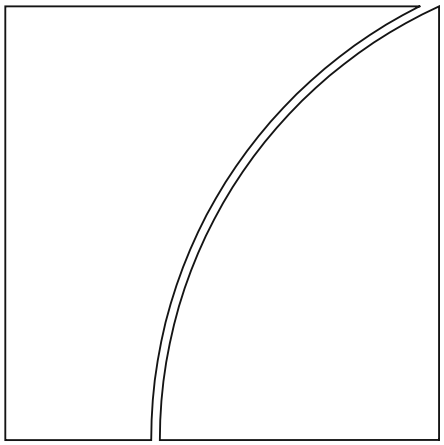


BIS Papers

No 173

Trusted execution environments for central banks

by Christof Fetzer, Co-Pierre Georg, Jack Ho, Bingle Kruger, Julius Wetzel and Jakub Demski

Monetary and Economic Department

September 2026

JEL classification: C81; L86; O33

Keywords: trusted execution environments; confidential computing; remote attestation; systemic stress testing; fraud detection; AML/CFT compliance; digital-asset wallets; cybersecurity; cross-jurisdiction data collaboration

The views expressed in this publication are those of the authors and do not necessarily reflect the views of the BIS or its member central banks. The authors acknowledge the use of artificial intelligence in research assistance and editorial refinement. Responsibility for all errors remains with the authors.

This publication is available on the BIS website (www.bis.org).

ISSN 1682-7651 (online)
ISBN 978-92-9259-982-9 (online)

Trusted execution environments for central banks*

Christof Fetzer,† Co-Pierre Georg,‡ Jack Ho,§ Bingle Kruger,‡ Julius Wetzel† and Jakub Demski§

Trusted execution environments (TEEs) offer a hardware-anchored way to protect "data in use" by enforcing code authenticity, runtime integrity and confidentiality within a well-bounded trusted computing base. This paper considers what TEEs can responsibly deliver for central banks, the conditions under which they are most effective and how they can be deployed without widening who must be trusted. It takes a balanced view, outlining core assurances and boundaries, highlighting implementation and governance dependencies and situating TEEs alongside alternative approaches to confidential computing. The discussion links these foundations to central bank needs – such as cross-border data collaboration, supervisory analytics and operational resilience – and summarises practical considerations in performance, procurement, portability and auditability. While TEEs can enable deeper analysis and stronger resilience, their effectiveness is conditional and they should be embedded in disciplined engineering and evidence-based governance. The paper closes with a forward look at emerging directions and their implications for medium-term planning.

Keywords: trusted execution environments; confidential computing; remote attestation; systemic stress testing; fraud detection; AML/CFT compliance; digital-asset wallets; cybersecurity; cross-jurisdiction data collaboration

JEL classification: C81; L86; O33

* We thank Mike Alonso, Miguel Diaz, Henry Holden and Kevin Tsui for their helpful comments. References to individual firms and projects are for illustrative purposes and should not be construed as any statement on the soundness or the legal or regulatory status of any firm or project.

† Technische Universität Dresden.

‡ Frankfurt School of Management & Finance.

§ Bank for International Settlements.

Contents

1. Introduction

Central banks safeguard monetary and financial stability, oversee payment and settlement systems and, in many jurisdictions, supervise or contribute to the supervision of financial institutions. These mandates increasingly require timely analysis of granular information that is often commercially sensitive, legally protected or distributed across institutions and jurisdictions. The analytical value of such information is highest when it can be examined at the level of individual institutions, counterparties, instruments or transactions. Yet the same granularity that makes the data useful also heightens confidentiality, legal and operational risks. This tension is particularly acute in cross-border settings, where direct sharing of microdata may be neither lawful nor prudent, even when joint analysis would improve visibility of liquidity flows, risk concentrations and contagion channels.

Encryption has substantially reduced confidentiality risks when data are stored or transmitted (Sabt et al (2015)), but it does not by itself protect data while they are being processed. In conventional systems, data normally have to be decrypted before computation, which creates a "data in use" gap. This is not only a confidentiality problem. It is also an integrity and verifiability problem, because a party that wishes to check whether a computation was performed correctly may otherwise need direct access to the data or the ability to re-run the computation, thereby widening the set of stakeholders exposed to sensitive inputs.

Trusted execution environments (TEEs) are relevant in part because they aim to narrow that trade-off: they seek to protect confidentiality during processing while also providing evidence about the code and state under which the computation occurred. Precisely when data are most valuable for analysis, they may be exposed to software, administrators, cloud operators or infrastructure components that are not intended to access them. In practice, this gap can force institutions to choose between analytical depth and confidentiality. They may rely on coarse aggregates, delayed reporting or bilateral legal arrangements, all of which can reduce timeliness and limit the ability to detect emerging risks.

TEEs are one response to this problem. A TEE is a hardware-backed, integrity-protected execution context in which approved code can process sensitive data while the surrounding operating system (OS), hypervisor (the software that hosts and isolates virtual machines) and infrastructure administrators are excluded from reading protected memory or altering protected execution. Put simply, a TEE creates a sealed and monitored space inside a computer. Data may be decrypted and processed within that space, but the host environment should not be able to inspect the plaintext or modify the computation. This does not remove the need for legal controls, governance or audit. Rather, it changes the technical basis of trust: instead of relying solely on assurances from an operator, participants can require cryptographic evidence of what code is running, how it is configured and whether the platform state satisfies an agreed policy.

Three assurances are central to this model. The first is code authenticity: the platform records cryptographic measurements of the software image and configuration so that other parties can verify what has been loaded. The second is

runtime integrity: hardware-enforced isolation and memory integrity protections restrict attempts by the host to redirect control flow, remap memory or tamper with the protected state. The third is confidentiality of data in use: plaintext data and keys are intended to appear only inside the protected boundary and not in host-accessible memory. These assurances are bounded by the trusted computing base (TCB), meaning the hardware, firmware and code that must be correct for the security policy to hold. The smaller and better understood this base is, the easier it is to assess and audit the resulting system (Gregor et al (2020); Narayanan et al (2023)).

Remote attestation operationalises these assurances. In an attestation protocol, a TEE produces signed evidence of its code, configuration and relevant platform state. A relying party compares this evidence with an agreed allow list of approved software hashes, firmware levels and configuration flags. Only if the evidence satisfies policy are secrets, credentials or encrypted inputs released. In practical deployments, the attested identity can also be bound to mutually authenticated Transport Layer Security (mTLS). This means that endpoints authenticate not merely as named services, but as specific approved software running on approved platforms. Session keys can then be generated and used inside the protected context, so that ciphertext decrypts only within the attested boundary.

Two server class styles are used in practice. Enclaves[1] protect narrowly scoped modules within an application, such as a signing routine, decryption function or policy enforcement component. Their advantage is a relatively small and auditable TCB. Confidential virtual machines (CVMs), by contrast, protect an entire guest operating system and its applications from the host. Their advantage is compatibility: existing analytics platforms, databases or payment components can often be moved into a confidential boundary with fewer code changes. Many practical architectures combine the two styles. A CVM may exclude the cloud operator and hypervisor from the broader workload, while enclaves inside the guest may protect the highest value keys or "crown jewel" routines (Graph 1).

For central banks, the relevance of TEEs is not confined to a single application. They may support confidential cross-border statistical analysis, systemic stress testing, fraud detection, AML/CFT and sanctions screening, digital asset wallets and operational cyber resilience. In each case, the common proposition is the same: institutions may be able to share insights without sharing raw data. A participating institution can contribute encrypted inputs to an attested environment; approved computation can occur inside the protected boundary; and only policy-approved outputs – such as aggregates, exception flags, approve/deny decisions or one-way audit tokens – may leave the TEE. This pattern is consistent with the broader direction of central bank work on privacy-enhancing technologies, which recognises that privacy, security, scalability and compliance are interdependent design choices rather than independent features. Recent central bank analysis on retail CBDC system design, for example, notes that privacy-enhancing technologies may offer additional

[1] In practical terms, an enclave is a small, sealed "room" inside an application where only the most sensitive logic and keys are kept. The application passes only the minimum necessary inputs into the enclave through a narrow, well audited interface; the enclave performs the critical operation (eg decrypt, risk score, sign) and returns only tightly controlled outputs (eg an approve/deny decision, an aggregate or a signature). Because the protected code is small, the TCB is small too, which reduces the attack surface and simplifies review and patching.

design flexibility but can also introduce latency, complexity and reliability concerns (BIS-CBG (2024)).

The case for TEEs must nevertheless be made with care. TEEs are not a complete security solution and their assurances are conditional. Side channel attacks may infer information from timing, cache contention or other indirect signals. Host-mediated attacks may manipulate inputs, system calls or input/output paths. Rollback may reintroduce vulnerable software or stale state. Supply chain compromise may affect the code before it reaches the TEE and vendor-specific attestation mechanisms may create operational dependencies. These risks do not invalidate the technology, but they do mean that TEEs should be treated as one component in a layered architecture that also includes robust attestation governance, side channel-aware engineering, signed and reproducible builds, hardware security modules (HSMs), independent audit logs, timely revocation and clear legal controls.

For decision-makers, the relative importance of confidentiality and integrity will vary by use case. In cross-institution data collaboration, confidentiality is often the immediate barrier to sharing granular inputs, because disclosure of institution-level data may be unlawful, destabilising or commercially damaging. In operational settings such as payments, wallets or supervisory decisions, integrity may be equally important, because an incorrect computation, unauthorised state change or forged output can directly affect market functioning or public trust. In practice, the two are linked: a system that preserves confidentiality but cannot give credible evidence of correct execution may still be unacceptable, while a system that improves verifiability only by exposing data to more actors may defeat the purpose of confidential computation.

This paper therefore advances a qualified thesis. TEEs may enable central banks and regulated institutions to conduct deeper analysis and strengthen operational resilience without widening the set of actors trusted with plaintext data, if adoption is disciplined and evidence-based. The remainder of the paper develops this argument in five steps. Section 2 explains the technical foundations of TEEs, including remote attestation, mTLS, enclaves, CVMs and adjacent privacy-enhancing technologies. Section 3 maps these foundations to central bank use cases. Section 4 examines threats, limitations, performance and governance. Section 5 concludes by setting out a practical adoption path that places attestation, policy binding and auditability at the centre of confidential computation.

2. Technical foundations of trusted execution environments

Understanding TEEs is easier if the technology is introduced in the order in which trust is established in practice. A confidential computing deployment does not begin with an abstract claim that data are "protected in use". It begins with a chain of technical and governance controls: the platform measures what has been loaded; isolated execution protects selected code and data during processing; sealed storage binds secrets to approved states; remote attestation exposes those states to external

verifiers; and policy-bound egress[2] determines what information, if any, may leave the protected boundary. In cross-institution settings, these technical controls must then be paired with operational ones, such as approval of software hashes, governance of allow lists, revocation handling and secure logging.

This sequencing matters for central banks because the core problem is rarely protection of one institution's application in isolation. More often, the challenge is to make a specific cross-organisational computation acceptable to multiple parties that do not share infrastructure, legal mandates or operational control. TEEs can help only if isolation, attestation, key release, logging and policy enforcement work as one chain. This section therefore begins with the building blocks and lineage of TEEs, then defines their guarantees and boundaries, distinguishes enclaves from CVMs, explains the programming model, develops the role of remote attestation and mTLS, and finally situates TEEs alongside adjacent privacy-enhancing technologies and current implementation patterns.

2.1 Building blocks and lineage

TEE designs build on a longer tradition of secure system architecture in which protection is achieved by separating security domains and minimising the amount of code that must be trusted. Separation kernels and high-assurance operating systems pursued the same principle: if a trusted base is small, specific and auditable, it is easier to reason about than a large, general purpose software stack (Rushby (1981); Ames (1983); NSA (2007)). Modern TEEs implement that principle in commodity hardware. Instead of requiring the host OS or hypervisor to be trusted, they use processor support for measurement, isolation, memory protection and attestation to protect selected computations even in an otherwise untrusted environment (Chakrabarti et al (2020)).

Measured boot provides the first link in the chain. At startup, the platform records cryptographic measurements of firmware, boot components, OS images or other relevant software. A measurement is typically a hash, meaning a fixed-length digest that changes if the measured file or configuration changes. These measurements do not prove that the software is free from defects. They do, however, allow a verifier to determine whether the system is running a known and approved version. For central bank deployments, this matters because legal or supervisory approval can be tied to exact software versions rather than to general descriptions of a system.

Isolated execution provides the second link. In an enclave, the processor restricts access to a protected memory region and mediates entry into the protected code. In a CVM, memory associated with the guest is encrypted and protected from the host and hypervisor. Although the implementation differs by vendor and architecture, the common purpose is to prevent host-level software from reading or modifying the protected state. This shifts the basis of confidentiality from administrative separation to hardware-enforced separation. It is particularly relevant in cloud or multi-

[2] Egress refers here to the release of data or results from the protected environment to an external recipient or system.

institution environments, where the infrastructure operator may be trusted to provide availability but not to access plaintext data.

Sealed storage connects runtime protection to persistence. A TEE can bind a secret to a measured software and platform state, so that the secret can be recovered only when the expected state is present. This mechanism is useful for keys, credentials, policy bundles and recovery material. For example, a digital asset wallet may seal a signing key so that it unseals only when the approved wallet software and policy version are loaded. Similarly, a data collaboration platform may seal decryption credentials to an approved analytics image. Sealing therefore prevents a key from being copied to a different binary or restored under a downgraded configuration, if rollback protection and freshness mechanisms are also in place.

Remote attestation exposes these internal facts to external parties. A TEE produces signed evidence of its measured state, and a verifier checks that evidence against an approved policy. This step is what turns local isolation into a coordination mechanism. Without attestation, a participant may believe that its data are being processed in a protected environment but cannot verify it independently. With attestation, the participant can require evidence before releasing encrypted inputs or credentials. In cross-border central bank settings, this feature is central: trust can be anchored in verifiable measurements rather than in the discretion of a single operator.

Trusted input/output completes the chain but is often the hardest part to implement. Even if computation is protected internally, data may be exposed if channels, devices or peripherals are not bound to the attested context. A host could otherwise redirect traffic, substitute inputs or observe plaintext through virtualised devices. Cryptographic channel binding, mTLS identities derived from attested states and device assignment controls help ensure that plaintext enters and leaves only through approved paths. These controls do not remove all risks at the boundary, but they make the path of data movement explicit and auditable.

Together, measured boot, isolated execution, sealed storage, remote attestation and trusted input/output form the technical lineage of TEE systems. Their significance for central banks is that they convert broad institutional claims into verifiable conditions. Data are not released because an operator asserts that a service is secure; they are released because an attested service presents evidence that it matches an approved software, configuration and platform state. The next subsection explains how this evidence can be bound to communication channels and policy decisions.

2.2 Definitions and guarantees

A TEE is a protected execution context, anchored in hardware, that aims to preserve the confidentiality and integrity of code and data while computation is taking place. The essential idea is not that computation occurs directly on ciphertext in the way associated with fully homomorphic encryption (FHE). Rather, plaintext may appear during computation, but only inside a protected context whose memory and execution are shielded from the host environment (Asokan et al (2014); Sabt et al (2015)).

A TEE can reduce the need to trust the infrastructure operator with plaintext data and can provide evidence that approved code and configuration were loaded before secrets were released. It does not, however, automatically guarantee that the code is

correct, that the output is privacy-preserving or that the wider system is free from inference risks (ICO (2026); Jauernig et al (2020); Muñoz et al (2023)). In other words, TEEs are a mechanism for controlling where plaintext appears and which measured software may process it; they are not, by themselves, a full privacy or correctness solution.

Three guarantees are therefore best understood as conditional assurances. The first is code authenticity. At initialisation, the platform measures the code and configuration, usually by computing cryptographic hashes of binaries, firmware and selected settings. These measurements allow a verifier to ask whether the running software corresponds to an approved version. The second is runtime integrity. Hardware isolation, page-table protections and memory integrity mechanisms are intended to prevent the host from modifying protected memory, remapping it maliciously or steering execution away from the approved control flow. The third is confidentiality of "data in use". A memory encryption engine keeps protected memory encrypted outside the processor's trusted boundary, so that a host-level attacker or cloud operator cannot simply read plaintext from main memory.

These guarantees are bounded by the TCB, comprising the hardware, firmware, microcode, attestation mechanism, trusted runtime and any application code inside the protected boundary. A larger or poorly understood TCB weakens practical assurance. This is why architectural choices matter. An enclave that contains only a small signing routine may be easier to reason about than a CVM that includes an entire guest OS, although the latter may be much easier to deploy. At the same time, the relationship is not mechanical. A broader and more general purpose environment may attract more operational scrutiny, more testing and more repeated audit than a narrowly scoped but rarely reviewed component, much as widely used cryptographic primitives often benefit from extensive scrutiny despite their complexity. For now, the balance of current practice still tends to favour smaller TCBs for high-assurance functions, given the continuing history of TEE vulnerabilities and the difficulty of validating large trusted stacks. However, that preference is contingent rather than absolute and may evolve as deployment patterns, evidence tooling and assurance practices mature.

Remote attestation translates these local guarantees into inter-institutional trust. A TEE can produce signed evidence describing the measured software, configuration and platform state. A relying party verifies that evidence against an allow list of approved measurements and releases secrets only if the evidence satisfies policy. In central bank settings, this mechanism is particularly important because the parties may not share infrastructure, jurisdiction or supervisory powers. Attestation makes trust portable: a participant does not need to rely only on the operator's assertion that the right code is running; it can require verifiable evidence before providing sensitive inputs.

These assurances are therefore conditional and bounded. Side channels may reveal information through timing, cache contention or other indirect signals. Host-mediated attacks may manipulate system call responses, inputs or virtualised devices (Checkoway and Shacham (2013)). Rollback may restore stale software or state unless freshness is enforced. Supply chain compromise may introduce unsafe code before measurement takes place. For this reason, TEEs are most persuasive when they are embedded in a broader governance model that combines attestation, revocation, output controls, secure logging and independent audit.

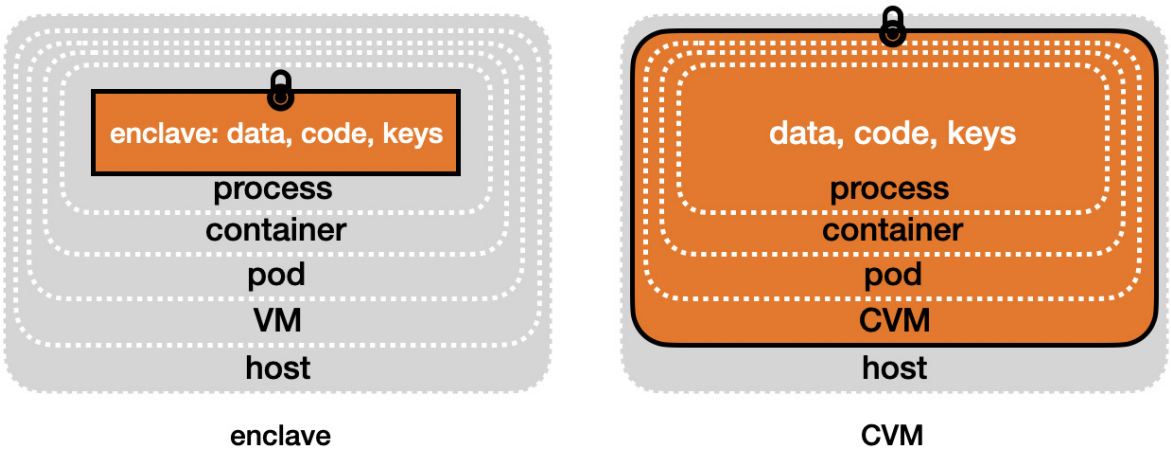
2.3 Architectural styles: enclaves and confidential virtual machines

TEE deployments generally follow one of two architectural styles: enclaves or CVMs. Both aim to protect computation from the host environment, but they draw the protected boundary in different places. This boundary determines not only the security properties of the system, but also its operational cost, portability and ease of adoption.

An enclave protects a narrowly defined region of code and memory inside an application. The rest of the application remains outside the protected boundary and communicates with the enclave through carefully defined entry and exit points. In practical terms, the enclave should contain only the most sensitive logic and keys: for example, a decryption function, a transaction signing routine, a sanctions screening decision rule or an egress function[3] that enforces output policy. Because the protected component is small, it can often be reviewed, tested and patched more easily than a full application stack. This makes enclaves attractive where the main objective is to minimise the TCB and provide high assurance for a specific operation.

The same narrow boundary also creates engineering demands. The application must be partitioned into trusted and untrusted components, and developers must decide what data cross the boundary, how often and in what format. Each boundary crossing may create latency and may increase the opportunity for mistakes. If the enclave relies on the untrusted host for file access, networking or system calls, the enclave must treat those responses as potentially adversarial. Enclave programming therefore rewards disciplined interface design: inputs should be minimised; outputs should be tightly constrained and any host-provided information should be validated before use (Graph 1).

Protected boundaries (enclave vs confidential VM) Graph 1



An enclave confines a small, encrypted memory region, while a CVM protects an entire virtual machine.

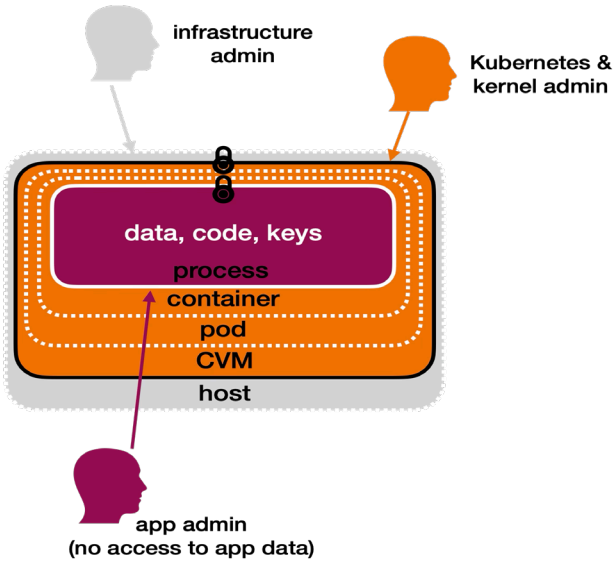
[3] An egress routine governs what information may leave the protected boundary and under which conditions.

A CVM takes a broader approach. Instead of isolating a small module, it protects an entire guest OS and its applications from the host, hypervisor and cloud operator. The guest can often run existing software with limited modification, which makes CVMs attractive for "lift-and-shift" workloads such as analytics platforms, databases, model execution environments or payment system components. This compatibility is operationally important for central banks and financial institutions, whose critical systems may depend on established software stacks, certification processes and operational runbooks.

The broader boundary of a CVM comes with a different assurance profile. By placing an entire guest OS inside the protected context, a CVM reduces exposure to the infrastructure operator but increases the amount of code that must be trusted inside the guest. A vulnerability in the guest kernel, application framework or internal service may still compromise data within the CVM, even though the host cannot read its memory. In this sense, CVMs shift the trust boundary rather than eliminating trust need. Their value lies in excluding the platform operator and host from plaintext access, while allowing conventional software to run in a more protected environment.

For many central bank workloads, the most robust pattern may be layered. A CVM can protect the broader workload from the host and cloud operator, while enclaves inside the guest protect the most sensitive routines and keys from compromise of the guest itself. For example, a stress testing platform might run its data ingestion and simulation pipeline inside a CVM, while the decryption keys, policy enforcement routine and final signing step reside in enclaves (Graph 2). If the host is compromised, the CVM protects the guest. If the guest is partially compromised, the enclave can still restrict access to the most sensitive secrets and outputs. This layered design is not costless, but it aligns the width of the protected boundary with the sensitivity of each function.

Trusted computing base within a confidential VM — Graph 2



Isolating application services within a CVM can reduce the size of the TCB and improve resource utilisation through better sharing. Placing "crown jewel" routines in enclaves within the CVM further shrinks the TCB for the most sensitive code.

The choice between enclaves and CVMs should therefore be framed as a design trade-off rather than a product choice. Enclaves are better suited to narrowly scoped, high-assurance functions where the TCB must be small. CVMs are better suited to whole-system confidentiality and rapid adoption of existing workloads. Hybrid designs can combine these strengths, but they also require clear evidence governance so that each layer's attestation, update and key release policies remain consistent. This architectural distinction prepares the ground for the next question: how applications should be structured so that protected boundaries remain meaningful in practice.

2.4 Programming and system model

Programming for TEEs begins with a boundary decision. The designer must decide which code and data genuinely require protection from the host environment, and which parts of the system can remain outside. This decision is not merely an implementation detail. It determines the size of the TCB, the complexity of review and the residual risks that remain if surrounding infrastructure is compromised. For central bank workloads, the protected boundary should normally contain only privacy-critical logic, cryptographic keys and policy enforcement routines, while OS, orchestration layers, storage services and administrators are treated as untrusted unless explicitly included in the attested state.

In enclave-based applications, this principle leads to a deliberately narrow programming model. The enclave should contain the smallest feasible module that performs the sensitive operation, such as decrypting inputs, applying an agreed risk model, signing a transaction or enforcing an output policy. The wider application passes inputs into the enclave through defined entry points and receives constrained outputs through defined exit points. This separation is valuable because a smaller enclave is easier to audit, test and patch. It also reduces the amount of code that must be written in a side channel-aware manner. The trade-off is that enclave applications require careful partitioning: each call into or out of the enclave crosses a security boundary, and excessive crossings may increase latency as well as enlarge the surface on which mistakes can occur.

Even where the host cannot read enclave memory, it still mediates many interactions with the outside world. Enclave code may rely on the host for networking, scheduling, file access or system call results, which means that these interfaces must be treated as adversarial. Inputs should therefore be authenticated, freshness should be verified where relevant and host-supplied metadata should not be trusted unless they are cryptographically protected. The practical lesson is that a TEE narrows the trusted boundary, but it does not remove the need for defensive interface design at that boundary.

Memory management is also part of the security model. Some enclave platforms provide only limited protected memory (for example, the Enclave Page Cache[4] stores enclave pages, which are decrypted and integrity-checked on access). When an application exceeds this capacity, pages may be moved between protected and unprotected memory under encryption and integrity protection. Although this

[4] On Intel SGX, the Enclave Page Cache is a secure, encrypted region of DRAM that stores enclave code and data; when capacity is exceeded, paging increases latency and may amplify side channel signals.

preserves the basic confidentiality claim, it can increase tail latency and may introduce observable access patterns that are relevant for side channel analysis. In practice, enclave workloads should keep hot data structures inside protected memory, avoid unnecessary copies and minimise data-dependent memory access where sensitive values are involved. This is particularly important for cryptographic routines and matching algorithms, where timing or access pattern differences may reveal information even though plaintext is never directly exposed.

CVM-based applications follow a broader but more familiar system model. A full guest OS and its applications run inside a protected virtual machine, while the host and hypervisor are excluded from reading guest memory. This makes CVMs attractive for existing analytics platforms, databases, payment components and model execution environments, because institutions can often reuse established toolchains and operational procedures. The benefit is compatibility and speed of adoption. The cost is that the guest operating system, libraries and application stack become part of the effective trust boundary. A CVM therefore protects against the cloud operator or compromised host, but it does not by itself protect one component inside the guest from another compromised component inside the same guest.

For this reason, microservice design can complement both enclaves and CVMs. A microservice architecture decomposes a system into small services with well defined responsibilities and interfaces. When each sensitive service runs in its own TEE, the compromise of one service need not expose the full workload. This design also aligns with attestation because each service can present evidence of its own code, configuration and platform state before receiving credentials or data. In a central bank data collaboration platform, for example, ingestion, reconciliation, model execution and output enforcement could be separated into distinct attested services. Such separation makes it easier to specify which component is allowed to see which data and which component is allowed to release which result.

The programming model is therefore inseparable from governance. A TEE does not decide by itself what information may leave the protected boundary. That decision must be encoded in application logic and bound to an approved policy. In practice, the egress routine inside the TEE should enforce minimum-group thresholds, suppression rules, query budgets, rate limits and permitted output formats. It should also attach a policy identifier or signed audit record to each result so that downstream recipients can verify which version of the policy was applied. This design links the technical boundary to the institutional objective: participants can contribute sensitive data because both computation and disclosure are governed by measured, reviewable and auditable code.

2.5 Remote attestation and policy binding, with mTLS

Remote attestation is the mechanism through which a TEE proves its state to a remote party. Its importance lies in the fact that confidentiality in a multi-institution system depends not only on encryption, but also on knowing where decryption will occur and which code will control the resulting plaintext. Attestation provides evidence for that decision. It allows a relying party to ask: is this the approved software, running with the approved configuration, on a platform that satisfies the agreed security policy?

The process can be described as a sequence. First, the platform measures relevant code, firmware and configuration. Second, the TEE produces an attestation report or evidence statement that includes those measurements and selected platform attributes. Third, the evidence is signed or endorsed through a vendor, platform or independent attestation mechanism.[5] Finally, a verifier compares the evidence with an allow list that records approved image hashes, firmware minima, configuration flags and policy versions. If the evidence matches, the verifier may release credentials, keys or encrypted inputs. If it does not match, the service should fail closed or continue only in a non-sensitive mode.

This sequence mitigates several named risks. It reduces stale trust risk because credentials can be short-lived and renewed only after fresh evidence is presented. It mitigates downgrade risk because older or vulnerable images can be removed from the allow list. It mitigates configuration tampering because secrets are released only when the measured configuration matches policy. It also reduces identity misbinding[6] because a service identity can be linked to a measured state rather than merely to a network address or operator-controlled certificate. These benefits depend, however, on disciplined governance of the allow list, revocation information and trust anchors. If an outdated image remains approved, or if revocation information is unavailable, attestation may provide a false sense of assurance.

mTLS extends attestation into the communication channel. In ordinary TLS, a client authenticates a server and negotiates encrypted communication. In mTLS, both endpoints authenticate each other. In a confidential computing setting, this mutual authentication can be tied to attested states. After attestation succeeds, each endpoint receives or uses a short-lived credential that represents not only "this service name", but "this approved code and configuration on this approved platform". During the mTLS handshake, the endpoints authenticate with those credentials and derive fresh session keys, typically using an ephemeral Diffie-Hellman exchange (Diffie and Hellman (2022)). The important point is that private key material and session keys should be generated and used inside the protected context, so the host and hypervisor cannot learn them.

This chain explains how attestation supports in-TEE-only decryption. Encrypted data are not sent merely to a machine or cloud account. They are sent to a channel whose endpoint has been bound to an attested software state. The decryption keys or session keys are available only inside the protected boundary, and the application code that uses plaintext has already been measured and approved. In a cross-border data collaboration, for example, one central bank may release encrypted microdata only after verifying that the receiving environment is running the agreed analytics code and output policy in the approved measured state. This design does not make the computation inherently correct, but it ensures that data are released only to the environment that participants agreed to trust.

Policy binding extends the same logic from identity to behaviour. A policy is a machine-verifiable specification of what computation may occur and what outputs may be released. It may define permitted inputs, approved model versions, minimum-group thresholds, suppression rules, query budgets, rate limits, logging requirements

[5] For example, Intel's DCAP supports attestation for SGX; Intel TDX and AMD SEV-SNP typically expose evidence via a virtual TPM (vTPM).

[6] These controls reduce identity misbinding, that is, the risk of associating secrets with an unintended binary.

and recipient permissions. When the policy is included in the measured configuration, any unauthorised change to it alters the measurement and prevents secrets from being released. The TEE's egress routine then enforces the policy before any result leaves the protected boundary. This is the mechanism that turns a TEE from a protected compute container into a governed confidential computation system.

The governance implications are substantial. Institutions must decide who can approve software hashes and update policies, and how revocations are distributed and exceptions are documented. They must also retain evidence so that auditors can reconstruct which software ran, which policy applied and which outputs were released. Append-only logs, hash chaining and independent timestamping can strengthen tamper evidence without requiring a distributed ledger. Where public auditability or shared rights management is required, a ledger may be used to register digests or execution receipts, but plaintext inputs and detailed outputs should remain off-ledger. The baseline should therefore be evidence-rich governance, not blockchain by default.

In multi-institution settings, these governance decisions cannot be left to informal operational practice. If code hashes, approved firmware versions or policy identifiers are circulated through unsigned email, spreadsheets or ad hoc messaging channels, the integrity of the whole attestation process is weakened before any cryptography is applied. A stale spreadsheet, an unsigned hash list or an unverified emergency update can result in secrets being released to the wrong code or to a revoked state. Secure logging and authenticated distribution of approval data are therefore not ancillary controls. They are part of the trust fabric. In an inter-central bank collaboration, this means that application development changes in practical ways: participating institutions need a joint process for approving code versions, agreeing which TEE implementations and vendors are eligible, validating measurement formats across platforms, distributing revocations and retaining tamper-evident records of those decisions.

The public key infrastructure behind this process is part of the trust boundary in practice, even if it is not part of the TEE hardware. Attestation evidence must be signed by keys that verifiers recognise, credentials must be issued only after evidence has been checked, and revocation information must be distributed quickly enough to prevent known vulnerable states from continuing to receive secrets. Allow lists therefore require the same operational discipline as other critical security assets. They should have defined owners, dual-control approval for changes, signed version histories, emergency revocation procedures and independent audit trails. A stale allow list, a compromised certificate authority or a delayed revocation feed can undermine confidentiality even when the TEE itself behaves as designed.

Residual risks remain even when attestation and mTLS are well implemented. Side channels may leak information from approved code; application defects may produce incorrect or over-disclosing outputs; supply chain compromise may affect a binary before it is measured and approved; evidence services and revocation channels may become availability dependencies. These limitations do not weaken the value of attestation; rather, they clarify its role. Attestation answers the question of whether the expected code and configuration are running. It does not, by itself, prove that the code is safe, privacy-preserving or free from defects.

Box A

Coordination patterns in multi-institution deployments

Coordination and auditability in multi-institution settings do not require blockchains, but they do require more than attestation alone. The practical baseline is a governed distribution process for approved code hashes, policy versions and revocations, supported by signed allow lists and append-only logs. Participants release keys only to TEEs that produce valid attestation evidence matching those approved states.

This model matters because many real-world failures arise not from cryptographic weakness, but from weak operational distribution of trust material. If one institution approves a new image digest while another continues to rely on an outdated spreadsheet, or if emergency revocations are distributed informally and without signatures, then the attestation chain becomes inconsistent across parties. Secure logging addresses this by recording attestation events, allow-list changes, policy approvals and key release decisions in a form that is tamper-evident and independently reviewable. Hash chaining and periodic timestamping strengthen this audit trail further.

In practice, inter-central bank application development under TEEs therefore requires joint approval of software releases, agreement on acceptable TEE vendors or evidence formats, and a shared understanding of who is authorised to rotate keys, revoke measurements and approve exceptions. These are governance choices as much as technical ones.

A distributed ledger or smart contract may be useful where participants require a shared, independently verifiable record of rights and execution receipts and do not wish to rely on a single log operator. Even then, plaintext inputs, detailed outputs and sensitive configuration data should remain off-ledger. The ledger, where used, records governance facts; it does not perform the confidential computation itself.

2.6 Adjacent technologies and comparative suitability

TEEs sit within a wider family of privacy-enhancing and confidential computing technologies. These approaches are sometimes discussed as substitutes, but they solve different parts of the problem and rely on different assumptions. For central bank use cases, the relevant question is not which technology is most advanced in the abstract. It is which technology best matches the computation, data volume, latency requirement, governance model and acceptable trust assumptions.

Multi-party computation (MPC) allows several parties to compute a function jointly without any one party revealing its private input to the others. This can be powerful when the computation can be expressed in protocol-friendly form, such as private set intersection, threshold signing or certain linear algebra tasks. MPC reduces reliance on a single hardware vendor or operator, but it introduces coordination and liveness requirements: several parties must remain online and follow the protocol, and performance may degrade for complex control flow, large joins or stateful analytics. In a central bank setting, MPC may therefore be attractive for narrowly scoped matching or deduplication tasks, but less suitable for large, iterative stress-testing models unless the model can be reformulated efficiently (Evans et al (2018)).

FHE takes a different approach. It allows specified computations to be performed directly on encrypted data, so the computing party need not see plaintext. This provides a strong confidentiality property, but current performance remains highly sensitive to the structure of the computation. In practice, FHE is best suited to

computations that can be expressed as well defined algebraic operations on encrypted values, especially where batching is possible and high latency is acceptable. Threshold FHE distributes decryption authority among several parties, which can reduce unilateral control over outputs, but it also adds governance and operational complexity (Acar et al (2018); Chillotti et al (2020); Carpov et al (2020)). For time-sensitive central bank analytics, FHE may therefore be valuable for narrow sub-routines rather than as a general replacement for TEE-based computation.

Zero-knowledge proofs (ZKPs) address a different question again: how one party can convince another that a statement about data or computation is true without revealing the underlying data. A prover can, for example, show that an output was computed according to a specified rule or that a compliance threshold was met, without disclosing the underlying data set. This is valuable for verification, but it is distinct from confidential multi-party computation. ZKPs can help others check correctness or policy compliance, yet they do not by themselves provide a shared environment in which several actors contribute confidential inputs for joint computation. In that respect, ZKPs are typically complements to TEEs or MPC rather than substitutes for them (Goldwasser et al (2015)).

TEEs differ from these cryptographic approaches less because they alone can run "ordinary code", and more because they generally preserve the conventional execution model of existing software more closely. Generic MPC frameworks and ZKP compilers are increasingly able to target familiar programming abstractions, but they still require the computation to be expressed in ways that fit their underlying trust and performance models. TEEs, by contrast, are often easier to integrate with iterative algorithms, branching logic, databases, model inference and stateful workflows that already exist in conventional software stacks. Their performance overheads are usually concentrated at start-up, attestation, input/output and protection-boundary crossings rather than in every arithmetic step. The trade-off is that they introduce hardware and vendor trust, and that those trust assumptions must be governed explicitly.

The most credible designs may combine these techniques rather than treat them as mutually exclusive. A TEE can orchestrate a workflow and hold policy state, while MPC performs a private set-intersection step. A TEE-hosted egress routine can generate a signed statement that a query budget was respected, while a ZKP provides independent verification of a simple output condition. Long-lived root keys may remain in HSMs, while TEEs host richer application logic and enforce policy before using short-lived credentials. Such hybrid designs should be justified by specific risk-reduction objectives, because each additional primitive adds implementation, monitoring and failure modes.

The comparison supports a cautious, workload-specific conclusion rather than a universal ranking. For low-latency or stateful analytics on large sensitive data sets, TEEs may offer a practical near-term balance between confidentiality, functional flexibility and operational feasibility, especially where existing software stacks need to be preserved. For narrowly scoped matching or joint computation under stronger non-hardware trust preferences, MPC may be preferable. For specialised encrypted arithmetic where latency is less critical, FHE may be appropriate. For verifiable compliance statements or checks on released outputs, ZKPs can complement either approach. The important point for decision-makers is that no single technology

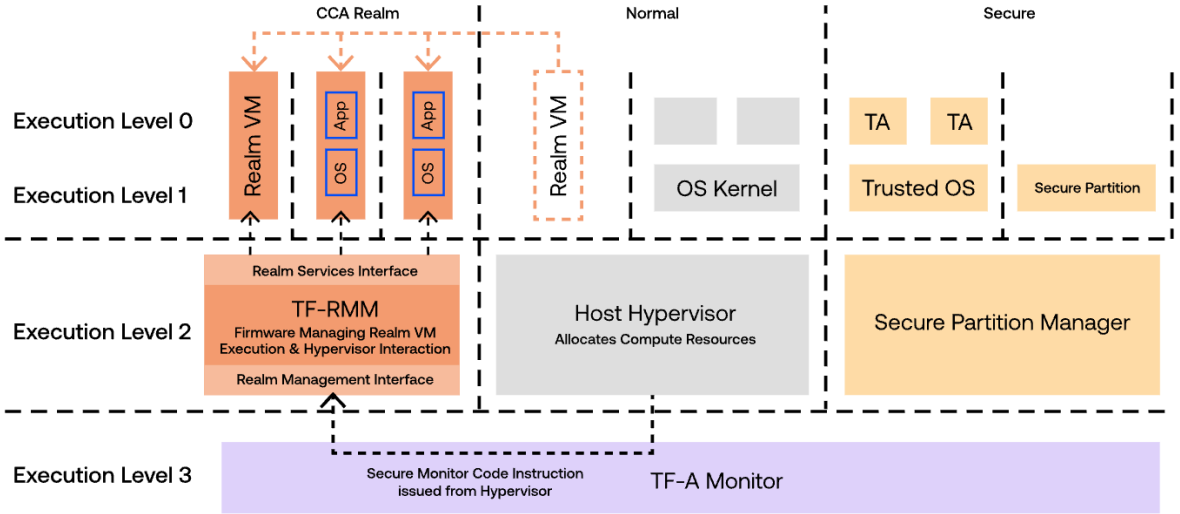
dominates across all dimensions. The choice depends on the computation, data volume, latency tolerance, operational complexity and trust assumptions involved.

2.7 Implementations, portability and illustrative deployments

TEE capabilities are now available across major processor families and public clouds, although their security models, evidence formats and maturity differ. Intel supports enclave-style computation through Software Guard Extensions (SGX) and CVM isolation through Trust Domain Extensions (TDX). AMD provides CVM capabilities via Secure Encrypted Virtualisation-Secure Nested Paging (SEV-SNP). ARM's Confidential Compute Architecture (CCA) introduces realm-based isolation for ARM systems (Graph 3). Large cloud providers expose these features through confidential-computing instance types and managed services. The direction of travel is clear: confidential computing is moving from specialised deployments towards mainstream infrastructure. At the same time, decision-makers should recognise that, outside secure storage, attested key handling and a relatively small number of high-sensitivity niches, TEEs remain an emerging rather than settled application pattern. Much of the literature remains academic or prototype-oriented, and production use beyond tightly bounded security functions is still developing.

The recent history of TEEs is therefore mixed. On the one hand, they are actively used in mobile secure elements, cloud confidential computing services and selected regulated sector environments. On the other hand, the field has also been shaped by a long sequence of documented vulnerabilities, redesigns and, in some cases, retrenchment in specific product lines. This combination of increasing operational interest and unresolved assurance challenges is central to understanding their present role.

Confidential virtual machine architecture (ARM CCA realms) Graph 3



Guest memory is encrypted and integrity-protected; host and hypervisor are excluded from the TCB.

Source: ARM (2025).

Portability has both software and governance dimensions. At the software layer, applications should place privacy-critical logic behind narrow, well defined interfaces so that this logic can be ported across TEE implementations with minimal change. At the governance layer, attestation evidence produced by different vendors must be interpreted consistently. Verifiers need to understand the semantics of measurements, the minimum acceptable firmware levels, which configuration flags are security-relevant and which trust anchors are recognised. Absent normalised evidence and policies, a multi-vendor strategy may breed false confidence: technical diversity reduces monoculture risk, but inconsistent verification can allow the weakest evidence path to define the effective perimeter.

A pragmatic adoption path is incremental. Initial pilots on isolated machines or single-service CVMs establish patterns for attestation, key release, logging and incident response without the complexity of a full confidential cluster. Selected services can then be moved into CVMs where compatibility matters, while the most sensitive routines remain in enclaves or HSMs. Only once evidence governance is established should confidential workloads be integrated with container orchestration and microservices. This sequencing reduces the risk that operational convenience overtakes assurance. It also generates audit artefacts early, including software bills of materials (SBOMs[7]), signed builds, reproducible build evidence, attestation logs, policy signatures and key release records.

From a maturity perspective, isolated machine or single-service CVM deployments are a prudent starting point. They let teams exercise attestation, evidence verification, key release, logging and incident response without the added complexity of a full confidential cluster. Containerised TEE workloads and Kubernetes-based orchestration can improve scalability and operational familiarity, but they also increase the number of moving parts, whose configuration, evidence and update cadence must be governed. For central bank pilots, portability and cluster orchestration should therefore follow – not precede – an established evidence and policy life cycle.

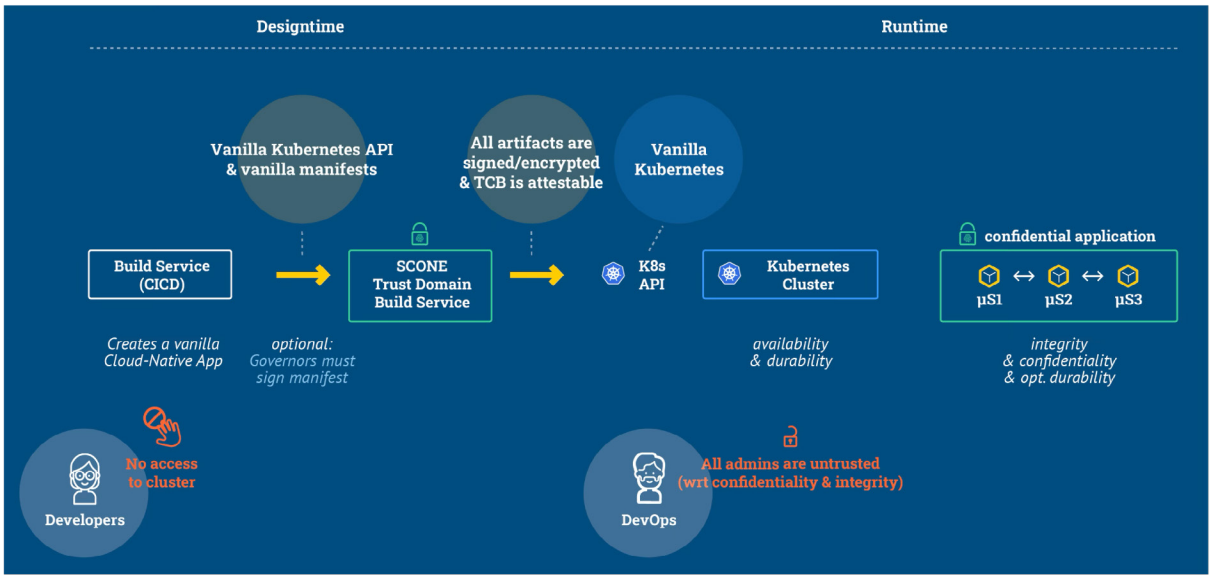
Secure computation in health systems

Health system infrastructure provides a useful analogue for central bank confidential computation because it combines sensitive personal data, statutory controls, cloud operational requirements and multi-organisation governance. In Germany, gematik[8] maintains extensive specifications for the digital health infrastructure, including security, public key infrastructure (PKI) and application-level services related to the electronic patient record and electronic prescription. These specifications illustrate the governance and operational demands of sensitive multi-organisation infrastructure: attestation, key release, update control, auditability and operator exclusion (Graph 4).

[7] An SBOM is a formal inventory of the software components and dependencies included in an application.

[8] https://gemspec.gematik.de/.

Secure computation with gematik Graph 4



Illustrative confidential computing deployment pattern in a cloud-native health system setting.

Source: gematik specifications and related materials.

The technical pattern is informative because it shows how confidential computing can be integrated with cloud-native operations. Developers may continue to define services using standard Kubernetes manifests,[9] while a confidential computing build and deployment process can encrypt and sign application artefacts, bind them to approved measurements and release secrets only to attested workloads. In such a model, orchestration frameworks continue to provide scheduling, availability and resilience, while TEEs provide confidentiality and integrity for the protected service boundary (Arnautov et al (2016)).

For central banks, the example is useful because it brings into view both the promise and the operational burden of confidential computing in a regulated, multi-organisation setting. It shows how operator exclusion and policy binding can be achieved at the service level, but it also makes clear that the hard part is not only enclave or CVM technology. It is the surrounding governance of updates, revocation, attestation evidence, audit rights and cross-organisational trust.

The example also highlights operational burdens. Firmware and microcode updates must be deployed promptly; attestation and revocation services become availability dependencies; evidence formats and semantics differ across hardware vendors; and side channel-aware engineering remains necessary where sensitive values might influence timing or memory access. These points support a staged adoption that tests the full life cycle – build provenance, evidence verification, policy rollover, incident response and audit access – before scaling out.

[9] Kubernetes manifests are declarative configuration files (usually YAML or JSON) that describe the desired state of resources in a cluster – eg deployments, services and roles.

Illustrative architecture: confidential credit bureau data-sharing

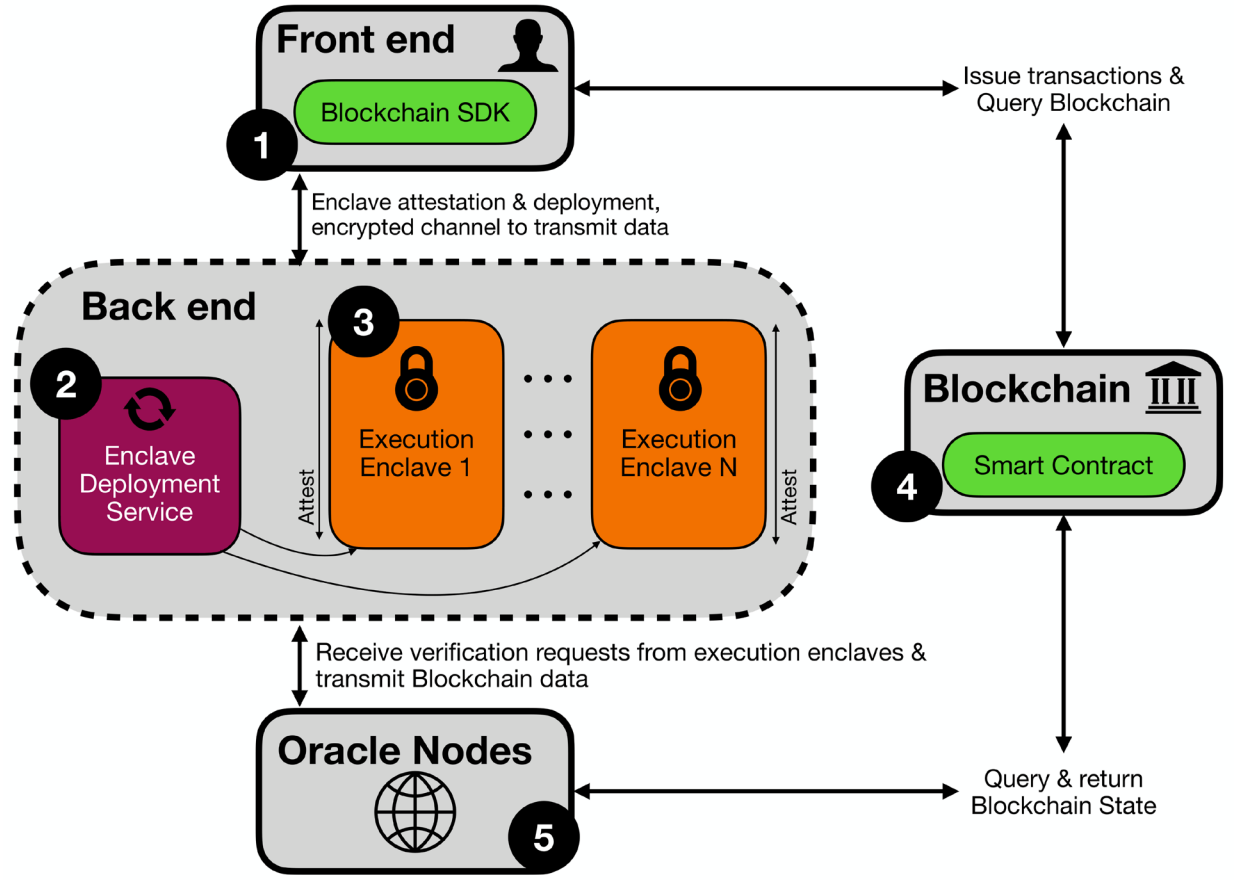
The following architecture is a conceptual illustration of how TEE-based confidential collaboration could work in a credit bureau setting. In such an entity, lenders may collectively benefit from shared detection of credit risk, fraud or identity signals, yet raw customer records cannot be exposed to other lenders, platform operators or unauthorised parties. A TEE-based design addresses this tension by allowing approved computation over encrypted inputs while limiting egress to authorised outputs.

In a baseline off-chain design, participating institutions would agree on a set of approved analyses and sign the corresponding code hashes and policy versions. A deployment service would launch an attested execution environment; each participant would verify its evidence before releasing encrypted inputs. A configuration service would provide the current allow list and revocation information. The TEE would proceed only if its code, configuration and platform state matched the approved policy. Outputs would be restricted to artefacts such as risk scores, anomaly flags, aggregates or one-way audit tokens. Raw records would remain sealed, and an attested log would record which approved computation ran, under which version and with which type of output.

A ledger-based variant may be useful where parties require a shared, tamper-evident record of rights and execution receipts without relying on a single log operator. In this design, participants can use a blockchain SDK to register rights to run specific analyses; each right is a token or registry entry bound to an exact code hash[10] and recorded on-ledger for coordination and audit. The ledger should not contain plaintext inputs, detailed outputs or sensitive operational metadata; its role is coordination and auditability, not computation. Confidentiality remains anchored in remote attestation, attestation-gated key release and egress controls. Concretely, an enclave deployment service launches the secure execution environment and performs remote attestation; the enclave produces signed evidence of its code, configuration and relevant platform state; other parties verify this evidence against an agreed allow list; and only then is an encrypted channel established and any data permitted to flow, so secrets move only to an environment that can prove it is in the approved state. A smart contract can validate submitted tokens, check requested code hashes against authorised analyses and record each attested execution event, while an oracle supplies current allow-list entries and revocations; equivalently, an off-chain configuration service can perform these roles. Execution receipts can be anchored on-ledger, binding "who computed what, when and under which version" to a tamper-evident record, while all plaintext processing remains confined to the attested TEE (Graph 5).

[10] A one-way function that maps any input to a fixed-length hash value. Altering the input changes the hash value; it is infeasible to reconstruct the input from the hash value or to find two different inputs with the same hash value.

Architecture for confidential credit bureau data-sharing Graph 5



Dashed lines show logical links and arrows indicate communication flows.

This architecture illustrates both the promise and the residual risks of confidential collaboration. It reduces the need for bilateral raw data-sharing and binds computation to verifiable code versions, yet it does not eliminate inference risk from repeated or adaptive queries, data quality differences across participants or false matches in pseudonymised identifiers and device fingerprints. It also creates operational dependencies on credential issuance, evidence verification and configuration services, and can be sensitive to denial-of-service or stale revocation information. These limitations are important because they show that confidential computing does not remove the need for output controls, policy-bound egress and attested logging under clear governance and audit.

As a conceptual illustration, this architecture clarifies both the promise and the residual risks of confidential collaboration. A TEE may protect memory from the host, but central bank use cases require more than protected memory. They require evidence that the right code is running, governance over which policies are approved, cryptographic channels that bind data release to measured states and egress controls that prevent raw or inferentially revealing information from leaving the protected boundary. Section 3 applies this chain to central bank functions in which the value of granular analysis is high, but the costs of raw data-sharing are prohibitive.

3. Use cases of trusted execution environments for central banks

TEEs' relevance for central banks follows from two related but distinct problems. One is the hardening of sensitive applications controlled by a single organisation, such as key management, payment processing or internal analytics. The other is confidential processing across organisational boundaries, where several institutions need to contribute sensitive data or rely on a jointly approved computation without widening access to the underlying records. TEEs can serve both roles, but the second is more demanding because it depends not only on application hardening but also on attestation, joint approval of code and policy, secure distribution of trust material and auditable coordination across institutions.

The use cases in this section therefore follow a common pattern. First, one or more institutions hold sensitive data whose analytical value increases when combined or examined at a finer level of detail. Second, direct sharing of those records is constrained by law, confidentiality obligations, commercial sensitivity or operational risk. Third, a TEE can serve as a protected compute environment in which the agreed computation is performed. Fourthly, remote attestation and policy-bound key release allow participants to verify the software, configuration and platform state before releasing inputs. Finally, output controls restrict results to aggregates, exception flags, approve/deny decisions, risk scores or one-way audit tokens. This pattern does not remove the need for legal authority or governance, but it can reduce the number of actors who must be trusted.

3.1 Central bank data collaborations – official statistics

Official statistics are a natural starting point because central banks often collect data that are most informative at the level of individual institutions, counterparties or instruments, while publication and cross-border sharing normally occur at higher levels of aggregation. Aggregation protects confidentiality, but it can also obscure the very structures that matter for analysis. Liquidity channels, risk concentrations and contagion pathways may depend on bilateral exposures or institution-level heterogeneity that are not visible in published aggregates (Cetorelli and Goldberg (2012)).

International banking statistics (IBS) illustrate the trade-off. National central banks and supervisory authorities collect granular information on cross-border claims and liabilities, while the public series are necessarily aggregated. Researchers have used such data to analyse the global banking network and systemic vulnerabilities (Minoiu and Reyes (2013); Cerutti et al (2015)), but aggregation can conceal how risks are distributed across individual banks and counterparties (Shin (2012); Bruno and Shin (2015)). Where bilateral exposures are unobserved, network reconstruction methods (eg maximum entropy) can be informative but may smooth away concentration risk, introducing uncertainty into contagion estimates (Anand et al (2015)). In other words, the confidentiality-preserving publication layer may weaken precisely the analytical features needed for systemic risk assessment.

A TEE-based collaboration would address this problem by allowing participating authorities to contribute encrypted institution-level data to an attested environment.

Before any data are released, each participant verifies that the environment is running the agreed analytics code, the approved configuration and the current output policy. The TEE can then compute network measures, exposure matrices, concentration indicators or exception flags at the required level of granularity, while the egress routine releases only policy-approved outputs. These outputs might include jurisdiction-level aggregates, sectoral exposure matrices, systemic importance indicators, outlier flags or one-way audit tokens. Raw institution-level records would remain sealed from operators, infrastructure providers and other participants.

The value of this design is not that it makes confidential data unconstrained. Legal mandates, data-sharing agreements and statistical confidentiality rules would still determine whether the collaboration is permissible. Rather, TEEs can provide a technical mechanism that makes those constraints enforceable and auditable. Minimum-group thresholds, suppression rules and query budgets can be encoded into the measured policy; attested logs can record which code version produced which output type; and mTLS identities derived from attested states can ensure that encrypted inputs are released only to approved software. The same pattern could support analysis of securities holdings, derivative exposures, large exposures, cross-border payment flows and regulatory arbitrage, where granular data are analytically valuable but direct sharing is restricted (Houston et al (2012); Aiyar et al (2012)).

3.2 Systemic stress testing

Systemic stress testing requires a detailed understanding of how shocks propagate through interconnected balance sheets, funding markets and common asset holdings. The most informative exercises are often bottom-up: they use institution-level information on claims, liabilities, collateral, liquidity buffers, derivative exposures and cash flow obligations. Such data are highly sensitive (Gai and Kapadia (2010); Greenwood et al (2015); Glasserman and Young (2016)). They may reveal business strategies, counterparty relationships and vulnerabilities that could be destabilising if disclosed. As a result, stress testing exercises may rely on aggregated, delayed or incomplete inputs, which protects confidentiality but limits the precision of the results.

TEEs can support a more granular model without requiring participating institutions to disclose raw exposures to one another or to the infrastructure operator. In a centralised design, a governed stress testing platform would launch an attested computation environment for each approved scenario. Participating institutions would verify the evidence, confirm that the model and output policy match the agreed version and provide encrypted inputs. Inside the TEE, the model could calculate payment shortfalls, liquidity strains, fire sale losses or clearing vectors under the scenario. Outputs would be restricted to supervisory artefacts such as breach lists, loss distributions, critical node indicators and aggregate contagion summaries.

This approach is particularly relevant for network-based stress testing. Frameworks such as Eisenberg and Noe's (2001) clearing model show how the distress of one institution may affect others through interbank obligations and payment priorities. Yet the practical application of such models depends on granular bilateral exposures, which are often the most sensitive data in the exercise. A TEE allows the bilateral data to be used without making them visible outside the protected boundary. For example, the model could identify institutions whose liquidity buffers

are breached under a margin shock, or estimate aggregate losses by sector and jurisdiction, without releasing the full matrix of bilateral exposures.

The same environment can also improve data integrity. If institution A reports a liability to institution B, and institution B reports the corresponding claim, the TEE can reconcile the two submissions and flag discrepancies without exposing either party's full balance sheet to the other. This cross-validation can raise confidence in inputs and may deter some forms of strategic misreporting, although it should not be treated as a substitute for supervisory judgment or enforcement powers. Attested one-way audit tokens could allow supervisors to follow up on discrepancies or threshold breaches without revealing unnecessary transaction-level detail.

The main policy significance is that TEEs may allow stress tests to become more frequent, more granular and more collaborative. They could support industry-led scenario exercises, cross-border supervisory simulations or rapid assessments during market stress. However, their use would need to be governed carefully. Scenario definitions, model versions, output categories and query limits should be approved in advance and bound to the measured policy. Repeated or adaptive scenario queries could otherwise create inference risks, especially if participants can compare outputs across small changes in assumptions. A credible TEE-based stress testing framework therefore combines protected computation with disciplined egress control and auditability.

3.3 Fraud detection and prevention

Fraud detection is a collaborative problem because fraudulent behaviour often spans institutions. A device fingerprint, identity attribute, merchant pattern or transaction sequence may look innocuous within one bank's data set but suspicious when observed across several banks, payment service providers or insurers. Direct sharing of customer-level records is often legally restricted and commercially sensitive, while aggregated reporting may be too coarse to detect fast-moving fraud. This creates a gap between the information needed for effective detection and the information that institutions can safely exchange.

A TEE-based model can narrow that gap by providing a neutral environment for cross-institution signal analysis. Participating institutions can contribute encrypted features such as pseudonymised identifiers,[11] device fingerprints, account opening attributes, merchant categories, transaction timing or behavioural signals. The TEE can then link patterns, score clusters and identify recurring devices or identities without exposing raw customer histories to other participants or to the platform operator. The egress policy can restrict outputs to risk scores, alert categories, pseudonymous tokens, aggregate typologies or one-way audit tokens for follow-up.

The practical value of this approach can be illustrated by synthetic identity fraud and mule network detection. A single institution may observe fragments of suspicious behaviour: repeated account openings, similar device attributes, low-value test transactions or unusual transfers to newly opened accounts. Across institutions, those

[11] Direct identifiers replaced with consistent cryptographic tokens. The token allows linkage without revealing the identity; the mapping is kept separate and protected. Unlike anonymisation, pseudonymised data remain personal data.

fragments may reveal a coordinated network. A TEE can build a shared graph of pseudonymised events, identify clusters that match an approved fraud typology and return only the minimum information required for action. Raw transaction histories, identity documents and institution-specific customer files remain sealed.

This use case also shows why output control is as important as input protection. Fraud models often operate at a fine level of detail, and repeated queries could reveal whether a particular individual, device or account appears in another institution's data set. The egress policy should therefore impose rate limits, minimum-group thresholds where aggregates are released, suppression rules for small cells and strict controls on alert content. Where alerts refer to pseudonymised identifiers, the mapping back to real identities should remain with the originating institution unless legal procedures authorise further disclosure. In this way, TEEs can support sector-wide detection while preserving institutional and legal boundaries.

Fraud detection can also be deployed at the edge.[12] Instead of sending sensitive features to a central platform, each institution can run an attested fraud screening module within its own infrastructure. Partners or supervisors can verify the module's code and policy through remote attestation before issuing credentials or distributing model updates. The local module returns approve/deny decisions, risk categories or audit tokens, while the underlying customer data remain within the institution. This pattern reduces latency and data movement, and it may be particularly useful for payment systems where decisions must be made in near real time.

3.4 Financial compliance checks

AML/CFT compliance, sanctions screening and related financial integrity controls require transaction-level analysis, but the relevant information is distributed across banks, payment service providers, correspondent relationships and jurisdictions. Each institution sees only part of a transaction chain, and legal constraints often limit the sharing of detailed customer or payment information. The result is a fragmented compliance environment in which suspicious patterns may remain invisible until after harm has occurred.

TEEs can support more consistent compliance checks by allowing approved models to run on sensitive transaction data under verifiable constraints. In a centralised pattern, a governed platform could ingest encrypted and, where appropriate, pseudonymised transaction features from participating institutions. The TEE would run approved screening, typology detection or network analysis models and return only policy-approved outputs. These might include risk scores, alert categories, aggregate typology statistics or one-way audit tokens that allow authorised supervisors to request more information through established legal channels. Platform administrators and cloud operators would remain outside the plaintext boundary.

A decentralised pattern may be more suitable for time-critical or jurisdictionally constrained payments. In this model, each institution runs a TEE-based compliance module at the edge, for example in a payment gateway or on-premises processing

[12] Running software close to the data source (eg devices, gateways or on-premises servers) instead of in a central cloud. In confidential computing settings, TEEs on edge nodes can attest approved code and keep sensitive data local while returning only minimal outputs.

environment. The module verifies transactions locally against approved policy and model versions. Remote attestation allows a central bank, scheme operator or supervisor to confirm that the approved binary and policy are active before issuing credentials or accepting signed compliance decisions. The module's egress policy permits only approve/deny signals, risk categories or audit tokens; each decision is signed under a key bound to the active, attested policy version.

This decentralised model is relevant to emerging digital cash and payment system designs. Project Sela, for example, considered a digital cash architecture in which the central bank provides tools that can support intermediaries' compliance obligations while preserving competition and open access (BISIH (2023a)). A TEE-based compliance module could serve a similar function in other architectures by enabling intermediaries to prove that they are applying approved controls without exposing all transaction data to a central operator. The point is not that TEEs replace legal obligations or supervisory review, but that they can make the application of those obligations more verifiable.

The residual risks are material. False positives and false negatives may arise from model design, data quality differences or configuration errors. Repeated compliance queries may create inference risks if outputs are too granular. Attestation or revocation service outages could affect time-critical screening if systems fail closed. Cross-border deployments may also face divergent legal standards for data retention, suspicious activity reporting and supervisory access. These concerns suggest that TEE-based compliance should be implemented with clear fallback arrangements, audit rights and change control procedures, rather than as an opaque automated gatekeeper.

3.5 Digital asset wallets

Digital asset wallets are a direct application of TEEs because they depend on the secure generation, storage and use of cryptographic keys. A wallet authorises transactions by producing digital signatures. If the signing key is stolen, an attacker may be able to transfer assets, impersonate an authorised service or alter governance decisions. If the policy logic around the key is compromised, a legitimate key may be used for illegitimate transactions. The security objective is therefore twofold: protect the key itself and protect the conditions under which the key may be used.

For central banks, this issue extends beyond retail wallets. Wallet-like components may hold issuance keys for digital cash, settlement keys for wholesale tokenised central bank money, governance keys for smart contract upgrades, credentials for cross-border corridors or authorisation keys for financial market infrastructures. These keys may control assets or permissions of systemic importance. The technical design must therefore assume capable adversaries, operational failures and the need for rapid policy rollover during incidents.

A TEE can strengthen such wallets by generating keys inside the protected environment and preventing private material from leaving it. Sealed storage can bind keys to approved software and configuration, so that they unseal only when the wallet is in an authorised state. Remote attestation allows counterparties, auditors or credential issuers to verify which wallet software and policy are active before accepting signatures or issuing credentials. Policy bundles can define transaction

limits, permitted counterparties, quorum requirements, cooling-off periods, recovery procedures and logging obligations. If the policy is measured, unauthorised changes prevent key release.

Rollback and replay are central concerns in wallet design. If an attacker can restore an earlier wallet state, a spent token may be reused, a revoked policy may be reintroduced or a transaction limit may be reset. Monotonic counters, trusted time and sealed state can reduce this risk by ensuring that state moves forward and that old messages cannot be accepted as fresh. These mechanisms are essential for offline payments, where a wallet must enforce one-time use without continuous contact with a central ledger. They are not costless: counters may be exhausted, trusted time sources may fail or be attacked, and device replacement creates recovery challenges.

TEEs complement, rather than replace, HSMs in this setting. HSMs are well suited to protecting long-lived root keys and performing high-assurance cryptographic operations under controlled conditions. TEEs are useful when richer application logic must run close to the key, such as policy checks, multifactor prompts, quorum workflows, offline-spend controls or attestation-gated authorisations. A prudent design may keep long-lived roots in HSMs while using TEEs for short-lived operational keys and policy enforcement. Threshold cryptography can further reduce single-device risk by requiring several devices or roles to cooperate before high-value actions are authorised.

The following scenarios are conceptual illustrations rather than descriptions of current central bank production deployments. In a wholesale tokenised central bank money system, a settlement wallet operated by a central bank or financial market infrastructure could hold keys inside a TEE and sign transactions only after proving its software and policy state to a matching, netting or settlement service. Large transfers could require quorum approval and trusted time delays, while each authorisation could produce a one-way audit token for supervisory review. In a digital cash model, intermediaries could run reference wallet software under attested configurations that encode holding limits, offline spend rules and compliance modules. In both cases, the wallet's value lies not only in key isolation but in binding key use to verifiable policy.

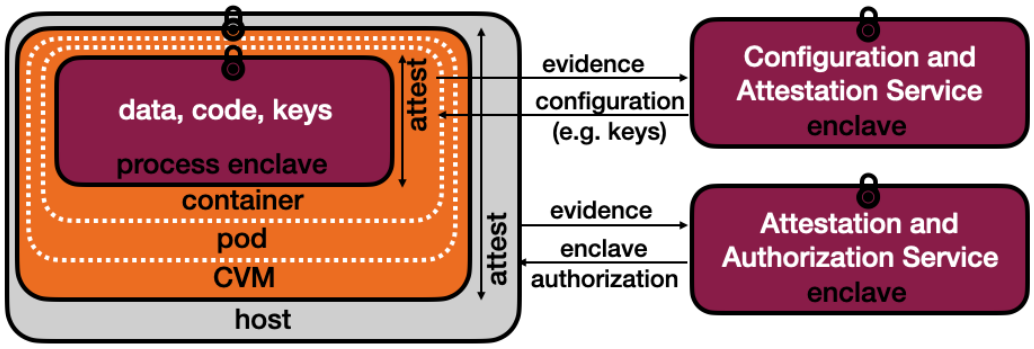
3.6 Improved cyber security

TEEs can also strengthen the cyber resilience of central bank infrastructure by adding a hardware-anchored layer of isolation around critical code and secrets. Traditional controls such as network segmentation, access management, monitoring, patching and incident response remain essential. However, they do not fully address the moment when secrets are actively used in memory. If an attacker compromises an administrator account, hypervisor, container runtime or host operating system, conventional controls may not prevent access to plaintext data or keys. A TEE can reduce this exposure by excluding those components from the protected boundary.

Layered isolation is especially relevant for critical systems. A CVM can protect an entire guest workload from the host and cloud operator, while enclaves within the guest can protect the highest-value keys or signing routines from compromise of the guest itself. This nested model does not make compromise impossible, but it can reduce the blast radius. If a virtual machine snapshot is copied, the confidentiality objective is that protected memory and sealed keys remain inaccessible outside the

attested boundary. If a container inside the guest is compromised, enclave-resident signing keys may still be shielded. If a host administrator attempts to inspect memory, the CVM boundary should prevent plaintext access (Graph 6).

Layered isolation Graph 6



Nesting an enclave within a CVM protects against host-level compromise and against compromise within the guest.

Attestation also supports a zero-trust operating model. Services can authenticate not simply as network endpoints, but as approved software running on approved platforms. Key management systems and HSMs can release secrets only to workloads presenting current evidence that satisfies policy. Configuration drift can be detected because a change in image, firmware or policy alters the measured state. During incident response, independent attestation logs can help reconstruct which software ran, when it ran and which secrets were released. This evidentiary function is particularly valuable for central banks, where accountability and auditability are as important as technical containment.

TEEs may also improve the safety of software updates. In a measured rollout, new images are signed, measured and added to the allow list only after review and testing. Keys unseal only for approved versions, and rollback to vulnerable builds can be blocked. Canary deployments can test an update on a limited set of workloads before full rollout, while attestation logs show whether any instance remained on an outdated configuration. This design can support faster patching without giving unverified software access to sensitive data.

The cyber security value of TEEs should nevertheless be stated proportionately. They do not prevent phishing, credential theft, insecure application logic, supply chain compromise or all forms of side channel leakage. They may also introduce dependencies on vendor firmware, attestation services and evidence verification infrastructure. Their contribution is narrower but important: they reduce the consequences of host-level compromise, make key release conditional on measured states and create audit evidence that links software configuration to access decisions. Used in this way, TEEs can strengthen resilience without replacing established cyber risk management.

4. Practical considerations

TEE adoption should be governed as a risk management exercise rather than as a technology procurement. The relevant question is not whether TEEs are "secure" in the abstract, but which risks they mitigate, which assumptions they introduce and which residual risks remain. This section therefore examines threats, limitations, performance and governance in the same terms used throughout the paper: measurable states, output controls, auditable evidence and disciplined operational control.

4.1 Threats and layered mitigations

TEEs reduce exposure to a compromised host, hypervisor or infrastructure administrator, but they do not eliminate all routes by which information may leak or computation may be subverted. The most important residual risks sit at the edges of the protected boundary, below it in hardware and firmware, or above it in application logic and governance. A credible deployment must therefore map each risk to a specific mitigation and must recognise the limits of that mitigation.

Side channels are a leading example. A side channel does not read protected memory directly. Instead, it infers information from observable behaviour such as timing, cache contention, page faults, power use or resource contention. Speculative execution and transient execution vulnerabilities have shown that microarchitectural behaviour can undermine assumptions about isolation. In enclave settings, the risk is heightened when secret-dependent control flow or memory access affects observable patterns. Mitigations include constant time cryptographic routines, data-oblivious access patterns where feasible, scheduler isolation, restrictions on performance counters, careful placement of workloads and minimisation of boundary crossings. These measures reduce leakage opportunities, but they do not prove the absence of leakage, especially after changes in compilers, libraries or microcode. Implementing constant time behaviour is difficult because compiler optimisations, branch prediction, memory hierarchy effects and microcode changes may reintroduce data-dependent timing or access patterns even when source code appears disciplined.

Host-mediated subversion is a second class of risk. The host may be unable to read protected memory, but it can still schedule the workload, provide system call results, deliver inputs and emulate or mediate devices. A malicious host may return stale data, suppress messages, reorder inputs or manipulate virtualised devices. Mitigations include authenticated inputs, freshness checks, trusted input/output paths, device binding and mTLS credentials derived from attested states. These controls make it harder for the host to redirect or tamper with data flows, but they require disciplined interface design. If the TEE blindly trusts host-provided metadata, the isolation boundary may be technically intact but operationally ineffective.

Rollback and replay threaten the freshness of software, state and messages. If an attacker can restore an older software image, policy bundle or sealed state, they may reintroduce a vulnerable configuration or reuse a previously valid authorisation.

Sealed storage, monotonic counters, trusted time, nonces[13] and versioned policy bundles help mitigate this risk by making old state detectable. These mechanisms are particularly important for digital wallets, offline payments and long-running confidential services. Their limitations are practical as well as technical: counters can be exhausted, time sources may be unavailable or compromised, and recovery after hardware replacement must be carefully governed.

Output inference is a distinct risk because it can arise even when input confidentiality and runtime integrity hold. A TEE may correctly process data and still release outputs that allow a participant to infer whether a particular institution, account or transaction was present in the data set. Repeated or adaptive queries are especially sensitive, because small differences in outputs can reveal membership or attributes. Mitigations include minimum-group thresholds, suppression rules, query budgets, rate limits, differential privacy where appropriate and careful review of permitted output types. The important point is that output privacy is an application and governance property, not a default consequence of using a TEE.

Supply chain compromise can weaken assurances before the workload reaches the TEE. A malicious dependency, compromised build pipeline, stolen signing key or vulnerable firmware may produce a binary that is measured and approved but unsafe. SBOMs, reproducible builds, signed provenance, hardware-backed signing keys and least-privilege continuous integration pipelines can raise assurance. They also make incident investigation faster by linking deployed binaries to source, dependencies and build processes. These controls do not guarantee the absence of defects or insider threats, but they make the policy bind between code review and attested execution more meaningful.

The layered mitigation strategy is therefore straightforward in principle but demanding in practice. Keep the TCB small; write sensitive routines in a side channel-aware way; bind secrets to measured states; authenticate channels to attested identities; constrain egress; maintain independent logs; and patch firmware and microcode promptly. Each measure addresses a named risk, and each has limits. For central banks, this explicit risk vocabulary is preferable to broad claims of confidentiality, because it supports audit, procurement, supervision and incident response.

4.2 Limits of TEE assurances

TEE assurances have been challenged repeatedly over the past decade. This history does not imply that TEEs are unsuitable for central bank use. It does imply that they should be adopted with a conservative threat model and with controls that address known failure modes. The relevant lesson from documented vulnerabilities is that hardware isolation is a powerful mitigation, but it is not a substitute for secure software, robust governance or operational discipline.

The practical context is mixed. TEEs have moved well beyond laboratory prototypes and are now used in cloud confidential computing services, secure mobile environments and selected regulated sector applications. At the same time, their deployment history includes a succession of vulnerabilities, redesigns and, in some

[13] Freshness should be enforced using nonces (values used once), monotonic counters and trusted time.

cases, product line retrenchment. Intel SGX, for example, helped establish the modern enclave model and remains foundational in the literature, yet some client-side support has been deprecated even as server-side confidential computing models have broadened. For decision-makers, the lesson is not that TEE adoption is stalled, but that current deployment should be assessed as an evolving engineering and governance practice rather than a settled security primitive.

Microarchitectural attacks provide the clearest caution. Incidents such as Foreshadow/L1TF, SGAxe, Load Value Injection and Plundervolt (Van Schaik (2020); Xiong and Szefer (2021); Murdock et al (2020a)) exploited features or behaviours below the application layer, including transient execution, cache effects and induced faults. These attacks did not break cryptography in the conventional sense. Rather, they exploited interactions between hardware behaviour and protected execution to infer or extract secrets. The lesson is that confidentiality of "data in use" depends not only on memory encryption and access control, but also on the processor's detailed behaviour under contention, speculation and fault conditions.

Cryptographic implementation failures reinforce the same point at the software layer. A cryptographic primitive may be mathematically sound but implemented in a way that leaks information through timing, memory access or exceptional cases. Enclave settings can amplify such problems because the host may be able to observe page faults, scheduling effects or other indirect signals. Verified or widely reviewed constant time libraries are therefore preferable to bespoke cryptography. Even then, assurance decays over time: compiler changes, library updates and microcode revisions may alter timing behaviour, and recent side channel findings such as EUCLEAK illustrate how implementation-level leakage can persist in widely used cryptographic devices (Roche (2025)). Periodic revalidation on target hardware should therefore be part of the operational life cycle.

The attestation path itself can also fail. If verifiers accept weak evidence, trust stale certificates or maintain inaccurate allow lists, secrets may be released to an unintended state. If revocation information is delayed, suppressed or unavailable, a known vulnerable platform may continue to receive credentials. Vendor-mediated attestation services can create availability and integrity dependencies, while heterogeneous environments[14] complicate evidence interpretation. Short-lived credentials, periodic re-attestation, independent log retention and fail-closed policies can reduce these risks, but they may also affect availability. Central banks should therefore treat attestation governance as a core control, not a technical add-on.

Input/output gaps are another recurring limitation. If devices, channels or peripherals are not cryptographically bound to the TEE, plaintext may be exposed outside the protected computation. Direct memory access, virtualised devices and host-mediated network paths can create opportunities for interception or redirection. mTLS tied to attested identity, trusted I/O paths and careful device assignment reduce these risks, but implementation is complex. In practice, the boundary of the confidential system is only as strong as the paths through which data enter and leave it (Checkoway and Shacham (2013); Muñoz et al (2023)).

[14] A heterogeneous TEE environment refers here to a deployment that uses more than one TEE technology, vendor or hardware platform, rather than relying on a single implementation throughout.

A heightened threat model is appropriate for central banks because some adversaries may have the resources to attempt physical or laboratory-class attacks. Secure elements and mobile TEEs have been attacked through fault injection,[15] debug interface abuse,[16] electromagnetic analysis[17] and other techniques under controlled conditions. Commodity server TEEs are not generally designed to withstand every form of physical attack (Pinto and Santos (2019); Koutroumpouchos et al (2021); Luo et al (2022)). Where the value-at-risk is systemic, the design should therefore distribute trust. Long-lived roots may remain in certified HSMs; high-value actions may require threshold authorisation; and TEEs may enforce policy around short-lived operational keys rather than serving as the sole line of defence.

Heterogeneity affects confidentiality and integrity differently. If several TEEs handle plaintext in the same workflow, confidentiality normally depends on each plaintext-handling component preserving secrecy; compromise of one such component may be sufficient to expose data. Integrity can sometimes be engineered under weaker assumptions. For example, replicated computation, threshold signing or k-of-n approval can tolerate a minority of faulty or compromised TEEs before an incorrect output or unauthorised action is accepted (Zhu (2020)). This distinction matters for central banks because a multi-vendor design may reduce correlated failure, but it does not automatically make confidential data safe if every implementation receives the same plaintext. Where possible, architectures should minimise how many components see plaintext and use threshold or replicated designs primarily to strengthen integrity and authorisation.

The implication is a balanced one. TEEs can materially reduce well specified risks, especially exposure to host-level compromise and unauthorised operator access. They can also make trust decisions more measurable through attestation. But they do not eliminate side channels, defective code, supply chain compromise, output inference or all physical attacks. Their value is highest when they are embedded in a broader architecture of layered controls, evidence retention, portability planning and legal governance.

4.3 Performance and scaling

TEE performance costs are best understood by identifying where the overhead arises. TEEs do not impose a uniform penalty on all computation. The largest costs usually occur at boundary crossings, memory protection limits, attestation and startup, input/output paths and operational orchestration. This means that performance can often be managed through architecture, batching and workload placement.

Enclaves incur overhead when execution enters or exits the protected region. If an application calls the enclave for every small operation, the fixed transition cost may

[15] Fault injection in a TEE is a technique that intentionally stresses or disrupts a processor's normal operation to bypass its built-in security protections.

[16] Debug interface abuse in a TEE occurs when attackers exploit hardware test or debug ports (like JTAG or SWD) to bypass security boundaries, extract secrets or inject malicious code into secure enclaves. It subverts the hardware-level isolation that TEEs are designed to enforce.

[17] Electromagnetic analysis is a type of side channel attack used against a TEE to extract sensitive data like cryptographic keys. It exploits the fact that physical microprocessors leak electromagnetic radiation when processing information.

dominate. If the application batches work and keep sensitive control flow inside the enclave, the overhead may be much lower. Protected memory limits (for example, the Enclave Page Cache[18]) can be more significant for memory-intensive workloads. When enclave pages are swapped, latency can increase and access patterns may become more observable. Enclave applications therefore benefit from micro-batching, pre-allocation of buffers, careful memory layout and minimisation of host interaction.

CVMs generally offer a different performance profile. Because a full guest OS runs inside the protected boundary, ordinary application code may run with relatively modest CPU and memory overhead. The more visible costs may arise during instance initialisation, attestation, key unsealing, encrypted input/output and interrupt handling. If a new CVM is created for each task, cold start latency may be material. If a pool of already attested instances is maintained, the cost can be amortised. In analytical workloads, NUMA placement,[19] storage encryption, network paths and container startup inside the CVM may also affect tail latency.

For central bank workloads, the relevant benchmark is not only raw overhead but operational fit. A cross-border stress test may tolerate minutes of startup time if the computation runs for hours and uses large data sets. A payment screening module may not tolerate even small increases in decision latency. A fraud detection batch job may benefit from confidential computation even if throughput is lower, if collaboration becomes legally and operationally feasible. Performance evaluation should therefore be workload-specific and should include the full life cycle: attestation, key release, data loading, computation, output enforcement, logging and recovery.

Comparison with other privacy-enhancing technologies should follow the same logic. TEEs can execute general purpose, stateful code and are therefore well suited to iterative models, graph analytics and existing software stacks. MPC may be preferable for narrow set operations or threshold signing when parties want to avoid hardware trust and can tolerate coordination overhead. FHE may be appropriate for specialised encrypted arithmetic when latency is acceptable. ZKPs may be best used to verify compliance properties or output constraints rather than to replace bulk computation. These distinctions prevent misleading comparisons based on a single performance metric.

Scaling strategies should preserve security assumptions. Warm pools of attested instances can reduce startup delays, but credentials should remain short-lived and re-attestation should occur on a defined cadence. Streaming can keep bulk encrypted payloads outside the TEE while passing only authenticated metadata or commitments into the protected boundary, but this must not expose plaintext or weaken integrity checks. Batching can improve throughput, but it should not combine data in ways that violate legal or policy constraints. Optimisation should therefore be treated as a controlled change: new builds, compiler settings or memory layouts should be remeasured, retested and reapproved before receiving secrets.

[18] The Enclave Page Cache is the hardware-protected region that stores enclave code and data.

[19] In multi-processor architectures, Non-Uniform Memory Access (NUMA) placement dictates to which CPU cores and local memory modules an application's threads are assigned.

Performance evidence should also be reproducible. Benchmarks should record the code hash, configuration flags, firmware and microcode versions, hardware type, workload characteristics and output policy. Without this information, performance claims are difficult to compare and may become misleading after updates. TEEs can often provide practical performance for central bank analytics and selected low-latency functions, but each use case requires empirical testing under the same attestation and policy controls that would apply in production.

4.4 Procurement guardrails and governance

Procurement should translate the technical model into enforceable obligations. A TEE deployment is only as strong as the evidence verifiers can inspect, the policies that govern key release and the operational processes that maintain those policies over time. Contracts and supervisory arrangements should therefore specify not only which hardware feature is used, but also how attestation evidence is produced, verified, logged, revoked and audited.

The first guardrail is independent verifiability. Attestation protocols should be open, documented and auditable, with trust anchors that are acceptable to the participating authorities. Verifiers should not have to rely solely on the cloud operator's assertion that a workload is confidential. They should be able to examine evidence, validate signatures, interpret measurements and compare them with approved policies. Where vendor services are involved, contracts should define availability, incident notification, revocation and evidence-retention obligations. This reduces operator-in-the-middle and stale trust risks, while recognising that attestation infrastructure itself becomes critical (Küçük et al (2022); Booth et al (2025)).

The second guardrail is disciplined allow list governance. Approved software hashes, firmware minima, configuration flags and policy versions should be maintained in signed allow lists with clear ownership and change control. Additions should be linked to review, testing and approval records. Removals should occur promptly when vulnerabilities are discovered or policies change. Exceptions should be time-limited and documented. This governance is essential because key release follows the allow list. If the list is inaccurate or stale, the attestation mechanism may faithfully approve the wrong state.

The third guardrail is supply chain assurance. Procurement should require SBOMs, signed provenance, reproducible builds where feasible and hardware-backed code signing keys. Continuous integration and release pipelines should follow least-privilege principles, with separation between those who approve code, those who operate infrastructure and those who can access sensitive data. These controls matter because attestation measures what runs, it does not prove that the measured binary was produced honestly. Build provenance and runtime evidence must therefore be joined (Booth et al (2025); Cerdeira et al (2020)).

The fourth guardrail is output governance. Contracts and policies should define the permitted output types before deployment. These may include aggregates, approve/deny signals, risk categories, exception flags or one-way audit tokens. Plaintext microdata should be excluded unless a separate legal authority permits disclosure. Minimum-group thresholds, suppression rules, rate limits and query budgets should be encoded as configuration and bound to the measured state.

Attested logs should record which output type was produced by which code and policy version, while avoiding leakage of sensitive content.

The fifth guardrail is portability and concentration risk management. Homogeneous deployments are easier to operate but may create monoculture exposure to vendor-specific vulnerabilities, evidence service outages or geopolitical constraints. Heterogeneous deployments can reduce correlated failure but introduce evidence normalisation challenges and operational complexity. A prudent procurement strategy may set a planning objective to support at least two TEE families or cloud environments over time, even if initial deployments begin with one. Where single-vendor dependence is unavoidable, compensating controls should include contractual disclosure obligations, rapid patch windows, migration runbooks and independent retention of evidence.

The acceptability of single-vendor dependence is ultimately a risk appetite and mandate question, not a purely technical one. For low-sensitivity pilots, a single TEE family may be acceptable if the workload is reversible and no systemic function depends on it. For high-sensitivity or systemic workloads, however, placing confidentiality on one vendor's hardware, firmware, attestation service and disclosure cadence creates technical, operational and geopolitical concentration risk. Such dependence should require compensating controls, including contractual vulnerability disclosure obligations, rapid patch and revocation windows, independent retention of attestation evidence, HSM-backed root keys, threshold approval for high-value actions and a tested exit path to another TEE family or non-TEE design.

The sixth guardrail is incident response. Runbooks should cover attestation failure, revocation events, key rotation, evidence service outages, suspected side channel vulnerabilities, compromised signing keys and emergency policy rollover. Systems that fail closed may protect confidentiality but affect availability; systems that fail open may preserve service but undermine trust. These choices should be made explicitly by workload, not improvised during an incident. For critical payment or settlement functions, fallback modes may need legal, operational and supervisory approval in advance.

Finally, legal and supervisory alignment should be built into the architecture. Cross-border deployments require agreement on audit artefacts, evidence formats, retention periods, deletion obligations and access rights. Technical evidence is useful only if the relevant authorities have the legal basis, operational ability and interpretive standards to use it. Procurement should therefore treat attestation logs, configuration signatures and policy histories as supervisory artefacts, not merely engineering records.

Box B

Outlook

The medium-term development of confidential computing is likely to focus less on the basic concept of protected execution and more on the governance infrastructure around it. For central banks, the most important developments are likely to be evidence portability, post-quantum migration, hybrid trust models and stronger transparency mechanisms.

Evidence portability is a priority. Today, different TEE vendors produce attestation evidence in different formats and with different semantics. This complicates multi-cloud and multi-vendor verification. Standardised evidence formats, common claims vocabularies and independently operated transparency logs could reduce verification costs and improve auditability. Such mechanisms would not replace PKI governance or timely revocation, but they would make it easier for institutions to compare evidence across platforms and detect unauthorised changes.

Post-quantum cryptography will become relevant to attestation chains, channel protection, egress signing and long-lived credentials. Hybrid approaches that combine classical and post-quantum algorithms may provide a transition path, but they may also increase key sizes, signature sizes and startup latency. Central banks should therefore inventory cryptographic dependencies early, especially in systems with long retention periods or long-lived trust anchors. Procurement should require algorithm agility so that cryptographic migration can occur without redesigning the full system.

Hybrid trust models are also likely to become more common. TEEs may orchestrate stateful workflows while MPC performs specific private set operations, ZKPs verify policy compliance and HSMs hold long-lived root keys. These combinations can reduce reliance on any single trust primitive, but they also increase liveness, monitoring and failure complexity. Hybrid designs should therefore be adopted for clear risk reduction reasons, not because combining advanced technologies is inherently desirable.

Decentralised-finance (DeFi) examples offer useful, though not directly transferable, technical lessons. Town Crier,① for instance, used SGX-based attestation to support authenticated data feeds for smart contracts, showing how a TEE can bridge off-chain data and on-chain execution (Zhang et al (2016)). Validity rollups illustrate a complementary idea: succinct proofs can make computation publicly verifiable, while confidential preparation of inputs may occur elsewhere. Together, these examples show how attestation and proofs can be combined, but they do not remove side channel, supply chain, availability or key management risks in the TEE layer. Central banks should therefore draw selectively from them, adapting mechanisms that improve verifiability while preserving public sector requirements for resilience, accountability and audit.

The planning implication is practical. Institutions considering TEEs should invest early in evidence normalisation, audit log design, policy versioning and portability tests. Pilot programmes should measure not only latency and throughput, but also revocation handling, update rollover, cross-jurisdiction verification and incident response. The future of confidential computing for central banks is therefore not only a hardware question. It is an institutional question about how measurable technical states can be turned into durable governance.

① Town Crier is a secure, authenticated data feed for smart contracts that acts as a bridge between blockchains and external, HTTPS-enabled websites. It uses Intel SGX to securely scrape and authenticate off-chain data and feed them to smart contracts without requiring trust in a central operator.

5. Conclusion

This paper has examined whether TEEs can help central banks and regulated institutions compute with sensitive data without widening access to the underlying records. The motivation is the persistent "data in use" gap. Encryption at rest and in transit is well established, but data normally must be processed in plaintext. At that moment, conventional systems may expose sensitive information to operating systems, hypervisors, administrators or cloud operators. TEEs address this gap by creating a hardware-backed context in which approved code can process plaintext while the surrounding host environment is excluded from protected memory and control flow.

The core contribution of TEEs is not secrecy in isolation, but verifiable conditional access. Code authenticity, runtime integrity and confidentiality of "data in use" are valuable because they can be exposed through remote attestation. Attestation allows a relying party to verify the software image, configuration and platform state before releasing secrets or encrypted inputs. When attested identity is bound to mTLS channels, the link from authentication to key exchange to in-TEE-only decryption becomes operational. When output policy is measured and enforced by an egress routine, the same chain can govern what information leaves the protected boundary. Practically, confidentiality should govern computation and disclosure: share insights, not data.

The paper has distinguished two architectural styles. Enclaves confine sensitive routines and keys within a small, auditable boundary. CVMs protect an entire guest operating system and application stack from the host, enabling easier migration of existing workloads. Neither model is universally superior. Enclaves are better suited to narrow, high-assurance functions; CVMs are better suited to whole-workload protection and operational compatibility. In many central bank settings, a layered design may be most appropriate: CVMs protect the broader platform from the host, while enclaves or HSMs protect the most sensitive signing, decryption and policy enforcement functions.

The central bank use cases share a common structure. In official statistics, TEEs may enable cross-jurisdiction analysis of microdata while releasing only aggregates and audit tokens. In systemic stress testing, they may support bottom-up simulations using institution-level exposures without exposing bilateral positions. In fraud detection and AML/CFT compliance, they may allow institutions to detect shared patterns while limiting outputs to risk signals or one-way audit tokens. In digital asset wallets, they may bind key use to measured software, policy versions, trusted time and quorum rules. In cyber resilience, they may reduce the consequences of host-level compromise and make key release conditional on verified configuration.

These benefits are conditional. Documented attacks on TEEs and related secure hardware show that side channels, transient execution, induced faults, input/output gaps, implementation defects and attestation path weaknesses can undermine assurances. Supply chain compromise may introduce malicious or vulnerable code before measurement, and homogeneous deployments may create concentration risk. Output inference may occur even when the protected computation itself works as intended. These limitations do not negate the technology, but they require proportionate claims and layered controls.

A risk-grounded adoption path would therefore combine TEE deployment with side channel-aware development, signed and reproducible builds, SBOMs, robust allow list governance, short-lived credentials, periodic re-attestation, independent evidence logs, HSM-backed roots, threshold authorisation for high-value actions and carefully specified output policies. Procurement should require open and auditable attestation, timely revocation, incident evidence, portability planning and cross-jurisdiction audit rights. Performance should be assessed at workload level, including attestation, key release, data loading, computation, output enforcement and recovery, rather than through isolated microbenchmarks.

TEEs may allow central banks to conduct deeper analysis and strengthen operational resilience without expanding the circle of actors trusted with plaintext data. They are most persuasive when used to bind computation to measurable states and disclosure to enforceable policy. They are least persuasive when presented as a black box guarantee of privacy or security. Their promise for central banks lies not in replacing governance with hardware, but in making governance technically verifiable.

List of acronyms

Acronym	Term
AML	anti-money laundering
API	application programming interface
ARM CCA	ARM Confidential Compute Architecture
BIS	Bank for International Settlements
CFT	countering the financing of terrorism
CPU	central processing unit
CVM	confidential virtual machine
DCAP	Data Center Attestation Primitives
DLT	distributed ledger technology
DMA	direct memory access
FHE	fully homomorphic encryption
FMI	financial market infrastructure
HSM	hardware security module
IBS	international banking statistics
IDP	identity provider
I/O	input/output
KYC	know your customer
MPC	multi-party computation
mTLS	mutually authenticated Transport Layer Security
NBFI	non-bank financial institution
OAuth	OAuth authorisation framework
OIDC	OpenID Connect
OS	operating system
PKI	public key infrastructure
PSP	payment service provider
RA	remote attestation
SBOM	software bill of materials
SDK	software development kit
SEV-SNP	Secure Encrypted Virtualisation – Secure Nested Paging
SGX	Software Guard Extensions
SKPP	Separation Kernel Protection Profile (NSA)
TCB	trusted computing base
TDX	Trust Domain Extensions
TEE	trusted execution environment
TLS	Transport Layer Security
vTPM	virtual Trusted Platform Module
ZKPs	zero-knowledge proofs

References

Acar, A, H Aksu, A Selcuk Uluagac and M Conti (2018): "A survey on homomorphic encryption schemes: theory and implementation", *ACM Computing Surveys*, vol 51, no 4.

Aiyar, S, C W Calomiris and T Wieladek (2012): "Does macro-pru leak? Evidence from a UK policy experiment", *NBER Working Paper Series*, no w17822.

Ames, S, M Gasser and R Schell (1983): "Security kernel design and implementation: an introduction", *Computer*, vol 16, no 7.

Anand, K, B Craig. and G von Peter (2015): "Filling in the blanks: network structure and interbank contagion", *Quantitative Finance*, vol 15, no 4.

ARM Ltd (2025): "ARM Confidential Compute Architecture", *www.arm.com/architecture/security-features/arm-confidential-compute-architecture*, accessed 2025.

Arnautov, S, B Trach, F Gregor, T Knauth, A Martin, C Priebe, J Lind, D Muthukumaran, D O'Keeffe, M Stillwell and D Goltzsche (2016): "SCONE: Secure linux containers with Intel SGX", *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*.

Asokan, N, J-E Ekberg, K Kostiainen, A Rajan, C Rozas, A-R Sadeghi, S Schulz and C Wachsmann (2014): "Mobile trusted computing", *Proceedings of the IEEE*, vol 102, no 8.

Bank for International Settlements and central bank group (BIS-CBG) (2020): *Central bank digital currencies: foundational principles and core features*.

——— (2021): *Central bank digital currencies: system design and interoperability*.

——— (2024): *Central bank digital currencies: system design*.

BIS Innovation Hub (BISIH) (2023a): *Project Sela: an accessible and secure retail CBDC ecosystem*.

Booth, H, M Ogata, K Kent, M Souppaya and D Dodson (2025): "Secure Software Development Framework (SSDF) Version 1.2: Recommendations for Mitigating the Risk of Software Vulnerabilities", *NIST Special Publication (SP) 800-218 Rev. 1 (Draft)*, National Institute of Standards and Technology.

Bruno, V and H S Shin (2015): "Capital flows and the risk-taking channel of monetary policy", *Journal of Monetary Economics*, vol 71.

Carpov, S, N Gama, M Georgieva and J R Troncoso-Pastoriza (2020): "Privacy-preserving semi-parallel logistic regression training with fully homomorphic encryption", *BMC medical genomics*, vol 13 (suppl 7).

Cerdeira, D, N Santos, P Fonseca and S Pinto (2020): "Sok: Understanding the prevailing security vulnerabilities in TrustZone-assisted tee systems", *2020 IEEE Symposium on Security and Privacy (SP)*.

Cerutti, E, G Hale and C Minoiu (2015): "Financial crises and the composition of cross-border lending", *Journal of International Money and Finance*, vol 52.

Cetorelli, N and L Goldberg (2012): "Liquidity management of US global banks: internal capital markets in the great recession", *Journal of International Economics*, vol 88, no 2.

Chakrabarti, S, T Knauth, D Kuvaiskii, M Steiner and M Vij (2020): "Trusted execution environment with Intel SGX", in X Jiang and H Tang (eds), *Responsible genomic data sharing*, 1st edition, Academic Press, London.

Checkoway, S and H Shacham (2013): "Iago attacks: why the system call API is a bad untrusted RPC interface", *ACM SIGARCH Computer Architecture News*, vol 41.

Chillotti, I, N Gama, M Georgieva and M Izabachène (2020): "TFHE: fast fully homomorphic encryption over the torus", *Journal of Cryptology*, vol 33, no 1.

Diffie, W and M Hellman (2022): "New directions in cryptography", in R Slayton (ed), *Democratizing cryptography: the work of Whitfield Diffie and Martin Hellman*.

Eisenberg, L and T Noe (2001): "Systemic risk in financial systems", *Management Science*, vol 47, no 2.

Evans, D, V Kolesnikov and M Rosulek (2018): "A pragmatic introduction to secure multi-party computation", *Foundations and Trends in Privacy and Security*, vol 2, nos 2–3.

Fetzer, C (2016): "Building critical applications using microservices", *IEEE Security & Privacy*, vol 14, no 6.

Gai, P and S Kapadia (2010): "Contagion in financial networks", *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol 466, no 2120.

Glasserman, P and H Young (2016): "Contagion in financial networks", *Journal of Economic Literature*, vol 54, no 3.

Goldwasser, S, Y Kalai and G Rothblum (2015): "Delegating computation: interactive proofs for muggles", *Journal of the ACM*, vol 62, no 4.

Greenwood, R, A Landier and D Thesmar (2015): "Vulnerable banks", *Journal of Financial Economics*, vol 115, no 3.

Gregor, F, W Ozga, S Vaucher, R Pires, D Le Quoc, S Arnautov, A Martin, V Schiavoni, P Felber and C Fetzer (2020): "Trust management as a service: enabling trusted execution in the face of byzantine stakeholders", *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*.

Hellman, M (1978): "An overview of public key cryptography"", *IEEE Communications Magazine*, vol 16, no 6.

Houston, J, C Lin and Y Ma (2012): "Regulatory arbitrage and international bank flows", *The Journal of Finance*, vol 67, no 5.

Information Commissioner's Office (ICO) (2026): "Trusted execution environments", available at *ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-sharing/privacy-enhancing-technologies/what-pets-are-there/trusted-execution-environments*, accessed 2026.

Jauernig, P, A Sadeghi and E Stapf (2020): "Trusted execution environments: properties, applications, and challenges", *IEEE Security & Privacy*, vol 18, no 2.

Koutroumpouchos, N, C Ntantogian and C Xenakis (2024): "Building trust for smart connected devices: the challenges and pitfalls of TrustZone", *Sensors* vol 21, no 2.

Kuvaiskii, D, G Kumar and M Vij (2022): "Computation offloading to hardware accelerators in Intel SGX and Gramine library OS", *arXiv preprint*, arXiv:2203.01813.

Küçük, K, S Moyle, A Martin, A Mereacre and N Allott (2022): "SoK: how not to architect your next-generation TEE malware?", *Proceedings of the 11th International Workshop on Hardware and Architectural Support for Security and Privacy*.

Luo, L, Y Zhang, C White, B Keating, B Pearson, X Shao, Z Ling, H Yu, C Zou and X Fu (2022): "On security of TrustZone-M-based IoT systems", *IEEE Internet of Things Journal*, vol 9, no 12.

Miwa, S and S Matsuo (2023): "Analyzing the performance impact of HPC workloads with Gramine+ SGX on 3rd generation Xeon scalable processors", *Proceedings of the SC'23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*.

Minoiu, C and J A Reyes (2013): "A network analysis of global banking: 1978–2010", *Journal of Financial Stability*, vol 9, no 2.

Muñoz, A, R Ríos, R Román and J López (2023): "A survey on the (in)security of trusted execution environments", *Computers & Security*, vol 129.

Murdock, K, D Oswald, F D Garcia, J Van Bulck, F Piessens and D Gruss (2020a): "Plundervolt: how a little bit of undervolting can create a lot of trouble", *IEEE Security & Privacy*, vol 18, no 5.

Murdock, K, D Oswald, F D Garcia, J Van Bulck, D Gruss and F Piessens (2020b): "Plundervolt: Software-based fault injection attacks against Intel SGX", *IEEE Symposium on Security and Privacy*.

Narayanan, V, C Carvalho, A Ruocco, G Almasi, J Bottomley, M Ye, T Feldman-Fitzthum, D Buono, H Franke and A Burtsev (2023): "Remote attestation of confidential VMs using ephemeral vTPMs", *Proceedings of the 39th Annual Computer Security Applications Conference*.

National Institute of Standards and Technology (NIST) (2017): "An introduction to information security", *NIST Special Publication*, no 800-12.

National Security Agency (NSA) (2007): "Protection profile for separation kernels in environments requiring high robustness", *US Government Information Assurance Directorate*.

Pinto, S and N Santos (2019): "Demystifying ARM TrustZone: a comprehensive survey", *ACM Computing Surveys,* vol 51, no 6.

Roche, T (2025): "EUCLEAK side-channel attack on the YubiKey 5 series (revealing and breaking Infineon ECDSA implementation on the way)", *2025 IEEE Symposium on Security and Privacy*.

Rushby, J (1981): "Design and verification of secure systems", *ACM SIGOPS Operating Systems Review* vol 15, no 5.

Sabt, M, M Achemlal and A Bouabdallah (2015): "Trusted execution environment: what it is, and what it is not", *2015 IEEE Trustcom/BigDataSE/Ispa*, vol 1.

Shin, H S (2012): "Global banking glut and loan risk premium", *IMF Economic Review*, vol 60, no 2.

Tsai, C-C, D Porter and M Vij (2017): "Graphene-SGX: a practical library OS for unmodified applications on SGX", *2017 USENIX annual technical conference (USENIX ATC 17)*.

Van Bulck, J, M Minkin, O Weisse, D Genkin, B Kasikci, F Piessens, M Silberstein, T Wenisch, Y Yarom and R Strackx (2018): "Foreshadow: extracting the keys to the Intel SGX kingdom with transient out-of-order execution", *27th USENIX Security Symposium (USENIX Security 18)*.

Van Bulck, J, D Moghimi, M Schwarz, M Lippi, M Minkin, D Genkin, Y Yarom, B Sunar, D Gruss and F Piessens (2020): "LVI: hijacking transient execution through microarchitectural load value injection", *2020 IEEE Symposium on Security and Privacy*.

Van Schaik, S, A Kwong, D Genkin and Y Yarom (2020): "SGAxe: How SGX fails in practice", available at https://sgaxe.com/files/SGAxe.pdf, accessed 2026.

Xiong, W and J Szefer (2021): "Survey of transient execution attacks and their mitigations", *ACM Computing Surveys*, vol 54, no 3.

Zhang, F, E Cecchetti, K Croman, A Juels and E Shi (2016): "Town Crier: an authenticated data feed for smart contracts", *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.

Zhu, J, R Hou, X Wang, W Wang, J Cao, B Zhao, Z Wang, Y Zhang, J Ying, L Zhang and D Meng (2020): "Enabling rack-scale confidential computing using heterogeneous trusted execution environment", *2020 IEEE Symposium on Security and Privacy*.