Smart contracts for spectrum sensing as a service

Suzan Bayhan, Anatolij Zubow, Piotr Gawłowicz, and Adam Wolisz
Technische Universität Berlin, Germany
Email: {bayhan, zubow, gawlowicz, wolisz}@tkn.tu-berlin.de

***Abstract*—Mobile network operators (MNO) can opportunistically use the licensed bands of the primary users (PU) provided that they monitor the spectrum and stop their transmission upon detection of the PU. As deploying spectrum sensors may be prohibitively expensive, the MNO can buy spectrum sensing service from sensing helpers in its proximity. However, such a trade requires a framework with three key functions: helper selection, faulty or malicious helper identification, and payment to honest helpers. Here, we introduce Spass which provides these functions and facilitates a fair exchange between the entities without a trusted third party via smart contracts (SC) running on a blockchain network. While payments via SCs seem conceptually simple, realizing it is difficult due to the cost of using SC functions which might be prohibitive as write/computation operations on the SCs might have a cost, e.g., in Ethereum. Considering our design goals and SC-related overhead, we derive the optimal Spass parameters maximizing the MNO's profit. Moreover, we propose a K-means clustering approach to identify independent malicious helpers, and using both lossless and lossy compression on the helpers' sensing report to decrease the cost of write operations. Via simulations, we show under which conditions Spass-powered service leads to a profitable business for an MNO.**

***Index Terms*—Smart contract, spectrum sensing, cooperative spectrum sensing, blockchain.**

I. Introduction

With increasing demand for bandwidth-hungry applications such as mobile video, a foreseen capacity shortage has become a key threat to mobile network operators (MNO). To mitigate this threat, MNOs can benefit from temporary, demand-driven expansion of their operation to other bands (licensed or unlicensed) where the spectrum is shared among multiple networks. For example, shared use of 3.6 GHz spectrum by several cellular operators is being considered for 5G roll-out in Germany [1] as a way to address increasing demand for spectrum on one hand and its inefficient use due to exclusive licenses on the other. In licensed bands, an MNO can act as a secondary user (SU) which has to ensure that the primary user (PU) of the licensed band does not experience harmful interference due to the MNO's opportunistic spectrum access. To satisfy this requirement, the MNO (referred to as *SU network*, SUN) must monitor the spectrum and evacuate the channel upon detection of a PU signal. While there might be different pricing policies for opportunistic use of the licensed band, e.g., [2], we consider here that the spectrum for the SUNs does not entail any license fees.

Generally speaking, the SUN may not prefer deploying the sensor infrastructure itself as the associated OPEX and CAPEX might be high. Instead, the SUN can launch a crowd-sensing campaign and collect sensing reports from multiple sensors to maintain a certain level of sensing accuracy. Crowd-sourced spectrum sensing is at the interest of many other parties, e.g., regulators or third-party spectrum sensing/database providers. Regulators need to monitor the spectrum activity both for better policy making and management of this precious natural resource and to catch unauthorized use of the spectrum, i.e., spectrum patrolling [3], [4]. Similarly, spectrum sensing providers can enjoy the sensing capacity of ubiquitous mobile devices to build their spectrum map in a scalable manner [5]. However, sensing units in general lack motivation to perform sensing only for altruistic reasons. Instead, to motivate sensors for participation in the crowd-sensing campaign, the SUN can offer some rewards, e.g., monetary rewards. Smart contracts (SC) running on blockchain distributed ledger [6], [7] are as if tailored to this purpose: a SUN, can publish its requirements for sensing service as in Fig.1 and nodes in its vicinity can subscribe to the contract if they agree with the contract terms. In this paper, building on our previous work [8], we introduce a framework for realizing crowd-sensing requested by a task owner and performed by multiple sensors, so called *helpers*. While our solution can be used for any crowd-sensing application, we consider spectrum sensing as our particular use case and call this solution *Spectrum Sensing as a Service* (Spass)[1].

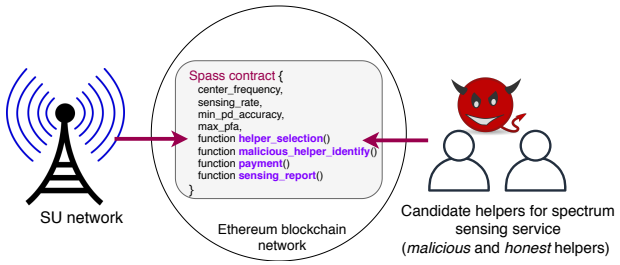


Fig. 1. Spass: Spectrum sensing as a service via smart contracts. Helpers which consist of both malicious and honest ones can offer their sensing service and get payments in return, if selected for the service and identified as honest.

Like any other service, efficiency and business feasibility of Spass matters for the SUN. For efficiency, helpers with high sensing accuracy should be selected by the SC so that idle spectrum can be discovered with a high probability with the minimum number of helpers. Additionally, the spectrum regulator requires a certain level of PU detection accuracy so that the PU network is minimally affected by the SUN. When it comes to feasibility, it depends on two factors. First, the helpers should be willing to participate in Spass. Second,

An earlier version of this paper was presented at IEEE DYSPAN 2018.

[1]Spass means *fun* in German.

cost of Spass should not outweigh SUN's expected profit that will be gained from the SUN's subscribers being served on the discovered secondary spectrum. The SUN has to pay the following two cost components: (i) cost of using SCs on the blockchain and (ii) payment to the helpers for their service. In Spass, the SC is used not only for payments but also for other tasks like selection of helpers among candidate helpers and detection of fraudulent ones. Thus, all these operations running on the blockchain result in computation at the miners and entail a cost that needs to be paid eventually by the SUN. Hence, it is challenging to design an SC that ensures high efficiency but also incurs low cost.

A desirable property of Spass is that it can assess whether a helper fulfills the requirements of the task. In case a helper fails to satisfy the requirements, Spass should exclude this helper (referred to as *malicious helper*) from payment and further service. But, this accounting has to be done at the SC as the SUN might have incentives to deviate from the contract terms, e.g., repudiating the payment for the useful data provided by helpers. The SC in Spass evaluates the accuracy of each helper based on *sensing reports* collected from each helper. As helpers transmit their reports asynchronously, the SC has to store the incoming reports before it processes all data for malicious helper detection. While an SC is usually limited in its storage space, it is also undesirable to store large amount of data as each write operation results in a transaction which is not free. Hence, Spass has to cope with the challenge of identifying malicious helpers while not requiring huge amounts of sensing data from helpers.

Contributions: In a nutshell, our contributions are fourfold:

- First, we design a framework wherein nodes equipped with spectrum sensors (ranging from smart phones to sophisticated spectrum analyzers) can offer their sensing service and will be paid in return if they meet the service level agreements described by the SUN when requesting this service. Note that prior crowd-sourced spectrum sensing proposals, e.g., [3]–[5], [9]–[11], focus on sensing accuracy and assume self-motivated helpers that participate in the sensing campaign without being paid. In contrast, our focus is on the practical mechanism to enable crowd-sourcing via SCs. Different than our earlier work [8], we optimize the SC parameters to ensure that SUN profits from Spass.
- Second, we design a new malicious helper detection scheme different than [8]. *Clustering-based Helper identification* scheme (CHI) achieves both high accuracy in detecting malicious helpers (specifically, *free-riders*[2]) and very low false alarm rate. The former is essential to motivate the SUNs implement Spass-based spectrum discovery whereas the latter is paramount to keep the helpers willing to provide their sensing service. Note that the earlier works on malicious sensor detection analyze a large amount of sensing data to identify sensing anomalies (e.g., attackers) [14] or rely on some trusted infrastructure nodes [5]. In contrast, Spass should keep the information collected from sensors low due to the cost of using the blockchain network. Compared to malicious worker detection in generic crowdsensing scenarios such as [15], Spass assesses the helpers in the SC without access to the ground truth.
- Third, to reduce the amount of data to be stored in the SC, we consider both lossy and lossless compression of the sensing reports. Note that [8] only provides a lossy compression scheme whereas this paper presents not only a lossless compression scheme but also elaborates on how lossless compression is affected by the PU dynamics.
- Fourth, we show the feasibility of Spass via prototype implemented in Solidity.[3]

The rest of the paper is organized as follows. Next, we summarize the related work in Sec. II. We first provide a brief background on SCs in Sec. III, present an overview of Spass in Sec. IV and detail the considered system in Sec. V. We discuss in Sec.VI how a SUN can optimize the contract parameters to maximize its profit while ensuring the design goals of Spass. In Sec. VII and Sec. VIII, we present our proposals for compressing the sensing reports and detecting the malicious helpers, respectively. We evaluate performance of our proposals in Sec.IX. Finally, we highlight the paper's contributions and conclude in Sec. X.

II. Related Work

The most relevant line of research is threefold: i) crowdsourced spectrum sensing, ii) SC-based crowdsourcing, and iii) malicious helper detection (MHD).

Crowdsourced spectrum sensing: We can categorize existing work into four as (i) feasibility analysis which aims at understanding whether low-end inexpensive RF sensors can provide sufficient sensing accuracy, (ii) selection of sensors among the candidates considering accuracy as well as security/privacy aspects, (iii) fusion of the sensor readings, and (iv) incentives or pricing for crowdsensing. One of the earliest studies suggesting crowdsourced spectrum sensing is [10] which provides a feasibility analysis. Similarly, through a large scale measurement study, Saeed et al. [11] show that for TV white space detection, a certain density of low-cost spectrum sensors can provide a comparable accuracy to that of a highly-complex, expensive spectrum analyzer. The accuracy of spectrum discovery improves further if the constructed spectrum map is combined with the local readings of a white-space device (WSD). Under a budget constraint, `SpecSense` [4] selects the minimum number of sensors whose measurements will be interpolated to predict the spectrum state in other areas. `SpecSense` aims at minimizing the prediction error at the location of spectrum queries and therefore favors the sensors in proximity of the query locations. As crowdsourcing is prone to inaccurate sensing decision due to reports from dishonest or malicious sensors, Zhang et al. [5] propose to deploy trusted anchor nodes to assess the trustworthiness of sensors in addition to reputation scores to account for both short term and longer term sensing accuracy of a sensor. Moreover, [5] weights the sensing reports in decision fusion

[2]Another threat is that attackers have enough computation, communication resources, and sufficient credits in their accounts to launch attacks. However, such cases are less probable and requires strongly motivated malicious helpers rather than the free rider model considered in this paper. For a more discussion on possible attacks in crowd-sourcing platforms, please refer to [12], [13].

[3]https://github.com/zubow/Spass_contract

based on a sensor's reputation. Finally, [16] shows the coupling between sensor selection and pricing due to the correlation between two sensors' sensing reports. All these proposals can be used as the sensor selection block in Spass and utility maximization problem should be changed accordingly. Our main focus is on a framework facilitating all these crowdsourcing based proposals via a payment mechanism.

SC-based crowdsourcing: Several studies [15], [17] suggest using SC to realize distributed crowdsourcing applications and micropayments in a dispute-free, trusted, and distributed manner. Spass shares many similarities in design with `CrowdBC` [15] which is a framework for a generic crowdsourcing application and enables a requester and *workers* to trade, the requester to define criteria for worker selection, and evaluation of a worker's service. Spass relies on data compression and encoding to decrease the amount of data to be written to the blockchain while `CrowdBC` proposes to store only hash of the actual data on-chain and the raw data off-chain in a distributed database. To prevent free-riding attack, `CrowdBC` requires that each worker gives some deposits before participating in crowdsourcing. Spass, on the other hand, is robust against free riders owing to its MHD scheme which renders free riding irrational. `PaySense` [17] is more focused on providing anonymity to the workers/helpers and proposes to use multiple Bitcoin addresses as pseudonyms. Another privacy-preserving scheme is [18] which designs a truthful auction scheme to decide the amount of payment to each worker. We acknowledge that the problem addressed by `PaySense` and [18] is paramount, and Spass can also be extended with privacy-preserving features. Spass differs itself from `CrowdBC` and `PaySense` by providing solutions on how to evaluate the service provided by helpers and identify malicious helpers, which are overlooked in [15] and [17]. Finally, while our main focus is on the design of a distributed crowd-sensing system and MHD in this system, we believe that solutions such as [19] can complement our proposal with improvements on the privacy and accountability of each entity.

Malicious helper detection: Our proposal CHI is inspired from [20] which suggests Hamming-distance based assessment of sensing reports. Different than [20], we apply K-means clustering using Hamming-distance based scores. As reducing information collected at the SC becomes a necessity, we also propose lossy and lossless compression of the sensing reports. [5] suggests that operator-deployed trusted nodes evaluate the reports from sensors and quantify their sensing accuracy. Instead, Spass relies only on data collected from helpers.

III. Ethereum smart contracts in a nutshell

Ethereum [21] is a blockchain-based decentralized computing platform which executes and validates transactions by the help of miners. As a compensation for their work, miners are paid for each transaction, e.g., writing a transaction in a block and performing other tasks such as tasks to keep transaction data safe. In Ethereum, cost of an operation is measured in *gas* cost, i.e., in units of "ether" (ETH) [21]. Total cost of a transaction is calculated as a multiplication of total gas cost and gas price which might change over time. A

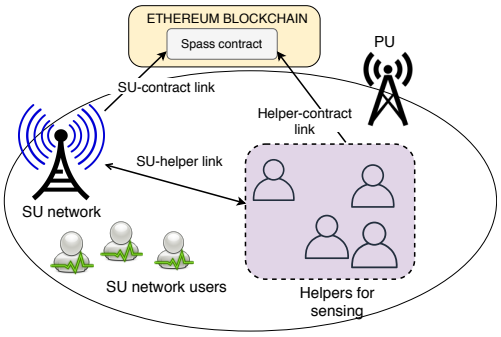


Fig. 2. Spass system model: SUN is interested in accessing the primary user's band opportunistically to serve its users. Helpers are the nodes offering sensing service. The agreement and transactions are processed through the smart contract defined in Ethereum blockchain.

smart contract (SC), identified by a 160-bit unique address in Ethereum, is a computer protocol running on the blockchain to define, verify, and enforce the process of a contract [6], [22], [23]. Different than ordinary accounts in Ethereum, the SCs have also storage space for its relevant data and a set of functions. Ether can be sent from an external account to an SC if the contract has a payable function. Moreover, a contract can send ether to an account identified by an external address. While SCs are stateful and allow storage, it is very costly to store large amounts of data on the contract. Similarly, computation is costly. Therefore, on-chain computation and storage should be avoided as much as possible [15].

IV. Overview of Spass

In this section, we overview how Spass works before we provide more details in Section V.

A. Crowd-sensing using smart contracts (SC)

Let us first introduce the key blocks of a crowd-sensing framework using SCs. A task requester publishes the task definition by deploying its SC on the Ethereum network. Interested helpers can register to the task by calling the SC, e.g., register function, to show their willingness to participate in the crowd-sensing campaign. Then, the crowd-sensing framework needs to implement the following three components: helper selection, malicious helper identification, and clearing. The logic of each component is application-specific. In the rest of the paper, we will consider spectrum sensing as our application and present the details of each component.

B. Crowd-sensing for spectrum discovery

We consider a system as in Fig.2. The SUN is an MNO interested in expanding its capacity with the spectrum that belongs to a PU but is underutilized and therefore can be used by the SUN if discovered via spectrum sensing. This scenario might reflect dynamic spectrum sharing among mobile networks as envisioned by UK's Ofcom [24] or Germany's Bundesnetzagentur [1] where a PU MNO is active in its spectrum with a certain probability. For spectrum discovery, the SUN uses Spass which facilitates it to publish its requirements for the sensing service and the helper nodes to subscribe to the

published SC in case they agree the terms of the SC. While we overview only the high level tasks in this paper, an interested reader can find more on SC design in our earlier work [8] or [15]. Next, we list the operation steps of Spass which are depicted also in Fig.3.

- **Step 0: (SC generation)** The SUN defines the requirements of the sensing service (e.g., minimum sensing accuracy of each helper) as well as the payment policy (e.g., when and under which conditions helpers are paid and exempted from payment) and publishes the SC on the blockchain network. Upon registration, the SUN receives the SC unique address.
- **Step 1:** The SUN broadcasts the address of the SC to let the nodes in its proximity know that it is willing to buy sensing service. Another option for the SUN could be to publish its contract on a third-party web service which lists all SCs.
- **Step 2:** Sensor nodes receiving this broadcast message check the SC using the unique SC address.
- **Step 3:** Each sensor node decides whether it wants to provide the requested sensing service or not comparing its own properties/cost and the SC's terms.
- **Step 4: (Helper registration)** If a sensor node agrees with the terms of the SC and can meet the requirements of the SC, it registers to the SC signalling its willingness to sense.[4] This subscription message can also include additional information such as sensor's accuracy of sensing in terms of probability of PU detection (p_d) and probability of false alarm (p_f) as well as its price for sensing service.
- **Step 5: (Helper selection)** The SC selects H helpers from the candidate helpers considering various parameters, e.g., the price, sensing accuracy, reputation of each helper. The SC has to make sure that the sensing accuracy after decision fusion of the sensing data from the selected helpers can meet the regulatory requirements, e.g., $p_d \geqslant 0.9$ and $p_f \leqslant 0.1$.
- **Step 6:** Selected helpers are notified and can start sensing with the required rate, e.g., sensing every 10 seconds.
- **Step 7: (Spectrum sensing)** Helpers send their sensing data directly to the SUN through the SUN-helper channel. The SUN applies a decision fusion rule, e.g., MAJORITY, on the received sensing data to decide on the state of the spectrum. The SUN takes the appropriate action based on the state of the channel, i.e., spectrum access in case of idle channel and defer access if spectrum is detected to be busy.
- **Step 8: (Verification)** Helpers send their sensing report to the SC at the end of each *verification round* through the helper-contract link. A verification round consists of multiple sensing slots and denotes the time where each *honest*[5] helper gets payment for its service in the previous verification period. To identify whether a helper was honest, the SC runs its *malicious helper detection* (MHD) scheme which takes all sensing reports as input and gives the list of claimed malicious helper IDs as output. The sensing report might be a compressed version of the sensing data representing the sensing outcome of the respective helper during the previous round.
- **Step 9: (Clearing)** The claimed malicious helpers are excluded from the SC and the rest are paid for their sensing service. Next, the SC selects either new helpers from the candidate helpers to replace the excluded malicious helpers or in **Step 5** the SC already admits more helpers than needed and there is still sufficient number of helpers in the SC. The SC goes back to **Step 6** until the SC's expiry time.

[4]Upon registration, the helper might also be requested to make a deposit for its participation as proposed in [15]. Such a deposit mechanism would discourage malicious helpers and DDoS attackers.

[5]The term *honest* refers to a helper which satisfies the required sensing accuracy. Here, we consider a helper *malicious* if, although it is honest by intention, it provides inaccurate sensing data, e.g., due to hardware errors.

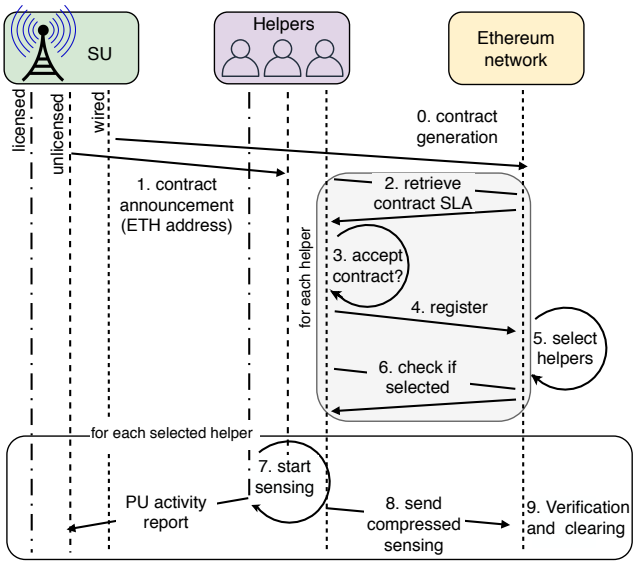


Fig. 3. Interaction between entities in Spass.

V. System Model

Our system consists of four entities: (i) SUN, (ii) helpers, (iii) SC running on a blockchain network, and (iv) miners in the blockchain network. Miners are computation-rich resources who need to solve a puzzle with tuneable difficulty to write any transaction and related information on to the Blockchain. Note that helpers and SUN are not part of the blockchain network. Please refer to [6], [25] and references therein for more details on blockchain.

Let assume that the SUN wants to get spectrum sensing service for a duration of T_c seconds.[6] Therefore, it specifies T_c in its SC with other information on the sensing service and deploys the SC to the blockchain. The spectrum that the SUN is interested in belongs to a PU who has no activity in its spectrum with probability p_0 and accesses the spectrum with probability (1-p_0). In other words, we assume that PU channel occupancy is a stationary random process following a Bernoulli distribution taking value 0 with probability p_0. Moreover, we assume that SUN can acquire or predict this information via spectrum observation and traffic prediction, e.g., [26]. Note that a PU can also decide to lease its spectrum to SUs as suggested by GSMA [27]. In that case, the SU can acquire the information directly from the PU about PU's network characteristics including the expected traffic load. The

[6]Traditional contracts have finite time. Upon expiry of a contract, if the parties are satisfied bilaterally with the exchange (of service and corresponding payment), they can renew the SC. This applies to SCs in our system.

SUN can serve its customers on PU's spectrum with a spectral efficiency of κ bps/Hz during the PU's inactive times and the SUN charges its customers μ ∈ per bps. Note that μ is an expected price calculated by the SUN, e.g., considering traffic fluctuations. Table I summarizes key parameters.

Let $\mathcal{H} = \{h_1, \cdots, h_N\}$ be the set of helpers each with certain sensing capability and cost. We assume that all honest helpers have identical sensing capability (expected p_d^h, p_f^h). However, the sensing outcome of two honest helpers might differ from each other at a particular sensing realization depending on the shadowing or fading conditions. Let N be the total number of nodes which can potentially act as a sensing helper for the SUN. Moreover, we denote by N_m the malicious nodes in the network. We denote the fraction of malicious nodes in the helper population by ψ and refer to this value as *probability that a node is malicious*. A malicious helper generates fake sensing data without sensing. In this way, it does not consume its energy and expects to receive payments if it is not detected to be malicious. We assume that the malicious helpers are aware of p_0 and generate a sensing outcome 1 with probability $\alpha_1 = (1 - p_0)$. With this model, a malicious helper's expected sensing accuracy is: $p_f^m = p_0\alpha_1 = p_0(1 - p_0)$ and $p_d^m = (1 - p_0)^2$. Note that the SUN can first collect some environment information using its infrastructure, similar to *trusted anchors* in [5], to develop some awareness of its operation environment, e.g., helper sensing accuracies, fraction of malicious helpers.

The SC will select H helpers from N candidates to meet the regulatory requirements on probability of detection (p_d*) and probability of false alarm (p_f*). As majority logic[7] is robust against malicious helpers [8], we assume that the SUN applies majority logic for deciding on the spectrum state using the collected spectrum sensing data from the sensing helpers. The SC selects helpers for sensing service for a duration of V time units, e.g., seconds, which is a system parameter set by the SUN. We call this duration *verification round* (cf. Fig.4) and assume that T_c/V is an integer. Then, the SC will be valid during $N_V = T_c/V$ rounds. During this time period V, each selected helper senses the spectrum with sensing rate R_s Hz.

At the end of a verification round, each helper sends a report generated from its sensing outcomes. The outcome of each sensing event is one bit information showing if the spectrum is observed as idle (0) or busy (1). Note that verification round concept is necessary for the SC to be able to take an action against misbehaving helpers. Moreover, receiving payments earlier attracts the honest helpers to participate in the SC, e.g., earlier than T_c. In addition, if helpers change their behavior or their sensing accuracy varies over time (e.g., due to mobility), then the SC can replace such helpers with new ones. But, verification results in additional cost to the SUN for the use of blockchain resources.

We denote the size of a helper's sensing report by S bits which equals to $S = R_{eth} \cdot V$ bits where R_{eth} Hz is the rate of data written to the SC. We denote the cost of write operation by μ_{eth} ∈ per bps and assume a flat-rate payment for sensing paid to each helper denoted by μ_s ∈ per bps. As size of the total sensing data for a verification round equals to R_sV bits, we denote the ratio between the total size of the sensing data and size of the sensing report by β and refer to this ratio as *compression factor*. We calculate the sensing report size as: $S = \frac{R_sV}{\beta}$ bits.

[7]We assume that at least half of the helpers are honest.

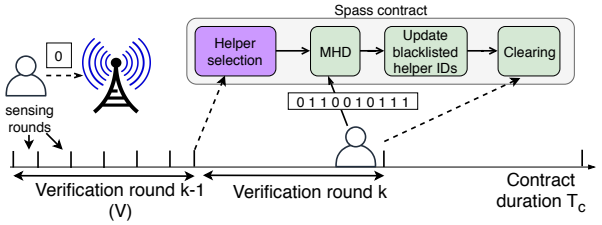


Fig. 4. Verification round consists of multiple sensing rounds. At the end of a verification round, helpers send their compressed sensing reports to the SC.

VI. Optimizing Spass for Maximum SUN Profit

A. Trade-offs

The SUN needs to design its SC parameters such that its profit is maximized. To this end, we need to understand several trade-offs. **Number of helpers (H):** Spectrum sensing accuracy can be improved by increasing number of helpers. Hence, for both regulatory compliance and high spectrum discovery probability, the SUN prefers setting H high. On the other hand, as each helper has to be paid for its sensing service, the SUN would prefer minimizing H. **Verification round duration (V):** The length of this period is relevant for two reasons. First, it determines the earliest time a sensing helper gets its payment. For motivating helpers, the SUN prefers shorter V. Second, it is the earliest time the contract can take an action about possible malicious helpers in the SC. Hence, the SUN would prefer a quick action from Spass in excluding malicious helpers and benefit from shorter V. But, the SC needs to collect sufficiently many sensing samples from helpers to run its malicious helper detection (MHD) algorithm with high accuracy. Hence, V should not be too small. Next, we formulate an optimization problem that the SUN needs to solve to decide on the optimal (H,V) parameters.

B. Cost and utility of the SUN

Let $p_f(H, j)$ denote the false alarm probability if H helpers participate in the sensing process and j of these helpers are malicious. Let $\bar{p_f}(H)$ define the expected false alarm probability under H helpers. We represent the expected utility of the SUN in terms of the spectrum discovered by Spass.[8]

$$\begin{aligned}\mathcal{U}_H &= p_0(1 - \bar{p_f}(H)) \\ &= p_0(1 - \underbrace{\sum_{j=0}^{H} \binom{H}{j} \psi^j (1-\psi)^{H-j} p_f(H, j)}_{\bar{p_f}(H)}) \end{aligned} \quad (1)$$

[8]Note that the utility is the upper bound as some opportunity is lost during the sensing periods where the SUN might need to wait for the sensors data.

where $p_f(H,j)$ for majority logic equals to [8]:

$$p_f(H,j) = \sum_{K=\lceil H/2 \rceil}^{H} \sum_{i=0}^{\min(H,j)} \binom{j}{i} (p_f^m)^i (1-p_f^m)^{j-i} \binom{H-j}{K-i} (p_f^h)^{K-i} (1-p_f^h)^{H-j-K+i}. \quad (2)$$

Next, we calculate the net profit of the SUN for 1 Hz of the discovered spectrum.

SU income Υ^+: We can calculate the monetary reward the SUN will earn by serving its customers via the discovered spectrum considering each verification round v. If we denote the utility at round v by $\mathcal{U}_v$, we then calculate Υ^+ as follows:

$$\Upsilon^+ = \mu\kappa V(\sum_{v=1}^{N_V} \mathcal{U}_v) \text{ Euros.} \quad (3)$$

SU payout Υ^-: If the contract selects H helpers and identifies $\hat{H}_{m,v}$ of the selected helpers as malicious, we can calculate the total amount paid to the service as follows:

$$\Upsilon^- = R_s(\frac{\mu_{eth}}{\beta} + \mu_s) V \sum_{v=1}^{N_V} (H - \hat{H}_{m,v}) \text{ Euros.} \quad (4)$$

Note that the helpers marked as malicious (i.e., blacklisted) receive no payments and will not be selected again.

Profit of the SUN: From (3) and (4), we calculate the profit of the SUN $\Delta\Upsilon = \Upsilon^+ - \Upsilon^-$ as follows:

$$\Delta\Upsilon = V \sum_{v=1}^{N_V} \mu\kappa\mathcal{U}_v - (H - \hat{H}_{m,v}) R_s(\frac{\mu_{eth}}{\beta} + \mu_s) \text{ Euros.} \quad (5)$$

The SUN's contract should be initiated with the optimal parameter values V^* and H^* such that its expected profit is maximized. Hence, we can write the objective of the SUN as: $\max_{V,H} \Delta\Upsilon$. Certainly, there are several constraints which the SUN has to consider. First, the regulatory requirements on p_f and p_d must be satisfied at each verification round. As the exact number of malicious helpers is not known by the contract, the expected $\bar{p_f}(H)$ and $\bar{p_d}(H)$ should be considered in deciding H, which gives us the following two constraints:

$$\bar{p_f}(H) \leqslant p_f * \text{ and } \bar{p_d}(H) \geqslant p_d * .$$

Second, Spass can attract nodes to act as helpers only if its MHD algorithm works with very low false alarm values, i.e., its MHD does not blacklist honest helpers. Let us denote false alarm rate of MHD by q_f and probability of correctly identifying a malicious helper by q_d. These two values depend on the MHD algorithm as well as the number of bits in the sensing report of each helper, number of sensing helpers, and number of malicious helpers. Spass has a target MHD accuracy values denoted by q_f* and q_d* corresponding to the desired false alarm and detection probability of MHD algorithm. We denote the constraint on MHD accuracy as:

$$q_f \leqslant q_f * \text{ and } q_d \geqslant q_d * . \quad (6)$$

Recall that first constraint of (6) is essential to ensure that helpers will participate in sensing service as they will be paid highly probably for their sensing service. Meanwhile, second constraint in (6) is paramount to attract the SUN as Spass guarantees that the likelihood of SUN paying to malicious helpers is very low.

TABLE I
KEY PARAMETERS

Parameter	Description
N, N_m	Number of total nodes and malicious nodes in the network
H, H_m	Number of helpers and malicious helpers in the contract
R_s, R_{eth}	Rate of sensing and rate of writing to the SC (bps)
V	Duration of a verification round
ψ	Probability that a helper is malicious
S	Report size of each helper
Υ, κ	SUN's profit and spectral efficiency
β	Lossy compression factor
p_f	Probability of false alarm
p_0	Probability that PU channel is idle
μ_{eth}	Cost of writing to Ethereum SC (bps)
μ_s	Price of sensing paid to each helper (bps)
μ	Price of serving user (bps)

While our goal is to find a closed-form formula representing the relation between q_f (and q_d) and the number of bits in the sensing report, number of helpers, and number of malicious helpers, it is not straightforward. Instead, we will experimentally analyze these functions to uncover their behavior. Given H and (q_d, q_f) values, we can calculate the total number of helpers identified as malicious as follows: $\hat{H}_{m,v} = H(\psi q_d + (1-\psi)q_f)$.

C. Problem formulation

SUN profit maximization problem can be formulated as:

$$\textbf{P1:} \max_{V,H} \left(V \sum_{v=1}^{N_V} \mu\kappa\mathcal{U}_v - (H - \hat{H}_{m,v}) R_s(\frac{\mu_{eth}}{\beta} + \mu_s) \right) \quad (7)$$

$$N_V = T_c/V \quad (8)$$

$$\mathcal{U}_v = p_0(1 - \bar{p_f}(H)) \quad (9)$$

$$\bar{p_f}(H) = \sum_{j=0}^{H} \binom{H}{j} \psi^j (1-\psi)^{H-j} p_f(H,j) \quad (10)$$

$$p_f(H,j) = \sum_{K=\lceil H/2 \rceil}^{H} \sum_{i=0}^{\min(H,j)} \binom{j}{i} (p_f^m)^i (1-p_f^m)^{j-i} \binom{H-j}{K-i} (p_f^h)^{K-i} (1-p_f^h)^{H-j-K+i}. \quad (11)$$

$$\hat{H}_{m,v} = H(\psi q_d + (1-\psi)q_f) \quad (12)$$

$$q_f \leqslant q_f * \text{ and } q_d \geqslant q_d * \quad (13)$$

$$\bar{p_f}(H) \leqslant p_f * \text{ and } \bar{p_d}(H) \geqslant p_d * \quad (14)$$

$$S = \frac{R_s V}{\beta} \quad (15)$$

$$V \in \{1, \cdots, T_c\} \text{ and } H \in \{1, \cdots, N\} \quad (16)$$

Objective in (7) formally states the expected monetary profit of the SUN achievable over N_V rounds after reduction of its payments both the Ethereum and the sensing helpers perceived as honest ($H - \hat{H}_{m,v}$). Constraint (8) defines the number of rounds in terms of the total contract period and length of one verification round. Constraint (9) defines the expected utility at each round v. Constraint (10) calculates the expected false

alarm probability considering all possible number of malicious helpers in the sensing group while Constraint (11) calculates the false alarm probability in case of j malicious helpers in the sensing group. Constraint (12) computes the expected number of helpers detected as malicious by MHD algorithm. Constraint (13) defines the target values of MHD false alarm value and detection probability, respectively. Similarly, target values for spectrum sensing accuracy are defined in Constraint (14) while Constraint (15) denotes the sensing report size in terms of compression factor, sensing rate, and the verification period length. Finally, variables are defined in Constraint (16). **P1** is difficult to solve analytically as it is hard to model q_d and q_f. Instead, we will experimentally find the number of bits needed to achieve $q_d = 1$ and $q_f = 0$ and will remove Constraint (13).

Assume that minimum sensing report size must be $S_{\min}$ to ensure $q_d = 1$ and $q_f = 0$. Then, assuming that the SC can identify all malicious helpers at the end of the first verification round, we can divide the operation of our solution into two phases. In the first phase which corresponds to the time after the contract selects helpers, there might be malicious helpers in the selected sensing helper set. Let T_0 denote the time from the contract initiation to the first verification time, H_0 denote the number of sensing helpers in this period, and $\mathcal{U}_0$ denote the corresponding utility. In the second phase which corresponds to the time after the first verification till the contract expiry, the SC will not have any malicious helpers unless helpers change their behavior. In this paper, we assume that helpers do not change their strategy.[9] We denote the remaining time for the sensing contract in this second phase by $T_1 = T_c - T_0$, the number of sensing helpers by H_1, and utility by $\mathcal{U}_1$. Now, we rewrite the simplified problem as:

$$\textbf{P2:} \max_{T_0, H_0, H_1} T_0(\mu\kappa\mathcal{U}_0 - H_0 R_s(\frac{\mu_{eth}}{\beta}+\mu_s)) + (T_c - T_0)(\mu\kappa\mathcal{U}_1 - H_1 R_s(\frac{\mu_{eth}}{\beta}+\mu_s)) \quad (17)$$

$$T_0 \leqslant T_c \quad (18)$$

$$\mathcal{U}_0 = p_0(1 - \bar{p_f}(H_0)) \quad (19)$$

$$\mathcal{U}_1 = p_0(1 - p_f(H_1)) \quad (20)$$

$$H_0 \geqslant H_1 \quad (21)$$

$$p_f(H_1) \geqslant p_f^* \text{ and } p_d(H_1) \geqslant p_d^* \quad (22)$$

$$\bar{p_f}(H_0) \geqslant p_f^* \text{ and } \bar{p_d}(H_0) \geqslant p_d^* \quad (23)$$

$$S_{\min} \geqslant \frac{R_s T_0}{\beta} \quad (24)$$

where decision variables are defined as: $T_0 > 0$ and $H_0, H_1 \in \{1, \cdots, N\}$. We will show that T_0 should be kept as short as possible set by Constraint (24) to maximize the objective function in (17). The reason is that $a = \mu\kappa\mathcal{U}_0 - H_0 R_s(\frac{\mu_{eth}}{\beta}+\mu_s)$ will never exceed $b = \mu\kappa\mathcal{U}_1 - H_1 R_s(\frac{\mu_{eth}}{\beta}+\mu_s)$ under the considered operation parameters. In other words, as long as the following inequality holds, T_0 is bounded by Constraint (24):

$$\mathcal{U}_1 - \mathcal{U}_0 \geqslant (H_1 - H_0)\frac{R_s(\frac{\mu_{eth}}{\beta}+\mu_s)}{\kappa\mu}.$$

Assuming the operation parameters to satisfy the above inequality, then we set T_0 using Constraint (24) as follows:

$$T_0 = \frac{S_{\min}\beta}{R_s} \text{ seconds.} \quad (25)$$

With T_0 derived from (25) and inserted in (17), we solve **P2** to find optimal H_0 and H_1. **P2** is a non-linear problem due to $\bar{p_d}(\cdot)$ and $\bar{p_f}(\cdot)$ functions in both the objective and the constraints. Therefore, we will find the solution via exhaustive search with complexity $\mathcal{O}(N^2)$. This complexity follows from our assumption that all helpers in each category are identical in their sensing accuracy.

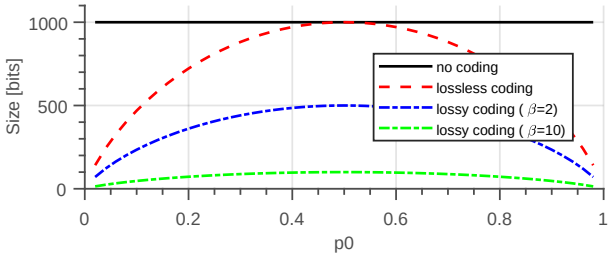


Fig. 5. Impact of compression - no coding, lossless vs. lossy coding.

VII. Compression for Sensing Reports

Our goal is to decrease the amount of data written to the SC to decrease the SUN's payment for Spass. In case of the Ethereum BC, it is the amount of gas to be paid each time new sensing data is reported to the SC. To understand the relation between the sensing report size and the gas consumption, we implemented in Solidity the process of reporting.[10] From the collected data, we obtained the below equation using linear regression ($R^2 \approx 1$). We see a linear relationship between the number of reported sensing bits S and the gas to be paid:

$$\text{gas}(S) = 1600S + 34500 \quad (26)$$

where S is the size of the reported (compressed) data in bits. Note that the gas consumption is dominated by the cost of storing S bits inside the contract which is needed as the sensing helpers report their sensing reports asynchronously. The lowest possible gas price at the time of writing is around 1 Gwei (10^9 Gwei = 1 ETH) whereas the market price of 1 ETH is 500€. This allows us to express the cost of using Ethereum network as below:

$$\mu_{eth} = (1600S + 34500) \cdot 10^{-9} \cdot 500€ = \frac{1}{1250}S + \frac{69}{4000}. \quad (27)$$

We can reduce the number of reported and stored sensing bits in the SC by applying compression techniques. Compression can be either lossy or lossless, which will affect the quality of MHD algorithm. If the compression is lossy (e.g., only

[9] Although based on this assumption, Spass could operate without malicious helper identification so that SUN pays only the sensing-related costs, we still include the verification step and its cost in our model. This is to remove incentives of trustworthy helpers to become malicious helpers.

[10] Please visit https://github.com/zubow/Spass_contract for the source code.

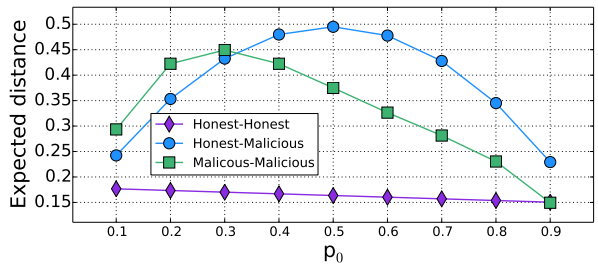


Fig. 6. Expected distance between two helpers according to their type with increasing p_0. $p_f^h = 0.08$ and $p_d^h = 0.90$.

1 bit is kept in every β bits), then the data used for MHD may not represent the reality resulting in low MHD performance, i.e., low detection accuracy and high false alarms. If MHD has a low detection accuracy, then the malicious nodes continue to receive payments while their incorrect input might result in low sensing accuracy at the SUN. The second impact of low MHD accuracy, particularly due to high false alarms in identifying malicious helpers, is the low incentives for nodes to act as helpers. With these goals in mind, we propose next a lossless compression scheme for spectrum sensing reports.

The idea of compression is to remove redundancy in data. The coding problem is to assign code-words to each of the symbols using as few bits as possible. Specifically, we exploit the fact that some symbols are more likely than others which allows us to use more efficient variable-length coding. According to [28], the theoretical optimum for the average number of bits needed to encode a symbol equals to the entropy $\mathcal{X}$ which is defined as:

$$\mathcal{X} = \sum_{i=1}^{L} q_i \log_2 \frac{1}{q_i} \tag{28}$$

where q_i represents the probability of the occurrence of symbol i. In our case, a symbol represents the occupancy state of the PU channel and hence $L = 2$. The corresponding probabilities for each symbol (idle and busy slot) are $q_0 = p_0$ and $q_1 = 1 - p_0$, respectively. Note that sensing helpers are able to estimate p_0 from the observed sensing data with sufficient accuracy or this value is provided by the regulator. Hence, a sensing report of size $S = \frac{R_s V}{\beta}$ with lossy compression factor β can be compressed using optimal source coding to S^*:

$$S^* = \mathcal{X} S \text{ bits}. \tag{29}$$

From a practical point of view, having just two symbols is not sufficient to design efficient coding, i.e. a Huffman code would just be a fixed length code with length of 1 bit/symbol. Hence, the usage of block codes is desirable.

Fig. 5 shows that the higher compression is possible for sparse sensing data, i.e., low/high p_0. For PU bands with low/high p_0, the helpers can have shorter sensing reports, which will decrease the cost of Spass. Moreover, lossy compression ($\beta > 1$) helps to further reduce the amount of data.

VIII. CHI: Clustering-based malicious helper identification

Until this section, we have assumed the existence of an MHD algorithm which can achieve perfect detection accuracy given a certain size ($S \geqslant S_{\min}$) of sensing report. Now, we present our MHD algorithm, namely clustering-based malicious helper identification (CHI).

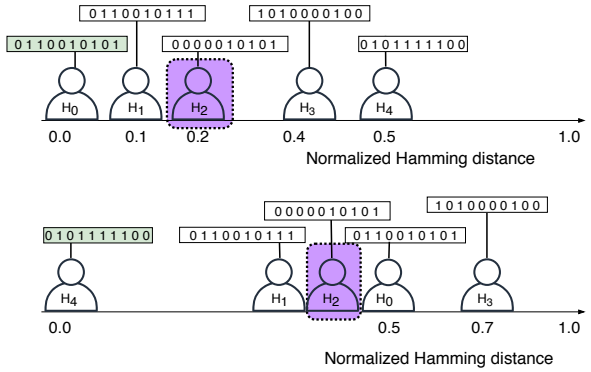


Fig. 7. A helper's score is determined based on its distance from other helpers calculated as normalized Hamming distance between 10-bit sensing reports. h_0's distance from all helpers is [0.1, 0.2, 0.4, 0.5] and therefore 50-percentile score is 0.2. However, h_4's score is 0.5 as it is a malicious helper and its sensing report differs from all others, even the report of malicious helper h_3.

CHI relies on the similarity of sensing reports collected from helpers. The key insight is that selected helpers in the proximity of the SUN should have similar sensing observations, if not the same due to differences in the propagation environment. Hence, if two reports are very similar, e.g., they differ in only few bits, the helpers are highly probably honest helpers. If sensing reports differ from each other noticeably, then at least one of the helpers is malicious.

Let $d_{i,j}$ be the normalized Hamming distance between h_i and h_j's sensing reports [8], [20]. Given p_f^h, p_d^h, $p_d^m = (1 - p_0)^2$, and $p_f^m = p_0(1 - p_0)$, the expected distance between two honest helpers equals to [20]:

$$E[d^{h,h}] = 2p_0 p_f^h (1 - p_f^h) + 2(1 - p_0) p_d^h (1 - p_d^h).$$

Expected distance between a malicious and honest helper is:

$$E[d^{h,m}] = p_0(p_f^h(1 - p_f^m) + p_f^m(1 - p_f^h)) + (1 - p_0)(p_d^h(1 - p_d^m) + (1 - p_d^h)p_d^m).$$

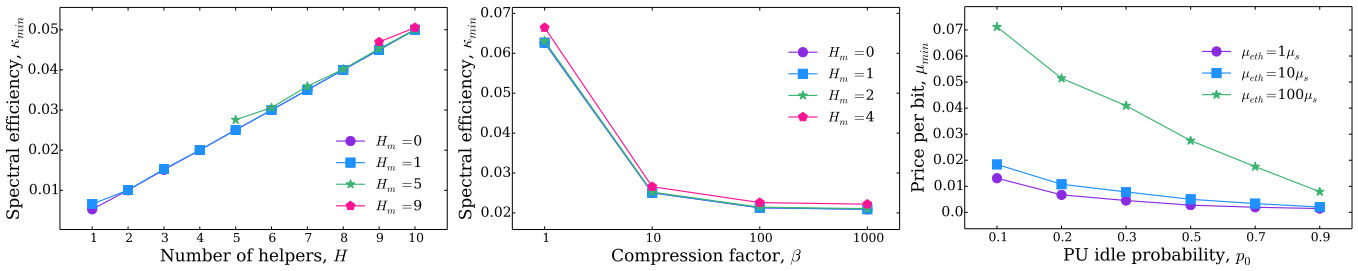
Finally, we can calculate the expected distance between two malicious helpers as follows:

$$E[d^{m,m}] = 2p_0 p_f^m (1 - p_f^m) + 2(1 - p_0) p_d^m (1 - p_d^m).$$

Fig.6 depicts the expected distance with increasing p_0 for two helpers of same and different types. As we observe in Fig.6, mostly the distance between two honest helpers are significantly lower compared to honest-malicious helper distance. In some region, e.g., low p_0 and $p_0 \approx 0.9$, honest and malicious helpers have similar sensing accuracy. For example, for $p_0 = 0.1$, we have the following sensing accuracy values: $p_d^m = 0.81$ and $p_f^m = 0.09$ whereas $p_d^h = 0.9$ and $p_f^h = 0.08$. In this case, it becomes more challenging to distinguish malicious helpers from honest ones.

Let w_i be the score of h_i representing its distance[11] from other helpers. To ensure some robustness against malicious

[11]When we refer to distance between two helpers, we mean the distance between the sensing report of the respective helpers.



(a) Impact of number of helpers. (b) Impact of compression factor. (c) Impact of PU p_0.

Fig. 8. Required operation parameters for the SUN to profit from Spass. Required minimum spectral efficiency (a) under various number of malicious helpers (H_m) with increasing number of helpers (H) and (b) under increasing compression factor β, (c) required minimum price of spectrum to charge the SUN's customers under various Ethereum usage cost $\mu_{eth} = \{\mu_s, 10\mu_s, 100\mu_s\}$ and $\mu_s = 0.05$, $\kappa = 10$, $\beta = 10$, and $H = 5$.

helpers, we compute w_i as the η-percentile of the distances of this helper from other helpers. The percentile value reflects the expected fraction of honest helpers in the contract. Below, we calculate w_i as follows:

$$w_i = P_\eta(d_{i,j} | \forall j \neq i). \tag{30}$$

Fig.7 depicts a toy example with five helpers, two being malicious (h_3 and h_4) and sensing reports include 10 bits. To calculate h_0's score, Spass calculates pairwise distances and sort them in increasing order. Then, Spass takes the 50-percentile in this example as a helper's score. As h_0 has similar sensing reports with those of h_1 and h_2, its score is low. But, malicious helper h_4's sensing score is much higher, i.e., 0.5.

CHI first clusters the scores using K-means algorithm into $K = 1$ clusters. Then, CHI calculates inertia which represents the spread of data points in a cluster from the cluster centroid. Next, CHI runs K-means algorithm for $K = 2$ to consider the possibility that there might be both honest and malicious helpers in the SC. In the existence of malicious helpers, we expect the honest helper scores to be clustered as well as the malicious helper scores being close to each other representing the second cluster. CHI uses this insight to decide on the existence of malicious helpers. After computing the clusters and inertia for $K = 1$ and $K = 2$, CHI checks if the difference between the inertia for $K = 1$ and inertia for $K = 2$ is lower than a threshold value, e.g., 0.05. If lower, CHI concludes that there should be only one cluster, i.e., all helpers are honest. Otherwise, CHI concludes two clusters and marks the helpers in the cluster with lower values as honest and the rest as malicious. CHI then adds the helpers in the malicious cluster to the blacklisted helper list, which will neither get payments nor be considered for sensing in further verification rounds.

IX. Performance Evaluation

We present our findings on the performance of Spass and CHI via simulations using our in-house system level Python simulator. Our goal is to address the following questions: (i) under which parameters can the SUN maintain a profit by Spass-based operation? (ii) how does compression factor and number of malicious helpers affect the accuracy of CHI?

A. When is Spass a feasible business model?

The SUN has to ensure that $\Delta\Upsilon > 0$ for a verification round so that it will benefit from Spass. Using $\Delta\Upsilon$ in (5), we derive a closed formula which gives us the relationship between various parameters of Spass as follows:

$$\mu\kappa\mathcal{U}_v - (H - \hat{H}_{m,v})R_s\left(\frac{\mu_{eth}}{\beta} + \mu_s\right) \geqslant 0. \tag{31}$$

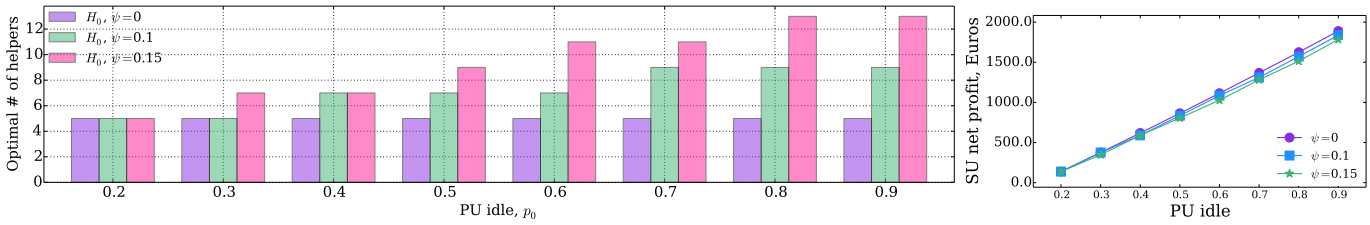
The SUN can use the above formula in many ways. For example, if it has some statistics about the helpers' price and other costs, it can calculate how many helpers it can pay or what is the minimum required spectral efficiency or compression factor ensuring $\Delta\Upsilon > 0$.

Fig.8 plots the required operation parameters under a number of scenarios. Fig.8a and Fig.8b plot the required minimum spectral efficiency κ_{min} for which Spass can provide profit to the SUN under R_s=1, μ_{eth}=0.1, μ_s=0.05, μ=1, and β=1. As expected, with increasing number of helpers, SUN's payment to helpers both for sensing and Ethereum usage will increase. Consequently, κ_{min} will increase. Although the impact of malicious helpers is not significant, higher number of malicious helpers requires slightly higher κ_{min}. Fig.8c shows the minimum price per bps the SUN has to charge its customers for maintaining balance between its cost and income. As different p_0 results in different bits per symbol according to our analysis in Fig.5, we find χ bits per symbol for each p_0 value as in (28). Then, we use $\mu_{eth} = \mu_{eth}\chi$ to reflect the lossless compression on the sensing report. As Fig.8c shows, higher Ethereum usage cost reflects itself in higher price for the customers. With increasing spectrum availability, i.e., higher p_0, the SUN's price will be lower.

Take-aways: Spass benefits slightly from decreasing the number of malicious helpers in the SCs and more significantly from increasing compression factor.

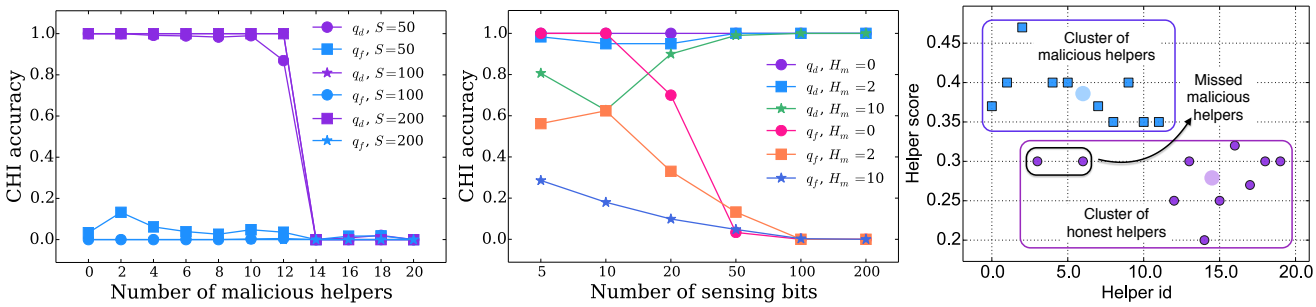
B. Optimal number of helpers

We find the optimal number of helpers via exhaustive search for **P2** with $\mu_{eth} = \chi\mu_{eth}$ to account also for lossless compression. Fig.9a plots H_0 for various p_0 and ψ values while Fig.9b plots the total profit. We have not included H_1 in Fig.9a as it equals to H_0 with $\psi = 0$. The optimal number of helpers is independent of p_0 if all helpers are honest. But, with increasing probability of maliciousness, the SC has to admit more helpers to ensure that the sensing outcome can satisfy the regulatory requirements. Moreover, even if the malicious helpers are excluded at the end of the first phase, there must be still sufficient number of helpers in the SC.



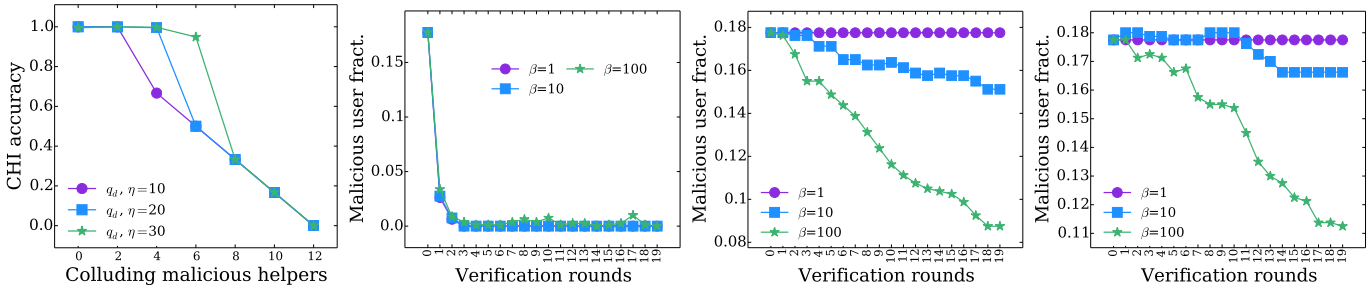
(a) Optimal number of helpers. (b) SUN's net profit.

Fig. 9. Impact of increasing p_0 for various ψ values on (a) optimal number of helpers and (b) SUN's net profit. Following parameters are used: $R_s = 5, \mu_{eth} = 0.1, \mu_s = 0.05, \mu = 1$, and $\beta = 10, \kappa = 10$.



(a) CHI accuracy for $H = 20$. (b) Impact of S on accuracy. (c) Clusters for H_m=12 for H=20 and S=20.

Fig. 10. Accuracy of CHI with increasing number of malicious helpers for $H = 20$, $p_0 = 0.7$, η=30. We repeat each simulation 1000 times.



(a) Total $H_m = 12$ and H=20. (b) $p_0 = 0.7$. (c) $p_0 = 0.9$. (d) $p_0 = 0.1$.

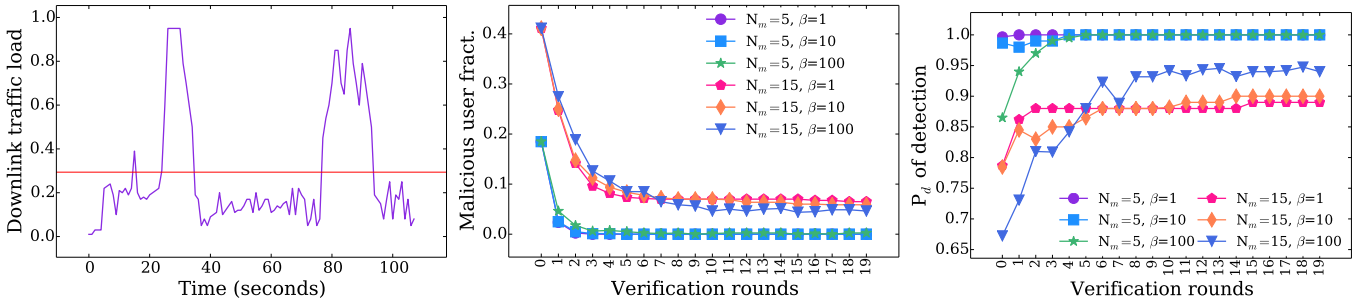
Fig. 11. Accuracy of CHI (a) under colluding malicious helpers, (b, c, d) over multiple verification rounds under different $\beta = \{1, 10, 100\}$ for $N = 40$, $H = 8$, $H_m = 5$, $\eta = 30$, inertia difference=0.02.

Take-aways: Number of helpers required increases under higher PU channel availability and the malicious helper probability. However, since the amount of discovered spectrum also increases, overall high PU channel availability region is the desirable operation region for the SUN as seen in Fig.9b.

C. Accuracy of CHI with increasing ψ

Fig. 10 plots the accuracy of CHI with increasing number of malicious helpers for $H = 20$ for various size of sensing reports, i.e., S. We set η to 30 percentile in these scenarios. Note that higher percentile is preferred for robustness of CHI against collusion of malicious helpers. But, a lower η performs better under higher H_m if malicious helpers are not colluding. As Fig.10a shows, CHI succeeds in identifying malicious helpers with very high accuracy if fraction of malicious helper population is lower than $(1-\eta/100)$. We observe almost perfect detection ($q_d \to 1$) and no false alarms ($q_f \to 0$) for all values for $H_m < 14$. When $H_m \geqslant 14$, a honest helper's score is dominated by the honest-malicious helper distance due to (30). As a result, a honest helper's score becomes very high, similar to malicious helpers. Consequently, CHI detects only a single cluster failing to identify malicious helpers. Given that low malicious helper probability is more likely in real world scenarios, we expect that CHI can identify almost all malicious helpers with a high accuracy in a single verification round. Moreover, if we consider multiple verification rounds, performance of CHI improves with each new round as a big fraction of malicious helpers are excluded from the system thereby resulting in a regime where the malicious helper population is lower. Hence, we expect CHI to identify all malicious helpers in only a few verification rounds.

Fig.10b plots the impact of number of sensing bits S on the accuracy of CHI for various H_m. Lower S results in higher false alarms, particularly in case where there are low number of malicious helpers, e.g., $H_m = 0$. As Spass aims at minimizing false alarms, S must be larger than 50, e.g., 100 bits, in this example. Finally, we plot the clusters of helper scores for $K = 2$ in an example setting in Fig. 10c. In this example, there are 12 malicious helpers and 10 of them are detected by CHI when S=20 bits. In case $K = 1$ (not



(a) An LTE eNodeB's downlink traffic load considering the number of allocated resource blocks.

(b) Malicious helper fraction in the SC.

(c) Malicious helper detection accuracy.

Fig. 12. Performance of Spass under realistic PU traffic load data for N=40, $H = 8$.

plotted), the inertia is markedly higher and CHI concludes that there must be two clusters, one for honest helpers and one for malicious helpers as depicted in Fig.10c.

To understand the impact of η and number of colluding malicious helpers, we simulate a scenario where H_m^{col} malicious helpers generate the same sensing data to achieve a low score. In case of low η, which is preferable for robustness against a high fraction of uncoordinated malicious helpers, it is easier for malicious helpers to form a collusion group with only a few others and maintain a low score which will help such helpers remain undetected by CHI. Fig.11a shows the impact of increasing collusion group. As we observe in Fig.11a, increasing H_m^{col} decreases CHI's accuracy in detecting malicious helpers and increases falsely blacklisted honest helpers (not plotted). Higher η values are more robust compared to smaller η as the latter requires a smaller collusion group for a low score.

Take-aways: When malicious helpers are independent attackers, CHI can identify them with almost perfect accuracy even when malicious helpers are the majority. Under increasing number of colluding malicious helpers, CHI is more robust if a higher η value is used for calculating score of each helper. By tuning η, CHI can set the desired trade-off between robustness against colluding malicious helpers and robustness against independent malicious helpers. Moreover, the sensing report should be long enough (e.g., 100 bits in the considered example) to avoid blacklisting honest helpers.

D. Performance of Spass over multiple verification rounds

Next, we relax our identical sensing accuracy assumption for honest helpers and use the following distributions: $p_d^h \approx U(0.80, 0.95)$ and $p_f^h \approx U(0.05, 0.15)$. As we see in Fig.11b, although the SC might initially have malicious helpers (18% of helpers), after the first round it is only 3% of the helpers. At the end of each verification round, the helpers identified as malicious are excluded from the SC and new ones are selected to replace these excluded ones. Only after two rounds, the sensing contract becomes malicious-helper-free owing to high success of CHI in identifying abnormal behavior. Moreover, CHI never blacklists a honest helper (figure not plotted). Even though there are malicious helpers in the SC initially, Spass achieves a very high performance in identifying the spectrum opportunities, e.g., $p_f \sim 0.01$ (not plotted). When $p_0 \approx 0.9$ or $p_0 \approx 0.1$, CHI performs with lower detection accuracy: there are malicious helpers in the SC as observed in Fig.11c and Fig.11d, respectively. But, no honest helper is blacklisted for $p_0 \approx 0.9$ and it is only 2% under $\beta = 100$ for $p_0 \approx 0.1$. While a spectrum band with $p_0 \approx 0.1$ is not usually considered for opportunistic spectrum access due to very high PU activity in the band, the later case with $p_0 \approx 0.9$ is a desirable setting where there is plenty of spectrum holes for the SUN to utilize. Hence, Spass can use more advanced algorithms such as multi-dimensional k-means clustering rather than CHI which applies clustering on one-dimensional helper scores. Note also that helper scores are affected by parameters inertia threshold and η. Interestingly, Fig.11c and Fig.11d show that a higher β results in a higher detection probability for CHI. We attribute this behavior to the distortion introduced by lossy compression. Two helper reports start to diverge from each other under lossy compression which then helps differentiating honest and malicious helpers. Recall that malicious helpers only know p_0 and therefore can only approximate its sensing accuracy to the expected p_0 if the sensing report is long. The honest helpers are not significantly affected by the distortion due to compression as they sense the spectrum and hence in short or long term their sensing outcomes agree with other honest helpers. As a result, we do not observe an increase in blacklisted honest helpers.

Take-aways: For $0.1<p_0<0.9$, time that a malicious helper stays in a contract is only one or two verification rounds. Hence, Spass can have a payment policy in the SC definition that each helper will be paid only after it stays in the SC for at least two rounds. As a result, malicious helpers will be discouraged to join the SC as it is very unlikely that they will be able to stay at the SC. For $p_0 \approx 0.1$ and $p_0 \approx 0.9$, Spass might benefit from lossy compression. But, it takes more verification rounds to detect all malicious helpers. Yet, Spass achieves high sensing accuracy in terms of p_d and p_f, the latter of which determines the net utility of the SUN. Hence, despite the existence of malicious helpers, an SUN can maintain a high net profit for $p_0 \approx 0.9$.

E. Performance of Spass under a realistic PU model

To evaluate the performance of our scheme under a more realistic PU model, we use the data provided in [29] which shows the change in the downlink traffic of an LTE eNodeB. The normalized load depicted in Fig.12a represents the fraction

of the OFDM resource blocks scheduled for the cell traffic. We use the normalized load as $(1 - p_0)$ in our model and generate PU activity using this probability for a certain time period. Each malicious helper observes the channel for an initial period corresponding to 10% of the whole trace and computes the average load of the channel. During this sensing phase, malicious helpers do not participate in Spass. Using this average load information, a malicious helper generates its sensing report. Each malicious helper observes a different period of the PU channel which is randomly picked from the trace. Consequently, malicious helpers might have a different perception of the average PU load. Also note that a single value for PU traffic falls short of representing the PU activity which shows spikes as in Fig.12a. On the other hand, the honest helpers operate as usual; they sense the spectrum and report their sensing outcome which might be inaccurate with some probability, i.e., $p_d \sim U(0.80, 0.95)$ and $p_f \sim U(0.05, 0.15)$. We assume $N = 40$, $H = 8$ helpers, and $H_m = \{5, 15\}$ independent malicious attackers.

As Fig.12b shows, the SC has a decreasing number of malicious helpers with each verification round. For lower H_m, almost all malicious helpers are detected after 5 rounds. For $H_m = 15$, the SC might include the malicious helpers (5-6%) whose observation gives a reasonable estimate of p_0. Also, note that the considered PU model in Fig.12a operates during some time period in the region where CHI is not expected to be highly accurate due to very similar honest-honest and honest-malicious distance (cf. to Fig. 6). In agreement with our analysis in Fig.11c and Fig.11d, we observe that higher compression factor results in a better performance in terms of malicious helper detection accuracy which comes also with a higher probability of blacklisting honest helpers (not plotted). Fig.12c shows the increase in detection accuracy with each verification round facilitated by lower number of malicious helpers remaining in the system. For $H_m = 15$ and $\beta = \{1, 10\}$, detection accuracy stabilizes around 89% while the false alarm is zero.

X. Conclusions

This paper introduces Spass which is a proposal for realizing spectrum sensing as a service in an untrusted and decentralized setting. Spass offers opportunistic spectrum discovery with high sensing accuracy to the MNOs who want to extend their licensed spectrum with secondary spectrum. Moreover, it offers payments to the sensing nodes as a compensation for their sensing service for the MNOs. Spass achieves its promises via smart contracts running on a blockchain network. We have examined thoroughly the entailed costs and impact of various parameters including cost of using Ethereum, probability that the PU channel is idle, and the number of malicious helpers. Our analysis shows that a feasible business model to the MNOs under a wide range of settings can be provided. As future work, we will consider malicious helpers who might optimize their strategy to remain undetected.

References

[1] "President's Chamber decision of 14 May 2018 on the order for and choice of proceedings for the award of spectrum in the 2 GHz and 3.6 GHz bands for mobile/fixed communication networks (MFCN)." [Online]. Available: https://www.bundesnetzagentur.de/, accessed on March 16, 2019

[2] M. Aditya, A. Raghuvanshi, and G. S. Kasbekar, "Price competition in spectrum markets: How accurate is the continuous prices approximation?" *IEEE Trans. on Cognitive Comm. and Nw.*, vol. 4, no. 4, pp. 773–86, 2018.

[3] A. Chakraborty *et al.*, "Spectrum patrolling with crowdsourced spectrum sensors," in *IEEE Conf. on Computer Comm.(INFOCOM)*, 2018.

[4] A. Chakraborty, M. S. Rahman, H. Gupta, and S. R. Das, "Specsense: Crowdsensing for efficient querying of spectrum occupancy," in *IEEE Conf. on Computer Comm. (INFOCOM)*, 2017.

[5] R. Zhang, J. Zhang, Y. Zhang, and C. Zhang, "Secure crowdsourcing-based cooperative spectrum sensing," in *IEEE INFOCOM*, 2013.

[6] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[7] N. Szabo, "The idea of smart contracts," 1997. [Online]. Available: http://szabo.best.vwh.net/smart contracts idea.html

[8] S. Bayhan, A. Zubow, and A. Wolisz, "Spass: Spectrum sensing as a service via smart contracts," in *IEEE DYSPAN*, 2018.

[9] X. Jin and Y. Zhang, "Privacy-preserving crowdsourced spectrum sensing," *IEEE/ACM Trans. on Nw.*, vol. 26, no. 3, pp. 1236–49, 2018.

[10] A. Nika, Z. Zhang, X. Zhou, B. Y. Zhao, and H. Zheng, "Towards commoditized real-time spectrum monitoring," in *ACM Workshop on Hot Topics in Wireless*, ser. HotWireless'14, 2014, pp. 25–30.

[11] A. Saeed, K. A. Harras, E. Zegura, and M. Ammar, "Local and low-cost white space detection," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 503–516.

[12] D. Yuan, G. Li, Q. Li, and Y. Zheng, "Sybil defense in crowdsourcing platforms," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1529–1538.

[13] W. Feng *et al.*, "A survey on security, privacy, and trust in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2971–2992, 2017.

[14] N. Marchang, A. Taggu, and A. K. Patra, "Detecting byzantine attack in cognitive radio networks by exploiting frequency and ordering properties," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 816–824, 2018.

[15] M. Li *et al.*, "CrowdBC: A Blockchain-based Decentralized Framework for Crowdsourcing," *IEEE Trans. on Parallel and Distr. Sys.*, 2018.

[16] X. Ying, S. Roy, and R. Poovendran, "Pricing mechanisms for crowdsensed spatial-statistics-based radio mapping," *IEEE Trans. on Cognitive Comm. and Nw.*, vol. 3, no. 2, pp. 242–254, June 2017.

[17] S. Delgado-Segura, C. Tanas, and J. Herrera-Joancomartí, "Reputation and reward: two sides of the same bitcoin," *Sensors*, vol. 16, no. 6, p. 776, 2016.

[18] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui, "Privacy preserving and cost optimal mobile crowdsensing using smart contracts on blockchain," *IEEE Int. Conf. on Mobile Ad-hoc and Sensor Sys.*, 2018.

[19] Y. Lu, Q. Tang, and G. Wang, "ZebraLancer: Private and Anonymous Crowdsourcing System atop Open Blockchain," in *IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, July 2018, pp. 853–865.

[20] H. Li and Z. Han, "Catch me if you can: An abnormality detection approach for collaborative spectrum sensing in CRNs," *IEEE Trans. on Wireless Communications*, vol. 9, no. 11, pp. 3554–3565, 2010.

[21] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[22] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 254–269.

[23] S. Hu *et al.*, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," *IEEE INFOCOM*, 2018.

[24] Ofcom, "Enabling opportunities for innovation shared access to spectrum supporting mobile technology," Dec.18, 2018. [Online]. Available: https://www.ofcom.org.uk/, accessed on March 16, 2019

[25] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: A state of the art survey," *IEEE Communications Surveys & Tutorials*, 2018.

[26] K. W. Sung, S.-L. Kim, and J. Zander, "Temporal spectrum sharing based on primary user activity prediction," *IEEE Transactions on Wireless Communications*, vol. 9, no. 12, pp. 3848–3855, 2010.

[27] GSMA, "5g spectrum gsma public policy position," Nov. 2018.

[28] S. R. Kulkarni, "Chapter 8: Information, entropy, and coding," https://bit.ly/2zXAqE0, November 2018.

[29] N. Bui and J. Widmer, "Data-driven evaluation of anticipatory networking in LTE networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 10, pp. 2252–2265, 2018.

Suzan Bayhan is a Docent in Computer Science at University of Helsinki since March 2017. Suzan got her Ph.D. in Computer Engineering from Bogazici University in 2012 and worked as a postdoctoral researcher at University of Helsinki between 2012-2016, and as a senior researcher at TU Berlin between 09/2016-09/2019. She was a visiting researcher at Princeton University (04/2016) and Aalto University (03/2019). She received Google EMEA Anita Borg scholarship in 2009, a Best Paper Award at ACM ICN 2015, and acted as a N2Women mentoring co-chair during 2017-2018. Suzan conducted research on mobile opportunistic networks, cognitive radio and spectrum sharing, and information-centric networks. Her current research interests include resource allocation for wireless networks, spectrum sharing, and edge computing. She will join University of Twente as an Assistant Professor on September 2019.

Anatolij Zubow received his M.Sc. in computer science (2004) and Ph.D.(2009) from Humboldt University Berlin. He is Interim (Gast-) Professor of Electrical Engineering and Computer Science at the chair for Telecommunication Networks (TKN), Technische Universität Berlin, since October 2018. His research interests are in architectures and protocols of wireless communication networks as well as in protocol engineering with impact on performance and QoS aspects. Recently he is focusing mainly on coexistence of heterogeneous wireless technologies in unlicensed spectrum, high-performance IEEE 802.11 networks, software-defined wireless networking and ultra-reliable low latency communication.

Piotr Gawłowicz is a researcher at TKN Group at TU Berlin, Germany. He received his M.Sc. and B.Sc. degrees in telecommunications from AGH University of Science and Technology, Kraków, Poland in 2014 and 2012, respectively. In the past, he visited Panasonic R&D Center Germany and Nokia R&D in Wrocław, Poland, where he worked on projects for future mobile networks and enhancements for LTE-A. Under the Google Summer of Code 2014 program, he contributed to the development of the ns-3 network simulator. At TKN, he has been involved in national and European research projects. His research interests include coexistence and collaboration of heterogeneous wireless systems, cross-technology communication, and recently reinforcement learning in the networking area. He has extensive experience in simulating and prototyping of wireless solutions.

Adam Wolisz received his degrees (Diploma 1972, Ph.D. 1976, Habil. 1983) from Silesian University of Technology, Gliwice. He joined Technical University of Berlin in 1993, where he is a Chaired Professor in telecommunication networks and Executive Director of the Institute for Telecommunication Systems. He is also an Adjunct Professor at the Department of Electrical Engineering and Computer Science, University of California, Berkeley. He has been a Fellow at the Einstein Center Digital Future (ECDF) since October 2018. His research interests are in architectures and protocols of communication networks. He is an IEEE Senior Member.