Novi Sad J. Math. Vol. 55, No. 2, 2025, 167–183
https://doi.org/10.30755/NSJOM.16637

Generating Boolean lattices with few elements and exchanging session keys[1]

Gábor Czédli [2]

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 $n \leq \mathrm{Sp}(k)$. Based on this fact and our 2021 protocol based on equivalence lattices, we outline a cryptographic protocol for exchanging session keys, that is, frequently changing secondary keys. In the present paper, which belongs mainly to lattice theory, we do not elaborate and prove those details of this protocol that modern cryptology would require to guarantee security; the security of the protocol relies on heuristic considerations. However, as a first step, we prove that if an eavesdropper could break every instance of an easier protocol in polynomial time, then **P** would equal **NP**. As a byproduct, it turns out that in each nontrivial finite lattice that has a prime filter, in particular, in each nontrivial finite Boolean lattice, the solvability of systems of equations with constant-free left sides but constant right sides is an **NP**-complete problem.

AMS Mathematics Subject Classification (2010): Primary: 06D99. Secondary: 94A62; 94A60; 68Q25

Key words and phrases: Boolean lattice; smallest generating set; cryptography; session key exchange; **NP**-complete

1. Introduction

1.1. Targeted readership

This is mainly a *lattice theoretic paper* and the main result belongs to lattice theory. However, the targeted readership is not restricted to lattice theorists. Those familiar with the concept of a Boolean lattice and that of **NP**-completeness should have no difficulty in reading the *results* and even most other parts of the paper. Some exceptions occur in Subsection 1.3, which surveys how a series of lattice theoretic investigations lead to the present paper, which could be interesting outside lattice theory and, perhaps, even outside mathematics.

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

[2]University of Szeged, Bolyai Institute, Szeged, Aradi vértanúk tere 1, Hungary 6720,
e-mail: czedli@math.u-szeged.hu, ORCID iD: orcid.org/0000-0001-9990-3573,
url: http://www.math.u-szeged.hu/~czedli/

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 B_n is isomorphic to (and so it can be defined as) the powerset lattice $(P(\{1, \dots, n\}); \cup, \cap)$, whence $|\mathsf{B}_n| = 2^n$. A subset X of B_n is a *generating set* of B_n if no proper subset of B_n that is closed with respect to join ($\vee$) and meet ($\wedge$) includes X as a subset. In Theorem 2.1, we are going to determine the smallest $k \in \mathbb{N}^+$ such that B_n has a k-element generating set. Section 3, which is computer-assisted, indicates that if $k' > k$ but k' is still small, then B_n has many k'-element generating sets. Based on the plenty of these generating sets, Section 4 outlines a cryptographic protocol for session key exchange. Section 5 shows that if an eavesdropper (that is, a monitoring adversary) could solve every instance of a related but easier lattice theoretic problem in polynomial time, then **P** would equal **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 [28], who gave a new proof of a result of Strietz [23]–[24] 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 [4] 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 [6] extended Zádori's result to $\mathrm{Equ}(A)$ with $|A| = \aleph_0$. Furthermore, Czédli [5, 7] and Takách [25] 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 λ such that $\lambda \leq |A|$. Moreover, the 1999 paper Czédli [7] proved that $\mathrm{Equ}(A)$ has a 4-element non-antichain generating set for these sets A. Note that Kuratowski [18] gave a model of ZFC in which there is no inaccessible cardinal at all.

Around 1999, Vilmos Totik[3] proved that our methods are insufficient to deal with inaccessible cardinals. Hence, the topic was put aside after the 1999 paper Czédli [7], 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 [14], one of Miklós Maróti's students, proved that $\mathrm{Quo}(A)$ is 5-generated for $|A| \leq \aleph_0$, and Kulin [17] extended this result to all sets A such that there is no inaccessible cardinal $\lambda \leq |A|$. Not much later, Czédli [8] and Czédli and Kulin [11] reduced the number of generators by proving that for all sets A such that $|A| \neq 4$ and there

[3] https://en.wikipedia.org/wiki/Vilmos_Totik

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 [8] 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 [23]–[24] and Zádori [28], some sporadic cases have recently been settled in Ahmed and Czédli [1] and Czédli and Oluoch [12].

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 [9] and Czédli and Oluoch [12]. Based on the abundance of the generating sets found in the just mentioned two papers, Czédli [9] in 2021 suggested a protocol (the *2021 protocol* for short) for authentication and cryptography based on lattices. The questions of security of this protocol get easier when the protocol is used only for session key exchange, because then we can practically disregard those adversaries that alter messages; recall from Buegler [3] that a *session key* is a secondary key to be changed before each usage of a cryptographic protocol while the *master key* remains unchanged. In other words, while Czédli [9] puts the emphasis on authentication, now we have a reason to put it on session key exchange. Quite recently, while looking for small generating sets of some filters of quasiorder lattices, a proof in Czédli [10] required to know the smallest size of a generating set of a finite Boolean lattice; this was the immediate motivation for the present paper.

Out of the several directions where the present paper will possibly lead, we only mention the following. A *weak congruence lattice* of an algebra $\mathcal{A} = (A; F)$ is a bunch of congruence lattices along a scaffolding, which is the subalgebra lattice of $\mathcal{A}$; see, e.g., Šešelja, Stepanović, and Tepavčević [26] for an exact definition and references. Now if A is finite and $F = \emptyset$, then we know from Zádori [28] that the congruence lattices in question are equivalence lattices generated by few (at most four) elements, and the scaffolding is also generated by few elements by the main result of the present paper. This raises the question how many of its elements are needed to generate the weak congruence lattice in the $F = \emptyset$ case.

2. Small generating sets of finite Boolean lattices

For $n \in \mathbb{N}^+$, we introduce the "vertical-space-saving" 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 $n \leq \mathrm{Sp}(k)$. Note the rule: $n \leq \mathrm{Sp}(k) \iff$

$\mathrm{LASp}(n) \leq k$; this explains the acronym, which comes from "Left Adjoint of Sp".

Theorem 2.1. *For $n, k \in \mathbb{N}^+$, B_n has an at most k-element generating set if and only if $n \leq \mathrm{Sp}(k)$ or, equivalently, if and only if $\mathrm{LASp}(n) \leq k$. In particular, $\mathrm{LASp}(n)$ is the smallest possible size of a generating set of 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 B_n. As usual, for an element u of a lattice L, $\downarrow u$ and $\uparrow u$ will stand for $\{x \in L : x \leq u\}$ and $\{x \in L : x \geq u\}$, respectively. First, we show that for any subset Y of 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 $Y = \{b_1, \ldots, b_m\}$, generates B_n and B_n is distributive, $a = t(b_1, \ldots, b_m)$ 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 "$\geq$" part of (2.3). The "$\leq$" is trivial, and we have proved (2.3).

Next, we claim that for any subset G of B_n,

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

To show this, assume that G is a k-element generating set of 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 $f_0 \colon X \to G$. The freeness of $\mathrm{FS}_\wedge(X)$ allows us to extend f_0 to a meet-homomorphism $f \colon \mathrm{FS}_\wedge(X) \to M$, 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, allows us to take an injective map $g \colon \mathrm{At}(\mathsf{B}_n) \to \mathrm{FS}_\wedge(X)$ such that, for all $a \in \mathrm{At}(\mathsf{B}_n)$, $f(g(a)) = a$. If we had that $g(a) \leq g(a')$ 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 $a \leq a'$ and contradicting that a and a' are distinct atoms of 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 [20, Page 240, §4] that $\{1\} \cup \mathrm{FS}_\wedge(X)$ is order isomorphic to $\mathsf{B}_{|X|} = \mathsf{B}_k$. So B_k has an n-element antichain. By Sperner's theorem [22], see also Grätzer [16, page 354], any antichain in B_k has at most $\mathrm{Sp}(k)$ elements. This implies (2.4) and the "only if" part of the theorem.

Next, observe that

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

It suffices to show the first part for $m = n - 1$. Let c be a coatom (that is, a lower cover of 1) in B_n. Then $\downarrow c \cong \mathsf{B}_m$. The function $f \colon \mathsf{B}_n \to \downarrow c$ 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 $n \leq \mathrm{Sp}(k)$; we are going to show that B_n has an at most k-element generating set. Based on (2.5), we can assume that $n = \mathrm{Sp}(k)$. As 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 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 $a \leq h$. Writing this in a more useful way,

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

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

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

Proof. As B_m is a sublattice of B_n for any $m \leq n$, we can assume that $n = \mathrm{Sp}(k)$. Theorem 2.1 yields a surjective homomorphism $f\colon \mathrm{FD}(k) \to \mathsf{B}_n$. Let $h\colon \mathsf{B}_n \to \mathsf{B}_n$ be the identity map (defined by $x \mapsto x$ for $x \in \mathsf{B}_n$). Since B_n is projective in the class of all distributive lattices by Balbes [2, Theorem 7.1(i),(iii')], there is a homomorphism $g\colon \mathsf{B}_n \to \mathrm{FD}(k)$ 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 B_n if the set $\{h_1, \dots, h_k\}$ of its components is a generating set of B_n. Here $|\{h_1, \dots, h_k\}| \leq k$ 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 B_n^k are pairwise distinct with high probability. Therefore, the ratio of the k-element generating sets to all k-element subsets of B_n is close to

the ratio of the k-dimensional generating vectors to all k-dimensional vectors belonging to B_n^k.

A computer program, written by the author and available from his website, counted the generating vectors of B_{1000} among one hundred thousand randomly selected k-dimensional vectors for some values of k. Some of the results are given below.

```
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 B_{1000}^{50} is a 50-dimensional generating vector of B_{1000} with probability at least $1/2$. Note that $\mathrm{LASp}(1000) = 13$.

4. A cryptographic protocol for session key exchange and encrypted communication

In this section, we outline how to tailor the 2021 protocol, see Czédli [9], from equivalence lattices to Boolean lattices but, as opposed to [9], now we put the emphasis on session key exchange. The reader need not be an expert of cryptology. We present only the main ideas in the paper; the caveats in Section 6 warn us that some details and proofs, which modern cryptology would expect, are still missing. As the session key exchange is (almost[4]) absolutely safe if the key remains unused, we are going to include the *usage* of session keys in the suggested cryptographic protocols. This section and the protocols suggested in it rely only on heuristic considerations. In order to mitigate this omen, note that the next section contains a proof of a related statement (and, hopefully, the heuristic considerations have some convincing value). Furthermore, a session key is to be used in some well-known cryptosystem like AES-256 (the 256-bit variant of Advanced Encryption Standard) or Vernam's cipher, and decades of experience show that these cryptosystems are safe when they are used appropriately. The full list of papers and other sources devoted to cryptosystems that can use session keys would possibly be longer than the present paper itself. Therefore, we reference only some introductory items[5] from Wikipedia (https://www.wikipedia.org/) that are sufficient to find lots of related literature but note that the reader need not read these items. However, as this concept occurs in the title, we mention that the terminology *session key* comes from Buegler [3].

A *session key* for a cryptosystem is a symmetric secret key used only once (or, at least, only very few times); symmetry means that encryption and decryption need the same key. The importance of a session key is that while

[4]It will be clear later that if a particular session key $\vec{p}(\vec{h})$ is unused, then $\vec{p}$, which the Adversary can intercept, gives only the information mentioned in Footnote 8; this is not enough to find $\vec{h}$ as there are astronomically many k-dimensional generating vectors.

[5]These Wikipedia items are the following: Advanced_Encryption_Standard , Key-recovery_attack, Chosen_plaintext_attack, Session_key, Gilbert_Vernam (they are clickable).

several traditional cryptosystems are vulnerable if the same key is used repeatedly, they are much safer if a key is used only once. We will always assume that

(4.1) vectors of elements of B_n, session keys, and *plaintexts* (also called *clear texts*) are regarded as *bit vectors*, that is, vectors over the 2-element field, when they are summands or a traditional cipher is applied to them.

With this convention (which is only a trivial question of encoding like ASCII), the well-known Vernam's cipher used with a session key $\vec{c}$ turns a clear text $\vec{x}$ into the *encrypted text* is $\vec{x} + \vec{c}$ (componentwise addition). (We assume that $\vec{x}$ does not have more bits than $\vec{c}$; if $\vec{x}$ has less bits than $\vec{c}$ then only the appropriate initial segment of $\vec{c}$ is used.)

In the subsequent two subsections, we give two versions of our protocol.

4.1. The basic version

In our model, let E be a fixed traditional cryptosystem like Vernam's cipher or AES-256. For a (secret) code $\vec{u}$ and a clear text $\vec{x}$, $E(\vec{u}, \vec{x})$ denotes the *encrypted text* that E produces. Kati[6] communicates with her Bank online. They have previously agreed upon a *secret master key* $\vec{h}$, which is a k-dimensional random generating vector $\vec{h} = (h_1, \dots, h_k)$ of B_n. This master key is a permanent key that both Kati and the Bank know and only they know; this is not a big restriction since most European banks do not allow anonymous clients, whereby Kati and the Bank have to meet before Kati opens a bank account.

(4.2) In addition to E and $\vec{h}$, there are three parameters in the model, k, n, and b. Let, say, $k := 50$, $n := 1000$, and $b := 100$.

When Kati wants to send a clear text $\vec{x} = (x_1, \dots, x_b) \in \mathsf{B}_n^b$ to the Bank, the first step is that she generates a random vector $\vec{p} = (p_1, \dots, p_b)$ of k-ary lattice terms or, rather, she requests[7] such a vector from the Bank; see (4.2).

(4.3)  With this $\vec{p}$ obtained from the Bank, Kati computes the *session key* $\vec{u} := \vec{p}(\vec{h}) = (p_1(\vec{h}), \dots, p_b(\vec{h}))$, uses the encryption E to obtain the encrypted text $E(\vec{u}, \vec{x})$, and sends $E(\vec{u}, \vec{x})$ together with $\vec{p}$ to the Bank. 

Changing their roles (except that the Bank generates $\vec{p}$), the Bank can also send an encrypted message to Kati. Too simple terms[8] should be avoided, of course. There is an Adversary who eavesdrops on the communication channel and intercepts messages. Furthermore, he can alter messages and pretending

[6] The Hungarian variant of "Cathy" and "Kate".

[7] The Bank would certainly vindicate rights to generate $\vec{p}$ and would not allow Kati to do so.

[8] The concept of "not too simple terms" should be defined. For example, we can require that $\{h_1, \dots, h_k\} \cap \{p_1(\vec{h}), \dots, p_b(\vec{h})\} = \emptyset$.

to be Kati or the Bank, he can send fake messages[9]. As exchanging session keys is meaningful only if these keys are used, (4.3) combines this exchange with the use of session keys. Observe that each of $\vec{u}$ and $\vec{x}$ can take $|\mathsf{B}_n|^b = 2^{nb}$ many values. If nb is small, then a random nb-dimensional vector equals $\vec{x}$ with probability 2^{-nb}, which cannot be neglected. So if nb is small, then the Adversary can experiment with a random $\vec{x}$ and he succeeds in breaking the protocol too often. In particular, we note for later reference that

(4.4) if $n = 2$, and $b = 2$, then, on average, the Adversary can break the protocol in every sixteenth step;

this would allow him to steal the money from Kati's bank account easily. It is reasonable to believe that if nb is large, say, if $nb \geq 5000$, then the protocol outlined above is safe. As there are 2^{nb} many candidate vectors for $\vec{x}$, the Adversary cannot try each of them in his lifetime (in fact, not even in the expected lifetime of the Solar System). Similarly, there are so many possible candidates for the master key $\vec{h}$ that the Adversary cannot find it by random trials.

Observe that the Adversary is not in the position to crack the cipher E with a chosen plaintext attack. Indeed,

(4.5) if he alters the $\vec{p}$ component in a message $(\vec{p},\, E(\vec{u},\vec{x}))$, then the addressee works with a corrupted session key, decrypts the message into something nonsense, and stops communication. The same happens if the Adversary changes $(\vec{p},\, E(\vec{u},\vec{x}))$ in any other way or he tries to send his own message; all he can learn is mentioned in Footnote 4 but this is not enough for him.

The chance that an altered or Adversary-made message leads to a meaningful decrypted text can be neglected. However, the Adversary might have a chance to focus on E in case he conjectures what $\vec{x}$ could be; he can even try thousands of candidate $\vec{x}$'s. For example, a few days before the deadline that all citizens have to pay 500 Euros as a local tax, it is reasonable to guess that the clear text in Kati's message is "transfer 500 Euros to the revenue account of the city". So there is a real chance that

(4.6) the Adversary knows a triple $\big(\vec{p},\, E(\vec{p}(\vec{h}),\vec{x}),\, \vec{x}\big)$.

Observe that

(4.7) if E is appropriately chosen, which we assume, then the Adversary hardly has any chance to extract $\vec{u} := \vec{p}(\vec{h})$ from the last two components, $E(\vec{u},\vec{x})$ and $\vec{x}$, of the triple mentioned in (4.6).

In other words, we chose an E that is not vulnerable for *key-recovery attacks*. In Section 5, we are going to have a closer look at the situation when

(4.8) the Adversary acquires a pair $(\vec{p},\vec{u}) = (\vec{p},\vec{p}(\vec{h}))$,

[9]Moreover, the Adversary can deploy some malware on Kati's computer, but this dangerous threat has not much to do with mathematics and so it is disregarded in the paper.

but let us emphasize here that it is very unlikely that the Adversary reaches (4.8) in his lifetime. Even if the Adversary collects this sort of information several times, this is still insufficient for him since there are astronomically many possible master keys. Therefore, we believe that protocol (4.3) is very safe and, in particular, it is safer than protocol E.

Vernam's cipher is known to be far from withstanding a key-recovery attack in situation (4.7), that is, when the Adversary knows $E(\vec{u},\vec{x}) = \vec{u} + \vec{x}$ and $\vec{x}$. Indeed, then he gets $\vec{u}$ by subtraction. The following ad hoc remark might help.

Remark 4.1. Assume that every clear text in our model is a string of characters (with ASCII codes) 0, 1,..., 127. (A value other then $128-1=127$ would not make much difference.) Call these characters *eligible characters*. Also, let us call the characters (with ASCII codes) 128, 129, ..., 255 *false characters*. Denote the clear text by $\vec{y}$. Insert at least as many false characters into $\vec{y}$ as the number of its (eligible) characters in an *appropriate*[10] *random way* to obtain another vector, $\vec{x}$. For example, if $\vec{y} = (65, 78, 32, 88, 58)$, then

$$\begin{aligned}\vec{x} \text{ can be } &(130, 156, 65, 78, 201, 32, 203, 88, 241, 58, 151)\\ \text{or } &(65, 143, 179, 78, 184, 32, 88, 182, 252, 137, 58), \text{ etc.}\end{aligned}$$

Now if Kati wants to send the clear text $\vec{y}$ to the Bank, then she turns $\vec{y}$ to a random $\vec{x}$ described above and applies protocol (4.3). So Kati sends $E(\vec{u},\vec{x})$ to the Bank. Then Bank, armed with $\vec{p}$ and $\vec{u} = \vec{p}(\vec{h})$, decrypts $E(\vec{u},\vec{x})$ to $\vec{x}$ as before and obtains $\vec{y}$ by omitting the false characters from $\vec{x}$.

4.2. A weaker version

In this version, the only purpose is *authentication*; that is, Kati wants to prove her identity to the Bank. This protocol has nothing to do with session keys or E. The parameters and the notations are the same an in (4.3).

(4.9) To authenticate herself, Kati asks a random $\vec{p}$ from the Bank and after receiving $\vec{p}$, she computes and sends $\vec{p}(\vec{h})$ to the Bank.

The idea is that the Bank assumes that (except for the Bank itself) only Kati knows $\vec{h}$ and only she can compute $\vec{p}(\vec{h})$. In our situation, Kati and the Bank do not have a symmetric role since the Bank does not authenticate itself. (When banking online, Kati avoids fake sites and clicks only on the real website of the Bank since her contract with the Bank, signed when she opened the account, contains the Bank's URL.) If the Adversary is passive, that is, if he can monitor messages but cannot alter them and cannot send his own messages, then it is only (4.8) where he can get to. Indeed, as it is clear from (4.9) and even from Footnote 7, the Adversary cannot choose $\vec{p}$ himself. Furthermore, the

[10] We suggest that the false characters and the eligible characters follow similar distributions; this makes it more difficult to separate these two sets of characters with statistical methods.

argument in (4.5) shows that the Adversary cannot alter a message in any other way. Hence, Kati has no reason to be afraid of an active adversary.

However, if protocol (4.9) is adapted by two equal communicating parties so that each of them can play the role of the Bank, then there can be an active adversary, who can alter and send messages. For this Adversary, the chance to find the (master and only) key $\vec{h}$ is now better than in (4.8) since he can choose $\vec{p}$. For this situation, we cannot prove anything but we guess that even the authentication protocol (4.9) would be safe in case of an appropriate strategy of choosing $\vec{p}$.

Remark 4.2. For authentication, it is safer to send a message something like "This is Kati" encrypted using protocol (4.3) fortified with Remark 4.1 rather than going after (4.9). This method works even if (4.3) is adapted by two equal communicating parties.

5. Even an easier problem is hard

This section is motivated by the Adversary's problem in the situation described in (4.8); we study the hardest case of this problem. Note that the result of this section is meaningful, although less motivated, even without referencing Section 4.

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 $\vec{x} = (x_1, \ldots, x_k)$ instead of $\vec{h} \in \mathsf{B}_n^k$, the problem corresponding to (4.8) is this:

(5.1)

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

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

(5.2)

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

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 respective numbers of bits; see (4.1).

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 $\mathrm{CPr}(n, b)$. (In particular, an algorithm solving $\mathrm{CPr}(n, b)$ need not even halt if $\vec{p}(\vec{x}) = \vec{u}$ unsolvable.) 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 $\mathcal{A}_1(d)$ later in the proof). However, we can safely agree to the following terminology:
(5.3)

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

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* $2 \leq b \in \mathbb{N}^+$ *and* $n \in \mathbb{N}^+$, *if* $\mathrm{CPr}(n,b)$, *defined in* (5.1), *is solvable in polynomial time 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 $2 \leq b \in \mathbb{N}^+$, $n \in \mathbb{N}^+$, and $\mathrm{CPr}(n,b)$ is solvable in polynomial time.

In principle, we should have written "Turing machine" in (5.3) rather than "algorithm"[11]. 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 "in polynomial time"; see, for example, Theorem 17.4 in Rich [21]. By the same theorem, for n' computer steps[12] (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,
(5.4)

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

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

(5.5) $\mathcal{A}_2(n,b)$ 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

(5.6) $\mathcal{A}_3$ 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.

[11] 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.

[12] 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.

Let $\mathcal{A}(n,b)$ 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 $\mathcal{A}(n,b)$ halts in $\bigl(\mathrm{size}(\vec{p}(\vec{x}) = \vec{u})\bigr)^{d(n,b)}$ steps for any solvable input $\vec{p}(\vec{x}) = \vec{u}$ but we do not know what $\mathcal{A}(n,b)$ does and whether it ever halts at other inputs. Using (5.4)–(5.6), we define another algorithm $\mathcal{B}(n,b)$ as follows. The input of $\mathcal{B}(n,b)$ is a system of equations $\vec{p}(\vec{x}) = \vec{u}$ from (5.2); let $s := \mathrm{size}(\vec{p}(\vec{x}) = \vec{u})$. The first task of $\mathcal{B}(n,b)$ 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 $\mathcal{B}(n,b)$ is $\mathcal{A}_3$, which borrows the input $\vec{p}(\vec{x}) = \vec{u}$ from $\mathcal{B}(n,b)$ and puts s to the output stream in $f_3(s)$ steps. The next part of $\mathcal{B}(n,b)$ is $\mathcal{A}_1(d(n,b))$, which considers the output of $\mathcal{A}_3$ 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 $\mathcal{B}(n,b)$ performs the steps of $\mathcal{A}(n,b)$ and the "(α)–(δ)-strides" given below alternately. (Here a "stride" means a finite sequence of steps, possibly just one step.) After the first $\mathcal{A}(n,b)$-step, $\mathcal{B}(n,b)$ performs the following strides.

(α) $\mathcal{B}(n,b)$ decreases c by 1.

(β) $\mathcal{B}(n,b)$ verifies whether $c = 0$.

(γ) $\mathcal{B}(n,b)$ checks whether $\mathcal{A}(n,b)$ has halted.

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

After these (α)–(δ)-strides, the next $\mathcal{A}(n,b)$-step is performed, then the (α)–(δ)-strides again, etc. The *kernel* of the (δ)-stride is its part following the *premise* "if $c = 0$ or $\mathcal{A}(n,b)$ has halted"; this kernel is performed only once. As $c \leq s^{d(n,b)}$, there is a polynomial $f_4^{(n,b)}$, not depending on the input of $\mathcal{B}(n,b)$, such that each of the (α)–(γ)-strides can be done in $f_4^{(n,b)}(s)$ many steps and, furthermore, the same holds for every $\mathcal{A}$-step (since it is only a one-step stride) and for the condition part of (δ). The $\mathcal{A}$-step, (α), (β), (γ), and the premise of (δ) are performed $f^{(n,b)}(s) = s^{d(n,b)}$ times, each. The kernel of the (δ)-part, which is performed only once, is the same as $\mathcal{A}_2(n,b)$. The input of $\mathcal{A}_2(n,b)$ 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 $\mathcal{A}(n,b)$. (Even if $\mathcal{A}(n,b)$ 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 B_n can be stored in n bits, $\mathrm{size}(\vec{h}) = nk$. Here n is a constant and $k \leq s$ since $\vec{x}$ has k components that occur in $\vec{p} = (\vec{x}) = \vec{u}$. Hence, $\mathrm{size}(\vec{h}) \leq ns$, whereby $\mathcal{A}_2(n,b)$ decides in $f_2^{(n,b)}(s+ns) = f_2^{(n,b)}((n+1)s)$ steps whether the output of $\mathcal{A}(n,b)$ is a solution of our system of equations. Therefore, $\mathcal{B}(n,b)$ 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 $\mathcal{A}$ finds any solution before the counter c becomes 0, $\mathcal{B}$ correctly decides whether $\vec{p}(\vec{x}) = \vec{u}$ has a solution or not. That is, $\mathcal{B}$ solves $\mathrm{DPr}(n, b)$. We have seen that $g^{(n,b)}$ in (5.7) is a polynomial, whereby

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

As the next step of the proof, we focus on another problem. An input of the 3-*coloring problem* is a finite graph $G = (\{1, \dots, t\}, E_G)$, where $t \in \mathbb{N}^+$ and the edge set E_G consists of some two-element subsets of $\{1, \dots, t\}$. By a 3-*coloring* we mean a sequence C_1, C_2, …, C_t of nonempty subsets of $\{r, w, g\} := \{\text{red, white, green}\}$ such that whenever $\{i, j\} \in E_G$, 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 ξ 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-coloring 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, …, 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 $\{i, j\} \in E_G$, 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

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

$\vec{p} := (p_1, \dots, p_t)$, and $\vec{u} = (u_1, \dots, u_t) := (1, 0, \dots, 0)$, where[13] $0 = 0_{\mathsf{B}_n}$ and $1 = 1_{\mathsf{B}_n}$. We claim that

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

To see this, assume that C_1, …, 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 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 B_n. For each $v \in \{1, \dots, t\}$, define $C_v \subseteq \{r, w, g\}$ by the rules $r \in C_v \iff e \leq r'_v$, $w \in C_v \iff e \leq w'_v$, and $g \in C_v \iff e \leq g'_v$. For any $v \in \{1, \dots, t\}$, $p_1(\vec{x}') = u_1 = 1$ and (5.10) give that $e \leq 1 = p_1(\vec{x}') \leq a_v(\vec{x}') = r'_v \vee w'_v \vee g'_v$. Using the well-known fact that every atom (and, in fact, any join-irreducible element) in a finite distributive lattice is join-prime, we obtain that at least one of the inequalities $e \leq r'_v$, $e \leq w'_v$, and $e \leq g'_v$ holds, whereby C_v is nonempty. For $\{i, j\} \in E_G$, $p_2(\vec{x}') = u_2 = 0$ and (5.10) give that

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

$(r'_i \wedge r'_j) \vee (w'_i \wedge w'_j) \vee (g'_i \wedge g'_j) = 0$. 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 $e \leq r'_i$ and $e \leq r'_j$ would lead to $e \leq r'_i \wedge r'_j = 0$, 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 $C_1, \ldots, C_t$ witness that G is 3-colorable, and we have shown (5.11).

Let $s_G := \text{size}(G)$ 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 μ and η not depending on G such that $\vec{p}(\vec{x}) = \vec{u}$ can be constructed from G in $\eta(s_G)$ steps and $s \leq \mu(s_G)$. We define an algorithm $\mathcal{M}$ as follows. For a graph G as an input, $\mathcal{M}$ constructs $\vec{p}(\vec{x}) = \vec{u}$, then it calls $\mathcal{B}(n, b)$ and, finally, it outputs the same answer that $\mathcal{B}(n, b)$ has given. By (5.8) and (5.11), $\mathcal{M}$ solves the 3-coloring problem. As $s = \text{size}(\vec{p}(\vec{x}) = \vec{u}) \leq \mu(s_G)$, $\mathcal{M}$ does so in $\nu(s_G) := \eta(s_G) + g^{(n,b)}(\mu(s_G))$ steps. As ν is a polynomial, we obtain that the 3-coloring problem is in **P**. On the other hand, we know from Garey, Johnson, and Stockmeyer [15], see also Dailey [13, Theorems 3 and 4], that 3-coloring 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-coloring problem to problem $\text{DPr}(n, b)$, defined in (5.2). Therefore, $\text{DPr}(n, b)$ is also an **NP**-complete problem for any $2 \leq b \in \mathbb{N}^+$ and any $n \in \mathbb{N}^+$.

Even more is true since the only property of B_n that the proof used is that e is join-prime, which means that $\{x : x \geq e\}$ is a prime filter.

Remark 5.3. In every finite lattice L that has a prime filter and for each $2 \leq b \in \mathbb{N}^+$, the solvability of systems of b equations with constant-free left sides but constant right sides is an **NP**-complete problem.

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 some comments.

Remark 6.1. Even though it is harder to break protocol (4.3) than to solve (5.1) and we know from Proposition 5.1 that (5.1) is a hard problem, these facts themselves do not guarantee the safety of protocol (4.3).

This is so because of (at least) two reasons. First, the Adversary might break protocol (4.3) without solving (5.1). For example, we know from Proposition 5.1 that (5.1) is hard even with the parameters given in (4.4) but (4.3) can easily be broken in this case. Second, a hard problem and even an **NP**-complete problem can have many easy instances (that is, inputs) for which the computation is fast and even an "average" instance can be such; see the Introduction in Wang [27] for details.

Remark 6.1 points out that we have not proved that protocols (4.3) and (4.9) are as safe as we believe. Furthermore, an exact proof would need a well-defined strategy of choosing $\vec{p}$, and a definition to capture what safety means. Actually, such a definition exists, see Levin [19] and see also Wang [27], but we do not need its complicated details here. Some ideas about a strategy (in another lattice) are outlined in Czédli [9] but without any proof, and we do not know whether these ideas can be supported by a proof. This is why we mention the *tiling problem* from Levin [19]; see also Wang [27] as a secondary source. This problem, which we do not define here, includes a probabilistic distribution. Levin [19] 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 could reduce the tiling problem to $\mathrm{DPr}(n, b)$ defined in (5.2) but this would require too much tedious work for a lattice theoretic paper. (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 could pick a random $\vec{p}$ for (4.3) 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 $\mathrm{CPr}(n, b)$, a problem easier than (4.3), 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; enlarging the paramaters reduces the practical value of (4.3) .

Finally, we note that protocols (4.3) and (4.9) become more economic if we decrease k so that B_n still has very many k-dimensional generating vectors; this is the point where Sections 2 and 3 are connected to the Section 4.

References

[1] Ahmed, D., and Czédli, G. $(1+1+2)$-generated lattices of quasiorders. *Acta Sci. Math. (Szeged) 87*, 3-4 (2021), 415–427.

[2] Balbes, R. Projective and injective distributive lattices. *Pacific J. Math. 21* (1967), 405–420.

[3] Buegler, M. 3des and secure pin-based electronic transaction processing. *GIAC Security Essentials Certification (GSEC)* (2004).

[4] Chajda, I., and Czédli, G. How to generate the involution lattice of quasiorders? *Studia Sci. Math. Hungar. 32*, 3-4 (1996), 415–427.

[5] Czédli, G. Four-generated large equivalence lattices. *Acta Sci. Math. (Szeged) 62*, 1-2 (1996), 47–69.

[6] Czédli, G. Lattice generation of small equivalences of a countable set. *Order 13*, 1 (1996), 11–16.

[7] Czédli, G. $(1+1+2)$-generated equivalence lattices. *J. Algebra 221*, 2 (1999), 439–462.

[8] Czédli, G. Four-generated quasiorder lattices and their atoms in a four-generated sublattice. *Comm. Algebra 45*, 9 (2017), 4037–4049.

[9] Czédli, G. Four-generated direct powers of partition lattices and authentication. *Publ. Math. Debrecen 99*, 3-4 (2021), 447–472.

[10] Czédli, G. Generating some large filters of quasiorder lattices. *Acta Sci. Math. 91* (2025), 1–21.

[11] Czédli, G., and Kulin, J. A concise approach to small generating sets of lattices of quasiorders and transitive relations. *Acta Sci. Math. (Szeged) 83*, 1-2 (2017), 3–12.

[12] Czédli, G., and Oluoch, L. Four-element generating sets of partition lattices and their direct products. *Acta Sci. Math. (Szeged) 86*, 3-4 (2020), 405–448.

[13] Dailey, D. P. Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete. *Discrete Math. 30*, 3 (1980), 289–293.

[14] Dolgos, T. Generating equivalence and quasiorder lattices over finite sets. BSc Thesis, University of Szeged (in Hungarian).

[15] Garey, M. R., Johnson, D. S., and Stockmeyer, L. Some simplified *NP*-complete problems. In *Sixth Annual ACM Symposium on Theory of Computing (Seattle, Wash., 1974)*. Association for Computing Machinery, New York, 1974, pp. 47–63.

[16] Grätzer, G. *Lattice theory: foundation*. Birkhäuser/Springer Basel AG, Basel, 2011.

[17] Kulin, J. Quasiorder lattices are five-generated. *Discuss. Math. Gen. Algebra Appl. 36*, 1 (2016), 59–70.

[18] Kuratowski, C. Sur l'état actuel de l'axiomatique de la théorie des ensembles. *Ann. Soc. Polon. Math. 3* (1925), 146–147.

[19] Levin, L. A. Average case complete problems. *SIAM J. Comput. 15*, 1 (1986), 285–286.

[20] McKenzie, R. N., McNulty, G. F., and Taylor, W. F. *Algebras, lattices, varieties. Vol. I.* The Wadsworth & Brooks/Cole Mathematics Series. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA, 1987.

[21] Rich, E. A. *Automata, Computability, and Complexity — Theory and Applications*. Pearson Prentice Hall, Upper Saddle River, NJ, 2008.

[22] Sperner, E. Ein Satz über Untermengen einer endlichen Menge. *Math. Z. 27*, 1 (1928), 544–548.

[23] Strietz, H. Finite partition lattices are four-generated. In *Proceedings of the Lattice Theory Conference (Ulm, 1975)* (1975), Univ. Ulm, Ulm, pp. 257–259.

[24] Strietz, H. Über Erzeugendenmengen endlicher Partitionenverbände. *Studia Sci. Math. Hungar. 12*, 1-2 (1977), 1–17.

[25] Takách, G. Three-generated quasiorder lattices. *Discuss. Math. Algebra Stochastic Methods 16*, 1 (1996), 81–98.

[26] Šešelja, B., Stepanović, V., and Tepavčević, A. A note on representation of lattices by weak congruences. *Algebra Universalis 68*, 3-4 (2012), 287–291.

[27] Wang, J. Average-case computational complexity theory. In *Complexity theory retrospective, II*. Springer, New York, 1997, pp. 295–328.

[28] Zádori, L. Generation of finite partition lattices. In *Lectures in universal algebra (Szeged, 1983)*, vol. 43 of *Colloq. Math. Soc. János Bolyai*. North-Holland, Amsterdam, 1986, pp. 573–586.

Received by the editors October 29, 2023
First published online October 8, 2024