 2016, pp. 689–703.

[3] A. Baumann, M. Peinado, and G. Hunt, "Shielding applications from an untrusted cloud with haven," *ACM Transactions on Computer Systems (TOCS)*, vol. 33, no. 3, p. 8, 2015.

[4] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and H. Isozaki, "An execution infrastructure for tcb minimization," 2007.

[5] C. Liu, A. Harris, M. Maas, M. Hicks, M. Tiwari, and E. Shi, "Ghostrider: A hardware-software system for memory trace oblivious computation," *ACM SIGPLAN Notices*, vol. 50, no. 4, pp. 87–101, 2015.

[6] Chromium Blog, "Goodbye PNaCl, hello WebAssembly!" May 2017. [Online]. Available: https://blog.chromium.org/2017/05/goodbye-pnacl-hello-webassembly.html

[7] D. O'Keeffe *et al.*, "Sample code demonstrating a spectre-like attack against an intel sgx enclave." January 2018. [Online]. Available: https://github.com/lsds/spectre-attack-sgx