Rapid Deployment of Confidential Cloud Applications with Gramine

Michał Kowalczyk
Invisible Things Lab
Switzerland
mkow@invisiblethingslab.com

Dmitrii Kuvaiskii
Intel
Germany
dmitrii.kuvaiskii@gmail.com

Paweł Marczewski
Invisible Things Lab
Poland
pawel@invisiblethingslab.com

Borys Popławski
Invisible Things Lab
Poland
borysp@invisiblethingslab.com

Wojtek Porczyk
Invisible Things Lab
Poland
woju@invisiblethingslab.com

Donald E. Porter
UNC — Chapel Hill
USA
porter@cs.unc.edu

Kailun Qin
Intel
China
kailun.qin@intel.com

Chia-Che Tsai
Texas A&M University
USA
chiache@tamu.edu

Mona Vij
Intel
USA
mona.vij@intel.com

Isaku Yamahata
Intel
USA
isaku.yamahata@intel.com

ABSTRACT

Gramine is a leading open-source tool for securely porting unmodified Linux applications onto Intel® SGX [67]. Gramine implements the "lift-and-shift" model of confidential computing—where one simply runs an entire, legacy application in a trusted execution environment (TEE), such as Intel SGX. Gramine was initially released as an artifact of the 2014 EuroSys Paper [96], and over the subsequent ten years has evolved into a production-ready, open-source community project, deployed commercially by multiple companies. Gramine has been *used* by over one hundred peer-reviewed papers to facilitate confidential computing research: in some cases, it is the benchmark against which other systems or TEE design choices are measured; as a baseline for security analysis or attacks and, in others, a building block for quick prototyping of applications in domains including health care, machine learning, genome analysis, speech processing, networking, autonomous vehicles, IoT management, and database systems. This short paper describes the Gramine project and its impact on industry and research.

1 INTRODUCTION

Gramine is an open-source tools for securely deploying unmodified Linux applications on Intel® SGX [67] and other trusted execution environments (TEEs). Gramine implements the "lift-and-shift" model of confidential computing—where one simply runs an entire, legacy application in the TEE. Figure 1 overviews the Gramine architecture [96] and how it is deployed in SGX, specifically. Gramine reimplements the Linux system call table in a user space library (hence the name **library OS**). The heavy black box indicates the boundary of the trusted code running inside the TEE (enclave). At the top is an unmodified application binary, followed by supporting libraries. Some system calls are implemented entirely within the Library OS, and some request (and dynamically check) functionality from the untrusted host kernel. A rare feature of Gramine is multi-process support, including `fork()`. Multiple processes are implemented using multiple separate host address spaces and passing encrypted, signed messages between enclaves. Although message passing in user-space is slower than an in-kernel implementation of inter-process communication primitives (IPC), this design choice avoids placing functionality and trust in the host kernel.

Gramine is designed for **Platform Independence**; it is implemented against a small, simple ABI (fewer than 50 calls) called the **Platform Adaptation Layer (PAL)**. The PAL ABI includes abstractions for memory, threads, files, and devices. In principle, if one reimplements the PAL ABI on a new host, the rest of Gramine and supported applications should "just work."

Although Gramine does not require application code changes or recompilation, each application does need a signed **manifest** file to specify what data should be loaded into the enclave. The manifest includes hashes of trusted files, as well as other virtualization parameters.

For running on an untrusted host, as in the SGX threat model, Gramine adds a **shielding layer** to the TEE interface (i.e., `ecalls` on SGX). Although there are minor differences between the enclave and PAL ABI, we expect these to consolidate in the future.

Gramine is compatible with multiple remote attestation frameworks. In the past, Gramine has supported Intel Attestation Service (IAS), and several protocols based on Intel Data Center Attestation Primitives (DCAP). As IAS is reaching end of life in 2025, Gramine is retiring IAS support at the time of writing.

We welcome and encourage contributions of useful changes to Gramine itself. Gramine is licensed under the LGPL, which does require users publish modifications to Gramine itself. However, our understanding in adopting the LGPL is that anyone is free to

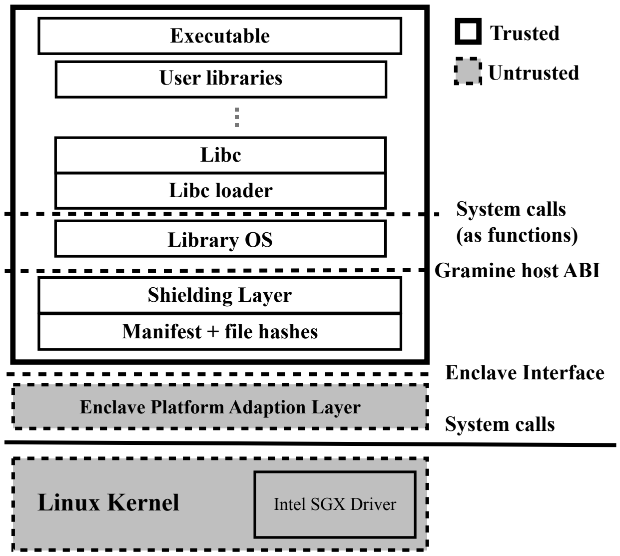

Figure 1: The Gramine architecture. The executable is position-dependent. The enclave includes an OS shield, a library OS, Glibc, and other user binaries.

use Gramine without sharing any code outside of Gramine; the obligation to share code stops at the boundary of the library OS. For instance, using unmodified Gramine to deploy a closed-source application incurs no obligation to share source code, nor does it prevent use with code under a different license.

1.1 Container Integration

In order to facilitate application deployment on Gramine, Gramine includes two tools to automate construction of containers that use Gramine to run on SGX.

The first feature, called **Gramine Shielded Containers** (**gsc**), automatically converts a Docker image into a version of the image that can run in Gramine on SGX. Containers, such as Docker, are now a very common way to package and deploy software. A dockerfile, which describes how to build an application image; a Gramine manifest has a similar goal of describing the runtime requirements of an application, and is perhaps one of the most challenging steps in adopting Gramine.

GSC automatically generates a Gramine manifest from a docker image, and then creates a new Docker image that includes the manifest, Gramine, and other required SGX runtime support. GSC also has the ability to sign the manifest and run the container in SGX.

The second tool is called **Scaffolding for Gramine (SCAG)**, and leverages a language runtime or other framework to generate a Docker image with Gramine. Unlike GSC, SCAG does not require an initial Docker Image. SCAG is currently compatible with Python, Flask, Node.js, Express.js, Koa.js, Java JAR files, Java with the Gradle build system, and .NET. SCAG also automatically adds SSL/TLS to plaintext networking apps.

These two tools indicate that most of the information that Gramine requires to run an application is already present in most deployment tools.

2 GRAMINE'S IMPACT

Gramine has been a key enabling technology for confidential computing research and practice to-date. Specifically, Gramine (1) is the first open-source, lift-and-shift framework; (2) is mature enough to run a significant number of real-world applications; (3) has a broad user base; and (4) has a growing contributor community. Gramine has been *used* by over one hundred peer-reviewed papers to facilitate confidential computing research: in some cases, it is the benchmark against which other systems or TEE design choices are measured [13, 19, 22, 27, 32, 34, 37, 47, 48, 50, 60, 83, 84, 90, 91, 94]; a baseline for security analysis or attacks [3, 4, 17, 18, 23, 24, 39, 49, 56, 63, 72, 81, 88, 95, 98, 99, 103]; and, in others, a building block for quick prototyping of applications in domains including health care, machine learning, genome analysis, speech processing, networking, autonomous vehicles, IoT management, and database systems [2, 5, 6, 12, 14–16, 25, 26, 33, 36, 38, 40–43, 45, 51–53, 57–59, 61, 62, 64–66, 68, 71, 74, 77–79, 85–87, 93, 97, 100–102, 104–108].

On the industry side, Gramine had its first production release in late 2021 and we are aware of *at least six companies* developing products built on Gramine. We keep a list of Gramine users online [1]. One example of Gramine's utility is IBM's ePrescription system, which is used by the German Ministry of Health to track prescriptions across the entire German population. Several cloud service providers have built solutions for SGX involving Gramine, including Tencent, JD Cloud, ByteDance, and Microsoft. Several start-ups use Gramine in their solutions; for instance, Edgeless [92] uses Gramine to secure blockchain infrastructure for banks. Gramine is used internally at Intel for validating and benchmarking SGX CPUs, and for internal services.

One major class of applications where Gramine has gained traction is in secure machine learning infrastructures. The OpenFL federated learning project has been built using Gramine [85]. The BigDL large language model framework integrates standard machine learning tools, such as Tensorflow and Pytorch, with system security features such as SGX and Gramine [9]. Simiarly, the OpenVINO machine learning toolkit for AI inference includes a set of security add-ons that also include SGX support, facilitated by Gramine [20]. ByteDance's support for securing Federated Learning on SGX in their Fedlearner project is also built upon Gramine [29].

Researchers and developers from Intel and Invisible Things Lab have joined the leadership team for the project, and Gramine has received significant code contributions from companies including IBM and Alibaba Cloud. Gramine has been adopted by the Linux Foundation's Confidential Computing Consortium (CCC), which helps the project with infrastructure, visibility, and mentorship to build a robust developer community. Gramine is the only project to go from the incubation to graduation stage [1]. Gramine is also the only CCC project to date to receive the OpenSSF's Security Best Practices Badge [76]. Gramine is a featured open-source enabler in Microsoft's Azure Confidential Cloud Services [70].

[1]https://gramine.readthedocs.io/en/latest/gramine-users.html

The overall maturity and utility of the project is also reflected in various metadata statistics. On github at the time of writing, the code has 607 stars, 201 forks, and 27 watchers. The original Graphene repository, now archived, still has 771 stars, 260 forks, and 51 watchers. The ATC paper describing SGX support in Gramine has 714 citations.

2.1 Evolving with SGX

SGX has evolved substantially since its first version, which made a number of simplifying assumptions about the workloads. For instance, the first version of SGX did not allow the program to dynamically create new mappings inside of a host enclave. This is a simplification not just for implementation, but reasoning about security. For instance, the intersection of exception handling and scheduling controlled by an untrusted host require careful reasoning about race conditions in how the TEE code is invoked; the recent AEX-Notify feature in SGX supports more efficient exception handling and can be used to help enclave code defend itself from side-channel attacks that leverage interrupts [21].

Gramine has both ongoing and completed efforts to stay abreast of new SGX versions. In the case of dynamic memory management, Gramine has basic support for the Enclave Dynamic Memory Management (EDMM) feature, and work is in progress to optimize performance of dynamic mappings in Gramine. Work is in progress to merge AEX-Notify support. Gramine has also merged support for using `ioctl` to communicate with computational accelerators in order to offload computation [54].

2.2 TEEs Beyond SGX

Although the Gramine project has focused supporting applications on SGX, the original design of Gramine predates SGX. Gramine was originally designed to simplify *porting* code from one platform to another. Gramine has a modular architecture that encapsulates most host-specific (or TEE-specific) code into a platform adaptation layer (PAL), which is designed for ease of implementation. In principle, to support a new TEE, one only need to implement a suitable PAL for the new TEE. Even if SGX were to fall out of usage, Gramine can be viewed as a general-purpose Linux compatibility layer, which can be still useful in deploying legacy Linux code on future TEE platforms.

For instance, in recent years, the confidential VM abstraction has grown in popularity, and is a better fit than the enclave model for some use cases. Intel's new TDX feature follows the confidential VM model.

A recent CCS paper [55] describes how Gramine has been ported to run applications on Intel TDX. The core Gramine library OS was not modified for TDX, only a TDX-specific PAL need be implemented, totaling about 17 kLoC. This PAL looked very different than the SGX PAL, implementing I/O abstractions over simple virtio drivers on TDX, whereas the SGX PAL implements these abstractions on host OS file handles. Gramine on TDX retains compatibility with many Linux applications, but an order-of-magnitude smaller TCB and an order of magnitude fewer inputs to check than running Linux inside of a confidential VM on TDX.

Gramine on TDX is currently an experimental feature, which we intend to continue maturing. We also note that IBM has also contributed patches to port Gramine to the IBM Power architecture. These porting efforts show both the generality of Gramine, and engineering investments in the core Gramine library OS may accelerate research and development future TEEs and other hardware platforms.

3 SIMILAR PROJECTS

Gramine shares much of its design and modular architecture with Drawbridge [80], a similar library OS created by refactoring the Windows kernel. The Haven library OS was the first project to demonstrate the power of a library OS for portability in a TEE [11], built upon Drawbridge. Drawbridge is now in production in Microsoft Azure cloud [30]. Haven was the inspiration to port Gramine to SGX, effectively creating an open-source alternative to closed-source Haven for researchers and developers.

Since Haven and Gramine, several other projects have created lift-and-shift frameworks. Key differentiating features include whether the project is open-source, the implementation language, and whether the application binaries must be recompiled. Within Linux/Unix compatibility layers, another key difference is whether `fork()` is supported, as well as whether processes run in one or multiple address spaces.

Gramine is implemented in C (Gramine predates the Rust language) and does not require recompilation. Gramine supports `fork()` and runs processes in separate address spaces, except when run on Intel's TDX, which is still in development. Gramine implements about 170 out of roughly 360 Linux system calls. Gramine also supports portions of the `/dev`, `/proc`, and `/sys` pseudo-file systems.

Occlum [91] (open-source) is a lightweight library OS that runs legacy code inside an enclave, but requires recompilation. Unlike Gramine, which implements multiple processes using multiple address spaces, Occlum uses Intel's Memory Protection eXtension (MPX) [35] hardware or Software Fault Isolation (SFI) to isolate regions within a single, shared address space. Occlum's library OS is the first libOS in Rust, and has implemented 200 out of 325 Linux system calls.

Scone [7] (closed-source) is another library OS that runs legacy Linux application binaries inside enclaves. Unlike Gramine, which virtualizes at the system call table, Scone virtualizes and shields at the C library interface, and requires cross-compilation. Scone also now advertises support for forking new processes.

Mystikos [73] (open-source) is a porting framework for confidential computing that packs each application with its libraries, a file-system image, and a library OS, into a single, encrypted image. The Mystikos kernel has implemented 95 Linux system calls, including experimental support for forking.

SGX-LKL [81] (open-source) is a port of the Linux Kernel Library (LKL) [82] as an open-source library OS to run inside enclaves. SGX-LKL reuses portions of the existing Linux source code (sometimes called a Rump kernel [46]), facilitating bug-for-bug compatibility. SGX-LKL does not support fork, multi-processing, or inter-process communication, since LKL is designed to be loaded into a single address space. For IO, SGX-LKL operates on raw disks and network devices; running the complete IO stacks in the enclave. SGX-LKL includes an *oblivious external I/O* feature to hide sensitive I/O patterns.

Development Frameworks. Outside of the "lift-and-shift" model, a number of frameworks and software development kits (SDKs) include tools for writing *new* application software for TEEs. Intel provides an Intel's SDK and Platform Software (PSW) [44] for writing SGX in C/C++ applications. Open Enclave SDK [75] is another SDK which aims to be universal for multiple platforms, currently supporting Intel SGX, with preview support for OP-TEE OS and ARM TrustZone.

Apache Teaclave [10] and Fortanix's EPD [31] are frameworks for writing TEE code in Rust. More recent development frameworks target portability across multiple TEE platforms, including Microsoft Confidential Consortium Framework [69], Google Asylo [8], and Enarx [28].

SGX-RA-TLS [89] is a communication framework that simplifies SGX application development by integrating SGX remote attestation into the Transport Layer Security (TLS) protocol. The SGX remote attestation requires remote entities to verify the certificates signed by the target enclaves, with an Intel-owned or cloud-owed attestation service. SGX-RA-TLS further embeds the secrets negotiated during TLS handshake into the certificates attestation, to prevent person-in-the-middle attacks, and can transparently do so in Gramine, SGX-LKL, or Scone.

4 CONCLUSION

Gramine has matured over ten years from a research prototype to a production-quality framework for deploying unmodified application binaries on SGX. Gramine has helped over a hundred research projects build SGX prototypes more quickly, and is in production use by multiple start-ups and cloud service providers. We expect Gramine to continue to add missing system calls and other OS features over time, we hope to completely match Linux's features for unprivileged user applications in future work. Early experience with TDX indicates that Gramine is likely to be useful on additional TEEs.

ACKNOWLEDGMENTS

Many people have contributed code to Gramine since its inception in 2011 — too many to list in this document — including Nehal Bandi, Bill Jannen, Vrushali Kulkarni, Bhushan Jain, Gurpreet Chadha, Jitin John, Kumar Saurabh Arora, Manikantan Subramanian, Naveen Kalaskar, Thomas Mathew, Anchal Agarwal, Amit Arya, Sourabh Yerfule, Ujwala Tulshigiri, Imran Brown, Rajendra Kumar, Jia Zhang, Li Xun, Rafał Wojdyła, Stefan Berger, Vijay Dhanraj, Anjo Vahldiek-Oberwagner, Mariusz Zaborski, Maja Kądziołka, Anees Sahib, Sankaranarayanan Venkatasubramanian, and Jitender Kumar. Gramine was designed and implemented in part while Porter and Tsai were at Stony Brook University and while Tsai was at UC Berkeley. We give especial thanks to David Cowhig for system administration support. Gramine development has been funded by NSF awards CNS-1149229, CNS-1228839, CNS-1700512, and CNS-2244937. We are grateful for infrastructure and hardware support from the Linux Foundation's Confidential Computing Consortium and Intel. Porter has a significant financial interest in Fortanix and Anjuna Security.

REFERENCES

[1] 2024. Linux Foundation Confidential Computing Consortium: Project Progression Policy. https://github.com/confidential-computing/governance/blob/main/project-progression-policy.md#graduation-stage.

[2] Alejandro Cabrera Aldaya and Billy Bob Brumley. 2020. Online template attacks: Revisited. *arXiv preprint arXiv:2007.05337* (2020).

[3] Alejandro Cabrera Aldaya, Cesar Pereida García, and Billy Bob Brumley. 2020. From A to Z: Projective coordinates leakage in the wild. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), 428–453.

[4] Fritz Alder, Jo Van Bulck, David Oswald, and Frank Piessens. 2020. Faulty point unit: Abi poisoning attacks on intel sgx. In *Annual Computer Security Applications Conference*. 415–427.

[5] Robin Ankele. 2020. *Addressing syntactic privacy for privacy-preserving data analysis and data release*. Ph. D. Dissertation. University of Oxford.

[6] Sam Ansmink. 2021. *Encrypted Query Processing in DuckDB*. Ph. D. Dissertation. Universiteit van Amsterdam.

[7] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Daniel O'Keeffe, Mark L. Stillwell, David Goltzsche, Dave Eyers, Rüdiger Kapitza, Peter Pietzuch, and Christof Fetzer. 2016. SCONE: Secure Linux Containers with Intel SGX. In *OSDI*.

[8] Asylo 2018. Asylo. https://asylo.dev/.

[9] BigDL Authors. 2020. Privacy Preserving Machine Learning (PPML) on Azure User Guide. https://bigdl.readthedocs.io/en/latest/doc/PPML/Overview/azure_ppml.html#privacy-preserving-machine-learning-ppml-on-azure-user-guide.

[10] Baidu. 2019. Rust SGX SDK. https://github.com/apache/incubator-teaclave-sgx-sdk.

[11] Andrew Baumann, Marcus Peinado, and Galen Hunt. 2014. Shielding Applications from an Untrusted Cloud with Haven. In *OSDI*. 267–283.

[12] Sebastian P Bayerl, Ferdinand Brasser, Christoph Busch, Tommaso Frassetto, Patrick Jauernig, Jascha Kolberg, Andreas Nautsch, Korbinian Riedhammer, Ahmad-Reza Sadeghi, Thomas Schneider, et al. 2019. Privacy-preserving speech processing via STPC and TEEs. (2019).

[13] André Brandão, João S Resende, and Rolando Martins. 2021. Hardening cryptographic operations through the use of secure enclaves. *Computers & Security* 108 (2021), 102327.

[14] Ferdinand Brasser, Tommaso Frassetto, Korbinian Riedhammer, Ahmad-Reza Sadeghi, Thomas Schneider, and Christian Weinert. 2018. VoiceGuard: Secure and Private Speech Processing.. In *Interspeech*. 1303–1307.

[15] Cláudia Brito, Pedro Ferreira, Bernardo Portela, Rui Oliveira, and João Paulo. 2021. Soteria: Privacy-Preserving Machine Learning for Apache Spark. *Cryptology ePrint Archive* (2021).

[16] Johannes Buchmann, Ghada Dessouky, Tommaso Frassetto, Ágnes Kiss, Ahmad-Reza Sadeghi, Thomas Schneider, Giulia Traverso, and Shaza Zeitouni. 2020. SAFE: A Secure and Efficient Long-Term Distributed Storage System. In *Proceedings of the 8th International Workshop on Security in Blockchain and Cloud Computing*. 8–13.

[17] Guoxing Chen. 2019. *Exploitable Hardware Features and Vulnerabilities Enhanced Side-Channel Attacks on Intel SGX and Their Countermeasures*. The Ohio State University.

[18] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai. 2019. SgxPectre: Stealing Intel Secrets from SGX Enclaves Via Speculative Execution. In *2019 IEEE European Symposium on Security and Privacy (EuroS P)*. 142–157.

[19] Lixia Chen, Jian Li, Ruhui Ma, Haibing Guan, and Hans-Arno Jacobsen. 2019. EnclaveCache: A Secure and Scalable Key-Value Cache in Multi-Tenant Clouds Using Intel SGX. In *Proceedings of the 20th International Middleware Conference* (Davis, CA, USA) *(Middleware '19)*. Association for Computing Machinery, New York, NY, USA, 14–27. https://doi.org/10.1145/3361525.3361533

[20] Mruthunjaya Chetty and Jie Ren. 2022. Secured AI Model Inferencing at the Edge with Intel Developer Cloud for Edge Workloads. https://www.intel.com/content/www/us/en/developer/articles/technical/secured-ai-model-inferencing-at-the-edge.html.

[21] Scott Constable, Jo Van Bulck, Xiang Cheng, Yuan Xiao, Cedric Xing, Ilya Alexandrovich, Taesoo Kim, Frank Piessens, Mona Vij, and Mark Silberstein. 2023. AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 4051–4068. https://www.usenix.org/conference/usenixsecurity23/presentation/constable

[22] Jinhua Cui, Shweta Shinde, Satyaki Sen, Prateek Saxena, and Pinghai Yuan. 2021. Dynamic Binary Translation for SGX Enclaves. *arXiv preprint arXiv:2103.15289* (2021).

[23] Jinhua Cui, Jason Zhijingcheng Yu, Shweta Shinde, Prateek Saxena, and Zhiping Cai. 2021. SmashEx: Smashing SGX Enclaves Using Exceptions. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 779–793.

[24] Rongzhen Cui, Lianying Zhao, and David Lie. 2021. Emilia: Catching Iago in Legacy Code. In *Proceedings of the 2021 Symposium on Network and Distributed System Security (NDSS)*. https://security.csl.toronto.edu/wp-content/uploads/2021/01/rcui-ndss2021-emilia.pdf

[25] Poulami Das, Lisa Eckey, Tommaso Frassetto, David Gens, Kristina Hostáková, Patrick Jauernig, Sebastian Faust, and Ahmad-Reza Sadeghi. 2019. Fastkitten: Practical smart contracts on bitcoin. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 801–818.

[26] Omar Dib, Clément Huyart, and Khalifa Toumi. 2020. A novel data exploitation framework based on blockchain. *Pervasive and Mobile Computing* 61 (2020), 101104. https://doi.org/10.1016/j.pmcj.2019.101104

[27] Brandon D'Agostino and Omer Khan. 2021. Seeds of SEED: Characterizing Enclave-level Parallelism in Secure Multicore Processors. In *2021 International Symposium on Secure and Private Execution Environment Design (SEED)*. IEEE, 203–209.

[28] Enarx 2022. Enarx. https://enarx.dev/.

[29] Fedlearner 2024. Vertical Federated Learning. https://github.com/bytedance/fedlearner/tree/fix_dev_sgx/sgx.

[30] Mary Jo Foley. 2013. Microsoft to offer its 'Drawbridge' virtualization technology on top of its Windows Azure cloud. http://www.zdnet.com/article/microsoft-to-offer-its-drawbridge-virtualization-technology-on-top-of-its-windows-azure-cloud/.

[31] Fortanix. 2019. Fortanix Rust Enclave Development Platform. https://github.com/fortanix/rust-sgx.

[32] Anders Tungeland Gjerdrum, Håvard Dagenborg Johansen, Lars Brenna, and Dag Johansen. 2019. Diggi: A secure framework for hosting native cloud functions with minimal trust. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 18–27.

[33] Christian Göttel, Rafael Pires, Isabelly Rocha, Sébastien Vaucher, Pascal Felber, Marcelo Pasin, and Valerio Schiavoni. 2018. Security, performance and energy implications of hardware-assisted memory protection mechanisms on event-based streaming systems. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 264–266.

[34] Christian Göttel, Rafael Pires, Isabelly Rocha, Sébastien Vaucher, Pascal Felber, Marcelo Pasin, and Valerio Schiavoni. 2018. Security, performance and energy trade-offs of hardware-assisted memory protection mechanisms. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 133–142.

[35] Dave Hansen. 2016. Intel® Memory protection Extensions (Intel® MPX) for Linux. https://01.org/blogs/2016/intel-mpx-linux. (2016).

[36] Taylor Hardin and David Kotz. 2021. Amanuensis: Information provenance for health-data systems. *Information Processing & Management* 58, 2 (2021), 102460. https://doi.org/10.1016/j.ipm.2020.102460

[37] Aisha Hasan, Ryan Riley, and Dmitry Ponomarev. 2020. Port or Shim? Stress Testing Application Performance on Intel SGX. In *2020 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 123–133.

[38] Sohaib Ul Hassan, Iaroslav Gridin, Ignacio M Delgado-Lozano, Cesar Pereida García, Jesús-Javier Chi-Domínguez, Alejandro Cabrera Aldaya, and Billy Bob Brumley. 2020. Déjà Vu: Side-Channel Analysis of Mozilla's NSS. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1887–1902.

[39] W. He, W. Zhang, S. Das, and Y. Liu. 2018. SGXlinger: A New Side-Channel Attack Vector Based on Interrupt Latency Against Enclave Execution. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*. 108–114. https://doi.org/10.1109/ICCD.2018.00025

[40] Emil Hemdal and Eliot Roxbergh. 2020. Erlang SGX-Protecting Confidential Erlang Workloads with Intel SGX. (2020).

[41] Stephen Herwig, Christina Garman, and Dave Levin. 2020. Achieving Keyless CDNs with Conclaves. In *29th USENIX Security Symposium (USENIX Security 20)*. 735–751.

[42] Helge Hoff. 2018. *SecureCached. Secure caching with the Diggi framework.* Master's thesis. UiT Norges arktiske universitet.

[43] Bumjin Im, Fangfei Yang, Chia-Che Tsai, Michael LeMay, Anjo Vahldiek-Oberwagner, and Nathan Dautenhahn. 2021. The Endokernel: Fast, Secure, and Programmable Subprocess Virtualization. *arXiv preprint arXiv:2108.03705* (2021).

[44] Intel Corporation. 2016. Intel® Software Guard Extensions for Linux OS - Intel SGX SDK. https://github.com/01org/linux-sgx.

[45] Ibrahim Tariq Javed, Fares Alharbi, Tiziana Margaria, Noel Crespi, and Kashif Naseer Qureshi. 2021. PETchain: A Blockchain-Based Privacy Enhancing Technology. *IEEE Access* 9 (2021), 41129–41143.

[46] Antti Kantee. 2012. *Flexible Operating System Internals: The Design and Implementation of the Anykernel and Rump Kernels.* Doctoral thesis. School of Science. http://urn.fi/URN:ISBN:978-952-60-4917-5

[47] Vishal Karande. 2018. *Protecting User Applications Using Trusted Execution Environment.* The University of Texas at Dallas.

[48] Taehoon Kim, Joongun Park, Jaewook Woo, Seungheun Jeon, and Jaehyuk Huh. 2019. Shieldstore: Shielded in-memory key-value storage with SGX. In *Proceedings of the Fourteenth EuroSys Conference 2019*. 1–15.

[49] Robin Klusman, Derk Barten, and Gijs Hollestelle. 2019. Practical Implications of Graphene-SGX. (2019).

[50] Robert Krahn, Donald Dragoti, Franz Gregor, Do Le Quoc, Valerio Schiavoni, Pascal Felber, Clenimar Souza, Andrey Brito, and Christof Fetzer. 2020. TEEMon: A Continuous Performance Monitoring Framework for TEEs. In *Proceedings of the 21st International Middleware Conference* (Delft, Netherlands) *(Middleware '20)*. Association for Computing Machinery, New York, NY, USA, 178–192. https://doi.org/10.1145/3423211.3425677

[51] Kubilay Ahmet Küçük, David Grawrock, and Andrew Martin. 2019. Managing confidentiality leaks through private algorithms on Software Guard eXtensions (SGX) enclaves. *EURASIP Journal on Information Security* 2019, 1 (2019), 1–22.

[52] Sandeep Kumar and Smruti R Sarangi. 2021. SecureFS: A Secure File System for Intel SGX. In *24th International Symposium on Research in Attacks, Intrusions and Defenses*. 91–102.

[53] Dmitrii Kuvaiskii, Somnath Chakrabarti, and Mona Vij. 2018. Snort Intrusion Detection System with Intel Software Guard Extension (Intel SGX). *arXiv preprint arXiv:1802.00508* (2018).

[54] Dmitrii Kuvaiskii, Gaurav Kumar, and Mona Vij. 2022. Computation offloading to hardware accelerators in Intel SGX and Gramine Library OS. arXiv:2203.01813 [cs.CR] https://arxiv.org/abs/2203.01813

[55] Dmitrii Kuvaiskii, Dimitrios Stavrakakis, Kailun Qin, Cedric Xing, Pramod Bhatotia, and Mona Vij. 2024. Gramine-TDX: A Lightweight OS Kernel for Confidential VMs. In *CCS*.

[56] Dayeol Lee, Dongha Jung, Ian T Fang, Chia-Che Tsai, and Raluca Ada Popa. 2020. An Off-Chip Attack on Hardware Enclaves via the Memory Bus. In *29th USENIX Security Symposium (USENIX Security 20)*.

[57] Dayeol Lee, Dmitrii Kuvaiskii, Anjo Vahldiek-Oberwagner, and Mona Vij. 2020. Privacy-Preserving Machine Learning in Untrusted Clouds Made Simple. https://doi.org/10.48550/ARXIV.2009.04390

[58] Jehyun Lee, Min Suk Kang, Dinil Mon Divakaran, Phyo May Thet, Videet Singhai, and Jun Seung You. 2022. A Step Towards On-Path Security Function Outsourcing. In *23rd International Conference on Distributed Computing and Networking*. 175–187.

[59] Carl Leijonberg. 2021. The Viability of Using Trusted Execution Environments to Protect Data in Node-RED: A study on using AMD-SEV and Intel SGX to protect sensitive data when Node-RED is deployed on the cloud.

[60] Joshua David Lind. 2019. Securing applications using trusted execution environments. (2019).

[61] Bingyu Liu, Shangyu Xie, and Yuan Hong. 2021. Efficient and Private Divisible Double Auction in Trusted Execution Environment. In *EAI International Conference on Applied Cryptography in Computer and Communications*. Springer, 75–92.

[62] Bingyu Liu, Shangyu Xie, Yuanzhou Yang, Rujia Wang, and Yuan Hong. 2021. Privacy preserving divisible double auction with a hybridized TEE-blockchain system. *Cybersecurity* 4, 1 (2021), 1–14.

[63] Weijie Liu, Hongbo Chen, XiaoFeng Wang, Zhi Li, Danfeng Zhang, Wenhao Wang, and Haixu Tang. 2021. Understanding TEE Containers, Easy to Use? Hard to Trust. *arXiv preprint arXiv:2109.01923* (2021).

[64] Ximing Liu, Wenwen Wang, Lizhi Wang, Xiaoli Gong, Ziyi Zhao, and Pen-Chung Yew. 2020. Regaining Lost Seconds: Efficient Page Preloading for SGX Enclaves. In *Proceedings of the 21st International Middleware Conference*. 326–340.

[65] Enio Marku, Gergely Biczók, and Colin Boyd. 2021. SafeLib: a practical library for outsourcing stateful network functions securely. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 244–252.

[66] Miti Mazmudar and Ian Goldberg. 2020. Mitigator: Privacy policy compliance using trusted hardware. *Proceedings on Privacy Enhancing Technologies* 2020, 3 (2020), 204–221.

[67] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. 2013. Innovative Instructions and Software Model for Isolated Execution. In *HASP*. ACM. http://doi.acm.org/10.1145/2487726.2488368

[68] Marcela S Melara, Michael J Freedman, and Mic Bowman. 2019. EnclaveDom: Privilege separation for large-TCB applications in trusted execution environments. *arXiv preprint arXiv:1907.13245* (2019).

[69] Microsoft. 2019. Microsoft Confidential Consortium Framework (CCF). https://www.microsoft.com/en-us/research/project/confidential-consortium-framework/.

[70] Microsoft. 2023. Confidential containers on Azure Kubernetes Service(AKS) with Intel SGX enclaves. https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-containers-enclaves.

[71] Marina Minkin and Mark Silberstein. 2019. *Improving Performance and Security of Intel SGX.* Ph. D. Dissertation. Computer Science Department, Technion.

[72] Kit Murdock, David Oswald, Flavio D Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. 2020. Plundervolt: Software-based fault injection attacks against

Intel SGX. In *2020 IEEE Symposium on Security and Privacy (SP)*.

[73] Mystikos 2022. Mystikos. https://github.com/deislabs/mystikos.

[74] Phillip Nguyen, Alex Silence, David Darais, and Joseph P Near. 2020. Duetsgx: Differential privacy with secure hardware. *arXiv preprint arXiv:2010.10664* (2020).

[75] Open Enclave 2018. Open Enclave SDK. https://github.com/openenclave/openenclave.

[76] OpenSSF. 2024. OpenSSF Best Practices - The Gramine Project. https://github.com/bytedance/fedlearner/tree/fix_dev_sgx/sgx.

[77] Meni Orenbach, Andrew Baumann, and Mark Silberstein. 2020. Autarky: Closing Controlled Channels with Self-Paging Enclaves. In *Proceedings of the Fifteenth European Conference on Computer Systems* (Heraklion, Greece) *(EuroSys '20)*. Association for Computing Machinery, New York, NY, USA, Article 7, 16 pages. https://doi.org/10.1145/3342195.3387541

[78] Túlio Pascoal, Jérémie Decouchant, Antoine Boutet, and Paulo Esteves-Verissimo. 2021. Dyps: Dynamic, private and secure gwas. *Proceedings on Privacy Enhancing Technologies* (2021).

[79] Rishabh Poddar, Ganesh Ananthanarayanan, Srinath Setty, Stavros Volos, and Raluca Ada Popa. 2020. Visor: Privacy-preserving video analytics as a cloud service. In *29th USENIX Security Symposium (USENIX Security 20)*. 1039–1056.

[80] Donald E. Porter, Silas Boyd-Wickizer, Jon Howell, Reuben Olinsky, and Galen Hunt. 2011. Rethinking the Library OS from the Top Down. In *ASPLOS*. 291–304.

[81] Christian Priebe, Divya Muthukumaran, Joshua Lind, Huanzhou Zhu, Shujie Cui, Vasily A Sartakov, and Peter Pietzuch. 2019. SGX-LKL: Securing the host OS interface for trusted execution. *arXiv preprint arXiv:1908.11143* (2019).

[82] O. Purdila, L. A. Grijincu, and N. Tapus. 2010. LKL: The Linux kernel library. In *9th RoEduNet IEEE International Conference*.

[83] Wenjie Qiu. 2020. A Performance Analysis of Hardware-assisted Security Technologies. (2020).

[84] Do Le Quoc, Franz Gregor, Sergei Arnautov, Roland Kunkel, Pramod Bhatotia, and Christof Fetzer. 2020. SecureTF: A Secure TensorFlow Framework. In *Proceedings of the 21st International Middleware Conference* (Delft, Netherlands) *(Middleware '20)*. Association for Computing Machinery, New York, NY, USA, 44–59. https://doi.org/10.1145/3423211.3425687

[85] G Anthony Reina, Alexey Gruzdev, Patrick Foley, Olga Perepelkina, Mansi Sharma, Igor Davidyuk, Ilya Trushkin, Maksim Radionov, Aleksandr Mokrov, Dmitry Agapov, et al. 2021. OpenFL: An open-source framework for Federated Learning. *arXiv preprint arXiv:2105.06413* (2021).

[86] Michael Reininger, Arushi Arora, Stephen Herwig, Nicholas Francino, Jayson Hurst, Christina Garman, and Dave Levin. 2021. Bento: Safely bringing network function virtualization to Tor. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 821–835.

[87] Fabian Schwarz and Christian Rossow. 2020. {SENG}, the {SGX-Enforcing} Network Gateway: Authorizing Communication from Shielded Clients. In *29th USENIX Security Symposium (USENIX Security 20)*. 753–770.

[88] Michael Schwarz, Samuel Weiser, and Daniel Gruss. 2019. Practical enclave malware with Intel SGX. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 177–196.

[89] SGX-RA-TLS 2022. SGX RA TLS. https://github.com/cloud-security-research/sgx-ra-tls.

[90] Kripa Shanker, Arun Joseph, and Vinod Ganapathy. 2020. *An Evaluation of Methods to Port Legacy Code to SGX Enclaves*. Association for Computing Machinery, New York, NY, USA, 1077–1088. https://doi.org/10.1145/3368089.3409726

[91] Youren Shen, Hongliang Tian, Yu Chen, Kang Chen, Runji Wang, Yi Xu, Yubin Xia, and Shoumeng Yan. 2020. *Occlum: Secure and Efficient Multitasking Inside a Single Enclave of Intel SGX*. Association for Computing Machinery, New York, NY, USA, 955–970. https://doi.org/10.1145/3373376.3378469

[92] Edgeless Systems. 2024. Case Study: Major European Bank. https://www.edgeless.systems/resource-library/major-european-bank.

[93] Jonathan Takeshita, Colin McKechney, Justin Pajak, Antonis Papadimitriou, Ryan Karl, and Taeho Jung. 2021. GPS: Integration of Graphene, PALISADE, and SGX for Large-scale Aggregations of Distributed Data. *Cryptology ePrint Archive* (2021).

[94] Hajime Tazaki, Akira Moroo, Yohei Kuga, and Ryo Nakamura. 2021. How to design a library OS for practical containers?. In *Proceedings of the 17th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. 15–28.

[95] Flavio Toffalini, Andrea Oliveri, Mariano Graziano, Jianying Zhou, and Davide Balzarotti. 2021. The evidence beyond the wall: Memory forensics in SGX environments. *Forensic Science International: Digital Investigation* 39 (2021), 301313.

[96] Chia-Che Tsai, Kumar Saurabh Arora, Nehal Bandi, Bhushan Jain, William Jannen, Jitin John, Harry A. Kalodner, Vrushali Kulkarni, Daniela Oliveira, and Donald E. Porter. 2014. Cooperation and Security Isolation of Library OSes for Multi-Process Applications. In *EuroSys*.

[97] David Übler, Johannes Götzfried, and Tilo Müller. 2018. Secure remote computation using intel sgx. *SICHERHEIT 2018* (2018).

[98] Jo Van Bulck, Fritz Alder, and Frank Piessens. 2022. A Case for Unified ABI Shielding in Intel SGX Runtimes. (2022).

[99] Jo Van Bulck, David Oswald, Eduard Marin, Abdulla Aldoseri, Flavio D Garcia, and Frank Piessens. 2019. A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1741–1758.

[100] Jo Van Bulck, Nico Weichbrodt, Rüdiger Kapitza, Frank Piessens, and Raoul Strackx. 2017. Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 1041–1056.

[101] Kobe Vrancken, Frank Piessens, and Raoul Strackx. 2019. Securely deploying distributed computation systems on peer-to-peer networks. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 328–337.

[102] Ivana Vukotic, Vincent Rahli, and Paulo Esteves-Veríssimo. 2019. Asphalion: trustworthy shielding against byzantine faults. *Proceedings of the ACM on Programming Languages* 3, OOPSLA (2019), 1–32.

[103] Wubing Wang, Yinqian Zhang, and Zhiqiang Lin. 2019. Time and Order: Towards Automatically Identifying {Side-Channel} Vulnerabilities in Enclave Binaries. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. 443–457.

[104] Chathura Widanage, Weijie Liu, Jiayu Li, Hongbo Chen, XiaoFeng Wang, Haixu Tang, and Judy Fox. 2021. HySec-Flow: Privacy-Preserving Genomic Computing with SGX-based Big-Data Analytics Framework. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*. IEEE, 733–743.

[105] Shaza Zeitouni, Jo Vliegen, Tommaso Frassetto, Dirk Koch, Ahmad-Reza Sadeghi, and Nele Mentens. 2021. Trusted configuration in cloud FPGAs. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 233–241.

[106] Denghui Zhang and Zhaoquan Gu. 2021. A high-quality authenticatable visual secret sharing scheme using SGX. *Wireless Communications and Mobile Computing* 2021 (2021).

[107] D. Zhang, G. Wang, W. Xu, and K. Gao. 2019. SGXPy: Protecting Integrity of Python Applications with Intel SGX. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*. 418–425.

[108] Hong Zhong, Wenwen Cao, Qingyang Zhang, Jing Zhang, and Jie Cui. 2021. Toward Trusted and Secure Communication Among Multiple Internal Modules in CAV. *IEEE Internet of Things Journal* 8, 24 (2021), 17734–17746.