information

Article

# Formalizing the Blockchain-Based BlockVoke Protocol for Fast Certificate Revocation Using Colored Petri Nets

**Anant Sujatanagarjuna [1,*,†], Arne Bochem [1,†] and Benjamin Leiding [2,†]**

[1] Institute of Computer Science, University of Goettingen, 37073 Goettingen, Germany; arne.bochem@cs.uni-goettingen.de
[2] Institute for Software and Systems Engineering, Clausthal University of Technology, 38678 Clausthal-Zellerfeld, Germany; benjamin.leiding@tu-clausthal.de
* Correspondence: a.sujatanagarjuna@stud.uni-goettingen.de
† These authors contributed equally to this work.

**Abstract:** Protocol flaws such as the well-known Heartbleed bug, security and privacy issues or incomplete specifications, in general, pose risks to the direct users of a protocol and further stakeholders. Formal methods, such as Colored Petri Nets (CPNs), facilitate the design, development, analysis and verification of new protocols; the detection of flaws; and the mitigation of identified security risks. BlockVoke is a blockchain-based scheme that decentralizes certificate revocations, allows certificate owners and certificate authorities to revoke certificates and rapidly distributes revocation information. CPNs in particular are well-suited to formalize blockchain-based protocols—thus, in this work, we formalize the BlockVoke protocol using CPNs, resulting in a verifiable CPN model and a formal specification of the protocol. We utilize an agent-oriented modeling (AOM) methodology to create goal models and corresponding behavior interface models of BlockVoke. Subsequently, protocols semantics are defined, and the CPN models are derived and implemented using CPN Tools. Moreover, a full state-space analysis of the resulting CPN model is performed to derive relevant model properties of the protocol. The result is a complete and correct formal BlockVoke specification used to guide future implementations and security assessments.

**Keywords:** colored petri nets; blockchain; certificate revocation; formal verification; security

check for updates

**Citation:** Sujatanagarjuna, A.; Bochem, A.; Leiding, B. Formalizing the Blockchain-Based BlockVoke Protocol for Fast Certificate Revocation Using Colored Petri Nets. *Information* **2021**, *12*, 277. https://doi.org/10.3390/info12070277

Academic Editor: Dung Duong

Received: 1 June 2021
Accepted: 2 July 2021
Published: 6 July 2021

## 1. Introduction

Design flaws, security and privacy issues, as well as incomplete specifications, pose risks not only to the direct users of a protocol but also to other stakeholders [1–4]. The Heartbleed bug [5] and SigSpoof [6] demonstrate the severe impact of such flaws in central components of the Internet that we rely on every day.

Formal methods, such as Petri Nets [7], Colored Petri Nets [8,9], $\pi$-calculus [10] and Communicating Sequential Processes [11], are utilized for the design, development, analysis and verification of new as well as existing protocols, hence eliminating, or at least minimizing, eventual design flaws [12,13].

Cohn-Gordon et al. [14] conducted a formal security analysis of the underlying protocol of the popular messaging application Signal (https://signal.org, accessed on 11 May 2021), thereby demonstrating several standard security properties of the protocol. Similarly, Fett et al. [15] analyzed the popular and widely deployed authorization/single sign-on (SSO) protocol OAuth 2.0 (https://oauth.net/2/, accessed on 11 May 2021) aiming to establish strong authorization, authentication and session integrity guarantees. In the process of doing so, the authors identified four attack vectors.

Colored Petri Nets in particular became popular for formalizing blockchain-based protocols, e.g., [16–18]. Both can be understood as state machines, and a mapping exists from blockchain data structures (tokens) to colored CPN tokens. Moreover, CPN-ML expressions are used to specify and implement data types and operations of the modeled system, which correspond to the functionalities of blockchain smart contracts. The authors of [19,20] formalized the Authcoin protocol [21] using CPNs and subsequently performed a risk and threat analysis using the ISSRM domain model [22] before mitigating identified risks using security risk-oriented patterns (SRPs) [23,24]. In addition, CPNs have also been used in the past for the specification of electronic business enactments and cross-enterprise collaborations [25].

While BlockVoke is described in depth in [26], a gap exists with respect to the creation process of the corresponding formal CPN model as well as analyzing and verifying the formal model. This work fills the detected gap and investigates the research question of how to formalize the BlockVoke protocol using Colored Petri Nets. In order to answer this question with a separation of concerns, the main research question is divided into three sub-questions: (1) What is the top-level of the formal CPN model of the BlockVoke protocol? (2) What are the required protocol semantics? (3) What are the refining CPN models?

The remainder of this paper is structured as follows. Section 2 provides supplementary literature as well as related works. Section 3 describes the BlockVoke protocol itself in detail. Section 4 focuses on the modeling strategy, while Section 5 details the protocol semantics of the formal BlockVoke model. Next, Section 6 presents the refined BlockVoke CPN models. Afterward, Section 7 presents an evaluation of BlockVoke based on the previously created CPN models and discusses our findings. Finally, Section 8 concludes this work and provides an outlook on future work.

## 2. Background and Related Works

Before going over the inner workings of the BlockVoke protocol, an overview of related techniques is given in this section.

### *2.1. Certificate Revocation Mechanisms*

Certificates usually have a specific lifetime after which they expire. Certain incidents, however, may make it necessary to invalidate a certificate before its validity expires, e.g., a compromised private key or a change in domain ownership may make it necessary to revoke a certificate to prevent attackers from performing MITM attacks or accessing encrypted data [27]. In such a case, the certificate authority (CA) that signed the certificate issues a revocation statement signed by its private key. Figure 1 shows the most common revocation mechanisms, which are briefly introduced in this section.

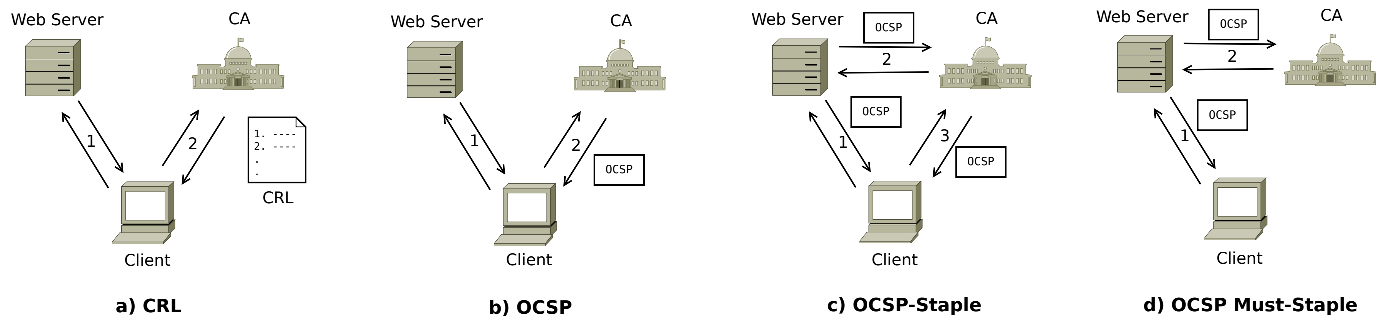


**Figure 1.** Certificate revocation (Based on [26–28]).

#### 2.1.1. Certificate Revocation Lists (CRLs)

CRLs are the most straightforward approach to distribute certificate revocations and are shown in Figure 1a. The CA provides a downloadable signed list of currently revoked certificate serial numbers [29]. Certificate expiration is determined solely by examining the certificate itself, so expired certificates are not listed in CRLs. The main drawback of this approach is that clients have to download the complete list to determine whether a specific certificate was revoked, which impacts both bandwidth and latency, thereby increasing costs for the CA and reducing responsiveness for the user. Additionally, as the whole list has to be updated, a delay between revocation and distribution is induced, and incidents where large amounts of revoked certificates are added to the list can cause the size of CRLs to grow noticeably.

#### 2.1.2. Online Certificate Status Protocol (OCSP)

OCSP allows the CA to provide the revocation status of singular certifications on request, as shown in Figure 1b. Thus, a client may inquire about the current validity of a certificate without having to download a full CRL [30]. Instead, the query is answered with a signed response, including the period of validity for the given certificate. This reduces the amount of transferred data, but it adds latency due to the additional request, and it also poses privacy issues, where CAs can track, for example, the web browsing behavior of clients through their OCSP requests and their IP address [28]. Furthermore, the issue of delays occurring between the revocation of a certificate and clients being notified about this revocation are also not solved by OCSP [31].

#### 2.1.3. OCSP Stapling

OCSP Stapling moves the responsibility of providing the user with certificate validity information to the server, which regularly requests a signed OCSP proof from the CA and bundles it with the certificate when sending it to the user [32]. This approach, as shown in Figure 1c, mitigates both the privacy and latency issue for the user. The user can verify the proof, which is valid only for short, pre-defined periods. If no stapled proof is provided, the user may fall back to the regular OSCP. Where such fallback behavior is not desired, the "Must Staple" extension, as shown in Figure 1d, may be employed to ensure that a connection can only be made when an OCSP response is bundled by the webserver, thus preventing attacks where an attacker disrupts a client's OCSP request while not bundling any OCSP data with a man-in-the-middle response.

### *2.2. Blockchain Technology*

Blockchains are a data structure of chained blocks containing transactional data and connected by including the previous block's hash in each block's header, as illustrated in Figure 2. Typical applications of blockchains are cryptocurrencies and smart contract platforms, such as Bitcoin [33] and Ethereum [34]. In these examples, blocks store cryptographically signed transactions with the whole blockchain, providing a public, decentralized, tamper-resistant ledger with ensured consistency and a global consensus among the network's peer-to-peer (P2P) nodes.

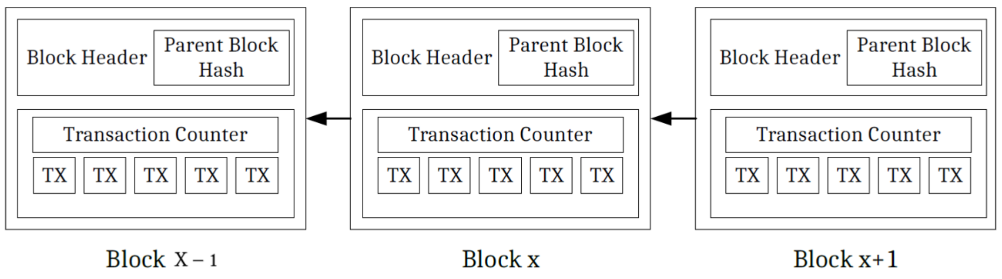


**Figure 2.** General blockchain structure (based on [33]).

The resulting data structure is decentralized, with a global consensus as defined by certain consensus mechanisms, such as proof of work in combination with the longest chain rule, where the blockchain with the highest accumulated work is considered the current, valid chain. The chaining of hashes also provides the property of immutability for past entries in the blockchain, as any modification would require the subsequent blocks to be redone, which in turn would require an attacker trying to modify the chain to redo all the necessary work faster than the rest of the network can add work to the actual main chain. There are also other consensus mechanisms, such as proof of stake, which handle this type of conflict in different ways. Additionally, the network is peer-to-peer (P2P)-based and open to participation by anyone. Security is provided by cryptographic signatures on transactional data in the blockchain.

### *2.3. Related Work*

Section 1 already discusses the work of Cohn-Gordon et al. [14], where the authors conducted a formal security analysis of the underlying protocol of the popular Signal messenger. It also mentions the work of Fett et al. [15], who analyzed the popular OAuth 2.0 scheme, demonstrating several standard security properties provided by the protocol.

Amoah et al. [35] identified a Denial-of-Service (DoS) attack against the DNP3 protocol as part of a security analysis using a CPN model. Duo et al. and others [16–18] utilized CPN models to formalize and verify blockchain-based protocols, smart contracts and blockchain-based applications. Leiding et al. [19,20] first formalized the Authcoin protocol [21] using CPNs and subsequently performed a risk and threat analysis using the ISSRM domain model [22] before mitigating identified risks using security risk-oriented patterns (SRPs) [23,24]. Finally, Bochem and Leiding [36] used CPNs to verify various properties of the blockchain-based Rechained protocol formally.

The research above exemplifies that formal analysis and verification facilitates the development and implementation of secure protocols, prevents incomplete specifications, demonstrates specific protocol properties and identifies and mitigates security and privacy issues identified through a subsequent security analysis of the formal models.

## 3. BlockVoke Protocol

BlockVoke provides a nearly instantaneous way of publishing revocations for any stakeholder who is interested in receiving them. It also ensures that published revocations remain available by being added to a blockchain. Another main point of consideration is that existing network effects, especially with regards to securing existing blockchains, can be leveraged by BlockVoke as it is not only compatible with existing blockchains such as the Bitcoin and the Ethereum blockchains but more general even blockchain-agnostic. An overview of the certificate lifecycle with BlockVoke is presented in Figure 3 and explained in detail in the subsequent sections.

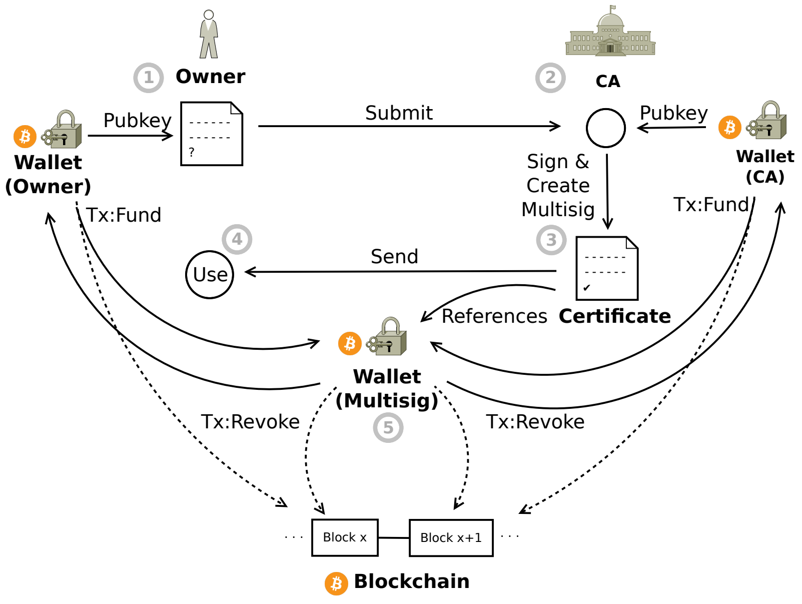


**Figure 3.** The BlockVoke certificate lifecycle (Based on [26]).

It is worth noting that the revocation of CA root certificates through BlockVoke is also supported. Usually, this type of certificate is difficult to revoke due to being self-signed.

However, the same technique as described below for regular certificate revocation using BlockVoke can be applied to CA root certificates as well.

### *3.1. Certificate Signing Request (CSR)*

The first step necessary to create an SSL/TLS certificate is creating a CSR, which is sent to the CA to create and sign the certificate. In BlockVoke, an additional attribute is added to the CSR that contains the public key of a Bitcoin address, which is controlled by the certificate owner (CO). The use of other blockchain-based platforms than Bitcoin with BlockVoke is possible as well. For the sake of simplicity, in the following, we however refer only to Bitcoin.

### *3.2. Certificate Creation*

When signing the certificate, a CA-controlled Bitcoin address is used in conjunction with the address provided in the CSR and combined into a 1-of-2 multi-signature address, which is then noted in an extension field of the certificate. To better support the combination of BlockVoke with the Certificate Revocation Vectors (CRV) [27] approach, a CA may also add the public key used to create the multi-signature address to the extension field. The signed certificate is used as any regular certificate by the certificate owner.

### *3.3. Revocation*

To check whether a certificate has been revoked using BlockVoke, users check if a valid BlockVoke transaction has been sent from the multi-signature address specified in the certificate's extension field. If a valid BlockVoke transaction has originated from this address, the certificate is considered revoked.

Due to the use of a multi-signature address, the CO and CA can revoke the certificate. To do so, a Bitcoin transaction is created that sends a small amount from the multi-signature address to any arbitrarily chosen other address after first funding the multi-signature address with the necessary funds in a separate transaction. The revocation transaction contains an additional output using Bitcoin's `OP_RETURN` script opcode. This opcode may contain a payload of up to 40 bytes. To make BlockVoke transactions easy to detect, the payload has to follow the structure given in Table 1. A Bitcoin revocation transaction with one input and one output plus the full `OP_RETURN` payload has an estimated size of 283 Bytes.

**Table 1.** `OP_RETURN` payload of a BlockVoke transaction.

| Offset | Length | Content |
|---|---|---|
| 0 | 10 | `BlockVoke` |
| 10 | 16 | First 16B of certificate fingerprint |
| 26 | 4 | Date of issuance in days since 2020-02-02 (uint32) |
| 30 | 1 | Revocation reason code according to RFC 5280 [29] (uint8) |
| 31 | 3 | Optional: If CA uses CRV, uint24 of revocation number RN |
| 34 | 6 | Optional: If CA uses CRV, unique CA identifier |

The included fingerprint allows users to verify that the revocation of a given certificate was intended. In addition, two optional fields enable the use of BlockVoke together with the CRV approach. The CRV-specific information is only added to the payload if the CA performs the BlockVoke revocation and if the CA supports CRV. In this case, a user processing the BlockVoke revocation confirms that the CA sent the transaction by matching the transaction's signature against the CA public key specified inside the certificate.

Transactions are broadcast instantly over Bitcoin's P2P network, in effect delivering BlockVoke revocation transactions to users. Consensus is not necessary for the revocations to become valid, so simply being distributed in the mempool of unconfirmed transactions is enough for users to be notified about the revocation. Once the transaction is mined, it will additionally be persistent and easy to look up for any user with access to the blockchain.

Entities with requirements for low latency revocation notifications or building CRLite [37] filters at higher rates may also use the rapidly distributed BlockVoke revocation information to update their filters.

## 4. Modeling the BlockVoke Protocol

Formalizing the BlockVoke protocol requires an appropriate modeling strategy, mapping the existing descriptions of the protocol, as outlined in Section 3, to the corresponding elements of a CPN model [8,9]. The resulting sound model allows the consideration of concurrency conflicts, the prevention of dependability issues and the detection and mitigation of design flaws.

Based on Petri Nets and introducing colored tokens, CPNs allow the construction, specification and simulation of formal models for concurrent systems in a graphical way. CPN models describe systems through states, where events or transitions allow the state of the system to change [38]. Most commonly, "CPN models are represented using a directed bipartite graph that consists of places, transitions, arcs and tokens. Places are denoted as circles and transitions as rectangles. Arcs connect places with transitions, or transitions with places and have inscriptions given as CPN-ML expressions [8,9,25,39,40]. CPN-ML is an expression programming language for inscriptions which are used to further specify data types and operations of the modeled system. CPN tokens and their colors represent the different data types of the modeled system" [36].

Besides their general suitability for system formalization, CPNs are especially well fit to be applied in the context of blockchain-based systems and protocols. Blockchains can be understood as discrete state machines, each block representing a state of the system. Adding a new block to the chain transitions the system from one state to a succeeding state. Likewise, CPNs model discrete state machines, changing states via transitions. "While in CPNs, data structures are represented in the form of colored tokens, many blockchain platforms also use a data structure concept of tokens for the same reason. Moreover, blockchain transactions can be easily mapped to CPN token data structures. Furthermore, CPNs use CPN-ML expressions to specify and implement data types and operations of the modeled system, which correspond to the functionalities of smart contracts in the context of blockchain technology. Finally, the hierarchical structure of CPN models can be used to formalize blockchain-based dApps (decentralized applications) components of interleaved smart contracts" [36].

In the following, CPN-Tools (http://cpntools.org, accessed on 14 May 2021) is used to design, evaluate and verify the BlockVoke CPN model.

### *4.1. Modeling Strategy*

Mahunnah et al. [41] described mapping heuristics from agent models to CPN models based on [42] socio-technical requirements—the engineering methodology of agent-oriented modeling. Thus, the first step is to create a hierarchical agent-oriented goal model, which is used to document the functional and non-functional goals of BlockVoke. Then, the goal models are further refined using behavioral interface modeling, which adds more detail to the activities performed by different agents in BlockVoke.

Subsequently, the CPN model of BlockVoke is created using CPN Tools [9]. "During the enactment of a CPN model, flow of control passes to the sub-goals or activities (in the AOM equivalent) associated with a parent goal represented as a module. This way, a CPN model represents a hierarchical structure of the goal model in AOM" [41]. The behavioral interface model describes the activities of the protocol by identifying the triggers, preconditions and postconditions of the activities. The activities detailed in the behavioral interface model are mapped to the transitions of the CPN model. The triggers, preconditions and postconditions are mapped by adding appropriate places and guards to the CPN model between the transitions. This mapping is explained in more detail in Section 4.4.

Sections 4.2 and 4.3 describe and give the result of these preliminary AOM tasks for creating the CPN model of BlockVoke, namely agent-oriented goal modeling and behavioral interface modeling.

### *4.2. AOM Goal Modeling*

Good requirements in software engineering and system development should fulfill certain criteria given in [43,44]. Each requirement should address no more than one issue and it should be fully specified, including all necessary information. Additionally, requirements shall be consistent and non-contradictionary both in themselves and with respect to other requirements. The final criteria that must be fulfilled are to be atomic and without conjunctions [45].

The AOM methodology allows technical and non-technical stakeholders to model complex systems by capturing and understanding their functional and non-functional requirements. A goal model describes the goal hierarchy of the system to be developed, starting with the purpose of the system [42]. Goal models generally contain three main components, as illustrated in Figure 4: 'Functional goals' are depicted as parallelograms, 'quality goals' are depicted as clouds and 'roles' are represented by stick figures. Functional goals satisfy a functional requirement of a system. Quality goals satisfy a non-functional requirement of a system, the attainability of which is usually ascertained using subjective reasoning. A role is some responsibility or position fulfilled by an actor or an 'agent' that can interact with the system. "The goal model follows a tree-like hierarchy with the root value proposition of the modeled system at the top. Next, the main goal is decomposed into sub-goals, where each sub-goal represents an aspect for achieving its parent goal [46]. Finally, goals are further decomposed into multi-layered sub-goals until the lowest atomic level is reached. Additionally, roles and quality goals may be assigned to goals and are inherited to lower-level goals" [36].

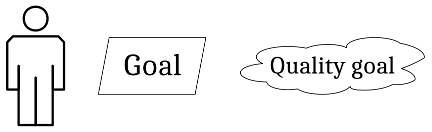


**Figure 4.** Selection of AOM notation elements.

The top-level goal model of BlockVoke is shown in Figure 5, while the complete goal model of BlockVoke with all refinements is available in Appendix A. The main objective of BlockVoke is to enable secure, timely and privacy-preserving certificate revocation. The value proposition is further divided into three sub-goals: 'Generate Certificate', 'Verify Certificate' and 'Revoke Certificate'. The two quality goals 'correct' and 'reliable' are attached to the main goal, thus being relevant and inherited to all sub-goals. Seven additional quality goals are attached to the sub-goals: 'tamper-resistant', 'secure', 'scalable', 'timely', 'cost-effective', 'available' and 'efficient'. In addition, we list three different roles: the 'End User', the 'CA' and the 'CO'.

### *4.3. Behavioral Interface Modeling*

A behavioral interface model identifies behavioral units and defines an interface for each behavioral unit [42]. A behavioral unit is an activity that is performed as part of one of the roles played by an agent described in the goal model. The behavioral interface identifies the triggers, preconditions and postconditions associated with each activity. A trigger is an event or a set of events that trigger the activity. Preconditions are sets of necessary conditions that must be fulfilled for the activity to proceed. Preconditions may be non-existent, indicating spontaneous triggering of the activity. Postconditions are a set of conditions that must be fulfilled to denote the successful proceeding of that activity.

The top-level behavioral interfaces of BlockVoke corresponding to the top-level AOM goal model of Figure 5 are shown in Table 2.

**Table 2.** Top-level behavioral interface model of BlockVoke.

| Activity | Trigger | Precondition(s) | Postcondition(s) |
|---|---|---|---|
| Generate Certificate | CA wants to generate CO's certificate | CO's CSR with information relevant to the certificate, CO's (wallet) public key, CA's signing key pair, CA's (wallet) public key | Generated certificate with CA's signature and multi-signature address ready to be verified by the end user |
| Verify Certificate | Certificate ready to be verified by the end user | Generated certificate, CA's public key | Certificate has been verified by an end user |
| Revoke Certificate | CA or CO wants to revoke a certificate that they have signed/own respectively | Crypto wallet with small credit amount, signed certificate, certificate verified, RFC 5280 code, optional CA identifier | Certificate has been revoked |

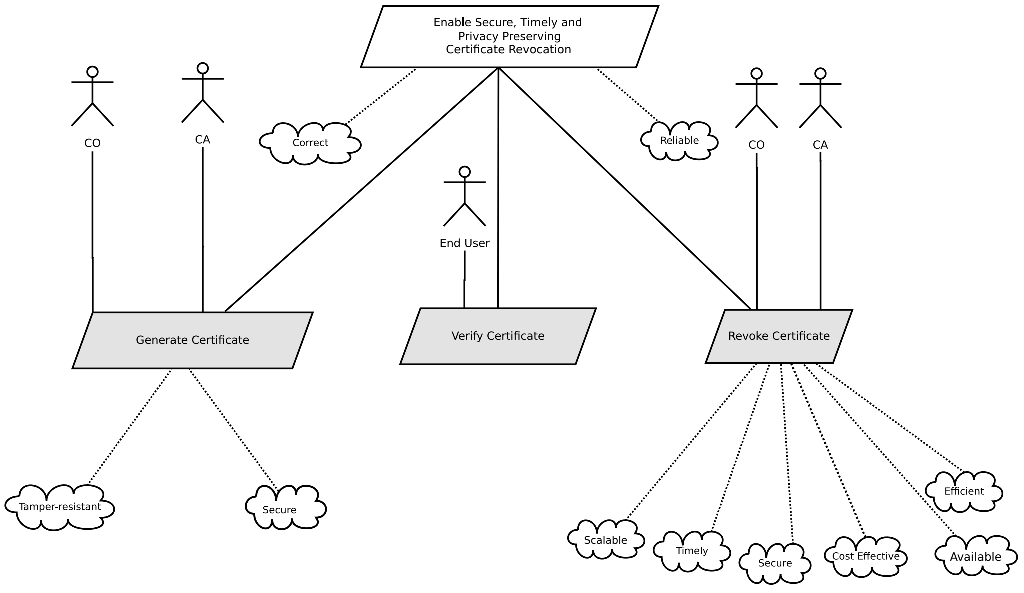


**Figure 5.** Top-level goal model of BlockVoke.

The 'Generate Certificate' activity is triggered after providing the required CSR, the CO's public wallet key, the CA's signing key and the CA's public wallet key. Next, the generated certificate is verified as part of the 'Verify Certificate' activity, which takes the certificate and the CA's public key as an input. Afterward, the certificate can be revoked via the 'Revoke Certificate' activity. To do so, the prefunded wallet, the signed certificate, the RFC code and the optional CA identifier are required. The resulting postcondition is a revoked certificate.

### *4.4. Mapping the Agent-Oriented Model to CPN Models*

At last, we map the AOM goal model and behavioral interface model of BlockVoke to its CPN representation resulting in the CPN model, as shown in Figure 6. Table 3 illustrates the mapping of agent-oriented modeling to CPN models. A transition can be used to represent sub-goals or activities. CPN hierarchical modules illustrate goals derived from the CPN model. Places with outgoing arcs represent triggers or postconditions. Places with incoming arcs represent preconditions. The notation used is adapted from Mahunnah et al. [41] while also adding guards since they can also be used to represent preconditions for an activity or a transition. Figure 7 specifies the mapping of behavioral interfaces to CPN.

Finally, Figure 6 shows the top-level CPN model of BlockVoke, which contains the three top-level goals of the AOM goal model, namely 'Generate Certificate', 'Verify Certificate' and 'Revoke Certificate'. The places 'CA KeyPair', 'CA Wallet','CO KeyPair' and 'CO Wallet' exist to produce the respective color sets necessary for the simulation. 'CO KeyPair' and 'CA KeyPair' are the CO's key pair, used for creating a certificate signing request, while the CA's key pair is used for signing the CO's certificate. The places 'CA wants to Generate Certificate','CO wants to create CSR', 'CO wants to revoke', 'CA wants to revoke', 'Certificate Verified' and 'Certificate ready to be verified by End User' are all mappings from the agent-oriented modeling. Further refinements of the formalized CPN model can be found in the complete hierarchical CPN model in Appendix A.

The initial token markings are shown in Table 4.

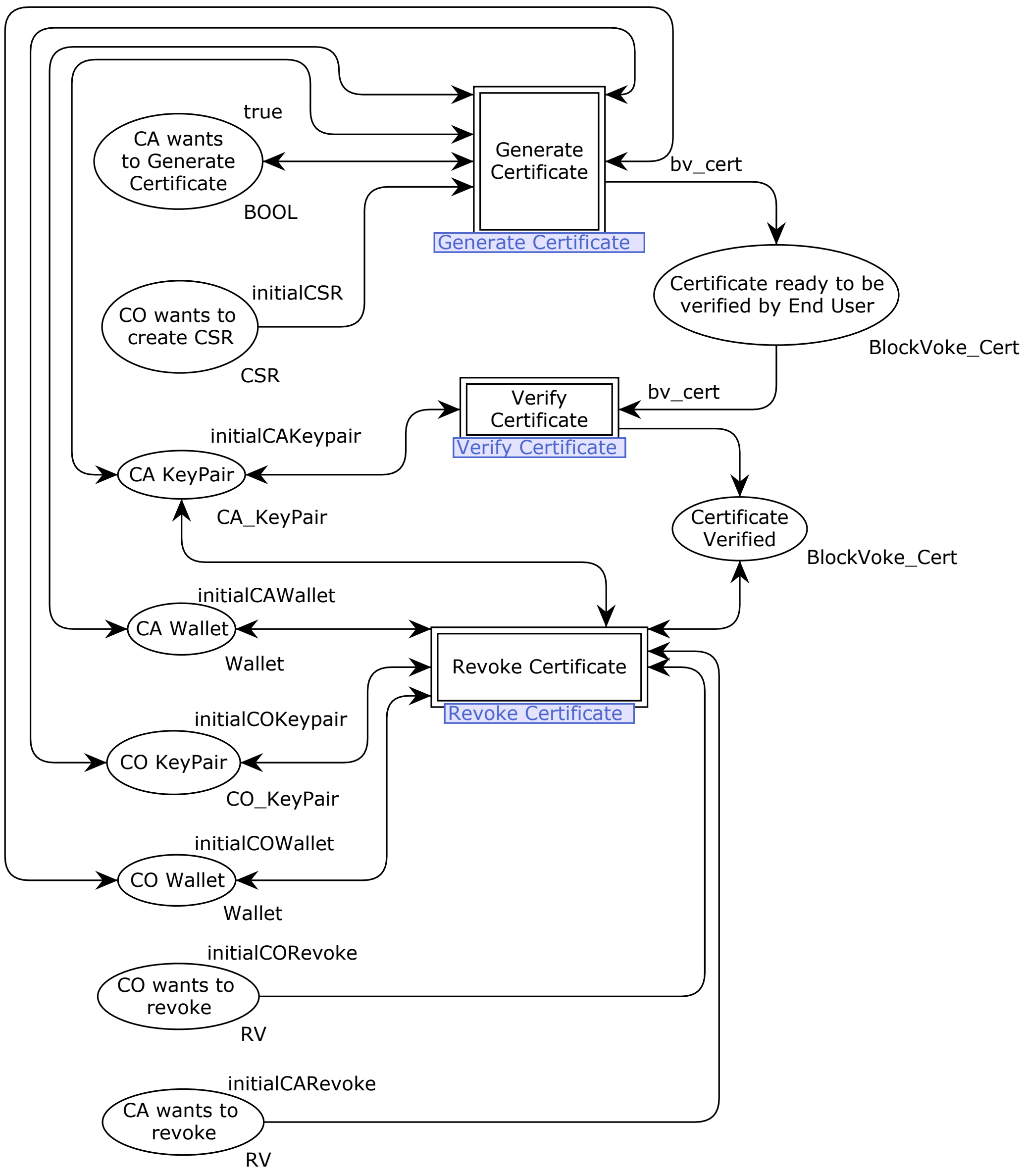


**Figure 6.** Top-level CPN model of Blockvoke.

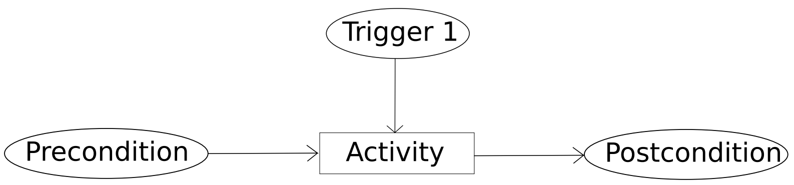


**Figure 7.** Mapping a behavioral interface to a CPN model (Adapted from [41]).

**Table 3.** Notation for mapping AOM to CPN (Adapted from [41]).

| Notation | Name |
|---|---|
| ⟶ | Connecting arc |
| ▭ | Sub-goal or activity |
| ⬭⟶ | Trigger or precondition |
| ⟶⬭ | Postcondition |
| ⧈ | Goal |
| `[<conditions>]` | Precondition(s) |

**Table 4.** Initial markings of the top-level CPN model of BlockVoke.

| Value Name | Value Declaration |
|---|---|
| initialCAWallet | `1‘(“0x1”, (“ca1_wallet_pubkey”, “ca1_wallet_privkey”), “0x001”, 100)` |
| initialCAKeypair | `1‘(“CA1”, (25877, 5), (25877, 20429), “ca1”)` |
| initialCOWallet | `1‘(“0x3”, (“co1_pubkey”, “co1_privkey”), “0x003”, 100)++`<br>`1‘(“0x4”, (“co2_pubkey”, “co2_privkey”), “0x004”, 100)++`<br>`1‘(“0x5”, (“co3_pubkey”, “co3_privkey”), “0x005”, 100)++`<br>`1‘(“0x6”, (“co4_pubkey”, “co4_privkey”), “0x006”, 100)` |
| initialCOKeypair | `1‘(“www.co1-website.com”, (33017, 7), (33017, 4663), “co1_website”)++`<br>`1‘(“www.co2-website.com”, (83767,13), (83767,6397), “co2_website”)++`<br>`1‘(“www.co3-website.com”, (69451,5), (69451,13781), “co3_website”)++`<br>`1‘(“www.co4-website.com”, (50299,3), (50299,33227), “co4_website”)` |
| initialCSR | `1‘((“www.co1-website.com”, (33017, 7), “co1_website”, “CA1”, (25877, 5), “ca1”, “02/02/2021”, “02/02/2022”, “”, 0, 0, “”), “0x3”)++`<br>`1‘((“www.co2-website.com”, (83767, 13), “co2_website”, “CA1”, (25877, 5), “ca1”, “03/03/2021”, “03/03/2022”, “”,0, 0,“”), “0x4”)++`<br>`1‘((“www.co3-website.com”, (69451, 5), “co3_website”, “CA1”, (25877, 5), “ca1”, “04/04/2021”, “04/04/2022”, “”, 0, 0,“”), “0x5”)++`<br>`1‘((“www.co4-website.com”, (50299, 3), “co4_website”, “CA1”, (25877, 5), “ca1”, “05/05/2021”, “05/05/2022”, “”, 0, 0,“”), “0x6”)` |
| initialCORevoke | `1‘(1803, false, 10, “0x4” , 1, “unused”, “0xmultisig2”, “12.02.2021”, “ca1”)` |
| initialCARevoke | `1‘(1825, true, 10, “0x1”, 1, “cACompromise”, “0xmultisig4”, “12.02.2021”, “ca1”)` |

## 5. Protocol Semantics

CPNs rely on so-called token colors, representing data structures of the modeled system, thereby representing the flow of data and corresponding objects throughout the protocol execution. This section briefly outlines the CPN token color sets, names and acronyms used in the BlockVoke CPN model. CPN transitions transfer and manipulate the processed tokens and ensure that the data objects adhere to the defined data syntax. The acronyms, names and token colors of BlockVoke's top-level CPN model are presented in Table 5. The first column specifies the module of the first occurrence of a certain acronym, name or token color, while the second column specifies its name. The third column details the data type and structure. The last column provides a short description. A complete listing is available in Appendix A.

**Table 5.** Acronyms, names and token colors of BlockVoke's top-level CPN model.

| **Module** | **Token color** | **Type** | **Description** |
|---|---|---|---|
| Top-level | CSR | Color Set (BlockVoke_Cert, Wallet_Addr) | Representation of a certificate signing request |
| Top-level | CA_KeyPair | Color Set (CA_CN, CA_PublicKey, CA_PrivateKey, CA_Key_ID) | CA's RSA keypair |
| Top-level | Wallet | Color Set (Wallet_Addr, Wallet_KeyPair, Wallet_Previous_Hash, Wallet_Balance) | A wallet |
| Top-level | CO_KeyPair | Color Set (CO_CN, CO_PublicKey, CO_PrivateKey, CO_Key_ID) | CO's RSA keypair |
| Top-level | RV | Color Set (Cert_Fingerprint, Is_CA, Funds, Wallet_Addr, Fees, RFC5280_RevocationCode, Cert_Multisig_addr, Cert_DOI, CA_Key_ID) | Color Set with information required for a revocation |
| Top-level | BlockVoke_Cert | Color Set (CO_CN, CO_PublicKey, CO_Key_ID, CO_CN, CO_PublicKey, CO_Key_ID, Cert_Valid_From, Cert_Valid_To, Cert_Multisig_addr, Cert_Sig, Cert_Fingerprint, Cert_DOI) | SSL certificate with extra BlockVoke fields |

## 6. Refined CPN Models of BlockVoke

This section describes the sub-modules of the top-level CPN model of BlockVoke, as presented in Figure 6.

### *6.1. Generate Certificate*

Figure 8 shows the 'Generate Certificate' sub-module of BlockVoke's top-level CPN model. For simulation purposes, the CA is always willing to generate a certificate for the CO.

The certificate generation process is initiated by the CO, who creates a CSR. The CO then send this CSR, including their Bitcoin wallet address, to the CA. Next, the CA verifies the CO's identity. In the CPN model, this process is purely symbolic. Following this, the CA generates a 1-of-2 multi-signature address using the CO's wallet address and their own wallet address. CPN Tools do not support this kind of dynamic wallet generation. Instead, a specific multi-signature address for every CA is returned. Along with this, the CA also sets the date of issuance (DOI) and the generated multi-signature address in the certificate.

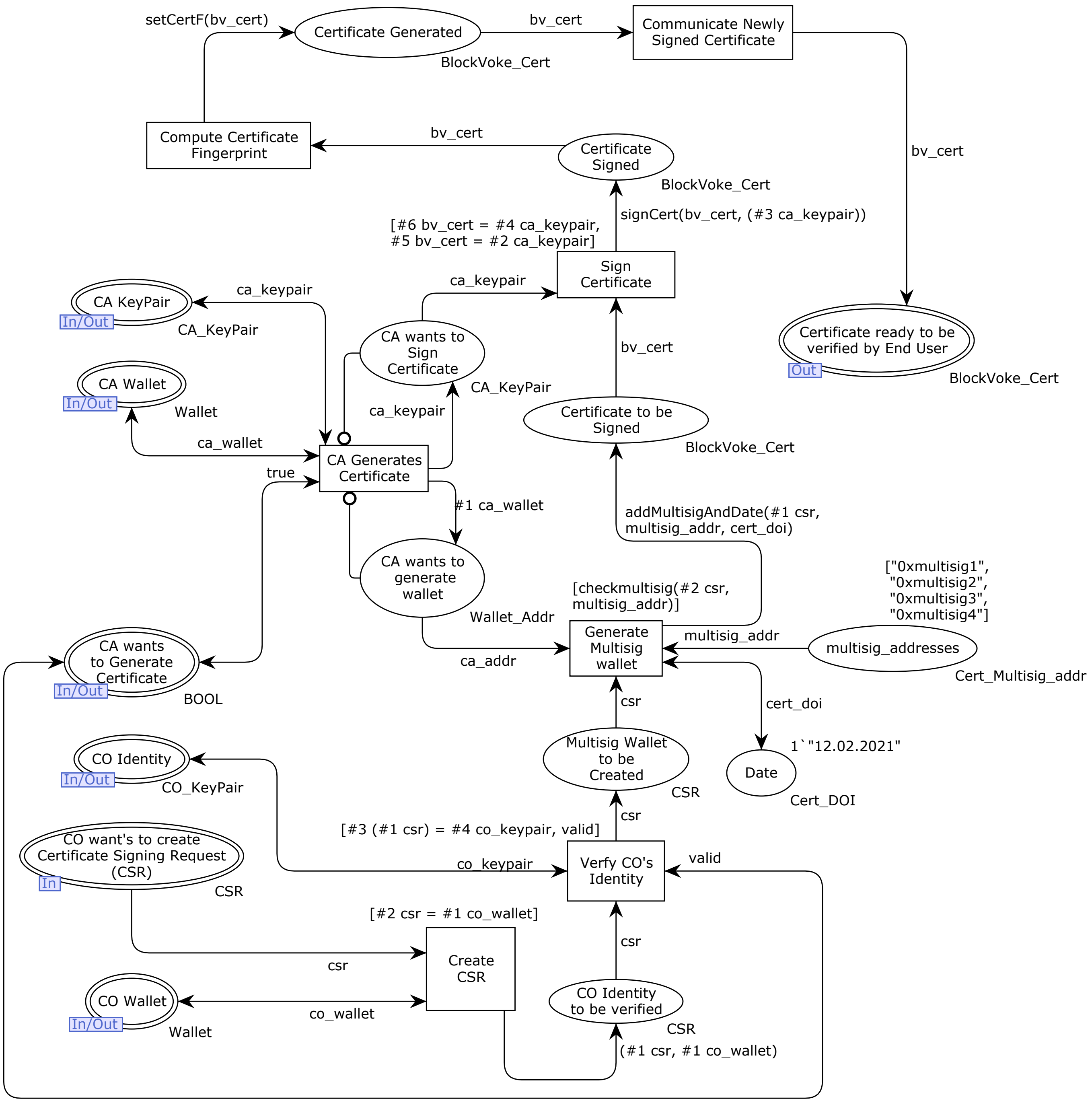


**Figure 8.** 'Generate Certificate' sub-module of the BlockVoke CPN model.

Next, the CA signs the certificate and computes the fingerprint. A salient feature of BlockVoke's CPN model is that it simulates the digital signing process in the form of RSA signatures. We implement a simplified modular hash computation in Simple ML, adapted from Fourman [47]. This enables us to compute RSA signatures using simple modular exponentiation, and similarly also to verify signatures that are computed in such a way. It should be noted that the computation of signatures and hashes is not cryptographically secure. They only exist to fulfill the purposes of the simulation. More information about our specific implementation of these operations can be found in the complete CPN model, as listed in Appendix A.

Once the CA signs the certificate, the certificate fingerprint is computed using the modular hashing technique. Next, this fingerprint is used to identify the certificate uniquely. Following this process, the certificate is in the 'Certificate ready to be verified by End User' place.

## 6.2. *Verify Certificate*

The 'Verify Certificate' sub-module is presented in Figure 9 and models the certificate verification that an end-user performs. First, the certificate's fingerprint is checked using the modular hashing method described in the previous section. Next, the CA's public key is used to verify the RSA signature in the certificate. Afterward, the certificate is in the 'Certificate Verified' place.

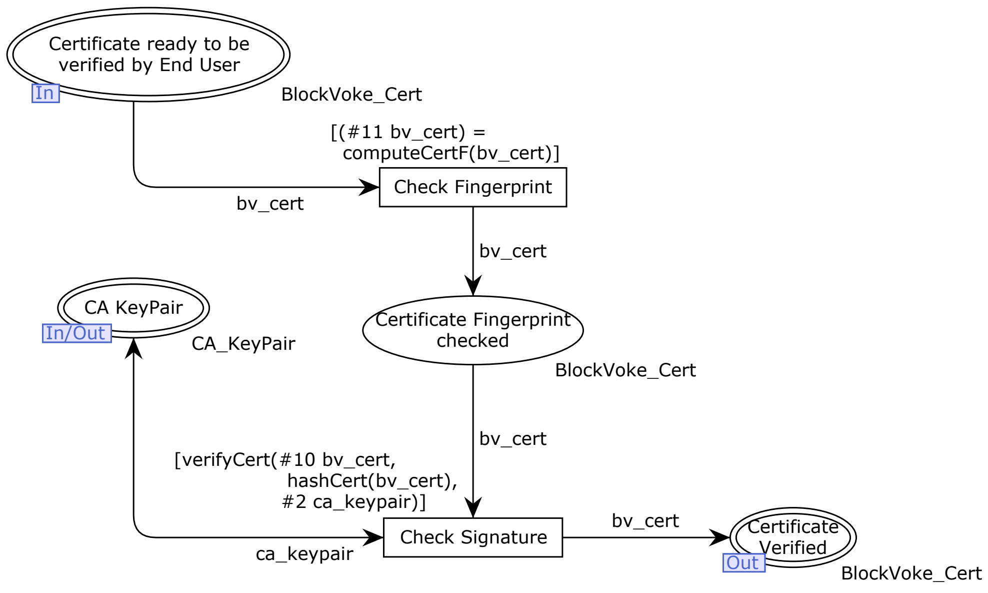

Certificate ready to be verified by End User
In
BlockVoke_Cert
[(#11 bv_cert) = computeCertF(bv_cert)]
Check Fingerprint
bv_cert
bv_cert
CA KeyPair
In/Out
CA_KeyPair
Certificate Fingerprint checked
BlockVoke_Cert
bv_cert
[verifyCert(#10 bv_cert, hashCert(bv_cert), #2 ca_keypair)]
Check Signature
ca_keypair
bv_cert
Certificate Verified
Out
BlockVoke_Cert

**Figure 9.** 'Verify Certificate' sub-module of the BlockVoke CPN model.

## 6.3. *Revoke Certificate*

Figure 10 illustrates the 'Revoke Certificate' sub-module of the CPN model and models the revocation process of BlockVoke. The Revocation process is initiated by either a CO or a CA. Following this, first, the revocation transactions are created in the 'Create Revocation Transactions' sub-module. The CPN model makes sure that any other revocation process cannot be initiated until the transaction pair is ready.

Once a transaction pair is created, they are 'Sent' to the 'Add Unconfirmed Revocation Transaction to Mempool' sub-module. This sub-module scrutinizes the revocation transactions and adds them to the 'Mempool'. This simulates the process by which unconfirmed transactions are accepted to the mempool of the Bitcoin network.

Following this, revocation transactions can be selected out of the 'Mempool Transactions' Place and marked as revoked in the 'MarkCertificate as Revoked' sub-module. This process will add the respective certificate to the 'Certificate Revoked' place. This process enables the certificate to be marked as revoked as soon as it is accepted into the mempool by the peers participating in the underlying network.

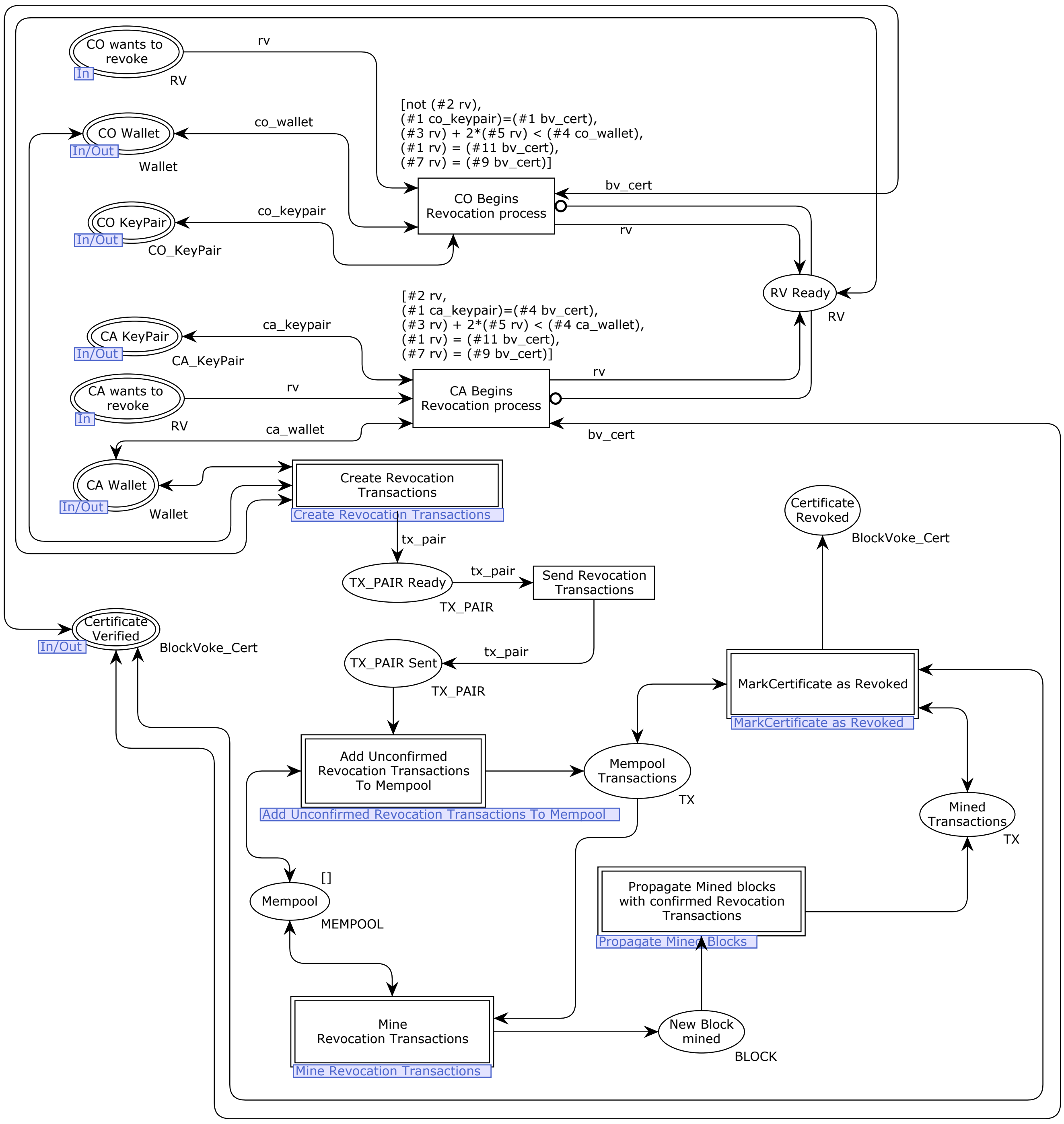


**Figure 10.** 'Revoke Certificate' sub-module of the BlockVoke CPN model.

Alternatively, the transactions in the mempool can be mined into blocks first, in the 'Mine Revocation Transactions' sub-module, and, once these new blocks are propagated on the blockchain's network, in 'Propagate Mined Blocks', they can be selected from the 'Mined Transactions' place and marked as revoked.

### 6.3.1. Create Revocation Transactions

The 'Create Revocation Transactions' sub-module of Figure 11 is composed of two further sub-modules, the 'Create Tx:Fund Transaction' and 'Create Tx:Revoke Transaction' sub-modules.

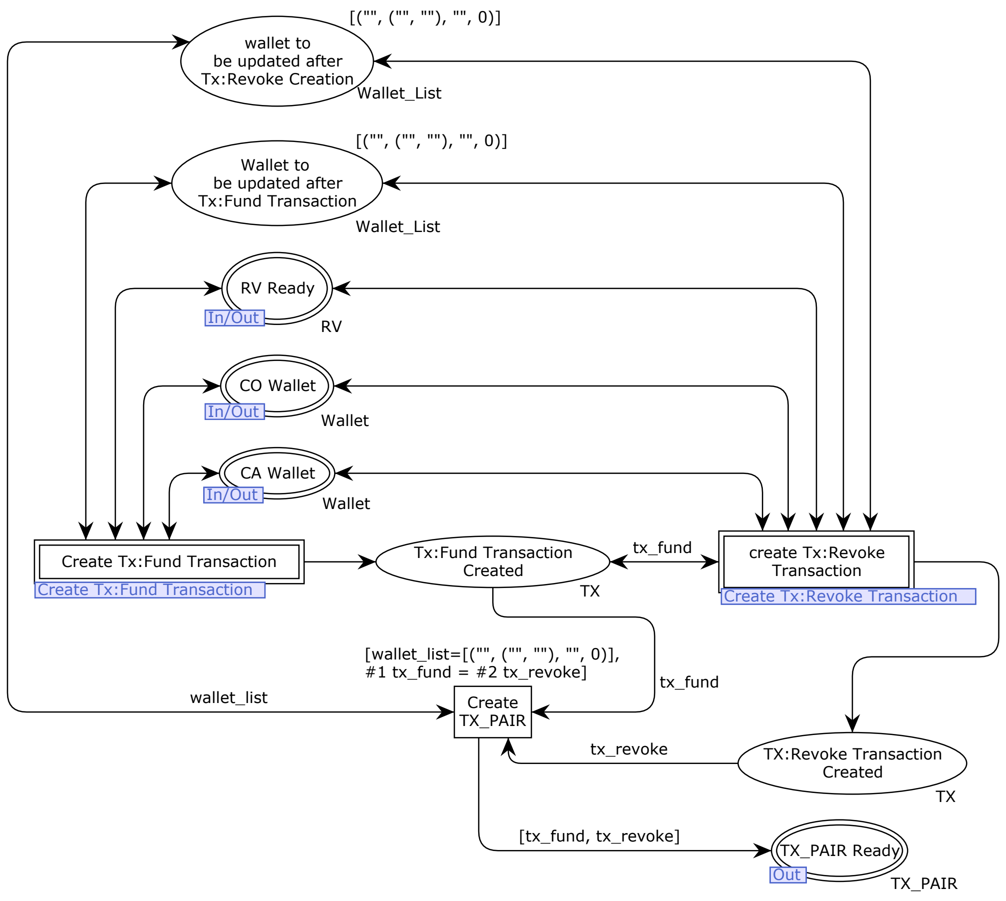

[("", ("", ""), "", 0)]
wallet to be updated after Tx:Revoke Creation
Wallet_List
[("", ("", ""), "", 0)]
Wallet to be updated after Tx:Fund Transaction
Wallet_List
RV Ready
In/Out
RV
CO Wallet
In/Out
Wallet
CA Wallet
In/Out
Wallet
Create Tx:Fund Transaction
Create Tx:Fund Transaction
Tx:Fund Transaction Created
TX
tx_fund
create Tx:Revoke Transaction
Create Tx:Revoke Transaction
[wallet_list=[("", ("", ""), "", 0)], #1 tx_fund = #2 tx_revoke]
tx_fund
wallet_list
Create TX_PAIR
tx_revoke
TX:Revoke Transaction Created
TX
[tx_fund, tx_revoke]
TX_PAIR Ready
Out
TX_PAIR

**Figure 11.** ‘Create Revocation Transactions’ sub-module of the BlockVoke CPN model.

The ‘Create Tx:Fund Transaction’ sub-module shown in Figure 12, as the name suggests, creates the funding transaction using the revocation information in the ‘RV Ready’ place and updates the ‘Wallet’ token of the revoking party by deducting the appropriate amount, and also updates the ‘Wallet_Previous_Hash’ component of their ‘Wallet’ token. It is important to note that the ‘Prepare Funds’ transition makes sure that the amount sent to the multi-signature address includes the fees that would be spent for the transaction in the ‘Create Tx:Revoke Transaction’ sub-module.

Once this is completed, the revoking transaction is created similarly in the ‘Create Tx:Revoke Transaction’ sub-module, which spends the funds received in the funding transaction back to the sender’s address, as shown in Figure 13. For the sake of simplicity, the model assumes that the funding amount for each revocation is ten credits, and the fees used are one credit each. Hence, once both transactions are created, the corresponding ‘Wallet_Balance’ is decreased by two.

Most of the complexity of the ‘Create Tx:* Transaction’ sub-modules is due to having to choose the correct ‘Wallet’ of the CO or the CA to be used for the revocation based on the given revocation information and then updating the same wallet by either deducting the amount in the ‘Create Tx:Fund Transaction’ sub-module or crediting the amount in the ‘Create Tx:Revoke Transaction’ sub-module while also updating the ‘Wallet_Previous_Hash’ in both cases. It should also be noted that, for the sake of simplicity, only one revocation from either the CO or the CA is processed by the ‘Create Revocation Transactions’ sub-module at a time. In other words, no other revocation can enter this module until a ‘TX_PAIR’ is created for the given current revocation.

Following the successful creation of both revocation transactions, the transactions are encapsulated in a ‘TX_PAIR’ token, which is placed in the ‘TX_PAIR’ place.

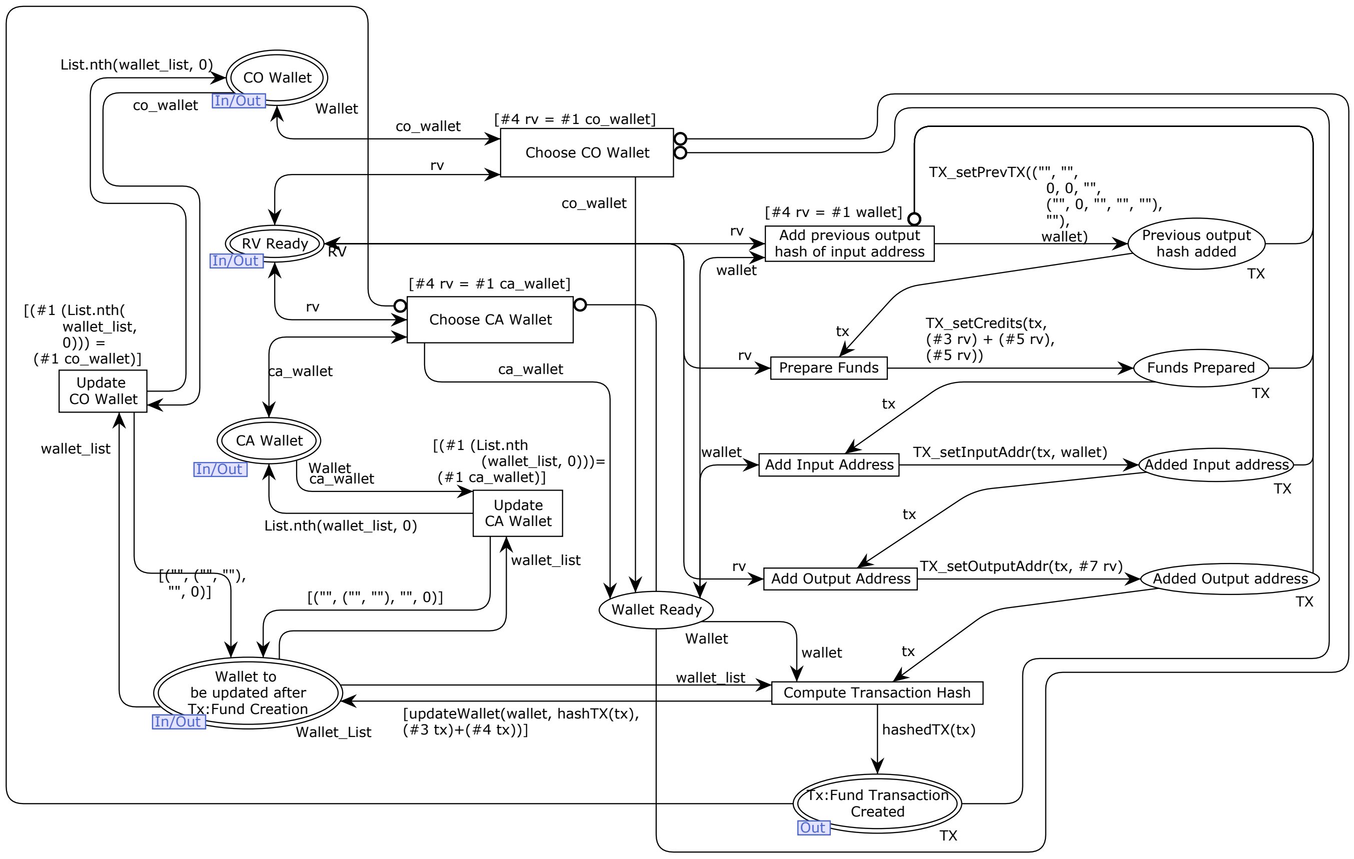
### 6.3.2. Add Unconfirmed Revocation Transactions to Mempool

As shown in Figure 14, the encapsulated revocation transactions 'TX_PAIR' created in the 'Create Revocation Transactions' sub-module is scrutinized in the 'Add Unconfirmed Revocation Transactions' sub-module by checking whether the amount spent in the funding transaction is at least as much as the sum of the amount spent and the fees required by the revoking transaction.

Afterward, both transactions are added to the 'Mempool Transactions' place and appended to the 'Mempool' Place. This purpose of having two places with the same transactions is due to the differing requirements in the 'Mark Certificates as Revoked' and 'Mine Revocation Transactions' sub-modules. Despite this fact, the list of 'TX' tokens in the 'Mempool' place and the 'TX' tokens in the 'Mempool Transactions' place are kept synchronized, as transactions are confirmed in later parts of the simulation.

### 6.3.3. Mine Revocation Transactions

The 'Mine Revocation Transactions' sub-module of Figure 15 presents the mining process of the revocation transactions. The first step is to choose the transactions from the 'Mempool'. Care is taken to synchronize this operation with both the 'Mempool' and 'Mempool Transactions' places. Without loss of generality, exactly two transactions are chosen for mining.



CO Wallet
In/Out
Wallet
List.nth(wallet_list, 0)
co_wallet
[#4 rv = #1 co_wallet]
Choose CO Wallet
rv
RV Ready
RV
Choose CA Wallet
[#4 rv = #1 ca_wallet]
ca_wallet
CA Wallet
Update CO Wallet
Update CA Wallet
[(#1 (List.nth( wallet_list, 0))) = (#1 co_wallet)]
[(#1 (List.nth (wallet_list, 0)))= (#1 ca_wallet)]
wallet_list
List.nth(wallet_list, 0)
[("", ("", ""), "", 0)]
Wallet to be updated after Tx:Fund Creation
Wallet_List
Wallet Ready
TX_setPrevTX(("", "", 0, 0, "", ("", 0, "", "", ""), ""), wallet)
[#4 rv = #1 wallet]
Add previous output hash of input address
Previous output hash added
TX
tx
TX_setCredits(tx, (#3 rv) + (#5 rv), (#5 rv))
Prepare Funds
Funds Prepared
Add Input Address
TX_setInputAddr(tx, wallet)
Added Input address
Add Output Address
TX_setOutputAddr(tx, #7 rv)
Added Output address
Compute Transaction Hash
wallet_list
[updateWallet(wallet, hashTX(tx), (#3 tx)+(#4 tx))]
hashedTX(tx)
Tx:Fund Transaction Created
Out

**Figure 12.** 'Create Tx:Fund Transaction' sub-module of the BlockVoke CPN model.

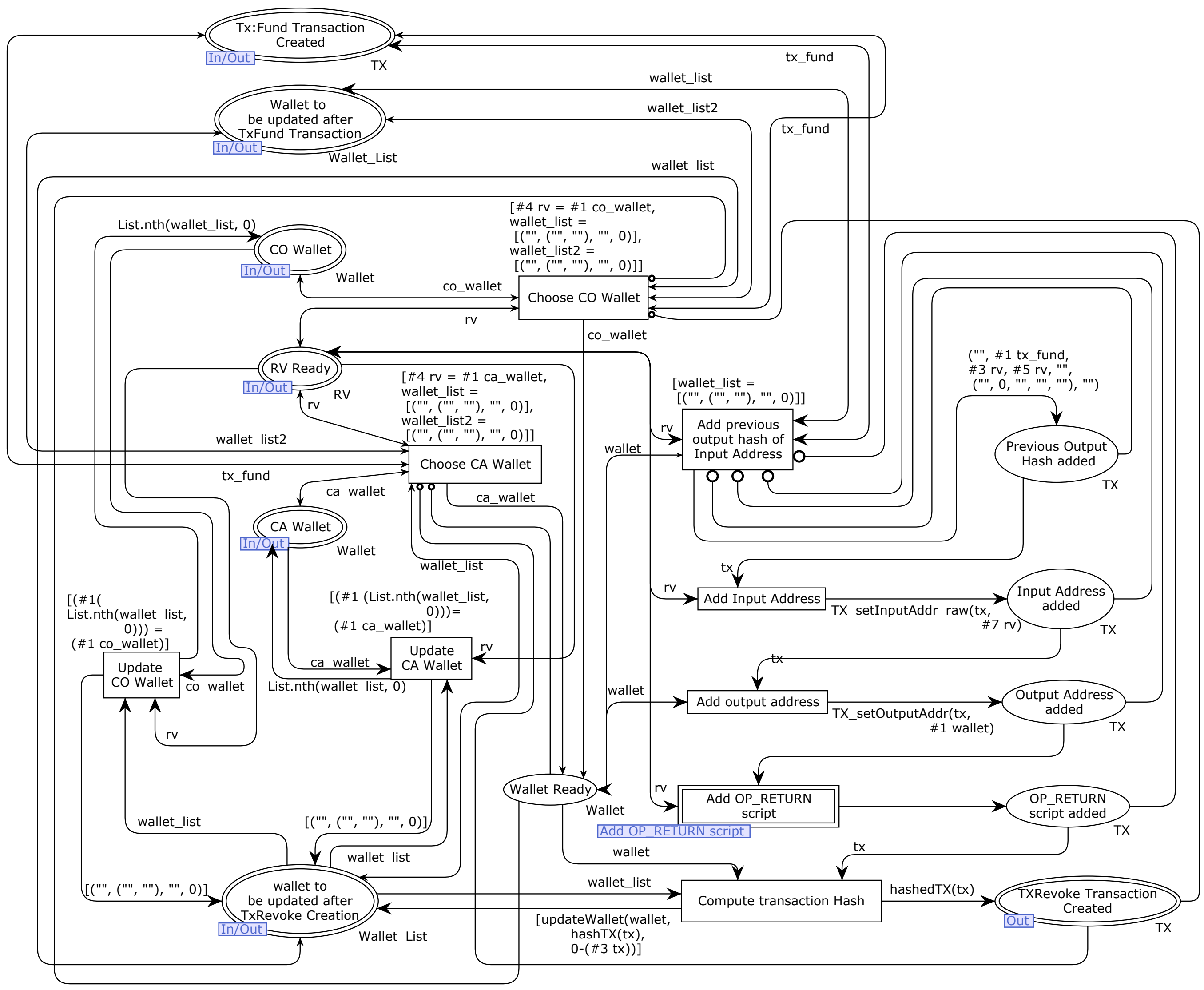

Tx:Fund Transaction Created
In/Out
TX
tx_fund
wallet_list
Wallet to be updated after TxFund Transaction
wallet_list2
In/Out
Wallet_List
tx_fund
wallet_list
[#4 rv = #1 co_wallet, wallet_list = [("", ("", ""), "", 0)], wallet_list2 = [("", ("", ""), "", 0)]]
List.nth(wallet_list, 0)
CO Wallet
In/Out
Wallet
co_wallet
Choose CO Wallet
rv
co_wallet
RV Ready
In/Out
RV
rv
[#4 rv = #1 ca_wallet, wallet_list = [("", ("", ""), "", 0)], wallet_list2 = [("", ("", ""), "", 0)]]
[wallet_list = [("", ("", ""), "", 0)]]
("", #1 tx_fund, #3 rv, #5 rv, "", ("", 0, "", "", ""), "")
Add previous output hash of Input Address
Previous Output Hash added
TX
wallet_list2
tx_fund
Choose CA Wallet
wallet
rv
ca_wallet
ca_wallet
CA Wallet
In/Out
Wallet
wallet_list
tx
rv
Add Input Address
Input Address added
TX_setInputAddr_raw(tx, #7 rv)
TX
[(#1( List.nth(wallet_list, 0))) = (#1 co_wallet)]
[(#1 (List.nth(wallet_list, 0)))= (#1 ca_wallet)]
Update CA Wallet
rv
Update CO Wallet
co_wallet
ca_wallet
List.nth(wallet_list, 0)
tx
wallet
Add output address
TX_setOutputAddr(tx, #1 wallet)
Output Address added
TX
rv
Wallet Ready
Wallet
Add OP_RETURN script
Add OP_RETURN script
OP_RETURN script added
TX
wallet_list
[("", ("", ""), "", 0)]
wallet
tx
wallet_list
[("", ("", ""), "", 0)]
wallet to be updated after TxRevoke Creation
In/Out
Wallet_List
wallet_list
Compute transaction Hash
hashedTX(tx)
TXRevoke Transaction Created
Out
TX
[updateWallet(wallet, hashTX(tx), 0-(#3 tx))]

**Figure 13.** 'Create Tx:Revoke Transaction' sub-module of the BlockVoke CPN model.

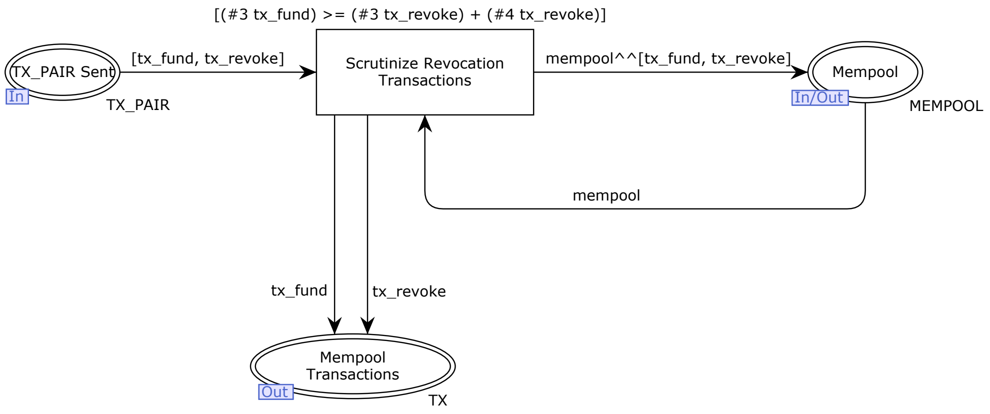

[(#3 tx_fund) >= (#3 tx_revoke) + (#4 tx_revoke)]
TX_PAIR Sent
In
TX_PAIR
[tx_fund, tx_revoke]
Scrutinize Revocation Transactions
mempool^^[tx_fund, tx_revoke]
Mempool
In/Out
MEMPOOL
mempool
tx_fund
tx_revoke
Mempool Transactions
Out
TX

**Figure 14.** 'Add Unconfirmed Revocation Transactions to Mempool' sub-module of the BlockVoke CPN model.

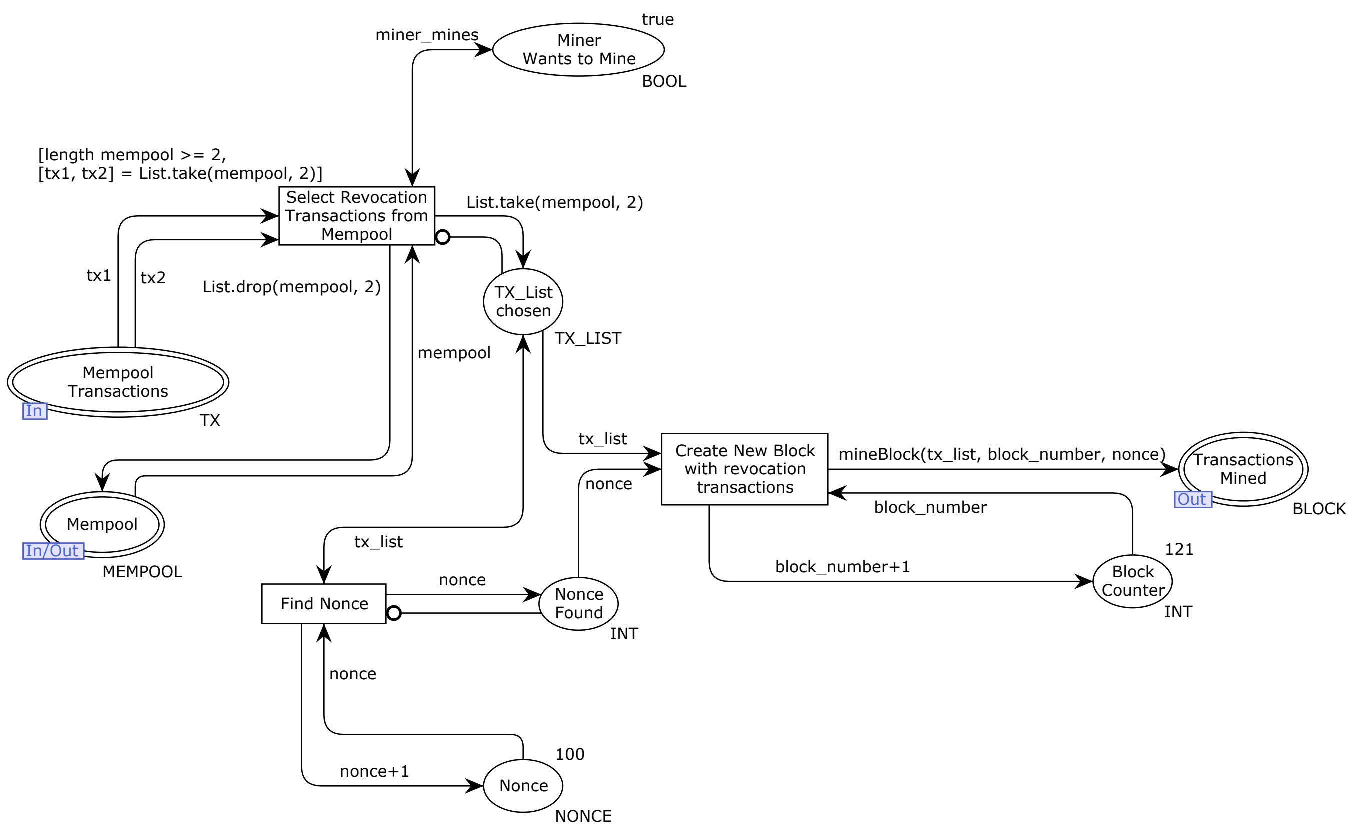


**Figure 15.** 'Mine Revocation Transactions' sub-module of the BlockVoke CPN model.

The 'Find Nonce' transition symbolically simulates the proof-of-work 'mining' operation that a miner performs. Once a nonce is 'found', the 'Create New Block with revocation transactions' transition is enabled, which subsequently assigns a symbolic block number to the new 'BLOCK'. The newly mined block is added to the 'Transactions Mined' place.

### 6.3.4. Propagate Mined Blocks

The 'Propagate Mined Blocks' sub-module moves the individual mined transactions to the 'Mined Transactions' place, as shown in Figure 16. This is a simplified model of the propagation of mined blocks on the blockchain network.

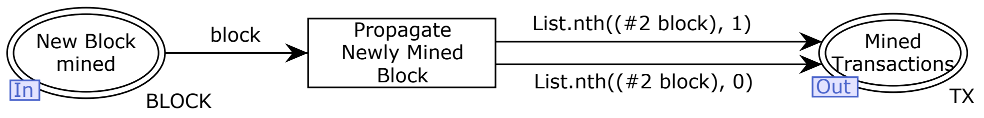


**Figure 16.** 'Propagate Mined Blocks' sub-module of the BlockVoke CPN model.

### 6.3.5. Mark Certificate as Revoked

The 'Mark Certificate as Revoked' sub-module is the final step in the revocation process and is shown in Figure 17. It models how transactions accepted into the mempool or the mined to blocks are communicated to the end-users. After that, among other things, the OP_RETURN payload is checked to confirm that the transaction is a BlockVoke transaction. The respective certificate is marked as 'Revoked' by adding the token to the 'Revoked Certificate' place.

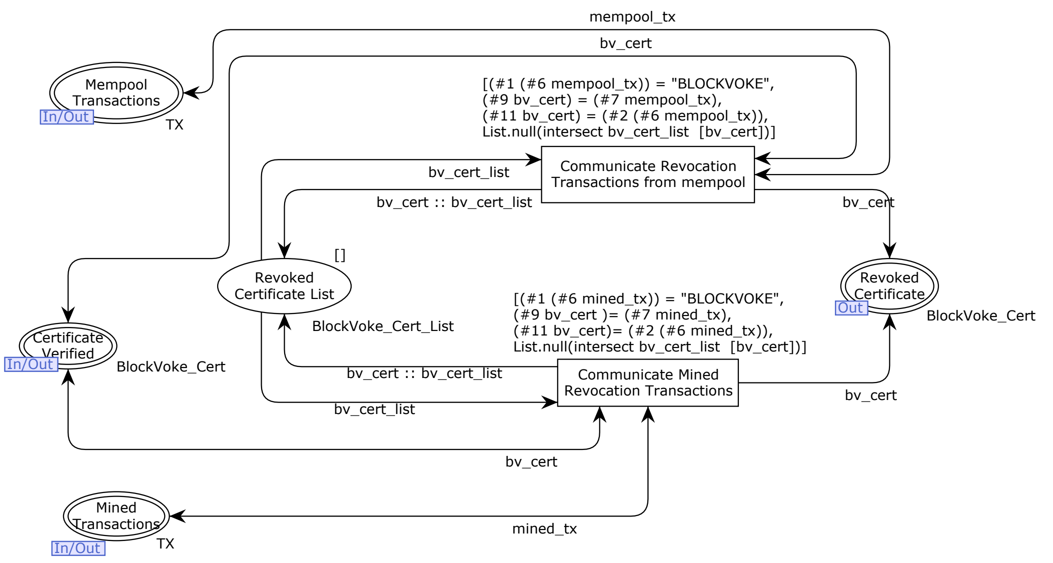

mempool_tx
bv_cert
Mempool Transactions
In/Out
TX
[(#1 (#6 mempool_tx)) = "BLOCKVOKE",
(#9 bv_cert) = (#7 mempool_tx),
(#11 bv_cert) = (#2 (#6 mempool_tx)),
List.null(intersect bv_cert_list [bv_cert])]
bv_cert_list
Communicate Revocation Transactions from mempool
bv_cert :: bv_cert_list
bv_cert
[]
Revoked Certificate List
Revoked Certificate
Out
BlockVoke_Cert
Certificate Verified
In/Out
BlockVoke_Cert
BlockVoke_Cert_List
[(#1 (#6 mined_tx)) = "BLOCKVOKE",
(#9 bv_cert )= (#7 mined_tx),
(#11 bv_cert)= (#2 (#6 mined_tx)),
List.null(intersect bv_cert_list [bv_cert])]
bv_cert :: bv_cert_list
Communicate Mined Revocation Transactions
bv_cert_list
bv_cert
bv_cert
Mined Transactions
In/Out
TX
mined_tx

**Figure 17.** 'Mark Certificate as Revoked' sub-module of the BlockVoke CPN model.

## 7. Evaluation and Discussion

First, Section 7.1 focuses on the evaluation of the BlockVoke CPN model by performing a state-space analysis. Thereupon, we derive model properties and explain their implications. Afterward, Section 7.2 discusses the findings of our work.

### *7.1. CPN State-Space Analysis*

The state-space analysis calculates all reachable states and state changes of a given CPN model. The result is a directed graph where the nodes correspond to the set of reachable markings, while the arcs correspond to occurring binding elements [9]. In addition to the directed state-space graph, the graph's Strongly Connected Component (SCC) is calculated. The nodes of the SCC are "obtained by making a disjoint division of the nodes in the state space such that two state-space nodes are in the same SCC if and only if they are mutually reachable, i.e., there exists a path in the state space from the first node to the second node and vice versa" [9]. Based on the directed graph and the SCC, properties of the system represented by the CPN model can be deduced. The state-space analysis and the computation of the SCC are performed using the corresponding built-in functionality of CPN-Tools.

The results of the state-space analysis are listed in Table 6. We computed a full state-space and the corresponding SCC for the CPN model. The full state-space analysis results are available in Appendix A.

**Table 6.** Selected state-space analysis results of the BlockVoke CPN model.

| | **Loops** | **Home Markings** | **Dead Markings** | **Dead Transitions** | **Live Transitions** |
|---|---|---|---|---|---|
| BlockVoke | No | No | Yes | No | No |

As presented in Table 6, the CPN model does not contain any loops. Thus, there are no infinite occurrences of execution paths in the state-space graph, which guarantees the model's, and therefore the protocol's, termination. Moreover, no home markings are present. Home markings can be reached from any other reachable marking, meaning that it is impossible to have a sequence occur that cannot be extended to reach the home marking.

The listed dead markings are intentionally caused by customized input values. "A dead marking is a marking in which no binding elements are enabled" [9]. Dead markings guarantee the termination of executable actions at a certain point, thereby preventing infinite runtime, and imply the absence of a live transition. "A transition is live if from any reachable marking we can always find an occurrence sequence containing the transition" [9].

Finally, the analysis reveals the absence of dead transitions. A transition is considered dead if there is no reachable marking that enables the transition. Therefore, we deduce that all transitions of the tested BlockVoke CPN model can be potentially enabled at runtime [9].

### 7.2. Discussion

Various works demonstrate the general applicability of CPNs for formalizing, analyzing and verifying existing protocols [39,40,48]. Vanek and Rohlik [48] implemented and simulated the DREAM (DoS-Resistant Efficient Authentication Mechanism) protocol [49], thereby showcasing the applicability of CPNs to create and simulate fully functional protocol models. The authors of [50–52] used CPNs to model attacks and conduct security assessments on selected security protocols and verify the corresponding CPN models.

However, formal models and their analysis/verification also have limitations, e.g., Cohn-Gordon et al. [14] acknowledged limitations of their formal analysis caused by simplification and intentionally limiting the scope of the model.

Evaluation limitations of the BlockVoke CPN model may concern the socio-technical nature of the BlockVoke protocol and the modeling process itself, which requires several simplifications, e.g., neither the Bitcoin nor the Ethereum consensus algorithm or mining process was implemented in the CPN model. Further limitations result from customized input statements of the CPN model used to prevent state-space explosions. Moreover, we simplified data structures of the BlockVoke protocol and the blockchain platform, e.g., no Merkle trees, blockchain blocks have only a symbolic nonce, transactions are not signed or verified in a way that guarantees authenticity and hard-coded multi-signature addresses are simply assigned to certificates instead of them being dynamically created by the CA based on their and the CO's wallet addresses.

## 8. Conclusions and Future Work

While a general description of BlockVoke is presented in [26], a formal model that could be analyzed and verified was missing. Formal methods, such as CPNs, facilitate the design, development, analysis and verification of new protocols; the detection of flaws; and the mitigation of identified security risks. In this work, the BlockVoke protocol is formalized using CPNs, resulting in a verifiable CPN model and a formal protocol specification. We utilize an agent-oriented modeling methodology to create goal models and corresponding behavior interface models of BlockVoke. Subsequently, protocol semantics are defined, and the CPN model is derived and implemented using CPN Tools.

The top-level CPN model consists of three sub-modules, i.e., *Generate Certificate*, *Verify Certificate* and *Revoke Certificate*, which are further refined and described in detail. We present the required protocol semantics by defining the necessary token color sets, thereby mapping them to the corresponding blockchain-equivalent actions and representing the used data structures. Moreover, we conduct a full state-space analysis on the resulting CPN model and derive specific model properties. Furthermore, we discuss and explain the implications of these model properties. The state-space analysis confirms the absence of loops, dead markings and home markings as well as dead- and live transitions. As a result, we present a complete and correct formal BlockVoke specification that is used to guide future implementations and security assessments of BlockVoke. Furthermore, we acknowledge and discuss the limitations of the CPN models resulting from either the simplification of operations and models or the socio-technical nature of the BlockVoke protocol.

Our findings allow for and support the correct and secure implementation of BlockVoke according to its specifications, while the graphical notation of CPN models provide an additional easy to understand and intuitive documentation of the protocol flow—not only for developers but also for stakeholders deploying and/or using BlockVoke. Finally, the similarities between blockchains and CPNs allow for an effortless mapping between blockchain-based state changes and the CPN state changes which drive the protocol execution flow.

The results of our work have some limitations caused by simplifications and intentionally limiting the scope of the formal BlockVoke model pertaining to the socio-technical nature of the protocol and the modeling process itself. For example, the BlockVoke CPN model uses simplified data structures, e.g., no Merkle trees, does not model the consensus algorithm or mining process of the underlying blockchain and relies on customized input statements of the CPN model used to prevent state-space explosions.

Besides implementing and deploying BlockVoke on the Bitcoin or Ethereum blockchain, we plan to conduct an extensive risk and threat analysis of the protocol based on the formal CPN model. Further security enhancements will follow by applying security risk-oriented patterns [23] to the BlockVoke CPN model. Besides, we plan to eliminate or minimize the limitations of the CPN model, e.g., the missing consensus and mining mechanisms, thereby improving the overall quality of the CPN model.

**Author Contributions:** All three authors contributed equally to the original research idea, evaluation, production of figures and writing the paper. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this work are openly available via https://github.com/bleidingGOE/2021_BlockVoke-CPN-Files (accessed on 30 May 2021).

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A. BlockVoke Protocol Formalization

### *Appendix A.1. AOM Goal Model*

https://github.com/bleidingGOE/2021_BlockVoke-CPN-Files/blob/c7cbdff5af5a022dd9829b1ffc542f8b6a5c2831/2021-05-31_BlockVoke-Journal-Paper--MDPI/20210530_BlockVoke--AOM-Goal-Model.pdf, accessed on 30 May 2021.

### *Appendix A.2. Behavior Interfaces of Activities*

https://github.com/bleidingGOE/2021_BlockVoke-CPN-Files/blob/c7cbdff5af5a022dd9829b1ffc542f8b6a5c2831/2021-05-31_BlockVoke-Journal-Paper--MDPI/20210530_BlockVoke--BIM.pdf, accessed on 30 May 2021.

### *Appendix A.3. CPN Model*

https://github.com/bleidingGOE/2021_BlockVoke-CPN-Files/blob/c7cbdff5af5a022dd9829b1ffc542f8b6a5c2831/2021-05-31_BlockVoke-Journal-Paper--MDPI/20210530_BlockVoke--CPN-v04.cpn, accessed on 30 May 2021.

### *Appendix A.4. CPN Protocol Semantics*

https://github.com/bleidingGOE/2021_BlockVoke-CPN-Files/blob/c7cbdff5af5a022dd9829b1ffc542f8b6a5c2831/2021-05-31_BlockVoke-Journal-Paper--MDPI/20210530_BlockVoke--CPN-Protocol-Semantics.pdf, accessed on 30 May 2021.

### *Appendix A.5. State-Space Analysis*

https://github.com/bleidingGOE/2021_BlockVoke-CPN-Files/blob/c7cbdff5af5a022dd9829b1ffc542f8b6a5c2831/2021-05-31_BlockVoke-Journal-Paper--MDPI/20210528_BlockVoke--State-Space-Analysis-v4.txt, accessed on 30 May 2021.

## References

1. Cam-Winget, N.; Housley, R.; Wagner, D.; Walker, J. Security Flaws in 802.11 Data Link Protocols. *Commun. ACM* **2003**, *46*, 35–39. [CrossRef]
2. Carlsen, U. Cryptographic Protocol Flaws: Know Your Enemy. In Proceedings of the Computer Security Foundations Workshop VII, CSFW 7, Franconia, NH, USA, 14–16 June 1994; pp. 192–200.
3. Fábrega, F.J.T.; Herzog, J.C.; Guttman, J.D. Strand Spaces: Why is a Security Protocol Correct? In Proceedings of the Security and Privacy, Oakland, CA, USA, 6 May 1998; pp. 160–171.
4. Vaudenay, S. Security Flaws Induced by CBC Padding-Applications to SSL, IPSEC, WTLS... In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 534–545.
5. Heartbleed–CVE-2014-0160. 2014. Available online: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160 (accessed on 11 May 2021).
6. Wuille, Pieter and Maxwell, Greg. SigSpoof–CVE-2018-12020. 2018. Available online: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-12020 (accessed on 11 May 2021).
7. Petri, C.A. Kommunikation Mit Automaten. Ph.D. Thesis, Technical University of Darmstadt, Darmstadt, Germany, 1962.
8. Jensen, K. Coloured Petri Nets. In Proceedings of the Discrete Event Systems: A New Challenge for Intelligent Control Systems, London, UK, 4 June 1993; pp. 1–5.
9. Jensen, K.; Kristensen, L.M.; Wells, L. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *Int. J. Softw. Tools Technol. Transf.* **2007**, *9*, 213–254. [CrossRef]
10. Milner, R.; Parrow, J.; Walker, D. A Calculus of Mobile Processes, I. *Inf. Comput.* **1992**, *100*, 1–40. [CrossRef]
11. Hoare, C.A.R. Communicating Sequential Processes. In *The Origin of Concurrent Programming*; Springer: Berlin/Heidelberg, Germany, 1978; pp. 413–443.
12. Crazzolara, F.; Winskel, G. Events in Security Protocols. In Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia PA, USA, 5 November 2001; pp. 96–105.
13. Abadi, M.; Gordon, A.D. A Calculus for Cryptographic Protocols: The Spi Calculus. In Proceedings of the 4th ACM Conference on Computer and Communications Security, Zurich, Switzerland, 4 April 1997; pp. 36–47.
14. Cohn-Gordon, K.; Cremers, C.; Dowling, B.; Garratt, L.; Stebila, D. A Formal Security Analysis of the Signal Messaging Protocol. *J. Cryptol.* **2020**, *33*, 1914–1983. [CrossRef]
15. Fett, D.; Küsters, R.; Schmitz, G. A Comprehensive Formal Security Analysis of OAuth 2.0. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24 October 2016; pp. 1204–1215.
16. Duo, W.; Xin, H.; Xiaofeng, M. Formal Analysis of Smart Contract Based on Colored Petri Nets. *IEEE Intell. Syst.* **2020**, *35*, 19–30. [CrossRef]
17. Rahman, M.S.; Khalil, I.; Bouras, A. Formalizing Dynamic Behaviors of Smart Contract Workflow in Smart Healthcare Supply Chain. In *International Conference on Security and Privacy in Communication Systems*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 391–402.
18. Liu, Z.; Liu, J. Formal Verification of Blockchain Smart Contract Based on Colored Petri Net Models. In Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 15–19 July 2019; Volume 2, pp. 555–560.
19. Leiding, B.; Norta, A. Mapping Requirements Specifications Into a Formalized Blockchain-Enabled Authentication Protocol for Secured Personal Identity Assurance. In *International Conference on Future Data and Security Engineering*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 181–196.
20. Norta, A.; Matulevičius, R.; Leiding, B. Safeguarding a Formalized Blockchain-Enabled Identity-Authentication Protocol by Applying Security Risk-Oriented Patterns. *Comput. Secur.* **2019**, *86*, 253–269. [CrossRef]
21. Leiding, B.; Cap, C.H.; Mundt, T.; Rashidibajgan, S. Authcoin: Validation and Authentication in Decentralized Networks. In Proceedings of the 10th Mediterranean Conference on Information Systems-MCIS, Paphos, Cyprus, 5 September 2016.
22. Dubois, E.; Heymans, P.; Mayer, N.; Matulevičius, R. A Systematic Approach to Define the Domain of Information System Security Risk Management. In *Intentional Perspectives on Information Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 289–306.
23. Ahmed, N.; Matulevičius, R. Securing Business Process Using Security Risk-oriented Patterns. *Comput. Stand. Interfaces* **2014**, *36*, 723–733. [CrossRef]
24. Ahmed, N.; Matulevičius, R. Presentation and Validation of Method for Security Requirements Elicitation from Business Processes. In Proceedings of the Information Systems Engineering in Complex Environments, Thessaloniki, Greece, 16–20 June 2014.
25. Norta, A.; Kutvonen, L. *Safeguarding Trusted eBusiness Transactions of Lifecycles for Cross-Enterprise Collaboration*; Technical Report; Department of Computer Science, University of Helsinki: Helsinki, Finland, 2012.
26. Garba, A.; Bochem, A.; Leiding, B. BlockVoke–Fast, Blockchain-Based Certificate Revocation for PKIs and the Web of Trust. In *International Conference on Information Security*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 315–333.
27. Smith, T.; Dickinson, L.; Seamons, K. Let's Revoke: Scalable Global Certificate Revocation. In Proceedings of the 27th Annual Network and Distributed System Security Symposium (NDSS 2020), San Diego, CA, USA, 23 February 2020.

28. Chung, T.; Lok, J.; Chandrasekaran, B.; Choffnes, D.; Levin, D.; Maggs, B.M.; Mislove, A.; Rula, J.; Sullivan, N.; Wilson, C. Is the Web Ready for OCSP Must-Staple? In Proceedings of the Internet Measurement Conference 2018, Boston, MA, USA, 31 October 2018; pp. 105–118.
29. Cooper, D.; Santesson, S.; Farrell, S.; Boeyen, S.; Housley, R.; Polk, W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF RFC5280, May 2008. Available online: https://tools.ietf.org/html/rfc5280 (accessed on 28 May 2020).
30. Santesson, S.; Myers, M.; Ankney, R.; Malpani, A.; Galperin, S.; Adams, C. X. 509 Internet Public Key Infrastructure Online Certificate Status Protocol–OCSP. IETF RFC6960, June 2013. Available online: https://tools.ietf.org/html/rfc6960 (accessed on 28 May 2020).
31. Basin, D.A.; Cremers, C.; Kim, T.H.; Perrig, A.; Sasse, R.; Szalachowski, P. Design, Analysis, and Implementation of ARPKI: An Attack-Resilient Public-Key Infrastructure. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15*, 393–408. [CrossRef]
32. Eastlake, D. Transport Layer Security (TLS) Extensions: Extension Definitions. IETF RFC6066, January 2011. Available online: https://tools.ietf.org/html/rfc6066 (accessed on 28 May 2020).
33. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 9 May 2021).
34. Wood, G. Ethereum: A Secure Decentralized Generalised Transaction Ledger. 2014. Available online: http://gavwood.com/paper.pdf (accessed on 30 May 2021).
35. Amoah, R.; Suriadi, S.; Camtepe, S.; Foo, E. Security Analysis of the Non-Aggressive Challenge Response of the DNP3 Protocol Using a CPN Model. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, Australia, 10–14 June 2014; pp. 827–833.
36. Bochem, A.; Leiding, B. Rechained: Sybil-Resistant Distributed Identities for the Internet of Things and Mobile Ad Hoc Networks. *Sensors* **2021**, *21*, 3257. [CrossRef]
37. Larisch, J.; Choffnes, D.; Levin, D.; Maggs, B.M.; Mislove, A.; Wilson, C. CRLite: A Scalable System for Pushing all TLS Revocations to all Browsers. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 539–556.
38. Jensen, K.; Kristensen, L.M. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2009.
39. Aly, S.; Mustafa, K. Protocol Verification and Analysis Using Colored Petri Nets. 2003. Available online: http://facweb.cs.depaul.edu/research/techreports/tr04-003.pdf (accessed on 13 February 2021).
40. Edwards, K. Cryptographic Protocol Specification and Analysis Using Coloured Petri Nets and Java. Ph.D. Thesis, Queen's University Kingston, Kingston, ON, Canada, 1999.
41. Mahunnah, M.; Norta, A.; Ma, L.; Taveter, K. Heuristics for Designing and Evaluating Socio-technical Agent-Oriented Behaviour Models with Coloured Petri Nets. In Proceedings of the 2014 IEEE 38th International Computer Software and Applications Conference Workshops (COMPSACW), Vasteras, Sweden, 21–25 July 2014; pp. 438–443.
42. Sterling, L.; Taveter, K. *The Art of Agent-Oriented Modeling*; MIT Press: Cambridge, MA, USA, 2009.
43. Davis, A.M. *Software Requirements: Objects, Functions, and States*; Prentice-Hall, Inc.: Yankee Ferry, NJ, USA, 1993.
44. IEEE Computer Society. Software Engineering Technology Committee and Institute of Electrical and Electronics Engineers. In *IEEE Recommended Practice for Software Requirements Specifications*; IEEE Std, Institute of Electrical and Electronics Engineers: Piscataway, NJ, USA, 1994.
45. Norta, A.; Grefen, P.; Narendra, N.C. A Reference Architecture for Managing Dynamic Inter-Organizational Business Processes. *Data Knowl. Eng.* **2014**, *91*, 52–89. [CrossRef]
46. Marshall, J. Agent-Based Modelling of Emotional Goals in Digital Media Design Projects. *Int. J.-People-Oriented Program. (IJPOP)* **2014**, *3*, 44–59. [CrossRef]
47. Fourman, M.P. Arrays. 2010. Available online: https://homepages.inf.ed.ac.uk/mfourman/teaching/mlCourse/notes/sml-arrays.html (accessed on 29 May 2021).
48. Vanek, T.; Rohlik, M. Model of DoS Resistant Broadcast Authentication Protocol in Colored Petri Net Environment. In *IWSSIP 2010 Proceedings 2010*; Rio de Janeiro, Brazil; pp. 264–267. Available online: http://www2.ic.uff.br/iwssip2010/Proceedings/nav/papers/paper_85.pdf (accessed on 30 May 2021).
49. Huang, Y.; He, W.; Nahrstedt, K.; Lee, W.C. DoS-resistant Broadcast Authentication Protocol with Low End-to-End Delay. In Proceedings of the IEEE INFOCOM Workshops 2008, Phoenix, AZ, USA, 13–18 April 2008; pp. 1–6.
50. Dresp, W. Security Analysis of the Secure Authentication Protocol by Means of Coloured Petri Nets. In *IFIP International Conference on Communications and Multimedia Security*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 230–239.
51. Sornkhom, P.; Permpoontanalarp, Y. Security Analysis of Micali's Fair Contract Signing Protocol by Using Coloured Petri Nets. In Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Phuket, Thailand, 6–8 August 2008; pp. 329–334.
52. Xu, Y.; Xie, X. Modeling and Analysis of Security Protocols Using Colored Petri Nets. *JCP* **2011**, *6*, 19–27. [CrossRef]

## Short Biography of Authors

**Anant Sujatanagarjuna** was born in Mumbai, India. He received his B.Sc. degree in computer science in 2018 and his B.Sc. degree in mathematics in 2019, both from the University of Mumbai, India and is currently working towards a M.Sc degree in Applied Computer Science at the University of Goettingen, Germany. His main field of research is blockchain technology.

**Arne Bochem** was born in Bad Mergentheim, Germany. He received the B.Sc. and M.Sc. degrees in applied computer science from the University of Goettingen and is currently working towards a Ph.D. degree at the same university. His main fields of research include secure localization for wireless sensor networks and the Internet of Things as well as blockchain technology.

**Benjamin Leiding** was born in Rostock, Germany. He received his B.Sc. degree in computer science in 2015 from the University of Rostock, Germany. Subsequently, he received the M.Sc. degree in Internet Technologies and Information Systems in 2017 as well as the Ph.D. degree in computer science in 2020 from the University of Goettingen, Germany. He is currently a Post Doctoral Research Fellow at the Clausthal University of Technology. His research interests include the Machine-to-Everything Economy (M2X Economy), the Circular Economy, distributed systems, and digital identities.