RAD HAZU. MATEMATIČKE ZNANOSTI
Vol. 25 = 546 (2021): 161-179
DOI: https://doi.org/10.21857/yvjrdcl04y

# MUSIPHER: HIDING INFORMATION IN MUSIC COMPOSITION

Peter Špaček and Pavol Sobota

Abstract. In this paper, we present a new way of hiding information. We store the information directly in the process of composing music, based on musical theory. We created an algorithm to produce music based on binary string, where each bit is transformed into a music composition decision. We follow simple rules to make music, which sounds good. We conducted survey to find whether our solution works, and found promising results of our approach.

## 1. Introduction

We present a new idea of hiding information in music. Not in sound, like modern steganography, but in music composition. The idea of hiding information into music is very old. Gaspar Schott in his book *Schola Steganographica* [11] published in 1680 introduced a simple scheme of how to hide messages in music, where each music note corresponds to one letter. This idea was used repeatedly by many composers, including Robert Schumann, Johann Sebastian Bach, Johannes Brahms, or more recent Dmitri Shostakovitch [5].

We can also find this idea in the 21st century. An interesting example is the use of radio hit in Columbia. In 2010, The Revolutionary Armed Forces of Colombia (FARC) held Colombian soldiers prisoners. Colombian Army decided to send a message of hope in Morse code, hidden in pop-song "Mejores Dias", which was broadcast nationwide [8].

We are presenting our research, a new system of information hiding based on musical theory, and discuss the questions:

1. How to hide information in composition of music;
2. How to make a cryptosystem, which has meaningful information as an input, and music as an output, following Kerckhoffs's principle;
3. How much information we can hide in one song.

2010 *Mathematics Subject Classification.* 94A99.
*Key words and phrases.* Musical cipher, cryptogram, music composition, steganography, cryptography.

## 2. Domain-related definitions

In order to understand the functionality of the system in detail, we present the basic musical concepts, with which we will work. The individual algorithms responsible for composing music will represent these concepts using various sets and their elements, which will be defined in following sections. But because the relations between those concepts are in the musical world, and not in the mathematical, we provide an explanation below, sourced from [1], [4], [9] and [10]:

- A **tone** represents a simple, regular vibration of the sound source. It differs from noise, as it creates a pleasant sensation for the human ear. From an acoustic point of view, a tone has four characteristics: **duration**, **pitch**, **intensity** (or loudness), and **timbre** (quality or color). In this work we will focus mainly on pitch and duration.
- A **pitch** tells us how many vibrations a sound makes in a single unit of time. The human ear is able to hear sounds between 16 Hz and 20 kHz. The standard classical music uses 88 tones, which are easy to distinguish by humans. The "distance" between tones is called an **interval**. An interval between a tone with a specific pitch and the second one with a double frequency is called an **octave**. An octave is split into twelve tones. These tones are usually labeled with letters of the alphabet: a, a#, b (h), c, c#, d, d#, e, f, f#, g, and g#. In order to determine which tone is in which octave, we use the notation C1, C2 etc. The smallest possible distance between the pitches of tones is called a **half step**.
- Ascending or descending sorted set of tones starting from the specific tone up to its octave is called a **scale**. The tones that form the basis of music composition are called a **key**. A **major key** consists of eight tones, and between the *third and fourth* and *seventh and eighth* tone is a half step distance. All other distances are whole step. Each tone in the scale has different musical meaning, so they have specific names:
  - Tonic - tone I (Do)
  - Supertonic - tone II (Re)
  - Mediant - tone III (Mi)
  - Subdominant - tone IV (Fa)
  - Dominant - tone V (Sol)
  - Submediant - tone VI (La)
  - Subtonic - tone VII (Si)
  - Tonic - octave tone VIII (Do)

  The scheme of the tones sequence can be seen in Figure 1.
- A tone **duration** represents the time in which a tone sounds. Conventional duration values are: whole (beat), half, quarter, eighth and sixteenth notes. Every value is always half the time of the previous

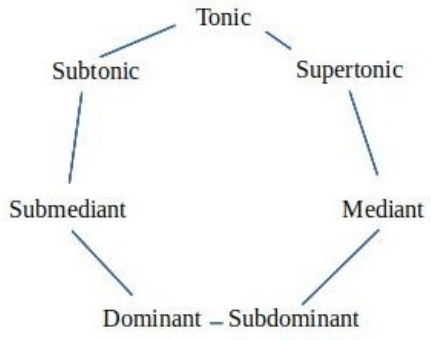


FIGURE 1. Musical degrees

one. Their duration depends on the tempo of a song. A **dotted note** is a note that has a duration one and a half times longer than a note without a dot. For example, if a quarter note is played for one second, a dotted quarter note is played for one and a half seconds. A **rest** indicates an absence of sound in a song. Like tones, rests also have a duration.

- A **bar** or a **measure** is the rhythmic and metric unit of music composition. A bar is split into sections of the same length, which are called **beats**. We determine how long should a beat be: a quarter, a half, etc. We denote the bar based on the split of the bar. For example, if it is divided into two beats with a quarter note duration, we are talking about a two-quarter measure signature.
- A **tempo** of the song defines the speed of the music, i.e. how many beats are in an individual period of a certain time unit, usually a minute. A tempo is closely related to the theme of the song; happier songs are usually faster. If we set the tempo to 120 beats per minute, the quarter note will be half a second.
- A **melody** is formed by a sequence of tones. It must be shaped in a way which is pleasant and rhythmically correct. In addition to the main tones of the melody, there may be other tones, so called ornaments, which make the composition sound more complex.
- A **melodic movement** tells us how fast the melody descends or ascends in terms of the pitch of the tones. A melody that ascends or descends in the smallest tonal intervals (whole step or half step) is

called a **conjunction**. Transition from one tone to another in conjunction is called a "**step**". A melody that ascends or descends in all other intervals (bigger) is called a **disjunction**. Transition between two tones is called a "**leap**". In general, melody is a combination of conjunctive and disjunctive melodic movements.

- A melodic **phrase** is a group of tones in a melody, usually 4 bars. An example can be seen in Figure 2. It ends with a **cadence** that gives the impression of termination, or expectations to continue with another motive.

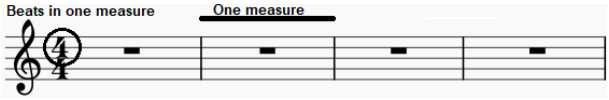


FIGURE 2. Phrase

- A **chord** is a group of at least three concurrent tones. Tones that create a chord are called **chord tones**. A **triad** is the simplest chord. In our work, we mostly use triads. Its lowest tone is called a base tone or a **root**. The root determines the name of the chord. The next tone is the tone III (2 steps distance) and tone IV (4 steps distance from the root ascending). A **chord progression** determines in what order the chords appear in our composition.
- A **bass line** refers to a melodic line, which is played at the bottom of the sound interval. This is usually one and a half octave lower than the C5 tone. The role of the bass line is to emphasize harmony and harmonious movement. In order for the bass line to comply with the melodic line in a higher pitch, their tones must be chord tones in certain places. In practice, we achieve this by having a chord tone according to position in chord progression in the first and the third measure.

## 3. System design

As mentioned in [3], a steganographic system is defined as a specific steganographic algorithm, which specifies the method of embedding (as well as the method of extraction). We define the set of all possible messages, set of "covers" - files or in our case musical pieces, and also key space. We need to set a function to hide (embed) the data, and, method to extract the data from a cover. The maximum size of the secret message that can be inserted into the cover is the called capacity of the cover.

But unlike modern steganography, we do not hide information in the sound, but in the music composition, similarly to the musical cryptograms of

classical composers. Our proposed system takes the information and transforms it into the musical structure of the song. It uses harmony, rhythm and melody, so that would not be easy to tell that the song was composed automatically and has a hidden meaning.

The challenge was to design the system that complied with the Kerkhoff principle [6] and is resistant to the practices used to crack classic ciphers [2].

We solved it with the system design that consists of two modules, as can be seen in Figure 3:

- crypto-module (encrypter): handles diffusion and confusion. The whole system is deterministic, so with the same key and the same input, we get the same music.
- stegano-module (encoder): is a music composition module. For decisions in music composition, we need high entropy. This is provided by the output of the encrypter.

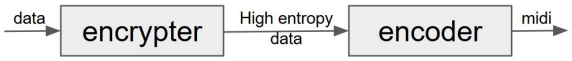


FIGURE 3. System scheme

To go from a text message to a bit string that is needed for the encrypter, we can use Base32. In fact, we can use any input format (image, sound, documents) and process it as a bit string, divided into parts, which the system is able to process.

3.1. *Crypto-module.* The task of the the encrypter is to go from an input data bit sequence to a random-like bit sequence with the use of a cryptographic key. Randomness of the output is needed for hiding the meaning of the message, as well as for the music encoder. We have chosen an AES cipher, but we can use any symmetric cipher. Our AES block size and key size is 128 bits, with electronic codebook mode (ECB). The input is padded.

3.2. *Stegano-module.* An input to the encoder is a random-like bit sequence. The task is how to go from this bit string to meaningful music (output in MIDI format). There is no easy method to measure the meaningfulness of the output music. The challenge to compose music that will "sound good" for the listener is not easy, and in some aspect, it is similar to the Turing test. We wanted the output from our music encoder to be indistinguishable from normal music, so that the listener would not be able to tell our deterministic composition from arbitrary simple human music composition.

3.3. *Formalization.* Let $T_1 = \{0,1\}^x$ be a plaintext, where $x$ is a number of bits that we are able to process. Similarly, let $T_2 = \{0,1\}^x$ be a ciphertext. $x$ can vary for each run of the algorithm, but for one run (for one song), it is fixed. For our blockcipher, AES, it is 128 bits. Then:

- $s_1$ is bijective mapping $s_1(k,...) : T_1 \to T_2$. The crypto-module (encrypter) with function $s_1$ is denoted $S_1$.
- $s_2$ is inverse function of $s_1$, $s_2(k,...) : T_2 \to T_1$. The inverse crypto-module (decrypter) with function $s_2$ is denoted by $S_2$.

Let $H = \{h_1, h_2, \dots h_n\}$ be a set formed by elements that represent a certain musical concept, note, rhythmic unit, interval, etc. Then:

- $c_1$ is injective mapping $c_1(...) : T_2 \to H$. The stegano-module (encoder) with function $c_1$ is denoted by $C_1$.
- $c_2$ is an inverse function of $c_1$, $c_2(...) : H \to T_2$. The inverse stegano-module (decoder) with a function $c_2$ is denoted by $C_2$.

$$plaintext \to S_1^{key} \to C_1 \to music$$
$$music \to C_2 \to S_2^{key} \to plaintext$$

We define specific elements in the set $H$ in the following sections.

## 4. Composing music from a bit string

The process of composing music based on bit string is not trivial. We need to understand how to create musical composition that is acceptable to listener's ear, but to find a way of storing as much information in the song as we can. We chose a simple song structure, in order to be able to create a deterministic algorithm to compose musical section.

4.1. *Song structure.* In music, a number of musical genres have evolved. Music composition belonging to a particular genre have specific characteristics, so there are various forms and structures of compositions. In this article, we focus on a simple modern musical composition or song that consists of sections:

- intro - I
- outro - O
- verse - V
- chorus - R
- bridge - B

The common length of one section is 32 beats. Each section is distinguished by its motif or theme, which it expresses. The task of **intro** section is to capture the attention of the listener, and to set the tempo, the rhythm and the general motif of the song. The intro may be slower than the rest of the song. The **verse** is usually a space for development of the theme of the song, lyrically the author describes the details of the song, the story, the events or

emotions, etc. The **chorus** usually creates a melodic, rhythmic or harmonic contrast to the verse. The task of the **bridge** is to break the repetitive pattern of the song and keep the listener's attention. **Outro** is a way to end a song. It can gradually slow down, turn down the volume of the music. For simplicity, we view these sections as the output of the same algorithm, with differences depending directly on the input. We use traditional layout of these sections:

$$I - V - R - V - R - B - R - O$$

We describe the system where the encoder $C_1$ is producing one section of the song. Within the whole song, we can store approximately 5 times as much information as in one section. We decided that one section (period) is 8 bars (two phrases).

4.2. *Musical requirements.* We define musical requirements in order to have a guidance on how to compose music, and which areas to focus on in the process, so that the music output is as pleasant as possible to the listeners, and, if possible, indistinguishable from the human-made music. In terms of the main aspects of music, we have further divided the musical requirements into melodic, harmonic and rhythmic.

- **Melodic requirements.** The melody played throughout the song will be in the same musical key. The musical key can be either selected for special purpose or generated based on the input to store more information. Because we want our system to be as simple as possible, the scales used in our composition are **major** or **minor** (which is the same as major, but we start the scale from tone VI). Those are the most frequently used scales in modern music. The elements of cadence and repetition of some melodic sections should appear in the melody. The melodic movement should not have many consecutive disjointed notes that would cause stress in the song. The range of tones in the melody should not use pitch tones that are too high or too low.
- **Harmonic requirements.** The song will follow a selected progression of the chords. We should also compose a simple bass line that will support the harmony. Also, the bass line will determine which chord is currently played from the chord progression. The melody should be in harmony with the chords, more specifically, the first tones in the first and third bar of each measure should be chord tonic.
- **Rhythmic requirements.** The measures in the periods will be divided into the same number of beats. Our period consist of 8 measures, each with 4 beats. We later specify the set of rhythmic units we use in composition.

These requirements are based on our experience with music, and they may be limited by our understanding of the topic.

## 5. Music composition algorithm

The following sections describe our solution. In general, the processes in $C_1$ and $C_2$ transform the input $I$ formed by the elements of $T_2$ into the output $O$ formed by the elements of $H$ and vice versa. When the $n$ bits are "removed" from the input $I$, and used for a specific operation, we write this as $I(n)$.

In each section, we define the principles of composition corresponding to the focus of the section. We define an inverse for each step. We also mention alternative solutions and the reasons why we did not to choose them. We formally visualise the algorithm with finite machine inspired by Mealy's automaton, and uml diagrams. There is some notation needed to describe the processes that can be seen in Figure 4

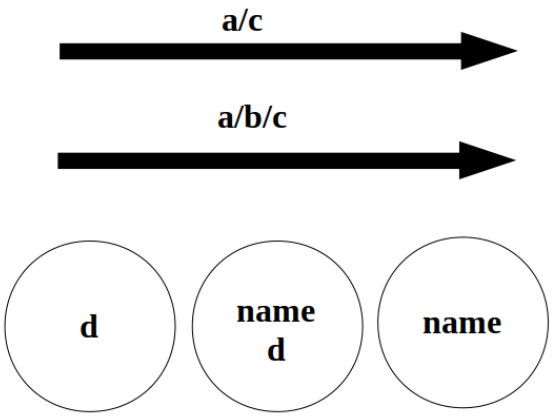


Figure 4. Notation

- a: the input $I$ from the elements in $T_2$
- b: the process performed in this transition
- c: the output $O$ from the elements in $H$
- name: the name of the state
- d: the process performed in the state
- ‰ the transition or the state with nothing to perform

The composition process is designed so that we can store a certain number of data into the range of one period. Due to this, there are some advantages, but also some limitations. First, the key and the chords are chosen, then the rhythm for the melody, and finally the pitches are added to the rhythm.

5.1. *Harmony.*

5.1.1. ***Generating of the musical key.*** The first step in the module $C_1$ is to generate musical key, whose tones will be used in melody of the period. We can generate 8 different musical keys, based entirely on the input. We convert $I(3)$ bits from the input to its decimal form. The decimal number representing musical key is also identical with tonic note that determines which major scale we will use. Therefore, for example, if we have the number 0, the scale will be C major. We can see that in the Figure 5. If we have determined which major scale we use, we can generate the set of tones in the corresponding musical key. This set of tones will then serve us as a space for melodic movement.

The algorithm we use for the calculation of this set is the following: Let $N = \{0, 1, ...127\}$ be the set of all possible notes in 12-tones musical system, represented numerically, and $t$ is the tonic of the major scale. Then the set $M \in N$ is a musical key defined by algorithm - Listing 1.

LISTING 1. Musical key set

```
1 m[0]=t
2 for i=1 .. 60
3     if(i mod 7 == 2 or i mod 7 == 6 )
4         m[i]=m[i-1]+1
5     else
6         m[i]=m[i-1]+2
```

This algorithm can be found labeled as $m(t)$ in the diagram 5. Based on our experiments, we determined that 60 tones will be sufficient for our purpose of composing music in the most common pitches.

5.1.2. ***Generating of frame notes.*** For a musical piece to have an aesthetic impression, composers use tones with special emotional effect in specific places. Notes with cadence, usually tonic notes, are used at the end of a musical sentence. They serve as a tension release. Other notes such as mediant, dominant or subtonic are commonly used in the middle of the sentence, with the effect of tension building. We took this information into consideration, and based on the input, we were able to encode two bits $I(2)$ into the note in the middle and one bit into the note at the end. For these notes, we used the term **frame notes**. In the reverse process of decomposing music, we were able to easily extract the data from frame notes, simply by just remembering their position in the period. For the last note in the period we always use tonic note, and therefore, based on this note, we were able to determine the musical key in decomposition process as well.

5.1.3. ***Generating of chord progressions and chord notes.*** Many modern songs use four chords. They are repeated throughout the song in an unique order. We used this composition model in our work. First, we

chose four chords for the whole period. Then we generated each note, which is included in every chord. Finally, we defined the chord progression as a permutation on a set of four. So the chosen chords were I V VI IV (0,1,2,3) for our major scale. For each chord, we generated four different chord notes, which can be later used in melody (four was chosen for its property to store two bits of information). The first three notes were the notes of a triad and the last fourth was a root note, one octave higher than the first root in triad. The chord progression was chosen based on four bits of information, which gave us 16 different possible options. We arranged these options lexicographically. So if the option was 0, the progression was (0, 1, 2 ,3), if it was 1, then the progression was (0, 1, 3, 2) and so on. We used chord notes not only in the melody, but also in the second voice of the base line. It serves two purposes: one, the aesthetic impression of the song is better; and two, we can extract the chord progression from it. That way we can get the four bits of information back in the decomposition process. The scheme can be seen in Figure 5.

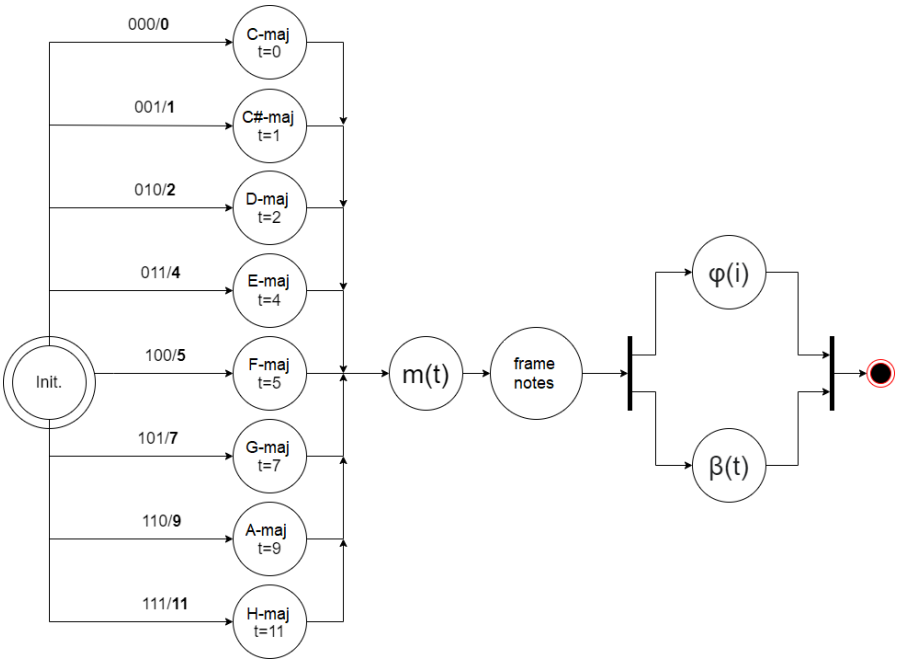


FIGURE 5. The harmony automaton

5.2. *Rhythm.* When it comes to the rhythmic layout of the song, we use a 32 beat form. Therefore, the period is divided into 8 measures, and each measure into 4 beats. Each beat has a length of a quarter note, so that the

time signature of the period is four-four time signature. Each particular beat can be also divided into group of notes with different length, whose sum of lengths is equal to a quarter. This fact gave us the option to create rhythmic groups of notes which can fill the length of one beat. Therefore, the decision, which of these pre-declared rhythmic groups we use, is determined by the binary input. Based on this fact, we decided that the number of possible rhythmic groups is 8, so we are able to store $I(3)$ bits of information into the decision of which group we use in each particular beat. We observed that if we choose random rhythmic group for every single beat, the aesthetic impression will not be as pleasant as we need. We had to incorporate some sort of rhythmic pulsation, which would give us the possibility to repeat the rhythm, but also store information in the decision based on the input. In order to satisfy these conditions, we chose only 12 beats instead of 32 to be filled with rhythmic groups generated based on the input. Then we dispersed those 12 beats into the whole period. The other 20 beats were filled with copies of those 12, based on the rules we chose according to the experiments. We can see the layout of the beats in table 1, where beat represents a specific location in period, and c represents uniqueness of the beat. We can see that we specified 11 beats for storing information and the rest of the beats are deterministic.

TABLE 1. Layout of the beats in the period

| Period | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Phrase 1 | | | | | | | | | | | | | | | |
| beat | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| c | "q" | 0 | 1 | 0 | 2 | 3 | 4 | 5 | "q" | 0 | 1 | 0 | 6 | 7 | 8 | "q" |
| Phrase 2 | | | | | | | | | | | | | | | | |
| beat | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| c | 9 | 10 | 9 | 11 | 2 | 3 | 4 | 5 | 9 | 10 | 9 | 11 | 6 | 7 | 8 | "q" |

There should be some repetition within each measure and within the whole period, and some measures should be rhythmically repeated. We were not able to store as many bits as when we were using all 32 beats, but the quality of music is considerably better. We now have set of rhythmic notations for every note in the period, which we denote by $R$. When it comes to the decomposition process, first we had to take the rhythmic notation of each note. Then we combined them together into groups chronologically, so that the sum of all lengths of notes within the group is equal to a quarter note (hence the period is four-four time). Finally, we remembered each position of the 12 beats chosen based on the input, so simply by comparing all possible rhythmic groups with the group found in the current beat, we were able to

extract the information encoded into that decision. Scheme can be seen in Figure 6.

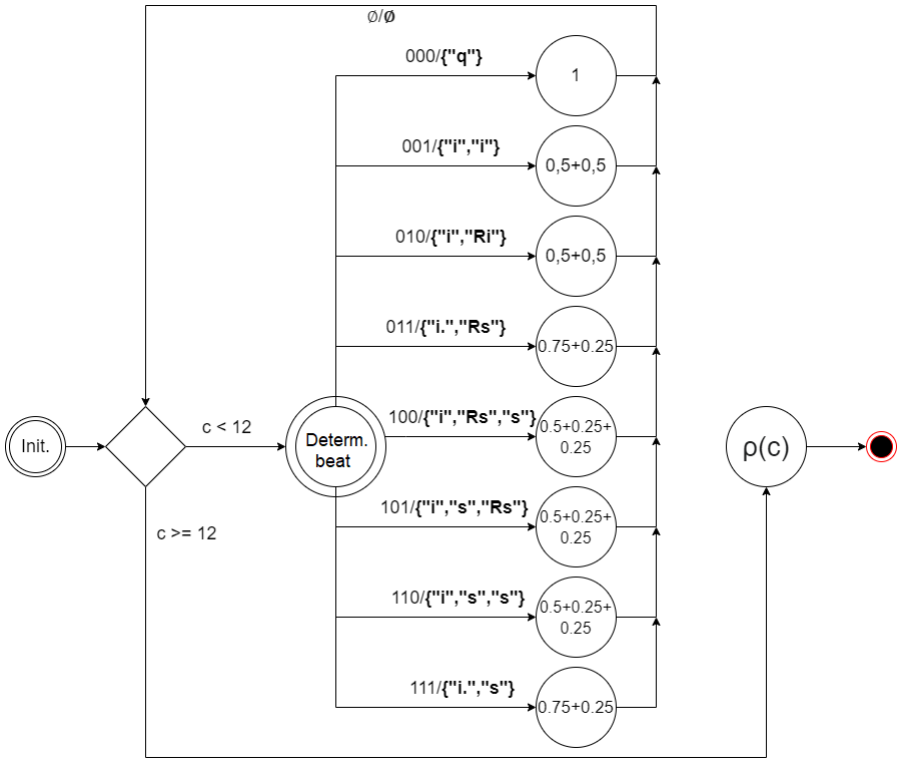


FIGURE 6. The rhythm automaton

5.3. *Melody.* In this section, we explain how we were able to create melody that sounds good and at the same time stores more information than in the rhythm or the harmony together. Let us summarize what we have generated up to this point. We have a basic harmonic frame of the song, which tells us which notes will be preferred in some measures, and which not. Then, we have the rhythm, which contains the length value for every note in the period. Now, we add a pitch value to the notes and the music composition is complete. For better illustration, we can imagine the melody as a dependency diagram of melodic pitch versus rhythmic length. Now we can define some variables, which will help us move in the span of the whole period. First, *beatCounter* is a numerical value that tells us, in which particular beat is the song situated at the moment. Then, *noteCounter* tells us the note within the beat. Value

$x$ tells us the pitch of last played note. We divided melody composition into 3 different algorithms, which are used in a different sections of the song.

- Melodic automaton. This algorithm is able to transform the largest number of bits out of all composition algorithms. Its representation can be seen in Figure 7. It begins in the state Init, and if the variable beatCounter reaches value $z$, the algorithm terminates. The value of $z$ is determined by the choice of the length of the period, in our case it is 8 measures, of 4 beats. First decision determines whether we are situated in a position where a chord note should be used. From music theory, we know that it should be the first note in the first and the third beat of every measure. Then we stored $I(2)$ bits into the decision, which exact note will be used from all the possible chord notes for the chord played in that measure. This also indicate the value of $x$ in the first beat. The musical pitch is written in the output with the proper duration from the set $R$. In other places, where the chord note is not used, we use **melodic motion** based on the previous note. First, we check if our current value of $x$ is above or under the pleasant sound interval. Then, according to this, the movement of the next note is either ascending or descending. If the current value $x$ is in a pleasant sound interval, the orientation of its melodic movement will be determined based on the input. So if $I(1)$ bit is 0, the melody will be descending, and if it is 1, ascending. Then this algorithm determines, based on the input, whether the change in the melodic movement will be a step or a leap.
  1. A step is chosen if $I(1)$ value is 0. If the melodic movement was predetermined to be descending, a value $x-1$ will be added into the melody, with length notation taken from set R. If it is ascending, then the value will be $x+1$.
  2. A leap is chosen if $I(1)$ value is 1. Afterwards, a choice of the interval of the leap is made. Since the leap indicates a change in the melodic movement greater than 1, we chose its possible range from 2 to 5. Based on $I(2)$, we were able to determine which value $K$ will the leap acquire. After that, based on whether the melody is ascending or descending, we added a note with a value $x+k$ or $x-k$ into the melody, with its proper length notation from R.

  After each of these steps, the algorithm returns into the Init state. This algorithm can be represented with Mealy state machine. Therefore, the decomposition of music can also be automated with a reverse model of this machine (Figure 7) The beatCounter and the noteCounter are inputs to this automaton that increment out of the scope of the automaton, with every beat (or note) that our system writes.

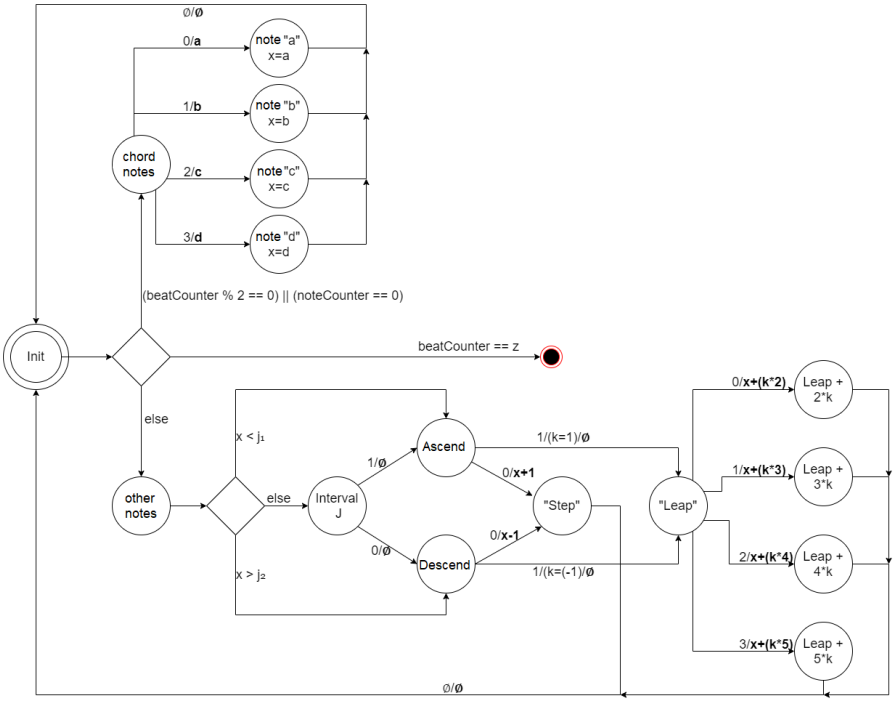


FIGURE 7. The melody automaton

- In accordance with the research, we decided that the melodic content of some measures should be repeated in others, as well. We have to take into account that in those measures, a different chord is probably played. In order to ensure that the harmony will be correct, we shifted the value of pitch for each note up or down to be compliant with the new chord. For example, if in the first measure A, the root note of chord C major triad is played, then in the new measure B, which is in G major triad, the root note will also be played. We can store $I(1)$ bit of information into the decision, whether the shift will be to a note with a higher pitch or a lower pitch. In the decomposition process, it is easy to check, whether the copied note is higher or lower than the original, and from that, we are able to get the information back.
- As we have already mentioned before, some notes were labeled as the frame notes. We use one at the end of the first phrase and one at the end of the period. Within the decision which note in the set of middle notes we use are stored $I(2)$ bits of information. And as a last note, a tonic note in the fifth or the sixth octave can be used, so $I(1)$ bit can be

stored into that decision. There can be a situation, where the previous note $x$ is melodically too far away from the predetermined frame notes. Going from this note to the frame note can sound very stressful and suspicious. We solved this problem by using a beat previous to that with a frame note as an approximation beat. Its sole purpose is to lower the distance from note $x$ to a frame note using steps. Reverse decomposition process then ignores this beat, since no information is stored within. It only checks the important middle and last beat, and by comparing with all possible notes in that place, it is able to decode the 3 bits that are stored there.

## 6. Implementation

This system is implemented into a simple Java application, where the user is able to transform a message or a data set into a short song with the use of a unique cipher key. The user can also decompose a song, decipher and extract the message. We use JFugue[7] library to work with music in Java. The code is available at Github [12].

## 7. Survey

The goal of our stegosystem is to create music, which is pleasant to the ear of the listener, or even unrecognizable from mainstream music. We conducted a survey on general public (86 respondents) to determine whether this goal had been fulfilled.

The main aim of this survey was to find out whether a respondent is able to distinguish music created by our stegosystem from man-made musical pieces. The secondary goal was to find out, what the best tempo is and what musical instrument should be used, so that the previous decision is even harder for the listener.

The conducted survey consisted of a recording with 8 musical pieces. Three of those were composed by our system. These are the specifications of each song:

1. Piece was played with an organ, it contained a baseline and the tempo was 90 bpm (beats per minute).
2. Piece was played on a harp, it did not contain baseline and the tempo was 120 bpm.
3. Piece was played on a piano, it did not contain baseline and the tempo was 90 bpm.

As we can see, one of these songs was voted to be man-made by a majority of the respondents (Figure 8). But approximately one third of the respondents were not able to recognize the remaining two machine-made pieces.

The other 5 recordings in the survey were excerpts from different musical pieces ranging from online musical theory lessons to classical music. The

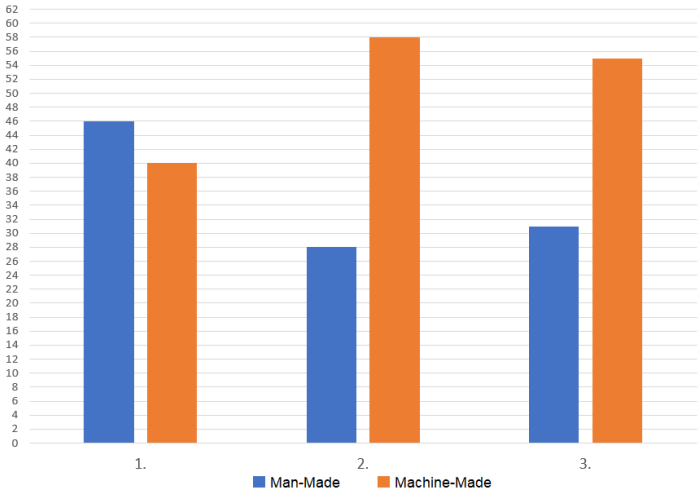


FIGURE 8. Our songs

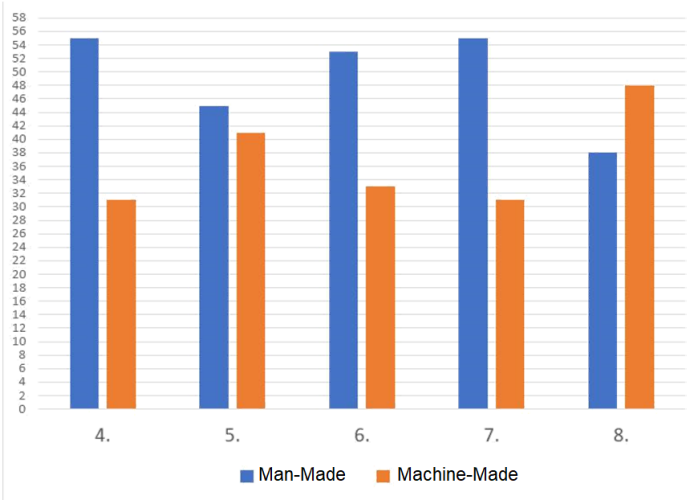


FIGURE 9. Human-made songs

length of these excerpts was identical to the length of our ciphered music. These are the names of the pieces:

1. Jamie Henke - How to Write a Melody?
2. Grant Kirkhope, Eveline Novakovic - Fungi forest daytime

3. Atsushi Chikuma, Tomoyuki Hamada - Level ok
4. Dmitri Shostakovich - Fugue in A major
5. Igor F. Stravinsky - Rite of Spring

We can see from Figure 9, Rite of Spring was voted to be machine made by majority of respondents. Respondents knew there are some songs that are machine-made, therefore we presume they picked the ones which they found unusual.

At the end, we asked one additional question: whether the questions were answered with or without certainty (Figure 10).

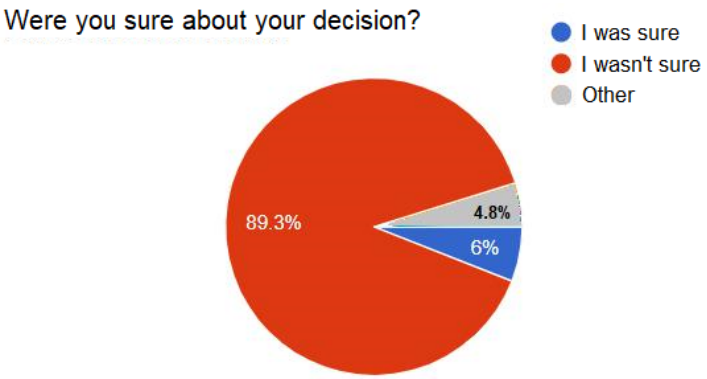


FIGURE 10. How certain were respondents about the answers

As we can see, the respondents were not sure which musical pieces were produced by our system.

## 8. CONCLUSIONS

Our input appears to contain some elements of randomness, as it is a result of AES encrypting. Music entirely generated from this input using only melodic automaton is unpredictable and usually unaesthetic. This flaw was solved by adding the frame notes, repeating measures or rhythmic groups. However, the negative consequence of these solutions is that the possible input that can be encoded in the period is lower. As a result, two rivalry factors arise and influence our overall solution: the aspect of data range and the aspect of aesthetics of music.

The ideal solution to this problem lies in our ability to find an equilibrium between these two factors. We should be able to ensure a good aesthetic quality of the music, but a useful data range at the same time. Most of the algorithms of our stegosystem work with a stable input, the exception is the melodic automaton. Due to this exception, the data range of one period differs

in every song. Below, you can see the minimum, maximum and average data range we were able to get during our experiments.

- Minimum:
  - Stable part: 66 bits
  - Unstable part: 10 bits
  - Sum: 76 bits
- Maximum:
  - Stable part: 66 bits
  - Unstable part: 150 bits
  - Sum: 216 bits
- Average:
  - 90-130 bits

It needs to be mentioned that from the point of view of music composition, periods generated by our system are the most similar to the musical verse. If we wanted to generate a chorus, an intro, an outro or a bridge, the composition module would need to be slightly modified. This would influence the data range of these periods, as well.

The survey tells us that our research has potential, as we were able to hide the information into music, and our respondents were not sure which songs were machine-made, and therefore contained hidden meaning. There is a space for improvements: how to make the music sound better, or how to store more information in the song. For future research, we are going to experiment with different genres of music, and more complex musical structures.

Acknowledgements.
This work is supported by project VEGA 1/0159/17.

## References

[1] A. Droppová, Elementárna hudobná teória, Prešovská univerzita, Prešov, 1998.
[2] O. Grošek, M. Vojvoda and P. Zajac, Klasické šifry, Vydavateľsvo STU, Bratislava, 2007.
[3] M. Gulášová and M. Jókay, *Steganalysis of stegostorage system*, Tatra Mt. Math. Publ. **64** (2015), 205–215.
[4] J. Henke, Basic Concepts of Music Theory, iTunes U, University of Wisconsin-Madison, 2011.
[5] T. Judd, *Musical Cryptograms: Five Scores that Contain Hidden Messages*, The Listeners' Club, 2019, `https://thelistenersclub.com/2019/05/03/musical-cryptograms-five-scores-that-contain-hidden-messages/`.
[6] A. Kerckhoffs, *La cryptographie militaire* Journal des sciences militaires, 5–83, 161–191, Jan, Feb, 1883.
[7] D. Koelle, *Music Programming for Java™ and JVM Languages*, 2020, `jfugue.org`.
[8] J. Maysh, *THE CODE: A declassified and unbelievable hostage rescue story*, The Verge, 2015, `https://www.theverge.com/2015/1/7/7483235/the-code-colombian-army-morsecode-hostages`.
[9] J. Pospíšil, Hudobná teória pre konzervatóriá, Slovenské pedagogické nakladateľstvo, Bratislava, 1985.

[10] C. Schmidt-Jones, The Basic Elements of Music, Rice University, 2014.
[11] G. Schott, Schola steganographica: in classes octo distributa, Jobus Hertz, printer, 1680.
[12] P. Sobota, *Musipher code*, `github.com/xsobotap/Music-cipher`.

## Musipher: Skrivanje informacija u glazbenoj skladbi

*Peter Špaček i Pavol Sobota*

Sažetak. U ovom članku predstavljamo novi način skrivanja informacija. Informacije pohranjujemo izravno u procesu skladanja glazbe, na temelju glazbene teorije. Stvorili smo algoritam za proizvodnju glazbe na binarnom nizu, gdje se svaki bit pretvara u odluku o glazbenoj skladbi. Slijedimo jednostavna pravila za stvaranje glazbe koja zvuči dobro. Proveli smo anketu kako bi utvrdili funkcionira li naše rješenje, i rezultati našeg pristupa su se pokazali obećavajućim.

Peter Špaček
Slovak University of Technology in Bratislava, Slovakia
*E-mail*: `peter.spacek@stuba.sk`

Pavol Sobota
Slovak University of Technology in Bratislava, Slovakia
*E-mail*: `xsobotap@stuba.sk`

*Received*: `14.11.2020.`
*Revised*: `12.3.2021.`
*Accepted*: `13.4.2021.`