A Performance Analysis for Confidential Federated Learning

Bruno Casella*, Iacopo Colonnelli*, Gianluca Mittone*, Robert Birke*,
Walter Riviera†, Antonio Sciarappa‡, Carlo Cavazzoni‡ and Marco Aldinucci*
**University of Turin, email: {bruno.casella, iacopo.colonnelli, gianluca.mittone, robert.birke, marco.aldinucci}@unito.it*
†*University of Verona, email: walter.riviera@intel.com*
‡*Leonardo SpA, email: {antonio.sciarappa, carlo.cavazzoni}@leonardo.com*

Abstract—**Federated Learning (FL) has emerged as a solution to preserve data privacy by keeping the data locally on each participant's device. However, FL alone is still vulnerable to attacks that can cause privacy leaks. Therefore, additional security measures, at the cost of increasing runtimes, become necessary. The Trusted Execution Environment (TEE) approach offers the highest degree of security during execution. However, TEEs suffer from memory limits which prevent safe end-to-end FL training of modern deep models. State-of-the-art approaches limit secure training to selected layers, failing to avert the full spectrum of attacks or adopt layer-wise training affecting model performance. We benchmark the usage of a library OS (LibOS) to run the full, unmodified end-to-end FL training inside the TEE. We extensively evaluate and model the overhead of the different security mechanisms needed to protect the data and model during computation (TEE), communication (TLS), and storage (disk encryption). The obtained results across three datasets and two models demonstrate that LibOSes are a viable way to seamlessly inject security into FL with limited overhead (at most 2x), offering valuable guidance for researchers and developers aiming to apply FL in data-security-focused contexts.**

1. Introduction

Federated Learning [1] has recently gained popularity as a promising solution to tackle the challenge of training models on distributed data without centralizing it. While FL preserves data privacy by keeping data locally, it raises important concerns regarding security during communication, storage, and model execution. It is crucial to ensure that sensitive data is protected from unauthorized access, the integrity of information is preserved throughout the process, and the system is hardened against adversarial attacks. To achieve this, encryption techniques are essential to guarantee the confidentiality and authenticity of data and models during various stages of FL. In this work, we benchmark the runtimes of FL for image classification tasks, employing a combination of encryption techniques to ensure security.

In a typical FL workload, Deep Learning (DL) models are trained on local devices, keeping data locally and without sharing them with a central entity. This approach ensures data privacy and security, reducing the risks of disclosure or breach of personal information. The parties involved in a federation combine locally learned models into a globally shared one in an iterative way. The first proposed FL algorithm is FederatedAveraging (FedAvg), where collaboration is achieved by averaging the local model parameters. However, FL can be subject to privacy leaks either due to compromised systems allowing unauthorized access to local training data or via machine learning-based adversarial attacks violating the privacy by inferring/reconstructing (properties of) the training data [24]. For example, it is possible to infer training samples held by clients by knowing the previous aggregated model and the gradient update [15]. Encryption of memory, communication, and storage is required to ensure an end-to-end robust and reliable FL workload.

TEEs offer the opportunity to move (part of) the FL process into a secure enclave whose code can be attested and verified. Current secure enclaves present significant challenges in terms of code porting and adaptation. TEEs impose architectural constraints, such as limited protected memory and restriction to CPU resources, to avoid enlarging the attack surface due to communications over the PCIe bus. However, the recently released Nvidia H100 allows for combining enclaves with GPUs, overcoming the CPU-only limitation by exploiting VM-level isolation solutions such as Intel TDX. Moreover, TEEs introduce overheads due to enclave operations, which slow down code execution compared to running in an unprotected environment. Additional challenges include the difficulty of testing and debugging enclaves and carefully controlling communication between the enclave and the host environment. Despite these difficulties, TEEs provide an important level of security for applications needing execution and data protection.

Using TEEs directly requires dealing with all the above constraints. To lower the entry barrier, recently, LibOSes, like Gramine [12] Fortanix [26] and Anjuna [28], allow to run unmodified applications inside an enclave with minimal host requirements and porting efforts. LibOSes can also lift some TEE restrictions, such as the memory limit, by transparently and securely swapping in/out memory pages, enabling more efficient use of TEE functionalities. However, the impact of such LibOSes on FL workloads has not yet been investigated systematically.

In this work, we leverage Gramine to run the unmodified OpenFL [13] federated learning framework and study the impact of the different security mechanisms required to run an end-to-end robust and reliable FL workload. We chose these two because OpenFL provides out-of-the-box support for Gramine. More in detail, underneath Gramine, we rely on the Intel® Software Guard Extension (SGX) [11], a set of CPU instructions that offer application and service providers a secure environment to stand when managing the use of the data they consume and collect. SGX provides a secure, isolated area within the processor where data can be processed confidentially and protected from potential threats. This includes preventing unauthorized access by root users or compromised systems.To ensure the confidentiality of data stored on disk, we use the secure storage provided by Gramine, in which files are transparently decrypted/encrypted when the application accesses them. Gramine also allows signing and checking the integrity of all accessed data and binaries. Finally, we adopt the Transport Layer Security [10] security protocol to create secure communication channels among the FL participants.

The main contributions of our work are:

- Using reference image classification datasets, we benchmark the use of LibOS to run secure end-to-end FL pipelines and demonstrate its practicality.
- We perform extensive experiments to provide a systematic study on the runtime overhead of the different security mechanisms required for confidential FL (CFL) and summarise the results in a model.
- We show that leveraging a LibOS allows the exploiting of conventional end-to-end training, thus preserving the model's performance.
- We release the code to replicate our experiments and get familiar with FL pipelines on Gramine and SGX.

2. Related Work

We organize the related work into three parts: threats to FL systems, TEE-enhanced DL and FL frameworks, and trusted containers.

Threats to FL Systems. FL provides a false sense of security by keeping data locally and exchanging only model parameters or gradients. Besides the classic threats, such as rootkits providing malicious users with root permissions [2], FL is subject to different types of generic ML attacks. From a privacy perspective, FL is subject to three main attack categories. Membership inference attacks [9] aim to identify if a sample is in the training set. They are particularly successful when having access to the trained model and its last layers [8]. Property inference attacks [7] want to infer a global private property of the data. Strong aggregations ease this attack [3]. Inversion/extraction attacks try to extract representative or complete examples from the training data [6]. These attacks leverage access to early model layers and their updates on small data batches. Besides those targeting user privacy, further attacks can be more pernicious in FL settings. Model extraction attacks aim to duplicate the functionality of the model [5]. Poisoning attacks try to bias the resulting model to degrade its performance or draw predictions that are preferable to the opponent [4]. A mechanism of secure aggregation, by hiding parameters/updates from opponents, although insufficient to avert all attacks under any threat model, significantly hardens FL in a broad range of scenarios [20], [24].

TEE-enhanced Deep and FL. Related work leverages secure enclaves for training DNNs [18]. However, due to their size, prior work often must adopt ad-hoc strategies to circumvent the memory limit. A recent solution [19] devises a ternary partition mechanism and enforces enclaved execution only on the most sensitive partitions. DarkneTZ [17] runs only the last layers inside the TEE to avert membership inference attacks. However, it is critical to protect the full model to address a broader range of attacks. A recent work [20] proposes PPFL, a privacy-preserving FL framework leveraging TEEs (Arm TrustZone) in mobile devices. Authors circumvent the TEE memory limit by adopting a layer-wise training strategy to train layers one at a time. While PPFL significantly improves privacy, layer-wise training introduces a 3x higher delay on the end-to-end model and affects the model's accuracy. In contrast, SecFL [21] proposes an FL framework leveraging SCONE, a LibOS, to perform training inside TEE enclaves. Like us, SecFL runs OpenFL inside SGX enclaves; however, the authors do not provide details on implementation or experimental results.

TEEs. TEEs rely on hardware support, e.g., Intel® SGX and Arm® TrustZone, to allow the creation of protected memory regions called enclaves. Leveraging such enclaves directly is cumbersome, requiring modified executables and dealing with inherent limitations. Trusted containers aim to bypass such burdens. Several solutions are proposed [16] acting either as library OS [12], [26], libc wrapper [22] or using the WebAssembly System Interface [27]. LibOSes provide the functionality of an OS as a set of libraries directly linked to the application in a single address space. As such, they allow us to easily define boundaries enforcing security policies and reduce the need to invoke costly system calls that access the insecure world.

Different from related work, we use Gramine for libOS-based trusted containers that are able to run unmodified applications inside SGX enclaves, reducing the porting effort to a minimum. We conduct an extensive series of experiments, training various image classification models end-to-end rather than layer-wise on popular datasets. We proceed with a thorough assessment of the overhead introduced by the different encryption techniques employed.

3. System overview

Fig. 1 presents an overview of the considered CFL system. We couple a containerized FL framework with LibOS-based trusted containers, disk encryption (DE) and secured network communications. While the overall architecture is agnostic to the specific FL framework and LibOS, we adopted OpenFL and Gramine to conduct our experiments. OpenFL is an FL framework developed by Intel® Internet of Things Group (IOTG) and Intel® Labs, which is DL framework-agnostic. Here, we couple it with PyTorch as

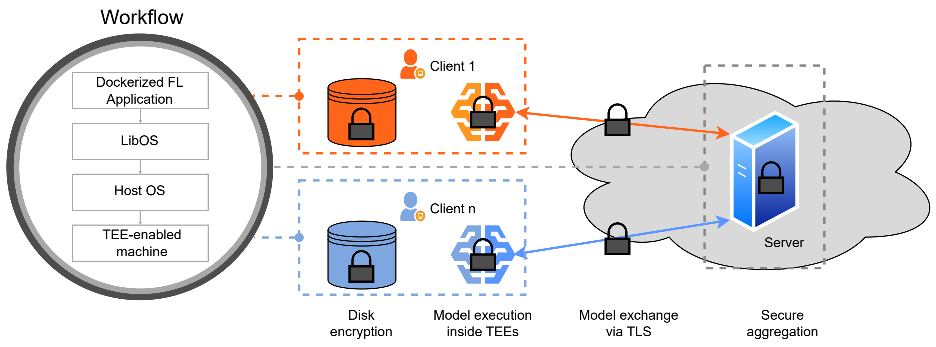


Figure 1: Overview of a generic system of CFL. In our proposed system, the FL framework OpenFL runs inside Gramine, a libOS requiring a Linux kernel with SGX drivers. Gramine provides disk encryption. Model execution (training and validation) and aggregation are performed inside Intel® SGX enclaves. TLS provides secure communication.

the DL library. Both client-side and server-side OpenFL applications are executed within SGX using Gramine. Moreover, we use Gramine to use encrypted disk data optionally and optionally configure OpenFL with TLS to secure all network communications. For ease of deployment, we further encapsulate the graminized OpenFL inside Docker images, providing an isolated and portable environment for execution. The combined use of SGX, Gramine, TLS, and Docker ensures a high level of security and portability of the OpenFL application. Below, we provide a detailed description of the main security-related components.

Intel® SGX. SGX is a hardware-based security technology for Intel processors first introduced on the Xeon® server line of CPUs. It aims to offer effective isolation and protection for sensitive data and code within a TEE. SGX depends on new processor instructions and architectural elements that create isolated memory areas known as *enclaves*. Enclave code and data are kept in a region of protected physical memory called the *enclave page cache* (EPC). These enclaves are encrypted and secured against outside access, ensuring that data and computations inside remain private and essential even in the presence of malicious software, privileged users, or compromised operating systems. When a thread needs to run enclave code, the CPU enters enclave mode, and the control jumps to a predefined enclave entry point, introducing a performance overhead. To prevent leaks, enclaves limit the memory, minimizing the trusted code base to reduce the attack surface and restrict access to OS system calls since they run as unprotected functions in kernel space. The use of cryptographic attestation ensures the integrity of the enclaved content.

Gramine. Gramine is a LibOS developed to run native, unmodified Linux applications on any platform. If SGX is available, Gramine can run applications inside the enclave with minimal requirements and porting efforts, thus providing security guarantees and protecting against malicious host OSes. Gramine supports multi-processing and multi-threading and virtualizes process/thread identifiers. Gramine attaches itself as a LibOS to an application and intercepts every request it makes to the host OS. Gramine fully handles some of these requests, while others are funnelled to the host OS via an API. In either case, the correctness and consistency of each application's request and each host's response are checked. Gramine keeps an internal, ”shadow” state for these checks. As a result, Gramine defends against Iago attacks [29], a type of assault that exploits vulnerabilities of an untrusted OS by returning malicious syscalls. Moreover, the syscalls handled fully by Gramine improve the application performance by avoiding costly switches to the kernel in the unprotected world.
Gramine runs inside a process called picoprocess, which contains the binary code and the libraries required for the application to be executed, the LibOS instance, and the Platform Adapter Layer (PAL), which provides an interface between the libOS and the kernel. The PAL is responsible for translating the system calls for the kernel. Gramine offers two backends: execution on the host OS and execution inside an SGX enclave. A trusted reference monitor ensures libOS isolation, handling system calls that affect both the internal and external picoprocess state. The libOS interfaces with a small number of PAL functions, thus reducing the attack surface. Native Gramine, before said *Graphene*, protects the system underlying the libOS by isolating the picoprocess. To benefit from this security, applications can run inside Gramine with the *gramine-direct* command. Gramine implemented the functionalities required to be compatible with SGX enclaves later. To execute applications inside an SGX enclave by using Gramine, the command is *gramine-sgx*.

We deployed a “graminized” version of a Docker image by installing Gramine and OpenFL, which can be run with the standard Docker commands.

Disk Encryption. Gramine does not simply mirror the filesystem of the host OS but creates its own view of the directories and files on the selected portion of the host. Gramine also provides the mechanism of DE. Encrypted files are transparently decrypted when accessed by Gramine or the application running inside Gramine on each file access via SGX SDK Merkle-tree format. It is possible to encrypt files by specifying a path and an encryption key. Additionally, Gramine allows the mounting of different files and directories with different encryption keys. The benefits of DE are data confidentiality (only authorized applications running in the SGX enclave can access the data), data integrity (tamper resistance: encrypted files protect data from unauthorized changes and alterations), and file swap protection (an encrypted file can only be accessed when in a specific host path, preventing unauthorized file swaps).

Transport Layer Security (TLS). TLS is a widespread cryptographic protocol used to secure communications over computer networks. TLS is used to protect data in client-server communications. It mainly introduces overhead at connection initiation to perform the handshake, and derive the encryption keys, and encrypt/decrypt all data when sending/receiving over the secured connection.

System Tuning. Various factors can combine to cause undesired performance degradation, depending on the number of threads and the size of the SGX enclave. In particular, glibc's malloc speculatively requests a 64 MB per-thread arena from the kernel when a thread first calls malloc as an optimization, as the default settings assume that virtual memory is almost limitless. However, SGX V1 breaks this because it requires that any memory request is immediately backed up by physical memory. When several threads are spawned (and all call malloc), the per-thread arenas may use up a significant amount of the memory reserved for the enclave. Calls to malloc will not fail in this situation since the allocator will allocate a single page to fulfil the request in its place. However, the remainder of the page will not be used, wasting most of this memory. This means that if $64 \cdot N_T > E$, with N_T the application's thread count, and E the enclave size, calls to malloc will be much slower and less memory-efficient than they should be. We found that our PyTorch workloads spawned 8 threads per physical core, exceeding $E = 16GB$ (E must be a power of 2, and it was impossible to increase it to 32 to avoid exceeding physical memory). We discovered empirically that the best results, in terms of runtime performance, are achieved when the number of threads is set equal to the number of core(s) per socket of the running machine (forty in our case).

Another possible tweaking point is the GNU OpenMP library (libgomp). The standard libgomp uses raw syscall instructions, which Gramine handles via a slow path, i.e., by catching exceptions. Gramine can be built with a patched version of libgomp, which uses a fast path by replacing system calls with direct function calls.Testing a PyTorch DL application running in Gramine with the patched version of libgomp showed better runtime performance than running with the standard one. However, our Dockerized application showed no significant benefits from using either version, so we adopted the standard one for our experiments.

4. Experiments Setup

Federation Setup. The entities of the federation workflow of OpenFL are the Aggregator (the server) and the Collaborators (the clients). Each collaborator executes one training and two validation stages for each federation round. Firstly, they validate the global model produced by the aggregator on the local test data. Then, they train the model on the local training data. Finally, they validate the updated (local) model on local test data.

	MNIST	CIFAR10	CIFAR100
Dimension	28x28	32x32	32x32
Representation	Binary	RGB	RGB
Train samples	60.000	50.000	50.000
Test samples	10.000	10.000	10.000
Size (MB)	64	170	169

TABLE 1: Statistics of the datasets.

	ResNet-18	MobileNetV3-Small
Parameters	11.181.642	1.528.106
Size (MB)	42.69	5.88
Layers	68	210

TABLE 2: Statistics of the models.

Testbed Setup. All experiments used a distributed environment encompassing one Aggregator and three Collaborators, each deployed on a dedicated server. All servers are dual sockets with two 3rd generation Intel® Xeon Scalable Platinum 8380 CPUs (40 cores @ 2.30GHz) with SGX support. The servers communicate via 100Gb/s Ethernet links. As software, we used OpenFL v1.5 and Gramine v1.4. We release all the files required to reproduce our experiments (Dockerfiles, OpenFL manifest, and Python code) at the following link: https://github.com/CasellaJr/SGX-FL-OPENFL-GRAMINE-BENCHMARK-IMAGES.

Datasets. We used three image classification benchmark datasets, i.e., MNIST [31], CIFAR10 and CIFAR100 [32]. The statistics of each dataset are summarized in Table 1.

Models. We employed two widely used literature image classification models: ResNet-18 [14] and MobileNetV3-Small [23]. We trained each model for 100 rounds using the cross-entropy loss. Adam was used as optimizer, with a learning rate $1e^{-4}$.Table 2 summarises the statistics of the two networks for the CIFAR10 dataset. Using more classes, i.e., CIFAR100, slightly increases the number of parameters and model size by 0.41%. Using smaller inputs/fewer colours (MNIST) slightly decreases them by 0.06%.

5. Evaluation

We analyzed the impact of all the components shown in Fig. 1 on the models' runtimes and final accuracies.

Model Runtime. We train the two models on the three datasets, starting from the baseline and incrementally adding security mechanisms. Each one adds some extra execution steps. Gramine-direct redirects the system calls through Gramine. SGX further runs the application in the security enclave. We also selectively enable encrypted disk data (DE) and secure communications (TLS). Table 3 summarizes the total wall clock times of each experiment.Results expressed in the HH:MM time format reveal an easily discernible pattern. As expected, each addition of a security measure leads to an increase in execution time. Overall, it is possible to state that a fully encrypted FL pipeline doubles the execution time. We elaborate on these results further via ablation studies in the subsequent section.

Model Accuracy. Since our approach preserves accuracy without suffering any loss compared to other mechanisms of CFL, the discussion and the plots of the model's performance are reported in the Appendix.

5.1. Ablation Studies

We conducted different ablation studies using combinations of the previously presented encryption techniques to build a performance overhead model and to investigate their impact on different execution phases.

Performance Overhead Model. Each security mechanism adds extra steps which intuitively (and as seen previously in Table 3) add a bit to the execution time. Hence, we can model the execution time T as a linear combination with one term to capture the relationship with each discussed security mechanism:

$$T = T_{baseline} + O_{Gramine} + O_{SGX} + O_{DE} + O_{TLS}$$

where $T_{baseline}$ is the runtime without any security mechanism and $O_{Gramine}, O_{SGX}, O_{DE}, O_{TLS}$ represent the overheads introduced by Gramine, SGX, DE, and TLS, respectively. We express each overhead as a cost coefficient C for the $T_{baseline}$, i.e. $O = C \cdot T_{baseline}$.

We fit this model using non-negative linear regression, leveraging the knowledge that each mechanism only adds extra work. Table 3 shows the fitted values for both models.

As shown in Table 3, the overhead introduced by Gramine and by SGX highly depends on the model architecture. ResNet-18 is more significantly impacted by the performance penalty caused by Gramine (about 2.7x), whilst MobileNet-V3 Small is more affected by the performance penalty caused by SGX (about 5.9x). The first difference results from ResNet-18's significantly higher memory occupancy, about seven times greater than MobileNet's. As a result, ResNet-18 relies more on the memory management introduced by Gramine to satisfy the memory constraints, resulting in an increased overhead attributable to Gramine. Conversely, in the case of MobileNetV3-Small, the slowdown attributed to SGX is more significant. This is because MobileNetV3-Small has a larger number of layers, about 3.1 times more, which, due to the internals of OpenFL, increases the number of I/O and network operations. These, in turn, increase the overhead caused by switching from the secure enclave to the unprotected world. Finally, the overheads introduced by DE and TLS are comparable for both models, although they are slightly higher in the case of ResNet-18, as its size and magnitude in MB are greater than MobileNetV3-Small.

Analysis of the Computation Overhead. For each collaborator of the federation, the typical OpenFL computation workload consists of three steps: validation of the previously received aggregated model from the server, refinement via training on the local data, and validation of the locally tuned model. Fig. 4 shows stacked bar plots of compute time for the three phases, for the three datasets, and different combinations of security mechanisms. Here, we exclude the communication as well as synchronization time.

First, as expected, training dominates the execution time at each collaborator, with validation times of the aggregated and locally tuned models being only a small fraction of each round's compute time. As previously explained, the slowdown introduced by Gramine is significantly higher for ResNet-18 because a model with larger layers stresses Gramine's memory manager more. In the case of MobileNetV3-Small, the overhead caused by SGX is more noticeable because a model with more layers tends to incur more switches between the secure enclave and unprotected worlds, which suffer from SGX overhead costs.

DE causes a small amount of overhead, which aligns with the Gramine documentation, which expects a 1-10$\%$ cost. Notably, in some cases, SGX and DE are slightly faster than SGX alone. This happens because SGX is very sensitive to small transient variations in the system conditions.

Surprisingly, we also observe a (small) impact of TLS on computation times. The combined use of SGX and TLS adds overhead to using only SGX. TLS has more impact during the training of ResNet-18 than of MobileNetV3-Small. However, we are not sure about the exact causes.

Analysis of the Communication Overhead. Unlike the previous section, here we focus on the sole communication times (the latency required for the different nodes of the network to exchange information with the server). These are reported in Table 4.

One can note that Gramine, SGX, and TLS all add communication overhead. This happens because Gramine must relay the system calls to send/receive data through the kernel. This also requires a switch between the secure enclave and the unprotected world, i.e. where the kernel is located, which incurs significant overheads [25]. However, the greater source of communication overhead, as seen from the cost coefficients of the linear regression models in Table 3, is given by TLS. This is an expected result because TLS adds a delay during connection setup to do the TLS handshake and key derivation and during data transfers to encrypt/decrypt the data sent over the secure communication channel. While the relative overhead is comparable between the two models, the absolute times are a bit surprising. Indeed, while ResNet-18 has around 7 times the number of trainable parameters and size (in megabytes) of MobileNetV3-Small, the communication time of MobileNetV3-Small is approximately twice that of ResNet-18. This happens because in OpenFL, collaborators request the aggregated model by layers, and MobileNetV3-Small has roughly 3x the layers of ResNet18 (even if all of them are smaller).
Finally, DE does not introduce any additional delay.

6. Conclusions

In this work, we proposed a performance analysis for CFL. We leveraged a libOS, Gramine, to execute a fully encrypted FL pipeline within a TEE, Intel$^{®}$ SGX. A thorough analysis and modelling of the overhead caused by various se-

	ResNet-18			MobileNetV3-Small		
	MNIST	**CIFAR10**	**CIFAR100**	**MNIST**	**CIFAR10**	**CIFAR100**
Baseline	01:55	02:01	01:54	01:40	01:38	01:29
Gramine-direct	03:10	03:33	03:32	02:12	01:54	02:03
SGX	03:17	03:36	03:49	02:43	02:24	02:36
SGX + DE	03:21	03:41	03:40	02:48	02:26	02:43
SGX + TLS	03:42	03:53	03:59	02:53	02:37	02:55
SGX + TLS + DE	03:51	04:07	04:11	02:54	02:38	03:01

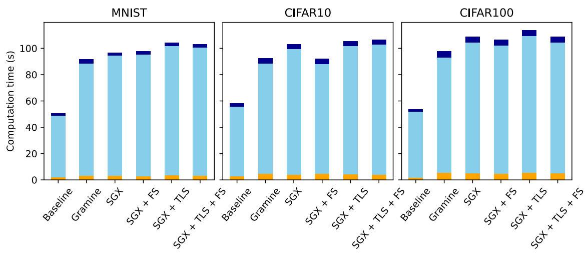
TABLE 3: Wall-clock execution times [HH:MM] to train models on 100 federated training rounds of 1 epoch.

$$\begin{aligned} O_{Gramine} &= C_{Gramine} \cdot T_{baseline} \\ O_{SGX} &= C_{SGX} \cdot T_{baseline} \\ O_{DE} &= C_{DE} \cdot T_{baseline} \\ O_{TLS} &= C_{TLS} \cdot T_{baseline} \end{aligned}$$

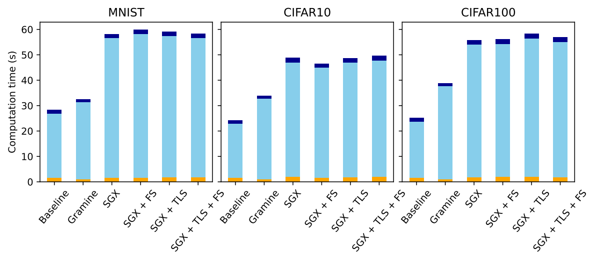
Figure 2: Overhead terms of the wall-clock times.

	ResNet-18	MobileNetV3-Small
$C_{Gramine}$	0.754	0.282
C_{SGX}	0.056	0.330
C_{DE}	0.046	0.040
C_{TLS}	0.197	0.133

Figure 3: Coefficients of the linear regression models.



(a) ResNet-18



(b) MobileNetV3-Small

Figure 4: Computation times [s] of the models (mean of three clients and 100 federated training rounds) for: a) ResNet-18 and b) MobileNetV3-Small. The orange bar refers to the validation of the aggregated model, the light blue bar refers to the training time, and the dark blue bar refers to the locally tuned model validation.

	ResNet-18			MobileNetV3-Small		
	MNIST	**CIFAR10**	**CIFAR100**	**MNIST**	**CIFAR10**	**CIFAR100**
Baseline	5.0 ± 0.0	5.0 ± 0.0	5.0 ± 0.0	10.0 ± 0.0	10.05 ± 0.26	5.0 ± 0.0
Gramine-direct	6.91 ± 0.81	6.71 ± 0.66	6.74 ± 0.63	13.32 ± 1.10	13.03 ± 0.79	13.35 ± 0.99
SGX	7.49 ± 0.60	7.44 ± 0.51	7.31 ± 0.53	14.25 ± 0.73	14.19 ± 0.65	14.69 ± 1.06
SGX + DE	7.52 ± 0.61	6.67 ± 0.66	7.26 ± 0.47	14.72 ± 1.11	14.32 ± 0.72	14.94 ± 1.32
SGX + TLS	11.04 ± 0.64	11.09 ± 0.54	11.0 ± 0.0	21.81 ± 0.91	21.28 ± 0.60	22.17 ± 1.28
SGX + TLS + FS	11.21 ± 0.59	11.16 ± 0.57	11.19 ± 0.57	21.33 ± 0.69	21.40 ± 0.64	22.32 ± 1.60

TABLE 4: Communication times in seconds (mean ± standard deviation of three clients and 100 federated training rounds).

curity mechanisms, including DE, TEE for model protection during computation, and TLS for communications, is carried out. Results spanning over three datasets and two models show that libOSes represent a practical way to integrate security into FL without the prohibitive overhead (up to 2x), providing crucial knowledge for researchers and developers looking to implement FL in contexts centred around data security. One drawback of our method is the underlying vulnerability of the TEEs, suffering from side-channel attacks, such as cache attacks [30]. Here, we consider such attacks out of scope, relying on the ideal concept of TEE.

For future work, we aim to test the performance of CFL applications based on VM-level isolation, thanks to recently released solutions such as Intel TDX, which also allow for the deployment of the world's first GPU with confidential computing capabilities, the Nvidia H100.

Acknowledgment

This work has been partly supported by the Spoke "FutureHPC & BigData" of the ICSC – Centro Nazionale di Ricerca in "High Performance Computing, Big Data and Quantum Computing", funded by European Union – NextGenerationEU, and by the Horizon2020 RIA EPI project (G.A. 826647). We thank the "Intel® SMG Customer Innovation Labs Swindon" laboratory for granting us access to their SGX-enabled machines.

References

[1] McMahan, B., Moore, E., Ramage, D., Hampson, S. & Arcas, B. Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proc. Of The 20th Intl. Conference On Artificial Intelligence And Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*. vol. 54 pp. 1273-1282 (2017)

[2] Blunden, B. The Rootkit arsenal: Escape and evasion in the dark corners of the system. (Jones & Bartlett Publishers,2012)

[3] Mo, F., Borovykh, A., Malekzadeh, M., Haddadi, H. & Demetriou, S. Layer-wise characterization of latent information leakage in federated learning. *Distributed And Private Machine Learning (DPML) Workshop ICLR*. (2021)

[4] Tian, Z., Cui, L., Liang, J. & Yu, S. A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning. *ACM Comput. Surv.*. vol. 55, 166:1-166:35 (2023), https://doi.org/10.1145/3551636

[5] Tramèr, F., Zhang, F., Juels, A., Reiter, M. & Ristenpart, T. Stealing Machine Learning Models via Prediction APIs. *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*. pp. 601-618 (2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer

[6] Fredrikson, M., Jha, S. & Ristenpart, T. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. *Proceedings Of The 22nd ACM SIGSAC Conference On Computer And Communications Security, Denver, CO, USA, October 12-16, 2015*. pp. 1322-1333 (2015), https://doi.org/10.1145/2810103.2813677

[7] Ganju, K., Wang, Q., Yang, W., Gunter, C. & Borisov, N. Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations. *Proceedings Of The 2018 ACM SIGSAC Conference On Computer And Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. pp. 619-633 (2018), https://doi.org/10.1145/3243734.3243834

[8] Nasr, M., Shokri, R. & Houmansadr, A. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. *2019 IEEE Symposium On Security And Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. pp. 739-753 (2019), https://doi.org/10.1109/SP.2019.00065

[9] Shokri, R., Stronati, M., Song, C. & Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. *2017 IEEE Symposium On Security And Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. pp. 3-18 (2017), https://doi.org/10.1109/SP.2017.41

[10] Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.3. (RFC Editor,2018,8), https://www.rfc-editor.org/info/rfc8446

[11] McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C., Shafi, H., Shanbhogue, V. & Savagaonkar, U. Innovative instructions and software model for isolated execution. *HASP 2013, The Second Workshop On Hardware And Architectural Support For Security And Privacy, Tel-Aviv, Israel, June 23-24, 2013*. pp. 10 (2013)

[12] Tsai, C., Arora, K., Bandi, N., Jain, B., Jannen, W., John, J., Kalodner, H., Kulkarni, V., Oliveira, D. & Porter, D. Cooperation and security isolation of library OSes for multi-process applications. *Ninth Eurosys Conference 2014, EuroSys 2014, Amsterdam, The Netherlands, April 13-16, 2014*. pp. 9:1-9:14 (2014)

[13] Reina, G., Gruzdev, A., Foley, P., Perepelkina, O., Sharma, M., Davidyuk, I., Trushkin, I., Radionov, M., Mokrov, A., Agapov, D., Martin, J., Edwards, B., Sheller, M., Pati, S., Moorthy, P., Wang, H., Shah, P. & Bakas, S. OpenFL: An open-source framework for Federated Learning. *CoRR*. abs/2105.06413 (2021), https://arxiv.org/abs/2105.06413

[14] He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *2016 IEEE Conference On Computer Vision And Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. pp. 770-778 (2016)

[15] Geiping, J., Bauermeister, H., Dröge, H. & Moeller, M. Inverting Gradients - How easy is it to break privacy in federated learning?. *Advances In Neural Information Processing Systems 33: Annual Conference On Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*. (2020)

[16] Paju, A., Javed, M., Nurmi, J., Savimäki, J., McGillion, B. & Brumley, B. SoK: A Systematic Review of TEE Usage for Developing Trusted Applications. *The 18th International Conference On Availability, Reliability And Security (ARES 2023), August 29 – September 01, 2023, Benevento, Italy*. pp. 15 (2023)

[17] Mo, F., Shamsabadi, A., Katevas, K., Demetriou, S., Leontiadis, I., Cavallaro, A. & Haddadi, H. DarkneTZ: towards model privacy at the edge using trusted execution environments. *MobiSys '20: The 18th Annual International Conference On Mobile Systems, Applications, And Services, Toronto, Ontario, Canada, June 15-19, 2020*. pp. 161-174 (2020), https://doi.org/10.1145/3386901.3388946

[18] Tanuwidjaja, H., Choi, R., Baek, S. & Kim, K. Privacy-Preserving Deep Learning on Machine Learning as a Service - a Comprehensive Survey. *IEEE Access*. **8** pp. 167425-167447 (2020), https://doi.org/10.1109/ACCESS.2020.3023084

[19] Gu, Z., Huang, H., Zhang, J., Su, D., Lamba, A., Pendarakis, D. & Molloy, I. Securing Input Data of Deep Learning Inference Systems via Partitioned Enclave Execution. *CoRR*. abs/1807.00969 (2018), http://arxiv.org/abs/1807.00969

[20] Mo, F., Haddadi, H., Katevas, K., Marin, E., Perino, D. & Kourtellis, N. PPFL: privacy-preserving federated learning with trusted execution environments. *MobiSys '21: The 19th Annual International Conference On Mobile Systems, Applications, And Services, Virtual Event, Wisconsin, USA, 24 June - 2 July, 2021*. pp. 94-108 (2021), https://doi.org/10.1145/3458864.3466628

[21] Quoc, D. & Fetzer, C. SecFL: Confidential Federated Learning using TEEs. *CoRR*. abs/2110.00981 (2021), https://arxiv.org/abs/2110.00981

[22] Arnautov, S., Trach, B., Gregor, F., Knauth, T., Martin, A., Priebe, C., Lind, J., Muthukumaran, D., O'Keeffe, D., Stillwell, M., Goltzsche, D., Eyers, D., Kapitza, R., Pietzuch, P. & Fetzer, C. SCONE: Secure Linux Containers with Intel SGX. *12th USENIX Symposium On Operating Systems Design And Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*. pp. 689-703 (2016), https://www.usenix.org/conference/osdi16/technical-sessions/presentation/arnautov

[23] Howard, A., Pang, R., Adam, H., Le, Q., Sandler, M., Chen, B., Wang, W., Chen, L., Tan, M., Chu, G., Vasudevan, V. & Zhu, Y. Searching for MobileNetV3. *2019 IEEE/CVF International Conference On Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. pp. 1314-1324 (2019), https://doi.org/10.1109/ICCV.2019.00140

[24] Kairouz, P., McMahan, H., Avent, B., Bellet, A., Bennis, M., Bhagoji, A., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R., Eichner, H., Rouayheb, S., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S., Sun, Z., Suresh, A., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F., Yu, H. & Zhao, S. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.*. vol. 14, 1-210 (2021), https://doi.org/10.1561/2200000083

[25] Orenbach, M., Lifshits, P., Minkin, M. & Silberstein, M. Eleos: ExitLess OS Services for SGX Enclaves. *Proceedings Of The Twelfth European Conference On Computer Systems, EuroSys 2017, Belgrade, Serbia, April 23-26, 2017*. pp. 238-253 (2017), https://doi.org/10.1145/3064176.3064219

[26] Fortanix Fortanix Confidential Computing Manager. (2023,8,8), https://www.fortanix.com/

[27] Teaclave Apache Teaclave (Incubating). (2023,5,31), https://teaclave.apache.org

[28] Anjuna Security, Inc. Anjuna Enterprise Enclaves. (2023,8,8), https://www.anjuna.io/

[29] Checkoway, S. & Shacham, H. Iago attacks: why the system call API is a bad untrusted RPC interface. *Architectural Support For Programming Languages And Operating Systems, ASPLOS 2013, Houston, TX, USA, March 16-20, 2013*. pp. 253-264 (2013), https://doi.org/10.1145/2451116.2451145

[30] Jiang, J., Soriente, C. & Karame, G. On the Challenges of Detecting Side-Channel Attacks in SGX. *25th International Symposium On Research In Attacks, Intrusions And Defenses, RAID 2022, Limassol, Cyprus, October 26-28, 2022*. pp. 86-98 (2022), https://doi.org/10.1145/3545948.3545972.

[31] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE*. **86**, 2278-2324 (1998)

[32] Krizhevsky, A., Hinton, G. & Others Learning multiple layers of features from tiny images. (Citeseer,2009), https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf

Appendix

Figures 5 to 7 show the accuracy per round of the aggregated model of a ResNet-18. We observe similar results across all experiments. As shown in Figures 5 to 7, the adoption of encryption techniques has no effect on the overall accuracy of the model. This is because the model is trained in a conventional end-to-end way inside Gramine and the SGX enclave. The observed variability is in line with the inherent variability of the training algorithm as can be seen from the baseline case. So, unlike layer-wise training techniques that have been employed in secure FL pipelines to circumvent memory constraints [20], the use of SGX through Gramine preserves the standard training procedure without any impact on model accuracy.

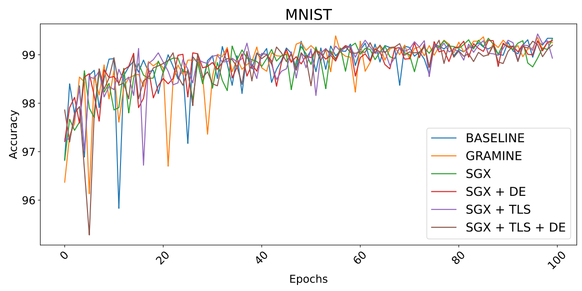


Figure 5: Evolution of ResNet-18 accuracy on MNIST with federated training rounds.

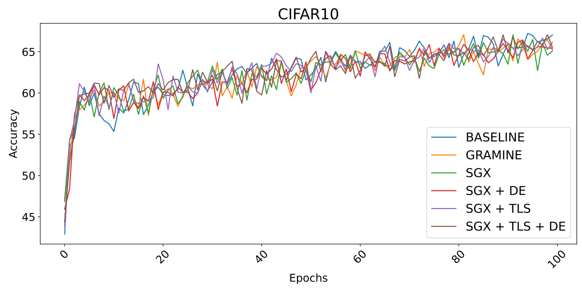


Figure 6: Evolution of ResNet-18 accuracy on CIFAR10 with federated training rounds.

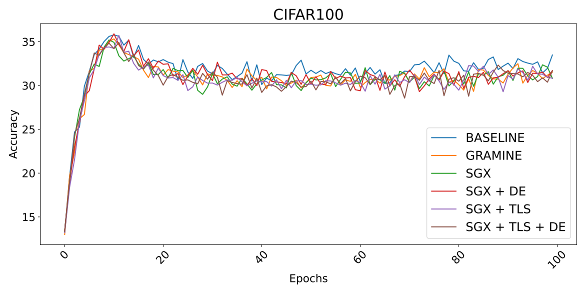


Figure 7: Evolution of ResNet-18 accuracy on CIFAR100 with federated training rounds.