RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Technical Position Paper on Confidential Computing

French Cybersecurity Agency (ANSSI)

October 1, 2025

Executive summary

Confidential Computing is a set of technologies designed to execute sensitive workloads in remote environments (eg. Virtual Machines). It complements effectively at-rest and in-transit encryption by encrypting data in-use, shielding them from direct inspection by the administrator on shared infrastructure. The technology is not without vulnerabilities and limitations, and lacks security certifications, but still provides significant defence-in-depth. If used correctly, it may raise the complexity of attacks from the host or other tenants on the same physical machine, and reduce the size of the Trusted Computing Base (TCB), leading to a smaller attack surface overall.

However, Confidential Computing is not secure enough to protect data integrity and confidentiality against a hostile administrator performing targeted, active attacks. Under such a threat model, users must avoid running on shared infrastructure operated by providers they cannot trust, and are rather encouraged to leverage Confidential Computing to increase their security posture on dedicated hardware instead. When manipulating highly sensitive data, trust and security of the entire software and hardware supply-chain must also be carefully taken into account.

In addition to the security limitations inherent to Confidential Computing, it is also very difficult to use in a secure manner. Workloads must be hardened against specific attacks from the host. They must also be proven genuine through Remote Attestation, and decryption of sensitive data should be cryptographically tied to this attestation process. This deeply changes the way workloads are architectured and deployed. The ecosystem currently lacks mature software stacks integrating attestation with secret delivery, and deploying a complete solution involves deep expertise to ensure the integrity of the entire TCB.

The current analysis shows that Confidential Computing is not sufficient on its own to secure an entire system, or to meet the requirements described in section 19.6 of the SecNumCloud 3.2 framework [1]. Users are encouraged to remain cautious, opting into the technology only when the expected security benefits exceed the development costs, factoring in residual risks into their threat model. On the other hand, providers need to offer solutions that allow remote attestation of the entire TCB, and collaborate on open protocols to ease interoperability between workload execution, remote attestation and secret provisioning.

In this position paper, the current ANSSI views on *Confidential Computing* are outlined. In particular, ANSSI recalls the attack models that Confidential Computing purports addressing, its main security mechanisms (Trusted Execution Environments and Remote Attestation) and their current limitations. The objective of the paper is twofold: (1) providing guidelines to *Cloud Service Providers* (CSPs) and other companies developing security products relying on *shared* computing resources featuring hardware extensions enabling Confidential Computing technologies, and (2) clarifying the residual risks for end-users of such products, even assuming those guidelines are fulfilled.

What is Confidential Computing?

The security of a computer system relies on a set of protection mechanisms, called the *Trusted Computing Base* (TCB), which consists in a "*combination of software, firmware and hardware that* *are critical to its security, in the sense that bugs or vulnerabilities occurring inside the TCB might jeopardize the security properties of the entire system*" [2]. In the context of shared systems, such as Cloud services, the TCB includes the underlying hardware, the host operating system (OS) and hypervisor, as well as the guest OS and software stack running the final user application. A vulnerability in any of those privileged components will impact the security of the whole.

This matters particularly in the context of Cloud computing, where a large part of the stack is operated by *Cloud Service Providers* (CSP) and cannot be controlled or verified by the user. In this model, data can be protected "*at rest*" (*i.e.* encrypted when it is stored on mass storage) and "*in transit*" (*i.e.* during exchanges over the network); however the CSP can still access without any restrictions the memory content of the virtual machines running on the operated computing resources. This lack of confidentiality and integrity guarantees for data "*in use*" have hindered the switch to cloud-based technologies for tasks and workflows involving sensitive data, due to the legitimate reluctance of some customers to trust the CSP in this context.

To try to address this problem, *Confidential Computing* is a set of technologies developed by processor vendors to protect data in use, by isolating computation on the main processor in a secure, segregated environment called an *enclave*. An enclave is a memory and CPU-only environment[1] that is isolated from all other processes on a same host. Confidential Computing aims at reducing the size of the TCB to the software components running in the enclave and the underlying hardware, excluding the boot firmware, host OS and hypervisor (that are typically provided and operated by the CSP). Since this secure execution environment is remote, the user also needs to be able to control its authenticity and the integrity of the services it runs.

In practice, Confidential Computing guarantees data integrity, data confidentiality and code integrity[2] through the combination of two key elements:

- a Trusted Execution Environment (TEE) to run the code;
- a remote attestation mechanism to verify the integrity and state of the TEE.

The **Trusted Execution Environment (TEE)** is a type of enclave design, implementing a specific context in processors to execute sofware workloads securely, isolated from other workloads — more details are provided in the remainder of this article. Those other, untrusted, workloads may be privileged software under control of the CSP, such as the firmware or hypervisor, but also unpriviledged software from other tenants on shared infrastructure. Since the TEE operates on data that is stored in shared memory (DRAM), Confidential Computing technologies encrypt DRAM with cryptographic material whose access is controlled by the TEE. At a high level, operations on sensitive data are performed in three steps:

1. the data is read from DRAM and decrypted with a key specific to the currently executing workload;
2. the workload operates on the unencrypted data within the TEE, isolated from other untrusted code;
3. the modified data is encrypted back when it leaves the TEE and is written back to DRAM.

[1] Note that, outside of Confidential Computing, *enclave* can have a different meaning of physically separated component, with its own dedicated processor and memory. In the context of this document, CPU and memory are shared with the host, with additional mechanisms such as cache clearing and memory encryption to enforce isolation.

[2] Note that the code confidentiality is not ensured, since the processor needs the workload to be unencrypted to start executing it. It is however possible to start with a minimal, unencrypted workload and fetch additional code from encrypted mass storage.

Remote Attestation (or **attestation**, for short) allows the user to verify that their workload is being executed unmodified in a genuine TEE. The hardware implementing the TEE generates a machine-readable report containing measurements of the running workload (*e.g.* a hash of the code), information about the underlying hardware (*e.g.* the processor model, firmware version and a manufacturer-allocated unique identifier) and a cryptographic signature bound to the TEE's hardware-protected keys. These TEE-specific keys are themselves signed by the processor vendor. The user first ensure that the report was generated on a genuine processor by verifying its signature and the certificate chain back to the processor vendor's public root of trust, then validates its content against their expectations about the workload and the state of the TEE. Since protecting data in-use is useless without in-transit and at-rest protection, attestation can be used to extend trust from the workload to those:

- in-transit encryption: an attestation report can embed a session key to establish a secure communication channel with a remote party over the network (*e.g.* TLS); like any information provided in the report, it is used only after the attestation process described above has completed successfully.
- at-rest encryption: key material required to decrypt sensitive data at rest is either bound to the specific machine the workload is running on by the TEE hardware, or provided by a trusted third-party upon successful attestation; the latter should be more common in a cloud setting, to allow workload migration between physical hosts.

These attestation mechanisms can be used to ensure that only a genuine, trusted workload is able manipulate the confidential data, without a human or untrusted code in the loop, with the aim of avoiding data leakage.

Confidential Computing is often presented by commercial providers as a solution to run remote workloads with the same level of confidentiality and integrity as a local setup, while complementing data at-rest and in-transit protections: sensitive workloads would therefore be protected from untrustworthy CSPs and other CSP customers. As highlighted in this document, and despite the real security gains of the technology, such assertions come with many caveats, as there exists flaws and attacks that currently make Confidential Computing solutions less secure in general than a local setup. In any case, users of those solutions and security experts must dive in more detail into the inner workings of the two main claimed properties of Confidential Computing — *i.e.* the TEE and the attestation. This document proposes some clarifications concerning their limitations, provides guidance on secure usage, and outlines the residual risks to factor into the threat model.

Trusted Execution Environment (TEE)

TEE technologies aim at providing a secure and isolated execution environment, called an *enclave*, for sensitive workloads executions. The objective is to offer strong confidentiality and integrity guarantees for both the code and data loaded and processed inside this protected environment, while preventing unauthorised access or modification of these elements from outside entities.

This objective is enabled through several hardware components that form the *Manufacturer TCB*, enforcing secure memory and process isolation for enclaves. They include the *Central Processing Unit* (CPU) and its associated memory controller, coupled with a *Memory Encryption Engine* — typically meant for encrypting specific range of *Memory* (DRAM) pages with secret keys held in hardware and not available to the operating system or any other software, even running at the highest privilege level. Those hardware components may execute firmware (*e.g.* CPU microcode), provided and signed by the processor vendor, that is also considered part of the Manufacturer

TCB. Software running within the enclaves, on the other hand, is part of the *User-provided TCB*, designed by other parties and covered in a later section of this document. Figure 1 illustrates those different types of TCB within the architectural layers composing a Confidential Computing deployment[3].

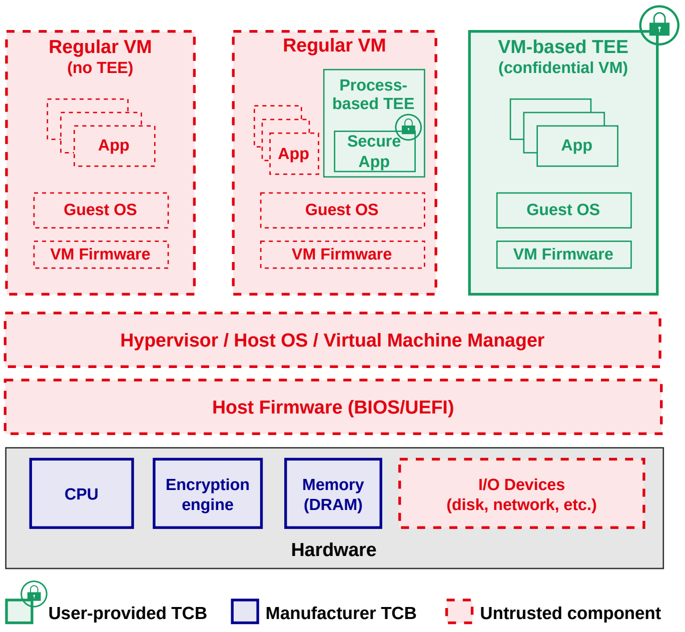

Figure 1: *Trusted Compute Base in a Confidential Computing environment*

TEE were first formalized as part of the Open Mobile Terminal Platform (OMTP) standard in 2006. Since 2010, GlobalPlatform, a technical standards organization partnering with the Trusted Computing Group (TCG), is responsible for driving the specifications of TEE system architectures, APIs and protection profiles for Common Criteria (CC) compliance for the industry. In the context of this document, we focus on TEE allowing *secure remote execution*, *i.e.* measurement and remote attestation of the code running in the enclave to ensure confidentiality and integrity on untrusted systems. This excludes TEE such as ARM TrustZone that does not implement this property. It should be noted that while several TEE have been certified by ANSSI using CC profiles, none of them allowed secure remote execution, let alone Confidential Computing. As a matter of fact, there exists no CC profile covering secure remote execution at the time of writing.

There are currently two leading architectural deployment models for secure remote execution:

- **Process-based TEE**, as popularized by the **Intel SGX** *(Software Guard eXtensions)* implementation, consist of processes split into two parts where one is considered secure and trusted, unlike the other. The trusted component resides in the encrypted memory (a **secure enclave**) and handles sensitive/confidential computations, while the untrusted component communicates with the OS and propagates I/O from encrypted memory to the rest of the system through predefined channels with strict checks on the size and type of data passing through. This smaller TCB is sometimes seen as an advantage as it is easier to review. However, the necessity to re-architect the applications according to these constraints led

[3] In practice, CSPs will often bundle their own software within the "user-provided TCB", making the trust boundaries of this figure hard or even impossible to enforce in some cases.

  to the development of frameworks to run entire applications or even containers within enclaves, negating the benefits of a reduced TCB. Performance issues (for I/O-intensive and memory-demanding workloads), programming restrictions and a lack of support for specialized hardware (such as GPUs) led to the development of VM-based TEEs which are now described.
- **VM-based TEEs** protect entire virtual machines running on top of an Hypervisor or a VMM (*Virtual Machine Monitor*), enabling the deployment of confidential Virtual Machines (cVMs) that leverage most of the traditional infrastructure for virtualization. This approach was introduced with the **AMD SEV** (*Secure Encrypted Virtualization*) technologies, and is now also proposed with the **Intel TDX** (*Trust Domain eXtension*) and **ARM CCA** (*Confidential Compute Architecture*) implementations. Separate hardware-protected ephemeral encryption keys are used for each VM in this model, therefore protecting the VMs from each other.

In recent years, manufacturers have decided to restrict Confidential Computing extensions to their server offering. Even though previous generations of SGX were available on consumer hardware, this document focuses on server hardware exclusively.

So far, process-based TEE are more popular in commercial deployments since they have been available for a decade. Despite a long history of reported security vulnerabilities [3], this longer availability gave them time to grow a more mature ecosystem and a better understanding of their limitations. More recently, VM-based TEE have gained popularity due to their increasing adoption within the CSP offers. Indeed, this technology is significantly more developper-friendly compared to process-based TEEs, as it allows applications to execute securely without any source code modification. It also paves the way towards trusting I/O devices, using authentication and attestation protocols such as SPDM. Most future developments will probably focus and occur on VM-based TEEs. However it is unclear at this stage when commercial support will reach parity with tooling available for process-based TEE, and therefore each software-TCB designer should review which model is more appropriate based on their specific security requirements.

Manufacturer TCB and User-provided TCB

Confidential Computing, regardless of its flavour, promises secure remote execution *i.e.* the ability to run code remotely and have it compute the expected result despite a privileged software adversary able to compromise the host OS, hypervisor, network, or persistent storage. Outside of this potentially-compromised stack lies the *Trusted Compute Base* (TCB). As mentionned before, the TCB in Confidential Computing deployments is split into two parts:

- the Manufacturer TCB: the hardware and associated firmware that make up the TEE executing the confidential workload;
- the User-provided TCB: the code of the confidential workload itself, running on top of the Manufacturer TCB.

In details, the main processor vendors all provide hardware-assisted isolation technologies to be used as a foundation to Confidential Computing stacks. This applies for instance to AMD (SEV-SNP cVM), Arm (CCA cVM) and Intel (SGX secure enclaves and TDX cVM) and the associated hardware and firmware form the basis for the Manufacturer TCB. Trust in these TCBs relies on documentation provided by the vendors, as well as the work of security researchers looking for vulnerabilities through a mix of white-box and black-box testing. As mentioned above, none of those platforms has been subjected to a Common Criteria evaluation at this stage; examples of known vulnerabilities are provided in the bibliography and referenced later in this document.

In a simplified view of Confidential Computing, the User-provided TCB is built by the user who controls the confidential workload. They first pick a provider offering a mature-enough TEE technology, and assume they can trust its Manufacturer to hold the security requirements against their threat model (detailed below). They then write their own workload, and can only rely on attestation at runtime to ensure their code and data are in principle securely executed on their hardware of choice — with no need to trust the CSP hosting it, assuming it does not collude with the Manufacturer.

The reality of real-world deployments is not that simple. Part of the code running in the User-provided TCB is often injected by the cloud-provider, especially in the case of cVMs: the firmware responsible for the early stages of booting the VM, as well as the virtual TPM (*Trusted Platform Module*) used to attest the later boot stages, are typically beyond the control of the user. Even in the more constrained case of secure enclaves (SGX), it is common to rely on off-the-shelf frameworks provided by third-parties to ease encapsulation of the workload. For this reason, the User-provided TCB is sometimes called the *Independent Software Vendor (ISV)* TCB: it is built with a combination of layers from various sources, some of which may not be measured in the attestation report. Application architecture, data flows, and supply-chain attacks should be carefully evaluated rather than blindly trusting a "confidential" offering. We provide more details in the following sections about attesting the full extent of the User-provided TCB, as well as reducing and hardening its attack surface.

It is also very rare for Cloud users to rely exclusively on VMs they control. A key advantage of CSP offers is to provide integrated services and architecture, for instance for identity access management (IAM), key management (KMS), database, or configuration and monitoring agent running inside the VM. In such mixed cases where user data is processed by CSP services, trust must be extended to the CSP and the TCB is effectively their entire environment. Most advantages of Confidential Computing become moot in this context.

To alleviate this concern, CSPs sometimes offer "confidential services" (not to be confused with Confidential Computing), supposedly running in secure enclaves or VMs. But a confidential service remains fully under the control of the provider, shared between users, and it is never possible for the user to meaningfully attest it. Confidential services should be seen as a mere good practice, defence-in-depth on the CSP side, and treated with the same level of caution as the non-confidential version of the same service.

Threat model

The user of a Confidential Computing technology must define their attacker model to decide what kind of architecture fits their security needs.

An attacker may be:

- **a tenant**: another tenant on a shared host (*e.g.* another VM or process). They may in principle access only their own resources but, in case of side-channel or isolation vulnerabilities, they may gather information about other tenants on the same physical host. If they manage to escape their sandbox, they may gain admin privilege level.
- **a malicious admin**: an administrator of the host computer system the workload is running on (*e.g.* a rogue employee or partner). They can control the creation and execution of workloads, including modifying them before they are started, and reading or writing their memory at runtime.
- **a physical attacker**: someone with physical access to the server (*e.g.* a malicious CSP). They may read and write the content of physical memory without any restriction, as well as

accessing and modifying any bus communication channel between the components of the host. In the most sophisticated cases, they can perform physical attacks to extract secrets from the TEE.
- **the manufacturer** or someone in their **supply-chain** (*e.g.* foundry, OEM): someone able to get a copy of the private keys stored in hardware, that underpin the security of the Manufacturer TCB. This category also includes attackers with a very high power to coerce suppliers.

As mentionned before, Confidential Computing is often presented by commercial providers as a solution to run remote workloads with the same level of confidentiality and integrity as a local setup, *i.e.* resistant to a physical attack. However, **physical attacks are explicitly out-of-scope of the security target defined by hardware vendors**. This means in particular that if a user is concerned about a cloud-provider conducting targeted attacks, instead of relying on a Confidential Computing approach they need to switch to a cloud-provider they trust, *i.e.* with strong counterparts or control capabilities, or use their own hardware with physical security protection measures. Likewise, the security of Confidential Computing assumes an uncompromised Manufacturer TCB: manufacturer and supply-chain attackers, including state-level ones, are thus explicitly out-of-scope.

Against an attack from a tenant, traditional isolation mechanisms (such as virtual machines) should ensure an adequate level of confidentiality and integrity. **Enabling confidential VMs (cVMs) is still recommended, if available, as a defence-in-depth layer**: it makes it harder to exploit side-channel and sandbox-escape vulnerabilities on the host. The user can verify its workload is running in confidential mode through remote attestation, or merely checking the system logs (if the threat model assumes an uncompromised host).

Against an attack from an admin, Confidential Computing is the only hardware-based protection mechanism currently available in commercial offerings. This is indeed the core value proposition of Confidential Computing: preventing privilege escalation as well as confidentiality and integrity violations by an untrusted hypervisor. However, two main caveats apply:

1. **the integrity of the entire Manufacturer and User-provided TCB must be ensured through remote attestation** because a compromised workload, or a genuine workload running outside of a TEE, could leak confidential data.
2. **confidential workloads must be hardened against specific attacks from the host** because the attack surface expands compared to a traditional model of trusted hypervisor and devices.

We detail why those important points are hard to put in practice, technically and operationally, in the next two sections.

A note about availability: an attacker privileged on the shared host or with physical access is obviously able to shutdown the whole system or a specific VM, resulting in a denial of service. This threat must be kept in mind when choosing a CSP, be it for traditional or confidential workloads. In this document, we focus solely on integrity and confidentiality threats specific to Confidential Computing.

Integrity: the need for attesting the entire TCB

Remote attestation serves two complementary purposes:

- **TEE integrity (Manufacturer TCB):** identifying the mode and state of the CPU executing the code, to verify that it is a genuine processor running with the latest security patches (firmware or microcode) and with confidential-computing features actually enabled;

- **code integrity (User-provided TCB):** ensuring the integrity of the code, *i.e.* that the running workload is the one expected by the user.

Both of those aspects are essential to preserve data confidentiality: TEE integrity because a genuine workload running outside of a TEE could have confidential data extracted by the hypervisor; and code integrity because a compromised workload could leak confidential data directly to the attacker.

TEE integrity is generally well-supported by solution providers. The attestation report contains information about the model of the CPU, its identity (to distinguish different CPUs by the same manufacturer), and current state (*e.g.* whether Confidential Computing is enabled). The report is then signed by a certificate whose secret key is buried in the hardware, and chained to a public certificate issued by the CPU vendor. Practical implementation is not easy to get right, because the minutiae of checking security options and firmware versions involve tedious and error-prone expert-level knowledge, but users may leverage external libraries taking care of the details. A third party can then verify those elements, and establish a secure channel of communication with the confidential workload, based on keys inaccessible to the host hypervisor. The system remains secure as long as the algorithms are sound, implemented correctly, and the secret keys do not leak — issues that have unfortunately happened in the past through hardware attacks [4, 5] or the use of weak cryptography [6].

But TEE integrity is meaningless from a security perspective if not tied with code integrity: by itself, it merely indicates that there exists a TEE that generated this report somewhere, not that the code of interest is actually running inside this TEE. The report could be captured in a genuine environment and then replayed outside of a TEE, or sent by compromised code running on the expected TEE. To prevent these attacks, attestation reports also contain a measurement of the User-provided TCB.

Code integrity is, unfortunately, much harder to verify properly for two reasons: not all the code running in the TEE is automatically measured in the attestation, and some of it may be outside of the control of the user. Upon setup of a confidential workload, the (untrusted) host copies the user-provided image in memory and hands it over to the TEE, which captures a hash of its content to later include in attestation reports. It is however possible to add more memory pages after startup, the content of which is not measured or reflected in the attestation report.[4] The usual solution to ensure full control of the executed code is a *trusted bootchain*: the initial memory pages are attested, and their code carefully reviewed to ensure that they only launch code that is itself trusted (*e.g.* verifying a cryptographic signature with a certificate embedded in the initial pages). As long as every step of the chain verifies the next step before executing it, the entire workload can be trusted. They can also record measurements of each step in an eventlog to be later verified by a remote-attestation procedure similar to the confidential workload attestation.

Attestation of every step of the bootchain is necessary to verify that the entire User-provided TCB has not been compromised by an admin attack, but it is impossible to perform on current cloud offerings for confidential VMs. The initial code running in the enclave is not under control of the user. Details vary per cloud provider, but none of the offers have reached a level of maturity where the full chain can be audited: the initial VM firmware is often not provided for review, or cannot be customised to include a certificate validating the next step; the bootloader or kernel cannot always be signed with a custom certificate; for legacy reasons, bootchain measurements

[4] The original version of SGX did not allow modifying the memory layout after startup but SGX 2.0 aligned enclave capabilities with those of cVMs and dropped memory integrity in favour of performance and dynamic memory management.

are stored in virtual TPMs, the code of which is also not available for review[5]; finally, bootchain verification usually stops after launching the OS (kernel), whereas the TCB encompasses the actual workload usually running in user space. Any of these gaps could allow a rogue administrator to inject compromised code subreptitiously [7]. Technical solutions exist to overcome each of those limitations [8], which can already be deployed by customers on bare-metal infrastructure; they still require expert engineering knowledge to design securely, but may become more common-place as they mature — if cloud providers evolve their offering towards more user control.

For process-based secure enclaves, the situation is generally better thanks to the model of reduced TCB, and it is possible to find cloud offerings that allow full control of the executed code. But porting existing applications, and designing a secure data flow between the unsecure application and the secure enclave, is complex. Hence, many third-party providers offer solutions based on a *library OS* to run the full workload inside the enclave — including their own, often unverified compatibility code. When relying on such solutions, **trustworthiness of the library OS and attacks on the software supply-chain should be carefully considered**. The architecture of the solution should always, at the very least, include some form of workload signature and verification by the user.

A final difficulty is that, regardless of the technology, **secret provisioning must be tied to remote attestation**. The code of the confidential workload is not encrypted at-rest (before it is executed). As a result, any secret it contains could be read by an attacker; hence the need to feed the secrets to the workload upon boot. But since a compromised workload could leak the secrets, the secret provider must first validate the code and TEE integrity. Therefore, the secret provider must only release the secrets after attestation verification of the Manufacturer and User-provided TCB. Figure 2 illustrates the overall architecture of such a solution. It is essential, when designing or buying a Confidential Computing solution, to specify the procedures to generate, store, access and exchange the secrets used to encrypt and decrypt the confidential data at-rest and in-transit. Solution providers should strive to develop, standardise and adopt open protocols, such as RA-TLS to tie TLS-encrypted connections with remote-attestation evidence.

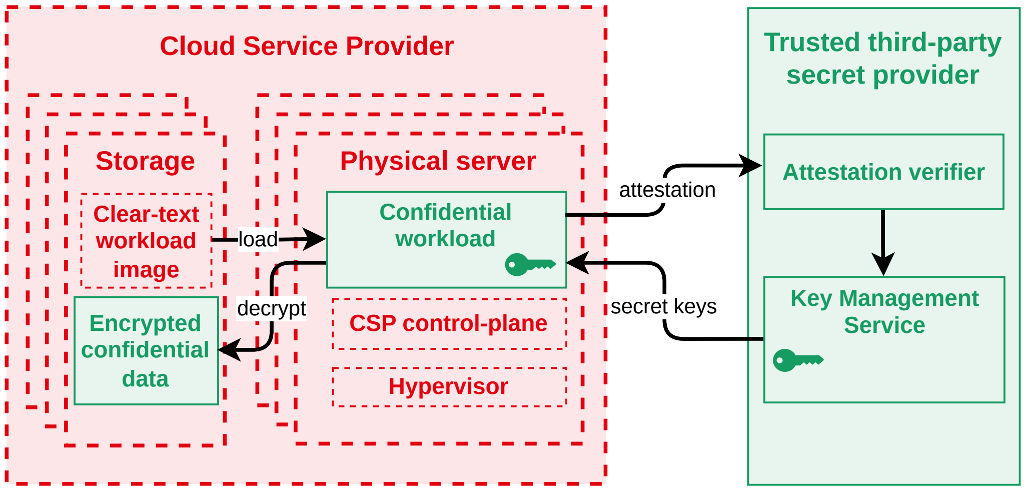

Figure 2: *Tying secret provisioning to remote attestation*

[5] Some CSPs even run the vTPM outside of the confidential VM, alongside the hypervisor, which makes them unverifiable devices that could not be trusted even if their source code was available.

As mentioned above, Cloud users often rely on services from their CSP. For instance, it is not uncommon for the CSP to also act as a secret provider through key management services. This is yet another reason why, in most cases, the threat model needs to trust the CSP explicitly, and Confidential Computing can at best be used as defence-in-depth against infrastructure vulnerabilities or attacks by other tenants. Alternatively, a secret provider distinct from the CSP and supporting remote attestation must be used, or operated by the user themselves. However, this brings additional concerns about the availability of the entire system. The failure of any of the two infrastructures (CSP or non-CSP), or the link in-between, would result in an outage of the whole system. Note that "Bring Your Own Key" (BYOK) approaches do not solve this issue, as the CSP still needs to be trusted to use the keys provided by the user[6].

Hardening confidential workloads against the host

Since the threat model of Confidential Computing includes a hostile host OS and hypervisor, the workloads need not only be attested, but also hardened against confidentiality and integrity attacks from the outside. It is indeed not enough to run a system hardened against classical attacks to protect against attacks from a malicious hypervisor: a kernel that has been designed to be secure on bare-metal will assume that the hardware is trusted and its drivers need not be concerned about hostile behaviour. Once ported to a virtual environment, the attack surface broadens. Public security assessments of real-world deployments provide good examples of actual scenarios to consider [9].

Against integrity attacks, software components interacting with the host should treat all inputs as untrusted. This includes PCI drivers, ACPI tables, interrupt handling, port I/O, MMIO regions and calls from the host to the confidential workload. Both the guest kernel and the initial firmware launching it should be hardened [10].

Over the last few years, many integrity attacks have been demonstrated against confidential VMs, in particular using unexpected interrupts injected from the hypervisor to modify the control-flow of the guest [11, 12]. Process-based enclaves are also commonly vulnerable to attacks on untrusted inputs [13], and most "library OS" frameworks have been found vulnerable to signal-injection attacks [14].

Confidentiality attacks use similar techniques, but are even harder to defend against because secrets can leak through side-channels which are not guarded by integrity-protection features. For instance, enclaves can be single-stepped [15] to detect patterns with instruction-level accuracy; performance counters [16] and interrupt [17], and even single-stepping mitigations [18] have been leveraged to leak information. In some cases [19], software mitigations cannot fully protect against the risk and hardware migrations are necessary. Side-channel attacks on confidentiality, leveraging techniques such as *Prime+Probe* measurement, page-level access patterns and performance counter tracking are not even considered mitigated by the vendors themselves in their threat model [20].

Finally, if hardware attacks are added to the threat model then, as noted before, cryptographic key material may be extracted from the TEE, or replay attacks be mounted, that void all guarantees provided by remote attestation.

It is worth noting that since their emergence 10 years ago, the protections offered by Confidential Computing technologies have been regularly circumvented. Therefore, they can not be fully

[6] One could envision cVM attestations verified directly by trusted third-party Hardware Security Modules (HSM), as part of their access-control policy, but no such attesting HSM is available at the time of writing, let alone integrated in public cloud offerings.

trusted. This does not mean that Confidential Computing technologies are useless, but providers should be ready to deploy firmware mitigations as soon as they become available to recover trust in the Manufacturer TCB, and users should include this failure mode in their threat model (*e.g.* adding a verification of the TCB version number in attestation reports). Despite these past vulnerabilities, the memory hardening provided by these technologies remains a good defense-in-depth measure against a number of threats such as cold-boot attack, side-channel attack, memory leak or protection against a malware running with elevated privileges.

Recommandations for users

As a rule of thumb, users running cloud workloads should:

- **Enable cVM when available, as a layer of defense-in-depth**, while keeping in mind that it provides only limited guarantees when used on its own (without remote attestation, secret-provisioning, VM hardening, etc.). It is still a net benefit in most cases, because it makes some kind of attacks harder, such as side-channels from other tenants and direct memory read with admin privileges.
- **Not rely on Confidential Computing if the cloud provider is considered untrusted or potentially hostile**: against an active attacker, there are too many known vulnerabilities to effectively guard against. In such situations, switching to a trusted CSP or using dedicated bare-metal hardware in a controled physical environment is a prerequisite.

Users of Confidential Computing should also:

- Define their threat model, and based on a solid understanding of the technologies, design clearly the boundaries and content of their TCB, how it is attested, how secrets are provisioned and used, and how a breach of integrity would be detected.
- Apply traditional security measures to all the services running inside confidential VMs: Confidential Computing is not a silver bullet, and will do nothing to protect against a misconfigured or vulnerable guest application (*e.g.* weak passwords or SQL injection).
- Design their workloads to encompass the continuum of at-rest, in-transit and in-use protections, taking care of encrypting all network and disk communications that leave the VM to shield them against hypervisor interception and modification.

When the security model makes Confidential Computing mandatory, *e.g.* when active attacks from the host are part of the threat model, users should imperatively:

- Use attestation of the entire TCB: manufacturer-provided (hardware and related firmware), user-provided (VM image) and CSP-provided (VM firmware, vTPM, etc.).
- Provision secrets within the enclave only upon successful attestation.
- Deploy software and firmware mitigations against the latest known attacks from the host.

Some specific trade-offs should be kept in mind:

- usability: users who want to avoid running CSP-controlled privileged agents inside their workloads may lose remote administration features such as KVM over IP or image provisioning and configuration on first boot.
- complexity: users currently need to either develop home-grown, error-prone solutions giving them full control of their confidential workload, or rely on opaque infrastructure from third-party providers that cannot be meaningfully verified and attested.
- efficiency : mitigations against integrity and confidentiality attacks, when they exist, come at a cost. They may not be available in default VM images, and require expert knowledge to enable and validate.

Recommandations for service providers

Providers of Confidential Computing offerings should:

- **Allow attestation of the entire bootchain** for confidential VMs, from the firmware to the user-space. This requires providing or allowing:
  - open-source VM firmware (and virtual TPM if applicable), measured by the Manufacturer TCB;
  - measurements of the kernel, initrd and boot parameters by the firmware, stored either in the Manufacturer hardware or in virtual TPM (vTPM) running within the enclave;
  - measurements of the root filesystem by the kernel to ensure user-space integrity;
  - user-customisation, or at least reproducible builds, for all those components.
- **Contribute tools to help users building their own confidential workloads**, that can be run standalone in a user-controlled environment or CI/CD.
- **Contribute tools to help users deploy and and maintain confidential workloads**, in particular to manage their lifecycle as well as the related secrets.
- **Promote the security evaluation of the Confidential Computing solutions**.
- **Contribute to, and adopt, standard protocols for remote attestation** tied with effective verification of the TCB (including user-built artifacts) and secret provisioning.

References

[1] Agence nationale de la sécurité des systèmes d'information. *Prestataires de services d'informatique en nuage (SecNumCloud)*. Référentiel d'exigences. Version 3.2. Mar. 8, 2022. URL: https://cyber.gouv.fr/sites/default/files/document/secnumcloud-referentiel-exigences-v3.2.pdf.

[2] *DoD 5200.28-STD: Trusted Computer System Evaluation Criteria (TCSEC) Aka the "Orange Book"*. Department of Defense (DoD), Dec. 1985. URL: http://csrc.nist.gov/publications/history/dod85.pdf.

[3] Stephan Van Schaik et al. "SoK: SGX.Fail: How Stuff Gets eXposed". In: *IEEE Symp. on Security and Privacy (SP 2024)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2024, pp. 4143–4162. DOI: 10.1109/SP54263.2024.00260.

[4] Robert Buhren et al. "One Glitch to Rule Them All: Fault Injection Attacks Against AMD's Secure Encrypted Virtualization". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS '21. New York, NY, USA: Association for Computing Machinery, Nov. 13, 2021, pp. 2875–2889. DOI: 10.1145/3460120.3484779.

[5] Jesse De Meulemeester et al. "BadRAM: Practical Memory Aliasing Attacks on Trusted Execution Environments". In: *2025 IEEE Symposium on Security and Privacy (SP)*. 2025 IEEE Symposium on Security and Privacy (SP). May 2025, pp. 4117–4135. DOI: 10.1109/SP61157.2025.00104.

[6] Google Security Team. *AMD: Microcode Signature Verification Vulnerability (CVE-2024-56161)*. Sept. 25, 2024. URL: https://github.com/google/security-research/security/advisories/GHSA-4xq7-4mgh-gp6w.

[7] Gianluca Scopelliti, Christoph Baumann, and Jan Tobias Mühlberg. "Understanding Trust Relationships in Cloud-Based Confidential Computing". In: *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). July 2024, pp. 169–176. DOI: 10.1109/EuroSPW61312.2024.00023.

[8] Anna Galanou et al. "Trustworthy Confidential Virtual Machines for the Masses". In: *Proceedings of the 24th International Middleware Conference*. Middleware '23. New York, NY, USA: Association for Computing Machinery, Nov. 27, 2023, pp. 316–328. DOI: 10.1145/3590140.3629124.

[9] Tjaden Hess et al. *Meta WhatsApp Private Processing*. Security Assessment with Fix Review. Trail of Bits, Aug. 26, 2025. URL: https://github.com/trailofbits/publications/blob/master/reviews/2025-08-meta-whatsapp-privateprocessing-securityreview.pdf.

[10] *Trust Domain Security Guidance for Developers*. Intel. URL: https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/best-practices/trusted-domain-security-guidance-for-developers.html.

[11] Benedict Schlüter et al. "WeSee: Using Malicious #VC Interrupts to Break AMD SEV-SNP". In: *2024 IEEE Symposium on Security and Privacy (SP)*. 2024 IEEE Symposium on Security and Privacy (SP). San Francisco, CA, USA: IEEE, May 19, 2024, pp. 4220–4238. DOI: 10.1109/sp54263.2024.00262.

[12] Benedict Schlüter et al. "HECKLER: Breaking Confidential VMs with Malicious Interrupts". In: 33rd USENIX Security Symposium (USENIX Security 24). 2024, pp. 3459–3476. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/schl%C3%BCter.

[13] Mustakimur Rahman Khandaker et al. "COIN Attacks: On Insecurity of Enclave Untrusted Interfaces in SGX". In: *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS '20: Architectural Support for Programming Languages and Operating Systems. Lausanne Switzerland: ACM, Mar. 9, 2020, pp. 971–985. DOI: 10.1145/3373376.3378486.

[14] Supraja Sridhara et al. "SIGY: Breaking Intel SGX Enclaves with Malicious Exceptions & Signals". In: *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*. ASIA CCS'25. Ha Noi, Vietnam: Association for Computing Machinery, 2025. arXiv: 2404.13998 [cs].

[15] Jo Van Bulck, Frank Piessens, and Raoul Strackx. "SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control". In: *Proceedings of the 2nd Workshop on System Software for Trusted Execution*. SysTEX'17. New York, NY, USA: Association for Computing Machinery, Oct. 28, 2017, pp. 1–6. DOI: 10.1145/3152701.3152706.

[16] "CounterSEVeillance: Performance-Counter Attacks on AMD SEV-SNP". In: *Proceedings of the Network and Distributed System Security Symposium 2025*. NDSS 2025. San Diego, CA, USA, Feb. 2025. DOI: 10.14722/ndss.2025.241038.

[17] Ruiyi Zhang et al. "CacheWarp: Software-based Fault Injection Using Selective State Reset". In: 33rd USENIX Security Symposium (USENIX Security 24). 2024, pp. 1135–1151. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/zhang-ruiyi.

[18] Luca Wilke, Florian Sieck, and Thomas Eisenbarth. "TDXdown: Single-Stepping and Instruction Counting Attacks against Intel TDX". In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. CCS '24. New York, NY, USA: Association for Computing Machinery, Dec. 9, 2024, pp. 79–93. DOI: 10.1145/3658644.3690230.

[19] Mengyuan Li et al. "CIPHERLEAKS: Breaking Constant-time Cryptography on AMD SEV via the Ciphertext Side Channel". In: 30th USENIX Security Symposium (USENIX Security 21). 2021, pp. 717–732. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/li-mengyuan.

[20] AMD. *AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More*. White Paper. Jan. 2020, p. 20. URL: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf.