Trustworthiness Verification and Integrity Testing for Wireless Communication Systems

Holger Boche[*], Rafael F. Schaefer[†], H. Vincent Poor[‡], and Gerhard P. Fettweis[§]

[*] Institute of Theoretical Information Technology, Technische Universität München
Centers 6G-life & CASA & MCQST

[†] Chair of Communications Engineering and Security, Universität Siegen
Center for Sensor Systems (ZESS), University of Siegen

[‡] Dept. of Electrical and Computer Engineering, Princeton University

[§] Vodafone Chair for Mobile Communication Systems, Technische Universität Dresden
Barkhausen Institut, Dresden
Centers 6G-life & CeTI & 5G^{++}Lab Germany & EKFZ

***Abstract*—Trustworthiness verification and integrity testing have been identified as key challenges for the sixth generation (6G) of mobile networks and its variety of envisioned features. In this paper, these issues are addressed from a fundamental, algorithmic point of view. For this purpose, the concept of Turing machines is used which provides the fundamental performance limits of digital computers. It is shown that, in general, trustworthiness and integrity cannot be verified by Turing machines and therewith by today's digital computers. In addition, the trustworthiness problem is further shown to be non-Banach-Mazur computable which is the weakest form of computability. Neuromorphic computing has an enormous potential to overcome the limitations of today's digital hardware and, accordingly, it is interesting to study the issues of trustworthiness verification and integrity testing also for such powerful computing models. In particular, as considerable progress in the hardware design for neuromorphic computing has been achieved.**

I. Introduction

As envisioned, the sixth generation (6G) of mobile networks will provide a variety of new features including joint communications and sensing, post Shannon communication, embedded and post quantum security, and many more [1].

These features impose new challenges on the design of wireless communication systems. In particular, the Tactile Internet will allow not only the control of data, but also of physical and virtual objects. With such applications comes the need to address the trustworthiness of the system and its services. This has to be done in a new context as it is envisioned that 6G will allow for localization of unprecedented precision, sensing that goes beyond radio and camera sensing, and also gesture recognition including emotions.

However, the significant improvement in sensing capabilities also enables advanced spying opportunities, and the question is then how these new (sensing) services can be provided without compromising legal and societal privacy requirements. As a consequence, trustworthiness is a key enabler for 6G and must be understood including privacy, security, integrity, resilience, reliability, availability, and device independence [2]. The theory for the formalization of trustworthiness must comprise different fields and many central questions and issues are open to date, see e.g. [2]. In this paper, we address the specific issue of algorithmic verification of trustworthiness based on digital hardware.

The design of a computing platform for trustworthiness poses several significant challenges. For example, standards of cellular systems are written in English text. As such standards are not machine readable for formal verification purposes, one actually needs a description that allows for formal verification and cross-check of implementation code. Such a process of formal verification must extend to software patches and updates as well, ensuring that the cellular infrastructure provides only those services it is designed for and not serving other interests. To this end, two conditions need to be satisfied: readability of the system description and implementation as well as formal verification. This is of crucial interest and importance.

Today's approaches to performance evaluation and verification are based on theoretical analyses and numerical simulations. Due to the high complexity of modern wireless communication systems, analytical approaches quickly reach their limits. For example, the capacity region of the broadcast channel remains unknown to date although it is a central element of each wireless communication system.

This work of H. Boche was supported in part by the German Federal Ministry of Education and Research (BMBF) within the national initiative on 6G Communication Systems through the research hub *6G-life* under Grant 16KISK002, within the national initiative for *Post Shannon Communication (NewCom)* under Grant 16KIS1003K, and the project *Hardware Platforms and Computing Models for Neuromorphic Computing (NeuroCM)* under Grant 16ME0442. It has further received funding by the Bavarian Ministry of Economic Affairs, Regional Development and Energy as part of the project 6G Future Lab Bavaria as well as in part by the German Research Foundation (DFG) within Germany's Excellence Strategy EXC-2092 – 390781972. This work of R. F. Schaefer was supported in part by the BMBF within NewCom under Grant 16KIS1004 and in part by the DFG under Grant SCHA 1944/6-1. This work of H. V. Poor was supported by the U.S. National Science Foundation under Grant CCF-1908308. This work of G. P. Fettweis was supported in part by the BMBF within 6G-Plattform under Grant 16KISK051.

It is clear that this highly depends on the particular hardware platform that is used for the integrity testing. In particular, numerical simulations are usually done on digital computers. To address this issue from a fundamental algorithmic point of view, we use the concept of a *Turing machine* [3–5] and the corresponding *computability framework*. The Turing machine is a mathematical model of an abstract machine that manipulates symbols on a strip of tape according to certain given rules. It can simulate any given algorithm and therewith provides a simple but very powerful model of computation. Turing machines have no limitations on computational complexity, unlimited computing capacity and storage, and execute programs completely error-free. They are further equivalent to the von Neumann-architecture without hardware limitations and the theory of recursive functions, cf. also [6–10]. Accordingly, Turing machines provide fundamental performance limits for today's digital computers and are the ideal concept to study whether or not trustworthiness verification and integrity testing can be performed algorithmically in principle (without putting any constraints on the computational complexity).

Trustworthiness verification and integrity testing of software are becoming particularly relevant as there is the recent trend towards shifting functionalities from the physical layer to higher layers by enabling software-focused solutions. The particular objective here is to create an infrastructure that is capable of interconnecting highly heterogeneous networks to support several different verticals. The concepts of *software-defined networking (SDN)* [11] and *network function virtualization (NFV)* [12] have been identified as key enablers at all levels. The aim of such network virtualization is to provide software-based solutions for functions, protocols, and operations such that they run on general purpose hardware and do not require specialized hardware anymore. This had laid the groundwork for novel paradigms such as cloud and edge/fog computing, unique and reconfigurable SDN-NFV architectures, and end-to-end network slicing [13, 14].

Softwarization shows great potential in terms of flexibility and cost reduction. However, at the same time it provides a larger attack surface including all interaction points an attacker can reach. This includes interfaces, protocols, and services both in software and hardware. This further shows the necessity of formal verification to verify the trustworthiness and the integrity of the system.

II. Computability Framework

We first introduce the computability framework based on Turing machines. For this we need some basic definitions and concepts of computability which are briefly reviewed. The concept of computability and computable real numbers was first introduced by Turing in [3] and [4]. This provides the needed background and notation to rigorously formalize the readability and formal verification of wireless communication systems on digital hardware platforms. The main crucial point is that wireless systems have to deal with real-valued channels and fading statistics that reflect the physical propagation of signals described by the Maxwell Equations.

Recursive functions $f : \mathbb{N} \to \mathbb{N}$ map natural numbers into natural numbers and are exactly those functions that are computable by a Turing machine. They are the smallest class of partial functions that includes the primitive functions (i.e., constant function, successor function, and projection function) and is further closed under composition, primitive recursion, and minimization. For a detailed introduction, we refer the reader to [15] and [16]. With this, we call a sequence of rational numbers $(r_n)_{n\in\mathbb{N}}$ a *computable sequence* if there exist recursive functions $a, b, s : \mathbb{N} \to \mathbb{N}$ with $b(n) \neq 0$ for all $n \in \mathbb{N}$ and

$$r_n = (-1)^{s(n)} \frac{a(n)}{b(n)}, \qquad n \in \mathbb{N}; \tag{1}$$

cf. [15, Def. 2.1 and 2.2] for a detailed treatment. A real number x is said to be computable if there exists a computable sequence of rational numbers $(r_n)_{n\in\mathbb{N}}$ and a recursive function φ such that we have for all $M \in \mathbb{N}$

$$|x - r_n| < 2^{-M} \tag{2}$$

for all $n \geq \varphi(M)$. Thus, the computable real x is represented by the pair $((r_n)_{n\in\mathbb{N}}, \varphi)$. This form of convergence with a computable control of the approximation error is called *effective convergence*. Note that if a computable sequence of real numbers $(x_n)_{n\in\mathbb{N}}$ converges effectively to a limit x, then x is a computable real number, cf. [16]. Furthermore, the set $\mathbb{R}_c$ of all computable real numbers is closed for addition, subtraction, multiplication, and division (excluding the division by zero). We denote the set of computable real numbers by $\mathbb{R}_c$. Based on this, we define the set of computable probability distributions $\mathcal{P}_c(\mathcal{X})$ as the set of all probability distributions $P_X \in \mathcal{P}(\mathcal{X})$ such that $P_X(x) \in \mathbb{R}_c$, $x \in \mathcal{X}$. Further, let $\mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ be the set of all computable channels, i.e., for a channel $W : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ we have $W(\cdot|x) \in \mathcal{P}_c(\mathcal{Y})$ for every $x \in \mathcal{X}$.

Definition 1. A function $f : \mathbb{R}_c \to \mathbb{R}_c$ is called *Borel-Turing computable* if there exists an algorithm that gets for every x an arbitrary representation $((r_n)_{n\in\mathbb{N}}, \varphi)$ for it as input and then computes a representation $((\hat{r}_n)_{n\in\mathbb{N}}, \hat{\varphi})$ for $f(x)$.

Turing's definition of computability conforms to the definition of Borel computability above. Here, we particularly consider the notion of a *computable continuous function*, cf. [16, Def. A].

Definition 2 ([16]). Let $\mathbb{I}_c = [0,1] \cap \mathbb{R}_c$ be the computable unit interval. A function $f : \mathbb{I}_c \to [0,1]$ is called *computable continuous* if:

1) f is *sequentially computable*, i.e., f maps every computable sequence $(x_n)_{n\in\mathbb{N}}$ of points $x_n \in \mathbb{I}_c$ into a computable sequence $(f(x_n))_{n\in\mathbb{N}}$ of real numbers,
2) f is *effectively uniformly continuous*, i.e., there is a recursive function $d : \mathbb{N} \to \mathbb{N}$ such that for all $x, y \in \mathbb{I}_c$ and all $N \in \mathbb{N}$ with $\|x - y\| \leq \frac{1}{d(N)}$ it holds that $|f(x) - f(y)| \leq \frac{1}{2^N}$.

Remark 1. There are other forms of computability such as *Markov computability* and *Banach-Mazur computability*, of which the latter one is the weakest form of computability. In particular, Borel or Markov computability both imply Banach-Mazur computability, but not vice versa. For an overview of the logical relations between different notions of computability we again refer to [6] and, for example, [5].

We further need the concepts of a recursive set and a recursively enumerable set as, for example, defined in [15].

Definition 3. A set $\mathcal{A} \subset \mathbb{N}$ is called *recursive* if there exists a computable function f such that $f(x) = 1$ if $x \in \mathcal{A}$ and $f(x) = 0$ if $x \notin \mathcal{A}$.

Definition 4. A set $\mathcal{A} \subset \mathbb{N}$ is *recursively enumerable* if there exists a recursive function whose range is exactly $\mathcal{A}$.

We have the following properties which will be crucial later for proving the desired results; cf. also [15] for further details.

- $\mathcal{A}$ is recursive is equivalent to $\mathcal{A}$ is recursively enumerable and $\mathcal{A}^c$ is recursively enumerable.
- There exist recursively enumerable sets $\mathcal{A} \subset \mathbb{N}$ that are not recursive, i.e., $\mathcal{A}^c$ is not recursively enumerable. This means there are no computable, i.e., recursive, functions $f : \mathbb{N} \to \mathcal{A}^c$ with $[f(\mathbb{N})] = \mathcal{A}^c$.

Turing machines are extremely powerful compared to state-of-the-art digital signal processing (DSP) and field gate programmable array (FPGA) platforms and even current supercomputers. It is the most general computing model and is even capable of performing arbitrary exhaustive search tasks on arbitrary large but finite structures. The complexity can even grow faster than double-exponentially with the set of parameters of the underlying communication system (such as time, frequencies, transmit power, modulation scheme, etc.).

III. Problem Formulation and Main Result

In wireless communication systems, transmitters and receivers are separated in space. As a consequence, the attack surface not only includes the hardware and software implementations such as the antenna design, power amplifier, baseband processors, software processors, etc., but also the communication channels between the users. These are integral parts of the communication system and are subject to various attacks including eavesdropping, jamming attacks, denial-of-service (DoS) attacks, non-legitimate sensing and others.

We want to study whether or not trustworthiness verification and integrity testing are possible when taking into account channels with practically relevant properties such as certain signal-to-noise ratios (SNRs) or fading statistics. For this purpose, we will introduce next some necessary notation and concepts that will allow us to describe the channels with their properties such that Turing machines can use these as inputs for trustworthiness verification and integrity testing.

We assume a fixed transmitter-receiver pair and a practically relevant channel such as an additive white Gaussian noise (AWGN) channel, Rayleigh fading channel, Nakagami-m channel, etc. We further assume a family of channels that is practically relevant and that depends on a single parameter such as the SNR for AWGN channels or Rayleigh parameter for Rayleigh fading channels. The case when the channel law or fading statistics depend on more than one parameter is discussed later after Theorem 1.

Now, we assume that the practically relevant parameters of a family of channels is described by a certain interval $\mathbb{I} = [a, b] \subset \mathbb{R}$, e.g., for AWGN channels the interval $\mathbb{I}$ describes the relevant SNR regime. We further assume that the corresponding mathematical relations and functions that describe the channel are continuous in the parameter interval $\mathbb{I}$.

The main goal of this work is study the problem of trustworthiness verification and integrity testing. We observe that this is a classification problem depending on the channel parameters of all users. To this end, this does not necessarily mean that we only have to decide on "trustworthy" or "non-trustworthy", but rather finitely many levels of trustworthiness can be introduced depending on the context and the particular requirements. The same applies to the integrity testing. Here, the hardware and protocol specifications are fixed and given and the task is to check whether or not the particular hardware and protocols realize the desired behavior for the practically relevant communication channels.

A trustworthiness verification task is said to be trivial, if for all relevant channels, i.e., all relevant channel parameters, the trustworthiness is either always satisfied or always not satisfied.

We have seen in [17] and [18] that the verification of resilience is trivial for a single transmitter-receiver pair if and only if it is guaranteed that there are no jamming attacks on the transmission. Since then, resilience against attacks is trivially satisfied. Otherwise, there is always a practically relevant set of channel parameters for which the attacker can successfully launch a DoS attack. Note that this remains true for quantum communication systems.

In this work, we address a corresponding question for trustworthiness verification and integrity testing. In particular, we want to understand whether or not these tasks can be solved algorithmically by a Turing machine.

In the following, we consider a wireless communication system that depends the parameters of the corresponding communication channel. Let us assume that we have K of these parameters, i.e., $(\tau_1, \tau_2, ..., \tau_K) \in \mathbb{I}_1 \times \mathbb{I}_2 \times ... \times \mathbb{I}_K$ with $K \in \mathbb{N}$ and $\mathbb{I}_l = [a_l, b_l]$, $a_l, b_l \in \mathbb{R}$, $a_l < b_l$, $1 \leq l \leq K$. As discussed above, within the AWGN model these parameters $\mathbb{I}_l$ could represent the admissible transit power range for the l-th transmitter-receiver pair or the possible noise power at the receiver. For fading channels, these parameters could represent the possible fading statistics.

We further assume that all possible K-tuples $(\tau_1, \tau_2, ..., \tau_K) \in \mathbb{I}_1 \times \mathbb{I}_2 \times ... \times \mathbb{I}_K \subset \mathbb{R}^K$ are practically relevant. As a consequence, any convex combination is relevant as well so that for

$$(\tau_1^{(1)}, \tau_2^{(1)}, ..., \tau_K^{(1)}) \in \mathbb{I}_1 \times \mathbb{I}_2 \times ... \times \mathbb{I}_K,$$

$$(\tau_1^{(2)}, \tau_2^{(2)}, ..., \tau_K^{(2)}) \in \mathbb{I}_1 \times \mathbb{I}_2 \times ... \times \mathbb{I}_K,$$

and all $\lambda \in [0,1]$, the corresponding parameters

$$\big((1-\lambda)\tau_1^{(1)} + \lambda\tau_1^{(2)}, ..., (1-\lambda)\tau_K^{(1)} + \lambda\tau_K^{(2)}\big)$$

are practically relevant as well. In the following, we further use the notation $\boldsymbol{\tau} = (\tau_1, \tau_2, ..., \tau_K)$ for simplicity.

Let $\mathcal{S} = \{s_1, s_2, ..., s_L\}$ be a finite state set describing L trustworthiness levels. Let F_{trust} be a function that maps a set $\mathbb{I}_1 \times \mathbb{I}_2 \times ... \times \mathbb{I}_K$ of parameters to $\mathcal{S}$, i.e., $F_{\text{trust}} : \mathbb{I}_1 \times \mathbb{I}_2 \times ... \times \mathbb{I}_K \to \mathcal{S}$. The function F_{trust} need not necessarily be defined for $\mathbb{R}^K$.

Definition 5. Let $F_{\text{trust}} : \mathbb{I}_1 \times \mathbb{I}_2 \times ... \times \mathbb{I}_K \to \mathcal{S} = \{s_1, s_2, ..., s_L\}$ be a function that maps every parameters $\boldsymbol{\tau}$ to a corresponding state $F_{\text{trust}}(\boldsymbol{\tau}) \in \mathcal{S}$. A trustworthiness verification problem is said to be *non-trivial* if there exist $\boldsymbol{\tau}^{(1)}$ and $\boldsymbol{\tau}^{(2)}$ that lead to different states, i.e., $F_{\text{trust}}(\boldsymbol{\tau}^{(1)}) \neq F_{\text{trust}}(\boldsymbol{\tau}^{(2)})$.

In the following, we want to study trustworthiness verification on Turing machines. From the introduction and discussion in Section II, it is clear that Turing machines can only work with computable real numbers and parameters as well as computable channels as inputs. Accordingly, let $\mathfrak{T}_{F_{\text{trust}}}$ be a Turing machine for the classification task F_{trust}, i.e., $\mathfrak{T}_{F_{\text{trust}}}$ is defined for the set of computable parameters $\mathbb{I}_{c,1} \times \mathbb{I}_{c,2} \times ... \times \mathbb{I}_{c,K} \subset \mathbb{R}_c^K$. Note that $\mathfrak{T}_{F_{\text{trust}}}$ need not necessarily be defined for channels from $\mathbb{R}_c^K$ that are not in the set of computable parameters. The following result shows that, in general, there is no Turing machine $\mathfrak{T}_{F_{\text{trust}}}$ or algorithm that can solve a non-trivial classification problem F_{trust}.

Theorem 1. For all wireless communication systems for which trustworthiness verification is non-trivial, the trustworthiness cannot be verified by Turing machines. The same remains true for the integrity testing.

Sketch of Proof: The key idea of the proof is the following: If a non-trivial trustworthiness problem would be verifiable by a Turing machine, then we would be able to construct another Turing machine based on the previous one that can solve the halting problem for a certain set $\mathcal{A} \subset \mathbb{N}$. However, for this certain subset $\mathcal{A}$ it is known that the halting problem cannot be solved by Turing machines. This will then prove the desired result by contradiction. A more detailed sketch of the proof is given in the appendix. ■

Some discussion is in order. In Theorem 1 we have considered channel parameters of the form $\mathbb{I}_1 \times ... \times \mathbb{I}_K$. From a practical point of view, this is the simplest form of dependency. We note that the proof of Theorem 1 is general in the sense that it allows arbitrary convex sets of parameters. For sake of presentation, we show that already for the simplest case of dependencies the trustworthiness verification and integrity testing cannot be solved on digital computers.

In Theorem 1, only digital hardware platforms are used. One may ask the question what happens if the trustworthiness verification and integrity testing are performed on other computing platforms. There is the concept of Blum-Shub-Smale (BSS) machines [19–21] which is the second-most used computing model after Turing machines. It is more powerful than Turing machines as it is able to work exactly with real numbers. Obviously, it is not clear if such a computing machine can be realized with today's hardware technology, since it would require the processing and storage of arbitrary real numbers. It provides the basis for neuromorphic or quantum computing.

In this context, jamming and particularly DoS attacks have been studied in [17, 18] and it has been shown for different communication scenarios that DoS attacks cannot be detected by Turing machines.

The verification of whether or not DoS attacks can be performed by adversaries is a central problem for the resilience verification and therewith also for trustworthiness. Let us consider the scenario with a transmitter (Alice), a receiver (Bob), and a Jammer that tries to disturb the transmission from Alice to Bob by transmitting an own jamming sequence. A malevolent Jammer will try to perform a DoS attack by completely disrupting the communication.

We briefly discuss the following example. Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ be discrete alphabets of Alice, Bob, and the Jammer, respectively. The receive probability of $y \in \mathcal{Y}$ at Bob is specified by $W(y|x,z)$ and depends on the legitimate input of Alice and the jamming input $z \in \mathcal{Z}$. In [17] and [18], the detectability of DoS attacks on Turing machines is studied. The potential Turing machine for the detection of such attacks and therewith for the verification of resilience gets the communication channel as input and then decides whether or not a DoS attack is possible.

As an example, for $|\mathcal{Y}| = 3$, $|\mathcal{X}| = 2$, $|\mathcal{Z}| = 2$ there are channels for which a DoS attack is possible, cf. [17, 18] for a detailed construction of such channels. This means that we have a non-trivial classification problem for resilience in this case and therewith a non-trivial verification problem for trustworthiness. As a conclusion, the verification of resilience with Turing machines is not possible in this case. Note that numerous techniques and approaches for the detection of DoS attacks have been proposed in the literature. However, [17] shows that these techniques are not sufficient for a guaranteed detection of DoS attacks.

On the other hand, BSS machines enable the detection of DoS attacks on classical [22] and quantum communication systems [23]. Finally, we note that currently, there is great effort in the semiconductor industry to develop chips that enable neuromorphic computing.

Next, we will further discuss a stronger result from which Theorem 1 then follows. To this end, we consider a wireless communication system with parameters $\mathbb{I}_1 \times ... \times \mathbb{I}_K$ and an arbitrary trustworthiness problem F_{trust} with $|\mathcal{S}|$ possible trustworthiness levels, where $|\mathcal{S}| \geq 2$ being finite. Then, $F_{\text{trust}} : \mathbb{I}_1 \times ... \times \mathbb{I}_K \to \mathcal{S}$ is a well defined function.

Theorem 1 says that no non-trivial function F_{trust} is Turing computable, i.e., there is no algorithm (or Turing machine) that can compute the output $F_{\text{trust}}(\boldsymbol{\tau}) \in \mathcal{S}$ for an input $\boldsymbol{\tau} \in \mathbb{I}_1 \times ... \times \mathbb{I}_K$. We will show the following result.

Theorem 2. Let F_{trust} be an arbitrary non-trivial trustworthiness problem for a wireless communication system. Then F_{trust} is not Banach-Mazur computable.

Sketch of Proof: The key idea of the proof is the following: We assume that there are a set of parameters $\mathbb{I}_{c,1} \times ... \times \mathbb{I}_{c,K}$ and a set of trustworthiness levels $\mathcal{S}$ of a wireless system for which the corresponding function F_{trust} is Banach-Mazur computable. For this function, we can then find a suitable recursively enumerable set $\mathcal{A} \subset \mathbb{N}$ that is not recursive. This implies the existence of a Turing machine $\mathfrak{T}_{\mathcal{A}}$ that decides for each $k \in \mathbb{N}$ whether $k \in \mathcal{A}$ or $k \notin \mathcal{A}$. But this is not possible establishing a contradiction. The detailed proof is omitted due to space constraints. ∎

Since every Turing computable function is also Banach-Mazur computable (but not vice versa), Theorem 2 immediately proves Theorem 1.

Interestingly, it can be shown this task is not solvable on a Turing machine, but can be solved on suitable neuromorphic computers such as BSS machines. Therefore, it would be interesting to study whether or not such neuromorphic computers are capable of verifying the trustworthiness, i.e., by computing the function F_{trust}.

IV. Discussion

In this paper, we have studied the problems of trustworthiness verification and integrity testing for computable channels and computable channel parameters respectively. The restriction to computable real numbers is necessary as any digital hardware platform (or Turing machine) requires computable real numbers as inputs. One may think that this restriction is the reason for the "no-go" result in Theorem 1.

For the theoretical analysis, more powerful Turing machines have been proposed that obtain their inputs from an oracle. Here, for each real parameter, the Turing machine obtains a converging sequence of rational numbers as input. Such a sequence exists always, but cannot be characterized algorithmically in general. However, also in this case, trustworthiness verification is not possible on Turing machines due to the following reasoning: If one uses only computable inputs for an oracle Turing machine, then it is not necessary at all to use an oracle. We obtain the very same results and, as a consequence, oracle Turing machines does not help to make trustworthiness verification and integrity testing possible.

To date, digital hardware is the only computing hardware platform for practical systems. Unfortunately, its capability for the requirements of provable, i.e., guaranteed, performance such as correctness for decision processes is rather limited. For example, for remote state estimation and stabilization via noisy communication channels, a Turing machine cannot decide whether or not an unstable linear system can be controlled [24, 25]. This is a central task for digital twins.

We already have seen that DoS attacks can be detected by BSS machines and therewith by more powerful neuromorphic computing hardware. In addition, the constructions used in the proof of Theorem 1 that are not solvable on Turing machines, become solvable for neuromorphic computing models. In [26] it has been shown that BSS machines are always capable of deciding whether or not a linear system can be stabilized via noisy communication channels. Therefore, it is interesting to study the trustworthiness verification problem also for more powerful computing models and new technologies. In particular, as there has been significant progress recently by the industry in the development of neuromorphic computing hardware platforms.

References

[1] G. P. Fettweis and H. Boche, "6G: The personal Tactile Internet - and open questions for information theory," *IEEE BITS the Information Theory Magazine*, vol. 1, no. 1, pp. 71–82, Aug. 2021.

[2] ——, "On 6G and trustworthiness," *Proceedings of ACM*, 2022, invited, to be published.

[3] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proc. London Math. Soc.*, vol. 2, no. 42, pp. 230–265, 1936.

[4] ——, "On computable numbers, with an application to the Entscheidungsproblem. A correction," *Proc. London Math. Soc.*, vol. 2, no. 43, pp. 544–546, 1937.

[5] K. Weihrauch, *Computable Analysis - An Introduction*. Berlin, Heidelberg: Springer-Verlag, 2000.

[6] J. Avigad and V. Brattka, "Computability and analysis: The legacy of Alan Turing," in *Turing's Legacy: Developments from Turing's Ideas in Logic*, R. Downey, Ed. Cambridge, UK: Cambridge University Press, 2014.

[7] K. Gödel, "Die Vollständigkeit der Axiome des logischen Funktionenkalküls," *Monatshefte für Mathematik*, vol. 37, no. 1, pp. 349–360, 1930.

[8] ——, "On undecidable propositions of formal mathematical systems," *Notes by Stephen C. Kleene and Barkely Rosser on Lectures at the Institute for Advanced Study, Princeton, NJ*, 1934.

[9] S. C. Kleene, *Introduction to Metamathematics*. Van Nostrand, New York: Wolters-Noordhoffv, 1952.

[10] M. Minsky, "Recursive unsolvability of Post's problem of 'tag' and other topics in theory of Turing machines," *Ann. Math.*, vol. 74, no. 3, pp. 437–455, 1961.

[11] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, Third Quarter 2014.

[12] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, First Quarter 2016.

[13] M. Richart, J. Baliosian, J. Serrat, and J.-L. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 462–476, Sep. 2016.

[14] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network slicing for 5G: Challenges and opportunities," *IEEE Internet Comput.*, vol. 21, no. 5, pp. 20–27, Sep./Oct. 2017.

[15] R. I. Soare, *Recursively Enumerable Sets and Degrees*. Berlin, Heidelberg: Springer-Verlag, 1987.

[16] M. B. Pour-El and J. I. Richards, *Computability in Analysis and Physics*. Cambridge: Cambridge University Press, 2017.

[17] H. Boche, R. F. Schaefer, and H. V. Poor, "Denial-of-service attacks on communication systems: Detectability and jammer knowledge," *IEEE Trans. Signal Process.*, vol. 68, pp. 3754–3768, 2020.

[18] ——, "On the algorithmic solvability of channel dependent classification problems in communication systems," *IEEE/ACM Trans. Netw.*, vol. 29, no. 3, pp. 1155–1168, Jun. 2021.

[19] L. Blum, M. Shub, and S. Smale, "On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines," *Bull. Amer. Math. Soc.*, vol. 21, no. 1, pp. 1–46, Jul. 1989.

[20] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and Real Computation*. New York: Springer Verlag, 1998.

[21] L. Blum, "Computing over the reals: Where Turing meets Newton," *Notices Amer. Math. Soc.*, vol. 51, no. 9, pp. 1024–1034, Oct. 2004.

[22] H. Boche, R. F. Schaefer, and H. V. Poor, "Real number signal processing can detect denial-of-service attacks," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, Toronto, ON, Canada, Jun. 2021, pp. 4765–4769.

[23] H. Boche, M. Cai, H. V. Poor, and R. F. Schaefer, "Detectability of denial-of-service attacks on arbitrarily varying classical-quantum channels," in *Proc. IEEE Int. Symp. Inf. Theory*, Melbourne, Australia, Jul. 2021, pp. 912–917.

[24] H. Boche, Y. Böck, and C. Deppe, "On the semidecidability of remote state estimation and stabilization via noisy communication channels," in *Proc. 60th Conf. Dec. Contr.*, Austin, TX, USA, Dec. 2021.

[25] ——, "On the semidecidability of the remote state estimation problem," *IEEE Trans. Autom. Control*, 2022, will appear.

[26] ——, "Deciding the problem of remote state estimationvia noisy communication channels on real number signal processing hardware," in *Proc. IEEE Int. Conf. Commun.*, Seoul, South Korea, May 2022.

[27] H. Boche, R. F. Schaefer, S. Baur, and H. V. Poor, "On the algorithmic computability of the secret key and authentication capacity under channel, storage, and privacy leakage constraints," *IEEE Trans. Signal Process.*, vol. 67, no. 17, pp. 4636–4648, Sep. 2019.

[28] M. B. Pour-El, "A comparison of five "computable" operators," *Math. Logic Quart.*, vol. 6, no. 15-22, pp. 325–340, 1960.

Appendix

Here, we present a sketch of proof for Theorem 1. The desired result is proven by showing that every non-trivial trustworthiness problem $F_{\text{trust}} : \mathbb{I}_1 \times \mathbb{I}_2 \times \dots \times \mathbb{I}_K \to \mathcal{S}$ is not algorithmically solvable, i.e., there exists no Turing machine $\mathfrak{T}_{F_{\text{trust}}}$ that takes the parameters $\boldsymbol{\tau}$ as inputs and outputs the corresponding state $F(\boldsymbol{\tau}) \in \mathcal{S}$.

We prove the desired result by contradiction. We assume that there is a non-trivial trustworthiness problem $F_{\text{trust}} : \mathbb{I}_1 \times \mathbb{I}_2 \times \dots \times \mathbb{I}_K \to \{s_1, \dots, s_L\}$, $L \in \mathbb{N}$, such that there exists a Turing machine $\mathfrak{T}_{F_{\text{trust}}}$ for which we have $\mathfrak{T}_{F_{\text{trust}}}(\boldsymbol{\tau}) = F_{\text{trust}}(\boldsymbol{\tau})$ for all $\boldsymbol{\tau} \in \mathbb{I}_{c,1} \times \mathbb{I}_{c,2} \times \dots \times \mathbb{I}_{c,K} \subset \mathbb{R}_c^K$.

Further, for $\mathcal{S} = \{s_1, s_2, \dots, s_L\}$ with $L > 2$ we can construct a Turing machine $\tilde{\mathfrak{T}}_{F_{\text{trust}}}$ that solves a non-trivial binary trustworthiness task as follows. For $L > 2$, there exist some $\boldsymbol{\tau}^{(i)}$ and $\boldsymbol{\tau}^{(j)}$ with $F_{\text{trust}}(\boldsymbol{\tau}^{(i)}) = s_i \neq s_j = F(\boldsymbol{\tau}^{(j)})$. Now we choose the state set $\hat{\mathcal{S}} = \mathcal{S}_1 \cup \mathcal{S}_2$ with $\mathcal{S}_1 = \{s_i\}$ and $\mathcal{S}_2 = \bigcup_{j \neq i} \{s_j\}$. We have $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$ and consider the trustworthiness problem $\hat{F}_{\text{trust}} : \mathbb{I}_1 \times \mathbb{I}_2 \times \dots \times \mathbb{I}_K \to \hat{\mathcal{S}}$ with $\hat{F}_{\text{trust}}(\boldsymbol{\tau}) \in \mathcal{S}_1 \Leftrightarrow F_{\text{trust}}(\boldsymbol{\tau}) = s_i$ and $\hat{F}_{\text{trust}}(\boldsymbol{\tau}) \in \mathcal{S}_2 \Leftrightarrow F_{\text{trust}}(\boldsymbol{\tau}) = s_j$ for $s_j \neq s_i$. This yields a non-trivial trustworthiness verification task for the binary case $L = 2$. Thus, if there would exist a Turing machine $\mathfrak{T}_{F_{\text{trust}}}$ for $L > 2$, then the modified Turing machine $\hat{\mathfrak{T}}_{\hat{F}_{\text{trust}}}$ with

$$\hat{\mathfrak{T}}_{\hat{F}_{\text{trust}}}(\boldsymbol{\tau}) = \begin{cases} s_i & \text{if } \mathfrak{T}_{F_{\text{trust}}}(\boldsymbol{\tau}) = s_i \\ s_j & \text{if } \mathfrak{T}_{F_{\text{trust}}}(\boldsymbol{\tau}) \in \{s_1, \dots, s_L\} \backslash \{s_i\} \end{cases} \quad (3)$$

would solve the corresponding binary problem. Without loss of generality, we accordingly assume only $L = 2$ different trustworthiness levels, i.e., $\mathcal{S} = \{s_1, s_2\}$ in the following.

Now, we prove the binary case by contradiction, i.e., we assume that there exists a non-trivial trustworthiness problem $F_{\text{trust}} : \mathbb{I}_1 \times \mathbb{I}_2 \times \dots \times \mathbb{I}_K \to \{s_1, s_2\}$ such that there exists a Turing machine $\mathfrak{T}_{F_{\text{trust}}}$ that correctly outputs $\mathfrak{T}_{F_{\text{trust}}}(\boldsymbol{\tau}) = F_{\text{trust}}(\boldsymbol{\tau})$ for all $\tau \in \mathbb{I}_{c,1} \times \mathbb{I}_{c,2} \times \dots \times \mathbb{I}_{c,K}$. Since F_{trust} is non-trivial, there must exist some $\boldsymbol{\tau}^{(1)}, \boldsymbol{\tau}^{(2)} \in \mathbb{I}_{c,1} \times \mathbb{I}_{c,2} \times \dots \times \mathbb{I}_{c,K}$ with $F_{\text{trust}}(\boldsymbol{\tau}^{(1)}) \neq F_{\text{trust}}(\boldsymbol{\tau}^{(2)})$.

Now, the proof idea is as follows. For $\lambda \in [0,1] \cap \mathbb{R}_c$ we consider for $\boldsymbol{\tau}(\lambda) = (1-\lambda)\boldsymbol{\tau}^{(1)} + \lambda \boldsymbol{\tau}^{(2)}$ the function $F_{\text{trust}}(\boldsymbol{\tau}(\lambda))$. Now, we construct two computable sequences $(\underline{\lambda}_n)_{n \in \mathbb{N}}$ and $(\overline{\lambda}_n)_{n \in \mathbb{N}}$ where the former is monotonically increasing and the latter monotonically decreasing so that both converge to a number $\lambda_* \in [0,1]$ where λ_* is a computable real number. Then, we either have

$$F_{\text{trust}}(\boldsymbol{\tau}(\underline{\lambda}_n)) = F_{\text{trust}}(\boldsymbol{\tau}^{(1)}), \ n \in \mathbb{N}, \quad (4a)$$

$$F_{\text{trust}}(\boldsymbol{\tau}(\lambda_*)) = F_{\text{trust}}(\boldsymbol{\tau}^{(2)}) \quad (4b)$$

or

$$F_{\text{trust}}(\boldsymbol{\tau}(\overline{\lambda}_n)) = F_{\text{trust}}(\boldsymbol{\tau}^{(2)}), \ n \in \mathbb{N}, \quad (5a)$$

$$F_{\text{trust}}(\boldsymbol{\tau}(\lambda_*)) = F_{\text{trust}}(\boldsymbol{\tau}^{(1)}). \quad (5b)$$

For $\lambda = \frac{1}{2}$ we must either have $F_{\text{trust}}(\boldsymbol{\tau}(\frac{1}{2})) = F_{\text{trust}}(\boldsymbol{\tau}^{(1)})$ or $F_{\text{trust}}(\boldsymbol{\tau}(\frac{1}{2})) = F_{\text{trust}}(\boldsymbol{\tau}^{(2)})$.

If the first case is true, then we set $\underline{\lambda}_1 = \frac{1}{2}$ and $\overline{\lambda}_1 = 1$. If the second case is true, then we set $\underline{\lambda}_1 = 0$ and $\overline{\lambda}_1 = \frac{1}{2}$.

Let $\mathcal{A} \subset \mathbb{N}$ be an arbitrary recursively enumerable set that is not recursive. With the definition of recursively enumerable sets, cf. Definition 4, we can construct a total function g, i.e., domain$(g) = \mathbb{N}$, such that the range of g is range$(g) = \mathcal{A}^c$ and g is recursive and therewith a computable function. Furthermore, without loss of generality, we can require that $g : \mathbb{N} \to \mathcal{A}$ is a one-to-one mapping from $\mathbb{N}$ to $\mathcal{A}$.

Next, we use a similar construction as in [17] and [27] which relies on a construction of Pour-El, cf. Case I on page 336 in [28]. For every $(n,m) \in \mathbb{N} \times \mathbb{N}$ we define the computable function $q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ as

$$q(n,m) = \begin{cases} 2^{m+2} & n \notin \{g(0), \dots, g(2^{m+2})\} \\ r & n \in \{g(0), \dots, g(2^{m+2})\} \text{ and } g(r) = n. \end{cases} \quad (6)$$

Note that r above is unique. Since $\mathcal{A}$ is recursively enumerable, the function q is indeed recursive and therewith computable.

For $n \in \mathbb{N}$ we define the sequence $(\hat{\lambda}^{(n)})_{n \in \mathbb{N}}$ with

$$\underline{\lambda}_n^* = \begin{cases} \lambda_* & \text{if } n \in \mathcal{A}^c \\ \underline{\lambda}_r & \text{if } n \in \mathcal{A} \text{ and } g(r) = n. \end{cases}$$

Note that for $n \in \mathcal{A}$ there can only be a single r with $g(r) = n$.

One can now show that the double sequence $(\underline{\lambda}_{q(n,m)})_{(n,m) \in \mathbb{N} \times \mathbb{N}}$ effectively converges for $m \to \infty$ to the sequence $(\underline{\lambda}_n^*)_{n \in \mathbb{N}}$. Then the sequence $(\underline{\lambda}_n^*)_{n \in \mathbb{N}}$ is a computable sequence and therewith $(F_{\text{trust}}(\lambda_n^*))_{n \in \mathbb{N}}$ must be a computable sequences as well, since F_{trust} must be Turing computable by assumption. Now, for $n \in \mathbb{N}$ we have

$$F_{\text{trust}}(\lambda_n^*) = \begin{cases} F_{\text{trust}}(\boldsymbol{\tau}_2) & \text{if } n \in \mathcal{A}^c \\ F_{\text{trust}}(\boldsymbol{\tau}_1) & \text{if } n \in \mathcal{A}. \end{cases}$$

We can use the computable sequence $(F_{\text{trust}}(\lambda_n^*))_{n \in \mathbb{N}}$ to find an algorithm that decides for $n \in \mathbb{N}$ whether $n \in \mathcal{A}$ or $n \in \mathcal{A}^c$ is satisfied. This would imply that the set $\mathcal{A}$ is recursively enumerable, which cannot be the case. This shows that F_{trust} cannot be Turing computable completing the proof. ■