IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

VOL. E100-A NO. 12
DECEMBER 2017

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER *Special Section on Information Theory and Its Applications*

Decoding Error of Sudoku for Erasure Channels*

Mikihiko NISHIARA[†a)], ***Senior Member and*** **Ryo HIDAI**[†], ***Nonmember***

SUMMARY Sudoku is a pencil puzzle. The aim of the solver is to complete the 9×9 grid by filling in a digit in every cell according to a certain rule. In this study, we regard the process of solving Sudoku as a process of decoding a codeword from a received word, and show the expected decoding error probability for erasure channels obtained by experiments.

key words: *Sudoku, erasure channels, decoding error probability*

1. Introduction

Sudoku is a pencil puzzle. The solver is presented with a 9×9 grid (Fig. 1), some of whose cells already contain a digit between 1 and 9 by the setter. The aim of the solver is to complete the 9×9 grid by filling in a digit in every cell such that each row, each column and each 3×3 box contains each of the digits 1–9 exactly once. We call a completed Sudoku grid a solution. The solver assumes that any Sudoku puzzle must have only one solution. A Sudoku puzzle which has only one solution is said to be proper.

We regard any solution as a codeword with length 81 for 9-ary memoryless erasure channels. Playing the role of the solver, the decoder tries to complete the received grid which may have blank cells to restore the codeword. The decoder can restore the correct codeword if the received word forms a proper Sudoku puzzle. Otherwise, the decoder chooses one of the possible solutions as the restored codeword.

We have some immediate observations as follows. There are

$$N_0 = 6670903572021072936960 \approx 6.671 \times 10^{21}$$

Sudoku solutions [2], and therefore the coding rate of our system is $\frac{1}{81}\log_9 N_0 \approx 0.2823$. Since the capacity of the erasure channel with erasure probability δ is $C = 1 - \delta$ (in $\log_9$), there is a code with the same coding rate achieving arbitrary small decoding error if $\delta < 0.7177$.

Our interest is in the average decoding error probability of a Sudoku communication system for erasure channels. For uncompleted grids received by the decoder, the number of blank cells does not characterize the decoding error probability of the codeword because this probability depends strongly on the individual configuration of the digits in the grid. However, we cannot examine every codeword with every erasure pattern due to the huge number of cases. Hence, we employ the Monte Carlo method to measure the decoding error probability.

		6						1
	7			6			5	
8			1		3	2		
		5		4		8		
	4		7		2		9	
		8		1		7		
		1	2		5			3
	6			7			8	
2						4		

Fig. 1 One of Sudoku puzzles [1].

The key is how to choose all Sudoku solutions with equal probability, but this is not straightforward. Indeed, one can generate random Sudoku grids by filling in random numbers in the cells. This plan, however, will collapse because of its awkward bias. To manage this task, you can apply the known work about numbering all Sudoku solutions [3]. Once you have a Sudoku index table according to this work, you need only to generate random numbers between 1 and N_0. We can refine this procedure so that we need no index table. In addition, we do not have to distinguish the codewords that have identical performance against erasure. Hence, our mission includes selecting some representative solutions with adequate probabilities. In other words, we classify all possible solutions into smaller groups to reduce the simulation complexity. This technique was originally developed to enumerate all possible solutions [2]. We demonstrate how this technique can be applied to analysis of a code.

From the conclusion of this paper, the performance of our Sudoku communication system is not good. Therefore, this study never aims at its practical use. One of the missions of information theorists should be to introduce information theory to students or ordinary people. Sudoku communication systems are an intimate topic for them. This study provides a concrete result of this topic that can be used in your classroom to involve students in error-correcting codes or information theory.

Recently, Sayir and Sarwar [4] investigated a similar Sudoku communication system. Contrary to our system, their system includes a message encoder to generate codewords. The distribution of the codewords has, unfortunately, no guarantee to be uniform because the encoder is greedy.

Manuscript received January 30, 2017.
Manuscript revised June 15, 2017.

[†]The authors are with Graduate School of Science and Technology, Shinshu University, Nagano-shi, 380-8553 Japan.

*The previous version of this paper was presented at the 2016 International Symposium on Information Theory and Its Applications (ISITA2016).

a) E-mail: mikihiko@shinshu-u.ac.jp

DOI: 10.1587/transfun.E100.A.2641

Although we will not deal with message encoders directly in our study, the discussion of Section 2 implies a straightforward method for generating codewords with equal probability using arithmetic coding.

In the next section, we illustrate how to distribute the codewords, that is, the Sudoku solutions uniformly. In Section 3, we explain the measurement we made to measure the expected decoding error probability of our Sudoku communication system and show the result. Finally in Section 4, we conclude the paper.

2. Generating Solutions

Let S_0 denote the set that consists of all the Sudoku solutions. We hereafter partition S_0 hierarchically to develop a method to select representative solutions adequately. In other words, we construct a tree structure with solutions attached at its leaves.

To indicate a Sudoku grid, we occasionally use a 9×9 matrix (a_{ij}), where i and j specify the row and the column, respectively.

2.1 Reduction

Let S_1 denote the collection of solutions such that the top left 3×3 box is

1	2	3
4	5	6
7	8	9

, (1)

that is,

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

It is enough to choose solutions out of S_1 with equal probability because any permutation of digits on a solution does not affect the error correcting performance of the codeword against erasure and S_0 is the union of the 9! permutations of S_1.

The individual average decoding error probability of a codeword is simply called its performance.

If we permutate the 4th, 5th and 6th columns of any solution in S_1, we get another solution in S_1 and both solutions have the same performance because every cell has the same erasure probability. We can say the same thing about the 7th, 8th, and 9th columns. Similarly, we can exchange the center and right column of boxes. Therefore, we can consider only the solutions in S_1 such that the 6 right cells of the top row $(a_{14}, a_{15}, a_{16}, a_{17}, a_{18}, a_{19})$ are one of the ten patterns:

$$\begin{array}{lll} (4,5,6,7,8,9), & & \\ (4,5,7,6,8,9), & (4,5,8,6,7,9), & (4,5,9,6,7,8), \\ (4,6,7,5,8,9), & (4,6,8,5,7,9), & (4,6,9,5,7,8), \\ (4,7,8,5,6,9), & (4,7,9,5,6,8), & (4,8,9,5,6,7). \end{array}$$

Any solution in S_1 is one of these solutions or one of its $3! \times 3! \times 2 = 72$ permutations. Furthermore, similar to the top row, we can use the same argument for the left column after appropriate permutations on the digits. Let S_2 denote the collection of solutions in S_1 such that the top row and the left column are respectively restricted to the ten patterns. Now, our aim has been reduced to choosing all the solutions in S_2 with a uniform distribution. Note that we have

$$N_0 = |S_0| = 9! \times |S_1| = 9! \times 72^2 \times |S_2|. \quad (2)$$

2.2 Partition by the Top Boxes

We have shown that, for the top row, it is enough to concentrate on ten patterns. As a next step, we will consider the patterns of the top three rows so as to get a fine partition of S_2.

The first pattern $(4, 5, 6, 7, 8, 9)$ has a different property than the others. With this pattern, the 12 remaining cells can be filled as

1	2	3	4	5	6	7	8	9
4	5	6	{7,8,9}			{1,2,3}		
7	8	9	{1,2,3}			{4,5,6}		

,

where $\{a, b, c\}$ indicates the numbers a, b and c in any order, that is,

$$\{a_{24}, a_{25}, a_{26}\} = \{7, 8, 9\}, \quad \{a_{27}, a_{28}, a_{29}\} = \{1, 2, 3\},$$
$$\{a_{34}, a_{35}, a_{36}\} = \{1, 2, 3\}, \quad \{a_{37}, a_{38}, a_{39}\} = \{4, 5, 6\}.$$

Thus, this row pattern gives $(3!)^4$ possible configurations for the top three rows.

Each of the other 9 patterns has a similar property. With $(4, 5, 7, 6, 8, 9)$, the 12 remaining cells can be filled as

1	2	3	4	5	7	6	8	9
4	5	6	{8,9,*a*}			{7,*b*,*c*}		
7	8	9	{6,*b*,*c*}			{4,5,*a*}		

,

where a, b and c stand for 1, 2 and 3 in some order (b and c are interchangeable). Thus, this row pattern gives $3 \times (3!)^4$ possible configurations.

In total, we have

$$(3!)^4 + 9 \times 3 \times (3!)^4 = 36288$$

possible configurations for the top three rows in S_2. For the sake of the following discussion, let $\mathcal{S}_3$ denote the partition of S_2 divided by the 36288 configurations. That is, $\mathcal{S}_3$ is a family of disjoint subsets of S_2.

2.3 Equivalent Subsets

Each subset in $\mathcal{S}_3$ consists of solutions with different erasure performance. Therefore, we can select no representative solution from each subset. However, some subsets have a bijection into another subset such that the two solutions connected by the bijection have identical performance. We say that two subsets are equivalent if, between the two subsets,

there is a bijection that maps a solution to another with the same performance. By finding equivalence classes, we can further reduce the solutions to be generated. Hereafter, we will give bijective operations on subsets in $\mathcal{S}_3$. Most of them were originally developed in order to count the number of Sudoku solutions [2], [3].

One of the simple operations is permutation of the top three rows. However, the resulting subset is not in $\mathcal{S}_3$. Hence, we need additional steps so that it belongs to $\mathcal{S}_3$. First, permutation of digits is applied so that the left top box is (1). Next, permutation of columns is applied so that the top row is one of the ten patterns. We call the last two steps the normalization. For example, consider a subset in $\mathcal{S}_3$ such that the three top rows are

1	2	3	4	5	7	6	8	9
4	5	6	8	9	1	7	2	3
7	8	9	6	2	3	4	5	1

. (3)

Interchanging the two top rows, we obtain

4	5	6	8	9	1	7	2	3
1	2	3	4	5	7	6	8	9
7	8	9	6	2	3	4	5	1

(4)

which is not in $\mathcal{S}_3$. By the first and second steps of normalization, we have

1	2	3	8	9	4	7	5	6
4	5	6	1	2	7	3	8	9
7	8	9	3	5	6	1	2	4

(5)

and

1	2	3	4	8	9	5	6	7
4	5	6	7	1	2	8	9	3
7	8	9	6	3	5	2	4	1

, (6)

respectively. The resulting grid after normalization is in $\mathcal{S}_3$ and equivalent to the original. The operation on the left three columns can be defined in the same manner.

If some digits are configured in a certain "rectangle" form, we can interchange them. Specifically, if $a_{i_1j_1} = a_{i_2j_2} = d_1$ and $a_{i_1j_2} = a_{i_2j_1} = d_2$ for some $i_1, i_2, j_1, j_2, d_1, d_2$, then we can let $a_{i_1j_1} = a_{i_2j_2} = d_2$ and $a_{i_1j_2} = a_{i_2j_1} = d_1$. For example, if

$$a_{21} = a_{37} = 4, \quad a_{31} = a_{27} = 7,$$

then we can let

$$a_{21} = a_{37} = 7, \quad a_{31} = a_{27} = 4.$$

This operation modifies

1	2	3	4	5	7	6	8	9
④	5	6	8	9	1	⑦	2	3
⑦	8	9	6	2	3	④	5	1

into

1	2	3	4	5	7	6	8	9
⑦	5	6	8	9	1	④	2	3
④	8	9	6	2	3	⑦	5	1

.

In this example, 4 and 7 are configured diagonally on a 2×2 subrectangle. By normalization, the latter, which is not in $\mathcal{S}_3$, is modified to

1	2	3	4	5	7	6	8	9
4	5	6	1	9	8	7	2	3
7	8	9	3	2	6	4	5	1

.

The extensions of this method allow us to find equivalent subsets of solutions with a subrectangle of size $k \times 2$ whose entries consist of two rows with the same numbers. As an example of a 3×2 subrectangle, if $a_{i_1j_1} = a_{i_2j_2} = d_1$, $a_{i_1j_2} = a_{i_2j_3} = d_2$ and $a_{i_1j_3} = a_{i_2j_1} = d_3$, then we can let $a_{i_1j_1} = a_{i_2j_3} = d_3$, $a_{i_2j_1} = a_{i_1j_2} = d_1$ and $a_{i_2j_2} = a_{i_1j_3} = d_2$. An example is

①	2	3	④	5	8	6	⑦	9
④	5	6	⑦	9	3	8	①	2
7	8	9	1	6	2	4	5	3

↦

④	2	3	⑦	5	8	6	①	9
①	5	6	④	9	3	8	⑦	2
7	8	9	1	6	2	4	5	3

.

Moreover, we can define an operation on a 2×3 subrectangle such as

①	2	3	④	5	8	6	7	9
④	5	6	⑦	9	3	8	1	2
⑦	8	9	①	6	2	4	5	3

↦

④	2	3	①	5	8	6	7	9
⑦	5	6	④	9	3	8	1	2
①	8	9	⑦	6	2	4	5	3

.

This operation is specified generally as: if $a_{i_1j_1} = a_{i_2j_2} = d_1$, $a_{i_2j_1} = a_{i_3j_2} = d_2$, $a_{i_3j_1} = a_{i_1j_2} = d_3$, we can let $a_{i_3j_2} = a_{i_1j_1} = d_3$, $a_{i_1j_2} = a_{i_2j_1} = d_1$, $a_{i_2j_2} = a_{i_3j_1} = d_2$.

Togami [3] found an operation named $3 \times 2 \times 3$ type, which operates on 3 subrectangles in the top boxes such that each row contains the same numbers. An example is

1	2	3	④	⑤	⑥	⑦	⑧	⑨
④	⑤	⑥	⑧	⑨	⑦	2	3	1
⑦	⑧	⑨	3	1	2	⑥	④	⑤

↦

1	2	3	⑧	⑨	⑦	⑥	④	⑤
⑦	⑧	⑨	④	⑤	⑥	2	3	1
④	⑤	⑥	3	1	2	⑦	⑧	⑨

.

Moreover, the authors found a $2 \times 2 \times 3$ operation such as

①	②	3	④	⑤	6	7	8	9
④	⑤	6	7	8	9	①	②	3
7	8	9	②	①	3	⑤	④	6

↦

④	⑤	3	②	①	6	7	8	9
①	②	6	7	8	9	⑤	④	3
7	8	9	④	⑤	3	①	②	6

.

With a computational exhaustive search by means of the above operations, the authors found that we could reduce the number of the subsets which should be considered from 36288 to 209. In other words, we can select 209 representative subsets from $\mathcal{S}_3$. Previous work [2], [3] reports that we can reduce the number of subsets to 71 by adding other operations we have not mentioned here. However, the authors did not employ other additional operations to reduce the number of subsets, because these operations take a long time on a computer to determine the configurations of the subsets. For the next step, let $\mathcal{S}_4$ denote the partition consisting of the 209 representatives.

2.4 Representative Subsets

We can partition S_2 alternatively by the configurations of the left column of boxes. Let $\mathcal{S}_5$ denote such a partition, which consists of 36288 disjoint subsets of S_2. One may think that, also in this case, we can select 209 representative subsets out of $\mathcal{S}_5$. However, we cannot employ the same analysis because of the following.

By analysis of the operation that exchanges the top two rows, we already know (3) and (6) are equivalent. Applying

permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 4 & 7 & 2 & 5 & 8 & 3 & 6 & 9 \end{pmatrix},$$

and transposing (i.e. switching the row and column indices of) these grids, we find that

1	2	3
4	5	6
7	8	9
2	6	8
5	9	4
3	1	7
8	3	2
6	4	5
9	7	1

and

1	2	3
4	5	6
7	8	9
2	3	8
6	1	7
9	4	5
5	6	4
8	9	2
3	7	1

are equivalent. However, these two facts do not imply that

1	2	3	4	5	7	6	8	9
4	5	6	8	9	1	7	2	3
7	8	9	6	2	3	4	5	1
2	6	8						
5	9	4						
3	1	7						
8	3	2						
6	4	5						
9	7	1						

and

1	2	3	4	8	9	5	6	7
4	5	6	7	1	2	8	9	3
7	8	9	6	3	5	2	4	1
2	3	8						
6	1	7						
9	4	5						
5	6	4						
8	9	2						
3	7	1						

are equivalent because the operation that exchanges the top two rows disturbs the left columns. In fact, by applying this operation to the former and then applying normalization, we have

4	5	6	8	9	1	7	2	3
1	2	3	4	5	7	6	8	9
7	8	9	6	2	3	4	5	1
2	6	8						
5	9	4						
3	1	7						
8	3	2						
6	4	5						
9	7	1						

and then

1	2	3	4	8	9	5	6	7
4	5	6	7	1	2	8	9	3
7	8	9	6	3	5	2	4	1
2	9	1						
5	3	8						
6	4	7						
8	6	5						
3	1	2						
9	7	4						

.

Note that exchanging the left two columns also disturbs the top rows.

To avoid this disorder, we employ the operations that affect only the bottom 6 rows with respect to the left three columns. After a computational exhaustive search, the authors found that we could select 24408 representative subsets from $\mathcal{S}_5$. Let $\mathcal{S}_6$ denote the collection of these representatives.

In summary, we have partitioned S_2 into 36288^2 subsets as

$$S_2 = \bigcup_{T \in \mathcal{S}_3} \bigcup_{L \in \mathcal{S}_5} T \cap L. \tag{7}$$

Also, we have selected $|\mathcal{S}_4| \times |\mathcal{S}_6| = 209 \times 24408 = 5101272$ representatives from this partition.

By the way, the exhaustive searches took a few months to obtain the 209 and 24408 representative subsets using a desktop PC.

2.5 Probability of Representatives

To determine the correct probabilities of the representatives each of which can be denoted by $T \cap L$, $T \in \mathcal{S}_4$, $L \in \mathcal{S}_6$, we have to count the number of solutions in each $T \cap L$, that is, $|T \cap L|$. This task is done by solving many (non-proper) Sudoku puzzles. Note that the algorithm to count solutions does not matter here.

Moreover, we have to count the size of each equivalence class. Let $N(T)$ and $N(L)$ denote the number of equivalents dominated by $T \in \mathcal{S}_4$ and $L \in \mathcal{S}_6$, respectively. Note that representative $T \cap L, T \in \mathcal{S}_4, L \in \mathcal{S}_6$ dominates $N(T) \times N(L)$ subsets in partition (7).

From the counting processes described above, we can obtain the size of S_2 as

$$\begin{aligned} |S_2| &= \sum_{T \in \mathcal{S}_4} \sum_{L \in \mathcal{S}_6} N(T) \times N(L) \times |T \cap L| \\ &= 3546146300288, \end{aligned}$$

which derives the correct value of N_0 by means of (2). The probability of representative subset $T \cap L$ is given by

$$P(T \cap L) = \frac{N(T) \times N(L) \times |T \cap L|}{|S_2|}.$$

After we choose a subset $T \cap L$ with this probability, we choose a solution in $T \cap L$ equiprobably, namely, with probability $1/|T \cap L|$.

3. Decoding Error

Decoding error does not occur when there are three or less blank cells on the received codeword because it gives a proper Sudoku puzzle. However, four or more blank cells may cause decoding error. In this research, we suppose that the decoder cannot restore the correct codeword from any non-proper grid. Further, ~~a~~ recent work [5] has shown that any grid with 65 or more blank cells is non-proper. Thus, to measure the decoding error of a codeword, it is enough to make 4 to 64 blank cells in the solution.

We measured decoding error probability ε_k of Sudoku grids with k blank cells. For each k, we generated about 180 thousand codewords with a uniform distribution. The positions of the blank cells are chosen randomly. Since the k blank cells with erasure probability δ are binomially distributed, which can be written as

$$p_\delta(k) = \binom{81}{k} \delta^k (1-\delta)^{81-k},$$

the decoding error $\varepsilon(\delta)$ is expressed as

$$\varepsilon(\delta) = \sum_{k=0}^{81} p_\delta(k) \varepsilon_k$$

for erasure probability δ of the erasure channel.

Figure 2 shows the decoding error probability ε_k for k blank cells. Figure 3 shows the decoding error probability $\varepsilon(\delta)$ for erasure probability δ.

To evaluate our system, we made a comparison with error correcting random codes for 9-ary erasure channels

with the condition that the blocklength is fixed to 81 and the coding rate is 0.2823. The decoding error probabilities were estimated by the technique of the finite blocklength regime [6]. The dashed line in Fig. 3 denotes the bound achieved by this code.

The decoding error probability of a similar Sudoku communication system by Sayir and Sarwar [4] looks almost the same as ours, though they reported the results only for $\delta < 0.4$. The distribution of the codewords of their system may not be far from uniform, or the individual decoding error probabilities of the codewords may have a small variance. Investigations of these points will be done in future work.

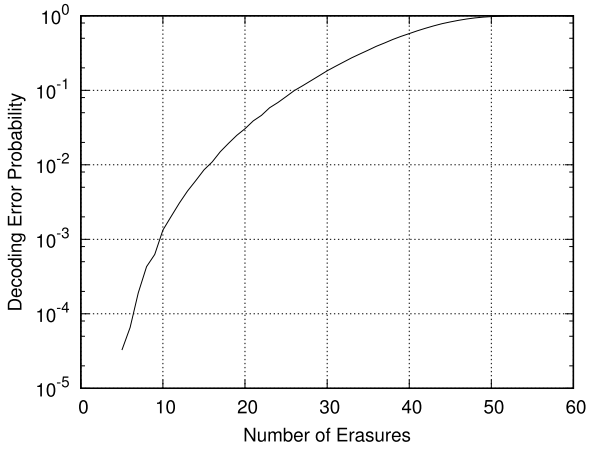


Fig. 2 The decoding error probability ε_k for k blank cells.

Figure 3 also indicates the decoding error probability of a smaller Sudoku that has a 4×4 grid (Fig. 4). This small Sudoku has four row restrictions, four column restrictions, and four 2×2 box restrictions. The number of solutions is 288, and the coding rate is $\frac{1}{16} \log_4 288 \approx 0.2553$. The decoding error probability has been calculated using all 2^{16} erasure patterns. The individual error probability has been correctly evaluated as $(n-1)/n$ if the received grid has n solutions. We also show the achievable bound for 4-ary erasure channels achieved by codes with blocklength 16 and coding rate 0.2553.

4. Conclusion

We regarded solutions of Sudoku as codewords. From a codeword, an erasure channel creates a Sudoku puzzle. The receiver solves the puzzle to restore the original Sudoku grid. We measured the average decoding error probability of Sudoku as a code for erasure channels. To realize a uniform distribution over the codewords, and to reduce the simulation complexity, we expanded the analysis for enumerating the solutions. It worked well. We should say that the performance is not for practical use. The concrete result can be shown in your classroom when you introduce the Sudoku

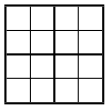


Fig. 4 4×4 Sudoku grid.

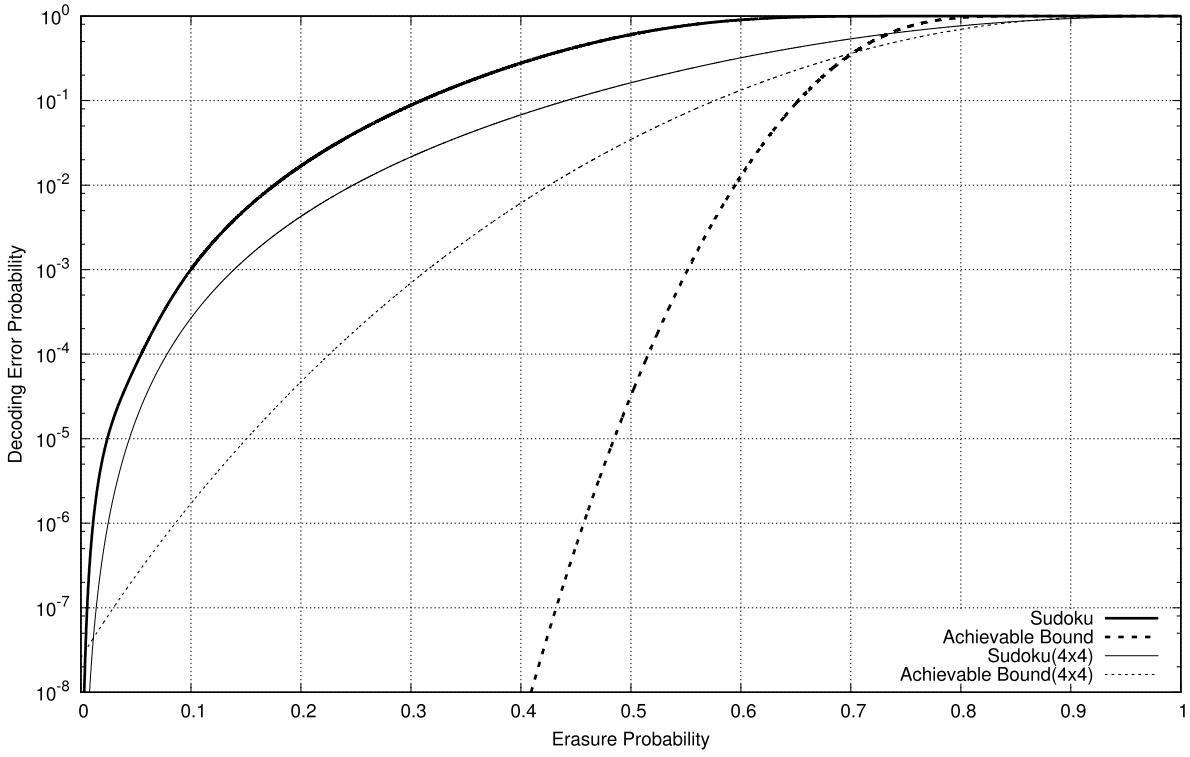


Fig. 3 The decoding error probability $\varepsilon(\delta)$ for erasure probability δ.

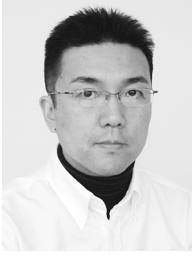
communication system to your students.

Acknowledgment

The authors are grateful to Prof. Ryutaroh Matsumoto of Tokyo Institute of Technology for helpful comments on the finite blocklength regime, and to Mr. Masashi Yamakawa, who is a student of one of the authors, for evaluating the achievable bounds included in Fig. 3. The authors also thank anonymous reviewers for their helpful comments. This work was partly supported by JSPS KAKENHI Grant Number JP26420353.

References

[1] Nikoli, "Sudoku tutorial," http://www.nikoli.com/en/puzzles/sudoku/rule.html
[2] B. Felgenhauer and F. Jarvis, "Enumerating possible Sudoku grids," http://www.afjarvis.staff.shef.ac.uk/sudoku/sudoku.pdf, 2005.
[3] S. Togami, "Genaration and numbering of Sudoku grids," Bachelor Thesis, Tokyo Institute of Technology, 2007 (in Japanese).
[4] J. Sayir and J. Sarwar, "An investigation of SUDOKU-inspired non-linear codes with local constraints," Proc. ISIT2015, pp.1921–1925, 2015.
[5] G. McGuire, B. Tugemann, and G. Civario, "There is no 16-clue Sudoku: Solving the Sudoku minimum number of clues problem," School of Mathematical Sciences, University College Dublin, Ireland, 2013.
[6] Y. Polyanskiy, H.V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," IEEE Trans. Inf. Theory, vol.56, no.5, pp.2307–2359, 2010.



Mikihiko Nishiara received the B. Eng. degree in Communication Engineering from Shibaura Institute of Technology, Tokyo, in 1992, and the M. Eng. degree and the D. degree in information systems from the University of Electro-Communications, Tokyo, in 1998 and 2001, respectively. From 2001 to 2007, he was a research assistant of the Graduate School of Information Systems, the University of Electro-Communications. Since 2007, he has been an Associate Professor in the Faculty of Engineering, Shinshu University. His research interest is in information theory including Shannon theory, data compression, and theoretical analysis of communication systems. He is a member of the IEEE.

Ryo Hidai received the B. Eng. degree and M. Eng. in electrical and electronic engineering from Shinshu University in 2012 and 2014, respectively. He worked on the research of this paper when he was a student.