On the Computability of the Secret Key Capacity Under Rate Constraints

Holger Boche

Technische Universität München
Lehrstuhl für Theoretische Informationstechnik

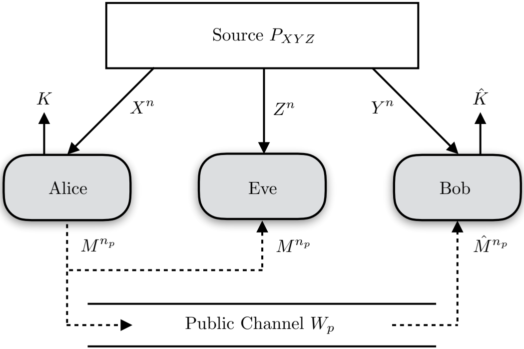
joint work with
Rafael Schaefer *(TU Berlin)*
H. Vincent Poor *(Princeton University)*

ICASSP 2019

IFS-L1: Information Forensics and Security I

May 16, 2019

Secret Key Generation



- Based on X^n and Y^n, Alice and Bob want to generate a secret key K
- Single forward transmission of helper data M^{n_p} over the public channel
  - Noisy W_p imposes a rate constraint $R_\mathsf{p} = C(W_\mathsf{p})$
  - Noiseless W_p results in no rate constraint
- Eve intercepts M^{n_p} error-free (worst case from a security perspective)

Forward SK Capacity

Theorem: [Csiszár/Narayan '00], [Bassi *et al.* '16]

The *forward SK capacity* $C_{\mathsf{SK}}(W_{\mathsf{p}}, P_{XYZ})$ for source P_{XYZ} and noisy public channel W_{p} is

$$\boldsymbol{C_{\mathsf{SK}}(W_{\mathsf{p}}, P_{XYZ}) = \max_{U,V} \left[I(V;Y|U) - I(V;Z|U) \right]}$$

where U and V are auxiliary random variables that satisfy the Markov chain relation $U - V - X - (Y, Z)$ and further satisfy the rate constraint

$$I(V;X|Y) \leq C(W_{\mathsf{p}}).$$

Moreover, it may be assumed that $V = (U, V')$ where the cardinalities of the alphabets of both U and V' are at most $|\mathcal{X}| + 1$.

I. Csiszár and P. Narayan, "Common randomness and secret key generation with a helper," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 344–366, Mar. 2000

G. Bassi, P. Piantanida, and S. Shamai (Shitz), "Secret key generation over noisy channels with common randomness," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 510–514

Forward SK Capacity (2)

- W_{p} becomes noiseless for $R_{\mathsf{p}} = C(W_{\mathsf{p}}) \to \infty$ and R_{p} is inactive

Corollary: [Ahlswede/Csiszár '93]

The *forward SK capacity* $C_{\mathsf{SK}}(P_{XYZ})$ for source P_{XYZ} is

$$\boldsymbol{C_{\mathsf{SK}}(P_{XYZ}) = \max_{U,V} \left[I(V;Y|U) - I(V;Z|U) \right]}$$

where U and V are auxiliary random variables that satisfy the Markov chain relation $U - V - X - (Y,Z)$. Moreover, it may be assumed that $V = (U, V')$ where the cardinalities of the alphabets of both U and V' are at most $|\mathcal{X}| + 1$.

R. Ahlswede and I. Csiszár, "Common Randomness in Information Theory and Cryptography-Part I: Secret Sharing," *IEEE Trans. Inf. Theory*, vol. 39, no. 4, pp. 1121–1132, Jul. 1993

Gold Standard / Holy Grail

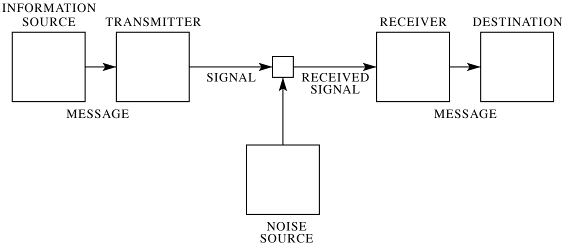
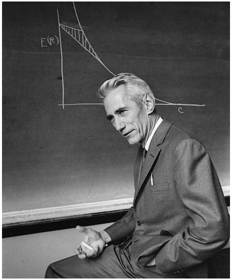


Fig. 1 — Schematic diagram of a general communication system.



The *capacity* $C(W)$ of a discrete memoryless channel (DMC) W is

$$\boldsymbol{C(W) = \max_{X} I(X;Y) = \max_{P_X} I(P_X, W)}$$

C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948

Capacity of DMCs

The *capacity* $C(W)$ of a discrete memoryless channel (DMC) W is

$$C(W) = \max_{X} I(X;Y) = \max_{P_X} I(P_X, W)$$

- *Entropic quantities*
- *Single-letter*
- *Convex optimization problem*

➠ Of particular relevance as it allows to compute the capacity $C(W)$ as a function of the channel W given by a convex optimization problem

What do we actually mean with "compute"?

Capacity of DMCs

The *capacity* $C(W)$ of a discrete memoryless channel (DMC) W is

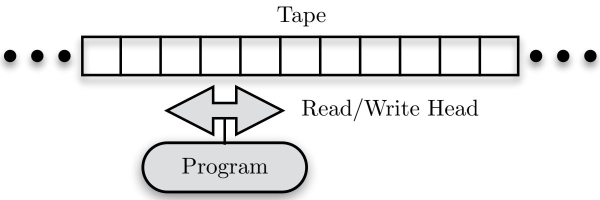
$$C(W) = \max_{X} I(X;Y) = \max_{P_X} I(P_X, W)$$

- *Entropic quantities*
- *Single-letter*
- *Convex optimization problem*

➠ Of particular relevance as it allows to **compute** the capacity $C(W)$ as a function of the channel W given by a convex optimization problem

What do we actually mean with "compute"?

Turing Machine



Mathematical model of an abstract machine that manipulates symbols on a strip of tape according to certain given rules

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

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

Turing Machine (2)

- Turing machines can simulate any given algorithm and therewith provide a simple but very powerful model of computation
- **No** limitations on computational complexity, unlimited computing capacity and storage, and execute programs completely error-free
- Extends to programming languages which are then called *Turing-complete*

➠ **Fundamental performance limits for today's digital computers**

➠ *Ideal concept to decide if the capacity can computed* **algorithmically** *(without putting any constraints on the computational complexity)*

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

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

Turing Machine (2)

- Turing machines can simulate any given algorithm and therewith provide a simple but very powerful model of computation
- **No** limitations on computational complexity, unlimited computing capacity and storage, and execute programs completely error-free
- Extends to programming languages which are then called *Turing-complete*

➠ **Fundamental performance limits for today's digital computers**

➠ *Ideal concept to decide if the capacity can computed* ***algorithmically*** *(without putting any constraints on the computational complexity)*

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

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

Turing Machine (2)

- Turing machines can simulate any given algorithm and therewith provide a simple but very powerful model of computation
- **No** limitations on computational complexity, unlimited computing capacity and storage, and execute programs completely error-free
- Extends to programming languages which are then called *Turing-complete*

➠ **Fundamental performance limits for today's digital computers**

➠ *Ideal concept to decide if the capacity can computed* **algorithmically** *(without putting any constraints on the computational complexity)*

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

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

Computability

- Computable numbers are real numbers that are computable by Turing machines

- A sequence $\{r_n\}_{n\in\mathbb{N}}$ is called 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

$$r_n = (-1)^{s(n)} \frac{a(n)}{b(n)}$$

- Then a real number x is said to be *computable* if there exists a computable sequence of rational numbers $\{r_n\}_{n\in\mathbb{N}}$ such that

$$|x - r_n| < 2^{-n}$$

➡ $\mathbb{R}_c$ is the set of computable real numbers

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

Computability (2)

➠ Based on this, we can define *computable probability distributions* and *computable channels*

- We define the set of computable probability distributions $\mathcal{P}_c(\mathcal{X})$ as the set of all probability distributions

$$P \in \mathcal{P}(\mathcal{X}) \text{ such that } P(x) \in \mathbb{R}_c,\ x \in \mathcal{X}$$

- Let $\mathcal{CH}_c$ be the set of all computable channels, i.e., for a channel

$$W : \mathcal{X} \to \mathcal{P}(\mathcal{Y}) \text{ we have } W(\cdot|x) \in \mathcal{P}_c(\mathcal{Y}) \text{ for every } x \in \mathcal{X}$$

Computability (3)

Definition: Borel Computability

A function $f : \mathbb{R}_c \to \mathbb{R}_c$ is called *Borel computable* if there is an algorithm that transforms each given computable sequence of a computable real x into a corresponding representation for $f(x)$.

- Turing's notion of computability conforms to Borel computability

Capacity $C(W) = \max_X I(X;Y)$ is Borel computable

Proof outline:

1. $x \log_2 x$, $x \in [0,1]$, is a Borel computable function
2. function $\sum x \log_2 x$ is computable
3. function $I(X;Y)$ is computable
4. $C(W) = \max_X I(X;Y)$ is a computable function since it is the maximum of computable functions ✓

Computability (3)

Definition: Borel Computability

A function $f : \mathbb{R}_c \to \mathbb{R}_c$ is called *Borel computable* if there is an algorithm that transforms each given computable sequence of a computable real x into a corresponding representation for $f(x)$.

- Turing's notion of computability conforms to Borel computability

Capacity $C(W) = \max_X I(X;Y)$ is Borel computable

Proof outline:

1. $x \log_2 x$, $x \in [0,1]$, is a Borel computable function
2. function $\sum x \log_2 x$ is computable
3. function $I(X;Y)$ is computable
4. $C(W) = \max_X I(X;Y)$ is a computable function since it is the maximum of computable functions ✓

Computability (4)

- There are weaker forms of computability including *Markov computability* and *Banach-Mazur computability*

Definition: Markov Computability

A function $f : \mathbb{R}_c \to \mathbb{R}_c$ is called *Markov computable* if there is an algorithm that converts an algorithm for a computable real x into an algorithm for $f(x)$.

Definition: Banach-Mazur Computability

A function $f : \mathbb{R}_c \to \mathbb{R}_c$ is called *Banach-Mazur computable* if f maps any given computable sequence $\{x_n\}_{n=1}^{\infty}$ of real numbers into a computable sequence $\{f(x_n)\}_{n=1}^{\infty}$ of real numbers.

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

Computability (4)

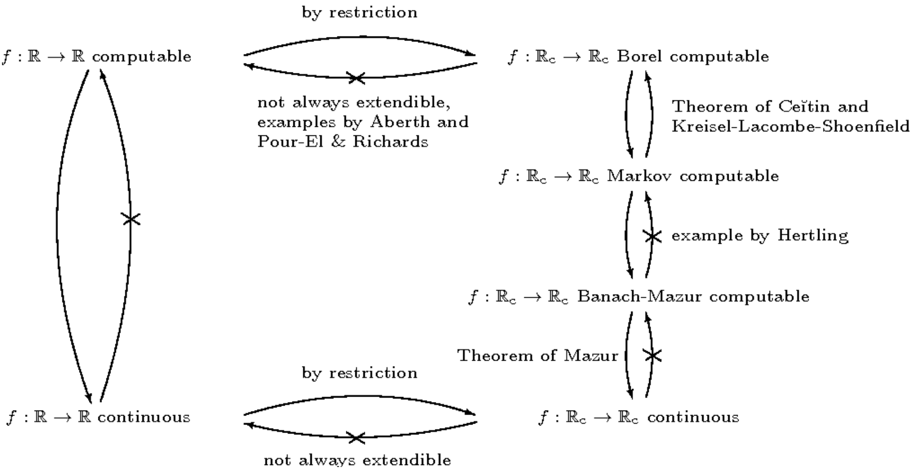


Figure 5 from

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

Back to the Forward SK Capacity

The *forward SK capacity* $C_{\mathsf{SK}}(W_{\mathsf{p}}, P_{XYZ})$ for source P_{XYZ} and noisy public channel W_{p} is

$$C_{\mathsf{SK}}(W_{\mathsf{p}}, P_{XYZ}) = \max_{U,V} \left[I(V;Y|U) - I(V;Z|U) \right]$$

with $U - V - X - (Y, Z)$ forming a Markov chain and further

$$I(V;X|Y) \leq C(W_{\mathsf{p}}).$$

The *forward SK capacity* $C_{\mathsf{SK}}(P_{XYZ})$ for source P_{XYZ} is

$$C_{\mathsf{SK}}(P_{XYZ}) = \max_{U,V} \left[I(V;Y|U) - I(V;Z|U) \right]$$

with $U - V - X - (Y, Z)$ forming a Markov chain.

QUESTION: Can we compute these capacity expressions?

Back to the Forward SK Capacity

The *forward SK capacity* $C_{\mathsf{SK}}(W_{\mathsf{p}}, P_{XYZ})$ for source P_{XYZ} and noisy public channel W_{p} is

$$\boldsymbol{C_{\mathsf{SK}}(W_{\mathsf{p}}, P_{XYZ}) = \max_{U,V} \left[I(V;Y|U) - I(V;Z|U) \right]}$$

with $U - V - X - (Y,Z)$ forming a Markov chain and further

$$I(V;X|Y) \leq C(W_{\mathsf{p}}).$$

The *forward SK capacity* $C_{\mathsf{SK}}(P_{XYZ})$ for source P_{XYZ} is

$$\boldsymbol{C_{\mathsf{SK}}(P_{XYZ}) = \max_{U,V} \left[I(V;Y|U) - I(V;Z|U) \right]}$$

with $U - V - X - (Y,Z)$ forming a Markov chain.

QUESTION: Can we compute these capacity expressions?

Without Rate Constraint

Theorem:

For all $|\mathcal{X}| \geq 2$, $|\mathcal{Y}| \geq 2$, and $|\mathcal{Z}| \geq 2$, the forward SK capacity

$$C_{\mathsf{SK}}(P_{XYZ}) = \max_{U,V} \left[I(V;Y|U) - I(V;Z|U) \right]$$

without public rate constraint for source P_{XYZ} is Borel computable.

Proof outline:

1. $x \log_2 x$, $x \in [0,1]$, is a Borel computable function
2. function $\sum x \log_2 x$ is computable
3. functions $I(V;Y|U)$ and $I(V;Z|U)$ are computable
4. $C_{\mathsf{SK}}(P_{XYZ}) = \max_{U,V}[I(V;Y|U) - I(V;Z|U)]$ is a computable function since it is the maximum of computable functions ✓

- This can even be strengthened: Forward SK capacity $C_{\mathsf{SK}}(P_{XYZ})$ is a *computable continuous function*!

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," 2018, submitted

With Rate Constraint

Theorem:

For all $|\mathcal{X}| \geq 2$, $|\mathcal{Y}| \geq 2$, and $|\mathcal{Z}| \geq 2$, the forward SK capacity $C_{\mathsf{SK}}(W_{\mathsf{p}}, P_{XYZ})$ for the source P_{XYZ} and the noisy public channel W_{p} is **not** Banach-Mazur computable.

Key ingredient:

- Proof by contradiction
- If $C_{\mathsf{SK}}(W_{\mathsf{p}}, P_{XYZ})$ would be Banach-Mazur computable, then the halting problem would be solvable!

➠ The forward SK capacity with rate-limited public public communication is not Banach-Mazur and therewith also **not** Turing computable!

ANSWER: The forward SK capacity is

- computable without public rate constraints and
- **non**-computable with public rate constraints!

With Rate Constraint

Theorem:

For all $|\mathcal{X}| \geq 2$, $|\mathcal{Y}| \geq 2$, and $|\mathcal{Z}| \geq 2$, the forward SK capacity $C_{\mathsf{SK}}(W_{\mathsf{p}}, P_{XYZ})$ for the source P_{XYZ} and the noisy public channel W_{p} is **not** Banach-Mazur computable.

Key ingredient:

- Proof by contradiction
- If $C_{\mathsf{SK}}(W_{\mathsf{p}}, P_{XYZ})$ would be Banach-Mazur computable, then the halting problem would be solvable!

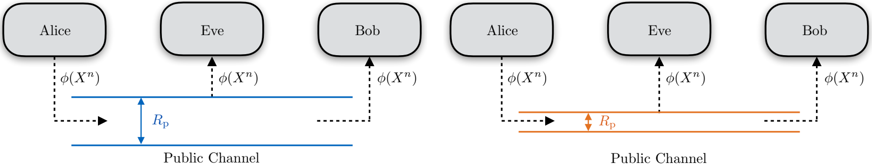
➠ The forward SK capacity with rate-limited public public communication is not Banach-Mazur and therewith also **not** Turing computable!

ANSWER: The forward SK capacity is

- computable without public rate constraints and
- **non**-computable with public rate constraints!

Conclusions

- Computability of the forward SK capacity has been studied
- Sharp phase transition between being computable and non-computable



- Rate constraint R_{p} inactive
- $C_{\mathsf{SK}}(P_{XYZ})$ computable (even computable continuous function)

- Rate constraint R_{p} active
- $C_{\mathsf{SK}}(R_{\mathsf{p}}, P_{XYZ})$ non-computable (not even Banach-Mazur)

Rate constraint on the public communication not only affects the performance, it is also turns an algorithmically non-tractable problem into a solvable problem!

Conclusions (2)

Thank you for your attention!

- Many extensions and other open problems including secure communication with active jammers [BSP '18], identification [BSP '18], secure authentication [BSBP '19], detection of denial-of-service attacks [BSP '19] and many others

H. Boche, R. F. Schaefer, and H. V. Poor, "Performance evaluation of secure communication systems on Turing machines," in *Proc. 10th IEEE Int. Workshop Inf. Forensics Security*, Hong Kong, Dec. 2018, pp. 1–7

H. Boche, R. F. Schaefer, and H. V. Poor, "Secure communication and identification systems – effective performance evaluation on Turing machines," 2018, submitted

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," 2018, submitted

H. Boche, R. F. Schaefer, and H. V. Poor, "Detectability of denial-of-service attacks on communication systems," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, Brighton, UK, May 2019

If time permits, another motivation for studying the computability....

Motivation

Status Quo[1]

- several 100.000 new software products every year
- all main products of top leading software companies are insecure, erroneous, and impossible to verify
- uncountable software updates to repair identified errors
- no secure infrastructure
- uncountable adversarial attacks on hardware and software (e.g. Vodafone monitors every 2ms an attack on their system)

[1] A. Schönbohm, President of Federal Office for Information Security (BSI), at *Technology Innovation 2018*, Feb 22, 2018

Verification

Industry with hardware and software products

⇒

Certification agency (e.g. BSI)

YES ⇒

Verified hard- and software infrastructure

⇓ **NO**

- **QUESTION:** How can we design this?

Verification

Industry with hardware and software products ⇒ Certification agency (e.g. BSI) **YES** ⇒ Verified hard- and software infrastructure

⇓ **NO**

- **QUESTION:** How can we design this?

Verification (2)

Implementation of solution of communication task:

Protocol
⊕ Physical channels
⊕ Attack classes

⇒

(universal) Turing machine to verify:

- security
- privacy
- efficiency
- ...

YES ⇒

Verified hard- and software infrastructure

⇓ **NO**

➠ Verification framework based on *Turing machines* and *computability* as developed in the beginning!

Verification (2)

Implementation of solution of communication task:

Protocol
⊕ Physical channels
⊕ Attack classes

⇒

(universal) Turing machine to verify:

- security
- privacy
- efficiency
- ...

YES ⇒

Verified hard- and software infrastructure

⇓ **NO**

➠ Verification framework based on *Turing machines* and *computability* as developed in the beginning!

Turing Machine (2)

Definition: Turing Machine

A Turing machine $\mathfrak{T}$ given by

$$\mathfrak{T} : \mathcal{CS} \times \mathcal{CH} \times \mathcal{CP} \times \mathbb{N} \to \{\text{yes / no}\}$$

is a mapping with

$$\mathfrak{T}(\mathrm{CS}, \mathrm{CH}, \mathrm{CP}, k) = \text{yes}$$

if and only if the performance requirements are satisfied and

$$C - R < \frac{1}{k} \tag{1}$$

where C denotes the capacity.

For a given communication scenario CS, the verification of security and spectral efficiency is called *effective*, if there exists a (universal) Turing machine such that for all valid channel inputs, communication protocols CP, and all $k \in \mathbb{N}$ the Turing machine always outputs the correct answer

Turing Machine (3)

- Within this framework, the task of the Turing machine is to answer the following two questions:

Question 1: Are all information theoretic performance requirements (such as probability of decoding error at the legitimate receiver or secrecy at an eavesdropper) satisfied?

Question 2: Is effectiveness given in the sense that the gap $1/k$ to the optimal performance limit can be controlled?

- In particular Question 2 requires that the **capacity C must be algorithmically computable**

➠ This defines a necessary condition

Turing Machine (3)

- Within this framework, the task of the Turing machine is to answer the following two questions:

Question 1: Are all information theoretic performance requirements (such as probability of decoding error at the legitimate receiver or secrecy at an eavesdropper) satisfied?

Question 2: Is effectiveness given in the sense that the gap $1/k$ to the optimal performance limit can be controlled?

- In particular Question 2 requires that the **capacity C must be algorithmically computable**

➠ This defines a necessary condition

References I

I. Csiszár and P. Narayan, “Common randomness and secret key generation with a helper,” *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 344–366, Mar. 2000.

G. Bassi, P. Piantanida, and S. Shamai (Shitz), “Secret key generation over noisy channels with common randomness,” in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 510–514.

R. Ahlswede and I. Csiszár, “Common Randomness in Information Theory and Cryptography-Part I: Secret Sharing,” *IEEE Trans. Inf. Theory*, vol. 39, no. 4, pp. 1121–1132, Jul. 1993.

C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

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

A. M. Turing, “On computable numbers, with an application to the Entscheidungsproblem. A correction,” *Proc. London Math. Soc.*, vol. 2, no. 43, pp. 544–546, 1937.

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

References II

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.

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," 2018, submitted.

H. Boche, R. F. Schaefer, and H. V. Poor, "Performance evaluation of secure communication systems on Turing machines," in *Proc. 10th IEEE Int. Workshop Inf. Forensics Security*, Hong Kong, Dec. 2018, pp. 1–7.

H. Boche, R. F. Schaefer, and H. V. Poor, "Secure communication and identification systems – effective performance evaluation on Turing machines," 2018, submitted.

H. Boche, R. F. Schaefer, and H. V. Poor, "Detectability of denial-of-service attacks on communication systems," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, Brighton, UK, May 2019.