A Transparent Certification Scheme Based on Blockchain for Service-Based Systems

Nicola Bena*, Marco Pedrinazzi*, Marco Anisetti*, Omar Hasan†, Lionel Brunie†
*Department of Computer Science
Università degli Studi di Milano
Milan, Italy
Email: *firstname.lastname*@unimi.it, marco.pedrinazzi1@studenti.unimi.it
†University of Lyon, CNRS
INSA Lyon, LIRIS, UMR5205, F-69621
Lyon, France
Email: *firstname.lastname*@insa-lyon.fr

***Abstract*—Modern service-based systems are characterized by applications composed of heterogeneous services provided by multiple, untrusted providers, and deployed along the (multi-) cloud-edge continuum. This scenario of increasing pervasiveness, complexity, and multi-party service recruitment urgently calls for solutions to increase applications privacy and security, on the one hand, and guarantee that applications behave as expected and support a given set of non-functional requirements, on the other hand. Certification schemes became the widespread means to answer this call, but they still build on old-fashioned assumptions that hardly hold in today's services world. They assume that all actors involved in a certification process are trusted "by definition", meaning that certificates are supposed to be correct and be safely usable for decision-making, such as certification-based service selection and composition. In this paper, we depart from such unrealistic assumptions and define the first certification scheme that is completely transparent to the involved actors and significantly more resistant to misbehavior (e.g., collusion). We design a blockchain-based architecture to support our scheme, redefining the actors and their roles. The quality and performance of our scheme are evaluated in a case study scenario.**

***Index Terms*—assurance, blockchain, certification, cloud, security, service**

I. Introduction

More than a decade ago, service-based architecture went mainstream and applications used to be implemented as precise compositions of services exposing a structured, fine-grained interface. Today, a "service" is not necessarily a server-side software exposing some APIs. It can also be a set of heterogeneous devices, an opaque data source, an ML-based service, or an entire distributed system [1]. Service-based systems are widespread and at the basis of modern, *smart* ecosystems, from agriculture [2] to smart energy grids [3] and transportation [4], to name but a few. This scenario of increasing pervasiveness, complexity, and long (software) supply chains urgently calls for solutions to increase application security, on the one hand, and to ensure that the application and its security countermeasures work as expected and demonstrate compliance to non-functional requirements, on the other hand.

In the last decade, security assurance, in general, and certification, in particular, became the premier solution to answer this call [5]. Certification schemes provide the means to evaluate whether a target (composite) application supports a given non-functional property and, eventually, release a certificate. Along the years, new certification schemes have been introduced to meet the peculiarities of distributed systems, from the *application* (the focus of this paper) [6], [7] to the *infrastructure layer* [8], [9], and, recently, machine learning [10], [11]. Despite these innovations, the majority of existing certification schemes follow a trust model which is increasingly questioned [5], [12], [13], [14], taking for granted that participating actors are honest. In detail, existing schemes assume that Certification Authorities (CAs) honestly manage the scheme, and other actors (e.g., service providers, cloud users) fully trust the certification process executed in an opaque manner by accredited bodies. Certificates are supposed to be unforgeable, accurate, and a sound basis for decision-making (e.g., selection, composition). This assumption hardly holds in the real world where services are recruited from multiple, untrusted providers across national and regulatory boundaries. In turn, this assumption impacts on the quality of certification as a whole and certification-based decision-making, leading to bad decisions and sub-optimal certification-based life cycle management. In addition, it may create market distortions, for instance, by falsely increasing the legitimacy of a certified provider and service [13], [15], [16].

The certification scheme in this paper aims to fill in the above gap, departing from the assumption of blind trust. Our scheme is trustworthy, completely open and transparent to any involved actors. It is executed within a blockchain as a set of smart contracts, ensuring that certification evidence is collected from the real world using specific countermeasures to nullify colluding actors; on- and off-chain storage then guarantees traceability and immutability of *all* actions that brought to the release of a certificate.

The contribution of our paper is twofold. We first design a blockchain-based certification architecture to support our certification scheme, mapping the actors and their roles in traditional certification to actors and components in the blockchain. We then move the entire certification process on

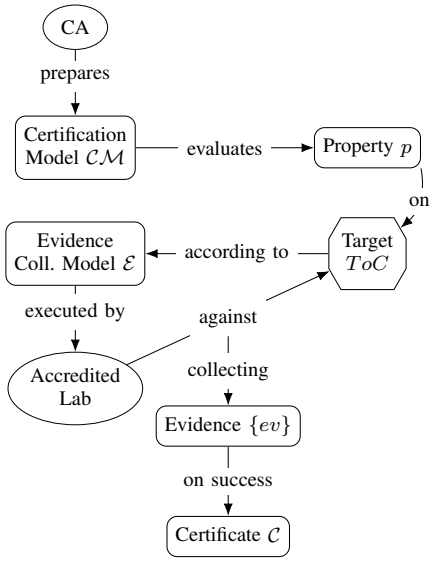


Fig. 1. Traditional certification [5].

the blockchain as a set of smart contracts, including specific constructs to increase transparency and prevent collusion.

The remainder of this paper is organized as follows. Section II discusses the state of the art while Section III its gaps. Section IV describes the certification process, while its implementation and results in a case study are described in Section V. Section VI presents an experimental evaluation of the proposed approach. Section VII discusses the main findings. Section VIII describes related work, while Section IX draws our conclusions.

II. Traditional Certification Scheme

A certification scheme implements the certification process to verify whether a target of certification ToC (e.g., a cloud service) supports a given non-functional property p (e.g., confidentiality) according to some evidence $\{ev\}$ (e.g., testing, monitoring) on the ToC behavior [5], [7]. The certification process in Figure 1 begins when a Cloud Service Provider (CSP) asks a Certification Authority (CA) to certify one of its services (ToC) for a given property (p). The CA prepares the *certification model* $\mathcal{CM}$, defined as follows [7].

Definition 1. *A certification model $\mathcal{CM}$ is a tuple $\langle p, ToC, \mathcal{E}, \mathcal{F}, \texttt{signature}_{CA}\rangle$, where* i) *$p$ is the non-functional property to be certified on* ii) *the target ToC according to* iii) *an evidence collection model $\mathcal{E}$ detailing the (test-based) evidence to be collected from ToC;* iv) *$\mathcal{F}$ is a Boolean function evaluating the collected evidence;* v) *$\texttt{signature}_{CA}$ is the signature of the CA that prepared $\mathcal{CM}$.*

Second, evidence $\{ev\}$ is concretely collected by an *accredited lab* from ToC and, upon successful evaluation (i.e., $\mathcal{F}(\{ev\})$=✓), a certificate $\mathcal{C}$ is finally awarded, as follows.

Definition 2. *Let $\mathcal{CM}$ be a certification model and $\{ev\}$ the evidence collected according to it. The corresponding certificate $\mathcal{C}$ is a tuple $\langle \mathcal{CM}, \{ev\}, \texttt{signature}_{AL}\rangle$, where* i) *$\mathcal{CM}$ is the certification model followed by the accredited lab;* ii) *$\{ev\}$ is the collected evidence;* iii) *$\texttt{signature}_{AL}$ is the signature of the accredited lab that collected evidence.*

Signatures on $\mathcal{CM}$ and $\mathcal{C}$ bind the two artifacts to the actor who created $\mathcal{CM}$ (CA) and the one who executed it (accredited lab), making such artifacts *traceable*. The process should be then re-executed according to the ToC life cycle [17].

Example 1. *Let us consider a Machine Learning as-a-Service cloud service s_1 (e.g. [10]). It receives data from end users, trains, and deploys an ML model on users behalf. Being training and inference data two of the most important assets, the CSP wants to certify the service for property* confidentiality-in-transit *(p_{conf}). The CA prepares a certification model $\mathcal{CM}$=$\langle p_{conf}, s_1, \{t_1, \dots\}, \text{AND}, \texttt{signature}_{CA}\rangle$, where t_1 is a test case part of $\mathcal{E}$ analyzing the libraries used in s_1 looking for CVEs affecting p_{conf}, and* AND *is the evaluation function requiring all evidence to be successfully collected. The CA delegates evidence collection according to $\mathcal{CM}.\mathcal{E}$ to an accredited lab. For instance, the evidence collected according to t_1 contains the list of CVEs found in s_1. Let us assume that such list is empty and $\mathcal{CM}.\mathcal{F}(\{ev\})$=✓, a certificate $\mathcal{C}$ is finally awarded to the service.*

III. Motivations and Scenario

Existing certification schemes builds on 5 main Assumptions as follows.

- **A1) Honest behavior of all actors.** All actors act honestly not attempting to break the certification scheme rules [7].
- **A2) Complete trust in the CA and accredited lab.** Following Assumption A1, CA and accredited lab are fully trusted in their activities, from the definition of a correct certification model to the collection of evidence.
- **A3) Opaque certification process.** There are no verifiable, open guarantees on the process that brought to the awarding of a certificate except the trust in the accredited lab according to Assumption A2.
- **A4) Undefined life cycle of certification artifacts.** The integrity of certification artifacts (i.e., certification model, evidence, certificate) is an open challenge [12]. Artifacts can be tampered compromising their truthfulness and impacting on the soundness of certification-based life cycle management.
- **A5) Chain of trust from the CA to the certificate.** The chain of trust links a certificate to its certification model and CA and accredited lab. Following Assumptions A3 and A4, no verifiable guarantees link real collected evidence to its certificate and certification model.

Let us consider the increasingly prevalent multi-cloud scenario [18]. CSPs across different countries and regulations release services and share resources. End users and CSPs implement composite applications (applications in the following) recruiting services implemented by themselves and other CSPs. In this scenario, service selection and composition are grounded on certified non-functional properties [5], [7]. Each CSP has its "go-to" CA and accredited lab for this purpose.

CSPs, CAs, and accredited labs do not have any strong trust relationships, but should trust each other (questioning A1, A2) in a blind manner (questioning A3, A5), assuming that certification artifacts can circulate safely across countries and regulations silos (questioning A4). Although appealing, the lack of trust between CSPs is one of the main hurdles towards multi-cloud applications [18]. Only a fraction of certificates is really trusted in practice, depending on each actor's point of view. The confidence in the application behavior is thus only partial. A CSP may build an application choosing services certified by a trusted CA (if any), or not certified at all, discarding services whose non-functional properties are stronger but come from an untrusted CA or lab.

Altogether, Assumptions A1–A5 reduce the practical value of certification, negatively impacting on applications soundness. In the remainder of this paper, we show how to move from a blind-trust to a fully trustable, blockchain-based certification scheme finally relaxing Assumption A1–A5.

Example 2. *Let us extend Example 1 to the composite, multi-cloud scenario. Different, cross-country CSPs join their forces and provide certified services to build an ML application working on sensitive data. There is no trust relationship between the CSPs. The fact that services to be selected are certified for confidentiality is of little usage due to the lack of trust and transparency of the certification process, making it difficult to create a secure, trustworthy application.*

IV. Blockchain-based Certification

Our blockchain-based certification scheme builds on a loose federation of multiple actors. It implements the certification process as a set of smart contracts on a blockchain. The latter is public and permissionless due to the need of public access, following the best practices in literature [19].

Figure 2 shows the actors and their correspondence in traditional certification (in bold). They are: *i)* CA managing identities in the blockchain (*CA Block.* in Figure 2); *ii)* CAs in charge of the certification process (*CA Cert.* in Figure 2 and *CA* in the following); *iii)* CSPs developing services to be certified (*CSP* and s_1–s_5 in Figure 2); *iv)* blockchain, its *oracles* and *storage*, implementing the certification process (*Blockchain*, *Oracles*, and *Evid. Storage* in Figure 2).

Figure 3 shows the 4-step process of our approach.

It consists of an initial preparatory step (Section IV-A) and three steps forming the certification process (Section IV-B–IV-D). In the first step (Step (0) in Figure 3), the architecture is deployed. It includes the blockchain, a storage to save evidence, and a set of oracles provided by some oracle providers. Oracles are computing nodes collecting evidence off-chain [20], mimicking accredited labs. Each actor is identified and receives a public/private key pair from CA Block. Finally, smart contracts supporting the certification processes are deployed (i.e., inserted) in the blockchain (Table I(b)).

A certification process (Steps (1)–(3) in Figure 3) starts when a CSP wants to certify one of its services. It asks a CA to prepare a certification model $\mathcal{CM}$ as a smart contract

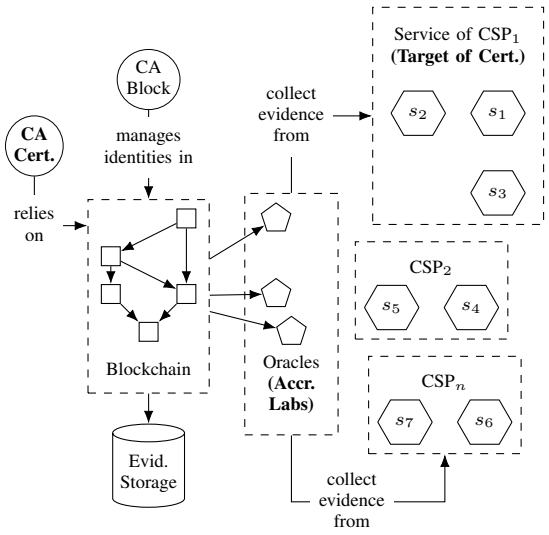


Fig. 2. Blockchain-based certification.

(Table I(a)), which is deployed in the blockchain with full visibility (Step (1) in Figure 3) (Assumptions A3, A4). The CSP executes such contract, collecting evidence from the service (Step (2) in Figure 3). Evidence is concretely collected through oracles; a *policy* guarantees that colluding actors do not impact on evidence correctness (Assumptions A1, A2). Finally, collected evidence $\{ev\}$ is evaluated using evaluation function $\mathcal{CM}.\mathcal{F}$: if the output is ✓, a certificate $\mathcal{C}$ is released (Step (3) in Figure 3). $\mathcal{C}$ is a smart contract pointing to $\{ev\}$ stored off-chain. All certification artifacts (i.e., $\mathcal{CM}$, $\mathcal{C}$, $\{ev\}$) and the process at the basis of their creation are traceable and immutably stored (Assumptions A3, A4, A5).

In the following, we detail each of the above steps.

A. Bootstrap

Step Bootstrap is executed once to set up the blockchain (Step (0) in Figure 3). First, oracles are made available by oracle providers representing accredited labs. Oracles are the bridge executing actions in a trusted manner outside the blockchain. We assume that every actor agrees that the available oracles are part of the blockchain, without necessarily trusting such oracles.

Second, two smart contracts are deployed by a CA of choice: `VRFv2SubscriptionManager` and `VRFv2Consumer`. They are used later during certification.

Third, each CA deploys its version of contract `Probes`. It lists the *probes* the CA implemented through oracles. A probe is a testing script collecting evidence from a target; each probe is referenced by a *function* in `Probes`.

Finally, each CA deploys contract `Coordinator`. It is used to manage oracle execution and evidence collection during certification.

Example 3. *Following Example 2, let us assume the presence of two CAs (CA_1 and CA_2) and three accredited labs (AL_1–AL_3). Each lab provides 5 oracles increasing certifi-*

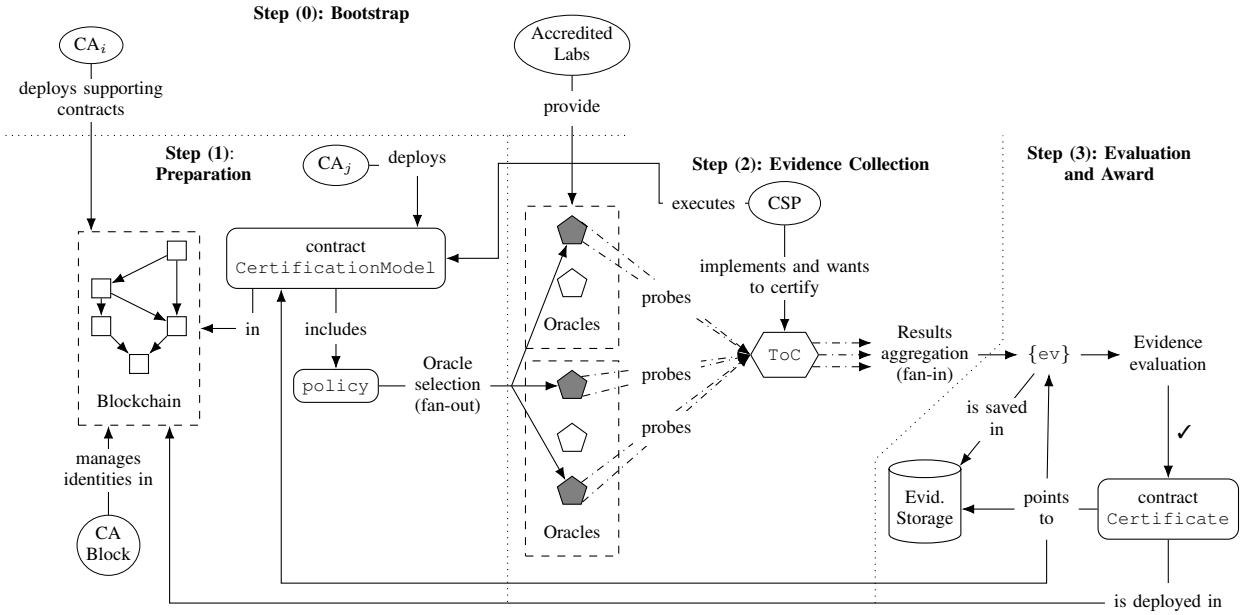


Fig. 3. Blockchain-based certification process.

TABLE I
SMART CONTRACTS.

Name	Depl. Step	Usage Steps	Description
`CertificationModel`	Step (1)	Steps (1)–(3)	Certify a property on a target
`CertificationModelExecution`	Step (1)	Steps (1)–(3)	Execute a `CertificationModel`
`Certificate`	Step (3)	Step (3)	Final certification process artifact

(a) Core contracts

Name	Depl. Step	Usage Steps	Description
`VRFv2SubscriptionManager`, `VRFv2Consumer`	Step (0)	Step (1)	Generate verifiable random numbers
`Probes`	Step (0)	Step (2)	Expose probes as functions then sent to oracles for execution
`Coordinator`	Step (0)	Steps (1), (2)	Define the *policy* for probes execution and retrieve their result

(b) Supporting contracts

cation robustness (e.g., against collusion), denoted as $o_1, \ldots, o_{15}$*, where* $o_1, \ldots, o_5$ *belong to* AL_1 *and so on. Two probes are implemented:* `HTTPS-Strength` *checking that HTTPS is well-configured in the target service, and* `HTTPS-Heartbleed` *checking the presence of vulnerability* Heartbleed*. During Step Bootstrap,* CA_1 *deploys contracts* `VRFv2SubscriptionManager` *and* `VRFv2Consumer`*, both CAs deploy contracts* `Probes` *and* `Coordinator`*. Each contract* `Probes` *contains functions* `executeHTTPS-Strength` *and* `executeHTTPS-Heartbleed` *referencing the respective probe implemented in oracles.*

B. Preparation

The certification process starts when a CSP wants to obtain a certificate for one of its services. The CSP selects a CA that prepares a $\mathcal{CM}$ (Step (1) in Figure 3). $\mathcal{CM}$ is defined in terms of contract `CertificationModel`, as follows.

Definition 3. *Let* $\mathcal{CM}$ *be a certification model in Definition 1, the corresponding contract* `CertificationModel` *is a tuple* $\langle p, \texttt{ToC}, \{\texttt{probe}_1, \ldots, \texttt{probe}_n\}, \texttt{policy}, F, \texttt{signature}_{CA}\rangle$*, where*

- p, `ToC`, F *are the counterparts of the same elements in* $\mathcal{CM}$*;*
- $\{\texttt{probe}_1, \ldots, \texttt{probe}_n\}$ *is the set of probes to be used to collect evidence, corresponding to* $\mathcal{CM}.\mathcal{E}$*;*
- `policy` *defines how probes* $\{\texttt{probe}_i\}$ *are spread to oracles and their results aggregated (Definition 4); and*
- $\texttt{signature}_{CA}$ *is the signature of the CA that prepared and deployed the contract.*

Element $\{\texttt{probe}_1, \ldots, \texttt{probe}_n\}$ specifies the probes to be used to collect evidence, as functions implemented in the contract `Probes` defined by the current CA. The same $\texttt{probe}_i$ in a `CertificationModel` is executed by multiple oracles to increase redundancy and prevent collusion, as specified in element `policy` as follows.

Definition 4. *A* policy `policy` *is a pair* $(\{o_1, \ldots, o_n\}, \texttt{Agg})$*, where*

- $\{o_1, \ldots, o_n\}$ *is the set of the chosen oracles for contract*

`CertificationModel`, *that is,* each probe therein is executed on all selected oracles *(*fan-out*); and*
- `Agg` *is function taking as input the evidence collected by each probe and oracle and returning as output an individual evidence for each probe (*fan-in*).*

The CA instantiates element `policy` by invoking a function in contract `Coordinator`. The oracles are randomly selected using *verifiable number generation* through contracts `VRFv2SubscriptionManager` and `VRFv2Consumer`.

Finally, contract `CertificationModel` is deployed together with contract `CertificationModelExecution`: the latter drives the execution of the former.

Example 4. *Following Example 3, let us assume that* CSP_1 *asks* CA_1 *to certify service* s_1 *for property* confidentiality-in-transit. CA_1 *deploys contract* `CertificationModel` *as* $\langle$`confidentiality-in-transit`, s_1, {`HTTPS-Strength`}, `policy`, `AND`, $signature_{CA_1}\rangle$. `AND` *indicates Boolean AND.* `policy` *is instantiated through contract* `Coordinator`*, and is defined as* ({o_1, o_4, o_{15}}, `Mode`)*, meaning that each probe in* `CertificationModel` *(i.e.,* `HTTPS-Strength`*) will be executed by the three oracles, and the collected evidence retrieved as the mode of evidence returned by each oracle.*

C. Evidence Collection

The CSP triggers the collection of evidence by invoking a specific function in contract `CertificationModelExecution` (Step (2) in Figure 3). The latter retrieves the necessary information stored in contract `CertificationModel` and invokes the functions corresponding to probes `CertificationModel.`{`probe`$_1$`, ..., probe`$_n$}. We recall that each `probe`$_i$ is concretely a function in contract `Probes`. Requests are then forwarded from `Probes` to oracles `CertificationModel.policy.`{o_i} via contract `Coordinator`. Each selected oracle o_i finally executes each probe `CertificationModel.`{`probe`$_i$} outside the blockchain.

Next, collected evidence {`ev`} is retrieved back on chain. Using contracts `Coordinator` and `Probes`, the set of evidence collected by each oracle for each probe is aggregated, retrieving an evidence for each individual probe according to `CertificationModel.policy.Agg` in Definition 4. Each concrete evidence `ev` consists of a Boolean result indicating the success (✓) or failure (✗) of the corresponding probe, and additional data, called *extradata*, providing detailed insights on the Boolean result (e.g., the output of test cases). Evidence is then stored: *extradata* are saved permanently off-chain in JSON format while their hash on-chain.

Example 5. *Following Example 4,* CSP_1 *triggers evidence collection. Function* `executeHTTPS-Strength` *in Example 3 is invoked, which, in turn, executes the corresponding probe in oracles* o_1, o_4, o_{15} *(fan-out). We recall that oracles are the only means to execute actions off-chain. Let us assume that* AL_3 *is colluding with* CSP_1 *retrieving fake evidence from* s_1. o_1 *and* o_4 *both collect evidence* {(✗, `weak ciphers`)}*, meaning that the property is not supported, opposed to* o_{15} *collecting* {(✓, `HTTPS ok`)}*. Evidence is aggregated (fan-in) according to* `CertificationModel.policy.Mode`*, returning* {(✗, `weak ciphers`)} *thus nullifying collusion.*

D. Evaluation and Award

The result of evidence collection is retrieved according to `CertificationModel.F`, taking as input {`ev`} and returning ✓ in case of success, ✗ otherwise (Step (3) in Figure 3). If the output is positive (✓), a new smart contract representing the certificate $\mathcal{C}$ is deployed as follows.

Definition 5. *Let* `CertificationModel` *be the executed contract in Definition 3 and* {`ev`$_i$} *the collected evidence. The corresponding contract* `Certificate` *is a tuple* $\langle$`address(CertificationModel)`, {`hash(ev`$_i$`)`}, $signature_{CSP}\rangle$*, where*

- `address(CertificationModel)` *is the blockchain address of* `CertificationModel`*;*[1]
- {`hash(ev`$_i$`)`} *is a set containing the hash of each evidence* `ev`$_i$ *stored off-chain;*
- $signature_{CSP}$ *is the signature of the CSP that triggered the process in Step (2).*

We note that in traditional certification, the accredited lab affixes its signature to the certificate. Here, the CSP affixes its signature, because it is the actor that started the process and thus inserted the transaction in the blockchain. The chain of trust and its guarantees are still preserved (Section VII-A).

Example 6. *Following Example 5, evidence* {(✗, `weak ciphers`)} *is aggregated using function* `CertificationModel.F` *defined as* `AND` *(see Example 4). The output is ✗, meaning that a certificate cannot be released. Later, let us then assume that* s_1 *supports the property and collected evidence is* {(✓, `HTTPS ok`)}*. The output of the aforementioned function is ✓: a contract* `Certificate` *is created and deployed in the blockchain. It includes the on-chain address of contract* `CertificationModel` *in Example 4 and the hash of evidence stored off-chain.*

V. Walkthrough and Implementation

We present a complete walkthrough of our approach implemented on Ethereum, the permissionless public blockchain with smart contracts that has the broadest adoption [21]. We implemented smart contracts using language *Solidity* v0.8.7 and wallet application *Metamask*.[2] We relied on the Ethereum test network *Sepolia* for practical reasons.[3] All artifacts are available at https://doi.org/10.13130/RD_UNIMI/FBGMZY.

The walkthrough is based on Examples 3–6, targeting a static website hosted on a Content Delivery Network (CDN) with automatic HTTPS setup. We performed all actions on behalf of each actor.

[1] An *address* is the transaction/contract identifier.
[2] https://metamask.io/
[3] https://ethereum.org/developers/docs/networks#sepolia

```
1  Contract CertificationModelExecution
       // The CM to execute.
2      CM;
3      Constructor (CertificationModel)
4          CM ← CertificationModel;
5      end
       // Launch evidence collection.
6      Function runCertModel public onlyCSP
7          CM.run();
8      end
       // Retrieve evidence from oracles.
9      Function retrieveEvidence public onlyCSP
10         CM.retrieveEvidenceBack();
11     end
       // Evaluate evidence and release
          certificate.
12     Function evaluateAndCreate public onlyCSP
13         result ←evaluationFuncAND();
14         if result =✓ then
15             Cert ←new();
16             Cert.cert_model ←address(CM);
17             Cert.evid ←store(CM.evidence);
18         end
19     end
       // Evaluation function.
20     Function evaluationFuncAND private
21         counter ← 0;
22         foreach evidence[j]∈ CM do
23             if evidence[j]=✓ then
24                 counter ←counter +1;
25             end
26         end
27         if counter =CM.evidence.length() then
28             return ✓;
29         else
30             return ✗;
31     end
32 end
```

Fig. 4. Pseudocode of contract `CertificationModelExecution`.

Step (0): Bootstrap. We assumed that 5 oracles have been deployed and are available. They are implemented using *ChainLink* and *ChainLink Any API*, a popular solution in industry and academia [20]. All supporting contracts in Table I(b) are configured and deployed on the blockchain.

Step (1): Preparation. The CA defines contract `CertificationModel` as ⟨`confidentiality-in-transit`, `web server`, {`Observatory`, `Vuln-Scan-Host`, `Vuln-Scan-Web`, `HTTPS-Strength`, `HTTPS-Heartbleed`}, `policy`, `AND`, $\texttt{signature}_{\texttt{CA}}$⟩, where

- `confidentiality-in-transit` and `web server` represent the property `p` and the target `ToC`, respectively;
- `Observatory` is a probe evaluating the compliance to *Mozilla Observatory* best practices, which can impact confidentiality.[4] It returns ✓ when the latter are followed;

Listing 1. Evidence collected by `HTTPS-Strength`.

```
1  {
2   "status": 1,
3   "data": {
4    "Certificate not trusted by": [],
5    "Protocol setup": {
6     "Versions": {
7      "Supported versions": ["tlsv1_2", "tlsv1_3"]
8     }
9    }
10  }
11 }
```

- `Vuln-Scan-Host` and `Vuln-Scan-Web` are probes looking for vulnerabilities in the hosting server and the web technologies it uses, respectively. They return ✓ when vulnerabilities are not found;
- `HTTPS-Strength` and `HTTPS-Heartbleed` are probes inspecting the HTTPS setup looking for compliance to best practices and vulnerability Heartbleed, respectively. They return ✓ when best practices are followed and vulnerability Heartbleed is not found;
- `policy` is defined as $(\{o_1, \ldots, o_5\}, \texttt{Median})$, where all oracles are selected and fan-in performed using median;
- `AND` is the evaluation function `F` requiring all evidence to be successfully collected (i.e., all probes returning ✓).

The CA then deploys contract `CertificationModel` with corresponding `CertificationModelExecution`. Finally, the CA *funds* oracles computation, sending them the required amount of *tokens LINK*.[5] Figure 4 shows the pseudocode of contract `CertificationModelExecution`. *Function modifier* `onlyCSP` restricts function execution to the CSP only (lines 6, 9, 12 in Figure 4).

Step (2): Evidence Collection. The CSP invokes function `CertificationModelExecution.runCertModel` (lines 6-8 in Figure 4), executing probes through oracles. Collected evidence is retrieved back from oracles invoking `CertificationModelExecution.retrieveEvidence` (lines 9-11 in Figure 4). This function aggregates evidence collected by oracles for each probe following `CertificationModel.policy.Median`. Listing 1 shows the evidence retrieved by `HTTPS-Strength`. The Boolean result is ✓ (`1` in line 2 in Listing 1), while extradata indicate that the X.509 certificate is trusted by anyone and the two most recent protocol versions are supported (`TLSv1.2` and `TLSv1.3` in lines 3-10 in Listing 1).

Step (3): Evaluation and Award. Retrieved evidence is then evaluated invoking function `CertificationModelExecution.evaluateAndCreate` (lines 12-19 in Figure 4). It aggregates the result of individual probes using aggregation `AND` (Boolean AND) (line 13 in Figure 4) and, upon success, awards a certificate in the form of contract `Certificate` (lines 14-18 in Figure 4). According to our scenario, such certificate can be awarded. Listing 2 shows its JSON representation. Line 2 in Listing 2 shows the on-chain address of

[4] https://observatory.mozilla.org/

[5] The currency used in ChainLink.

the corresponding contract `CertificationModel`, while lines 3-9 contain the off-chain hash of collected evidence.

Listing 2. Contract `Certificate` represented in JSON format.

```
1  {
2   "cert_model_addr":
    ↪  "0x6a40960f63e60e9801236700b7397a69d851C71d",
3   "hashed_evidence": {
4    "Observatory": "dd17b3d1f793a9f422fdb1af2b76a5b376716
     ↪  950c3249f04efd321742d20e830",
5    "Vuln-Scan-Host": "87d559453fd84a932e6005ab7fc2f97064
     ↪  4e19ba17ef383d1bb920b94feeeb33",
6    "Vuln-Scan-Web": "b41849ac2566381d485787f4e9311eeea0b
     ↪  3374ef14b7ebc01b331c16ff22765",
7    "HTTPS-Strength": "8977b48717adbaa5309dbdf35d537f0693
     ↪  7496cb14578bcb40adb05467a8ead5",
8    "HTTPS-Heartbleed": "33ed8ff37dccc2ddf2c8ae7e0eb9699c
     ↪  5e786d51181d5f659343c2e7a539ea10"
9   }
10 }
```

VI. Experiments

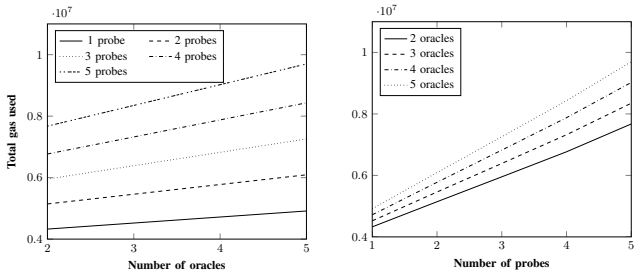
We evaluated the performance of the walkthrough in Section V (Section VI-A) and measured the scalability of our approach (Section VI-B). To prevent fluctuations caused by probes execution, we executed probes offline and collected the evidence via oracles online. We note that the lack of similar solutions (Section VIII) makes a performance and scalability comparison with the state of the art inapplicable.

A. Performance

We measured *gas* consumption and corresponding monetary value throughout Steps (0)–(3). Gas is an abstraction of the computational effort to execute a blockchain transaction (e.g., smart contract deployment, execution of a smart contract function) [22]. Gas consumption corresponds to a *fee* to be paid in the Ethereum currency denoted as *ETH*. Table II shows our results. "Gas limit" indicates the maximum amount of gas available for a transaction, which is estimated automatically [22], [23]. The fee is retrieved as *gas used*$\times$*cost of a unit of gas*, where the latter is ETH 45.85×10^{-9}. The value in € is retrieved as ETH 1=€3,147.28.[6] We assumed that ETH on the test network *Sepolia* had the same values of the main Ethereum production blockchain's for the sake of experiments, and retrieved gas quotation therefrom.

We can observe that the total cost is ≈€2,700, nearly half of which is necessary to bootstrap the blockchain (Step (0)). The cost of a certification is ≈€1,400 evenly split between the CA and the CSP. As expected, the most expensive action at certification time is the execution of the certification model (`CertificationModelExecution.runCertModel`).

In addition, interaction with oracles is funded through *tokens LINK*, a token standardized in ERC677 (Ethereum Request for Comments 677) [24]. Each execution of a probe on an oracle required LINK 0.1, while verifiable random generation LINK 0.32 (price fixed by the oracle provider). It corresponds to €2 and €6.3 (LINK 1=€19.68),[7] respectively, assuming the usage of a production blockchain.



(a) Varying the number of oracles (b) Varying the number of probes

Fig. 5. Scalability evaluation of our approach.

To sum up, any interactions with the blockchain require a fee. Step Bootstrap is expensive but executed only once. It is entirely funded by CAs due to their role. This cost can be recovered, for instance, by a shared fund between CAs and CSPs. At certification time, fees are split between CAs and CSPs. The former contribute to contract deployment (`CertificationModel`, `CertificationModelExecution`). The latter, that is, the actor that benefits the most from certification, contribute to the execution of the most expensive contract (`CertificationModelExecution`). In addition, the CA funds oracles interaction sending *tokens LINK*, meaning that accredited labs are remunerated. CAs expenses can be fully recovered by, in turn, charging the CSPs, making our approach fully sustainable. We note that future developments in blockchains may further reduce costs.

B. Scalability

We evaluated the scalability of the certification process (Steps (1)–(3)) in terms of gas consumption varying the number of probes in [1, 5] and oracles for each probe in [2, 5], the two parameters with the highest impact. The latter is >1 to achieve some fan-out (see Definition 4). Each probe implements an HTTP call.

Figure 5(a) shows that the number of oracles moderately impacts scalability, with gas consumption increasing from 7,674,006 to 9,697,726 in the worst case of 5 probes. Figure 5(b) shows that the number of probes has a larger impact on scalability, though it remains linear. Gas consumption increases from 4,914,802 to 9,697,726 in the worst case of 5 oracles. Overall, our experiments show that our approach is scalable, even without any optimizations, left for future work.

VII. Discussion

In this paper, we enhanced the traditional certification scheme and process, strengthening their guarantees (Section VII-A) though some issues remain open (Section VII-B).

[6]Gas quotation and conversion rate retrieved from https://etherscan.io/gastracker#historicaldata and https://www.coinbase.com/price/ethereum as of March 2nd, 2024, 11pm.

[7]Conversion rate retrieved from https://www.coinbase.com/price/chainlink as of March 2nd, 2024, 11pm.

TABLE II
GAS CONSUMPTION AND MONETARY CORRESPONDENCE.

Action	Gas used/Gas limit	Fee	
		ETH	€
Deploy `VRFv2SubscriptionManager`	792,127/792,127 (100.0%)	0.036	114.306
Send LINK to `VRFv2SubscriptionManager`	51,658/77,487 (66.67%)	0.002	7.454
`VRFv2SubscriptionManager.topUpSubscription()`	81,826/87,583 (93.43%)	0.004	11.808
Deploy `VRFv2Consumer`	1,295,442/1,295,442 (100.0%)	0.059	186.936
`VRFv2SubscriptionManager.addConsumer()`	102,950/104,075 (98.92%)	0.005	14.856
Deploy `Coordinator`	4,052,945/4,052,945 (100.0%)	0.186	584.851
Deploy `Probes`	2,555,101/2,555,101 (100.0%)	0.117	368.708
Total Step (0)	**8,932,049**	**0.41**	**1,288.920**
`Coordinator.createPolicy()`	534,994/534,994 (100.0%)	0.025	77.201
`VRFv2Consumer.requestRandomWords()`	164,754/164,754 (100.0%)	0.008	23.774
Deploy `CertificationModel`	2,862,577/2,862,577 (100.0%)	0.131	413.078
Deploy `CertificationModelExecution`	1,522,032/1,522,032 (100.0%)	0.07	219.633
Send LINK to `Probes`	51,658/77,487 (66.67%)	0.002	7.454
Total Step (1)	**5,136,015**	**0.235**	**741.141**
`CertificationModelExecution.runCertModel()`	3,672,865/3,875,370 (94.77%)	0.168	530.005
`CertificationModelExecution.retrieveEvidence()`	403,463/409,377 (98.56%)	0.018	58.221
Total Step (2)	**4,076,328**	**0.187**	**588.225**
`CertificationModelExecution.evaluateAndCreate()` and **total Step (3)**	**537,041**/537,041 (100.0%)	**0.025**	**77.497**
Total	**18,681,433**	**0.857**	**2,695.783**

A. Guarantees

Our approach addresses Assumptions A1–A5 as follows.

A1) Honest behavior of all actors. Our approach removes this unrealistic assumption. Malicious behavior of CSPs, CAs and (some) labs is prevented.

A2) Complete trust in the CA and accredited lab. Our approach partially relaxes this assumption. On the one hand, the CA is only trusted in terms of preparing a correct certification model. On the other hand, accredited labs are replaced by oracle providers, a minority of which can even be rogue. The policy in Definition 4 can be tuned according to the scenario (e.g., larger fan-out).

A3) Opaque certification process. Our approach removes this assumption, executing the entire process on-chain in a fully transparent manner.

A4) Undefined life cycle of certification artifacts. Our approach removes this assumption, providing an on-chain life cycle for certification artifacts, including evidence. Certification model and certificate are saved as immutable and traceable smart contracts. Evidence is saved following the typical pattern where the full content is saved off-chain and its hash on-chain within contract `Certificate`, preventing tampering.

A5) Chain of trust from the CA to the certificate. Our approach enhances the traditional chain of trust. Signatures are *automatically* added (and recorded) to *any* actions, meaning that the process that brought to the release of a certificate `Certificate` given a certification model `CertificationModel` is fully traceable. In traditional certification, traceability is only partial.

Following A1–A5, our approach prevents the most important threat against certification: the release of a certificate for a service that does not really support the given property. In traditional certification, the rogue CSP may be able to fool an individual probe. In our approach, the CSP needs to fool *all/most oracles executing the probes*, which is significantly more challenging due to the higher cardinality (A1, A3).

Next, let us assume that the rogue CSP colludes with other actors. In traditional certification, it can collude with the accredited lab, the latter inserting fake evidence in the certificate. In our approach, the lab corresponds to *one* oracle provider. The effect of this collusion is negligible due to the policy spreading probe execution across multiple oracles at random and aggregating results in a clever manner (Definition 4). For instance, aggregation `Mode` can nullify most collusion attempts, even across multiple oracle providers (A1, A2, A3). Advanced policies can further reduce collusion effects [25].

Next, let us assume that the rogue CSP wants to alter the evidence that drove the release of a certificate of a rival CSP. In traditional certification, the rogue CSP can collude with the accredited lab that released the certificate: such lab can alter the certificate with an updated signature. In our approach, the released certificate is stored as an immutable contract `Certificate`. It stores the hash of the collected evidence, making evidence tampering impossible (A1, A2, A4, A5).

Next, let us assume that the only rogue actor is the accredited lab. As already mentioned, in traditional certification, it can easily and unnoticeably release fake certificates without collecting evidence or inserting fake evidence in certificates. In our approach, the rogue lab has a negligible impact because the majority of oracles remain benign (A1, A2, A3, A5).

Finally, let us assume that the rogue CSP deploys a contract `Certificate` without collecting any evidence (i.e., an empty certificate, or a certificate pointing to other evidence). This danger is peculiar to our approach: in traditional certification, it requires collusion with an accredited lab. In our

approach, smart contracts *modifiers*[8] prevent many malicious flows. In addition, artifacts transparency ensures that such an attempt is immediately caught (A1, A3, A4, A5).

Overall, our blockchain-based certification departs from a scenario where trust is taken for granted to a scenario where trust is transparent, verifiable, and the danger of malicious behaviors is significantly reduced.

B. Limitations

To the best of our knowledge, the approach in this paper is the first approach defining and implementing a practical, complete solution of certification based on blockchain. However, it suffers from a number of limitations mainly motivated by technological immaturity as follows.

- **Role of the CAs.** We assume the CAs to be *functionally trusted*, that is, they define contracts `Certification-Model` that are correct for their purpose. No further assumptions are made on CAs behavior.
- **Oracles.** We assume the existence of a set of oracles which are fit for certification, worldwide-known and socially agreed. At present, oracles are the only solution to execute trustworthy actions from the blockchain towards the outer world and back. As such, we assume oracles to be correct and their majority benign and non-colluding.
- **Identities.** We assume the existence of a trusted CA managing identities (*CA Block* in Section IV), allowing actors to sign transactions and create a link between the two with limited involvement of such CA.

VIII. Related Work

Today, certification is the most widespread technique to increase system trustworthiness. It is applied along the entire (cloud) system stack, from applications (the focus of this paper) [6], [7], [13] to infrastructures [9], Big Data pipelines [26], IoT [27], [28], and even in risk management [29]. For instance, Faqeh et al. [30] proposed a certification process for evolving systems. When a new component is going to be deployed, the new and old versions are executed and evidence is collected. If evidence supports the *safety* of the new component, the latter replaces the old one. Bornholt et al. [31] defined a certification process based on lightweight formal methods. It focuses on *functional* property *correctness*, and collects evidence from the (annotated) target source code. Milánkovich et al. [6] proposed a certification scheme for cloud services. It certifies different non-functional properties in an *incremental* manner using AI, collecting evidence mainly from the target source code. Finally, Anisetti et al. [7], [17] designed a (continuous) certification scheme for cloud services that enlarges the typical scope of evidence collection beyond the target service software artifacts. Certificates are then the basis for certification-based decision-making (e.g., selection).

Research already focused on partially relaxing Assumptions A1–A5. For instance, Stephanow et al. [14] assumed that dishonest CSPs attempt to obtain certificates without being entitled to (A1). The proposed certification process addresses the issue using randomized test-based evidence. Prüfer [13] assumed that *many* actors are dishonest, from CSPs to CAs and accredited labs (A1, A2, A3). In this scenario, a *cloud association* enforces an *equilibrium* incentivizing actors in reporting misbehavior while punishing it. Lins et al. [12] mentioned the existence of issues such as data manipulation and opaque process in continuous cloud service certification (A1, A3, A4). Stephanow et al. [32] focused on securing certification schemes implementation, considering *traditional* attacks such as DoS and tampering (A1, A4).

Research also investigated blockchains to support certification. For instance, Neisse et al. [27] presented a blockchain-based solution where certificates referred to IoT devices are stored and exchanged. The process that brought to the release of such certificates remains off-chain. Similarly, Ardagna et al. [33] relied on the blockchain as trusted repository of certification artifacts. The certification process is executed off-chain and is the basis of a certified service composition. Turan et al. [34] proposed a blockchain-based alternative to the conventional Public Key Infrastructure. It includes mechanisms to incentive honest behavior and penalize misbehaving actors.

Certification schemes implementing specifically designed processes based on TPMs (Trusted Platform Modules) typically increase evidence trustworthiness and integrity, (partially) addressing Assumptions A4 and A5. For instance, Bertholon et al. [35] proposed a certification process in which users can certify, on demand, properties *integrity of software* and *data* of cloud nodes. Muñoz et al. [36] designed a certification scheme certifying both the software and the hardware layers of the cloud stack. It achieves the former using traditional certification, while the latter using TPMs, again focusing on property *integrity*. Nawab [37] focused on the cloud-edge scenario. The certification process collects evidence using a low-overhead cryptographic protocol to guarantee the correct behavior of (potentially dishonest) edge datastores. In the same context, Aslam et al. [8] proposed a TPM-based certification process targeting property *integrity*: certified edge nodes can then join the cloud federation. Khurshid et al. [28] proposed a similar solution in the context of IoT devices. The certification process is based on PKI (Public Key Infrastructure) and TPMs and focuses on properties *authentication* and *integrity*.

Finally, certification is addressing ML-based applications [11] where trust issues also emerge. For instance, Zhao et al. [10] designed a certification process focused on property *billing correctness* in MLaaS. It collects evidence using a custom, cryptographic-like protocol.

Table III summarizes related work with respect to Assumptions A1–A5; ✓ means that the assumption is fully relaxed, ≈ partially relaxed, ✗ not relaxed, - not possible to understand if relaxed. We note that Table III reports only works relaxing at least one assumption. It clearly emerges that most assumptions are not addressed, addressed partially in the best case. On the contrary, our approach (last row in Table III) completely addresses Assumptions A1, A3–A5, and partially A2.

[8] A modifier changes the behavior of a function, for instance, by allowing its execution if some preconditions are met (https://docs.soliditylang.org/en/v0.8.24/contracts.html#modifiers).

TABLE III
COMPARISON WITH RELATED WORK. ✓ MEANS RELAXED, ✗ NOT RELAXED, ≈ PARTIALLY RELAXED, - NOT CONSIDERED.

Ref.	A1	A2	A3	A4	A5
Bertholon et al. (2011) [35]	≈	✗	≈	≈	≈
Muñoz et al. (2013) [36]	✗	✗	≈	≈	≈
Stephanow et al. (2016) [14]	≈	✗	-	-	-
Lins et al. (2017) [12]	≈	✗	≈	≈	-
Stephanow et al. (2016) [32]	≈	✗	✗	≈	-
Prüfer (2018) [13]	✓	✓	≈	✗	✗
Neisse et al. (2019) [27]	✗	✗	✗	≈	✗
Ardagna et al. (2020) [33]	✗	✗	≈	≈	✗
Faqeh et al. (2020) [30]	✗	✗	-	✗	-
Aslam et al. (2020) [8]	≈	✗	✗	≈	≈
Nawab (2021) [37]	≈	✗	≈	≈	≈
Bornholt et al. (2021) [31]	✗	✗	≈	-	-
Zhao et al. (2021) [10]	≈	✗	≈	✗	≈
Khurshid et al. (2023) [28]	≈	✗	≈	≈	≈
Milánkovich et al. (2022) [6]	✗	✗	✗	≈	-
Our approach	✓	≈	✓	✓	✓

IX. CONCLUSIONS

The shift towards multi-cloud and multi-party services is affecting many established practices, including challenging the trust pillars of certification. The approach in this paper proposes a first solution to this issue, defining a blockchain-based certification scheme and implementing a certification process that is decentralized, open, and fully traceable. Our approach finally departs from taking trust for granted and rather builds trust using specific, on- and off-chain constructs.

The paper leaves space for future work. We plan to introduce novel policies with stronger guarantees on oracle selection while formally studying the degree of protection against collusion. We also plan to introduce reward-punishment mechanisms for bad-behaving actors. For instance, oracles whose collected evidence are similar to the majority of collected evidence, can be rewarded with funds and selected for the next iterations.

ACKNOWLEDGMENTS

The work was partially supported by *i)* project MUSA – Multilayered Urban Sustainability Action – project, funded by the European Union - NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and creation of R&D "innovation ecosystems", set up of "territorial leaders in R&D" (CUP G43C22001370007, Code ECS00000037); *ii)* project SERICS (PE00000014) under the NRRP MUR program funded by the EU – NextGenerationEU; *iii)* Università degli Studi di Milano via the program "piano sostegno alla ricerca" and "One Health Action Hub: University Task Force for the resilience of territorial ecosystems" – PSR 2021 – GSA – Linea 6; *iv)* SCUSI projet – Région Auvergne-Rhône-Alpes. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor the Italian MUR can be held responsible for them.

REFERENCES

[1] H. Taktak, K. Boukadi, M. Mrissa, C. Ghedira Guegan, and F. Gargouri, "A Model-Driven Approach for Semantic Data-as-a-Service Generation," in *Proc. of IEEE WETICE 2020*, Bayonne, France, Sep. 2020.

[2] F. Berto, C. A. Ardagna, M. Torrente, D. Manenti, E. Ferrari, A. Calcante, R. Oberti, C. Fra', and L. Ciani, "A 5G-IoT enabled Big Data infrastructure for data-driven agronomy," in *Proc. of IEEE GC-Wkshps 2022*, Rio de Janeiro, Brazil, Dec. 2022.

[3] D. A. Chekired and L. Khoukhi, "Smart Grid Solution for Charging and Discharging Services Based on Cloud Computing Scheduling," *IEEE TII*, vol. 13, no. 6, 2017.

[4] H.-W. Deng, M. Rahman, M. Chowdhury, M. S. Salek, and M. Shue, "Commercial Cloud Computing for Connected Vehicle Applications in Transportation Cyberphysical Systems: A Case Study," *IEEE ITSM*, vol. 13, no. 1, 2021.

[5] C. A. Ardagna and N. Bena, "Non-Functional Certification of Modern Distributed Systems: A Research Manifesto," in *Proc. of IEEE SSE 2023*, Chicago, IL, USA, Jul. 2023.

[6] Á. Milánkovich, G. Eberhardt, and D. Lukács, "The AssureMOSS Security Certification Scheme," in *Proc. of ARES 2022*, Vienna, Austria, Aug. 2022.

[7] M. Anisetti, C. A. Ardagna, and N. Bena, "Multi-Dimensional Certification of Modern Distributed Systems," *IEEE TSC*, vol. 16, no. 3, 2023.

[8] M. Aslam, B. Mohsin, A. Nasir, and S. Raza, "FoNAC - An automated Fog Node Audit and Certification scheme," *COSE*, vol. 93, 2020.

[9] M. Anisetti, C. A. Ardagna, F. Berto, and E. Damiani, "A Security Certification Scheme for Information-Centric Networks," *IEEE TNSM*, vol. 19, no. 3, 2022.

[10] L. Zhao, Q. Wang, C. Wang, Q. Li, C. Shen, and B. Feng, "VeriML: Enabling Integrity Assurances and Fair Payments for Machine Learning as a Service," *IEEE TPDS*, vol. 32, no. 10, 2021.

[11] M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani, "Rethinking Certification for Trustworthy Machine-Learning-Based Applications," *IEEE IC*, vol. 27, no. 6, 2023.

[12] S. Lins, P. Grochol, S. Schneider, and A. Sunyaev, "Dynamic Certification of Cloud Services: Trust, but Verify!" *IEEE S&P*, vol. 14, no. 2, 2016.

[13] J. Prüfer, "Trusting privacy in the cloud," *IEP*, vol. 45, 2018.

[14] P. Stephanow, G. Srivastava, and J. Schütte, "Test-Based Cloud Service Certification of Opportunistic Providers," in *Proc. of IEEE CLOUD 2016*, San Francisco, CA, USA, Jun.–Jul. 2016.

[15] H. Teigeler, S. Lins, and A. Sunyaev, "Drivers vs. Inhibitors - What Clinches Continuous Service Certification Adoption by Cloud Service Providers?" in *Proc. of HICSS 2018*, Waikoloa, HI, USA, Jan. 2018.

[16] J. Lansing, A. Benlian, and A. Sunyaev, ""Unblackboxing" Decision Makers' Interpretations of IS Certifications in the Context of Cloud Service Certifications," *JAIS*, vol. 19, 2018.

[17] M. Anisetti, C. A. Ardagna, and N. Bena, "Continuous Certification of Non-Functional Properties Across System Changes," in *Proc. of ICSOC 2023*, Rome, Italy, Nov.–Dec. 2023.

[18] J. Alonso, L. Orue-Echevarria, V. Casola, A. I. Torre, M. Huarte, E. Osaba, and J. L. Lobo, "Understanding the challenges and novel architectural models of multi-cloud native applications – a systematic literature review," *JoCCASA*, vol. 12, no. 1, 2023.

[19] A. B. Pedersen, M. Risius, and R. Beck, "A Ten-Step Decision Path to Determine When to Use Blockchain Technologies," *MISQE*, vol. 18, no. 2, 2019.

[20] S. K. Ezzat, Y. N. Saleh, and A. A. Abdel-Hamid, "Blockchain oracles: State-of-the-art and research directions," *IEEE Access*, vol. 10, 2022.

[21] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee, "Ethereum smart contract analysis tools: A systematic review," *IEEE Access*, vol. 10, 2022.

[22] Ethereum. Gas and fees. [Online]. Available: https://ethereum.org/developers/docs/gas

[23] MetaMask. User guide: Gas. [Online]. Available: https://support.metamask.io/hc/en-us/articles/4404600179227-User-Guide-Gas

[24] Chainlink. Link token contracts. [Online]. Available: https://docs.chain.link/resources/link-token-contracts

[25] M. Bettinger, L. Barbero, and O. Hasan, "Collusion-Resistant Worker Set Selection for Transparent and Verifiable Voting," *SN Computer Science*, vol. 3, no. 5, 2022.

[26] C. A. Ardagna, N. Bena, C. Hebert, M. Krotsiani, C. Kloukinas, and G. Spanoudakis, "Big Data Assurance: An Approach Based on Service-Level Agreements," *Big Data*, vol. 11, 2023.

[27] R. Neisse, J. L. Hernández-Ramos, S. N. Matheu, G. Baldini, and A. Skarmeta, "Toward a Blockchain-based Platform to Manage Cybersecurity Certification of IoT devices," in *Proc. of IEEE CSCN 2019*, Granada, Spain, Oct. 2019.

[28] A. Khurshid and S. Raza, "AutoCert: Automated TOCTOU-secure digital certification for IoT with combined authentication and assurance," *COSE*, vol. 124, 2023.

[29] M. Anisetti, C. A. Ardagna, N. Bena, and A. Foppiani, "An Assurance-Based Risk Management Framework for Distributed Systems," in *Proc. of IEEE ICWS 2021*, Chicago, IL, USA, Sep. 2021.

[30] R. Faqeh, C. Fetzer, H. Hermanns, J. Hoffmann, M. Klauck, M. A. Köhl, M. Steinmetz, and C. Weidenbach, "Towards Dynamic Dependable Systems Through Evidence-Based Continuous Certification," in *Proc. of ISoLA 2020*, Rhodes, Greece, Oct. 2020.

[31] J. Bornholt, R. Joshi, V. Astrauskas, B. Cully, B. Kragl, S. Markle, K. Sauri, D. Schleit, G. Slatton, S. Tasiran, J. Van Geffen, and A. Warfield, "Using Lightweight Formal Methods to Validate a Key-Value Storage Node in Amazon S3," in *Proc. of ACM SOSP 2021*, Virtual, Oct. 2021.

[32] P. Stephanow, C. Banse, and J. Schütte, "Generating Threat Profiles for Cloud Service Certification Systems," in *Proc. of IEEE JASE 2016*, Orlando, FL, USA, Jan. 2016.

[33] C. A. Ardagna, M. Anisetti, B. Carminati, E. Damiani, E. Ferrari, and C. Rondanini, "A Blockchain-based Trustworthy Certification Process for Composite Services," in *Proc. of IEEE SCC 2020*, Beijing, China, Nov. 2020.

[34] E. Turan, S. Sen, and T. Ergun, "A Semi-Decentralized PKI Based on Blockchain With a Stake-Based Reward-Punishment Mechanism," *IEEE Access*, vol. 12, 2024.

[35] B. Bertholon, S. Varrette, and P. Bouvry, "CERTICLOUD: A Novel TPM-based Approach to Ensure Cloud IaaS Security," in *Proc. of IEEE CLOUD 2011*, Washington, DC, USA, Jul. 2011.

[36] A. Muñoz and A. Maña, "Bridging the GAP between Software Certification and Trusted Computing for Securing Cloud Computing," in *Proc. of IEEE SEPAC 2013*, Santa Clara, CA, USA, Jun.–Jul. 2013.

[37] F. Nawab, "WedgeChain: A Trusted Edge-Cloud Store With Asynchronous (Lazy) Trust," in *Proc. of IEEE ICDE 2021*, Chania, Greece, Apr. 2021.