UDC 004.056.54:519.7
DOI: https://doi.org/10.17721/ISTS.2025.9.54-60

Yevhenii PONOMARENKO, PhD Student
ORCID ID: 0009-0000-2647-3381
e-mail: allagrir@gmail.com
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

Oleksandr LAPTIEV, DSc (Engin.), Assoc. Prof., Senior Researcher
ORCID ID: 0000-0002-4194-402X
e-mail: olaptiev@knu.ua
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

MATHEMATICAL MODEL OF STEGANOGRAPHY USING SUBOPTIMAL DECISIONS IN DATA COMPRESSION ALGORITHMS

Background. *Protecting access to information in the digital age requires the development of digital tools to encrypt and hide the information from everyone who is not meant to be able to access it. While encryption is great at preventing unauthorized people from accessing the information, it lets people know that there is something hidden behind the transformation. Steganography instead allows us to obscure the fact of information concealment in the first place. Unfortunately, common types of steganography rely on altering the source signal, leaving subtle footprints of data manipulation that can be traced and detected. The goal of this research paper is to provide a model that has the potential of preserving the source signal in its entirety, instead relying on changing the way the source signal is represented in digital media in a way that allows us to encode secret data into the output stream.*

Methods. *A theoretical analysis of steganography approaches on data compression algorithms was conducted. Methods of preserving source data stream were investigated.*

Results. *A new model has been developed that uses decision-making processes in data compression algorithms to encode steganographic data in a resulting data stream. In lossless data compression schemes, it is possible to achieve perfect reproduction of source data stream, making detection through analysis of underlying signal encoded in data compression algorithms useless.*

Conclusions. *The need for information security has been increasing over time, with tensions between countries resulting in new armed conflicts around the world. The ability to embed and covertly send data through steganography may provide a competitive economic, political, and/or military edge. The developed model can be applied to further develop specific methods of steganographic data encoding that is resilient to analysis and detection by existing approaches that rely on statistical analysis of underlying signal stream.*

Keywords: *steganography, data compression algorithms, signal processing, data security, decision trees, entropy encoding, arithmetic encoding.*

Background

With the advent of digital computers, a need to store information about the real world in digital form emerged. This information takes a lengthy and complex route in order to turn from physical phenomena of a light bouncing off of the surface of a physical object or pressure waves travelling through the air into a string of numbers, zeroes and ones, that can be stored in computer's memory, and then it needs additional work to be turned from those back into physical light, sound, or other form of real-life, analog phenomena – or even objects. Photosensitive diodes turn photons of light into electrical signals that we can register and process, allowing us to take a photograph of the real world. A transducer turns pressure waves into digital signals, allowing us to record sound.

Of course, without a way to represent this information in a digital, binary form, there is no way we can store this information on computer systems. For photographic signals, we rasterize visual data, storing color information in discrete, individual cells called pixels. This means that we quantize the infinite precision of the real world in two ways: spacial and spectral. Rasterized pixels would be the spacial quantization, while the color data of those pixels – being limited to a finite number of bits we can use to store color information in the pixel – would be spectral. For sound, the quantization would be temporal and amplitude: the former splits the infinitely precise stretches of time into distinct (and usually equal) durations of time, during which a single intensity of the sound signal is assumed. This signal intensity, just like with pictures, could only be encoded with a limited number of bits, leading to amplitude quantization. Video signals, therefore, are usually nothing more than a sequence of pictures presented at equal intervals of time between one another, synchronized with a sound signal coming from one or more sources in physical space. There is a temporal quantization happening between frames, with values of dozens of frames per second being common.

All these encoding solutions are made with decoding and reproduction back into physical phenomena in mind. A digital encoding of a photo has no use if we cannot present it back into a grid of colors that we can perceive in the real world with our human eyes. A sound recording is useless if we cannot listen back to it. The process of both capturing analog signals and capturing them into digital form, and reproducing these digital signals back into physical realm is imperfect, since the components we use to both transform light, sound, and other sensations into electrical signals, and perform the inverse of that, are subject to distortions, noise, lack of performance, bias, and other flaws – though the quality of devices to both capture and reproduce those signals has been improving since their inception, and continues to this day (Galal, 2016; Rumsey, & Mccormick, 2013, pp. 201–256).

Storing information in digital form, however, poses a challenge of providing enough storage capacity for a given media. Storing color data of a single pixel in a 24-bit image takes up three bytes of storage. Multiply that by multiple megapixels that a commonly used 1080p display would use, and you arrive at approximately six megabytes of raw picture data. Videos require thirty of these frames per second, which quickly turns into a gigabyte of storage capacity being taken up by 5–6 seconds of raw video stream. This explosion of storage capacity that is necessary for storing modern pieces of media means that we must somehow reduce the amount of data our media object requires to store in digital form. In other words, we need to compress digital data.

Modern schemes of digital compression can reach file size reduction of dozens of times, if not more (Barina, 2021; Öztürk, & Mesut, 2021, pp. 15–20) while preserving most (if not all) of the detail of the original, uncompressed

ISSN 2707-1758

data stream. To achieve high compression ratios, compression algorithms adapt their approach for various parts of the source media. For example, when compressing sound, an algorithm might dedicate more bandwidth to a more intense part of a song with lots of different instruments all playing at once, and less bandwidth to sections with silence or quiet sounds. In other words, they make decisions based on the source data stream. Encoding schemes have a limited number of ways to decide how to compress a certain primitive structure of the data stream, like a block of pixels or a set of audio samples. Commonly, they tend to select the best ways of encoding the data that they expect would take the least number of bits. In the end, these compression algorithms produce a smaller, more compact representation of source media, potentially with some losses in perceptual fidelity that the algorithm authors considered acceptable for a given compression ratio. Importantly, these compressed representations can be decompressed and turned back into a raw stream of data that can be presented to the user in the form of a picture, sound, or video.

With established methods of representing – and compressing – analog media in digital formats, we can shift our focus to another need: the need to covertly transmit information. There are two ways we can achieve this: encryption and steganography. Encryption is the process of converting readable information into unreadable information to prevent unauthorized access. Steganography is the process of embedding secret data within another object in a way that conceals the presence of secret data from an unsuspecting observer. Encryption is great at preventing people from accessing information they should not have access to, but encrypted data on its own is obvious, anyone can see there is something there, painting a target on it. After all, why would somebody encrypt something that is not worth protecting? Steganography, on the other hand, does not, on its own, prevent others from accessing the data, but instead attempts to hide it in plain sight, preventing people from knowing it is even there unless they know about it.

There are many different approaches to steganography, and they depend on the cover media used. Common types of cover media include pictures, video, audio, and text (Anas, Ridzuan, & Pitchay, 2025). Broadly, these techniques tend to either modify the source data directly (i.e., manipulating pixel color or sound intensity values by changing their least significant bits (LSB)) or indirectly through manipulating values in the frequency domain (Apau et al., 2024). This leads to a material change to the underlying representation of media; in other words, after transformation, the raw data that is transmitted to the screen, played back through the speakers, or printed out on a sheet of paper, is different from the original data of the cover media. Though there exist adaptive techniques that utilize machine learning, artificial intelligence, blockchains, and other tools to prevent existing steganalysis and statistical analysis techniques from being able to detect steganography (Apau et al., 2024, p. 5), they still change the apparent representation of carrier media. This leaves a gap in current research, as it is difficult to find a steganography method that would leave the apparent representation unchanged.

The objective of this research, therefore, is to develop a new model of steganographic encoding of data that does not change the apparent representation of some, if not all, media objects.

Methods

In this article, we performed theoretical analysis and systematization of data compression algorithms and ways of exploiting their decision-making processes to encode a secondary, secret message within the encoded cover media such that apparent result is not changed. Methods of preserving the apparent representation of carrier media were investigated.

Results

There are many ways to represent the same data. Data compression algorithms exploit this idea, attempting to find a specific representation of a given set of data that takes up as little storage space as possible. Of course, there are many trade-offs: processing difficulty (how long does it take to compress/decompress a file?), compression ratio (how much smaller the file has become after compression compared to the original output?), legal constraints (are there any patents or licensing issues that prevent us from being able to use this algorithm?), memory requirements (how much RAM do we need to process this algorithm?), and many more. There are other considerations that influence data compression algorithm design. For example, are there any special properties for the underlying data we can exploit to achieve better efficiency? With time, new data structures are discovered, new mathematical properties are found, and new ways of representing the same set of data are found, resulting in innovation in data compression algorithms. One thing that is worth keeping in mind, however, is that even within the constraints of a given data compression/decompression format, there are many ways of representing that data. Compression programs try to find the best one among many, but there are other ways to compress the data that will allow the decompressor to decode that representation back into the same original raw data.

What if we manipulated the way compression algorithms worked to produce another, equally valid string of bits that would decode into the same picture, sound, video, text, or anything else? Better yet, what if we did so in a way that allowed us to encode additional information, allowing us to embed secret information into the compressed result that we could then transmit without any changes to the source media? That is the goal of this research.

The overall flow of many compression algorithms takes three basic steps: take the source data object, define certain transformations on it, change the way it should presented through using a certain data structure; perform decisions as to how the source object should be converted into this new representation; and finally, convert it into a binary form that can be stored on a digital storage medium. For example, with pictures, instead of operating on raw pixel data, groups of pixels could be analyzed in an attempt to find larger structural blocks, information about which could be conveyed not through the raw pixel data, but with a relationship between a certain overall expectation and the difference between expectation and reality (W3C, 2025; International Organization for Standardization, 1994; Google, 2025). As could be seen in Fig. 1, on one of the compression stages in WebP image format, the compression algorithm performs various predictions on a blocks of pixels of fixed size and then chooses the one that has the best match with actual data (Google, 2025).

Though the original compression algorithm aims to produce an output that takes up the least amount of space, it does not have to. If we instead force the compression algorithm to take different, suboptimal decisions, we could encode additional data into the resulting compressed file.

ISSN 2707-1758

Fig. 2 shows a generalized view of proposed suboptimal compression decision hijacking steganography approach. For a given compression format, a set of decisions is outlined. During compression stage, a steganographic encoder performs the compression of data as normal until a decision point is reached. Then, instead of selecting an option that would lead to biggest reductions in file size, a different option may be selected, thus encoding a certain amount of secret data into the resulting file.

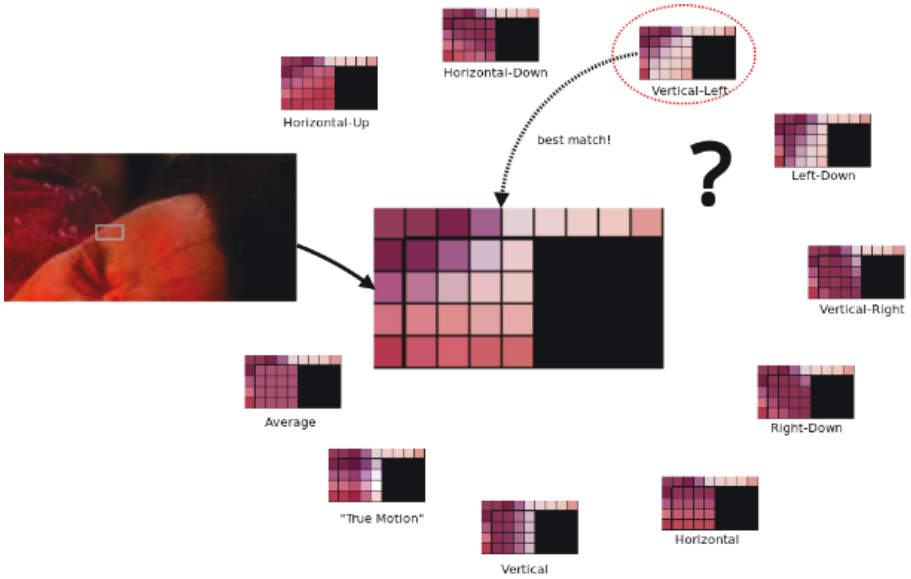

Fig. 1. Example of prediction modes in WebP

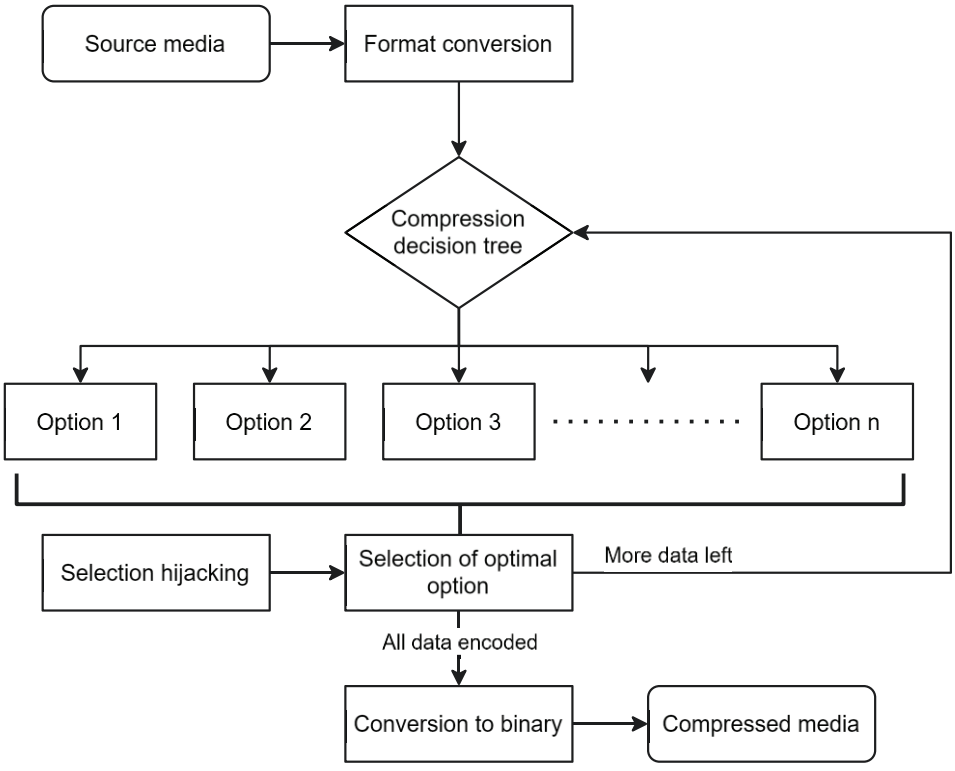

Fig. 2. Compression hijacking diagram

Selection hijacking engines (SHjEs) can be designed to adapt to specific needs. The most naive approach would be to hijack all decisions and utilize the entire decision space available to encode the cyphertext as quickly as possible. While it is a viable approach, it may result with significant distortions to the beginning of the source file, especially if compression is lossy. Additionally, the apparent dip in compression ratio between the start and the end of the file may allow for some steganographic analysis. A much better approach would be to spread the hijacked decision points around the entire length of the file. This would require either multiple compression passes to

ISSN 2707-1758

perform precisely, or a decent heuristic approach that would predict the amount of decision points and their capacity within a couple of percentage points from ground truth. Additionally, SHjE can limit its decision space to powers of two, allowing it to encode bytes directly instead of resorting to techniques like arithmetic coding (Witten, Neal, & Cleary, 1977).

Nonetheless, arithmetic coding, or other similar entropy encoding techniques, could be used in cases where maximal capacity for encoding secret data is necessary. What is especially useful in entropy encoding is the ability to use mixed-radix numeral systems, allowing us to minimize losses incurred by limiting ourselves to just binary codes. The reason it is possible to use mixed-radix numeral systems is because the compression algorithms tend to have a certain and well-defined set of options at specific decision points. In other words, the number of decisions at a specific decision point utilized to encode secret data stream defines corresponding radix in the entropy encoding solution. Fig. 3 provides an example visual illustration of steganographic encoding in action. At certain points in a compression algorithm, a decision with multiple options needs to be made. SHjE decides to use specific decision points to encode the data it needs. Because selection space is well-defined, the decoder would know the radix of this hijacked selection, allowing it to interpret it appropriately.

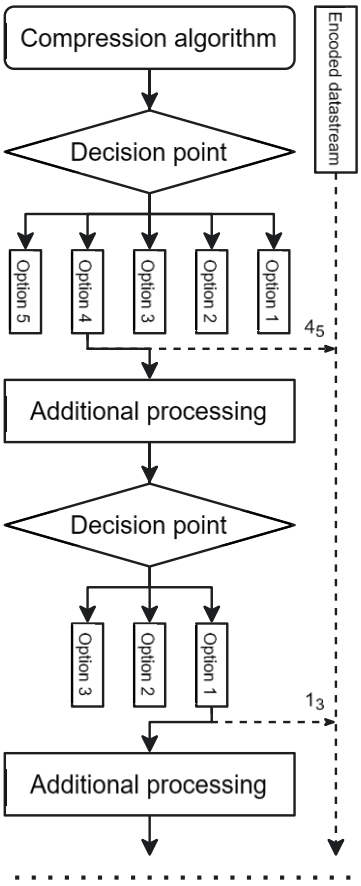

Fig. 3. Encoding of data through selection hijacking

Decoding secret data stream from a steganographic media boils down to performing the same steps in reverse: perform the decoding process; figure out the decision space that was available during the encoding step; and then derive a numerical value that corresponds to an option that was chosen during encoding.

There are two main components of this new model: SHjE itself and an embedding engine that integrates it into a given compression algorithm. While the same SHjE could be used within different data formats and compression algorithms, it requires modifying them to provide hooks and interfaces for the SHjE to operate on. While specifics of these implementations are outside of the scope of this research article (and therefore avenues for further research), they will need to rely upon specific data format and compression algorithm details to define valid decision points that could be used to encode secret data.

Mathematical model. To begin expressing this model in mathematical terms, we need to introduce some definitions. Let us define the following:

- $R = (y_1, y_2, y_3, \dots, y_n)$ – raw media object of length n, where y_i are media primitives (i.e., pixel brightness, sound intensity at a given point in time, etc.)
- e_i – i-th encoding decision with capacity c_i.
- $\mathbb{E}$ – encoding decision space for a given data compression format.
- $E(R) = (e_1, e_2, e_3, \dots, e_n)$ – encoding decision chain for a given raw media object R, $\forall i \in \overline{1,n}: e_i \in \mathbb{E}, \forall E(R) \in \mathbb{E}^*$.
- $e_i^{k_i}$ – resolved encoding decision with resolution $k_i: 1 \le k_i \le c_i$.
- $C_E(R) = \left(e_1^{k_1}, e_2^{k_2}, e_3^{k_3}, \dots, e_m^{k_m}\right)$ – a compression of R, which is a tuple of resolved encoding decisions which represent the raw source object.
- $\hat{C}_{E,S}(R) = \left(e_1^{\hat{k}_1}, e_2^{\hat{k}_2}, e_3^{\hat{k}_3}, \dots, e_m^{\hat{k}_m}\right)$ – a hijacked compression of R that carries a secret message S.
- $D\left(C_E(R)\right) = \bar{R}$ – a decompression of $C_E(R)$ which retrieves a decompressed object $\bar{R}$.

A compression $C_E(R)$ can be represented as a sequence of bits $B\left(C_E(R)\right) = (b_1, b_2, b_3, \dots, b_l)$ of length $\left|B\left(C_E(R)\right)\right| = l$. An optimal compression function $C_{E,optimal}(R)$ derives such a sequence of encoding decisions e_i and their resolutions k_i that its length is as small as possible. In other words, optimal compression satisfies the condition:

$$\min\left|B\left(C_E(R)\right)\right| = \left|B\left(C_{E,optimal}(R)\right)\right|. \quad (1)$$

It is important to note, however, that for a given lossless compression (i.e., the one for which $D\left(C_E(R)\right) = R$ always holds true for any $C_E(R)$), there is more than one way to compress a given raw media object R. In other words, there are many compressions $C(R)$ that can be decompressed back to R. We can use a subset of encoding decisions $\hat{E}(R) \subseteq E(R) \in \mathbb{E}^*$ as a carrier for our secret message S. To do this, we can take each encoding decision $\hat{e}_i \in \hat{E}(R)$ and, instead of letting the compression algorithm resolve it, give it to a SHjE, which will use it to encode a part of a secret message.

To dig deeper into a SHjE, we first need to establish an arithmetic coding system. Let us assume we have the following:

- x_l – an l-digit mixed-radix number with a finite number of digits.
- d_i – an i-th digit with a radix $r_i \in \mathbb{N}$, $d_i \in \overline{0,(r_i - 1)}$.
- $\mathcal{D}(x_l) = (d_1, d_2, \dots, d_l)$ – a string of digits for an input number x_l.
- w_i^j – a weight assigned to the j-th value for i-th digit, $j \in \overline{1, r_i}$, $w_i^j \in [0,1]$.

ISSN 2707-1758

- $W_i = \{w_i^1, w_i^2, \dots, w_i^{r_i}\}$ – a set of weights assigned to each value an i-th digit could be, where:

$$\sum_{k=1}^{r_i} w_i^k = 1. \quad (1)$$

- $\mathcal{W}(x_l) = \mathcal{W}_{x_l} = \{W_1, W_2, \dots, W_l\}$ – a collection of sets of weights for an input number x_l.
- $A\big(\mathcal{D}(x_l), \mathcal{W}(x_l)\big) = [a, b)$ – an arithmetic coding of an input number x_l, $0 \le a < b < 1$.

The arithmetic coding system works by progressively dividing a given range $[a, b)$ into smaller regions $[a, b_1)$, $[b_1, b_2)$, …, $[b_{r_i-1}, b)$ (assuming $b_0 = a$, $b_{r_i} = b$), where the length of each region is proportional to the weights W_i:

$$b_n - b_{n-1} = (b - a)w_i^n. \quad (2)$$

With that in mind, an arithmetic coding of a number x_l could be defined recursively. Let $x_h = \lfloor x_l \rfloor_h$ be a number comprised of the first h digits of x_l, where $h \in \overline{1, l}$. Therefore:

$$A\big(\mathcal{D}(x_h), \mathcal{W}(x_h)\big) = [a_h, b_h),\ a_0 = 0, b_0 = 1. \quad (4)$$

$$a_h = a_{h-1} + (b_{h-1} - a_{h-1}) \sum_{i=0}^{d_h - 1} w_h^i,\ b_h = a_h + (b_{h-1} - a_{h-1}) w_h^{d_h}. \quad (5)$$

Equation (4) establishes the base case for recursion, and equation (5) defines the bounds a_h and b_h for a number x_h.

While this approach allows us to encode any number, we have no way of knowing how long that number is. To solve this problem, a stop digit can be introduced. This would increase the radix of every digit by one and require adding a new weight $w_i^{r_i+1} \in (0, 1)$. To satisfy the condition set by equation (1), the full set of weights W_i will need to be rebalanced for each digit to produce a new set of weights $\overline{W}_h$:

$$\forall i \in \overline{1, r_i}: \overline{w}_h^i = w_h^i \times \left(1 - w_h^{r_i+1}\right). \quad (6)$$

With that settled, a SHjE is a function that performs arithmetic coding through manipulating encoding resolutions $\hat{k}_i$ of encoding decisions $\hat{e}_i$ in a subset of an encoding decision chain for a given media object $\hat{E}$ provided by the embedding engine. More specifically, we use $\hat{e}_i$ as digits with corresponding radices c_i. The SHjE is also responsible for deciding upon a set of weights W_i for each digit. After that, SHjE can produce arithmetic coding T of a target value that will embed the secret data S into the hijacked compression $\hat{C}_{E,S}(R)$:

$$T = A\big(\hat{C}_{E,S}(R), \mathcal{W}_T\big) = [a_T, b_T) \in [a_S, b_S) = A\big(\mathcal{D}(S), \mathcal{W}(S)\big), \quad (7)$$

where $\mathcal{W}_T$ – weight set collection for a hijacked compression $\hat{C}_{E,S}(R)$ defined internally as one of the parameters of a SHjE, $A\big(\mathcal{D}(S), \mathcal{W}(S)\big)$ – arithmetic coding of secret data S.

As such, the hijacked compression $\hat{C}_{E,S}(R)$ is, in some sense, a byproduct of an attempt to arithmetically code our secret value using an unconventional mixed radix numbering system that consists of encoding decisions of a compression algorithm.

Note that it is not necessary to have a stop digit in the SHjE, since as soon as it encodes a value that is fully inside one of the stop digits of arithmetically coded secret data S, it can stop its processing.

Decoding the secret value S encoded in a hijacked compression $\hat{C}_{E,S}(R)$ is therefore a simple task of letting the embedding engine define the same subset of an encoding decision chain $\hat{E}$, providing the resolved encoded decisions $e_i^{k_i}$ to the SHjE, and then using them to recreate the value of T, and from that the value of S, assuming the radices of $\mathcal{D}(S)$ and the weight set collection $\mathcal{W}(S)$ is known ahead of time (either by being agreed upon by the parties using this model to exchange steganographic information, or being a part of the SHjE configuration).

To summarize, the selection hijacking engine has the following parameters:

- $\mathcal{W}_T$ – weight set collection used for creating a target arithmetic coding.
- $\mathcal{W}(S)$ – weight set collection for secret data S.
- $r_{i,S}$ – radices of digits of secret data S.
- Whereas the embedding engine has the following parameters:
- Mapping $E(R) \to \hat{E}(R)$.
- r_i – radices of digits of a hijacked encoding, $1 \le r_i \le c_i$.
- Ordering of resolutions $\hat{k}_i$ for all encoding decisions $\hat{e}_i$.

As such, the embedding engine may decide to artificially restrict decision space for the SHjE in cases where taking a particular decision may result in undesirable side effects like particularly large file size increases. Additionally, it is responsible for mapping the numeric representation of resolutions $\hat{k}_i$ into actual decisions an encoder it embeds itself into needs to take.

Comparison with existing approaches. Definitions of this research article could be used to describe most of the existing steganographic algorithms as well. While their construction varies in complexity, most of them boil down to manipulating the raw media object R, performing a transformation that turns it into an $\hat{R}$, the changes in which compared to the original are imperceptible to the human senses while providing it a certain property that is useful for covertly embedding a representation of secret data S. Least significant bits (LSB) methods operate on raw bits directly. Transform domain techniques like Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), etc. also operate on raw media object data. Same with phase coding, masking, filtering, and other transformative forms of steganography.

Selection hijacking method therefore has a unique advantage that separates it from previous models: it does not change the raw source media object but merely changes the way in which it is compressed and stored on disk. In a lossless compression algorithm, the hijacked compression leaves the raw source media completely intact, since, by the definition of a lossless compression algorithm, $D\big(C_E(R)\big) = D\left(\hat{C}_{E,S}(R)\right) = R$.

This model does have disadvantages, however. Any deviations from the optimal compression $C_{E,optimal}(R)$ will result in enlargement of the bit sequence $B\left(\hat{C}_{E,S}(R)\right)$. The compression space may not necessarily be monotonic, there may be local minima, and theoretically it is possible that a given hijacked encoding decision $e_i^{\hat{k}_i}$ may be able to decrease the overall length of the resulting bit sequence, it

ISSN 2707-1758

is expected to be an exception, not a rule. This would mean that the resulting compression $\hat{C}_{E,S}(R)$ will result in a larger file size compared to the optimal compression $C_{E,optimal}(R)$ achieved by the original compression algorithm. At the same time, it is worth noting that the resulting file contains more entropy than just the entropy of the source media object R, since it also contains a string of secret data S. Since the source media object R is left intact, it would mean that the entropy of a file with steganographically embedded data using this model is larger than the one without it. Simply put, this model results in larger file sizes.

Moreover, the storage capacity for secret data is limited by the number of decisions available to the SHjE. Depending on the compression algorithm used, it may result in orders of magnitude less storage capacity compared to the resulting file size. Since no methods based on this model have been developed yet, and no experimental data has been provided, it is difficult to estimate the exact impact on storage capacity it would allow.

Discussion and conclusions

With various data representation and storage schemes theorized, a problem arises: SHjE synchronization. To be able to correctly decode secret data stream encoded with this method, SHjEs on both ends of the process – encoding and decoding – need to be able to arrive at the same conclusions in each step of the way.

One solution is to share the implementation details and SHjE parameters beforehand. While it is a viable approach, it requires communicating these engine parameters beforehand through another, external channel. Once these engine parameters are locked in and hardcoded into the program that would perform the encoding/decoding, it would be impossible to change them without communicating through these external channels, whether for getting a new copy of the software with new hardcoded values, or for a new set of configuration options the end user would be able to input manually after receiving the necessary SHjE configuration.

A more user-friendly, though more challenging for implementors way of solving engine synchronization problems would be to embed SHjE configuration in the embedded secret data stream itself in the form of a predefined header. As long as both parties support the same base set of engine configurations, the encoding party can easily change engine parameters individually per media object, or even on the fly. Developing precise methods of engine synchronization is one of the possibilities for future research.

Secondly, since the hijacked selections will lead to lower compression performance, this will lead to a measurable and noticeable increase in storage capacity required for storing the compressed (encoded) media objects. This is an inherent property of this model. Because the resulting encoded media object theoretically retains all its original data, encoding additional secret data stream into it will necessitate an increase in file size. While it is a feature of this design, it is neither good nor bad. It is simply a design choice that can either be extremely useful to some users, or unnecessary to others.

This model would work well on lossless compression algorithms since, no matter what option is chosen by the SHjE, the implementation details and design of the compression algorithm used would eventually lead it to completely and identically representing the entire source media object in its processed (encoded) form. However, this may not be the case for lossy compression algorithms. By their nature, they discard certain information from the resulting data representation, and the decisions performed by the encoding algorithm influence which datapoints specifically get discarded. Hijacking those decisions would mean discarding different data from the resulting compressed media object. Depending on the design of the SHjE, this might lead to an encoder SHjE having different input data compared to the decoder SHjE, preventing them from being able to synchronize properly. Further research is required to adapt this model into specific methods that would address these concerns and make it possible to encode steganographic data in lossy media objects.

Furthermore, this model's performance is limited by the compression algorithm it is implemented on. To encode the secret data stream, it needs to not only perform the native compression steps but also spend computational resources into hijacking those decisions. Since those decisions are suboptimal, it is entirely likely that the compression algorithm would also need to spend more computational resources “cleaning up” after taking that suboptimal decision, further limiting its performance. Currently, it is only speculation, and further research is necessary to observe the impact of selection hijacking on compression speed and efficiency.

Additionally, without further SHjE tuning, with secret data streams that are shorter than the potential embedding capacity of a given media object, all the suboptimal compression decisions would be bunched up at the front of the file. This would allow for a potential statistical analysis attack that would allow adversaries to potentially detect this kind of manipulation. It is important to find ways of spreading out the hijacked decisions in a consistent way that is indistinguishable from randomness.

Finally, this model is well-suited for a wide variety of compression algorithms, but its implementation into actual methods requires specializing it for a particular compression algorithm, or even a particular implementation of encoders and decoders. While the theory might be broad, practical implementations need to be the focus of further research to make sure this model can be effectively leveraged in the future.

In conclusion, a new model of encoding steganographic data into media objects without changing their apparent representation was developed and proposed. This model is capable of being used on any kind of media if it has a compression algorithm that needs to perform decisions to optimize its performance. Further research is required to develop specific methods of implementing this model for specific data formats and compression algorithm implementations to bring this model from the realm of theory into the realm of practical application.

Authors' contribution: Yevhenii Ponomarenko – conceptualization, data curation, methodology, formal analysis, analysis of sources, writing – original draft; Oleksandr Laptiev – data curation, supervision, methodology, writing – review & editing.

Sources of funding. This study did not receive any grant from a funding institution in the public, commercial or non-commercial sectors.

References

Anas, T., Ridzuan, F., & Pitchay, S. A. (2025). Cover Selection in Steganography: A Systematic Literature Review. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 52(2), 107–129. https://doi.org/10.37934/araset.52.2.107129

Apau, R., Asante, M., Twum, F., Ben Hayfron-Acquah, J., & Peasah, K. O. (2024). Image steganography techniques for resisting statistical steganalysis attacks: A systematic literature review. *PLOS ONE*, 19(9), 1–47. https://doi.org/10.1371/journal.pone.0308807

ISSN 2707-1758

Barina, D. (2021). Comparison of Lossless Image Formats. *Computer Science Research Notes*, 3101, 339–342. https://doi.org/10.24132/CSRN.2021.3101.38

Galal, A. M. (2016). An analytical study on the modern history of digital photography. *International Design Journal*, 6(2), 203–215. https://www.faa-design.com/files/6/18/6-2-amr.pdf

Google. (2025, April 08). *Compression Techniques. WebP.* Google for Developers. https://developers.google.com/speed/webp/docs/compression

International Organization for Standardization. (1994). *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines* (ISO/IEC 10918-1:1994). https://www.iso.org/standard/18902.html

Öztürk, E., & Mesut, A. (2021). Performance evaluation of JPEG standards, WebP and PNG in terms of compression ratio and time for lossless encoding. In M. Yılmaz, & S. Kaya (Eds.). *Proceedings of the 2021 6th International Conference on Computer Science and Engineering (UBMK)* (n.pp.). IEEE. https://doi.org/10.1109/UBMK52708.2021.9558922

Rumsey, F., & Mccormick, T. (2013). *Sound and Recording* (6th ed.). New York: Focal Press. https://doi.org/10.4324/9780080953960

W3C. (2025, May 15). *Portable Network Graphics (PNG) Specification* (3rd ed.). W3C. https://www.w3.org/TR/png-3/

Witten, I. H., Neal, R. M., & Cleary, J. G. (1977, May). Arithmetic coding for data compression. *Communications of the ACM,* 30(6), 520–540. https://doi.org/10.1145/214762.214771

Отримано редакцією журналу / Received: 16.05.25
Прорецензовано / Revised: 16.05.25
Схвалено до друку / Accepted: 30.06.25

Євгеній ПОНОМАРЕНКО, асп.
ORCID ID: 0009-0000-2647-3381
e-mail: ponomarenkoye@fit.knu.ua
Київський національний університет імені Тараса Шевченка, Київ, Україна

Олександр ЛАПТЄВ, д-р техн. наук, доц., ст. наук. співроб.
ORCID ID: 0000-0002-4194-402X
e-mail: olaptiev@knu.ua
Київський національний університет імені Тараса Шевченка, Київ, Україна

МАТЕМАТИЧНА МОДЕЛЬ СТЕГАНОГРАФІЇ, ЩО ВИКОРИСТОВУЄ НЕОПТИМАЛЬНІ РІШЕННЯ В АЛГОРИТМАХ СТИСНЕННЯ ДАНИХ

Вступ. *Захист доступу до інформації в цифрову епоху вимагає розроблення цифрових інструментів для шифрування та приховування інформації від усіх, хто не повинен мати до неї доступ. Хоча шифрування чудово запобігає доступу до інформації неавторизованими особами, воно дозволяє людям знати, що за перетворенням щось приховано. Натомість, стеганографія дає змогу приховати сам факт приховування інформації. На жаль, поширені типи стеганографії покладаються на зміну вихідного сигналу, залишаючи малопомітні сліди маніпуляцій із даними, які можна відстежити та виявити. Метою цієї статті є надання моделі, яка має потенціал для збереження вихідного сигналу в повному обсязі, замість цього покладаючись на зміну способу представлення вихідного сигналу в цифрових носіях у такий спосіб, щоб мати можливість кодувати секретні дані у вихідний потік.*

Методи. *Проведено теоретичний аналіз стеганографічних підходів до алгоритмів стиснення даних. Досліджено методи збереження вихідного потоку даних.*

Результати. *Розроблено нову модель, яка використовує процеси прийняття рішень в алгоритмах стиснення даних для кодування стеганографічних даних у результуючому потоці даних. У схемах стиснення даних без втрат можливо досягти ідеального відтворення вихідного потоку даних, що унеможливлює виявлення шляхом аналізу основного сигналу, закодованого в алгоритмах стиснення даних.*

Висновки. *Потреба в інформаційній безпеці із часом зростає, а напруженість між країнами призводить до нових збройних конфліктів по всьому світу. Можливість вбудовувати та приховано надсилати дані за допомогою стеганографії може забезпечити конкурентну економічну, політичну та/або військову перевагу. Розроблену модель можна застосувати для подальшого розроблення конкретних методів стеганографічного кодування даних, стійких до аналізу та виявлення за допомогою існуючих підходів, що спираються на статистичний аналіз потоку базового сигналу.*

Ключові слова: *стеганографія, алгоритми стиснення даних, оброблення сигналів, захист інформації, дерева прийняття рішень, ентропійне кодування, арифметичне кодування.*

Автори заявляють про відсутність конфлікту інтересів. Спонсори не брали участі в розробленні дослідження; у зборі, аналізі чи інтерпретації даних; у написанні рукопису; в рішенні про публікацію результатів.

The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; in the decision to publish the results.

ISSN 2707-1758