**Mathematica Pannonica *New Series* xx /NS xx-26/ (yyyy) i, bp-ep**
DOI: 10.1556/2062.yyyy.—

AKADÉMIAI KIADÓ

# GENERATING BOOLEAN LATTICES BY FEW ELEMENTS AND A RELATED CRYPTOGRAPHIC PROTOCOL FOR AUTHENTICATION

Gábor Czédli[1]

[1] University of Szeged, Hungary, www.math.u-szeged.hu/~czedli/

Communicated by Handling Editor

**Original Research Paper**

**Received: Month Day, Year • Accepted: Month Day, Year**



## ABSTRACT

Let $\mathrm{Sp}(k)$ denote the number of the $\lfloor k/2\rfloor$-element subsets of a finite $k$-element set. We prove that the least size of a generating subset of the Boolean lattice with $n$ atoms (or, equivalently, the powerset lattice of an $n$-element set) is the least number $k$ such that $n \le \mathrm{Sp}(k)$. Based on this fact and our 2021 protocol, which was based on equivalence lattices, we present a secret key cryptographic protocol for authentication. We prove that the underlying mathematical problem of this protocol is hard in the sense that if it belongs to the complexity class **P** then **P** equals **NP**.

## KEYWORDS

**Boolean lattice, generating set, cryptography, secret key, smallest generating set, NP-complete, average NP-complete, authentication, Vernam cipher.**

## MATHEMATICS SUBJECT CLASSIFICATION (2020)

Primary 06D99; Secondary 94A62, 94A60, 68Q25

July 25, 2023

## 1. INTRODUCTION

### 1.1. Targeted readership

This paper targets a large readership. Indeed, those familiar with the concept of a Boolean lattice and that of **NP**-completeness should have no difficulty in reading the *results* [1] and even most other parts of the paper. Most of the exceptions, which need a little familiarity with lattices or universal algebra, occur in Subsection 1.3. This short subsection surveys how a series of lattice theoretic investigations lead to the present paper, which could be interesting outside lattice theory and even outside mathematics.

### 1.2. Our goal

As usual, $\mathbb{N}^+ = \{1, 2, \dots\}$ stands for the set of positive integers. For $n \in \mathbb{N}^+$, let $\mathsf{B}_n = (\mathsf{B}_n; \vee, \wedge)$ be the *Boolean lattice* with $n$ atoms. Note that $\mathsf{B}_n$ is isomorphic to (and so it can be defined as)

1 Technical Editor: My system has several problems with fonts. I could not use italic under the documentclass MathPannX; *dark magenta* texts should be italic or slanted. Similarly, **A** and **M** should be in \mathcal font while **P** and **NP** in \textbf.

the powerset lattice $(P(\{1,\dots,n\});\cup,\cap)$, whence $|\mathsf{B}_n| = 2^n$. A subset $X$ of $\mathsf{B}_n$ is a generating set of $\mathsf{B}_n$ if no proper subset of $\mathsf{B}_n$ is closed with respect to join ($\vee$) and meet ($\wedge$). In Theorem 2.1, we are going to determine the smallest $k \in \mathbb{N}^+$ such that $\mathsf{B}_n$ has a $k$-element generating set. Section 3, which is a computer-assisted, indicates that if $k' > k$ but $k'$ is still small, then $\mathsf{B}_n$ has many $k'$-element generating sets. Based on the plenty of these generating sets, Section 4 outlines a secret key cryptographic protocol for authentication. Section 5 shows that the underlying problem of this protocol is hard in the sense that if it belongs to the complexity class **P** then **P** equals **NP**. Finally, Section 6 warns the reader that this connection with **NP** does not guarantee security in itself and, on the positive side, Section 6 shows some perspectives.

### 1.3. A historical mini-survey

From the author's perspective, the story started with Zádori [26], who gave a new proof of a result of Strietz [22]–[23] asserting that the equivalence lattice $\mathrm{Equ}(A)$ (consisting of all equivalences of $A$) has a 4-element generating set provided that $A$ is a finite set and $|A| \geq 3$. For short, we say that $\mathrm{Equ}(A)$ is 4-generated for these finite sets $A$. In the next step, based on Zádori's method, Chajda and Czédli [3] proved that the lattices $\mathrm{Quo}(A)$ of all quasiorders (AKA preorders) of these finite sets $A$ and even some infinite sets $A$ are 6-generated; in fact, they are 3-generated if we add the unary operation $\rho \mapsto \rho^{-1} = \{(y,x) : (x,y) \in \rho\}$ of forming inverses to the set $\{\bigvee, \bigwedge\}$ of infinitary lattice operations. Next, Czédli [5] extended Zádori's result to $\mathrm{Equ}(A)$ with $|A| = \aleph_0$. Furthermore, Czédli [4, 6] and Takách [24] proved that $\mathrm{Equ}(A)$ and $\mathrm{Quo}(A)$ are 4-generated and 6-generated, respectively, provided that $A$ is an infinite set and there is no inaccessible cardinal $\lambda$ such that $\lambda \leq |A|$. Moreover, the 1999 paper Czédli [6] proved that $\mathrm{Equ}(A)$ has a 4-element non-antichain generating set for these sets $A$. Note that Kuratowski [17] gave a model of ZFC in which there is no inaccessible cardinal at all.

Around 1999, Vilmos Totik proved that our methods are insufficient to deal with inaccessible cardinals. Hence, the topic was put aside after the 1999 paper Czédli [6], and it is still an open problem whether $\mathrm{Equ}(A)$ and $\mathrm{Quo}(A)$ are finitely generated (as complete lattices) if there exists an inaccessible cardinal $\leq |A|$.

The research started again in 2015, when Dolgos [13], one of Miklós Maróti's students, proved that $\mathrm{Quo}(A)$ is 5-generated for $|A| \leq \aleph_0$, and Kulin [16] extended this result to all sets $|A|$ such that there is no inaccessible cardinal $\lambda \leq |A|$. Not much later, Czédli [7] and Czédli and Kulin [10] reduced the number of generators by proving that for all sets $A$ such that $|A| \neq 4$ and there is no inaccessible cardinal $\lambda \leq |A|$, the complete lattice $\mathrm{Quo}(A)$ is 4-generated. The case $|A| = 4$ is still open but the result was optimal for many other sets, as [7] proved that $\mathrm{Quo}(A)$ is not 3-generated if $|A| \geq 3$. Finding 4-element generating sets that are not antichains is more difficult but, after Strietz [22]–[22] and Zádori [26], some sporadic cases have recently been settled in Ahmed and Czédli [1] and Czédli and Oluoch [11].

In 2020, it appeared that the technique developed for infinite sets is appropriate to show that even some direct powers and products of some finite equivalence lattices are 4-generated and (consequently) $\mathrm{Equ}(A)$ and $\mathrm{Quo}(A)$ have very many 4-element generating sets if $|A|$ is a large finite number; see Czédli [8] and Czédli and Oluoch [11]. Based on the abundance of the generating sets found in the just mentioned two papers, Czédli [8] in 2021 suggested a protocol (the 2021 protocol for short) for authentication and cryptography based on lattices. Quite recently, while looking for small generating sets of some filters of quasiorder lattices, a proof in Czédli [9] required to know the smallest size of a generating set of a finite Boolean lattice; this was the immediate motivation for the present paper.

## 2. SMALL GENERATING SETS OF FINITE BOOLEAN LATTICES

For $n \in \mathbb{N}^+$, we introduce the notation

$$\mathrm{Sp}(n) := \binom{n}{\lfloor n/2 \rfloor} \tag{2.1}$$

where $\lfloor n/2 \rfloor$ is the (lower) integer part of $n/2$. For example,

$$\mathrm{Sp}(32) = 601\,080\,390 \quad \text{and} \quad \mathrm{Sp}(33) = 1\,166\,803\,110. \tag{2.2}$$

The notation Sp comes from "Sperner"; see later. For $n \in \mathbb{N}^+$, let $\mathrm{LASp}(n)$ be the smallest $k \in \mathbb{N}^+$ such that $n \le \mathrm{Sp}(k)$. Note the rule: $n \le \mathrm{Sp}(k) \iff \mathrm{LASp}(n) \le k$; this explains the acronym, which comes from "Left Adjoint of Sp".

**THEOREM 2.1.** For $n, k \in \mathbb{N}^+$, $\mathsf{B}_n$ has an at most $k$-element generating set if and only if $n \le \mathrm{Sp}(k)$ or, equivalently, if and only if $\mathrm{LASp}(n) \le k$. In particular, $\mathrm{LASp}(n)$ is the smallest possible size of a generating set of $\mathsf{B}_n$.

For example, this theorem together with (2.2) give that $\mathsf{B}_{1\,000\,000\,000}$ is 33-generated but not 32-generated.

**Proof.** Let $\mathrm{At}(\mathsf{B}_n)$ be the set of atoms of $\mathsf{B}_n$. As usual, for an element $u$ of a lattice $L$, $\downarrow u$ and $\uparrow u$ will stand for $\{x \in L : x \le u\}$ and $\{x \in L : x \ge u\}$, respectively. First, we show that for any subset $Y$ of $\mathsf{B}_n$

$$\text{if } Y \text{ generates } \mathsf{B}_n \text{ and } a \in \mathrm{At}(\mathsf{B}_n), \text{ then } a = \bigwedge (Y \cap \uparrow a). \tag{2.3}$$

As $Y$, say $Y = \{b_1, \dots, b_m\}$, generates $\mathsf{B}_n$ and $\mathsf{B}_n$ is distributive, $a = t(b_1, \dots, b_m)$ for an $m$-ary disjunctive normal form, that is, $a$ is the join of meets of elements of $Y$. But $a$ is join-irreducible, whereby it is the meet of some elements of $Y$. This shows the "$\ge$" part of (2.3). The "$\le$" is trivial, and we have proved (2.3).

Next, we claim that for any subset $G$ of $\mathsf{B}_n$,

$$\text{if } G \text{ generates } \mathsf{B}_n \text{ and } k = |G|, \text{ then } n \le \mathrm{Sp}(k). \tag{2.4}$$

To show this, assume that $G$ is a $k$-element generating set of $\mathsf{B}_n$. Let $X$ be a $k$-element set and denote by $\mathrm{FS}_\wedge(X)$ the meet-semilattice freely generated by $X$. Denote by $M$ the meet-subsemilattice of $(\mathsf{B}_n; \wedge)$ generated by $G$. Pick a bijective map $f_0 : X \to G$. The freeness of $\mathrm{FS}_\wedge(X)$ allows us to extend $f_0$ to a meet-homomorphism $f : \mathrm{FS}_\wedge(X) \to M$, which is surjective since $f(X) = G$ generates $M$. By (2.3), $\mathrm{At}(\mathsf{B}_n) \subseteq M$. This together with the surjectivity of $f$ allow us to take an injective map $g : \mathrm{At}(\mathsf{B}_n) \to \mathrm{FS}_\wedge(X)$ such that, for all $a \in \mathrm{At}(\mathsf{B}_n)$, $f(g(a)) = a$. If we had that $g(a) \le g(a')$ for distinct $a, a' \in \mathrm{At}(\mathsf{B}_n)$, then $g(a) = g(a) \wedge g(a')$ would lead to $a = f(g(a)) = f(g(a) \wedge g(a')) = f(g(a)) \wedge f(g(a')) = a \wedge a'$, yielding that $a \le a'$ and contradicting that $a$ and $a'$ are distinct atoms of $\mathsf{B}_n$. Therefore $g(a) \parallel g(a')$, that is, $g(\mathrm{At}(\mathsf{B}_n))$ is an $n$-element antichain in $\mathrm{FS}_\wedge(X)$. Adding a top element to $\mathrm{FS}_\wedge(X)$, we obtain another semilattice, $\{1\} \cup \mathrm{FS}_\wedge(X)$. We know from the folklore or from McKenzie, McNulty, and Taylor [19, Page 240, §4] that $\{1\} \cup \mathrm{FS}_\wedge(X)$ is order isomorphic to $\mathsf{B}_{|X|} = \mathsf{B}_k$. So $\mathsf{B}_k$ has an $n$-element antichain. By Sperner's theorem [21], see also Grätzer [15, page 354], any antichain in $\mathsf{B}_k$ has at most $\mathrm{Sp}(k)$ elements. This implies (2.4) and the "only if" part of the theorem.

Next, observe that

$$\left.\begin{array}{l}\text{for any } m \le n \in \mathbb{N}^+, \mathsf{B}_m \text{ is a homomorphic image of } \mathsf{B}_n. \text{ Therefore, if } \mathsf{B}_n \text{ has}\\ \text{an at most } k\text{-element generating set, then so does } \mathsf{B}_m.\end{array}\right\} \tag{2.5}$$

It suffices to show the first part for $m = n - 1$. Let $c$ be a coatom (that is, a lower cover of 1) in $\mathsf{B}_n$. Then $\downarrow c \cong \mathsf{B}_m$. The function $f : \mathsf{B}_n \to \downarrow c$ defined by $x \mapsto c \wedge x$ is a homomorphism by distributivity. As $x = f(x)$ for each $x \in \downarrow c$, we conclude (2.5).

Next, to show the "if" part of the theorem, assume that $n \le \mathrm{Sp}(k)$; we are going to show that $\mathsf{B}_n$ has an at most $k$-element generating set. Based on (2.5), we can assume that $n = \mathrm{Sp}(k)$. As $\mathsf{B}_k$ is isomorphic to the powerset lattice $(P(\{1, \dots, k\}); \cup, \cap)$ and the $\lfloor k/2 \rfloor$-element subsets of $\{1, \dots, k\}$ form an $n = \mathrm{Sp}(k)$-element antichain in $(P(\{1, \dots, k\}); \cup, \cap)$, it follows that $\mathsf{B}_k$ has an $n$-element antichain $H$. As $(P(H); \cup, \cap) \cong \mathsf{B}_n$, it suffices to find a $k$-element generating set of the powerset lattice $P(H) = (P(H); \cup, \cap)$. For each $a \in \mathrm{At}(\mathsf{B}_k)$, we let $X_a := H \cap \uparrow a$. Then $X_a \in P(H)$ and $G := \{X_a : a \in \mathrm{At}(\mathsf{B}_k)\}$ is an at most $k$-element subset of $P(H)$. To show that $G$ generates $P(H)$, it suffices to show that for every $h \in H$,

$$\{h\} = \bigcap \{X_a : a \in \mathrm{At}(\mathsf{B}_k) \cap \downarrow h\}. \tag{2.6}$$

For every $a \in \mathrm{At}(\mathsf{B}_k) \cap {\downarrow}h$, we have that $h \in H \cap {\uparrow}a = X_a$, showing the "$\subseteq$" part of (2.6). Now assume that $h' \in H$ belongs to the intersection in (2.6). Then $h' \in X_a$ for every $a \in \mathrm{At}(\mathsf{B}_k)$ such that $a \le h$. Writing this in a more useful way,

$$(\forall a \in \mathrm{At}(\mathsf{B}_k))\ \big(a \le h \Rightarrow a \le h'\big), \text{ that is, } \mathrm{At}(\mathsf{B}_k) \cap {\downarrow}h \subseteq \mathrm{At}(\mathsf{B}_k) \cap {\downarrow}h'.$$

Hence, using that each element of $\mathsf{B}_k$ is the join of all atoms below it, $h = \bigvee(\mathrm{At}(\mathsf{B}_k) \cap {\downarrow}h) \le \bigvee(\mathrm{At}(\mathsf{B}_k) \cap {\downarrow}h') = h'$. But $h, h' \in H$ and $H$ is an antichain, whereby $h \le h'$ gives that $h' = h \in \{h\}$, showing the "$\supseteq$" part of (2.6). Therefore, (2.6) and the "if" part of the theorem hold. □

**COROLLARY 2.2.** If $2 \le k \in \mathbb{N}^+$ and $n \le \mathrm{Sp}(k)$, then the free distributive lattice $\mathrm{FD}(k)$ has a sublattice isomorphic to $\mathsf{B}_n$.

**Proof.** As $\mathsf{B}_m$ is a sublattice of $\mathsf{B}_n$ for any $m \le n$, we can assume that $n = \mathrm{Sp}(k)$. Theorem 2.1 yields a surjective homomorphism $f : \mathrm{FD}(k) \to \mathsf{B}_n$. Let $h : \mathsf{B}_n \to \mathsf{B}_n$ be the identity map (defined by $x \mapsto x$ for $x \in \mathsf{B}_n$). Since $\mathsf{B}_n$ is projective in the class of all distributive lattices by Balbes [2, Theorem 7.1(i),(iii')], there is a homomorphism $g : \mathsf{B}_n \to \mathrm{FD}(k)$ such that $fg = h$. As the product $h$ is injective, so is $g$. Thus, $g(\mathsf{B}_n) \cong \mathsf{B}_n$ and $g(\mathsf{B}_n)$ is a required sublattice of $\mathrm{FD}(k)$. □

## 3. THE ABUNDANCE OF SMALL GENERATING SETS OF FINITE BOOLEAN LATTICES

We call a $k$-dimensional vector $\vec{h} = (h_1, \dots, h_k)$ a generating vector of $\mathsf{B}_n$ if the set $\{h_1, \dots, h_k\}$ of its components is a generating set of $\mathsf{B}_n$. Here $|\{h_1, \dots, h_k\}| \le k$ and no equality is required. If $k < n$, then $k$ is much smaller than $|\mathsf{B}_n| = 2^n$, whereby the components of a randomly chosen $k$-dimensional vector from $\mathsf{B}_n^k$ are pairwise distinct with high probability. Therefore, the ratio of the $k$-element generating sets to all $k$-element subsets of $\mathsf{B}_n$ is close to the ratio of the $k$-dimensional generating vectors to all $k$-dimensional vectors belonging to $\mathsf{B}_n^k$.

A computer program, written by the author and available from his website, counted the generating vectors of $\mathsf{B}_{1000}$ among one hundred thousand randomly selected $k$-dimensional vectors for some $k$. Some of the results are given below while some others in arXiv:2303.10790, the extended version of the present paper.

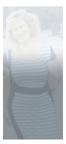
```
n=1000 k=40  Tested:100000 Generating:    42;      506.867 seconds.
n=1000 k=50  Tested:100000 Generating: 59003;     1305.780 seconds.
n=1000 k=80  Tested:100000 Generating: 99990;     2647.147 seconds.
n=1000 k=90  Tested:100000 Generating: 99999;     2974.364 seconds.
n=1000 k=100 Tested:100000 Generating:100000;     3265.869 seconds.
```

Thus, we conjecture that a random member of $\mathsf{B}_{1000}^{50}$ is a 50-dimensional generating vector of $\mathsf{B}_{1000}$ with probability at least $1/2$. Note that $\mathrm{LASp}(1000) = 13$.

## 4. A CRYPTOGRAPHIC PROTOCOL FOR AUTHENTICATION

In this section, we outline how to tailor the 2021 protocol, see Czédli [8], from equivalence lattices to Boolean lattices. We only present the main ideas here; the extended version of the paper contains further (mostly straightforward) details.

In our model, Kati[2] communicates with her Bank online. They agree upon a secret key, which only Kati and her Bank know. Let, say, $k = 50$, $n = 1000$, and $b = 100$. The secret key is a randomly selected $k$-dimensional generating vector $\vec{h} = (h_1, \dots, h_k)$ of $\mathsf{B}_n$. It follows from Section 3 that a computer program can find such an $\vec{h}$ in less than a second. In case of authentication, which means that Kati wants to prove her identity to the Bank, Kati requests a random vector $\vec{p} = (p_1, \dots, p_b)$ of $k$-ary lattice terms from the Bank. Then



the Bank generates such a random vector $\vec{p}$, sends $\vec{p}$ to Kati, Kati computes $\vec{u} := \vec{p}(\vec{h}) = (p_1(\vec{h}), \dots, p_b(\vec{h}))$ and sends it to the Bank, and the Bank checks whether $\vec{u}$ equals $\vec{p}(\vec{h})$. (4.1)

2 The Hungarian variant of "Cathy" and "Kate".

Changing their roles, the Bank can also prove its identity upon Kati's request. Note that $\vec{p}(\vec{h})$ can be used as a secret key in various cryptographic protocols including Vernam's cipher, see the extended version of the paper, but here we focus only on authentication. There is an Adversary who not only eavesdrops on the communication channel and intercepts messages but he can also modify messages and send his own messages pretending as if he was Kati or the Bank.

In (4.1), $\vec{u}$ can take $|\mathsf{B}_n|^b = 2^{nb}$ many values. If $nb$ is small, then a random $\vec{u}$ equals $\vec{p}(\vec{h})$ with probability $2^{-nb}$, which cannot be neglected. So if $nb$ is small, then the Adversary can experiment with a random $\vec{u}$ and he succeeds in breaking the protocol too often. In particular, we note for later reference that

$$\text{if } n = 1 \text{ and } b = 2\text{, then, on average, the Adversary can break the protocol in every fourth step.} \tag{4.2}$$

## 5. THE UNDERLYING PROBLEM IS HARD

For $n = 1000$ and $b = 100$, the Adversary has no chance to find $\vec{u}$ for (4.1) by random choices in his lifetime. Solving the underlying problem seems to be the only way for him. That is, from and intercepted pair $(\vec{p}, \vec{u} = \vec{p}(\vec{h}))$, he should find (at least one) $\vec{h}$. (Intercepting several such pairs corresponds to enlarging $b$ and does not help.)

As in Section 4, we will assume that $n, k, b \in \mathbb{N}^+$, $\vec{p}$ is a $b$-dimensional vector of $k$-ary lattice terms, and $\vec{u} \in \mathsf{B}_n^b$. Writing $\vec{x} = (x_1, \dots, x_k)$ instead of $\vec{h} \in \mathsf{B}_n^k$, the underlying problem of protocol (4.1) is this:

$$\mathrm{CPr}(n,b) : \quad \text{given an input } \vec{p}(\vec{x}) = \vec{u} \text{ with } \vec{u} \in \mathsf{B}_n^b\text{, find a solution of the equation } \vec{p}(\vec{x}) = \vec{u} \text{ for the unknown } \vec{x} \in \mathsf{B}_n^k \text{ in those cases where there exists a solution.} \tag{5.1}$$

With the same meaning of $n$, $k$, $b$, $\vec{p}$, and $\vec{u}$, we also define a related decision problem:

$$\mathrm{DPr}(n,b) : \quad \text{given an input } \vec{p}(\vec{x}) = \vec{u} \text{ with } \vec{u} \in \mathsf{B}_n^b\text{, decide whether the equation } \vec{p}(\vec{x}) = \vec{u} \text{ has a solution in } \mathsf{B}_n^k \text{ for the unknown } \vec{x}. \tag{5.2}$$

The acronyms CPr and DPr come from "Construction Problem" and "Decision Problem", respectively. Let $\mathrm{size}(\vec{p}(\vec{x}) = \vec{u})$ and $\mathrm{size}(\vec{h})$ denote the size of $\vec{p}(\vec{x}) = \vec{u}$ and that of $\vec{h}$, respectively; these sizes are the numbers of bits in (the usual) binary representations of $\vec{p}(\vec{x}) = \vec{u}$ and $\vec{h}$.

There are many books and papers dealing with the widely known concept of the complexity classes **P** and **NP**; some of them will be cited later but even Wikipedia is sufficient for us. However, **P**, **NP**, and **NP**-completeness are usually about decision problems while $\mathrm{CPr}(n,b)$ in (5.1) is not such. There is another difference: while we require an answer for each input string in case of a decision problem, this is not so in case of $\mathrm{CPr}(n,b)$. These circumstances constitute our excuse that we neither define what the **NP**-completeness of $\mathrm{CPr}(n,b)$ could mean nor we know whether $\mathrm{CPr}(n,b)$ would have such a property (as we would experience difficulty with a suitable replacement of $\mathbf{A}_1(d)$ later in the proof). However, we can safely agree to the following terminology:

$$\mathrm{CPr}(n,b)\text{, given in (5.1), belongs to } \mathbf{P} \overset{\text{def}}{\iff} \text{there are an algorithm } \mathbf{A}(n,b) \text{ and a polynomial } f^{(n,b)} \text{ such that for every input equation } \vec{p}(\vec{x}) = \vec{u} \text{ of } \mathrm{CPr}(n,b)\text{, if } \vec{p}(\vec{x}) = \vec{u} \text{ has a solution, then } \mathbf{A}(n,b) \text{ finds one of its solutions in (at most) } f^{(n,b)}(\mathrm{size}(\vec{p}(\vec{x}) = \vec{u})) \text{ steps.} \tag{5.3}$$

The algorithm and the polynomial depend on the parameters $n$ and $b$. We could have written "time" instead of "steps". Later, we will always omit "(at most)".

We have the following statement, in which $b$ denotes the dimension of $\vec{p}$.

**PROPOSITION 5.1.** For $2 \le b \in \mathbb{N}^+$ and $n \in \mathbb{N}^+$, if $\mathrm{CPr}(n,b)$, defined in (5.1), belongs to the complexity class **P** in the sense of (5.3), then **P** is equal to **NP**.

Even if the famous "is **P** equal to **NP**?" problem is, unexpectedly, solved affirmatively in the future, the proof below will still say something on the difficulty of $\mathrm{CPr}(n,b)$.

**Proof.** In the whole proof, we assume that $2 \le b \in \mathbb{N}^+$, $n \in \mathbb{N}^+$, and $\mathrm{CPr}(n, b)$ belongs to **P**.

In principle, we should have written "Turing machine" in (5.3) rather than "algorithm"[3]. Fortunately, the algorithms in the proof (which are clearly equivalent to usual computer programs) can be simulated by Turing machines and this simulation preserves the property "being in **P**"; see, for example, Theorem 17.4 in Rich [20]. By the same theorem, for $n'$ computer steps[4] (and for $n'$ steps in our mind), the simulating Turing machine needs $(O(n'))^6$ steps. Therefore, we will mostly speak of polynomials without specifying their degrees even when a sub-algorithm is clearly linear (or even better) in our mind, that is, for our computers. For example,

for each fixed $d \in \mathbb{N}^+$, there are a polynomial $f_1^{(d)}$ and an algorithm $\mathbf{A}_1(d)$ such that, for each $\xi \in \mathbb{N}^+$, $\mathbf{A}_1(d)$ computes and stores $\xi^d$ in $f_1^{(d)}(\xi)$ steps. (5.4)

Clearly, there are polynomials $f_2^{(n,b)}$ and $f_3$ and algorithms $\mathbf{A}_2(n, b)$ and $\mathbf{A}_3$ such that for all inputs $\vec{p}(\vec{x}) = \vec{u}$, as in (5.1), and $\vec{h} \in \mathsf{B}_n^k$,

$\mathbf{A}_2(n, b)$ decides in $f_2^{(n,b)}\big(\mathrm{size}(\vec{p}(\vec{x}) = \vec{u}) + \mathrm{size}(\vec{h})\big)$ steps whether $\vec{h}$ is a solution of $\vec{p}(\vec{x}) = \vec{u}$, and (5.5)

$\mathbf{A}_3$ computes and stores the number $\mathrm{size}(\vec{p}(\vec{x}) = \vec{u})$ in $f_3\big(\mathrm{size}(\vec{p}(\vec{x}) = \vec{u})\big)$ steps. (5.6)

Let $\mathbf{A}(n, b)$ and $f^{(n,b)}$ be chosen according to (5.3). We can assume that $f^{(n,b)}$ is of the form $f^{(n,b)}(\xi) = \xi^{d(n,b)}$ for some $d(n, b) \in \mathbb{N}^+$. Then $\mathbf{A}(n, b)$ halts in $\big(\mathrm{size}(\vec{p}(\vec{x}) = \vec{u})\big)^{d(n,b)}$ steps for any solvable input $\vec{p}(\vec{x}) = \vec{u}$ but we do not know what $\mathbf{A}(n, b)$ does and whether it ever halts at other inputs. Using (5.4)–(5.6), we define another algorithm $\mathbf{B}(n, b)$ as follows. The input of $\mathbf{B}(n, b)$ is an equation $\vec{p}(\vec{x}) = \vec{u}$ from (5.2); let $s := \mathrm{size}(\vec{p}(\vec{x}) = \vec{u})$. The first task of $\mathbf{B}(n, b)$ is to save a copy of $\vec{p}(\vec{x}) = \vec{u}$; this needs $f_0(s)$ steps where $f_0$ is a polynomial not depending on the parameters $n$ and $b$ and the input $\vec{p}(\vec{x}) = \vec{u}$. The second part of $\mathbf{B}(n, b)$ is $\mathbf{A}_3$, which borrows the input $\vec{p}(\vec{x}) = \vec{u}$ from $\mathbf{B}(n, b)$ and puts $s$ to the output stream in $f_3(s)$ steps. The next part of $\mathbf{B}(n, b)$ is $\mathbf{A}_1(d(n, b))$, which considers the output of $\mathbf{A}_3$ as an input and puts $f^{(n,b)}(s) = s^{d(n,b)}$ into a (counter) variable $c$ in $f_1^{(d(n,b))}(s)$ steps. Then $\mathbf{B}(n, b)$ performs the steps of $\mathbf{A}(n, b)$ and the "$(\alpha)$–$(\delta)$-strides" given below alternately. (Here a "stride" means a finite sequence of steps, possibly just one step.) After the first $\mathbf{A}(n, b)$-step, $\mathbf{B}(n, b)$ performs the following strides.

($\alpha$) $\mathbf{B}(n, b)$ decreases $c$ by 1.
($\beta$) $\mathbf{B}(n, b)$ verifies whether $c = 0$.
($\gamma$) $\mathbf{B}(n, b)$ checks whether $\mathbf{A}(n, b)$ has halted.
($\delta$) If $c = 0$ or $\mathbf{A}(n, b)$ has halted then, using the saved copy of $\vec{p}(\vec{x}) = \vec{u}$, $\mathbf{B}(n, b)$ executes $\mathbf{A}_2(n, b)$ to verify whether the output of $\mathbf{A}$ is a solution of $\vec{p}(\vec{x}) = \vec{u}$. If $\mathbf{A}_2(n, b)$ terminates with "yes", then $\mathbf{B}(n, b)$ outputs "yes, the equation is solvable" and halts. Otherwise, if $\mathbf{A}_2(n, b)$ terminates with "no", then $\mathbf{B}(n, b)$ outputs "no, the equation is not solvable" and halts.

After these $(\alpha)$–$(\delta)$-strides, the next $\mathbf{A}(n, b)$-step is performed, then the $(\alpha)$–$(\delta)$-strides again, etc. The kernel of the $(\delta)$-stride is its part following the premise "if $c = 0$ or $\mathbf{A}(n, b)$ has halted"; this kernel is performed only once. As $c \le s^{d(n,b)}$, there is a polynomial $f_4^{(n,b)}$, not depending on the input of $\mathbf{B}(n, b)$, such that each of the $(\alpha)$–$(\gamma)$-strides can be done in $f_4^{(n,b)}(s)$ many steps and, furthermore, the same holds for every $\mathbf{A}$-step (since it is only a one-step stride) and for the condition part of $(\delta)$. The $\mathbf{A}$-step, $(\alpha)$, $(\beta)$, $(\gamma)$, and the premise of $(\delta)$ are performed $f^{(n,b)}(s) = s^{d(n,b)}$ times, each. The kernel of the $(\delta)$-part, which is performed only once, is the same as $\mathbf{A}_2(n, b)$. The input of $\mathbf{A}_2(n, b)$ in this case is (the saved copy of) $\vec{p}(\vec{x}) = \vec{u}$ (of size $s$) together with $\vec{h}$, taken from the output stream of $\mathbf{A}(n, b)$. (Even if $\mathbf{A}(n, b)$ does not halt, there is a memory space — or, in case of a Turing machine, there is an output tape — where $\vec{h}$ is expected when it exists.) As an element of $\mathsf{B}_n$ can be stored in $n$

3 and "input string" rather than "input equation", but this distinction would not make an essential difference as the syntax of the input string can be checked in polynomial time.

4 We can think of the commands in low-level computer programming languages but not of compound commands like "NextPrimeAbove(n)" of "InvertMatrix(A)" in high-level programming languages.

bits, $\mathrm{size}(\vec{h}) = nk$. Here $n$ is a constant and $k \le s$ since $\vec{x}$ has $k$ components that occur in $\vec{p} = (\vec{x}) = \vec{u}$. Hence, $\mathrm{size}(\vec{h}) \le ns$, whereby $\mathbf{A}_2(n,b)$ decides in $f_2^{(n,b)}(s+ns) = f_2^{(n,b)}((n+1)s)$ steps whether the output of $\mathbf{A}(n,b)$ is a solution of our equation. Therefore, $\mathbf{B}(n,b)$ halts after

$$g^{(n,b)}(s) := f_0(s) + f_3(s) + f_1^{(d(n,b))}(s) + f^{(n,b)}(s) \cdot f_4^{(n,b)}(s) + f_2^{(n,b)}((n+1)s) \tag{5.7}$$

steps. As we treat the parameters $n$ and $b$ as constants, $g^{(n,b)}$ is a univariate polynomial. Since the simulated $\mathbf{A}$ finds any solution before the counter $c$ becomes 0, $\mathbf{B}$ correctly decides whether $\vec{p}(\vec{x}) = \vec{u}$ has a solution or not. That is, $\mathbf{B}$ solves $\mathrm{DPr}(n,b)$. We have seen that $g^{(n,b)}$ in (5.7) is a polynomial, whereby

$$\mathrm{DPr}(n,b), \text{ defined in (5.2), is in } \mathbf{P}, \text{ and } \mathbf{B}(n,b) \text{ solves it in } g^{(n,b)}(\text{input size}) \text{ steps.} \tag{5.8}$$

As the next step of the proof, we focus on another problem. An input of the 3-colorability problem is a finite graph $G = (\{1,\dots,t\}, E)$, where $t \in \mathbb{N}^+$ and the edge set $E$ consists of some two-element subsets of $\{1,\dots,t\}$. By a 3-coloring we mean a sequence $C_1, C_2, \dots, C_t$ of nonempty subsets of $\{r,w,g\} := \{\text{red}, \text{white}, \text{green}\}$ such that whenever $\{i,j\} \in E$, then $C_i \cap C_j = \emptyset$. (This is equivalent to the original definition, where each vertex has exactly one color since we can change a color $\xi$ to $\{\xi\}$ and, in the converse direction, we can take the lexicographically first element of each nonempty subset of $\{r,w,g\}$.)

To reduce the 3-colorability problem to problem $\mathrm{DPr}(n,b)$, let $G$ be the graph from the previous paragraph, and let $s_G := \mathrm{size}(G)$. Let $r_1, w_1, g_1, \dots, r_t, w_t, g_t$ be variables; their task is to determine a 3-coloring. These $k := 3t$ variables form the components of a vector denoted by $\vec{x}$. For each vertex $v \in \{1,\dots,t\}$ and each edge $\{i,j\} \in E$, consider the $k$-ary lattice terms

$$a_v(\vec{x}) := r_v \vee w_v \vee g_v \text{ and } b_{ij}(\vec{x}) := (r_i \wedge r_j) \vee (w_i \wedge w_j) \vee (g_i \wedge g_j). \tag{5.9}$$

For $m \in \{2,\dots,t\}$, let

$$p_1 := \bigwedge\{a_v(\vec{x}) : v \in \{1,\dots,t\}\} \quad \text{and} \quad p_m := \bigvee\{b_{ij}(\vec{x}) : \{i,j\} \in E\}, \tag{5.10}$$

$\vec{p} := (p_1,\dots,p_t)$, and $\vec{u} = (u_1,\dots,u_t) := (1,0,\dots,0)$, where[5] $0 = 0_{\mathsf{B}_n}$ and $1 = 1_{\mathsf{B}_n}$. We claim that

$$\vec{p}(\vec{x}) = \vec{u} \text{ has a solution in } \mathsf{B}_n^k \text{ if and only if } G \text{ is 3-colorable.} \tag{5.11}$$

To see this, assume that $C_1,\dots,C_t$ are color sets witnessing that $G$ is 3-colorable. For $v \in \{1,\dots,t\}$, let $r_v := 1 \iff r \in C_v$, $w_v := 1 \iff w \in C_v$, and $g_v := 1 \iff g \in C_v$. If a variable is not 1, then let it be 0. Clearly, these assignments yield a solution in $\mathsf{B}_n^k$ of $\vec{p}(\vec{x}) = \vec{u}$. Conversely, assume that $\vec{p}(\vec{x}) = \vec{u}$ has a solution $\vec{x}' = (r_1', w_1', g_1', \dots, r_t', w_t', g_t') \in \mathsf{B}_n^k$ for the unknown $\vec{x}$, and fix an atom $e$ in $\mathsf{B}_n$. For each $v \in \{1,\dots,t\}$, define $C_v \subseteq \{r,w,g\}$ by the rules $r \in C_v \iff e \le r_v'$, $w \in C_v \iff e \le w_v'$, and $g \in C_v \iff e \le g_v'$. For any $v \in \{1,\dots,t\}$, $p_1(\vec{x}') = u_1 = 1$ and (5.10) give that $e \le 1 = p_1(\vec{x}') \le a_v(\vec{x}') = r_v' \vee w_v' \vee g_v'$. Using the well-known fact that every atoms (and, in fact, any join-irreducible element) in a finite distributive lattice is join-prime, we obtain that at least one of the inequalities $e \le r_v'$, $e \le w_v'$, and $e \le g_v'$ holds, whereby $C_v$ is nonempty. For $\{i,j\} \in E$, $p_2(\vec{x}') = u_2 = 0$ and (5.10) give that $(r_i' \wedge r_j') \vee (w_i' \wedge w_j') \vee (g_i' \wedge g_j') = 0$. Hence, $r_i' \wedge r_j' = w_i' \wedge w_j' = g_i' \wedge g_j' = 0$. If, say, we had that $r \in C_i \cap C_j$, then $e \le r_i'$ and $e \le r_j'$ would lead to $e \le r_i' \wedge r_j' = 0$, a contradiction. Hence, $r \notin C_i \cap C_j$, and similarly for the colors $w$ and $g$, showing that $C_i \cap C_j = \emptyset$. So $C_1,\dots,C_t$ witness that $G$ is 3-colorable, and we have shown (5.11).

Let $s_G := \mathrm{size}(G)$ and $s$ stand for the size of $G$ and, complying with the earlier notation, the size of the equation in (5.11), respectively. It is not hard to see that there are polynomials $\mu$ and $\eta$ not depending on $G$ such that $\vec{p}(\vec{x}) = \vec{u}$ can be constructed from $G$ in $\eta(s_G)$ steps and $s \le \mu(s_G)$. We define an algorithm $\mathbf{M}$ as follows. For a graph $G$ as an input, $\mathbf{M}$ constructs $\vec{p}(\vec{x}) = \vec{u}$, then it calls $\mathbf{B}(n,b)$ and, finally, it outputs the same answer that $\mathbf{B}(n,b)$ has given. By (5.8) and (5.11), $\mathbf{M}$ solves the 3-colorability problem. As $s = \mathrm{size}(\vec{p}(\vec{x}) = \vec{u}) \le \mu(s_G)$, $\mathbf{M}$ does so in $\nu(s_G) := \eta(s_G) + g^{(n,b)}(\mu(s_G))$ steps. As $\nu$ is a polynomial, we obtain that the 3-colorability problem is in $\boldsymbol{P}$. On the other hand, we know from Garey, Johnson, and Stockmeyer [14], see also Dailey [12, Theorems 3 and 4], that

5 Note that, to reduce the size of $\vec{p}$, we could have let $p_3 = \dots = p_t := r_1 \vee w_1 \vee g_1$ together with $u_3 = \dots = u_t = 1$.

3-colorability is an **NP**-complete problem. Now that an **NP**-complete problem turned out to be in **P**, it follows that **NP** = **P**, completing the proof. □

**REMARK 5.2.** The proof above has reduced the **NP**-complete 3-colorability problem to problem $\mathrm{DPr}(n, b)$, defined in (5.2). Therefore, $\mathrm{DPr}(n, b)$ is also an **NP**-complete problem for any $2 \le b \in \mathbb{N}^+$ and any $n \in \mathbb{N}^+$.

## 6. WARNING AND PERSPECTIVES

Sometimes, cryptography goes after conjectures and experience if no rigorous mathematical proof is available. For example, we only believe that the RSA crypto-system is safe and **P** ≠ **NP**. This can justify that no proof occurs in Sections 4 and 6. However, we have two remarks.

**REMARK 6.1.** An authentication or cryptographic protocol with a hard underlying problem need not be safe. Thus, Proposition 5.1 in itself does not guarantee the safety of protocol (4.1).

In part, this is so because the Adversary might break a protocol without solving the underlying problem. For example, the Adversary can break (4.1) with parameters given in (4.2) even though the underlying problem is hard by Proposition 5.1.

The second explanation of Remark 6.1 is that even a hard problem can have many easy instances (i.e., inputs) for which the computation is fast. For example, there are fast algorithms that work on the "average cases" of some **NP**-complete problems even though these algorithms cannot deal with the hard cases; see the Introduction in Wang [25] for details. It is needless to say how much harm the Adversary can cause if he can apply a fast algorithm for, say, every tenth case.

**REMARK 6.2.** We would need an algorithm that chooses $\vec{p}$ for protocol (4.1) so that, modulo this algorithm, the average case of the underlying problem $\mathrm{CPr}(n, b)$, defined in (5.1), is hard.

Even though Czédli [8] and the extended version of the present paper give some ideas how we could choose a random $\vec{p}$ and these heuristic ideas are likely to satisfy the requirement of Remark 6.2, these ideas are not supported by proofs. This is why we mention the tiling problem from Levin [18]; see also Wang [25] as a secondary source. This problem, which we do not define here, includes a probabilistic distribution. Levin [18] proved that, with respect to this distribution, the average case of the tiling problem is hard in some (sophisticated) sense.

Similarly to the proof of Proposition 5.1, see also Remark 5.2, we can reduce the tiling problem to $\mathrm{DPr}(n, b)$ defined in (5.2). (The **NP**-completeness of $\mathrm{DPr}(n, b)$ implies the existence of such a reduction but we need a concrete one that is sufficiently economic.) Then we can pick a random $\vec{p}$ for (4.1) so that first we take a random instance $y$ of the tiling problem and then we let $\vec{p}$ be the polynomial vector in the "$\mathrm{DPr}(n, b)$-representative" of $y$. As $y$ and, thus, the corresponding equation in $\mathrm{DPr}(n, b)$ are hard on average, we can hope that this $\vec{p}$ turns $\mathrm{CPr}(n, b)$, the underlying problem of (4.1), hard. However, the details of this plan have not been elaborated yet. In particular, we have not proved that the above-suggested method of choosing $\vec{p}$ (which is only a part of the $\mathrm{DPr}(n, b)$-representative of $y$) turns $\mathrm{CPr}(n, b)$ (which is another problem) hard on average. Furthermore, it is not clear whether the parameters suggested in Section 4 are large enough for the plan suggested above. Hence, the heuristic ideas of [8] and the extended version of the present paper and would also deserve further investigations.

Finally, we note that protocol (4.1) becomes more economic if we decrease $k$ so that $\vec{h}$ remains a generating vector of $\mathsf{B}_n$; this is the point where Sections 2 and 3 are connected to the Section 4.

## ACKNOWLEDGMENT

This research was supported by the National Research, Development and Innovation Fund of Hungary, under funding scheme K 138892. [6]

6 Editor and Reviewer: The most up-to-date preprints of my papers are available from my website (if you click on this link). Most of the links in the References section will be removed from the final version.

## REFERENCES

[1] Ahmed, D., and Czédli, G. (1+1+2)-generated lattices of quasiorders. Acta Sci. Math. (Szeged) 87 (2021), 415–427.
[2] Balbes, R. Projective and injective distributive lattices. Pacific J. Math. 21 (1967), 405–420.
[3] Chajda, I., and Czédli, G. How to generate the involution lattice of quasiorders? Studia Sci. Math. Hungar. 32 (1996), 415–427.
[4] Czédli, G. Four-generated large equivalence lattices. Acta Sci. Math. (Szeged) 62 (1996), 47–69.
[5] Czédli, G. Lattice generation of small equivalences of a countable set. Order 13 (1996), 11–16.
[6] Czédli, G. (1+1+2)-generated equivalence lattices. J. Algebra 221 (1999), 439–462.
[7] Czédli, G. Four-generated quasiorder lattices and their atoms in a four generated sublattice. Communications in Algebra 45, 9 (2017), 4037–4049.
[8] Czédli, G. Four-generated direct powers of partition lattices and authentication. Publicationes Mathematicae (Debrecen) 99 (2021), 447–472.
[9] Czédli, G. Generating some large filters of quasiorder lattices. arXiv:2302.13911 (2023).
[10] Czédli, G., and Kulin, J. A concise approach to small generating sets of lattices of quasiorders and transitive relations. Acta Sci. Math. (Szeged) 83 (2017), 3–12.
[11] Czédli, G., and Oluoch, L. Four-element generating sets of partition lattices and their direct products. Acta Sci. Math. (Szeged) 86 (2020), 405–448.
[12] Dailey, D. P. Uniqueness of colorability and colorability of planar 4-regular graphs are np-complete. Discrete Mathematics 30, 3 (1980), 289–293.
[13] Dolgos, T. Generating equivalence and quasiorder lattices over finite sets. BSc Theses, University of Szeged (2015).
[14] Garey, M. R., Johnson, D. S., and Stockmeyer, L. Some simplified np-complete problems. In Sixth Annual ACM Symposium on Theory of Computing (Seattle, Washington). Association for Computing Machinery, New York, 1974, pp. 47–63 (available here).
[15] Grätzer, G. Lattice Theory: Foundation. Birkhäuser, Basel, 2011.
[16] Kulin, J. Quasiorder lattices are five-generated. Discuss. Math. Gen. Algebra Appl. 36 (2016), 59–70.
[17] Kuratowski, K. Sur l'état actuel de l'axiomatique de la théorie des ensembles. Ann. Soc. Polon. Math. 3 (1925), 146–147.
[18] Levin, L. A. Average case complete problems. SIAM J. Comput. 15, 1 (February 1986), 285–286 (available here).
[19] McKenzie, R. N., McNulty, G. F., and Taylor, W. F. Algebras, Lattices, Varieties. Vol. 1. Wadsworth & Brooks/Cole, Monterey, California, 1987.
[20] Rich, E. Computability, and Complexity – Theory and Applications. Pearson Prentice Hall, Upper Saddle River, NJ, 2008.
[21] Sperner, E. Ein Satz über Untermengen einer endlichen Menge. Math. Z. 27, (available here) (1928), 544–548.
[22] Strietz, H. Finite partition lattices are four-generated. In Proceedings of the Lattice Theory Conference (Ulm,1975), G. Kalmbach and E. T. Schmidt, Eds. Universität Ulm, Ulm, 1975, pp. 257–259.
[23] Strietz, H. Über Erzeugendenmengen endlicher Partitionverbände. Studia Sci. Math. Hungarica 12 (1977), 1–17.
[24] Takách, G. Three-generated quasiorder lattices. Discuss. Math. Algebra Stochastic Methods 16 (1996), 81–98.
[25] Wang, J. Average-case computational complexity theory. In Complexity Theory Retrospective II, Hemaspaandra and Selmen, Eds. Springer, (available here) , 1977, pp. 295–328.
[26] Zádori, L. Generation of finite partition lattices. In Lectures in universal algebra (Proc. Colloq. Szeged, 1983), S. L. and S. Á., Eds. North-Holland, Amsterdam, 1986, pp. 573–586.