Received July 21, 2021, accepted September 17, 2021, date of publication September 23, 2021, date of current version October 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3115343

Efficient Certification of Endpoint Control on Blockchain

DIEGO PENNINO[1], MAURIZIO PIZZONIA[1], ANDREA VITALETTI[2], AND MARCO ZECCHINI[2]

[1]Dipartimento di Ingegneria, Sezione di Informatica e Automazione, Università degli Studi Roma Tre, 00146 Rome, Italy
[2]Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Università di Roma "La Sapienza," 00185 Rome, Italy

Corresponding author: Diego Pennino (pennino@ing.uniroma3.it)

This work was supported in part by POR FESR LAZIO 2014–2020, call for "Gruppi di ricerca 2020" Det. n. G04052 of April 4, 2019, under the "LazioChain" Project (CUP F85F21001550009—POR project code A0375E0116) and in part by the Sapienza Project "La disintermediazione della Pubblica Amministrazione: il ruolo della tecnologia blockchain e le sue implicazioni nei processi e nei ruoli della PA."

ABSTRACT Proving that an *endpoint* (e.g. URL, telephone number, etc.) is controlled by a *subject* is crucial in many applications. In the web, this is witnessed by the widespread adoption of HTTPS. In centralized architectures, this task is usually carried out by trusted *certification authorities* (CAs). In decentralized applications, for example based on blockchains, or for self-sovereign identity management (SSI), it would be desirable to perform these checks in a decentralized way, relying on the collective behavior of a society of individuals rather than on a single trusted entity. In any case, the result should be a widely usable certificate, as in the centralized CA case. In this paper, we show two blockchain-based methods to prove the association between a subject and an endpoint in a decentralized manner. Our methods are compatible with a wide variety of endpoints and contribute to fill the gap of the current SSI approaches with respect to decentralization. We analyze the security of our proposal and provide a proof-of-concept implementation. We also evaluate performances, costs, and compatibility with current standardization efforts about SSI.

INDEX TERMS Endpoint, digital certificate, decentralized certification authority, blockchain, identity management, self-sovereign identity, decentralized blockchain oracle.

I. INTRODUCTION

A *subject*, for example a person, can be contacted on a number of *channels*, like telephone, email, postal mail, etc. We call *endpoint* the one end of a communication channel that have to be known by the other party to initiate a communication with that subject (or with a service or device acting on behalf of the subject). Examples of endpoints include phone numbers, e-mail addresses, IP addresses, web URLs, postal mail addresses, etc.

We say that a subject *controls*, an endpoint E if it can send data from E and/or receive data at E. Proving the control of an endpoint is crucial in modern internet. This is the fundamental property that is guaranteed by digital certificates employed in HTTPS and certified e-mail, just to mention two prominent examples. However, digital certificates rely on centralised architectures and entail the trust into a *Certification Authority* (*CA*) that checks that the subject indeed controls that endpoint and unequivocally proves this fact signing a *certificate*. Whoever trusts the CA and sees the certificate can easily believe that the subject controls the endpoint.

The associate editor coordinating the review of this manuscript and approving it for publication was Rakesh Matam.

While this solution is absolutely consistent with the dominant centralized approach (e.g. for cloud services), it is not adequate for decentralized permissionless blockchains. Indeed, in these P2P architectures peers are mutually untrusted and it is not desirable to introduce a trusted entity, since this would reduce the benefit of adopting decentralization in the first place. The first challenge addressed by our paper is the following: *Can we design a fully decentralised method to certify the endpoint control on blockchain?* A formal statement is provided in Section III.

In our vision, the ability of proving the control of an endpoint in the blockchain is crucial, because it is a natural place where the controller of an endpoint can publicly specify its willingness of being contacted on that endpoint and for what purpose. A prominent example for this use case is a Robinson list [24], where users can specify whether they accept to be contacted on their mobile phone (i.e. an endpoint) for advertising purposes. We consider it in our experimental evaluation.

Regarding how to tackle our problem, we can easily envision a simple and fully decentralized technique to prove the control of endpoints. For simplicity, we focus on proving the control of a mobile phone number, but the same approach can

be easily generalized to other contexts. The entity needing the proof sends a suitable code by SMS to the subject, which provides it back over the Web (i.e. a different channel), thus proving its ability to receive that code at its phone number. However, that technique requires that each entity must independently perform the verification process and this might be unpractical, costly, slow, or unreliable, depending on the specific kind of endpoint and on the application context (see also Sections II, III, and X). Hence, we also consider the following challenge: *Can we design a decentralized endpoint certification method that is comparable in terms of efficiency and costs with state-of-the-art centralized approaches?*

Contribution of the Paper: In this paper, we show how to perform *an efficient decentralized check that a subject controls an endpoint* and how to *publicly certifying it, without relying on a central authority*. Further, our approach is compatible with many kinds of endpoints, which makes it applicable in many different application contexts.

We assume each subject to be uniquely associated with its public/private key pair. Our result can be viewed as a decentralized and automated certification authority for certifying on a public/permissionless blockchain that the subject associated with a public key controls an endpoint. Our main idea is to randomly select a limited number of other subjects, that already participate in the blockchain, to form a *committee*. Each *committee member* uses the endpoint to interact (automatically, for most kinds of endpoints) with the subject and check its endpoint control, also exploiting cryptographic challenges. If a sufficiently high number of committee members can autonomously and independently check the endpoint control, a certificate is stored in the blockchain, immediately proving the control to all the other users.

To meet our challenges, we have to design protocols to (1) allow a subject to request an endpoint control certification, (2) select a committee, (3) allow committee members to assess that the subject requesting certification actually controls the endpoint, (4) allow committee members to express the result of their assessment, and (5) turns the assessment results into a unique certification. Further, for these protocols, we have to prove (a) security against an attacker that can collude with a bounded number of potential committee members, (b) ease of realization using current blockchain technology, (c) efficiency (cost and time) comparable with current centralized approach.

While, in principle, our approach might be realized without any blockchain, this would be unpractical since we leverage a number of blockchain-related features: reliable multicast, history records, cryptographic integrity proofs, smart contracts, etc.

Our results are very well suited to be used in conjunction with blockchain-based applications, however, they also complement current *self-sovereign identity management* (SSI). ISO/IEC standards [6] defines an identity as a "set of attributes related to an entity", as a consequence, a controlled endpoint can be a component of the identity of a subject. SSI manages the identities in blockchain, but attribute certifications, still relies on *issuers* that should be trusted by *relying parties*. Our contribution enables SSI systems to avoid relying on trusted centralized issuers for the certification of control for several kinds of endpoints.

Our proposal is somehow similar to a decentralized oracle, with some notable differences with respect to other works [15], [16], [27], [30]. Actually, endpoint control is very specific and we can take advantage of this specificity. The relation between an endpoint with a subject is usually quite stable over time and the endpoint can be involved in a cryptographic protocol. For this reasons, our committee members just act as independent executors of a cryptographic certification protocol. The protocol is designed to effectively prove the control of an endpoint. The only possibility to prove a fake control requires the subject to collude with a fair number of committee members. Since the committee is randomly selected, this event can be made highly unlikely.

We developed a proof-of-concept which not simply shows the feasibility of the proposed approach, but also allow us to prove that it is *efficient*, because both the incurred costs and verification times are comparable with those of centralized certification authorities. We also address specificities of several kinds of endpoints, and conformity to W3C SSI standards.

Main Contributions: The main contributions of this paper are the following.

- We present two blockchain-based methods to certify endpoint control in a decentralized manner.
- We formally prove the security of our approaches against miscertification and denial of service attacks.
- We evaluate the practicality of our approach on the basis of an Ethereum-based proof-of-concept realization.

Structure of the Paper: The structure of the paper is the following. In Section II we discuss the spectrum of application of endpoint control certification with respect to several kinds of endpoints. In Section III, we introduce notation and formalize our problem and naive decentralized verification protocols. In Section IV, we states the assumptions of our work. In Section V, we formally describe our two certification methods. In Section VI, we describe two approaches to securely select the committee members and we discuss how to motivate their participation. In Section VII, we define our threat model and, we formally analyze the security of our methods. In Section VIII, we discuss revocation and other aspects related to certificate management. In Section IX, we evaluate our approach from several points of view and show a proof-of-concept realization. In Section X, we discuss technical aspects related to the applicability of our methods for several kinds of endpoints and discuss endpoint-specific security issues. In Section XI, we review the related work about blockchain oracles and SSI and compare our approach against it.

II. EXAMPLES OF APPLICATION CONTEXTS

The results shown in this paper can be applied in many application contexts for different kinds of endpoints and for

TABLE 1. Notable examples of endpoints. In the selected sample applications, endpoint certification can provide substantial added value.

Endpoint	Sample Application(s)	Sample Channels
Phone number	Consent to receive marketing calls or messages	SMS, phone voice call
Postal mail address	Receive (sensible) documents or goods	Postal service companies
Email address	Receive (sensible) documents/information	Email service
IP address	Vulnerability assessment service, authentication	TCP or UDP communication
URL	Vulnerability assessment service, authentication	HTTP or HTTPS communication
Domain name	Domain name marketing service	DNS resolution protocol
Bank account details / IBAN	Fiat money transfers	Bank transfer, direct debit

different purposes. In this section, we introduce examples to show this diversity.

Example of reasons for adopting decentralized solutions (e.g., blockchain-based ones) are avoiding intermediaries to reduce costs, improve transparency, avoiding drift toward market monopoly or oligopoly, avoid interference from improper behavior of central authorities (e.g., censorship), etc. In decentralized applications, endpoints are used to provide services, information, goods or money to users, which usually implies some form of legal liability. Generally, a malicious third party M may want to trick a subject A to use the endpoint of a subject B with the intent to cause some damage or legal problems to A or B. This makes endpoint certification a relevant problem for many decentralized applications.

We give a few examples.

- Suppose to have a decentralized system to buy and sell remote web security assessment. Let A be a cybersecurity firm. Subject M might pretend to control a web service of B and ask A to perform a vulnerability assessment on that service, and obtain its results. To avoid this, A asks B to prove the control of its web service before performing the assessment. This can take time. Can B obtain a certificate to be shown to any cybersecurity firm in a decentralized manner?
- Suppose B is the owner of a sensible website, A is any website visitor, and M may want to impersonate B to steal the password of A. B is afraid that regular CAs may revoke certificates at any time under government pressure. Can B obtain a certification of owning the website to show to any visitor without relying on a centralized CA?
- Suppose to a have a decentralized daily fresh milk market. If A is a supplier that asks for weekly payments using bank direct debit, a malicious subject M might order milk delivery at address of B and ask for debiting current account of a third subject C. To avoid this, A needs certification that the address and the current account actually belong to the person who ordered the milk (C). Direct certification by A can be done but it is boring, costly and slow. This reduces the freedom of buyers to switch different milk suppliers each week. Can certifications be obtained in a decentralized manner once and for all?

In general, we expect that for many decentralized applications relations between subjects need to be rapidly created. In this setting, autonomous verification may be too costly and too slow, to be useful.

Table 1 lists several examples of endpoints that can be certified with the methods described in this paper, where our approach can provide a substantial value added. In the sample application(s) column, we give some examples of sensible services for each of the listed endpoints. The *sample channels* column puts in evidence the fact that the same endpoint can be used in several different ways. For example, a telephone number can be used with voice calls or with SMS, an IP address can be used with a number of different protocols, an URL can be used with HTTP or HTTPS protocols, etc. Clearly, when an endpoint is certified making use of one channel, this certification can be used also with other channels.

For our methods to work, the endpoint must allow the user to exchange a small cyptographically signed string. A channel should not necessarily be designed to exchange general data for this. For example, for bank money transfers, we can exploit their *purpose* annotation and, for domain names, we can exploit the TXT DNS record. It may be worth to note that, when the endpoint is especially designed for communication, certifying the association of a public key with the endpoint provides a mean to authenticate the other party, hence, providing a functionality that is similar to that provided by common website certificates used in HTTPS communications.

More technical aspects related to the application of our certification methods to the endpoints listed in Table 1 are discussed at the end of the paper, in Section X.

III. NOTATION, BASIC PROTOCOLS, AND PROBLEM STATEMENT

In our setting, each subject S is associated with at least one pair $\langle p, s\rangle$, where p is a public key and s is the corresponding secret key. We denote by $[m]_s$ or $[m]_S$ the signature of a string m performed by S using s. Subject S can easily prove its association with p by considering a publicly known random string r (a *challenge*), never used before, whose generation is not controlled by S, and providing $[r]_s$. In interactive protocols with two parties, r can be generated by the party that needs the proof. In the blockchain context, r can be a cryptographic hash of some *new*, unpredictable, piece of data. For example, r can be the cryptographic hash of a suitably chosen block.

We now formally introduce our verification problem.

Definition 1 (Endpoint-Control Verification Problem): In the endpoint-control verification problem, a subject V, called *verifier*, intends to assess that a subject S, called *prover*, controls an endpoint E.

The vast majority of techniques to solve the endpoint-control verification problem, in a *decentralized* manner, can be traced back to the following two protocols, that assume that V and S can securely communicate through a connection c (that does not involve E).

Protocol 1:

1) V chooses a random string Q and sends it to endpoint E.
2) S **receives** Q at endpoint E and sends back Q to V by c.

Protocol 2:

1) V chooses a random string Q and sends it to S by c.
2) S **sends** back Q to V from endpoint E.

These protocols prove that who can communicate through c also controls E, under the hypothesis that no attacker can impersonate S or eavesdrop messages sent to S on c and E.

These protocols can be adapted to bind the fact that S controls E with a key pair $\langle p, s\rangle$ of S. In the adapted protocols, S sends back to V a signed version of Q ($[Q]_s$), actually proving that the subject that knows s also controls E. In this variation, Q is essentially a challenge. In the rest of this paper, we refer to the adapted versions of Protocols 1 and 2 as *basic protocols*.

The basic protocols have the following drawbacks that are especially relevant in decentralized applications in which verifiers might be many and may need to perform endpoint control verification very often.

Specificity. The verification performed by each verifier is *specific* to that verifier and cannot be leveraged by other verifiers, since there is no trust among verfiers in a decentralized setting. This means that each time a new verifier V intends to assess if a subject S controls a certain endpoint E, a new verification procedure should be performed by V and S. This might be a serious problem for S and/or V if verifications are many and have a non-negligible cost for S and/or V.

Interactivity. Each verification requires to set up an *interactive* protocol. This means that either S should always be ready for the verification or V should be willing to wait for S to complete the verification. Even if S is ready for the verification, communication on E can be slow or even unreliable. This might be incompatible with requirements of specific applications.

As mentioned in the introduction, the endpoint-control verification problem can be solved in a centralized manner by resorting to a certification authority. Centralized certifications have the notable advantage that are non-specific and non-interactive, but require trust in a central authority, which is not desirable for blockchain-based applications, decentralized peer-to-peer applications, and self-sovereign identity management. In this paper, we aim at solving the following problem

Definition 2 (Decentralized Endpoint-Control Certification Problem): In the decentralized endpoint-control certification problem, a prover S intends to obtain a *certificate*

- stating that S controls endpoint E and
- verifiable by any verifier without interacting with S and without trusting any specific subject.

IV. MAIN CONCEPTS AND ASSUMPTIONS

In this section, we describe the high-level architecture of our system and the roles of the actors. We also clearly states our assumptions about subjects capabilities and their behavior, endpoint technology, and blockchain technology. While, some of these assumptions are discussed and formally described further in the paper (see Sections V, VI and VII), this first informal description clarifies the context in which the results are shown.

A. ARCHITECTURE AND ROLES

In the rest of the paper, we refer to the high-level architecture shown in Figure 1.

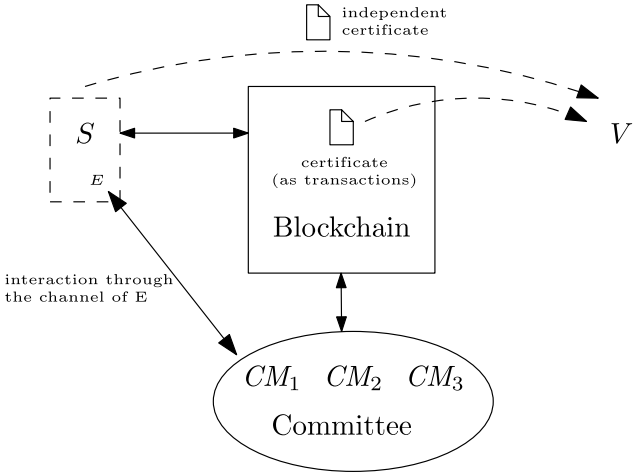

FIGURE 1. High-level reference architecture. Notation is as follows: S is a prover, V is a verifier, each CM_i is a committee member, E is an endpoint. For simplicity, only one prover, one verifier, one endpoint, and one committee are shown. Certificates can be expressed as blockchain transactions or as independent documents (see Section V).

Our results assume the presence of a multiplicity of *subjects* and of a *blockchain network* whose infrastructure is made of a number of *nodes*. A subject might operate a node but in general this is not true and we do not assume it. We assume devices owned by subjects and nodes of the blockchain to be interconnected with standard internet telecommunication networks. The only interaction among subjects outside the blockchain is through the endpoint E (and its associated channel).

Each subject can have one or more of the following three roles.

Prover. A *prover*, denoted S in the figure, intends to obtain a *certification* that it controls an endpoint E to be shown to verifiers.

Verifier. A *verifier* (or *relaying party*), denoted V, is interested in ascertain if S controls E. To do that it can obtain (and check) the certificate of endpoint control from the blockchain (for certificates expressed as transactions, see Sections V-A and V-B) or from S (for *independent certificates*, see Section V-C).

Committee member. A *committee member* contributes to the process of creating the endpoint-control certificate that was requested by S.

How each subject performs each of these roles is detailed in Section V.

B. ENDPOINTS

An endpoint is one end of a communication channel. Some notable families of endpoints are described in Section II. In Section V, we only require the channel of an endpoint E to be usable to send or receive some general short data (our protocols need only to send the signature of a challenge). Since E may not support bidirectional communication, we define two certification methods. In each method, E is used in only one direction. Other aspects related to endpoints are discussed in Section X.

For simplicity, the reader can assume the endpoint channel to be reliable. Committee members with an unreliable channel are easily dealt with by considering them as non-cooperating (see also Section VII).

Concerning security, our protocols do not require confidentiality of communication through the blockchain. On the contrary, our protocols relies on the fact that it is hard for an attacker to send messages from E (spoofing) or receiving (eavesdropping) a message at E, in place of the subject that controls E. A formal security analysis of the protocols is provided in Section VII, while a discussion of security concerns regarding some families of endpoints is provided in Section X.

C. BLOCKCHAIN

In our protocols, we use a blockchain as a reliable bus in which subjects can broadcast messages as *transactions*. From the adoption of a blockchain, we obtain several advantages beyond reliability. A blockchain stores all transactions, with their ordering, in a tamper-proof way and natively provides cryptographic proofs of the existence of those transactions. We use these proofs in Section V-C to create certificates that are easy to verify without knowing the blockchain state. It also provides smart contracts that we use to shorten certificates. Finally, it provides an easy mean to create a shared and unpredictable seed for pseudo-random number generation (see Section VI).

Each transaction that is accepted by the blockchain network ends up to be recorded in a block of the blockchain. Each block is identified with its *block number*. Each accepted transaction has an *identifier* which uniquely identifies it. The history of the accepted transactions is supposed to be totally ordered and immutable. We treat the blockchain as a completely independent system and we assume its reliability, security, and liveness.

Each subject is identified in the blockchain by its public key with which anyone can check the correctness of transactions stored in the blockchain or submitted for acceptance. In principle, for a single subject, it is possible to have two distinct public keys: one to be associated with the endpoint and one that serves as identifier in the blockchain. For simplicity, we assume that one single public key serves both roles. This means that with the subject private key it should be possible to sign both transactions and generic data.

For the methods described in Sections V-A and V-B, it is not strictly needed for the blockchain to support smart contracts. However, for the developments described in Sections V-C, and VI-B, we assume the blockchain technology supports smart contracts with the capability to change a state persisted in the blockchain itself. In Section IX-C, we describe a proof-of-concept based on the Ethereum [20] technology, which supports this kind of smart contracts.

To participate in our protocols, each subject should be able to send transactions to the blockchain network and to observe new transactions accepted by the blockchain network. To do that, a subject not necessarily must operate a blockchain node. In fact, a subject can just have a *light client* that leverages a connection to an untrusted full blockchain node to send transactions, get new blocks, and get a past transaction starting from its identifier, with guarantee of integrity. The light client does not need to store the whole new blocks, but only their headers. Light clients technology is well established and described, for example, in [19] and [20].

It is useful to point out that our approach is independent from the kind of consensus algorithm the blockchain is based on and it is also independent from the choice of permissioned or unpermissioned technologies, provided that data can be publicly accessible by light clients.

Finally, we point out that decentralized certification is naturally needed for decentralized applications (see Section II) and in conjunction with self-sovereign identity management systems. In both cases, blockchain-based architectures are common and the need for a blockchain does not imply additional architectural complexity and cost, in those cases.

V. TWO BLOCKCHAIN-BASED METHODS TO CERTIFY ENDPOINT CONTROL

In this section, we describe two methods called `prover-receives` and `prover-sends`, to solve the decentralized endpoint-control certification problem. Each method comprises a blockchain-based *certification protocol* (or *certificate creation protocol*), to compute a certificate in a decentralized manner, and its corresponding *verification algorithm*, to be run by the verifier to check the certificate. Methods presented in this section, produce certificates that are in the form of one or more transactions in the blockchain. In Section V-C, we show how to derive a certificate that is a single document that depends only from the hash of one of

the blocks and do not need any access to blockchain content to be verified, but just to block headers.

In the `prover-receives` method, the certification is a proof that a prover S can *receive* at endpoint E a challenge from a number of other subjects. Vice-versa, in the `prover-sends` method, the certification is a proof that S can *send* from E a signed challenge to a number of other subjects.

By writing the pair $\langle S, E\rangle$, we intend the statement "S controls E", which may or may not be true, or may be proved by a certificate. If p is the public key of S, we can also write $\langle p, E\rangle$ with the same meaning. When S obtains the certification for $\langle S, E\rangle$, we say that E *is certified* (*for* S) and also that S *is certified* (*for* E).

The methods we show exploit the collaboration of a number of subjects. We say that a subject is *on-line* if it can

1) send transactions to the blockchain,
2) monitor the transactions accepted in the blockchain and react to them,
3) access the transactions that are contained in past blocks of the blockchain, and
4) receive messages at E or send messages from E, where E is an endpoint controlled by S.

Our blockchain-based certification protocols involves a *committee* C of (approximately) k subjects, called *committee members*, that are asked to interact with S through the endpoint that S is supposed to control. Committee C is pseudo-randomly selected, for each *certification request*, according to one of the procedures described in Section VI. The selection procedures have to respect a number of requirements that we discuss in the following. For now, we just state that the committee selection is implicit, in the sense that the committee depends on a random seed that is not under the control of any subject (in Section VI, we adopt the hash of a yet-to-create block), but any subject can check if a certain other subject is part of the committee, for a given certification request. Since it is unlikely that all members of a randomly selected committee are on-line and willing to cooperate, we suppose that only $\bar{k}$ members of the committee, out of (approximately) k, are on-line and cooperating, with $1 \leq \bar{k} \leq k$. Situations in which a large fraction of potential committee members are non-cooperating can be handled by rising k. In general, we envision application of our methods when the number of potential committee members N is large and k can be easily risen. Parameters k and $\bar{k}$ are independent from the certification request frequency, since each request is independently dealt with and generally has a distinct committee. These parameters are also independent from the number of nodes participating in the underlying blockchain. Finally, k and $\bar{k}$ affect security, but when N increases they do not need to be increased. Actually, in Section VII, we show the contrary, i.e., the security of our approach increases (and tends to be ideal) for large N, whatever the values of k and $\bar{k}$ are.

To simplify the description of our certification protocols, we assume that no previous certification involves S and E (we discuss these cases in Section VIII). Now, we introduce our two methods.

A. CERTIFYING THAT A SUBJECT CAN RECEIVE A MESSAGE AT AN ENDPOINT

In the `prover-receives` method, S proves its control of endpoint E by showing that it was able to receive messages at E from the members of a randomly selected committee.

This idea is formalized in the protocol described in Algorithm 1 and in the corresponding interaction diagram depicted in Figure 2. At the end of the protocol, the certificate is published by S in the blockchain in the form of a *certificate transaction* P that contains the messages received by S. Algorithm 2, shows the details of the checks that a verifier should carry out to accept the certificate.

The certification protocol is started by S, publishing on the blockchain a transaction containing the *certification request* for $\langle S, E\rangle$. We denote by t_R the (accepted) transaction containing it and by R its identifier. We denote by C the randomly selected committee for R. The committee selection adopted in this protocol is probabilistic and described in Section VI-A. When the block containing R is published, each subject c, that is on-line, recognizes the new certification request t_R and autonomously check if it is part of the committee C for that request. If this is the case, c generates a new random bit string Q_c and cryptographically links it to R by signing $\text{hash}(Q_c|R)$. The result is the *challenge* Π_c. Then, c sends Π_c to S through E. When S has received at least $\bar{k}$ correct challenges, it publishes in the blockchain the *certificate transaction* P containing all received challenges, with the indication of the corresponding committee member.

A verifier can check P by executing the verification procedure formalized in Algorithm 2, which essentially checks signatures and correctness of all the challenges.

About correctness, we expect that, if a prover S actually controls an endpoint E, the certificate creation protocol for $\langle S, E\rangle$ successfully terminates producing the certificate P and P is successfully verified by the verification procedure. The latter statement is trivially true, since verification just checks the result of each creation step. The critical aspect about the first statement is that enough members (at least $\bar{k}$) of the selected committee should conclude the protocol. The fraction of committee members that concludes the protocol depends on how many potential committee members are on-line, reachable, and willing to cooperate, and clearly depends on the application context. However, the probability to have at least $\bar{k}$ members concluding the protocol can be risen by properly tuning the selection process (i.e., rising parameter k in Section VI-A).

About security, we refer the reader to Theorems 2 and 3, and to their proofs.

B. CERTIFYING THAT A SUBJECT CAN SEND A MESSAGE FROM AN ENDPOINT

In our second method, called `prover-sends`, prover S proves its control of endpoint E by showing that it was able to

Algorithm 1 Method `Prover-Receives`: Protocol for Certificate Creation

Require: Prover S, with public key p, is not certified for any endpoint. Endpoint E is not certified for any subject.

Ensure: The blockchain contains a certificate transaction P containing all challenges received from committee members, proving that S controls E, in the sense that S **can receive** messages at E.

1: S creates a *certification request* for the pair $\langle p, E\rangle$ and publishes it as transaction t_R in the blockchain, with identifier R.
2: Let b be the hash of the block containing transaction t_R.
▷ C is a committee randomly selected (on the basis of R and b) according to the probabilistic method described in Section VI-A.
3: $G \leftarrow \emptyset$ ▷ G is the set of received "good" challenges.
4: **concurrently** each subject c **upon** observing t_R **do**
5: c checks if it belongs to C according to the probabilistic method described in Section VI-A.
6: **if** $c \in C$ **then**
7: $Q_c \leftarrow$ a random bit string privately generated by c
8: $\Pi_c \leftarrow Q_c|[\text{hash}(Q_c|R)]_c$
9: c sends Π_c to E
10: **end if**
11: **end concurrently**
12: **concurrently** S **upon** receiving $\Pi = Q_c|[\text{hash}(Q_c|R)]_c$ **do**
13: If $|G| < \bar{k}$, continue below, otherwise do nothing.
14: S checks the correctness of the signature in Π.
15: S checks if $c \in C$ according to the probabilistic method described in Section VI-A.
16: If the above checks are successful, $G \leftarrow G \cup (\Pi, p_c)$
17: If $|G| = \bar{k}$, S publishes *certificate transaction* $P = [G, R]_S$ on the blockchain.
18: **end concurrently**

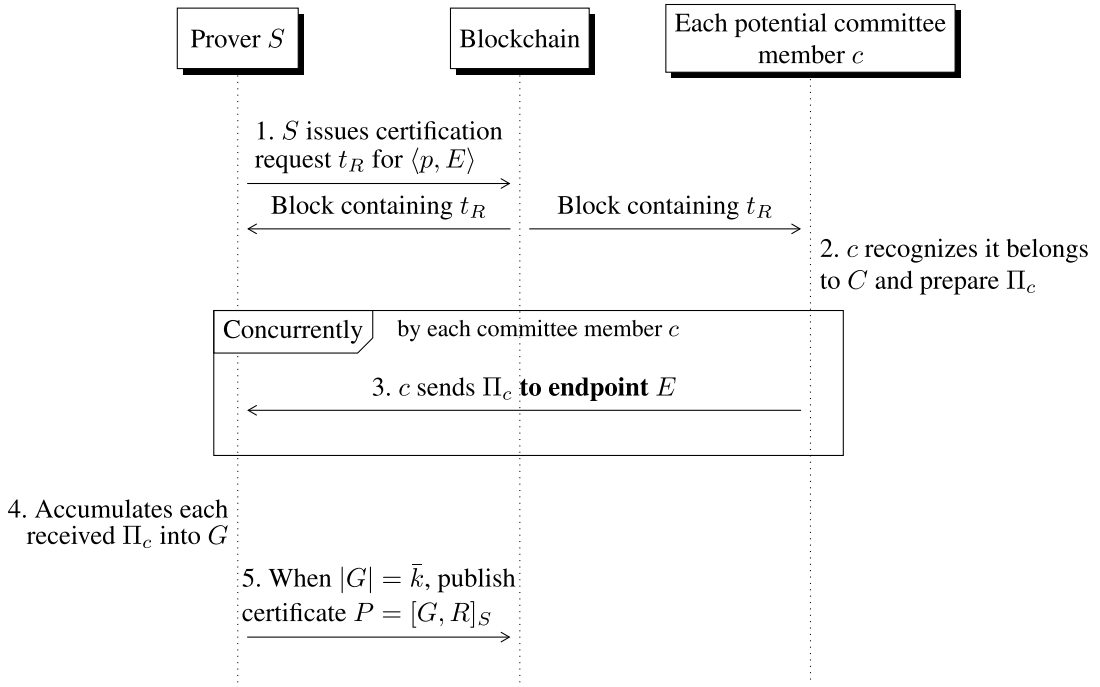

FIGURE 2. Sequence diagram for the certificate creation protocol of the `prover-receives` method shown in Algorithm 1.

send messages from E to the members of a pseudo-randomly selected committee.

This idea is formalized in the protocol described in Algorithm 3 and in the corresponding interaction diagram depicted in Figure 3. At the end of the protocol, the certificate is published in the blockchain in the form of $\bar{k}$ *acceptance transactions*. Each of them is denoted A_c and is sent by member c of the committee, stating that c has received the

Algorithm 2 Method `Prover-Receives`: Algorithm for Certificate Verification

Require: The identifier R of a certification request. A certificate P published in the blockchain.
Ensure: **Success**, if S proved that it can receive data at E, **fail** otherwise.

If P occurs before R in the blockchain history, **return fail**
Let $\langle p, E\rangle$ be the request identified by R, where p is the public key of the prover S.
If P is not in the form $[G, R]_S$, **return fail**
If P is not correctly signed by S, **return fail**
If $|G| < \bar{k}$, **return fail**.
for each $(\Pi, p_c) \in G$ **do**
    If Π does not match the form $Q_c|[\text{hash}(Q_c|R)]_c$, **return fail**
    If c is not part of the committee according to the probabilistic approach described in Section VI-A, **return fail**
    If Π is not correctly signed by c, **return fail**
end for
return success

Algorithm 3 Method `Prover-Sends`: Protocol for Certificate Creation

Require: Prover S, with public key p, is not certified for any endpoint. Endpoint E is not certified for any subject.
Ensure: The blockchain contains one transaction A_c for each committee member c that has checked that S controls E, in the sense that S **can send** messages from E.

S creates a *certification request* for the pair $\langle p, E\rangle$ and publishes it as a transaction t_R in the blockchain, with identifier R.
Let b be the hash of the block containing transaction t_R.
Let C be the committee whose members are randomly selected (on the basis of R and b) according to the deterministic approach described in Section VI-B.
for each $c \in C$ **do**
    Let p_c be the public key of c, and E_c an endpoint certified for c.
    $Q_c \leftarrow \text{hash}(b|p_c)$
    S sends $\Pi_c = [Q_c, R]_S$ from E to c at E_c.
end for
concurrently each committee member c **upon** receiving $\Pi_c = [Q_c, R]_S$ at E_c **do**
    It gets the transaction t_R by means of its identifier R.
    It gets p from t_R and checks the signature of Π_c.
    It gets E form t_R and checks that Π_c was sent from E.
    It gets the hash b of the block containing t_R and checks that $Q_c = \text{hash}(b|p_c)$.
    If the above checks are successful, it publishes on the blockchain its *acceptance transaction* $A_c = [\Pi_c, p_c]_c$.
end concurrently

message from S through E. Algorithm 4, shows the details of the checks to be carried out to accept a certificate.

As in the `prover-receives` method, the certification protocol is started by S, publishing on the blockchain a certification request t_R, for $\langle S, E\rangle$, identified by R. Again, a pseudo-random committee C is selected for t_R. The committee selection adopted in this protocol is deterministic and it is described in Section VI-B. Note that, Algorithm 3 needs to enumerate all committee members and, for each member c, it needs to know its public key p_c and a certified endpoint E_c. The committee selection approach described in Section VI-B has these features.

Each potential committee member c can continuously listen on E_c or start to listen only when a transaction t_R (identified by R) is published, for which c is a committee member. For each committee member c, prover S is supposed to send from E a distinct message Π_c to c (at E_c). Each Π_c is so that it is provably created by S and related to R. For further security, it also contains a publicly known random string Q_c, that it is not under the control of S and it is different for each c. The corresponding acceptance transaction A_c is simply the publication of Π_c, signed by c, which proves that c received Π_c.

A verifier can execute Algorithm 4 to check that a set of acceptance transactions are indeed a valid certificate for $\langle S, E\rangle$. That algorithm simply checks signatures and format of the messages.

About correctness, the considerations at the end of Section V-A, for the `prover-receives` method, also applies to the `prover-sends` method. In particular, it is possible to increase the probability that we got at least $\bar{k}$ acceptance transactions by rising the parameter k in the committee selection process described in Section VI-B.

About security, we refer the reader to Theorems 4 and 5, and to their proofs.

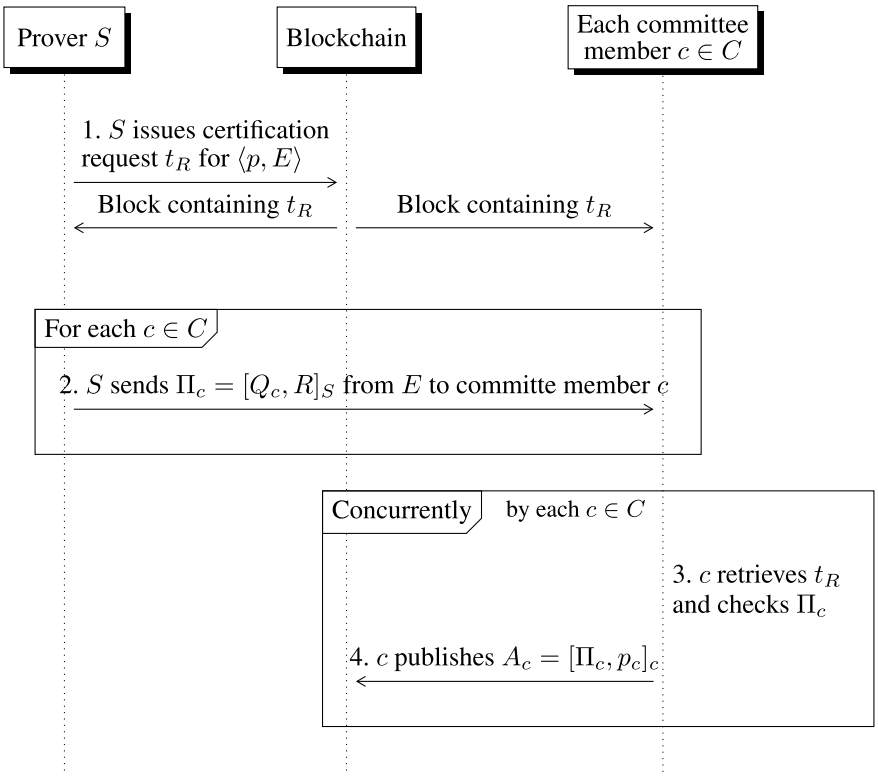

FIGURE 3. Sequence diagram for the certificate creation protocol of the `prover-sends` method shown in Algorithm 3.

Algorithm 4 Method `Prover-Sends`: Algorithm for Certificate Verification

Require: The identifier R of a certification request. A set $\mathcal{A}$ of identifiers of acceptance transactions.
Ensure: **Success**, if S proved that it can send data from E, **fail** otherwise.

If $|\mathcal{A}| < \bar{k}$, **return fail**.
Check that R is a certification request transaction.
Let b the hash of the block containing t_R and S the subject that sent t_R.
for each $A \in \mathcal{A}$ **do**
    If A occurs before R in the blockchain history, **return fail**
    Let c be the subject that published A and p_c its public key.
    If A is not correctly signed by c, **return fail**.
    If c is not part of the committee according to the deterministic approach described in Section VI-B, **return fail**.
    Let (Π, p_c) be the signed message of A.
    $Q_c \leftarrow \text{hash}(b|p_c)$
    If the signed message in Π is not equal to (Q_c, R), **return fail**.
    If Π is not correctly signed by S, **return fail**.
end for
return success

C. EXPRESSING CERTIFICATES AS INDEPENDENTLY VERIFIABLE DOCUMENTS

The methods proposed in Sections V-A and V-B solve the decentralized end-point control certification problem without being interactive and specific (see Section III). However, they publish their results in the blockchain. While this is theoretically sound, it may be unhandy to use in practice, since each certificate verification requires a lookup in the state of the blockchain.

We now show how it is possible to build an *independent certificate*, which is a document that can be used as a certificate that only depends on the headers of the blocks for its verification. This allows the verifier to complete the verification without interacting with any blockchain node, supposing that block headers are known. This is desirable since distribution of block headers is a well-supported feature in currently available blockchains, their size is small, and can be easily kept by a verifier (as is for common blockchain

light clients). To explain how we can build an independent certificate, we recall some design concepts about blockchains.

A common blockchain design is to organize transactions into a Merkle tree [39] (but in principle other *authenticated data structures* can be adopted as well [38], [44]). This data structure is a balanced tree that stores at each leaf the cryptographic hash of the content of the leaf, and at each internal node the hash of its children. This ends up having the root storing, essentially, a hash of all leafs, which is called *root-hash*. This is the only thing stored in a block header about the transactions of the block. A cryptographic proof that a transaction t is in a block b can be given by considering the path from t to the root of the Merkle tree for b and providing the sibling of all nodes in the path. This proof can be checked against the root-hash in the header of b, efficiently and with no further access to the network.

An independent certificate contains, for each transaction that is needed to certify the control of the endpoint, a triple $\langle t, i, p\rangle$, where t is the transaction, i is the block number of the transaction, and p is the Merkle proof of t with respect to the root-hash contained in the i-th block.

For the `prover-receives` method, only one transaction of size $O(\bar{k})$ is needed. For the `prover-sends` method, $O(\bar{k})$ transactions of constant size are needed. Depending on the actual magnitude of $\bar{k}$, the cost for transmitting, storing and verifying independent certificates can be too high. We now introduce a *succint* form of independent certificates.

We introduce a smart contract called *summarizer*. We assume this smart contract has a state (as, for example, in Ethereum). The idea is that the certificate transaction (for `prover-receives`) or the acceptance transactions (for `prover-sends`) should be directed to the summarizer, which checks them and, if they are recognized as a valid certificate, can set a variable of its state to assert that the certification process is valid. This variable can simply store a Boolean value or an integer expressing the number of committee members that was positively involved in the certification. This state can be considered trusted by verifiers since blockchain itself is considered trusted in our context. For blockchains that explicitly represent state, this is represented in each block by a distinct Merkle tree whose root-hash is stored in the header, hence, the same techniques that we suggested for transactions can be used for the state. A *succinct independent certificate* contains a triple $\langle s, i, p\rangle$, where s is the piece of the state that expresses the validity of the certificate and p is the Merkle proof of s with respect to the root-hash contained in the i-th block.

Independent certificates (either succinct or not) can be provided by the prover to a verifier when needed, like traditional certificates are provided by a server to its clients in the TLS [10] protocol. They can also be stored in a public repository (e.g., a blockchain or a self sovereign identity management system) so that any subject can play the verifier role when it needs to assess the control of an endpoint by another subject, without interacting with it by any means.

VI. COMMITTEE SELECTION PROCEDURES AND MOTIVATING THE PARTICIPATION

In this section, we describe the procedures to select the committee that are needed by certification protocols shown as Algorithms 1 and 3. To properly serve its purpose, a committee selection procedure should fulfill the following requirements.

Verifiability. Given a certification request, anyone should be able to determine whether a certain subject is member of the committee related to that request.

Unpredictability. Suppose the committee can be predicted on the basis of only the certification request, before publishing it on the blockchain. In this case, a malicious prover could try many request (off-line, without submitting them) until if finds a request with a favorable committee with many members under its control, making easy a miscertification attack (see Section VII). An unpredictable procedure entails that the committee is known only once the request is accepted by the blockchain. In this case, a brute force attack would be extremely slow and publicly logged.

Uniformity. The probability for a subject to be selected as a committee member should be uniform across all potential committee members. Otherwise, an attacker could try to control the subjects that have more probability to be part of a committee invalidating the proofs of Lemmas 1 and 2.

Robustness. The desired committee size k should be a parameter given to the procedure. The actual size k' of the selected committee should be ideally equal to k. If this is the case, we say that the selection procedure is *deterministically robust*. However, there are cases in which this is hard to achieve. A selection procedure is *probabilistically robust* if $k' \in [k, \alpha k]$ with high probability, for increasing α and $\alpha > 1$. Having $k' > k$ does not affect security (see Section VII), but a large k' may affect efficiency. Tuning α can help us to strike a good trade-off.

Efficiency. The prover should ideally execute its part of the procedure in time $O(k)$ and the committee members in time $O(1)$. However, if we take advantage of blockchain facilities (e.g., smart contracts) to store data, we admit a further logarithmic factor in the size of the blockchain state.

First, we note that, to achieve unpredictability, we need a source of randomness to seed our selection procedure. We use as seed $s = \text{hash}(b|R)$, where b is the hash of the block where the certification request is recorded and R is the identifier of the certification request itself. If we base our pseudo-random generation of the committee on s, we meet the unpredictability requirement, since the prover cannot predict or control b.

We devised two distinct approaches, each one suitable for each of the two methods described in Section V. They are supposed to be applied by the prover or by other

subjects right after the certification request is accepted in the blockchain, i.e., when seed s is available. Section VI-A describes a probabilistically robust approach that is simple to implement completely off-chain, but it does not support an efficient enumeration of the members of the committee (it just support membership check). This approach cannot be applied in Algorithm 3, which needs enumeration (see line 4). Section VI-B, describes a deterministically robust approach that supports both efficient enumeration and membership check. However, it requires a support from the blockchain (that can be realized as a smart contract).

When a pull-style communication is adopted to communicate through the endpoint, the preferred committee selection approach is switched. See Section X for further details.

A. PROBABILISTIC APPROACH

The probabilistic approach is suited to be used in the `prover-receives` method and provides probabilistic robustness, i.e., if k is the desired committee size, the actual committee size k' is in $[k, \alpha k]$. Parameters k and α (with $\alpha > 1$) are supposed to be constant and provided as input to the procedure.

In this approach, for each subject c with public key p_c, we derive from s a random value $s_c = \text{hash}(s|p_c) \in [0, M-1]$. Subject c is selected to be a committee member if $s_c < \frac{(1+\alpha)kM}{2N}$, where N is the number of potential committee members. Requirements verifiability, unpredictability, uniformity and efficiency are trivially met. Concerning robustness, we state the following lemma.

Theorem 1: The probabilistic approach for selecting the committee is probabilistically robust, if the number of potential committee members is large.

Proof: We have to show that, for increasing α, $\Pr\left[k' \in [k, \alpha k]\right]$ tends to 1. For the probabilistic approach, each subject is selected to be part of the committee with probability $p = \frac{(1+\alpha)k}{2N}$. The committee size k' is a random variable with binomial distribution with probability p for N samples, hence, its expected value is $\mu = Np = k(1+\alpha)/2$ (middle point of $[k, \alpha k]$) and its variance is $\sigma^2 = Np(1-p)$, which both depend on α. For large N, that binomial distribution can be approximated by a normal distribution with expected value μ and variance σ^2. We denote by $\Phi(\cdot)$ the cumulative distribution function of a standard normal distribution. Hence $P[k' < k] \simeq \Phi(\frac{k-\mu(\alpha)}{\sigma(\alpha)})$. It turns out that, by increasing α, $\frac{k-\mu(\alpha)}{\sigma(\alpha)}$ is always decreasing. Hence, since $\Phi(\cdot)$ is monotonic increasing, $P[k' < k]$ decreases when α increases. The symmetry of the probability density around μ allows us to extend this result also to $P[k' > \alpha k]$, which proves the statement. ∎

B. DETERMINISTIC APPROACH

The deterministic approach is suitable for use in the `prover-sends` method where the prover has to know in advance all the committee members to contact. We introduce in the blockchain a *collector* smart contract to which each potential committee member should voluntarily subscribe to participate in the selection. The collector keeps in its state an (arbitrarily ordered) list of pairs (p_S, E_S), where S is a subscribed subjects, E_S is its certified endpoint and p_S is its public key. We denote by $\bar{N}$ the size of this list. Each subscription should be performed by means of a constant-cost blockchain transaction.

To select k committee members, we pseudo-randomly select k positions $q_1, \ldots, q_k$ (each from 0 to $\bar{N}-1$) according to the following rule

$$q_i = \text{hash}(i|s) \mod \bar{N} \tag{1}$$

where s is the seed for the certification request, as defined before. If $\bar{N} \gg k$, the probability of accidentally doubly selecting the same subject twice is negligible. The prover needs to access the pairs at positions $q_1, \ldots, q_k$. This can be performed securely off-chain by resorting to Merkle proofs, that can be verified by solely trusting on block headers, with techniques similar to those presented in Section V-C. Any untrusted blockchain node can reply with the selected pairs, equipping each of them with a cryptographic proof that it is at its position in the list.

The verifiability, unpredictability, uniformity and robustness requirements are trivially met. Concerning efficiency, note that, Equation 1 should be computed k times. The length of the Merkle proofs is $O(\log z)$, where z is the size of the blockchain state. Committee membership can be verified in constant time after having verified Merkle proof in $O(\log z)$ time. The total processing cost for the prover is $O(k \log z)$.

C. MOTIVATING THE PARTICIPATION

Independently from the procedure adopted to select the members of the committee, our approach is effective if at least a fraction of the selected members are willing to participate in the protocol. While there could be intrinsic motivations, that depends on the specific application contexts, these may be too dull to be effective. Actually, the prover is the only subject that is really interested in certifying the control of its endpoints. Hence, a general solution is to put on the prover any effort for motivating the committee. In particular, it should sustain all the necessary costs, paying the members of the committee for their work in participating to the certification procedure. When covering the expenses is not sufficient to motivate the participation, additional incentives might help. The proof-of-concept implementation shown in Section IX-C, also considers how to integrate incentives in our approach. However, the design of an incentive (or a penalty for non cooperating members) is tightly dependent on the specific application context and is beyond the scope of this paper. The new-born discipline of *tokenomics* deals with this kind of design, often involving game theoretic considerations (see [37] for an introduction about tokenomics).

VII. SECURITY

In this section, we introduce our threat model and analyze the security of `prover-receives` and `prover-sends`

methods in that model. For simplicity, we assume that the control of subjects over their endpoints does not change over time. See Section VIII for a discussion of this aspect.

A. THREAT MODEL

In analyzing the security of our methods, we consider the following threats (or malicious objectives).

Miscertification. Certification of untrue control of subject S on endpoint E.

Denial of service (DoS). Denial of certification or certificate verification for a legitimate pair $\langle S, E\rangle$.

We denote by N the number of potential committee members in the network. We say that a committee member is *inconclusive*, if it is not on-line, it does not participate in the protocol or if its behavior is perceivably broken (e.g., producing malformed transactions or wrong signatures), otherwise we say that it is *conclusive*. A conclusive member can be *honest* or *colluding* (with the prover S). A colluding member correctly executes the certification protocol to certify $\langle S, E\rangle$, but receives/sends messages for S, that are supposed to go through E, through a different channel. One honest member would be enough to create a certificate, however, it is not possible to know if a specific subject is honest. In Section VII-B, we prove that one honest member is present in a committee with high probability when the number of potential committee members is large.

In our model, any *adversary-controlled* (certified or non-certified) subject may deviate from the protocol in an arbitrary way. For example, by being inconclusive or by misbehave maliciously. A notable case is the *Sybil attack*, which in our context consists in an adversarial prover attempting to miscertify an endpoint (not under its control) by creating a number of fictitious (colluding) subjects, called *sybil-subjects*. Sybil-subjects may be used by the adversary as potential committee members. We assume that the adversary can control only a limited number m of sybil-subjects, with m independent from N. The rationale behind this assumption is the possibility to defend against Sybil attacks imposing a cost for each subject [58]. This cost is supposed to be affordable for a regular user that is represented in the system as one subject but it should be paid by the adversary for each sybil-subject it creates. If the number of sybil-subjects needed by the adversary to perform the attack is large, the total cost of the attack might become unbearable for the adversary or, at least, dominate the advantage of performing the attack. This cost can be enforced in practice in several ways. For example, consider the `prover-sends` method, in which potential committee members have to be certified. If the blockchain supports a cryptocurrency, a fee can be periodically charged to keep a certificate active.

We assume the underlying blockchain technology to be secure, in the sense that an adversary

1) cannot tamper with the transactions in the blockchain history or with blockchain state,
2) cannot stop a non-controlled subject from asking to the blockchain to perform a transaction, and
3) cannot stop a non-controlled subject from reading from the blockchain the committed transactions.

This also implies that the adversary cannot add, delete or tamper with past certifications recorded in the blockchain.

To focus on the security aspects of our protocol, we assume the channel of E to be immune from certain specific attacks. (1) For both protocols, we assume no network-level denial of service is possible. (2) For the `prover-sends` method, we assume that the adversary cannot create spoofed messages that look as if they are sent from E. (3) For the `prover-receives` method, we assume the adversary cannot eavesdrop messages sent to E. We discuss security of specific kinds of endpoints in Section X. Finally, we assume to adopt ideal cryptographic hash functions and cryptographic signatures that cannot be attacked by the adversary.

B. SECURITY ANALYSIS

To prove the security of our methods against miscertification and DoS attacks, we first prove lemmas about the possibility to arbitrarily reduce the probability of obtaining a "bad" committee. Since we have introduced two ways to randomly select a committee, we have two of these lemmas. The bad members can be intended to be colluding or inconclusive members, depending on how these lemmas are used. We use the following notation. We consider a universe U of potential committee members containing $B \subset U$ *bad* potential members. We also denote $N = |U|$ and $m = |B|$. The resulting randomly selected committee is denoted by C and $\tilde{B} \subseteq C \cap B$ is the set of selected bad members.

The following lemma applies to the probabilistic random selection of a committee introduced in Section VI-A and is used in the security proofs for the `prover-receives` method.

Lemma 1: Consider a committee C in which each member is selected from U, independently with probability $p = a/N$, where $N = |U|$, a is constant, and $a < N$. The probability that $|\tilde{B}| \geq \bar{k}$, with $\bar{k} \geq 1$, tends to zero as $N \to \infty$.

Proof: Consider $B = \{b_1, \ldots, b_m\}$ and random variables $Y_1, \ldots, Y_m$ each associated to a corresponding bad committee member, where Y_i assumes values 1 if b_i is selected and 0 otherwise. Consider a random variable $Y = \sum_i Y_i = |\tilde{B}|$. The expected value of Y is $\mu = mp = ma/N$. Since Y is a sum of independent variables, the Chernoff bound can be applied: for any $\delta > 0$, $\Pr[Y > (1+\delta)\mu] < (\frac{e^{\delta}}{(1+\delta)^{1+\delta}})^{\mu}$. To obtain $\Pr\left[Y > \bar{k} - 1\right]$, we equate $(\bar{k} - 1) = (1+\delta)ma/N$, hence, $\delta = \frac{(\bar{k}-1)N}{ma} - 1$. Substituting δ in the bound we obtain the following.

$$\Pr\left[Y > \bar{k} - 1\right] < \left(\frac{e^{\frac{(\bar{k}-1)N}{ma} - 1}}{\left(\frac{(\bar{k}-1)N}{ma}\right)^{\frac{(\bar{k}-1)N}{ma}}} \right)^{\frac{ma}{N}} \tag{2}$$

It is easy to prove, by distributing the exponent ma/N into the inner factors, that the right hand side of Equation 2 tends to zero for $N \to \infty$. Hence, the statement is proved. ∎

The following theorem asserts the security of the `prover-receives` method against the miscertification attack, where the attacker is the prover.

Theorem 2 (`Prover-Receives` Miscertification Security): Let S be a prover and E an endpoint. If certificate $\langle S, E\rangle$ is created and verified according to the `prover-receives` method, then S controls E with an arbitrarily high probability, that can be increased by increasing the number of potential committee members participating in the network.

Proof: Input to Algorithm 2 is $P = [G, R]_S$, where $G = \{(\Pi_1, p_{c_1}), \ldots, (\Pi_q, p_{c_q})\}$, where $\Pi_i = Q_{c_i}|[\text{hash}(Q_{c_i}|R)]_{c_i}$. We focus on a single $\Pi = \Pi_i$ related to $c = c_i$. Algorithm 2 checks correctness of the signature of Π and that c belongs to the committee. If c is honest, it has sent Π to E and has not sent it to S by other means. Further, Q_c was randomly generated locally by c and since in Π the signature $[\text{hash}(Q_c)|R]_c$ involves R, Π was generated by c for certification request R, which makes impossible for S to reuse it from a certificate request procedure performed in the past. According to our threat model, it is not possible for the adversary not to control E and read a message sent to E. Hence, if the verifier were sure that c is honest, Π alone would suffice as a certification that S controls E. However, this is not the case. For this reason, Algorithm 2 checks that $q \geq \bar{k}$. The statement follows form the application of Lemma 1, assuming B to be the set of potential committee members colluding with the attacker. ∎

The following theorem asserts the security of the `prover-receives` method against the DoS attack, where the attacker intends to block a certification of a distinct prover.

Theorem 3 (`Prover-Receives` DoS Security): In the `prover-receives` methods, the probability that an adversary can successfully perform a denial of service can be made arbitrarily low by increasing the number of potential committee members participating in the network.

Proof: A DoS attack can be carried out blocking a certification or blocking the verification of a certificate. Note that a verifier performs certificate verification by running Algorithm 2. This algorithm does not interact with any subject but only with the blockchain, which is considered secure in our threat model (see Section VII-A). Hence, an attacker cannot deny the verification of a certificate. For this reason, we focus on the denial of the certification procedure. According to our threat model, the adversary cannot perform a network-level DoS or a blockchain-targeted DoS. Hence, the only way for the adversary to perform a DoS attack is to induce the selection of a committee of all inconclusive members. The statement follows form the application of Lemma 1, assuming B to be the set of inconclusive potential committee members controlled by the attacker. ∎

The following lemma applies to the deterministic random selection of a committee introduced in Section VI-B and is used in the security proofs for the `prover-sends` method.

Lemma 2: Consider a committee C, of k members, randomly selected from U. The probability that $|\tilde{B}| \geq \bar{k}$, with $1 \leq \bar{k} \leq k \leq m$, tends to zero as $N \to \infty$.

Proof: The probability of the statement can be expressed as follows.

$$\Pr\left[|\tilde{B}| \geq \bar{k}\right] = \frac{\sum_{i=\bar{k}}^{k} \binom{m}{i}\binom{\text{N-m}}{k-i}}{\binom{N}{k}} \tag{3}$$

Distributing the denominator, each term can be expressed in the following form.

$$\frac{\binom{m}{i}\frac{(N-m)\ldots(N-m-k+i+1)}{(k-i)!}}{\frac{N\ldots(N-k+1)}{k!}} \tag{4}$$

For each term, the number of factor depending on N that are at the numerator is $k - i$, while at the denominator their number is k. Hence, each term tends to zero as $N \to \infty$, which proves the statement. ∎

The following theorem asserts the security of the `prover-sends` method against the miscertification attack, where the attacker is the prover.

Theorem 4 (`Prover-Sends` Miscertification Security): Let S be a prover and E an endpoint. If certificate $\langle S, E\rangle$ is created and verified according to the `prover-sends` method, then S controls E with arbitrarily high probability that can be increased by increasing the number of potential committee members participating in the network.

Proof: According to Algorithm 4, the verifier was able to gather $|\mathcal{A}|$ acceptance transactions that are related to a certificate request transaction t_R for $\langle S, E\rangle$, with $\bar{k} \leq |\mathcal{A}| \leq k$. Let us consider one of them, say $A_c = [\Pi_c, p_c]_c$ published by committee member c, whose semantic is that c accepted challenge Π_c, thus supporting $\langle S, E\rangle$. Algorithm 4 checks that c is entitled to accept Π_c being part of the committee, and that the challenge is correct and originated from S. If c is honest, according to Algorithm 3, it publishes A_c only if it recognizes that Π_c was sent by S from E (Line 12 of Algorithm 3). According to our threat model, it is not possible for the adversary not to control E and cheat c by making Π_c to appear as if it were sent from E. If the verifier were sure that c is honest, A_c alone would suffice as a certification that S controls E. However, since this is not the case, the verifier checks that at least $\bar{k}$ acceptance transactions support $\langle S, E\rangle$. The statement derives from Lemma 2 assuming B to be the set of potential committee members colluding with the attacker. ∎

The following theorem asserts the security of the `prover-sends` method against the DoS attack.

Theorem 5 (`prover-Sends` DoS Security): In the `prover-sends` methods, the probability that an adversary can successfully perform a denial of service can be made arbitrarily low by increasing the number of potential committee members participating in the network.

Proof: As in the proof of Theorem 3, verification cannot be a target of DoS attack, as well as, the network and the blockchain (by threat model). Hence, the attacker should

force a committee of all inconclusive members. The statement follows form the application of Lemma 2, assuming B to be the set of inconclusive potential committee members controlled by the attacker. ■

VIII. RE-CERTIFICATION, DE-CERTIFICATION, AND MULTIPLE CERTIFICATIONS

Regular CAs allows customers to request the revocation of a certificate to face situations in which a private key is compromised. CAs check that the request comes from a subject entitled to do so. If that check is successful, the CA makes revocation public and the verifiers are supposed to actively connect to CA facilities to check for revoked certificates.

In our endpoint certification approach, we have to deal with both *key compromise* and *endpoint transfer*. In the following, we analyze and provide solutions form both cases.

Let S^E be a subject that holds the endpoint E and S^s be the subject that holds a secret key s related to the public key p. An endpoint certificate $\langle p, E\rangle$ states that $S^E = S^s$. The relevant cases, in which this is no longer true, are the following.

Key compromise. Secret key s was published or stolen and hence there exists (potentially) a subject $\bar{S}^s$ ($\neq S^s = S^E$) that knows s. In this case, S^s can be easily impersonated by $\bar{S}^s$. The impersonator can make assertions pretending to be the owner of E. This case is analogous to the one that usually leads to revocation for centralized CAs. This situation is critical in application contexts where E may not be general purpose, not always available, or not comfortable to use. Indeed, these applications may comprehend protocols asserting something about S^E without directly communicating through E.

Endpoint transfer. E was transferred to a different subject $\bar{S}^E$ ($\neq S^E = S^s$, e.g., reassignment of telephone number, selling of domain name, etc.). This case is specific for endpoint certification. In this situation, both $\bar{S}^E$ and S^s can potentially behave maliciously. Subject S^s can still pretend to be the owner of E, creating a situation similar to that described for the key compromise case. On the other side, $\bar{S}^E$ contacted through E can pretend to be the same subject S^s and possibly interact with other subjects through E impersonating $S^s = S^E$.

We now introduce two new primitives that allow us realizing revocation in all cases: de-certification and re-certification.

A *de-certification* transaction is published by S^s on blockchain on block B_i and it contains [decertify, $\langle p, E\rangle$, hash$(B_{i-1})]_s$, where hash(B_{i-1}) can be substituted by any random challenge not under the control of any single subject. The semantic of de-certification is that verifiers should no longer successfully verify certificate $\langle p, E\rangle$. Clearly, the verification procedures (Algorithms 2 and 4) should be enriched with the check that no de-certification was issued up to the instant in which verification is performed (further details about this are provided in the following).

De-certification can be used to address the key compromise problem. We note that, in the key compromise case, de-certification can also be issued by $\bar{S}^s$, but this ends up in blocking any further malicious use of s by $\bar{S}^s$. De-certification can also partially address the endpoint transfer problem, namely, when $\bar{S}^E$ behaves maliciously. To avoid this possibility, S^s should explicitly *de-certify* E right after the transfer of E to $\bar{S}^E$. De-certification is quick since it does not involve any committee and any communication through E.

Re-certification is an extended form of certification and is accomplished by extending the use of the very same protocols presented for certification (Algorithms 1 and 3) in the following way.

- For re-certification of E, we allow E to be already certified, i.e., we ditch the second pre-condition in the require clause of Algorithms 1 and 3.
- Suppose E is already certified by $\langle p, E\rangle$. After a re-certification, a new certificate $\langle p', E\rangle$ is created and verifiers should no longer successfully verify certificate $\langle p, E\rangle$, but should successfully verify the new certificate $\langle p', E\rangle$.

Re-certification can be used to address the endpoint transfer problem for the case not covered by de-certification. Suppose the control of E is legitimately transferred from S^E (owning key pair (s, p)) to $\bar{S}^E$ (owning key pair $(\bar{s}, \bar{p})$). By exploiting re-certification, $\bar{S}^E$ can revoke $\langle p, E\rangle$ and contextually substitute it with $\langle \bar{p}, E\rangle$. Note that both actions are justified by the fact that $\bar{S}^E$ controls E.

Consider a verifier that have to check the validity of a certificate $R = \langle p, E\rangle$. If the verifier has access to the full blockchain history, it can easily check, in the blockchain history, if R is currently not de-certified and not re-certified. For verifiers that act as light clients, we should provide a succinct independent certificate that includes information about R not being recently de-certified and re-certified. A way to achieve this is by storing the set $\mathcal{V}$ of all currently valid certificates in the state of a single smart contract and provide a Merkle-proof of the fact that, at a certain block, R was in $\mathcal{V}$, with the same approach described in Section V-C. Then, this Merkle-proof can be included in a succinct independent certificate. Set $\mathcal{V}$ should be updated as the last step of certification and re-certification, which can be easily accomplished by the same smart contract. It is not hard to include in this framework also de-certification by handling them by the same smart contract. For protocols in which the prover provides succinct independent certificate to the verifier on-demand, the same prover may take care to provide a recent enough certificate. Otherwise, the verifier should take care to obtain a recent proof my itself.

We just mention the possibility to admit *multiple certifications* for the same endpoint E assuming that any new certification coexists with the previous ones (for example for using different public keys for different purposes). In this case, the re-certification request should explicitly state that all previous certifications have to be invalidated, or should list which previous certifications should be invalidated.

This has no impact on the certification protocol but should be taken into account by verifiers.

Finally, we note that the above procedures are secure in our threat model (see Section VII-A): an attacker that has no knowledge of the private key cannot trigger the de-certification of a certified endpoint and an attacker that has no control on the endpoint cannot trigger its re-certification.

IX. EVALUATION

We evaluate our approach from several points of view with the intent to assess the applicability of our results in a practical contexts. In particular,

- we check the compliance of our approach with the W3C standardization work related to self-sovereign identity management,
- we evaluate latency and other aspects of both methods and compare them against the basic protocols recalled in Section III,
- we describe a proof of concept realization (PoC), based on the Ethereum technology, and describe practical issues and solutions,
- we discuss the costs of the actors in our PoC,
- we provide insights about the compatibility of our approach with other blockchain technologies.

A. CONFORMITY WITH W3C STANDARDS

In this section, we investigate the possibility to use our methods in conjunction with the standardization effort by the W3C consortium that are related to Self Sovereign Identity management, namely, *Verifiable Credentials* (VC) [51], which is a W3C recommendation, and *Decentralized IDentrifiers* (DID) [46], which currently is a W3C working draft. This standardization work is complex, covering a wide spectrum of use cases. For clarity, in this section, we refer only to the simplest cases. We start by focusing on VCes which is a stable tool to express general claims. Then, we focus on DIDes, on the basis of the current draft. DIDes are interesting, since they explicitly state a relationship between subjects and their endpoints.

VCes are objects (usually encoded as JSON objects) that intend to substitute *physical credentials*. A *subject* obtains, from a centralized *issuer*, a VC that states a *claim* about the subject itself (e.g., "has a driving license"). The VC contains a *proof* (for example a signature by the issuer) that can be verified by a *verifier* by cryptographic means. Normally, the verifier trusts the issuer, and the authenticity proof is enough to also trust the claim. The independent certificate introduced in Section V-C can be considered a VC, where the claim specifies the prover and its controlled endpoint while, technically, the issuer is a randomly selected committee. The verifier trusts its claim because of the properties of the certificate creation procedure and of the trust in the adopted blockchain. According to the design described in Section V-C, the signature is substituted by the Merkle proof of a part of the blockchain state and by the indication of the corresponding block. Clearly, the verifier should check the Merkle proof starting from a trusted block header. What we require is compatible with the VC recommendation, since in a VC the proof is equipped with a *type*, which specifies the kind of proof. W3C have a dedicated standardization work for proofs [26], in which a proof type is required to specify a *proof algorithm* that generates a *proof value*, that can be checked by running a *proof verification algorithm*. This schema is flexible enough to fit the certification methods proposed in Section V and the design described in Section V-C.

A DID is a URI in the form `did:`*method*:*identifier* that identifies a subject. A DID has to be resolved into a *DID document*, which specifies attributes for that subject and it is usually encoded as a JSON object. Details on how this resolution is performed is implicitly specified by the *method* of the URI. For example, the `ethr` method refers to the uPort identity manager (see Section XI), which we consider for our proof of concept in Section IX-C. The DID document can list *endpoints* related to the subject, under the *service* JSON key. For each of them, an *id*, a *type*, and a *serviceEndpoint* should be given. The *serviceEndpoint* may be a string (a URL), or a complex object, whose syntax and semantic depend on the *type* of the endpoint. While currently the draft does not specify many types, the proposed structure is extremely flexible and it is clearly able to cover a wide range of endpoints. This meets the flexibility required by the wide number of application contexts that we mentioned in Section II. Additional fields may be specified with each endpoint, this allows DID documents to support the design described in Section V-C. An example of DID document realizing that design is given in Section IX-C.

B. COMPARISON AGAINST BASIC PROTOCOLS

In this section, we theoretically analyze the `prover-receives` and `prover-sends` methods regarding to the time taken to obtain a certification and we compare the whole proving process against the basic protocols shown in see Section III.

We define the *latency* of a certification protocol as the time taken by the protocol from when the transaction t_R for the certification request is broadcasted in the blockchain network to when the certificate is completely known to all nodes. We assume the delay due to local computation to be negligible. We assume that the blockchain produces new blocks, regularly. We call b the interval of time between two consecutive blocks. For simplicity, we assume b to be constant and the number of transactions a single block can host to be much larger than the committee size k. We assume that the load of the blockchain (transactions submitted per unit of time) to be much lower than its maximum throughput, so that, each transaction is accepted in the next available block (blockchain scalability issues are widely discussed in other works, see for example [36], [42], [59], and [21]). If the instant in which a new transaction is received by the blockchain is independent from the previous block commit time, the expected time that a transaction waits before acceptance is $b/2$. We call p_b the time

to propagate a block to all nodes and p_t the time to propagate a new (still unaccepted) transaction to all nodes. We always assume $p_b < b$ and $p_t < b$.

Concerning communication of challenges through the endpoint, we denote by W the time taken by a message to reach destination, comprising transmission, propagation and possibly any form of interaction (see Section X for further considerations) and depends on the underlying communication technology of the endpoint. We assume that the bandwidth of the endpoint channel is large enough to send/receive all messages to/from the committee members in parallel.

Under these assumptions, the expected latency for the certification protocol of the `prover-receives` method can be derived by observing the diagram in Figure 4 and it turns out to be

$$p_t + \frac{b}{2} + \left\lceil \frac{p_t + W + p_b}{b} \right\rceil b + p_b. \tag{5}$$

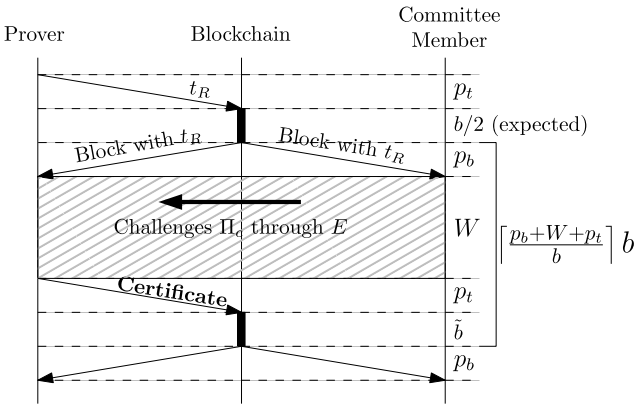

FIGURE 4. Timings for the certificate creation protocol for the `prover-receives` method (compare with Figure 2).

Note that, in the figure, $\tilde{b}$ denotes the expected time left for the next block to be produced, which depends on $p_t + W + p_b$ and in general is different from $b/2$. The same formula also holds for the latency of the certification protocol of the `prover-sends` method. Figure 5 shows the corresponding timing diagram, where *acceptance* messages are actually as many as the committee members that successfully terminate the protocol.

For both protocols, the fact that the certification is recorded in the chain means that the prover has received blocks with the corresponding transactions and can autonomously produce an independent certificate according to the description given in Section V-C.

For both methods, any verifier, which knows the header of the blockchain (i.e., a blockchain light client, see Section IV), takes negligible time to verify an independent certificate, since no network interaction is needed in this process.

The basic protocols shown in Section III, do not require any preliminary certification, but need to interact with the prover through the endpoint during verification. This interaction

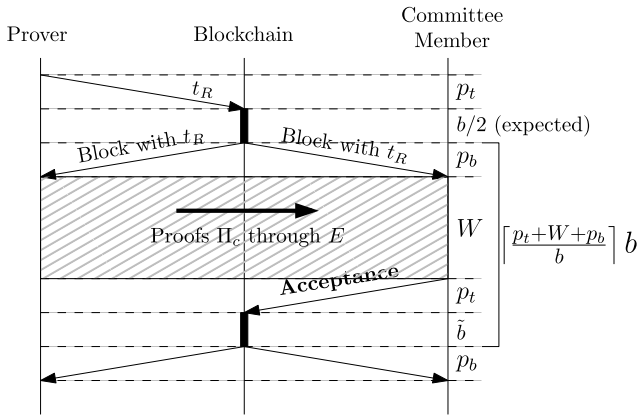

FIGURE 5. Timings for the certificate creation protocol for the `prover-sends` method (compare with Figure 3).

takes time W but should be performed each time a verifier needs to ascertain the endpoint control.

Concerning the number of messages that have to be transmitted through the endpoint, the basic protocols send one message for each verification, while both our methods only send one message for each committee member during certification.

This comparison is summarized in Table 2. We can conclude that our approach is favorable, with respect to the basic protocols in the following cases:

- when verifiers are many (or unbounded) and messages through the endpoint have a (monetary or procedural) cost,
- when W is too large for the verifier,
- when the prover may be off-line when the verifier intend to ascertain endpoint control.

On the other hand, the following aspects should be taken into account.

- The use of a blockchain introduces new costs (discussed in Section IX-E).
- Committee members should be motivated to participate to a certification protocol (see Section VI-C).
- Our approach may need procedures for certificate management (discussed in Section VIII).

C. PROOF OF CONCEPT

We realized a Proof of Concept[1] (PoC) with the purpose of showing that the proposed approach can be implemented with state-of-the-art technologies and with the intent to estimate blockchain costs (analyzed in Section IX-E).

Our PoC realizes the `prover-receives` method (see Section V-A) with a probabilistic selection of the committee (see Section VI-A). Endpoints are ⟨ip address, port⟩ pairs for UDP sockets. The PoC is based on the Ethereum blockchain technology.

[1]The code of the PoC is available at https://gitlab.com/uniroma3/compunet/networks/bindingpoc. The *master* branch contains the code described in this section. The branch *multiple_clients_performance* is related to experiments described in Section IX-D.

TABLE 2. Comparison between methods proposed in this paper (Section V), basic protocols (Section III), and centralized Certification Authority. See also comments in Sections IX-B, IX-E, and XI-C. We evidenced in bold the aspects that might be considered critical for the adoption in decentralized applications.

	Certification time	Verification time	Prover needs to be on-line during verification?	Number of messages through the endpoint	Centralization	Costs
This paper	$p_t + \frac{b}{2} + \left\lceil \frac{p_t+W+p_b}{b} \right\rceil b + p_b$	negligible	no	constant: one for each committee member during certification	decentralized	Tenths of dollars. Depends on $\bar{k}$, and cryptocurrency quotation. Cost may be lowered switching to cheaper blockchains.
Basic protocols	(not applicable)	W	**yes**	**one for each verification**	decentralized	In typical use cases, costs are negligible for a prover and **high for a verifier**.
Centralized CA	from minutes to **days**	negligible	no	constant	**centralized**	From tenth to **hundreds of dollars** for a single-address web certificate

It realizes $N = 1000$ potential committee members, one prover, and one verifier. Other parameters for the probabilistic committee selection are $k = 20$ and $\alpha = 3$. Since in our PoC there is no risk for committee members to be inconclusive, off-line, or colluding, we set $\bar{k} = k$. In this setting the average committee size is 40 and at least 20 cooperating members are enough to produce a certificate. The probability of having a committee with less than k members is less than 10^{-4} (see proof of Theorem 1). To give a measure of the security, consider an attacker that controls $m = 100$ members (out of $N = 1000$). The probability to successfully carry out a miscertification attack is less than 10^{-6} (using the Chernoff bound as in the proof of Lemma 1).

The PoC produces independent certificates (see Section V-C). They are stored in the blockchain as a standard DID according to the `ethr` method [3], [53] (currently provisional), which makes use of a central DID repository realized by a smart contract, called *Ethr-DID-Registry*.

The PoC also implements a mechanism to reward the committee members for their participation to the certification procedure (see Section VI-C). To do that, the PoC encompasses a smart contract, that we name *Binding*, to record each certification request, keep money contextually paid by the prover, and pay involved committee members when the prover shows the certificate to the Binding smart contract. This last transaction is also used to produce the independent certificate for the DID.

The architecture of our PoC is shown in Figure 6. The Binding smart contract is written in solidity, compiled as EVM bytecode and executed in the blockchain.[2] The rest of the software is written in javascript and runs in a single NodeJS [8] instance. Actors (i.e., the prover, the verifier, and potential committee members) are just objects in that instance. In our PoC, all actors are similar to light clients that are connected to the Ropsten testnet by nodes operated by Infura [2]. To limit the number of connections open at the same time to the Infura node, in our PoC, all actors access

[2]The Binding smart contract is deployed at address 0x71260C67A26Dd23CA9B2595b25e83974a0D2Aa8B of the Ropsten test network.

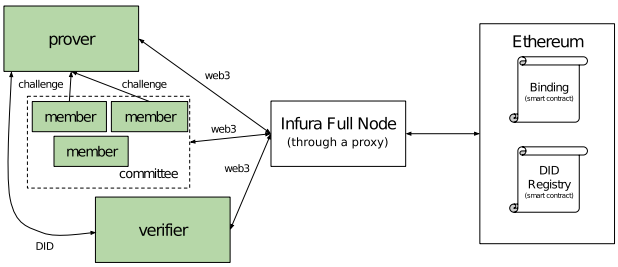

FIGURE 6. Architecture of our proof-of-concept realization and interaction between modules.

the Infura node by means of a special proxy object, which, in turn, uses the web3.js [14] library to contact the node.

We now provide some details about the execution of the certification protocol and the verification. The prover issues the request calling method *newRequest* of the Binding contract to which it provides its public key p, its endpoint E and the offered reward (in the Ether cryptocurrency). The transaction of this call is denoted t_R. Since the Ethereum technology does not allow to access the id of t_R within its execution, the Binding contract generates a new identifier R, for this request. The Binding contract keeps track, in its state, of R, p, E, the submitted reward, and the block number n_R. An *event* $e = \langle p, E, R, n_R \rangle$ is notified by means of the *emit event* feature of Ethereum. The block number n_R is needed to retrieve the hash b_R of block n_R, which is needed, together with R, to compute a shared and unpredictable seed for the random committee selection (see Section VI-A). Potential committee members are subscribed (through the proxy) to receive this kind of events. Each event e is enough for each potential committee member to check if it is part of the committee for the corresponding request. If c is a committee member, it chooses Q_c, prepares and signs Π_c (see Algorithm 1), and sends it in an UDP packet to E. The prover collects all Π_c's and, if there are enough of them that are correct, it packs the correct ones in G. Then, the prover calls method *payCommittee* of the Binding contract passing G, R, and b_R (b_R is not strictly needed, but its presence simplifies the task of the verifier). This call is

```
"publicKey": [ public key of the prover (see DID standard) ],
...
"services": [
    {
    "type": "certifiedUDPEndpoint",
    "serviceEndpoint":
        '{
        "endpoint": "ipaddress:port",
        "proof":
            {
            "tBlockNumber": n_P,
            "tAndMerkleProof": "transaction P (containing G, R, and b_R) and its Merkle proof, encoded",
            "receiptAndMerkleProof": "receipt for P and its Merkle proof, encoded",
            "tIndex": i
            }
        }'
    }
]
```

FIGURE 7. Excerpt of the DID document produced by our PoC, representing an independent certificate of endpoint control.

recorded in a certificate transaction P, which has two effects. First, P permanently stores G, R, and b_R in the blockchain ensuring, by smart contract check, that R exists and that it is coherent with b_R. Second, the Binding contract equally divides and pays the reward to all committee members that participated to G. Verification of the correctness of P is delegated to the verifier to reduce execution costs, however, it is in the interest of the prover to provide a correct P since the reward is paid in any case. When P is accepted, in a block whose number is n_P, the prover prepares an independent certificate which ideally should be a triple $\langle P, n_P, \text{MerkleProof}\rangle$ (see Section V-C). The Merkle proof is obtained exploiting method *eth_getProof* of Ethereum nodes [32] accessed by a suitable library [1]. Actually, there are some technical subtleties in using that API, since, it returns a byte string that encodes both P and its Merkle proof plus a transaction index i (i.e., the position of P in the block n_P), which is needed to verify the proof. Note that, Ethereum records transactions even if the corresponding contract execution rises an exception, hence, it is not enough to have a proof for the transaction. Ethereum stores, in each block, *receipts* with the outcome of each transaction. They are also equipped with their own Merkle tree. The *eth_getProof* API can also obtain proofs for them (with the same encoding). This information should also be part of our independent certificate as well.

After P is recorded in the blockchain, the prover prepares a DID containing the independent certificate and stores it in the Ether-DID-Registry, which is realized as a single smart contract. This operation is supported by a specific library [57]. The DID is also directly sent to the verifier, which checks the validity of all the above mentioned information against the trusted header of block n_P, which it stores as a light client. Verifier checks encompass the validity of G in P, as described in Algorithm 2.

The DID document produced by our PoC is shown in Figure 7. Unfortunately, at the time of writing, the library for the ether-did registry only support a simplified DID schema that does not include the possibility to have complex objects as `serviceEndpoint`. We worked around this problem by including as value for `serviceEndpoint` a string containing the JSON representation of the complex object we intend to include. Note the presence of P plus its proof, the receipt of P plus its proof, the index of P, and the block number n_P. This is enough for the verifier to check the integrity of the independent certificate starting from a trusted chain of block headers that it keeps, as a light client does.

D. EXPERIMENTAL RESULTS

To further provide evidence of the practical applicability of our approach, we tested our PoC simulating the load of a real application. Additionally, we also experimentally measure the maximum throughput that we can achieve in our setting and we compare it with a theoretical maximum.

We focus on a Robinson list application, where phone line subscribers can register to allow or deny phone calls by marketing operators. We imagine a blockchain-based application where users express their option, along the lines of the one described in [24]. In the blockchain, users are identified by public keys, while expressed options are related to telephone numbers. Associating a public key with a telephone number, is exactly the problem solved by our approach.

The Italian Robinson list is called *Registro Pubblico delle Opposizioni* (RPO) [9] and it is centrally managed, since 2011, by Fondazione Ugo Bordoni (FUB). Currently, users express their deny of marketing calls by using a web-based interface or a call center. FUB allowed us to analyze the history of new users' registrations. Currently, on the average, RPO receives about 50 subscriptions per day, but in the past a peak with about 25,000 subscriptions in single day (24 hours) was recorded, corresponding to 1041 subscriptions per hour, on average.

We set up an experiment to prove that our PoC can achieve a throughput (number of certifications per hour) that allows us to handle such a request peak. To this purpose, we focused

TABLE 3. Results of throughput measurement experiments.

Provers (n)	Interval starting block	Interval ending block	Experiment duration (sec)	N. of req.	N. of cert.	Throughput (cert/hour)
5	10.528.639	10.528.699	485	56	49	364
15	10.528.519	10.528.577	560	141	151	971
30	10.528.121	10.528.169	522	269	257	1772
50	10.572.394	10.572.449	522	395	387	2669
70	10.572.302	10.572.355	535	352	387	2604

on the performance of the decentralized certification process, hence, we deactivated DID creation and publishing features. In fact, they depend on an external service that we do not intend to involve in the test.

To measure the maximum throughput of our system, we considered an increasing number of provers n continuously performing requests. Each prover sends a request as soon as the previous one is recognized as fulfilled. In this setting, when the system is up to speed, n is ideally also the total number of requests that are submitted and not yet fulfilled. This setting also imposes a self-timing behavior so that, on average, the frequency of submitted requests matches the frequency of produced certificates. We performed the tests using the Ropsten testnet. We run each test for about ten minutes. When analyzing our data, we eliminated transient at the beginning (1 minute) and end (45 seconds) of the test. Currently, Ropsten allows about 8 million gas units per block. For our PoC, each certification request requires two transactions (request + certificate) for a total of about 600k gas units (this is true when the system is up to speed, see also Section IX-E), giving about 13 transactions per block. The Ropsten testnet currently produces one block every 10 seconds, on average, which gives a theoretical maximum throughput of 4,680 certifications per hour. Since our goal was to measure it in practice, we cared about running our experiments when the load of the testnet was low. Results of our experiments are shown in Table 3. Each row corresponds to one experiment, for which we provide the range of involved blocks,[3] the duration, the amount certification requests transactions recorded in the blocks, the amount of certificate transactions recorded in the blocks, and the computed throughput.[4] The indicated block ranges are those resulting after eliminating the transient, as specified above, and hence, the system can be considered up to speed, within the specified intervals. Note that, the amount of requests is always quite close to the amount of certifications, which confirms the described self-timing behavior. Differences between the number of certifications and the number of requests are due to boundary effects: requests that are issued in the transient before the starting block are not always perfectly compensated by requests that are accounted in the interval but have certifications after the ending block.

Figure 8, graphically shows how throughput varies by varying n. By increasing n, we correspondingly linearly increase the throughput of certifications that we request. However, after $n = 50$ no throughput increase is observed, which means we have approached the maximum throughput of the whole system and increasing n just gives an increase of the latency of each request.

The maximum throughput we obtained is about 2,600 certifications per hour, which is well above the peak of about 1,000 certifications per hour needed to support the highest request peak in the history of RPO. In fact, at 2,600 certifications per hour we can process the peak of 25,000 requests arrived in 24 hours, in about 10 hours.

Regarding the inability to reach the maximum theoretical throughput, this can be ascribed to non-ideal conditions that are not under our control, in particular, the unavoidable presence of other users and the observed ''unfair'' behavior of some miners of the Ropsten testnet that silently discarded all our transactions in their mined blocks.

Even if our experiments are carried out in a testnet, we believe that they prove that our approach can scale to the throughput of real applications, provided that the underlying blockchain is adequately dimensioned.

E. COST ANALYSIS

In this section, we analyze the costs of our approach, focusing on the `prover-receives` method and on our PoC realization.

The following is a general list of costs with the subjects that are supposed to sustain them.

- The prover sustains the cost of placing the certification request on the blockchain and of notarizing the certificate transaction P in the blockchain.
- The cost to interact through the endpoint is sustained by the prover and/or by committee members depending on the kind of endpoint (see also Section X).
- The prover sustains the cost of the creation of the DID document that contains the independent certificate.
- The verifier sustains the cost to verify the independent certificate in the DID document.

[3] Blocks are publicly accessible to everyone. To access them directly from the blockchain, several systems are available. For example, http://ropsten.etherscan.io can be used.

[4] We published both data already extracted from blocks and scripts to compute the statistics shown in the table. The reader can find them at https://gitlab.com/uniroma3/compunet/networks/bindingpoc/-/tree/multiple_clients_performance/bindingPerformance.

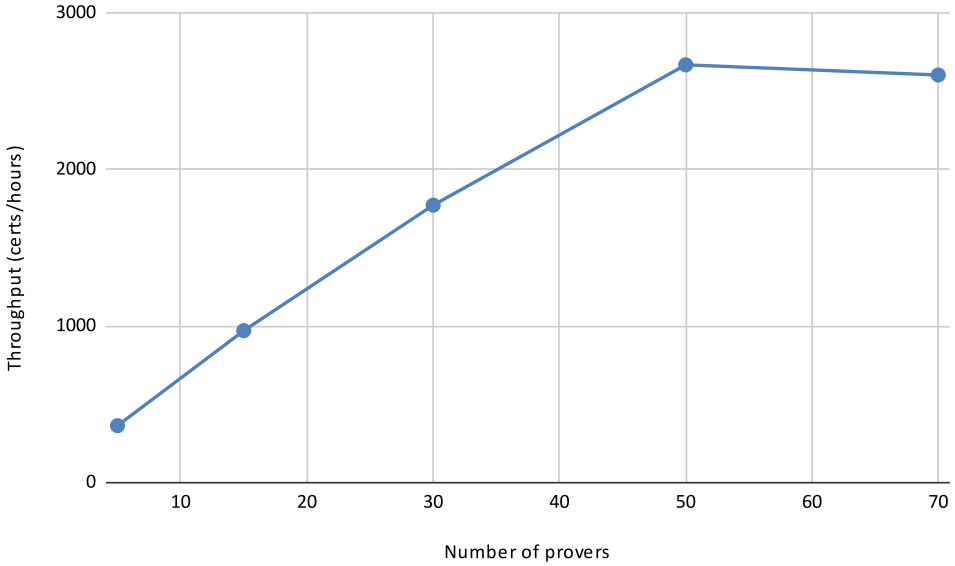

FIGURE 8. Throughput for increasing number of provers. The maximum throughput turns out to be about 2,600 certifications per hour.

All actors have to sustain the cost of operating a light client, which we do not consider in our analysis. Actually, the light client approach is targeted to have a low operating cost and reducing it is a general problem with several contributions (see, for example, [35] and [22]).

We note that the verifier has only to perform local computation, hence its cost is negligible. This is coherent with our objective to support a large number of verifications.

Concerning the endpoint interaction cost, in the PoC, this is clearly irrelevant, since we use just UDP communications. However, in a real context, this can be substantial. If the communication through the endpoint have to be paid by the prover, no problem arises, since it is in its interest to pay it. However, if it must be paid by the committee members, this may turn out to be a disincentive to cooperate in the certification protocol. Hence, it is important to provide a mean to pay committee members to both reward them for their cooperation and to reimburse the cost they bear to participate. Our PoC shows how it is possible to integrate that reward in our certification protocol.

We now focus on the costs of the transactions. We provide costs in Ethereum gas units, which is the accounting unit for smart-contract execution costs in Ethereum. Each gas unit has a cost which is fixed by the market. Afterward, we discuss actual costs in *ether* (the cryptocurrency of Ethereum, denoted with its code *ETH*) and in fiat currency (US dollars, USD).

- The call to the *newRequest* method costs 208,092 gas units, for our PoC, independently from the committee size. In addition, the prover transfers an amount of ether to reward its cooperation and reimburse the costs of communicating on the endpoint channel. Since this amount is application specific, we do not discuss it further.
- The publication of the certification transaction, made by calling the *payCommittee* method, has a cost that linearly depends by the number of committee members x involved in G. In our experiments, the needed gas units are approximately given by $41598.1\ x + 30162.4$. For our PoC we have $x = \bar{k} = 20$, giving about 862,124 gas units. While carrying out the experiments of Section IX-D, we noted that when the call is iterated many times, the needed gas units can be reduced to about 372,000. In the following cost estimation, we do not consider this reduction.
- To update the DID Document, the prover calls the *setAttribute* method of the Ether-DID-Registry. The cost of this call also depends on $\bar{k}$. For our PoC ($\bar{k} = 20$), this cost is 391,521 gas units.

The actual cost for each gas unit, called *gas price* and valued in ETH, may change very much over time, as well as, the USD/ETH exchange ratio. We considered the average of these values between February 1st, 2020 and February 1st, 2021. The average USD/ETH exchange ratio turned out to be 398.44 USD for one ETH and the average gas price for the Ethereum main network turned out to be 69×10^{-9} ETH. With this values, the total amount spent by the prover in our PoC is about 40 USD to obtain a certificate on the Ethereum main network. This cost is in line with costs of certificates issued by a certification authority for certifying the control of a web server, for one year [5]

This analysis shows that our solution does not simply provide a fully decentralized certification service, but also its cost is comparable with analogous centralized solutions. In any case, to reduce costs, one may consider to adopt a

[5]Some pricing examples are available here: https://www.webhostingsecretrevealed.net/blog/ecommerce/buy-ssl-certificate/. The average price of these providers for a single domain certificate for one year, at the time of writing, is about 142 USD per year.

TABLE 4. Human involvement and latency of the channels considered in Section II.

Channels (corresponding endpoint)	Latency magnitude order	Human involvement
SMS (phone number)	seconds	no
Phone call with IVR (phone number)	minutes	required
Postal mail (address)	days	required
Email (email address)	minutes	no
TCP or UDP communication (IP addr.)	seconds	no
HTTP static page (URL)	seconds	optional
HTTP web app (URL)	seconds	no
Domain name system (domain name)	seconds	optional
Bank transfer (bank details, IBAN)	days	optional

different and cheaper blockchain technology. Section IX-F discusses this.

F. COMPATIBILITY WITH DIFFERENT BLOCKCHAIN TECHNOLOGIES

We realized our PoC on Ethereum because it is currently one of the most adopted blockchains for the development of smart contracts. However, Ethereum transaction fees may be considered very expensive. There are other blockchains (such as Tezos [11], EOS [4], etc.) offering smart contracts with the same expressive power but with much lower costs making our protocol much more affordable. Note that, the smart contract essentially guarantees two requirements: (1) it forces the prover to block a stake used to reward the committee members; (2) when the prover declares its certificate, it pays only the specified committee. We can achieve these requirements, even on blockchains providing smart contract with less expressiveness. For instance, on Algorand [23], we can realize our approach using one stateless smart contract per certification instead of the singleton smart contract that we adopted in our PoC. This stateless smart contract should have, wired in its code, the total reward for the committee, the number of required committee members (i.e. $\bar{k}$) and the pair $\langle p, E\rangle$. At deployment, the prover transfers the total reward to the contract. When the prover receives $\bar{k}$ partial challenges, it cannot send a single transaction paying all committee members but $\bar{k}$ transactions are needed, each triggered by one of the involved committee members. Our method can be applied, also, on blockchains where it is not possible to create smart contracts. In this case, we can still achieve endpoint certification, exploiting transactions with arbitrary data, but we cannot enforce payment of committee members.

X. TECHNICAL CONSIDERATIONS

In this section, we discuss technical aspects related to the endpoints considered in Section II and we discuss if and how they impact the adoption of our certification methods. We also provide some guidelines to choose between our two certification methods, with some examples.

Table 4 shows the order of magnitude of the time taken by endpoint channels to deliver a single message (*latency* column) and the need of a human to use channels (*human involvement* column). About latency, we note that, for the considered channels, it ranges from seconds to days. About human involvement, we state that, for some channels, no human involvement is needed. These are cases in which communication can be easily managed via software, like for email or SMS. For others, the involvement of a human is required by the very nature of the communication channel, like for postal mail or for phone calls. Other situations are in the middle, in the sense that, by design, a human is supposed to be involved in configuring a system (e.g., for DNS records at registrars, static web page editing, bank transfer order placement). However, this can be considered optional, in the sense that it can be avoided by proper system integration. A related aspect is that a channel that does not require human involvement may require specific equipment (e.g., an *Interactive Voice Response*, *IVR*, for phone calls) or custom software (e.g., a web app), which may or may not be easily available to committee members or to provers, depending on the application context.

In principle, involving humans and using a slow channel do not impact the adoption of our certification methods. However, in an application, it should be carefully considered if the effort of the human activity and the timings are compatible with the application requirements and with the rewards offered to the committee members. Additionally, we note that, slow and human-needing channels pose a further pressure for the adoption of our solutions with respect to basic protocols, especially when the number of verifications is expected to be large.

After a kind of endpoint/channel is selected, to choose between the `prover-receives` and the `prover-sends` methods, it is useful to take into consideration several other aspects, which are summarized in Table 5.

The first column specifies if sending messages through the endpoint channel has a cost. This is true, for example, for bank transfers, phone calls, postcards, etc. We specified "no" where costs are negligible in practice. We do not go further into this. Just note that the subject that pays usually depends on the direction of the communication and, for sustained use, flat pricing might be available, which might ease the choice of a specific endpoint/channel.

Columns *protocol* and *communication style* put in evidence that channels are very different with respect to how they deliver messages. Some of them are based on an underlying protocol (e.g., TCP, HTTP, DNS), while others can be used by simply sending one message (e.g., UDP, email, SMS, bank transfers). In the latter case, channel is normally used in push-style (i.e., data source initiates the transmission), but pull-style (i.e., data receiver initiates the transmission asking for the data) can still be adopted, if there are constraints on who can initiate the transmission. For example, this is the case for IP-based communications where private addressing [47] and network address translation [52] may be present. When a channel encompasses a protocol, this may impose the use of a pull-style communication. For example, this is the case of DNS or HTTP with static page. In other situations, the choice between push and pull style is leaved to the designer (e.g., for HTTP with a web app, or for plain TCP).

TABLE 5. Several features of the endpoint/channel are listed. The *choice* columns, provides informal examples of choices of certification method and push/pull communication style with a brief reason.

Endpoint (with channel, specified when options are available)	Cost	Protocol	Comm. style	Eavesdropping difficulty	Spoofing difficulty	Choice		
						Method	Style	Reason
withSMSyesnopushmediumhardproverPhone number with SMS	**yes**	no	push	medium	**hard**	prover-sends	push	prover pays, security
Phone number with IVR call	**yes**	yes	any	medium	**hard**	prover-sends	push	prover pays, security
Postal mail address	yes	no	push	**medium**	easy	prover-receives	push	security
Email address	no	no	push	**medium**	easy	prover-receives	push	security
IP address with UDP	no	no	push	**medium**	easy	prover-receives	push	security
IP address with TCP	no	yes	any	medium	medium	prover-receives	push	easier to realize
URL with HTTP static page	no	yes	pull*	medium, hard if a CDN is used	medium or hard	prover-sends	pull*	easier to realize
URL with HTTP web app	no	yes	any	medium, hard if a CDN is used	medium or hard	prover-sends	pull	easier to realize
Domain name	no	yes	pull*	medium, hard for distributed DNS	medium or hard	prover-sends	pull*	sending data is supported by standard infrastructure
Bank account details / IBAN	**yes**	no	push	medium	**hard**	prover-sends	push	prover pays, security

*This is mandated by the technical features of the channel.

It is important to note that, adopting pull-style communication, has an impact on the committee selection approach. Namely, this switches the suitable approach with respect to that suggested in Section VI. In fact, in Section V, we described certification protocols adopting a push-style communication, but adopting pull-style communication reverses the initiator of the communication, hence, it makes the deterministic approach suitable for the `prover-receives` method and the probabilistic approach suitable for the `prover-sends` method.

Depending on the criticality of the application context, an important aspect to consider is the security of the channel, which may have an impact on the overall security of the certification procedure. In Section VII, we analyzed the security of our certification protocols assuming an ideal channel. However, in practice, security also depends on the specific kind of endpoint and channel considered. To put the following discussion in the right light, we point out that all security concerns about eavesdropping and spoofing that we are going to consider, also affect basic protocols commonly used in the current practice, for example, for two factor authentication.

Suppose that an endpoint E is under the control of a subject S and an attacker M intends to miscertify that E is under its control. Normally, messages to E are routed in a network (e.g., Internet, telephone network, postal logistic network, etc.). Attacker M has a limited number of *points of presence* (*PoP*) in that network that can be used to perform an attack. Depending on the context, a PoP can be a controlled computer or appliance, a postal mail insider, etc. From these PoPs, in principle, M can create spoofed messages, that look as if they were sent from E, and can eavesdrop messages destined to E passing through that PoP. Different channels present different security levels for spoofing and eavesdropping, hence, it is interesting to analyze what capability are actually needed by M for accomplishing an attack that exploits channel vulnerabilities.

Let us consider channels with no protocol and used according to a push-style with the `prover-sends` method. In these cases, M requires only the spoofing capability to perform a miscertification attack and this is easy to perform from any PoP. For the `prover-receives` method, M requires only eavesdropping capability to attack. However, in this case, M can see all messages only if it has a PoP close to E. In certain cases, this means close to S, but if E is served in a distributed manner, like for certain DNS networks or *Content Delivery Networks* that adopt anycast routing, eavesdropping can be very hard.

We now consider pull-style communications. In these cases, some form of protocol is present that involve sending messages in both directions and all messages have to be successfully delivered to conclude the communication. Hence, M requires both spoofing and eavesdropping capability for a successful attack. In this case, the overall difficulty of the attack is given by the most difficult task between spoofing and eavesdropping. Since, eavesdropping requires a PoP close to E, this is needed for the whole attack if certification is based on a pull-style communication channel. Again, if E is served in a distributed manner this is hard to achieve.

In columns *eavesdropping difficulty* and *spoofing difficulty*, we provide an informal rough evaluation of the difficulty of an attacker to perform these attacks for all endpoint/channels considered. A successful eavesdropping is in general not that easy to achieve since the attacker PoP have to be close to E, and it is hard if E is on a distributed infrastructure. Spoofing is generally easy for channels with no protocol, but, if the organization that runs the network (e.g., a telco operator or a bank) deploys countermeasures to contrasts the attack, this becomes hard, and harder than

TABLE 6. Comparison with related literature.

	Oracle					IdM		Our protocols
	TLSNotary	AEternity	Witnet	Chainlink	Astrea	Centralized CA	SSI approach	
Support creation of new certificates	Yes	Yes	Yes	Yes	Yes	Yes	No: need external issuer	Yes
Majority-based/Absence of Trusted Third Party/Decentralized	No	Yes	Yes	Yes	Yes	No	N.A.	Yes
Absence of Lazy Voter problem	N.A.	No	No	No	Yes	N.A.	N.A.	Yes
Interaction through the endpoint during certification	No	No	No	No	No	Yes	N.A.	Yes
Implementation of the protocols	Yes	Yes	Yes	Yes	No	Yes	Yes	PoC is available

just eavesdropping. If the channel encompasses a protocol, spoofing is at least as harder as eavesdropping.

In the *choice* columns, we propose reasoned choices between `prover-receives` and `prover-sends` methods, with associated push/pull style. These should be intended as examples of applications of the above considerations and not as mathematically derived results. In fact, a complete decision process should consider also other aspects. For example, we did not took into account rewards and reimbursement from the prover to each committee member.

XI. RELATED WORK AND COMPETITORS

Since endpoints exist independently from a blockchain and the certificate (i.e., the output of our certification protocols) is written in the blockchain, the research about blockchain *oracles* is of interest for this paper. Further, a certificate of endpoint control can be used as an attribute of the identity of a user in applications (which may be blockchain-based or not), hence *identity management* research is also related to this paper. In this section, we briefly review relevant research in both areas and describe its relation with our contribution. Table 6 summarizes the content of this section.

A. ORACLES

The need of feeding a blockchain-based application with off-chain data is handled by the so-called (inbound) *oracles*. Most oracles (but not all) focus on retrieval of data published on the web. The first simple oracles had to be considered trusted, but lately a number of publicly accessible solutions was proposed that address the trust problem in different ways: for example TLSNotary [12] is used to certify data taken from HTTPS-based web-pages and a trusted execution environments (like Intel SGX [25]) is used to ensure correct software execution on third party premises. A common technique is to involve a multiplicity of subjects that assert on-chain what they observed off-chain. In this case, contradictory assertions should be properly handled. A recent survey on trustworthy oracles [18] provides a comprehensive comparison of currently known solutions and approaches. To the best of our knowledge, no proposed oracle specifically deal with the problem we address in this paper. However, some proposals are general-purpose in nature and can be somehow adapted to address our problem. At the end of this section, we point out the distinctive aspects of our proposal.

Æternity [16] is a general purpose blockchain ecosystem that also explicitly supports oracles. It reuses the consensus mechanism of the blockchain to also agree on the state of the outside world. Witnet [27] is a blockchain designed as an infrastructure to support the execution of decentralized oracles in which each participant has more chance to contribute (and to be rewarded) on the basis of its *reputation*, earned by who agree with the majority. Simple reputation-based approaches are affected by the so-called *lazy voter* problem (defined in [15]). A lazy voter always participate with a cheap answer that is independent from the state of the real world and has some good chance to be the majority answer and get a reward. In fact, in certain cases, being lazy may be considered the most rational choice by all participants. This is supported by classic results about behavior under scarcity of information or coordination, in economics (e.g., the Akerlof's "The Lemon Market" [17]) and game theory (e.g., the "focal point" concept by Schelling [48]). ChainLink [30] is a reputation-based general-purpose decentralized oracle that motivates each *oracle operator* to behave honestly by making reputation statistics public. Astraea [15] is a proposal (currently still theoretical) of a decentralized oracle based on a voting system that address the lazy voter problem by setting up a *game* between two class of players: voters and certifiers. Voters play a low-risk/low-reward role that is resistant to adversarial manipulation while certifiers play a high-risk/high-reward role that guarantee an high level of accuracy. In Astraea, a Nash equilibrium exists where all rational players behave honestly.

See Section XI-C for a discussion of trustworthy oracles with respect to our approach.

B. IDENTITY MANAGEMENT

An *Identity Management* (IdM) is a framework (of policies and technologies) to ensure proper access of users to resources. It is standardized by ISO [7] and regulated by the *eIDAS* [56] regulation of the European Union. Protocols and standards related to IdM systems are surveyed in [41]. Identities can be useful across several organizations. For this reason, *single sign-on* approaches, such as Facebook

connect, are adopted [43], but usually rely on centralized architectures. The idea of realizing IdM on top of blockchains is a further step toward making IdM independent from a specific organization. In private/permissioned blockchains some kind of trust among participants exists, hence implementing IdM over them does not introduce new conceptual problems. The *Self-Sovereign Identity (SSI)* approach, surveyed in [40], envisions solutions in which subjects should be able to create and control their own identity, without relying on any centralized authority. In this context, public/permissionless blockchains are fundamental tools. W3C hosts ongoing efforts to standardize the building blocks of SSI, like *Decentralized Identifiers (DIDs)* [46] and *Verifiable Claims/Credentials* (*VC*) [51]. A realization of this framework is backed by the Decentralised Identity Foundation [28]. The relation between DIDs and eIDAS is analyzed in [31].

One of the first attempts to design an IdM system deployed on the blockchain trying to accomplish self-sovereign identities is Namecoin [33]. The uPort system [13] makes use of Ethereum smart contracts to record and retrieve simple DIDs. Hyperledger Indy [5] is an identity management system built on a permissioned blockchain. Sovrin [50] is an public IdM network built on Indy composed only by trusted institutions. ShoCard [49] binds existing trusted credentials (e.g., a passport), with additional identity attributes by means of Bitcoin transactions. The last three systems are also analyzed and compared in [29]. Further recent contributions are Sora [54], DNS-IdM [34], and [55]. All the above mentioned SSI systems include decentralized identity registries but still rely on external and centralized certification authorities to obtain any certification.

Our contribution is a decentralized methodology to create certificates of endpoint control. Hence, it is complementary to any SSI proposal. Our PoC, described in Section IX-C, is based on Ethereum and on the uPort realization of the W3C DID standard. A detailed relation between our approach and the W3C standardization efforts is provided in Sections IX-A and IX-C.

C. COMPARISON WITH EXISTING SOLUTIONS

In section IX-B, we explained how our solution compares with basic protocols, which is the natural decentralized and currently-adopted alternative. In this section, we investigate how our approach compares against the main other competitors.

Decentralized trustworthy oracles (see Section XI-A) appear as natural competitors. However, they are based on the concept of *voters*, where each voter declares what it is supposed to be its observation. For this reason they are affected by the lazy voter problem (see Section XI-A): lazy voters chose a cheap answer that is likely to be the majority answer when many voters are lazy. The main problem to solve in these systems is to assess (or force) honesty of voters.

On the contrary, in our approach, endpoint control is *not* an opinion of a voter. Actually, our committee members just collect the cryptographic proof of endpoint control, derived from an interaction through the endpoint, and communicate them to the prover for inclusion in the certificate. Hence, in our approach, by design, committee members cannot be lazy voters. Even in the cases in which a human is involved (see Section X), (s)he just ''copy'' the received cryptographic proof into a digital form. From this point of view, our problem is fundamentally different from a generic oracle problem. For this reason, we was able to formally prove the high accuracy of our approach, with high probability, relying only on random committee selection. For the same reason, our approach is easy to implement even on a common Ethereum network.

Regarding the possibility to compare our approach against a trustworthy oracle, consider the following cases.

1) Consider a procedure realized with a trustworthy oracle that does not involve any interaction through the endpoint. In this case, the oracle is just a decentralized way to query a third-party centralized source of information (e.g., yellow pages or the billing statement of the telecom operator), that should be considered trusted. However, we excluded trust in a centralized third party since the beginning (see Section I).
2) Suppose to adopt a human-based procedure that interacts with the human prover through the endpoint. In this case, the significance of any experimental comparison is limited, since timings and accuracy are negatively affected by the presence of the human. Further, this experimentation might turn out to be quite tricky: should it involve real humans or humans should be modeled in some way?
3) Suppose to consider an automatic approach adopted on top of a trustworthy oracle infrastructure with the purpose to avoid the involvement of humans. It may or may not be affected by the lazy-voter problem. If this approach is not affected by the lazy-voter problem, it essentially realizes a solution very similar to the one described in this paper, using a trustworthy oracle infrastructure as a regular blockchain. In particular, any specific feature of the chosen infrastructure to contrast the lazy-voter problem is disregarded. We think that this approach is too similar to the setting adopted in Section IX to be considered a competing solution. On the contrary, we are not aware of any reasonable automatic approach affected by the lazy-voter problem.

On these bases, we think that performing comparison tests of our approach against a trustworthy oracle is not useful and possibly misleading.

Regular (centralized) certification authorities (CAs) are clearly competitors of our approach. As already mentioned in the introduction, CAs are the de-facto standard to obtain electronic certification about many kinds of human-verifiable assertions, and this is true also for certain endpoints, the notable case being secure website certification. First of all, it must be stressed that CAs are centralized, which is a big drawback for decentralized applications, as already

mentioned in the introduction. In any case, it is useful to compare CAs against our approach with respect to other aspects. In Table 2, we (also) summarize the comparison of our approach against regular CAs. Since, both approaches produce an easily verifiable certificate, verification time is negligible and prover does not need to be on-line. However, while in regular CAs the verifier need to trust the public key of the CA, in our approach the verifier does not need this trust, but have to be a light client of the underlying blockchain infrastructure. About the number of messages sent through the endpoint for each certification, this is constant both for CAs and for our approach. Due to decentralization, in our approach, this number is typically higher, but it may tuned to strike a good application-dependent trade-off between the number of messages to be sent on the channel (see the k parameter, in Section V) and security. Costs are typically larger for certification authorities (see Section IX-E), since checks are not completely automatically performed. However, CAs usually perform additional checks besides just technical endpoint control, like, for example, a check on the real-life identity of the public key owner. These checks may provide additional security or may represent a problem, for example if just endpoint control certification is needed while real-life identity have to be concealed. These additional checks also make the time of producing a certificate by a CA typically much longer than in our approach.

XII. CONCLUSION

We described two decentralized (blockchain-based) methods to create certificates regarding control of an endpoint by a subject identified solely by its private/public key-pair. We explored many aspects of our proposal including, security, applicability with several kinds of endpoints, and certificate management. We provided a proof-of-concept realization showing that our approach is easy to implement in practice. We also evaluated latency and blockchain-related costs.

With respect to currently adopted naive decentralized approaches, our protocols do not require the subject to be on-line and the endpoint load does not depend on the number of verifications. This makes our approach especially suited for applications where a large number of subjects, which may be not always on-line, have to be verified by a large number of verifiers.

Our contribution complements current state of the art in the field of decentralized self-sovereign identity management by providing a fully decentralized way of certifying endpoints. Our approach can also be compared with decentralized oracles in the sense that it provides, on-chain, a proof of an off-chain truth. However, we leveraged the specificities of our problem to provide an easy-to-realize solution that can be based on most of the existing blockchain networks.

As a future work, we envision the application of our approach in practical contexts. We also plan to evolve our proof-of-concept realization into a publicly usable library and to provide support for its adoption with the most common blockchain technologies.

ACKNOWLEDGMENT

A preliminary version of this article was presented at the 2020 IEEE Symposium on Computers and Communications (ISCC) [DOI: 10.1109/ISCC50000.2020.9219594].

REFERENCES

[1] (Feb. 2021). *Eth Proof 2.0.0—Get a Merkle-Proof From the Blockchain*. [Online]. Available: https://github.com/zmitton/eth-proof
[2] (Dec. 2020). *Ethereum API|IPFS API Gateway|ETH Nodes as A Service|Infura*. [Online]. Available: https://infura.io/
[3] (Feb. 2021). *ETHR DID Method Specification*. [Online]. Available: https://github.com/decentralized-identity/ethr-did-resolver/blob/master/doc/did-method-spec.md
[4] (Feb. 2021). *Home|EOSIO Blockchain Software & Services*. [Online]. Available: https://eos.io/
[5] (Jun. 2020). *Hyperledger Indy*. [Online]. Available: https://github.com/hyperledger-archives/education/blob/master/LFS171x/docs/introduction-to-hyperledger-indy.md#hyperledger-indy-references
[6] *Security and Privacy—A Framework for Identity Management—Part 1: Terminology and Concepts*, document ISO/IEC 24760, New York, NY, USA, 2019.
[7] *Security and Privacy—A Framework for Identity Management*. ISO/IEC 24760, New York, NY, USA, 2019.
[8] (Dec. 2020). *Nodejs*. [Online]. Available: https://nodejs.org/it/
[9] (Apr. 2020). *Registro Pubblico Delle Opposizioni*. [Online]. Available: http://www.registrodelleopposizioni.it/en
[10] *The Transport Layer Security (TLS) Protocol Version 1.2*, document RFC 5246, Mar. 2021. [Online]. Available: https://tools.ietf.org/html/rfc5246
[11] (Feb. 2021). *Tezos|Secure Upgradable Built*. [Online]. Available: https://tezos.com/
[12] (Mar. 2021). *Tlsnotary—A Mechanism for Independently Audited*. [Online]. Available: https://tlsnotary.org/TLSNotary.pdf
[13] (Apr. 2020). *UPort: A Self-Sovereign Identity and User-Centric Data Platform*. [Online]. Available: https://github.com/uport-project/specs
[14] (Dec. 2020). *Ethereum JavaScript API—Web3.js 1.0.0 Documentation*. [Online]. Available: https://web3js.readthedocs.io/en/v1.3.0/
[15] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain oracle," in *Proc. IEEE Int. Conf. Internet Things*, Oct. 2018, pp. 1145–1152.
[16] T. Arts, Y. Malahov, and S. Hanse, "Eternity: Open source blockchain for scalable and secure smart contracts," Æternity Community, Tech. Rep., May 2020.
[17] G. A. Akerlof, "The market for 'Lemons': Quality uncertainty and the market mechanism," in *Uncertainty in Economics*. Amsterdam, The Netherlands: Elsevier, 1978, pp. 235–251.
[18] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, "Trustworthy blockchain oracles: Review, comparison, and open research challenges," *IEEE Access*, vol. 8, pp. 85675–85685, 2020.
[19] A. M. Antonopoulos, *Mastering Bitcoin: Programming Open Blockchain*. Newton, MA, USA: O'Reilly Media, 2017.
[20] A. M. Antonopoulos and G. Wood, *Mastering Ethereum: Building Smart Contracts and DApps*. Newton, MA, USA: O'Reilly Media, 2018.
[21] M. Bernardini, D. Pennino, and M. Pizzonia, "Blockchains meet distributed hash tables: Decoupling validation from state storage," in *Proc. Distrib. Ledger Technol. Workshop*, vol. 2334, P. Mori, M. Bartoletti, S. Bistarelli, Eds., 2019, pp. 43–55.
[22] B. Bunz, L. Kiffer, L. Luu, and M. Zamani, "FlyClient: Super-light clients for cryptocurrencies," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 928–946.
[23] J. Chen and S. Micali, "Algorand: A secure and efficient distributed ledger," *Theor. Comput. Sci.*, vol. 777, pp. 155–183, Jul. 2019.
[24] A. Cirillo, A. Mauro, D. Pennino, M. Pizzonia, A. Vitaletti, and M. Zecchini, "Decentralized robinson list," in *Proc. 3rd Workshop Cryptocurrencies Blockchains Distrib. Syst.*, Sep. 2020, pp. 1–6.
[25] V. Costan and S. Devadas, "Intel SGX explained," in *Proc. IACR*, 2016, vol. 86, pp. 1–118.
[26] D. Longley and M. Sporny. (Jan. 2021). *Linked Data Proofs 1.0*. [Online]. Available: https://w3c-ccg.github.io/ld-proofs/
[27] A. Sánchez de Pedro, D. Levi, and L. Iván Cuende, "Witnet: A decentralized Oracle network protocol," 2017, *arXiv:1711.09756*. [Online]. Available: http://arxiv.org/abs/1711.09756

[28] (2018). *Decentralized Identity Foundation*. [Online]. Available: https://identity.foundation/
[29] P. Dunphy and F. A. P. Petitcolas, "A first look at identity management schemes on the blockchain," *IEEE Secur. Privacy*, vol. 16, no. 4, pp. 20–29, Jul. 2018.
[30] S. Ellis, A. Juels, and S. Nazarov. (Sep. 2017). *Chainlink: A Decentralized Oracle Network (V1.0)*. Accessed: Mar. 2021. [Online]. Available: https://dev.exodus.io/assets/docs/chainlink-whitepaper.pdf
[31] EU. (Apr. 2020). *EIDAS Supported Self-Sovereign Identity*. [Online]. Available: https://ec.europa.eu/futurium/en/system/files/ged/eidas_supported_ssi_may_2019_0.pdf
[32] S. Jentzsch and C. Jentzsch. (Nov. 2020). *EIP-1186: RPC-Method to Get Merkle Proofs*. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1186
[33] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of Namecoin and lessons for decentralized namespace design," in *Proc. 14th Annu. Workshop Econ. Inf. Secur.*, Delft, The Netherlands, Jun. 2015, pp. 22–23.
[34] J. A. Kassem, S. Sayeed, H. Marco-Gisbert, Z. Pervez, and K. Dahal, "DNS-IdM: A blockchain identity management system to secure personal data sharing in a network," *Appl. Sci.*, vol. 9, no. 15, p. 2953, Jul. 2019.
[35] A. Kiayias, A. Miller, and D. Zindros, "Non-interactive proofs of proof-of-work," in *Proc. 24th Int. Conf. Financial Cryptogr. Data Secur. (FC)*, Kota Kinabalu, Malaysia. Springer, 2020, pp. 505–522.
[36] S. Kim, Y. Kwon, and S. Cho, "A survey of scalability solutions on blockchain," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2018, pp. 1204–1207.
[37] R. Lamberty, D. de Waard, and A. Poddey, "Leading digital socio-economy to efficiency—A primer on tokenomics," 2020, *arXiv:2008.02538*. [Online]. Available: http://arxiv.org/abs/2008.02538
[38] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine, "A general model for authenticated data structures," *Algorithmica*, vol. 39, no. 1, pp. 21–41, May 2004.
[39] R. C. Merkle, "Protocols for public key cryptosystems," in *Proc. IEEE Symp. Secur. Privacy*, Apr. 1980, p. 122.
[40] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Comput. Sci. Rev.*, vol. 30, pp. 80–86, Nov. 2018.
[41] T. Miyata, "A survey on identity management protocols and standards," *IEICE Trans. Inf. Syst.*, vol. E89-D, no. 1, pp. 112–123, Jan. 2006.
[42] G. D. Monte, D. Pennino, and M. Pizzonia, "Scaling blockchains without giving up decentralization and security: A solution to the blockchain scalability trilemma," in *Proc. 3rd Workshop Cryptocurrencies Blockchains Distrib. Syst.*, 2020, pp. 71–76.
[43] A. Pashalidis and J. C. Mitchell, "A taxonomy of single sign-on systems," in *Information Security Privacy*, R. Safavi-Naini and J. Seberry, Eds. Berlin, Germany: Springer, 2003, pp. 249–264.
[44] D. Pennino, M. Pizzonia, and A. Papi, "Overlay indexes: Efficiently supporting aggregate range queries and authenticated data structures in off-the-shelf databases," *IEEE Access*, vol. 7, pp. 175642–175670, 2019.
[45] D. Pennino, M. Pizzonia, A. Vitaletti, and M. Zecchini, "Binding of endpoints to identifiers by on-chain proofs," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2020, pp. 1–6.
[46] D. Reed and M. Sporny. (Jan. 2021). *Decentralized Identifiers (DIDs) V1.0: Core Architecture, Data Model, and Representations*. [Online]. Available: https://www.w3.org/TR/did-core
[47] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. D. Groot, and E. Lear, *Address Allocation for Private Internets*, document RFC1918, 1996.
[48] T. C. Schelling, *The Strategy Conflict*. Cambridge, MA, USA: Harvard Univ. Press, 1960.
[49] ShoCard. (Apr. 2020). *Shocard Whitepaper*. [Online]. Available: http://shocard.com/wp-content/uploads/2018/01/ShoCard-Whitepaper-Dec13-2.pdf
[50] Sovrin. (Apr. 2020). *Sovrin: Control Your Digital Identity*. [Online]. Available: https://sovrin.org/
[51] M. Sporny, D. Longley, and D. Chadwick. (Jan. 2021). *Verifiable Credentials Data Model 1.0: Expressing Verifiable Information on the Web*. [Online]. Available: https://www.w3.org/TR/vc-data-model
[52] P. Srisuresh and M. Holdrege, *IP Network Address Translator (NAT) Terminology and Considerations*, document RFC2663, 1999.
[53] O. Steele and M. Sporny. (Feb. 2021). *DID Specification Registries: The Inter Operability Registry for Decentralized Identifiers*. [Online]. Available: https://www.w3.org/TR/did-spec-registries/
[54] M. Takemiya and B. Vanieiev, "Sora identity: Secure, digital identity on the blockchain," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Oct. 2018, pp. 582–587.
[55] M. Toorani and C. Gehrmann, "A decentralized dynamic PKI based on blockchain," in *Proc. 36th Annu. ACM Symp. Appl. Comput.*, Mar. 2021, pp. 1646–1655.
[56] *EIDAS—Electronic Identification, Authentication and Trust Services*, European Union, Maastricht, The Netherlands, 2014.
[57] Uport-Project. (Feb. 2021). *Ethereum DID Registry—Ethereum Registry for ERC-1056 Ethr Did Methods*. [Online]. Available: https://github.com/uport-project/ethr-did-registry
[58] G. Urdaneta, G. Pierre, and M. V. Steen, "A survey of DHT security techniques," *ACM Comput. Surv.*, vol. 43, no. 2, pp. 1–49, Jan. 2011.
[59] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.

DIEGO PENNINO received the Ph.D. degree in computer science from Roma Tre University, in 2021. He is currently a Researcher in computer science with the Engineering Department, Roma Tre University. His research is involved in the field of integrity and security of the data, and in the field of blockchain and smartcontracts.

MAURIZIO PIZZONIA is currently an Assistant Professor with Roma Tre University, Italy. He is very much oriented in solving real problems and in devising innovative solutions that can lead to applications. In 2011, he co-founded a start up in cloud security. He has been teaching cybersecurity courses, since 2004. He has published more than 50 papers in scientific conferences and journals, and served as a coordinator for research units and work packages in two European projects. His research interests include design of algorithms and software systems for cybersecurity, blockchain, internet analysis, and information visualization.

ANDREA VITALETTI is currently an Associate Professor with the Sapienza University of Rome. He founded the spin-offs WLAB (sold in 2016) and WSENSE (left in 2019). He is also the Co-Founder of GIUSTA. He teaches networking, data management, and the IoT topics in engineering and product design for M.Sc. and B.Sc. students at Sapienza. He has coauthored more than 60 papers in scientific conferences and journals. He has been involved in a number of EU projects as a researcher and a principal investigator, and coordinated the FET Open PLEASED. His research interests include algorithms and protocols for wireless networks and the IoT, private and distributed data management, and distributed ledger technologies.

MARCO ZECCHINI received the M.S. degree in computer science engineering from the "La Sapienza" University of Rome, in 2019, where he is currently pursuing the Ph.D. degree in data science with the Department of Computer, Control and Management Engineering. His research is involved in the field of the Internet of Things, blockchain, and smart contracts.