ADAN

Association régie par la loi du 1er juillet 1901,
Déclarée en Préfecture des Hauts-de-Seine le 24/12/2019,
Publiée au Journal Officiel du 28/12/2019,
Immatriculée au RNA sous le numéro W922017256,
Siège social : 25, rue du 4 Septembre, 75002 PARIS

Working group on Smart contracts certification Consultation questionnaire

This questionnaire accompanies the report of the ACPR-AMF Fintech Forum working group on the certification of smart contracts.

Its aim is to gather the opinions of stakeholders in the financial and crypto-asset sectors, as well as those of any other interested parties (researchers, service providers, supervisory authorities, etc.) on the issues raised in this report. Respondents are invited to illustrate their comments with concrete cases and to specify the references of any work.

Part 1 of the document: standards

Q 1: Do you have any comments on the **security** principles set out in the document?

(Principle 1): Requiring smart contract code to comply with known secure coding practices and avoid any errors that have already been documented. The DeFi industry has learned hard lessons from past exploits – e.g. the 2016 DAO re-entrancy attack or integer overflow bugs – which are now well-documented in resources like the SWC Registry and OWASP Top 10 for smart contracts. A concrete example is the Ethereum Enterprise Alliance's EthTrust standard, which defines 21 security requirements to ensure Solidity contracts aren't vulnerable to known attack vectors. Adhering to such community-driven standards helps developers pre-empt issues. We recommend industry-led knowledge sharing, such as **maintaining an open database of smart contract vulnerabilities and past exploits, so developers can continuously update their practices.** Many projects already leverage automated analysis tools (e.g. Slither, Mythril) and linters that flag known bad patterns, embodying this principle in a self-regulatory way. However, given the operational complexity associated with the frequent dependencies between contracts, imposing ex ante certification for each update could prove difficult to maintain and slow down the urgent connection of vulnerability. A voluntary or ex-post approach would be preferable to preserve the responsiveness that is essential in these environments.

(Principle 2): We agree that code should be written to mitigate environment-driven vulnerabilities. In practice, this means using known secure patterns for randomness (avoiding predictable on-chain randomness), handling external oracles cautiously (with sanity checks on data feeds), and being mindful of gas constraints and re-entrancy across calls. Oracles, while essential, introduce different risks: centralized oracles create single points of failure, while decentralized ones require strong governance mechanisms to prevent manipulation. Multi-source aggregation and cryptographic validation strengthen reliability. The EthTrust standard explicitly covers many of these environment issues, including oracle interactions and MEV manipulation. By following such industry standards, developers can adapt to the evolving threat landscape faster than any static regulation could. It's worth highlighting that this principle also encourages defense-in-depth. For example, some protocols implement circuit-breakers or rate limits to contain damage if an external dependency misbehaves. We support community-driven best practice guides (similar to NIST's guidelines for secure software) that teach new developers how to build robust contracts resilient to their execution context.

(Principle 3): Deterministic builds ensure that the published source code truly matches the on-chain bytecode. This is analogous to reproducible builds in open-source software, which prevent malicious or accidental inconsistencies. In the blockchain context, projects like Ethereum and Solana have introduced verifiers so that compiled contract code can be byte-for-byte reproduced by anyone. Reproducibility enables independent verification by the community or auditors that no hidden code was introduced. The industry can self-impose this standard by using verified compiler settings and bytecode audits. For example, many Ethereum projects verify their contracts on Etherscan, allowing anyone to compare the source and bytecode – a de-facto self-certification of code integrity. **We believe regulators should encourage such transparency practices but not mandate a specific process; the community is already incentivized to adopt them because it builds user confidence.**

Principle 4): Making the smart contract code public is now an unwritten rule in DeFi and we fully support it being a principle. Open-source code allows for community review, crowdsourced auditing, and greater user trust. Many high-profile protocols (Uniswap, Aave, Compound, etc.) open source their code and often users will avoid protocols that do not publish code, as opaque code is a red flag. **This norm emerged from industry self-regulation – essentially, market pressure forces transparency.** We encourage this

to remain a best practice rather than a legal requirement. Developers should want to publish code to signal quality. In addition, open code enables independent security researchers to continually inspect and even formally verify it, beyond any one-time certification. After the Compound protocol open-sourced its code, community contributors identified potential issues and improvements, strengthening the protocol without any regulatory mandate. This principle greatly enhances cyber resilience by tapping into collective scrutiny. However, in certain specific cases (enterprise applications, proprietary smart contracts, or contracts that require strict confidentiality), alternative security measures like third-party audits or cryptographic proofs can ensure trust without full public disclosure.

(Principle 5): Smart contracts should implement strict role-based permissions and minimal access scopes. In practice, this translates to patterns like only allowing specific addresses or contracts to perform admin functions and limiting those functions to what's absolutely required. For instance, a DeFi lending protocol might restrict liquidation functions to a dedicated module or require certain conditions, rather than giving a single address sweeping power. Many exploits occur from overly broad privileges (e.g. an admin key that can drain funds). By designing for least privilege, even if one component is compromised, the blast radius is limited. The blockchain community has been embracing this: for example, Gnosis Safe multisig wallets allow distributing control among multiple parties, and upgradeable contracts often employ timelocks and multi-signature confirmations. We advocate industry-led standards for access control (similar to how NIST SP 800-53 outlines access controls in traditional IT) – for example, OpenZeppelin's library provides battle-tested implementations of role-based access which projects voluntarily adopt. Self-regulation via code libraries and peer audits often catches privilege issues, as seen when auditors flag any function that doesn't enforce proper onlyOwner or access modifiers. The same role-based access rules should apply to smart contract upgrades, ensuring that no single entity can unilaterally modify a deployed contract without multi-party validation and timelocks.

(Principle 6): We support the requirement to distribute tasks and privileges among distinct people or entities. This principle goes hand-in-hand with least privilege to prevent any single point of failure or opportunity for malfeasance. In decentralized governance, this often means multi-signature control or DAO governance for critical actions. Governance structures can vary depending on the maturity of a protocol. DAOs and multisig wallets provide robust decentralization, but hybrid governance models - where decision-making transitions from a core team to a community-driven mechanisms over time - are also relevant for balancing security and flexibility. A real-world example is MakerDAO's governance. Changes to critical parameters require approval through a token-holder vote and are executed via a time-locked governance contract, rather than one individual unilaterally changing the code. Even in more centralized teams, many have adopted 2-of-3 or 4-of-6 multisig wallets for admin keys, distributing control among founders or even trusted community members. This practice arose organically after incidents where a lone developer's key compromise led to a hack. By comparison, in traditional finance ISO 27001 or SOC2 frameworks, separation of duties is a baseline requirement for internal controls. The blockchain industry can mirror these proven governance controls on its own. Self-certification could include attesting that no single actor has unilateral control, with evidence like multisig addresses provided. This again is something industry participants (especially those seeking to attract institutional users) are motivated to implement without needing a regulator to dictate it.

(Principle 7): We agree that security must cover the entire lifecycle of a smart contract, especially if it can be updated or modified. This means not only securing the initial deployment but also having processes for safe upgrades, patches, or parameter changes. Many exploits occur during upgrades or due to improper upgrade mechanisms (for

example, proxy contracts that were upgradable without access control have been hijacked in the past). Best practices to support this principle include using time-locks on upgrades (giving users advance notice of changes), performing audits on new versions, and providing an emergency "pause" or rollback mechanism if a faulty update is deployed. Governance minimization is another approach. Some projects choose to make contracts immutable (no upgrades) to avoid lifecycle risks entirely, at the cost of flexibility. Both approaches have merit and should be accommodated by standards: either design for safe upgrades or clearly document immutability. However, security requirements should be applied with granularity based on the contract's criticality. Some contracts require strict upgrade controls, while others, especially truly immutable ones, need different risk management approaches. Then, the certification frameworks should reflect this distinction. We note the working group's report discusses how even "immutable" protocols often have changeable elements so any certification must carefully examine what kind of changes are possible. We advocate that the industry develop guidelines for secure upgrade mechanisms, perhaps drawing on frameworks like the OpenZeppelin Upgrades library and community audits. **By sharing proven patterns (e.g. Proxy + logic contract with Transparent Upgrade Proxy pattern), developers can work to ensure lifecycle changes don't introduce vulnerabilities. External independent audits before and after upgrades are already a de facto industry standard for serious projects.**

Voluntary certification programs or seals of approval could be introduced to recognize projects that adhere to these principles, much like how SSL/TLS certificates or SOC 2 reports are used in web services – but participation should remain optional. This will create a competitive incentive for projects to harden their contracts and undergo audits, without erecting barriers to entry for new, innovative teams.

Q 2: Do you have any comments on the **governance** principles set out in the document?

(Principle 8): The document rightly calls for "processes of making and implementing decisions [to be] clearly and accurately described, updated without delay, and published." This is fundamental. Users and stakeholders must know how a protocol is governed, who has the power to change what, and how they can participate or exit. Many leading DeFi projects set positive examples here. To borrow from Uniswap and Compound again, both have openly documented governance via community forums and published governance charters. Any change goes through a proposal process visible to all token holders, often discussed in public before on-chain voting. Governance frameworks should also account for the varying degrees of decentralization across protocols. Whether fully transparent governance is a best practice for mature DeFi projects, early-stage or experimental protocols may require staged approaches that evolve as governance mechanisms mature.

One approach could be that projects self-publish a "governance document" or transparency report covering roles, decision processes, and update mechanisms (much like a corporate prospectus or an open-source project's governance.md file). Principle 8 effectively encourages a "constitution" for the protocol – something that forward-looking projects already embrace to earn user trust. This can be done via industry best practice: e.g. the DeFi Safety initiative provides transparency scores based on documentation of governance and other processes, incentivizing projects to clearly publish this information. This market-driven approach (rating the level of disclosure) could be highly effective. Regulators could also support it by endorsing the notion that good governance disclosure is part of being a reputable protocol, but it need not be enforced via a heavy-handed rule.

(Principle 9): This principle ensures that governance arrangements include rules to "safeguard the interests of customers and users in the event of a change." For example, by providing timely information, advance notice, and an option for users to exit if they disagree with changes. This is crucial for maintaining user trust and fair treatment. In practice, many

decentralized protocols have implemented exactly such safeguards. Time-lock delays on upgrades are a common mechanism that serves this purpose. For instance, Compound uses a 2-day time-lock on any governance-approved changes to its smart contracts; during that window, users who are uncomfortable can withdraw funds or unwind positions. Another example is MakerDAO's emergency shutdown mechanism. While not exactly a notice of change, it's a user safeguard that lets users exit (redeem collateral) if the community decides to shut the system due to an emergency or major change. We advocate that the principle of user consent or graceful exit be upheld by industry standards. **Concretely, an industry consortium or DAO could develop a code template for upgradeable contracts that enforces a delay and announcement, so any project using it inherently meets this principle.** Many projects already voluntarily announce upcoming upgrades on social media or blogs well in advance; formalizing this as a best practice is wise. **However, we caution against regulators trying to specify exact notice periods or methods in law – flexibility is needed because the appropriate safeguard might differ by protocol.** For example, a high-frequency trading DeFi platform might choose a shorter upgrade notice but more frequent audits, whereas a large stablecoin might promise a longer notice for contract changes. The key is that the expectation of transparency and consideration for user impact becomes an industry norm. Alternatively, some protocols mitigate risks through real-time monitoring or automated rollback mechanisms to ensure security without requiring a long notice period.

(Principle 10): Generally, it makes sense that protocols "include contingency mechanisms to ensure prompt responses to attacks or vulnerabilities," with safeguards to prevent abuse of those emergency powers. This principle addresses the reality that even with audits and best efforts, incidents happen (e.g. hacks, bugs) and protocols need a way to react quickly to protect users. Many DeFi platforms have implemented pause switches or circuit breakers. For example, Aave and Compound both have the ability (via governance or an appointed security guardian) to pause borrowing on a market if a bug or attack is detected, halting further damage. Similarly, Uniswap v2 had no admin controls (fully immutable), but others like SushiSwap initially had emergency admin keys (later governed by multisig) to disable a pool if needed. We note the delicate balance mentioned in the paper: emergency measures should avoid "risk of takeover by entities responsible for temporarily protecting the protocol." In other words, if a privileged user or committee can hit the kill switch, there must be controls on that power. Best practices here include multi-signature authorization for emergency actions, predefined conditions for use of emergency powers, and possibly expiring privileges (e.g. a pause that auto-expires after X days unless extended by broader governance).

This principle can be successfully implemented via DAO-driven security committees and documented incident response plans, rather than through regulatory mandates. In fact, decentralized governance can be very effective. After the April 2022 Beanstalk governance attack, many DAOs instituted faster emergency voting processes for crises. Sharing these lessons industry-wide (perhaps via a consortium that publishes recommended contingency mechanisms) would allow protocols to self-improve. Regulators could help by acting as a coordinator in convening industry stakeholders to develop such contingency frameworks, but the actual design and activation should remain at the protocol/DAO level to be effective.

Additionally, from a legal standpoint, if emergency powers were mandated by law, it could inadvertently centralize control which is contrary to decentralization. The EU's recent Data Act introduced an obligation for a "kill switch" in certain smart contracts for data services, which sparked concern in the blockchain community that it might undermine immutability. Contingency should be encouraged, but not necessarily a legal requirement for every contract, because one of the innovations of this space is the option to have truly immutable, autonomous code. Those protocols choose to handle failure differently,

perhaps via user insurance or new deployments, and that approach should remain valid. Protocols that choose not to implement emergency controls should explicitly disclose this design choice, as such users and stakeholders will be fully aware of its implications.

Q 3: Do you have any comments on the **service compliance** principles set out in the document?

(Principle 11): Ensuring that "a smart contract is accompanied by documentation that accurately describes the services provided, the expected level of quality, the risks, stakeholder responsibilities, and governance mechanisms" is a best practice that improves user understanding and trust. In traditional finance, offering documents or prospectuses serve this role; for smart contracts, a well-written whitepaper or user guide can fulfill a similar function. Users should not have to read raw code to know how a protocol operates and what risks they bear. Many reputable projects voluntarily produce detailed documentation: for example, MakerDAO's "Purple Paper" and risk documentation clearly outline how the system works and what could go wrong (liquidation risks, etc.), and Aave's documentation defines the roles of liquidity providers, borrowers, and the parameters affecting them. Additionally, audit reports are often published, which describe in accessible terms what the code is intended to do and any limitations.

The industry, perhaps via a consortium or open-source template, could develop a standard documentation format for DeFi services, similar to an open standard where certain sections including overview, risk factors, admin powers, upgrade process, and others are expected to be covered. An example initiative is DeFiSafety's transparency questionnaire, which many projects answer and publish to boost their score. By self-adhering to a documentation standard, protocols can effectively "certify" that they have disclosed all relevant information to users, much like how consumer protection laws in finance require clear terms – but here it would be an industry-enforced norm. **Regulators could endorse the importance of documentation, but making it mandatory might introduce liability concerns. For example, would every typo be legally punishable?. Voluntary compliance bolstered by market expectations may be sufficient as projects that hide or misrepresent details face reputational damage and loss of users. One innovative idea is using the blockchain itself for transparency – for instance, embedding a hash of the documentation in the contract or providing a user interface link to the documents.**

(Principle 12): This principle states that a smart contract should charge "a reasonable amount of user fees for the service provided" and be optimized in code such that it doesn't waste resources or gas beyond what security requires. We agree that user-cost considerations are important – exorbitant or opaque fees can harm users and the ecosystem's reputation. However, **caution is required when interpreting "reasonable" fees.** In a decentralized market, fees are often determined by supply and demand (e.g. liquidity provider fees on exchanges, or gas costs based on network congestion). What is reasonable may vary. **We think the spirit of this principle is to discourage unnecessarily complex or inefficient code that leads to high gas costs, and to ensure any protocol fees are transparently communicated.** From a best practices standpoint, gas optimization and cost-efficiency are already a focus for many blockchain developers and projects. So, the developer community is naturally incentivized to optimize code. Lower gas usage attracts more users. Many audits include gas efficiency recommendations as well.

Additionally, with Ethereum's push toward layer-2 networks, protocols are considering cost in their deployment choices (moving to rollups for cheaper transactions), again an industry-driven solution. On the topic of fee levels (like protocol fees or yields taken)**, transparency is key. This overlaps with documentation (Principle 11) – clearly stating any protocol-imposed fees or cuts.** We note that extreme fee structures often

get exposed by users on social media and lose market share to fairer competitors, so the market does provide checks. Therefore, while we embrace the principle of reasonable fees, we advise against any hard regulatory cap or formula for fees. Flexibility and competition yield the best outcome for users. One concrete suggestion is that the certification process (if one exists) could include a performance and cost efficiency review: e.g., as part of a voluntary certification, an auditor might report gas usage per function and compare it to known standards, ensuring no egregious inefficiency. This is similar to how software products might get performance benchmarks as part of quality certification. Such information could be published for users to make informed choices. In a self-certification regime, a project team could even publish their own analysis of costs and why they believe it's reasonable, subject to community validation.

Beyond principles 11 and 12, the discussion in the paper's Part 1 also touches on compliance with legal/regulatory requirements via smart contract design (for example, embedding AML/CFT measures). While not explicitly one of the numbered principles, this is a "service compliance" aspect worth addressing. The report notes that integrating compliance standards like preventing interaction with blacklisted addresses or using zero-knowledge proofs for customer information could facilitate regulated institutions using DeFi. Our view is that technical compliance features should be encouraged on an optional, case-by-case basis rather than imposed universally. Moreover, extending audits to cover regulatory compliance aspects without clear regulatory criteria previously defined could introduce significant uncertainty, leading to potentially subjective evaluations by technical auditors. Regulatory expectations must therefore be clarified beforehand to ensure audits remain objective and focused.

It's positive to see experiments in this area e.g., Aave Arc, which created permissioned liquidity pools where only whitelisted KYC'ed addresses can participate. This was an industry response to institutional demand, achieved without a law but through innovation. Similarly, some projects use identity tokens, verifiable credentials , or soulbound tokens and NFTs proving an address is KYC'ed to restrict certain actions to verified users. These are promising voluntary mechanisms that certain services can adopt to satisfy regulatory-minded participants. However, forcing every smart contract to include blacklist checks or KYC gating would carry significant downsides. It could break composability (contracts expecting free interaction might not interoperate with ones requiring credentials), it introduces centralized control points (who maintains the blacklist?), and it might simply drive truly decentralized activity to less transparent venues.

Q 4: Do you wish to comment on **other aspects** developed in **part 1** of the document?

The working group report references existing initiatives like the EEA EthTrust security standard and others (e.g. ERC-3643 for tokenized assets) as potential bases for smart contract standards. We strongly encourage building on such industry-led frameworks instead of reinventing the wheel through regulation. EthTrust, for instance, not only enumerates known vulnerability checks but also defines multiple levels of security assurance (S, M, Q) with increasing rigor. This stratification is similar to traditional certification schemes such as Common Criteria EAL levels, or maturity levels in ISO standards, and was created by a consortium of blockchain experts under EEA. It shows that the industry has the expertise to develop robust standards when motivated. It could be beneficial for regulators to encourage or otherwise find ways to collaborate with these standard-setting efforts rather than impose an all-new governmental standard. By doing so, any certification regime would be grounded in what practitioners already find useful.

An open, industry-led smart contract standard could explicitly cross-reference such well-known frameworks to gain credibility and completeness. Similarly, in cloud computing, the industry gravitated towards SOC 2 audits and ISO certifications to demonstrate security – not due to regulation, but due to customer expectations and

third-party due diligence. **Perhaps smart contract platforms could do something analogous (e.g. a SOC 2-type report for a DeFi platform's smart contracts covering security, availability, processing integrity).** This would be driven by investor and user demands for assurance. The role of authorities should be to facilitate recognition of these assurances (perhaps allow them to satisfy any regulatory requirements that do exist) rather than micro-manage their creation.

The question of "who can, and should, set standards for certification" was raised in the report. We advocate that standards be set by a broad industry consortium or standards body rather than solely by regulatory authorities. The working group itself is a mix of public and private participants, which is a good model. **One approach could be to establish something akin to an "International Smart Contract Standards Board" under the auspices of an existing global body (maybe the ISO or IEEE, or a new consortium), where blockchain developers, security experts, audit firms, and regulators all contribute to evolving standards.** This mirrors how technical standards in the internet are set (IETF, W3C) – industry-driven but with input from governments and academia.

If instead each national regulator tries to set their own standard, it risks fragmentation. A protocol might meet a French standard but not a German one, etc., which is untenable for global DeFi services. The report rightly notes any regulation should apply at least at European level and we'd go further to say global coordination is needed. A positive example is the Smart Contract Security Alliance (SCSA) which is a group of security firms and organizations (including Quantstamp, MythX, Chainsecurity, etc.) that came together to publish recommended security audit standards. This is the kind of industry consortium that could evolve into a formal standard-setter. Regulators could perhaps contribute observers or expertise, but allow the technical community to drive specifics. Industry-led standards have the advantage of agility. They can be updated whenever new vulnerabilities or techniques emerge, without waiting for legislative cycles. They also usually undergo public feedback (e.g. EthTrust had multiple iterations with community input.

Part 1 also highlights the importance of continuously consulting up-to-date data sources on vulnerabilities and the value of learning from an observable history of failures (presumably to refine standards). We agree that the standard for smart contracts must not be static. One idea is to establish an open registry of smart contract incidents and lessons learned (perhaps maintained by a neutral party or community group). A relevant parallel can be found in NIST's National Vulnerability Database (NVD), or the aviation industry's incident reporting systems, but instead for blockchain. In fact, unofficial sources like rekt.news and academic compilations already track DeFi hacks. Formalizing this into a knowledge base can greatly inform standard development.

For example, if a new type of flash-loan exploit is observed, the standard could be updated to include a check or recommendation against that pattern. The DeFi community often publishes post-mortems of hacks (e.g. the Poly Network hack analysis, the Uranium Finance exploit analysis) – feeding these into training for auditors and developers is crucial. We suggest that as part of any certification ecosystem, there should be a mechanism to rapidly incorporate new threat intelligence. Industry consortia can handle this more efficiently than regulatory processes. Perhaps a certification DAO could even be formed, where members vote to update standards as new information comes in**. The key point is that we should avoid ossifying the standards. A static checklist can become outdated and give false confidence. Instead, a living standard maintained by industry experts (with transparency in revisions) will yield much stronger security outcomes.**

The report notes that the principles were intended to be broad across different execution environments (different blockchains, etc.). It is important to emphasize that any standard or certification needs to account for the diversity of smart contract platforms (Ethereum, Solana, Tezos, Cosmos WASM, etc.). Security considerations can differ (for example, resource metering on Ethereum gas vs. on Tezos, or upgradability features in Cosmos modules). An industry-led approach is naturally suited to handle this diversity, as specialists from each ecosystem can contribute their expertise. A government-mandated single standard might inadvertently be Ethereum-centric or assume an EVM model, which could disadvantage other technologies or fail to cover their unique aspects. **Instead, it may be beneficial to have baseline principles (like the ones given) that are platform-agnostic, supplemented by platform-specific control objectives.** For instance, an Ethereum smart contract certification might include checks for re-entrancy and gas griefing, whereas a Solana program's certification might emphasize memory safety or BPF security. Platform communities (like the Solana Foundation, Tezos community, etc.) can formulate their own extensions of the general standard. This federated model of standard-setting can be coordinated through the consortium approach mentioned. It avoids any one-size-fits-all regulation and leverages the self-regulation within each blockchain ecosystem. In practice, there are already platform-specific security initiatives – e.g., Trail of Bits published a secure Rust (for Solana) guideline, and Tezos has a formally verified standard library.

Part 2 of the document: audit

Q 5: Do you have any comments or additions to make on the **audit methods** set out in part 2-1?

Part 2-1 of the document outlines various audit methods for smart contract certification – including manual code review, automated static analysis, dynamic testing (unit tests, fuzzing), and formal verification. Overall, this overview is appropriate and quite comprehensive. In fact, one of DeFi's strengths has been the creative use of a mix of these audit techniques by the community.

No single audit technique is sufficient on its own and combining them yields the best results. The report acknowledges this by mentioning the range of methods and that the state of the art is "sufficiently advanced" for robust certification. This is true. A typical industry best practice already involves at least two layers of auditing; (1) automated scanning tools and internal testing by the developers, and (2) an independent third-party audit by professional auditors who do manual review and additional fuzzing. For high-value protocols, formal verification is increasingly being added as a third layer. For example, Compound employed formal methods (using Certora and OpenZeppelin) to verify certain properties of their smart contracts after the standard audit process.

Similarly, Ethereum's deposit contract for the Beacon Chain was formally verified to prevent any flaws in that critical piece of code. These examples show the industry voluntarily applying rigorous methods where warranted. Any certification framework should encourage this layered approach. **Perhaps a guidance that multiple distinct audit techniques be used could be part of the standard – but importantly, which techniques to use might depend on the contract's complexity and risk.** The industry could potentially develop a risk-based guideline. For simple contracts managing low value, basic automated analysis and one manual review suffices; for complex or high-value protocols, incorporate fuzzing and formal proofs.

Automated static analysis tools (MythX, Slither, Mythril, etc.) are quite effective at catching common vulnerabilities (reentrancy, arithmetic errors, unsafe external calls). They map closely to principle 1 (avoiding known bugs). It seems reasonable that any serious security audit will include running such tools as they are both low-cost and high-value. The report

notes these are generally prescribed for validating security principles. **One benefit of a certification regime, even if voluntary, is that it could standardize the use of certain tools or checks.** For instance, a certification standard might require that the code passes all checks of a given baseline ruleset (like the SWC Registry rules or OWASP Top 10) with no critical issues flagged. The industry can collaboratively maintain such a checklist. Considering the high frequency of smart contract deployments and updates, a mandatory rigorous certification for all contracts may quickly become impractical, creating bottlenecks. A selective, risk-based approach targeting mainly critical financial or infrastructure-related contracts would be more realistic. Expanding auditor accreditation internationally could also enhance quality and cost-efficiency. **As an addition, it is important to consider open-source tooling.** Many tools are open and community-maintained, which lowers the barrier for developers to self-audit before seeking formal certification. Encouraging their use (maybe through workshops or published how-to guides as part of the certification program) will improve overall security. There is also future potential in leveraging new tech like AI-assisted code analysis, which some start-ups are exploring. A dynamic industry-driven certification can adopt such innovations faster than a static regulation could.

We appreciate that the report includes unit testing and fuzzing. In practice, well-developed projects often come with extensive test suites (sometimes with >90% code coverage) and are deploying fuzz testing to simulate random inputs and economic scenarios. For example, Yearn Finance reported using fuzzing to test their vaults against a variety of market conditions, and this helped discover edge-case bugs. A certification process could require evidence of a robust test suite and fuzzing results. Industry best practice could be to have an independent auditor also run their own fuzz tests to try to break the contract (audit firms like Trail of Bits and ConsenSys Diligence do this routinely).

One innovative development is auditing contests (e.g., Code4rena) where multiple community auditors review a new contract concurrently in a competitive format, essentially a distributed manual + dynamic testing approach. This has proven successful at finding issues that single audits might miss, and it's an industry-born idea. It could be prudent to include the recognition of such novel audit methods as part of any future framework. A project that undergoes a public audit contest or maintains a long-term bug bounty (more on that below) could be seen as meeting certain audit requirements through community vetting.

Formal methods provide the highest level of assurance by mathematically proving properties of the contract (e.g., "this invariant holds after every function"). They are resource-intensive and require specialized expertise, which is currently in short supply. The report mentions formal verification as part of the arsenal. **Formal verification should be recommended for the most critical smart contracts (those holding very large TVL or performing novel financial logic), but not required universally.** In an industry-led standard, one could imagine a tiered certification where the highest tier requires formal proofs of key properties, similar to how EthTrust had an advanced "Q" level requiring holistic analysis.

Already, certain blockchain ecosystems emphasize formal methods (e.g., Tezos and Cardano encourage it through their smart contract languages). A regulator mandating formal verification for all DeFi code would be impractical (not enough experts to do it, and not needed for simple contracts), but a voluntary top-tier certification could include it. Importantly, formal verification efforts benefit from sharing and reuse – for instance, if someone proves an ERC-20 implementation correct, that proof can be reused by others using the same code. This again highlights the value of open industry collaboration. **Any certification knowledge base could include templates or open-source proofs for**

common components (e.g., standard token contracts, math libraries), to lower the burden on individual projects.

The report raises the issue of a shortage of skilled auditors in the market and notes that a broader certification scheme might cultivate a deeper market of auditors over time. Indeed the lack of a deep talent pool is an ongoing concern. The explosion of DeFi in 2020-2021 showed that only a handful of firms (Trail of Bits, OpenZeppelin, Certik, etc.) were available, and they had long backlogs. If certification became mandatory overnight, it would create a bottleneck and exorbitant costs, pricing out small innovators. Instead, it may be wiser to encourage more people to enter the smart contract security field by increasing demand organically (e.g., projects voluntarily getting certified to attract users). Over time, this will indeed create more auditing firms and security experts – as the report suggests, a market response to the need. To that effect, there are already signs of growth. Many new audit firms and independent auditors have appeared in the last couple of years because there's business opportunity. This growth can be nourished further through educational initiatives and possibly an auditor accreditation program led by industry. The community could establish a certification for auditors (not just for contracts), in the way traditional IT has CISSP or certified ethical hacker qualifications.

It is necessary to stress the importance of independent audits. While developers should absolutely do their own testing and even internal audits, an external set of eyes is crucial for objectivity. Many projects have learned this the hard way. Code reviewed only by its authors often misses flaws due to familiarity bias. **Any certification regime (especially if self-certification-based) should require disclosure of audit reports by independent experts. Even in absence of formal regulation, the industry norm has become that projects publish their third-party audit report for transparency.** This practice should be continued and even strengthened. A voluntary certification program could maintain a public registry of projects and links to their audit reports, making it easy for investors or users to verify that an audit was done and by whom, similar to how one can verify a company's financial audit in traditional markets. **Moreover, auditors themselves should follow standards – e.g., using consistent severity ratings for findings, and perhaps using the aforementioned standard checklists so reports are comparable.**

Q 6: Do you have an opinion on the **validity period** of a certification?

Determining an appropriate validity period for a smart contract certification is crucial, as it balances the need for up-to-date security assurance against the practical burdens of frequent re-certification. The working group report suggests that certification should be for a limited time and identifies two possible approaches: a fixed period (e.g., 3 years, by analogy to other frameworks) or a variable period based on the contract's criticality/complexity. It also clearly states that any major changes to the smart contract should result in a new certification, with the initial audit defining what changes are considered major.

Certifications should indeed be time-bound and change-sensitive, but the specifics should be determined in a risk-based, flexible manner – ideally by industry guidelines rather than hard rules. In other words, we support principle 13 from the report: certification tied to a given code version and knowledge state, and expiring after a period.

Software security is not static. New vulnerabilities might be discovered in algorithms, new attack techniques arise, or the threat landscape changes. A smart contract considered secure today might be found vulnerable tomorrow due to factors outside the code (for instance, a breakthrough in cryptography or a change in the underlying blockchain). Therefore, a certification with no expiration could mislead users into a false sense of security. Setting a validity period forces a re-evaluation with fresh eyes and updated tools.

Traditional certifications, like ISO 27001, typically last ~3 years with interim surveillance, and even SSL/TLS certificates for websites expire within 1-2 years to ensure keys/standards stay fresh.
However, 3 years may be too long in the fast-moving DeFi space; many professionals lean towards shorter cycles, that accounts explicitly for the immutability and complexity of contracts, and are likely better to address industry realities (perhaps 1 or 2 years). **One could consider a model where a full re-certification audit is required every 2 years, with a lighter touch annual review in between (similar to an annual SOC 2 Type 2 report in tech, which checks controls yearly). The appropriate duration might also depend on how static the contract is. An immutable contract that has been battle-tested for a year or two without issues might reasonably go a longer interval, whereas a complex upgradable protocol might warrant more frequent review.**

The idea of a variable period based on criticality is very sensible. In practice, this could be implemented via tiered certification levels. A baseline certification for low-impact contracts could last longer (maybe 3 years) and require recertification only on major changes. These would be for contracts holding little value or with very simple logic. This could be followed by a critical certification for high-value or systemically important contracts expiring within 1 year, and also requiring more continuous monitoring. High TVL DeFi protocols might fall here, as well as foundational contracts like bridges, reflecting that these are lucrative targets for attackers.

The criteria for which bucket a protocol falls into can be tied to those proportionality metrics discussed (TVL, user count, etc.). The determination could be made by the certifying body or self-declared with oversight. Again, it could be beneficial to involve industry in the development of any recommendations for this (possibly with regulator input). It also makes sense that any significant change to a smart contract's code or dependencies should invalidate the prior certification until reassessment is possible**. This is analogous to product safety certifications. If you modify the product, you need to recertify the new version.** In practice, many DeFi projects use upgradeable contracts or proxy patterns. **A possible mechanism is that whenever a new implementation is deployed behind a proxy, it must go through an audit/certification process before being considered "certified." Perhaps the certification could be tied to a specific code hash; if the hash changes, the certificate auto-revokes (principle 15 hints at an automated removal when no longer valid)**.

Here, it could be interesting to explore on-chain certificate artifacts which include the hash of the approved code. If the code is upgraded, a new artifact must be issued. This could even be enforced programmatically if desired though that might be complex to universally implement. Importantly, modification would need to be defined broadly, i.e., not just literal code changes but also changes to linked libraries or external modules that the contract heavily relies on may also potentially trigger a review. For example, if a protocol points to a new price oracle or changes a key parameter contract, that could be security-relevant. By contrast, adjustments to non-critical configurations (like UI settings, or interest rate parameters within safe bounds) might not need full recertification. Those could be handled by the proportional approach such as minor changes possibly just notifying the certifier.

Following this, it may be relevant to consider moving from a static certification model to a more continuous attestation model. In other domains, there's a trend towards continuous monitoring instead of periodic re-audits. **For smart contracts, one could imagine an ongoing certification where certain automated checks are continuously run (on-chain or off-chain monitors checking invariants), and the results feed into the certification status, publishable as an on-chain attestation.** For instance, if an invariant is violated in operation (if a liquidity pool accounting doesn't sum up correctly), the

certification could flag itself or suspend. This is quite advanced, but not impossible given the transparent nature of blockchains. However, this might be beyond the current scope, but is nonetheless a space where industry can innovate.

From a practical viewpoint, whatever validity period is chosen, there must be a clear and accessible way for users and intermediaries to know if a contract's certificate is current. The report's principle 15 emphasizes that the certificate should be easily available to users (perhaps via the UI or a public register) and automatically removed when invalid. A user-centric approach is likely the most fruitful. One can imagine a wallet interface showing a green checkmark if a contract is certified and up-to-date, or a warning if the certificate expired. This is analogous to how browsers show warnings for expired TLS certificates. Implementing such a system could be done via an open registry (maybe an Ethereum registry contract or an off-chain database API that wallets can query). The removal or flagging of expired certificates should be automated – meaning once the time is up or a code change is detected, the status flips. **Achieving that likely requires an infrastructure of certificate issuance and monitoring that is probably overseen by some entity (be it a government body or an industry governance mechanism). However, this should not become overly centralized. Ideally, the certificate registry could itself be a decentralized service or a public-good maintained by a consortium, to avoid single points of failure or censorship.**
There may be scenarios where a certificate needs to be extended briefly (maybe a renewal audit is in progress but not finished by expiry, and shutting down interaction with the contract in the interim would be disruptive). Conversely, there may be cases of zero-day vulnerabilities where even before the time is up, a certificate should be suspended. Thus, a governance process for the certification scheme is needed. In an industry-led model, the certification authority or consortium might have the power to issue short grace periods or immediate revocations. If a critical flaw is found in a cryptographic library that affects many contracts, the certifying body might announce that all certifications relying on that library are temporarily suspended pending patching. This is analogous to emergency directives in other fields.

Q7: Would you like to comment on **other aspects** developed in **part 2** of the document?

Among the options, the third scheme seems to be the most palatable and sustainable long-term, where a smart contract provider (developer/team/DAO) conducts the audit and certification of their contract, subject to ex-post oversight.

This essentially means the project is responsible for ensuring it meets the standard and perhaps even issuing themselves a certificate of compliance and regulators or other bodies can enforce against false claims or intervene if something goes awry. This model keeps the onus on the project to maintain security which is appropriate, since they know their system best and avoids creating a bottleneck or single point of failure in the certification process. Additionally, this approach reduces potential entry barriers for new developers or smaller projects, allowing innovation to thrive while incentivizing strong accountability among project teams. Ex-post oversight could involve authorities randomly spot-checking some certified contracts or responding when an incident indicates perhaps the self-certification was not done diligently.

This approach maximizes flexibility and scalability. It allows the industry to innovate in how audits are done as the provider can hire whichever auditor or use whichever tools they deem fit to meet the standard. It also doesn't constrain the throughput by a limited number of official certifiers. and fosters a sense of accountability on the project. If they self-certify, they are effectively staking their reputation and possibly legal liability on the security of their product. That is a strong incentive to do it right, arguably stronger than simply fulfilling a checklist for an external auditor.

The other schemes involve either a government authority issuing the certification (after an audit by an accredited body) or the auditor firm issuing the certification directly as a third party. While these can provide assurance, they have downsides. Public authority issuance is akin to regulators giving a stamp of approval. It may ensure high consistency and oversight, but it is slow and centralized. If every upgrade or new contract had to wait in line for a regulator's approval, innovation would grind to a halt. Authorities might be inundated given the sheer number of contracts. **The report itself notes if the number of certifications is large, a direct authority issuance is unrealistic.** It also concentrates a lot of power in the authority to decide what gets certified – potentially they could refuse certification for reasons beyond technical security (e.g., policy reasons), which may start to resemble a de facto permissioning regime for DeFi. Additionally, authorities might not have the cutting-edge expertise in-house and would rely on auditors anyway, so it adds an extra bureaucratic layer.

Conversely, auditor issuance via accredited auditors is closer to how things like PCI DSS or some product certifications work. Approved independent auditors assess and directly certify compliance. It is more scalable than the regulator option and leverages market competition among auditors but still introduces potential bottlenecks and conflicts. **If only certain accredited firms can certify, smaller audit start-ups or community auditors might be excluded, possibly reducing competition and increasing costs.** It could become like the financial audit industry – a few big firms dominating, which might drive up fees beyond the reach of new projects. Moreover, auditors being commercial entities might face conflicts of interest – e.g., an auditor could be lenient on a paying client to keep business, undermining trust in the cert. If they are given quasi-regulatory power to issue certifications, the oversight of auditors becomes critical (who watches the watchers?).

In contrast, self-certification with strong penalties for false claims might actually yield more honest outcomes because the project cannot just buy a clean report and has to stand behind it. Today, voluntary certification exists without being mandated, including tech companies voluntarily getting SOC 2 reports from auditors to prove security to clients and not because the law requires SOC 2 per se. In DeFi, many projects already seek multiple independent audits to earn community trust via market-driven certification.

Regardless of scheme, ensuring auditors are competent is important. The report leans towards public authorities having a role in accrediting auditors and validating standards. Even under self-certification, regulators might say "if you use an external auditor, they must meet certain qualifications." A balanced approach may be where industry can establish an accreditation or rating system for audit firms (perhaps in collaboration with agencies like ANSSI in France or ENISA in Europe). In any case, global acceptance would be crucial for the uptake of these certifications. If an audit firm in the US or Asia audits a contract used in Europe, will that be recognized or would the EU require an EU-accredited firm to redo it? Requiring local accreditation could slow things and cause redundancy. It's better if there's mutual recognition of competent auditors internationally, something that could be handled via industry partnerships or international MoUs between regulators.

A major risk of heavy mandatory certification is cost. Security audits are already expensive (tens of thousands of dollars for a basic contract, up to hundreds of thousands for complex systems). If a regulatory certification adds administrative overhead, costs could rise further. This could barrier-out small open-source developers or start-ups. Many innovative DeFi ideas come from small teams or even individuals. If they faced a compliance bill before they can even deploy or get users, many would simply not attempt it or they would deploy anonymously/unofficially, which is worse for oversight. Self-certification mitigates this as teams can do their best with the resources they have, perhaps starting with a lighter security review and improving over time. The market can decide if that's acceptable (maybe only small funds initially until they

prove security). A rigid certification requirement from day one would push these experiments underground or out of regulated markets.

An added benefit of self-certification plus oversight is that authorities can focus on the bad actors or negligent cases rather than gatekeeping everyone. If a project self-certified but clearly did nothing and still got hacked in a basic way, the authority could investigate and perhaps sanction them for misrepresentation or for harm caused to users (if within jurisdiction). This is reactive rather than preventive, but it targets actual problems. One might argue this is too late for the users who lost money, but even a mandatory cert regime cannot guarantee no losses – it can only reduce likelihood. With ex-post enforcement, the threat of action encourages diligence without requiring upfront permission for every deployment. This approach also aligns with how MiCA handles some compliance. MiCA will require crypto-asset whitepapers to have certain information but it's not an approval regime. The issuer is liable if they mislead.

Part 3 of the document: regulatory avenues

Q 8: Do you have any comments or remarks on the developments relating to the **regulatory bases** (objectives, scope, proportionality criteria, different possible regulatory schemes)?

The report identifies customer protection and fostering confidence in DeFi as the main goals of a certification regulation. We fully agree these are important objectives. In fact, they mirror objectives in MiCA and traditional financial regulation – ensuring investors/users are not exposed to undue risk and that they can trust the services they use. We believe these objectives can be achieved effectively through industry-led standards and transparency, supplemented by targeted oversight, rather than heavy mandatory certification of every smart contract. Building trust in DeFi is crucial for its growth. Voluntary certification or audits (market-driven) already serve that to an extent – users tend to trust audited/certified protocols more. A regulatory framework should reinforce trust by perhaps making information about security readily available and setting baseline expectations, but it must be careful not to equate trust solely with government approval. In other words, the objective should be phrased as ensuring that robust security practices are widely adopted and that users are informed about the security status of protocols.

This is slightly different from implying that if it's not certified by the government it's untrustworthy or illegal, which could send the wrong message and drive users to shadow markets. We also note an objective of such regulation could be to provide legal clarity for DeFi developers and participants. Currently, the status of liability if a smart contract fails, or the standards they should meet, is unclear. A framework could clarify that following certain best practices (like obtaining a certification or audit) would be a defense against negligence claims, for instance. That kind of clarity would encourage compliance without being forceful.

The paper suggests limiting the scope to DeFi services that are analogous to financial services and excluding non-financial uses (like pure NFTs) and things already regulated under MiCA like stablecoins. **We support a clearly delineated scope, otherwise the rule could inadvertently cover a huge swath of unrelated blockchain activity.** Focusing on DeFi makes sense because that's where there is significant user funds at risk and a regulatory gap. Furthermore, given the inherently global nature of DeFi, it's essential that any certification framework aligns with or anticipates international standards to avoid regulatory fragmentation, which could otherwise disadvantage local innovation. **Even**

within DeFi, scope should probably focus on protocols that are open to the public or have a significant user base. Private or experimental contracts (like someone's hackathon project that technically offers a financial function but only to themselves) shouldn't need formal certification. MiCA covers intermediated crypto-asset services (exchanges, custodians, etc.) and certain tokens (asset-referenced and e-money tokens). Those entities already have operational resilience obligations and oversight.

A crypto exchange under MiCA must have policies to ensure the safety of the assets it deals with, which presumably includes vetting any smart contracts it interacts with. It would be redundant or contradictory to also require the underlying contracts such as the multi-sig wallet they use to separately be certified by another regime. So, indeed exclude or harmonize with MiCA. Stablecoins and especially e-money tokens under MiCA are issued by regulated entities who must manage risk; if their smart contract is critical, then presumably, the regulator can already ask about its security under MiCA supervision. Additional certification might not be needed, or if it is, it should be integrated into the MiCA process for efficiency.

We strongly support incorporating proportionality to modulate requirements based on risk. This concept is prevalent in financial regulation where small payment institutions get lighter requirements than big banks, and it's wise to apply it to DeFi as well. The criteria mentioned – Total Value Locked, transaction volume, number of users – are reasonable proxies for the impact level of a protocol. This tiering prevents the regulation from being a one-size-fits-all sledgehammer and encourages small players to innovate knowing that if they become big, they should then step up their compliance accordingly – which is fair and often easier to do once you have resources from success.

The document outlines three schemes: optional certification, mandatory for all, mandatory with proportionality. **Our stance is to lean towards the optional certification scheme, with market incentives to encourage uptake.** Optional (voluntary) certification could mean a framework is established and a certificate is offered (maybe by a regulatory body or recognized entity) to those protocols that apply and meet the criteria. If this certification proves its value (certified protocols have fewer incidents, users flock to them, etc.), it will naturally become an industry norm. If it doesn't prove useful, then making it mandatory would have just added burden without benefit. The report's analysis concluded that fully optional might not give enough guarantees to users, whereas fully mandatory for all might hamper innovation.

One of our overarching concerns is that a mandatory certification requirement could inadvertently centralize the DeFi ecosystem. If only a handful of protocols get certified (perhaps due to cost or strategic choice), users might flock to those few, and liquidity could concentrate there, reducing diversity. Smaller, innovative projects might not get traction or might need to partner with big ones to survive, which could reduce competition.

Q 9: What is your opinion on the discussion developed in III-1.2.3 tending to reconcile the certification of **protocols** and that of the underlying smart **contracts**?

A DeFi "protocol" often consists of multiple interdependent smart contracts working together. For example, consider a decentralized exchange like Uniswap: it has many liquidity pool contracts, a factory contract, and possibly router contracts. Certifying just one contract in isolation such as a single pool would be myopic; one needs to consider the system as a whole. Indeed, certifying individual smart contracts without fully taking into account their interdependencies could miss critical cross-contract vulnerabilities. A systemic approach ensures that certification truly captures the holistic security of interconnected contracts. The report's proposal to certify at the protocol level means evaluating how all these contracts interoperate to deliver the service. This is crucial

because certain security or governance properties only emerge at the system level. For instance, an automated market maker (AMM) might have each pool contract secure on its own, but the protocol-level property could be something like “the pools collectively maintain invariant X” or “the governance can only add new pools via the factory.”

If you certified each contract separately, you might miss cross-contract issues like re-entrancy loops between contracts, or an upgrade in one contract affecting another’s security. If certification were to be conducted, it should consider the protocol architecture and how contracts call each other, any centralized components (or off-chain oracles) that the overall service relies on, etc.

Within that holistic view, each smart contract can still be assessed for its standalone security against vulnerabilities through code-level checks. The report indicates each contract’s certification would be “granted on the basis of security elements alone.” This suggests that while governance and compliance are looked at as a whole, individual contracts are mainly judged on bug-free, secure code. That approach makes sense. The code audit covers individual contracts and their direct interactions, whereas governance/compliance principles often make sense only in the context of the entire protocol (e.g., how upgradeability is governed which might involve multiple contracts).

It is also important to define what constitutes the “protocol” for certification. In DeFi’s composable world, protocols often interconnect. For example, a yield farming protocol might deposit funds into another lending protocol. Should the yield farm’s certification depend on the lending protocol’s certification? If not, users might get a false sense of security. The yield farm could be secure in itself but lose funds due to a hack in the lending protocol it integrated. If yes, it creates a chain of certifications which can be complex. One protocol’s status depends on another’s. The report’s discussion likely touches on this tricky point of composability. Reconciling this, perhaps the certification at protocol level should require disclosure of external dependencies and an assessment of their risks. For instance, protocol X’s certification could come with a caveat: “Protocol X relies on Protocol Y for pricing and the security of that external protocol is not covered by this certification.” Or ideally: “Protocol X’s certification is only valid when interacting with certified protocols Y and Z; use with uncertified ones is at users’ risk.”

Some services are essentially one smart contract (e.g., an AMM like Uniswap v1 was basically a set of identical pool contracts without a central brain; or an on-chain options contract might be standalone). In those cases, the protocol-level certification and contract-level certification collapse into one. The contract must meet security, and governance/compliance is minimal. If it’s immutable, governance principles might not apply except to say “no governance possible, which eliminates certain risks but also certain safeguards”. The framework should allow that an isolated smart contract can be certified by itself, with the understanding it satisfies all principles (if it has no governance mechanism, you might mark N/A for governance principles but that itself is a design choice that should be made clear).

Many protocols evolve by adding new contracts for new features or retiring old ones. Certification should be flexible to update the scope. If a certified protocol adds a new module, that module’s code would need to be audited and added to the certification. Possibly a “delta certification” just for the addition could be done, rather than redoing everything. **Again, an industry-led process can figure out streamlined ways like an amendment to the certificate.**

Part of protocol-level certification is governance and contingency mechanisms. If a protocol is certified on governance principles, that implies the entire governance process (which might be off-chain voting, on-chain DAO, or a multi-sig etc.) was reviewed and deemed

satisfactory. That certification of governance shouldn't be applied contract by contract, it only makes sense at the whole protocol. For instance, you wouldn't certify one contract's governance – you certify the protocol's governance structure that oversees possibly many contracts. So, it is right to separate that out. Now, if one of the underlying contracts has its own specific administrative controls (like some contracts might have an administrative key separate from the main governance in the form of a time-lock on each contract), the auditors need to ensure those align with the overall governance model. This is a nuance but important. The protocol certification should confirm that all privileged roles across all contracts are accounted for in the governance model, ensuring no hidden backdoors. Generally, industry auditors are well-suited to do this type of cross-contract analysis.

There's an even more foundational layer: the blockchain itself is the platform. The security of smart contracts depends on the security of the blockchain protocol. The report possibly touches on whether base layer protocols should/could be certified. **It might be outside scope, but for completeness, major blockchain protocols undergo their own audits. Ethereum's client software, consensus algorithms, etc., are heavily reviewed by core developers and researchers. There's no formal "certification" of Ethereum or others by any authority. The assumption is that large open-source projects have enough eyes and bug bounties to be robust. If a DeFi certification scheme is introduced, it should state that it assumes the underlying layer is secure. If the layer fails due to a 51% attack or bug in the EVM, that's beyond the scope of the DeFi certification to manage.** This matters because a contract could be perfectly secure code-wise but lose funds due to a blockchain reorganization or exploit at the consensus level. So users should be aware that certification of a contract/protocol doesn't guarantee the underlying chain's security. This again should be perhaps disclosed in the certification documentation ("This certification assumes normal operation of the Ethereum blockchain and does not cover layer1 risks").

Given all that, our opinion is that the approach of certifying at the protocol level for comprehensive properties, and at contract level for code security, is the correct one. It aligns with how audits are practically done as auditors audit all components and then give an overall opinion on the system's security. It prevents a piecemeal view that could miss interactions. It also ensures governance and compliance principles which often are system-wide are evaluated in context. Any certification framework should therefore explicitly define protocol vs contract to avoid confusion. For example, a "protocol" could be defined as a set of smart contracts that collectively provide a service and are governed together. In some cases, identifying the boundaries might be tricky (is a DAO with multiple products one protocol or many?). That might be left to the applicant to propose and the certifier to agree on. Possibly a single project team might have multiple protocol certifications for different products they deploy (one for their DEX, one for their lending platform, etc., even if under one DAO umbrella).

Finally, reconciling these two levels also has a benefit for user understanding. A user might not care about each contract, but they care that "Protocol Y" is certified. Within that, all pieces are covered. It simplifies the communication, you advertise protocol certifications, not dozens of contract certifications. Meanwhile, for integrators or developers, having contract-level information (like a specific library is certified) is useful for building new protocols. So, both levels of information dissemination are valuable.

Q 10: Do you wish to comment on **other aspects** developed in **part 3** of the document?

Our perspective is that while regulators (ACPR, AMF, ESMA, etc.) should have visibility on the process, the day-to-day operation and evolution of the certification standard should be left to an industry-led governance body.

We want to note there are other regulatory approaches to DeFi being discussed, such as on-chain monitoring of protocols for risk or requiring protocols to have insurance/reserve funds to cover hacks. While outside the direct scope of certification, these could complement it. For example, **if a certified protocol carries insurance for users against smart contract failure, that provides an extra layer of protection.** Perhaps regulators could encourage or even require high-risk protocols to have some insurance or emergency fund (maybe community-governed) to compensate victims of a failure. This isn't a substitute for prevention via security, but a backstop. Some protocols already have such funds (e.g., MakerDAO's surplus buffer acts as a kind of insurance for certain losses).

Finally, it would be important to ensure that any certification framework is accessible to a diverse set of participants, including open-source hobbyist projects, not just VC-funded start-ups. If only well-funded entities can afford to certify, there is a risk of missing out on grassroots innovation and perhaps skewing the DeFi ecosystem towards cartelization and oligopolies. Additionally, if a mandatory certification regime is considered, clear incentives or legal protections should be offered to certified developers to prevent innovation from migrating to jurisdictions with lighter regulatory burdens. An experimental or phased approach to certification could also be beneficial to evaluate effectiveness before broader implementation One way to support inclusion is perhaps a tiered pricing or grants for audits for small projects. **Another is simplified self-certification templates for simple contracts such as an automated audit pipeline that covers basic ERC-20 tokens cheaply.** Indeed there are efforts in the community for automated audit tools; those could be leveraged to give a quick "green light/red light" for simpler cases).

About Adan

Adan brings together 200 professionals – new players and established companies – who are developing innovation and the use of Web3 in all sectors of the economy on a daily basis. By removing obstacles to their growth and competitiveness, Adan is working for the emergence and influence of French and European champions, in the service of our digital sovereignty.

Adan promotes a regulatory framework that is appropriate, proportionate and a driver of innovation, as well as a better understanding of new blockchain technologies, crypto-assets and the opportunities they offer.

Website : www.adan.eu
LinkedIn : Adan
X : @adan_asso

Contact presse :

- Jules Dubourg jules.dubourg@adan.eu
- Lucas Cherfils lucas.cherfils@adan.eu