Steganography is one of the information hiding techniques, which conceals secret messages in cover media. Digital image and audio are the most studied cover media for steganography. However, so far, there is no research on steganography to utilize complex networks as cover media. To investigate the possibility and feasibility of complex networks as cover media for steganography, we introduce steganography of complex networks through three algorithms: BIND, BYMOND, and BYNIS. BIND hides two bits of a secret message in an edge, while BYMOND encodes a byte in an edge, without changing the original network structures. Encoding simulation experiments for the networks of Open Graph Benchmark demonstrated BIND and BYMOND can successfully hide random messages in the edge lists. BYNIS synthesizes edges by generating node identifiers from a given message. The degree distribution of stego network synthesized by BYNIS was mostly close to a power-law. Steganography of complex networks is expected to have applications such as watermarking to protect proprietary datasets, or sensitive information hiding for privacy preservation.

**Keywords** Information Hiding - Steganography - Complex Network - Network Topology

Steganography is a type of invisible communications that hide secret messages in media [Fridrich 2009, Subbedar and Mankar 2014]. One of the important goals in steganography is to hide the existence of secret message as well as the message itself [Cheddad et al. 2010]. Steganography has been seriously studied since terrorists, spies, and hackers have been suspected of utilizing steganography to conceal secret messages for their malicious purposes until recently [Homer-Dixon, 2002, Rosner, 2006, Zielinska et al., 2014]. In modern digital steganography, a steganographic algorithm basically involves encoder and decoder (Fig. 1a). The encoder hides message bits in a medium, called cover. The encoded medium that contains the message, called stego, can be conveyed through public communication channels, as it is usually difficult to distinguish stegos from covers by human perception. The role of decoder is to recover the secret message from the stego without loss of information. Image and audio files are the most popular covers for steganography. For instance, Least-significant-bit (LSB) embedding for image steganography is the simplest algorithm, which embeds message bits in the LSB of each RGB color pixel in a lossless image format (Fig. 1b). The modified pixel is interpreted as a bit according to the parity of pixel value in the decoding process. This LSB embedding method can also be applied to audio steganography, where encoder hides the message bits in audio samples, as in RGB pixels (Fig. 1c).

Scientific data is also gaining attention as scalable cover media for steganography today, since the size of scientific data varies from kilobytes to petabytes. A representative example is DNA steganography. Yachie et al. and Shipman et al. successfully demonstrated the genomes of living organisms can encode digital data, which allows us to adopt a variety of genomes to store and hide information in DNA bases ranging from millions of bacterial bases to billions of human bases [Yachie et al., 2007, Shipman et al., 2017]. Clelland et al. first introduced a DNA steganography, in which DNA triplet represents a single alphabetic letter [Clelland et al., 1999]. Na devised a method that hides secret messages in the variable regions of genome (single nucleotide polymorphisms) to evade detection [Na, 2020]. Li et al. developed an experimental method using CRISPR/Cas12a system, which takes advantage of the specific and non-specific primers in polymerase chain reaction (PCR) as real and fake keys, respectively, to enhance security [Li et al., 2018]. Moreover, in
recent years, a DNA steganalysis based on deep learning has been developed to detect DNA steganography [Bae et al., 2020].

In addition to DNA-based steganography, various methods have shown steganography of scientific data. Kim et al. introduced a steganography based on an immuno-chemical system, where immuno-specific interactions on the ELISA plate result in the combinations of colors that encode text messages [Kim et al., 2011]. Sakar et al. developed a chemical-based method that hides secret messages within the emission spectra of a unimolecular fluorescent sensor [Sarkar et al., 2016]. Boukis et al. demonstrated a secret chemical communication system, where the molecular keys are decoded by high resolution tandem mass spectrometry, and the decoded messages can be used as passwords for digital encryption algorithms such as AES [Boukis et al., 2018]. Purcell et al. developed an experimental method that hides the original topology of a synthetic gene circuit by camouflaging the circuit, and recovers the original circuit by adding molecular keys to remove the activity of genes in the camouflaged circuit [Purcell et al., 2018]. Zhang et al. experimentally demonstrated a protein binding-based steganography based on DNA origami cryptography [Zhang et al., 2019].

However, so far, there is no research on steganography using network structure data as cover media. A variety of phenomena in the real world can be represented and analyzed in complex networks, and the field of network science
has emerged to systematically analyze networks [Barabási, 2014]. As graph theory and network science advance, some satisfactory solutions for complex network problems such as traffic system [Dijkstra et al., 1959], search engine [Page et al., 1999], terrorism [Carley et al., 2002], etc. have been obtained. Recently, deep neural networks for graphs, called graph neural networks (GNN), are bringing the power of artificial intelligence to network science [Wu et al., 2020]. Recognizing the importance of systematic evaluation, a benchmarking dataset including carefully curated networks of different sizes, types, and tasks has been opened for developing and comparing the performance of machine learning models [Hu et al., 2020]. Despite the advent of various types of real-world network datasets and corresponding analytical models, steganography for complex networks has not been studied.

To investigate the possibility and feasibility of complex networks as cover media for information hiding, we introduce a novel steganography for network datasets (Fig. 1d). We developed three steganographic algorithms: BIND, BYMOND, and BYNIS. BIND and BYMOND algorithms hide secret messages in existing network data files such as edge lists, without changing the original topology of the networks. BIND encodes the bits of a given message into the node degrees of an edge in an edge list, while BYMOND encodes the message bytes, rather than the bits, into the node degrees. We demonstrate BIND and BYMOND algorithms can successfully hide randomly generated messages in the real-world network datasets of Open Graph Benchmark (OGB) [Hu et al., 2020]. On the other hand, BYNIS creates a synthetic network for a given message, while the degree distribution of the synthetic network is close to power-law distribution. We expect that steganography of complex networks will have important applications such as watermarking to protect proprietary datasets of a company, or sensitive information hiding for privacy preservation of patients.

Results

Steganographic Algorithms for Real-World Networks

One of the important goals of steganographic algorithms is to hide a secret message in the existing data. Therefore, we developed two steganographic algorithms for real-world network datasets: BIND and BYMOND (Fig. 2 and Fig. 4). The format of input and output in both algorithms is an edge list, which is the most common and simplest data format for real-world networks.

BIND algorithm

BIND (BIT is encoded in Node Degrees of an edge) encodes the bits of a given message into a list of edges according to node parity (Fig. 2a). First, BIND categorizes each edge according to the parities of two node degrees. For instance, if the first node has degree 3 and the second node has degree 4, the edge is categorized as “OE”, which represents an edge of the “Odd” and “Even” degrees (Fig. 2, the edge with k=3 and k=4). The important point in this process is that the order of appearance of the nodes must be distinguished in an edge. So, “OE” and “EO” are categorized as different edge types. BIND then reads the message bits in units of two bits, and matches the corresponding edge. For instance, “01” in the message bits corresponds to an edge between node A and node B, whose type is “EO” (Fig. 2, the edge shown in red shade). Finally, stego edges are arranged by random indexing with a seed that is usually assigned by user password. Only the order of edges in the stego edge list is different from the order of edges in the cover edge list. Recovering the original message from the stego edge list of BIND is reversing the random indexing with the seed, and sequentially interpreting the stego edge types as message bits.

The real-world networks with the evenly distributed edge types are promising covers for BIND algorithm, if we assume that lossless compression and cryptographic algorithms can randomize the message bits following a uniform distribution [Bassham et al., 2010, Klein and Shapira, 2020]. Hence, we analyzed the edge categorization of BIND for OGB datasets (Fig. 2b). Interestingly, the proportions of the four edge types are almost the same in each dataset except "wikikg2". The distribution of edges in "wikikg2" is slightly biased towards the "OE" type. These results imply that real-world networks exemplified by OGB datasets are suitable for BIND algorithm, if we assume that the 2-bit patterns in secret messages are uniformly distributed.

To analyze the payload capacity of complex network steganography, we defined a payload capacity measure, named BPE (Bits Per Edges), as follows:

\[ BPE := \frac{|B_{\text{msg}}|}{|E|} \]

where \(|B_{\text{msg}}|\) is the number of message bits, and \(|E|\) is the number of edges in the edge list. As a single edge encodes two bits in BIND, the maximum BPE of BIND is theoretically calculated as follows:

\[ BPE_{\text{max}} := \frac{|B_{\text{thr}}|}{|E|} = 2 \cdot \frac{|E|}{|E|} = 2 \]
To validate the steganographic algorithms for complex networks, we selected various sizes of edge lists from Open Graph Benchmark (OGB) datasets. The names in parentheses are the original ID of OGB. The items are sorted by the number of edges.

| No. | Dataset        | Num. Nodes (|V|) | Num. Edges (|E|) | Description                                                                 |
|-----|----------------|----------|----------------|------------------------------------------------|-----------------------------------------------------------------------------|
| 1   | ddi (ogbl-ddi) | 4,267    | 1,334,889      | A undirected network of drug-drug interactions [Wishart et al., 2018].     |
| 2   | arxiv (ogbn-arxiv) | 169,343  | 1,166,243      | A directed network of citations between all computer science papers of arXiv indexed by Microsoft Academic Graph (MAG) [Wang et al., 2020]. |
| 3   | collab (ogbl-collab) | 235,868  | 1,285,465      | A undirected network of collaborations between authors indexed by MAG [Wang et al., 2020]. |
| 4   | wikig2 (ogbl-wikig2) | 2,500,604 | 17,137,181    | A knowledge graph from Wikidata knowledgebase [Vrandečić and Krötzsch, 2014]. |
| 5   | ppa (ogbl-ppa)  | 576,289  | 30,326,273     | A undirected network of protein-protein associations [Szklarczyk et al., 2019]. |
| 6   | citation2 (ogbl-citation2) | 2,927,963 | 30,561,187   | A directed network of citations between papers indexed by MAG [Wang et al., 2020]. |
| 7   | proteins (ogbn-proteins) | 132,534  | 39,561,252     | A undirected network of protein-protein associations. [Szklarczyk et al., 2019]. |
| 8   | products (ogbn-products) | 2,449,029 | 61,859,140    | A undirected network of Amazon co-purchase [Bhatia et al., 2016]. |

where the theoretical maximum number of the message bits, \( |B_{th}^{max}| \), is equal to \( 2 \cdot |E| \), which means all edges encode the message bits in the edge list. However, the minimum size of edge set among the sets of the four edge types determines the lower bound of the maximum BPE in BIND, since the encoding is impossible if the edges of a certain edge type corresponding to a 2-bit pattern are insufficient. Therefore, the lower bound of \( BPE_{max} \) can be described as follows:

\[
|B_{max}| \geq 4 \cdot |E_{min}| \quad \Rightarrow \quad BPE_{max} = \frac{|B_{max}|}{|E|} \geq \frac{4 \cdot |E_{min}|}{|E|}
\]

where \( E_{min} \) is the set of the minimum size among the sets of the four edge types in BIND and \( |E_{min}| \) is the size of \( E_{min} \). We can use the lower bound to estimate actual \( |B_{max}| \) and \( BPE_{max} \) of a given edge list if we do not know the exact \( BPE_{max} \) as follows:

\[
|B_{est_{max}}| = 4 \cdot |E_{min}| \quad \Rightarrow \quad BPE_{est_{max}} := \frac{4 \cdot |E_{min}|}{|E|}
\]

where \( B_{est_{max}} \) and \( BPE_{est_{max}} \) are the estimates of \( |B_{max}| \) and \( BPE_{max} \), respectively. The \( BPE_{est_{max}} \) can be utilized as a basic measure for evaluating and comparing the payload capacities of complex network covers. We also defined \( R_{A/E} \) to efficiently analyze and control the message size as follows:

\[
R_{A/E} := \frac{|B_{msg}|}{|B_{est_{max}}|} \quad \Rightarrow \quad |B_{msg}| = R_{A/E} \cdot |B_{est_{max}}| = R_{A/E} \cdot (4 \cdot |E_{min}|)
\]

where the subscript, \( A/E \), represents a ratio between the actual number of message bits and the estimated maximum.

We performed simulation experiments to analyze the payload capacity in BIND (Fig. 3). We repeated the encoding simulation 100 times for random messages and measured the success rate. Based on equation (1), random messages of \( |B_{msg}| \) for simulation were generated following a uniform distribution. BIND successfully encoded random messages, but the success rate dramatically decreased for \( R_{A/E} = 1.0 \) (Fig. 3). In the process of encoding the messages generated with \( R_{A/E} = 1.0 \), the number of edges of a particular type corresponding to a 2-bit pattern could be insufficient with a high probability, since the message size, \( |B_{msg}| = |B_{est_{max}}| \), was very close to the actual maximum size. To understand
that the success rate decreases under the condition of $R_{A/E} = 1.0$, we also counted the failed cases of each edge type in the encoding simulation experiments for BIND algorithm. We found that almost all failed cases are attributed to the lack of $E_{\text{min}}$ edges (Table 2 and Fig. 3b, dark cells). These results imply that the payload size must be carefully determined in BIND algorithm, and we can use $R_{A/E}$ less than 1.0 to control the payload capacity.

**BYMOND algorithm**

To improve the payload capacity, we developed **BYMOND** (BYte is encoded in MOdulo of the sum of Node Degrees of an edge) algorithm (Fig. 4a). Instead of encoding two bits, an edge encodes a byte (8 bits) in BYMOND. As one byte can have 256 different values (i.e., 0 to 255), BYMOND categorizes edges into 256 types according to the modulo of the sum of node degrees. For instance, if an edge with $k = 3$ and $k = 4$ is categorized as 7, then the edge encodes a single byte, 7 (Fig. 4b). The reason for applying modulo operation to the sum of degrees is that the sum of degrees can exceed 255, which is the maximum value of a byte. The rest of encoding and decoding processes in BYMOND is the same as BIND algorithm, except that an edge encodes a byte. Figure 4b shows the results of edge categorization in BYMOND. Unlike the results of BIND, the edge types of BYMOND are not evenly distributed, but have a variety of distributions. For example, the distributions of "arxiv", "collab", and "citation2" are skewed towards the edge types of lower values. However, the edges of "protein" dataset are almost evenly distributed compared to the other datasets.
Steganography of Complex Networks

Figure 3: Encoding simulation experiments for BIND algorithm. (a) We randomly generated messages according to $R_{A/E}$, and observed whether the steganographic algorithm had successfully completed encoding the given messages for OGB datasets. $R_{A/E}$ is the ratio of the actual number of message bits to the estimated maximum number of message bits. Success rate represents the fraction of successful completions in the 100 simulation experiments. (b) To understand the success rate decreases under the condition of $R_{A/E} = 1.0$, we also counted the failed cases of each edge type in the simulation experiments. The color represents the ratio of failed cases to the total failed cases for each edge type.

Table 2: Estimation of payload capacity in BIND. $E_{\text{min}}$: the set of minimum size among the sets of the 4 edge types in BIND. $|E_{\text{min}}|$: the size of $E_{\text{min}}$; $|B_{\text{est}}^{\text{max}}|$: the estimated maximum number of the message bits, which is calculated by $4 \cdot |E_{\text{min}}|$; $|B_{\text{thr}}^{\text{max}}|$: the theoretical maximum number of the message bits; $R_{E/T}$: the ratio of $|B_{\text{est}}^{\text{max}}|$ to $|B_{\text{thr}}^{\text{max}}|$ (i.e., $|B_{\text{est}}^{\text{max}}|/|B_{\text{thr}}^{\text{max}}|$), which indicates how close the estimate is to the theoretical value.

| Dataset | Type of $E_{\text{min}}$ | $|E_{\text{min}}|$ | $|B_{\text{est}}^{\text{max}}|$ | $|B_{\text{thr}}^{\text{max}}|$ | $R_{E/T}$ |
|---------|--------------------------|-----------------|-----------------|-----------------|---------|
| ddi     | EE                       | 248,568         | 994,272         | 1,067,910       | 0.931   |
| arxiv   | EE                       | 290,238         | 1,160,952       | 1,166,242       | 0.995   |
| collab  | OE                       | 252,044         | 1,008,176       | 1,179,051       | 0.855   |
| wikikg2 | EO                       | 3,043,263       | 12,173,052      | 16,109,181      | 0.756   |
| ppa     | EE                       | 5,266,145       | 21,064,580      | 21,231,930      | 0.992   |
| citation2| EE                      | 7,541,565       | 30,166,260      | 30,387,994      | 0.993   |
| proteins| EE                       | 9,500,440       | 38,001,760      | 39,561,251      | 0.961   |
| products| OE                       | 15,426,359      | 61,705,436      | 61,859,139      | 0.998   |

We also performed the encoding simulation experiments for BYMOND, where the simulation was repeated 1,000 times for random messages. In BYMOND, $B_{\text{est}}^{\text{max}}$ and $|B_{\text{msg}}|$ are determined as follows:

$$|B_{\text{est}}^{\text{max}}| = 256 \cdot |E_{\text{min}}| \quad \Rightarrow \quad |B_{\text{msg}}| = R_{A/E} \cdot (256 \cdot |E_{\text{min}}|)$$

(2)

where 256 is the number of edge types in BYMOND. The encoding success rates of BIND rate ranged from 0.4 to 0.6 for OGB datasets when $R_{A/E} = 1.0$ (Fig. 3a), whereas the success rates of BYMOND were different depending on the dataset (Fig. 5a). BYMOND failed to encode random messages under $R_{A/E} = 1.0$ for "ddi", "arxiv" and "ppa" with a high probability. For the other datasets such as "collab", "wikikg2", "citation2", "proteins" and "products", the success rates of BYMOND were between 0.35 and 0.5. Figure 5b shows the ratio of failed cases to the total failed cases for each edge type in the 1,000 simulations. The encoding failures for "collab", "wikikg2", "citation2", "proteins" and "products" were mainly due to the lack of $E_{\text{min}}$ edges (Table 3 and Fig. 5b, dark bars). However, the failures for "ddi", "arxiv" and "ppa" datasets were attributed to the lack of several edge types (Fig. 5b, several peach and orange bars). These results suggest that the probability of encoding failure increases in BYMOND when the sizes of multiple edge sets in a network dataset are close to $|E_{\text{min}}|$. Therefore, the distribution of edge types and $|E_{\text{min}}|$ should be carefully considered to select cover networks for BYMOND algorithm.
BYMOND (BYte is encoded in MOdulo of the sum of Node Degrees of an edge)

| Node1 | Node2 |
|-------|-------|
| A     | B     |
| A     | C     |
| A     | D     |
| B     | C     |
| B     | E     |
| C     | F     |
| C     | H     |

Edge Categorization

(k1 + k2) % 256

| Node1 | Node2 | MOD(k) |
|-------|-------|--------|
| A     | B     | 173    |
| A     | C     | 193    |
| A     | D     | 185    |
| A     | G     | 190    |
| B     | C     | 170    |
| B     | E     | 164    |
| C     | F     | 179    |
| C     | H     | 162    |

Total 256 Edge Types

BYNIS (BYte is encoded in Nodenumber of a Synthetic edge)

BYNIS algorithm

BIND and BYMOND algorithms utilize existing real-world network datasets. However, high payload capacity is not guaranteed if the degree distribution of cover network does not conform to the bit or byte patterns of message data (Fig. 3 and Fig. 5). Hence, we developed a steganographic algorithm, named BYNIS(BYte is encoded in the sum of Node IDs of a Synthetic edge), which synthesizes the edges of complex networks according to a given message (Fig. 6). First, BYNIS splits a byte into two integers, which represent the two node identifiers (IDs) of an edge. In contrast to BYMOND, BYNIS algorithm encodes a message byte into the sum of node IDs, not node degrees. A bias can be added to the message bytes to prevent the failure of generating node IDs when the byte values of message are too small. BYNIS can also refer to reference degrees for generating node IDs. The node ID generation algorithm is a kind of greedy algorithm, which generates a node ID whose degree is currently the maximum. This greedy algorithm sequentially divides message bytes to generate a specific node ID until the reference degree for the specific node ID is exhausted to create synthetic edges. For instance, if the first degree is 3 in reference degrees, BYNIS splits the first 3 bytes into 3 pairs of node IDs that must have node ID “0” (Fig. 6a). As the node degree 3 is exhausted for node ID
Figure 5: **Encoding simulation experiments for BYMOND algorithm.** All the same as in Figure 3, except that the number of edge types in BYMOND is 256, and simulation experiment was repeated 1,000 times (see the main text for details).

Table 3: **Estimation of payload capacity in BYMOND.** $E_{min}$: the set of minimum size among the sets of the 256 edge types in BYMOND; $B_{est\ max}$: the estimated maximum number of the message bits, which is calculated by $256 \cdot |E_{min}|$; the others are the same as those in Table 2.

| Dataset      | Type of $E_{min}$ | $|E_{min}|$   | $|B_{est\ max}|$ | $|B_{thr\ max}|$ | $R_{E/T}$ |
|--------------|------------------|--------------|------------------|------------------|----------|
| ddi          | 109              | 3,804        | 7,790,592        | 8,543,280        | 0.912    |
| arxiv        | 253              | 1,536        | 3,145,728        | 9,329,936        | 0.337    |
| collab       | 254              | 275          | 563,200          | 9,432,408        | 0.060    |
| wikikg2      | 0                | 20,206       | 41,381,888       | 128,873,448      | 0.321    |
| ppa          | 9                | 65,625       | 134,400,000      | 169,855,440      | 0.791    |
| citation2    | 1                | 39,703       | 81,311,744       | 243,103,952      | 0.334    |
| proteins     | 26               | 149,477      | 306,128,896      | 316,490,008      | 0.967    |
| products     | 1                | 197,578      | 404,639,744      | 494,873,112      | 0.818    |

"0", the next node ID is "1". To recover the original message from the stego edge list of BYNIS, we should apply the modulo operation to the sum of the node IDs as follows:

$$B_{rec}^i := (id_1 + id_2) \ mod \ bias$$

where $B_{rec}^i$ is the $i$-th recovered byte, and $id_1$ and $id_2$ represent the two node IDs of the $i$-th synthetic edge in a stego edge list. In Figure 6 for instance, the 7-th edge is decoded as follows:

$$B_{rec}^7 = (3 + 271) \ mod \ 256 = 18$$

To characterize synthetic networks created by BYNIS, we investigated how synthetic networks are created according to the reference degrees of different random network models. Given three reference degrees generated from Watts-Strogatz, Erdős-Rényi, and Barabási-Albert models, the network synthesized with Barabási-Albert model was most similar to the original random network (Fig. 6b, p-value > 0.1 for two degree distributions). In other words, splitting message bytes into node IDs based on the greedy approach for edge generation in BYNIS creates a network, whose degree distribution is close to a power law distribution [Barabási and Albert, 1999].

**Discussion**

We introduce steganography of complex networks by demonstrating possibility and feasibility of three steganographic algorithms through analysis of real-world network datasets. BIND algorithm encodes two bits of a given secret message into two degrees of an edge according to node degree parity, while BYMOND algorithm encodes a single byte of message into the sum of node degrees of an edge adjusted by modulo operation. BYNIS algorithm synthesizes edges by splitting message bytes into node identifiers based on a greedy approach, resulting in a synthetic network whose degree distribution follows a power-law.

As payload capacity is one of the most important aspects in steganography, we define $BPE$ (Bits Per Edges), and explain how to estimate $BPE_{max}$ with $|B_{est\ max}|$ and $|E_{min}|$. In the ideal case, the number of a specific bit or byte
Figure 6: **BYNIS algorithm.** (a) Schematic diagram of BYNIS algorithm. An edge is synthesized by splitting a single byte of secret message into two node identifiers of the edge. (b) Structures and degree distributions of synthetic networks created by BYNIS referring to random network models. $P$ represents $p$-value of Kolmogorov-Smirnov test for two samples, where null hypothesis is that two samples are drawn from the same distribution.
Steganography of Complex Networks

pattern in secret message exactly matches the number of edges for each edge type of BIND or BYMOND. However, it is usually hard to find a real-world network that perfectly conform to a secret message data. In general, \( |E_{\text{min}}| \) limits the maximum payload capacity, \( |B_{\text{max}}| = \text{the number of edge types} \cdot |E_{\text{min}}| \) can guide the payload capacity of a given cover network for BIND or BYMOND algorithm.

Encoding simulation experiments for OGB datasets show that BIND and BYMOND can fail to encode messages of high capacity that are close to the maximum capacity of message, \( |B_{\text{max}}| \). Since the distribution of the edge types of BYMOND in each OGB dataset do not follow uniform distribution (Fig. 4b), the encoding success rate of BYMOND varied depending on dataset in the simulation experiments (Fig. 5a). In other words, BYMOND is more sensitive to edge type distribution than BIND, although the payload capacity of BYMOND is theoretically 4 times larger than that of BIND. To make the distribution of edge types follow the uniform distribution, we can define a new function that determines edge types based on node degrees as follows:

\[
B_i := f(k_1, k_2)
\]

where \( B_i \) is the \( i \)-th message byte, and \( k_1 \) and \( k_2 \) are two node degrees of an edge. In BYMOND, \( f \) is defined as follows:

\[
f(k_1, k_2) := (k_1 + k_2) \mod 256
\]

A well-designed function \( f \) is expected to achieve both evenly distributed edge types and high payload capacities for a variety of real-world networks, overcoming the limitations of BYMOND. Steganography of synthetic networks is exemplified by BYNIS, which is essentially a hybrid algorithm that integrates text steganography [Taleby Ahvanoee et al., 2019] with the structural information of complex network. In other words, the node identifier characters of an edge are generated to create a synthetic network, referring to a degree distribution. To reflect the properties of real-world networks to synthetic networks, BYNIS utilizes the degree distribution of a reference network. However, BYNIS does not exactly reflect any degree distribution as shown in the experiments of random network models, except power-law distribution (Fig. 6b). Hence, networks with power-law distributions should be preferred to other distributions in order to take advantage of the BYNIS algorithm. Recently, Broido et al. has demonstrated that scale-free networks are empirically rare, and log-normal distributions (i.e., \( \frac{1}{\sigma \sqrt{2\pi}} e^{-(\log x - \mu)^2 / 2\sigma^2} \)) are more appropriate than power laws to explain degree distributions of most real-world networks [Broido and Clauset, 2019]. Therefore, the edge synthesis algorithm, currently based on a greedy approach in BYNIS, can be improved to reproduce various degree distributions including log-normal as well as power-law. We expect steganography of complex networks to have useful applications. As networks in industry such as social networks or product networks are becoming important assets [Haenlein, 2011, Carmi et al., 2017], steganographic algorithms for complex networks can be adopted as watermarking techniques to protect the proprietary datasets. Another important application is to hide sensitive information in network datasets. For instance, when we need to publicly distribute patient-specific cancer networks [Drake et al., 2016], we can hide personal information of a patient in his or her network data without changing the original structure of the network. We hope that this study will be the first step to facilitate the development of various algorithms and applications based on steganography of complex networks.
Methods

Algorithm implementation

All steganographic algorithms have been implemented in Python programming language. To enhance performance of algorithms, we use NumPy [Harris et al., 2020] and Pandas [Wes McKinney, 2010] packages, which enable us to utilize high performance vectorization in array programming. We can choose between NetworkX [Hagberg et al., 2008] and igraph [Csardi and Nepusz, 2006] to handle the data structures of undirected and directed networks in our implementation. bitstring [Griffiths, 2020] is adopted to efficiently process bits and bytes of message data. We provide the core part of each algorithm written in pseudo-code as follows:

Algorithm 1: Message encoding algorithm of BIND

Input: cover network $G_c$; cover edge list $E_c$; message bits $M_b$; password $P$
Output: stego edge list $E_s$

// Edge categorization
foreach edge in $E_c$
    node1, node2 ← edge
    $D_1$ ← Degree($G_c$, node1)
    $D_2$ ← Degree($G_c$, node2)
    if ($D_1$ mod 2 = 0) and ($D_2$ mod 2 = 0)
        Append($E_{ee}$, edge)
    else if ($D_1$ mod 2 = 0) and ($D_2$ mod 2 = 1)
        Append($E_{eo}$, edge)
    else if ($D_1$ mod 2 = 1) and ($D_2$ mod 2 = 0)
        Append($E_{oe}$, edge)
    else if ($D_1$ mod 2 = 1) and ($D_2$ mod 2 = 1)
        Append($E_{oo}$, edge)

// Message encoding
foreach two bits $b_1, b_2$ in $M_b$
    if ($b_1$ mod 2 = 0) and ($b_2$ mod 2 = 0)
        edge ← Pop($E_{ee}$)
    else if ($b_1$ mod 2 = 0) and ($b_2$ mod 2 = 1)
        edge ← Pop($E_{eo}$)
    else if ($b_1$ mod 2 = 1) and ($b_2$ mod 2 = 0)
        edge ← Pop($E_{oe}$)
    else if ($b_1$ mod 2 = 1) and ($b_2$ mod 2 = 1)
        edge ← Pop($E_{oo}$)
    Append($E_s$, edge)

// Randomization of edge sequence
Seed($P$)
Randomize($E_s$)
return $E_s$
Algorithm 2: Message encoding algorithm of BYMOND

Input: cover network \( G_c \); cover edge list \( E_c \); message bytes \( M_B \); password \( P \)

Output: stego edge list: \( E_s \)

// Edge categorization
foreach edge in \( E_c \) do
    node\(_1\), node\(_2\) ← edge
    \( D_1 \) ← Degree\( (G_c, node\(_1\)) \)
    \( D_2 \) ← Degree\( (G_c, node\(_2\)) \)
    \( t \) ← \( (D_1 + D_2) \mod 256 \)
    Append\( (E_s[t], edge) \)

// Message encoding
foreach one byte \( B \) in \( M_B \) do
    \( t \) ← \( B \)
    edge ← Pop\( (E_s[t], edge) \)
    Append\( (E_s, edge) \)

// Randomization of edge sequence
Seed\( (P) \)
Randomize\( (E_s) \)
return \( E_s \)

Algorithm 3: Network synthesis algorithm of BYNIS

Input: reference degrees \( D_r \); message bytes \( M_B \)

Output: stego edge list \( E_s \)

// \( D_r \) is assumed to be a sorted array
bias ← 256
\( M_B \) ← \( M_B + bias \) // Add a bias to message bytes.
\( D \) ← ZeroLike\( (D_r) \) // Zero-initialized array like \( D_r \)
i ← 0 // Current node ID

foreach one byte \( B \) in \( M_B \) do
    if \( D[i] < D_r[i] \) then
        // Consume the current node degree.
        \( D[i] \) ← \( D[i] + 1 \)
    else
        // Update to the next node.
        \( i \) ← \( i + 1 \)
        // Greedy approach: select a node whose degree is currently the maximum.
        id\(_1\) ← \( i \) // \( i \) represents the node that has the max degree.
        id\(_2\) ← \( B - i \)
        // Create a new edge with node IDs.
        edge ← \( (id_1, id_2) \)
        // Change the second node ID if the synthetic edge already exists.
        \( j \) ← \( 1 \)
        while edge exists in \( E_s \) do
            \( id_2 \) ← \( id_2 + bias \ast j \)
            edge ← \( (id_1, id_2) \)
            \( j \) ← \( j + 1 \)
        Append\( (E_s, edge) \)
    return \( E_s \)
Network datasets

Open Graph Benchmark (OGB) provides real-world networks, which we use to validate our steganographic algorithms. We can download the OGB datasets using OGB python package. When creating a dataset object such as PygNodePropPredDataset or PygLinkPropPredDataset, OGB package downloads each dataset if it does not exist in a local storage. The following Python source code is an example for downloading the datasets.

```python
from ogb.nodeproppred import PygNodePropPredDataset
from ogb.linkproppred import PygLinkPropPredDataset

list_nodeproppred_datasets = [
    "ogbn-arxiv",
    "ogbn-proteins",
    "ogbn-products"
]

list_linkproppred_datasets = [
    "ogbl-ddi",
    "ogbl-collab",
    "ogbl-wikikg2",
    "ogbl-ppa",
    "ogbl-citation2",
]

dpath_download = "\data\ogb\"

for name in list_nodeproppred_datasets:
    dataset = PygNodePropPredDataset(name=name, root=dpath_download)

for name in list_linkproppred_datasets:
    dataset = PygLinkPropPredDataset(name=name, root=dpath_download)
```

In the directory of each OGB dataset, we only use the edge list of a raw format (e.g., /data/ogb/ogbl_ddi/raw/edge.csv). Some datasets including ogbl-collab have redundant rows as multiple edges in their raw edge lists. All redundant edges in the edge list are maintained and utilized for message encoding. However, BIND and BYMOND assume all edges are unique in a directed graph when interpreting the properties of network structure. So, the number of edges in a edge list and the number of edges in a network structure can be different.

Encoding simulation experiments

We performed encoding simulation experiments to understand the encoding algorithms of BIND and BYMOND for random messages. In the experiments for BIND, we increased $R_{A/E}$ from 0.7 to 1.0 by 0.5, and performed 100 simulations for each $R_{A/E}$. In a single simulation, message size, $|B_{msg}|$, was determined by equation (1). In the experiments for BYMOND, the simulation conditions were the same as in BIND except the number of simulations and the message size. To catch encoding failures as many as possible, we set the number of simulations greater than the number of edge types, 256. The number of simulations was set 1,000 ($\geq 256$), and the message size was determined by equation (2) in BYMOND.

Network synthesis with random network models

To generate reference degrees, we used random network generation functions of NetworkX\[Hagberg et al. 2008\]. The following example shows the functions and parameters we used for generating the reference networks in Figure 6. We computed Kolmogorov-Smirnov static to compare the degree distributions between reference network and stego network.

```python
import networkx as nx
import scipy as sp

list_ref = []
num_nodes = 200

# Watts-Strogatz
```
Steganography of Complex Networks

```python
# Create networks

g = nx.newman_watts_strogatz_graph(num_nodes, 2, 0.01)
list_ref.append(g)

# Erdos-Renyi

g = nx.fast_gnp_random_graph(int(1.2*num_nodes), 0.008, directed=False)
list_ref.append(g)

# Barabasi-Albert

g = nx.barabasi_albert_graph(num_nodes, 1)
list_ref.append(g)

# Get the giant component
for i, g in enumerate(list_ref):
    nodes_gc = sorted(nx.connected_components(g), key=len, reverse=True)
    g_ref = g.subgraph(nodes_gc[0])

    # BYNIS algorithm
    g_stego = encode(msg_bytes, g_ref)

    degrees_ref = get_degrees(g_ref)
    degrees_stego = get_degrees(g_stego)

    # Compare two samples using Kolmogorov-Smirnov test
    res = sp.stats.ks_2samp(degrees_ref, degrees_stego)
    print("P = %.4e (%f)"%(res.pvalue, res.pvalue))
```

Data availability

To obtain the network datasets in this study, we need to download OGB datasets [https://ogb.stanford.edu](https://ogb.stanford.edu). Refer to "Network datasets" in "Methods" section for details.

Code availability

We provide a GitHub repository for the steganographic algorithms: [https://github.com/dwgoon/sgcn](https://github.com/dwgoon/sgcn)
Steganography of Complex Networks

References

Jessica Fridrich. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, USA, 1st edition, 2009. ISBN 0521190193.

Mansi S. Subhedar and Vijay H. Mankar. Current status and key issues in image steganography: A survey. *Computer Science Review*, 13-14(C):95–113, nov 2014. ISSN 15740137. doi:10.1016/j.cosrev.2014.09.001.

Abbas Cheddad, Joan Condell, Kevin Curran, and Paul Mc Kevitt. Digital image steganography: Survey and analysis of current methods. *Signal Processing*, 90(3):727–752, 2010. ISSN 01651684. doi:10.1016/j.sigpro.2009.08.010. URL http://dx.doi.org/10.1016/j.sigpro.2009.08.010.

Thomas Homer-Dixon. The rise of complex terrorism. *Foreign Policy*, (128):52–62, 2002. ISSN 00157228. URL http://www.jstor.org/stable/3183356.

Chet Hosmer. Discovering hidden evidence. *Journal of Digital Forensic Practice*, 1(1):47–56, 2006. doi:10.1080/15567280500541447. URL https://doi.org/10.1080/15567280500541447.

Elżbieta Zielińska, Wojciech Mazurczyk, and Krzysztof Szczypiorski. Trends in steganography. *Commun. ACM*, 57(3):86–95, March 2014. ISSN 0001-0782. doi:10.1145/2566590.2566610. URL https://doi.org/10.1145/2566590.2566610.

Nozomu Yachie, Kazuhide Sekiyama, Junichi Sugahara, Yoshiaki Ohashi, and Masaru Tomita. Alignment-based approach for durable data storage into living organisms. *Biotechnology progress*, 23(2):501–505, 2007.

Seth L Shipman, Jeff Nivala, Jeffrey D Macklis, and George M Church. Crispr–cas encoding of a digital movie into the genomes of a population of living bacteria. *Nature*, 547(7663):345–349, 2017.

Catherine Taylor Clelland, Viviana Risca, and Carter Bancroft. Hiding messages in dna microdots. *Nature*, 399(6736):533–534, 1999.

Dokyun Na. Dna steganography: hiding undetectable secret messages within the single nucleotide polymorphisms of a genome and detecting mutation-induced errors. *Microbial cell factories*, 19(1):1–9, 2020.

Shi-Yuan Li, Jia-Kun Liu, Guo-Ping Zhao, and Jin Wang. Cads: Crispr/cas12a-assisted dna steganography for securing the storage and transfer of dna-encoded information. *ACS synthetic biology*, 7(4):1174–1178, 2018.

Ho Bae, Seonwoo Min, Hyun-Soo Choi, and Sungroh Yoon. Dna privacy: Analyzing malicious dna sequences using deep neural networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

Kyung-Woo Kim, Vera Bocharova, Jan Halánek, Min-Kyu Oh, and Evgeny Katz. Steganography and encrypting based on immunochromical systems. *Biotechnology and bioengineering*, 108(5):1100–1107, 2011.

Tanmay Sarkar, Karuthapandi Selvakumar, Leila Motiei, and David Margulies. Message in a molecule. *Nature communications*, 7(1):1–9, 2016.

Andreas C Boukis, Kevin Reiter, Maximiliane Frölich, Dennis Hofheinz, and Michael AR Meier. Multicomponent reactions provide key molecules for secret communication. *Nature communications*, 9(1):1–10, 2018.

Oliver Purcell, Jerry Wang, Piro Siuti, and Timothy K Lu. Encryption and steganography of synthetic gene circuits. *Nature communications*, 9(1):1–10, 2018.

Yinan Zhang, Fei Wang, Jie Chao, Mo Xie, Huajie Liu, Muchen Pan, Enzo Kopperger, Xiaoguo Liu, Qian Li, Jiye Shi, et al. Dna origami cryptography for secure communication. *Nature communications*, 10(1):1–8, 2019.

Albert-László Barabási. Network science book. *Network Science*, 625, 2014.

Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Kathleen M Carley, Ju-Sung Lee, and David Krackhardt. Destabilizing networks. *Connections*, 24(3):79–92, 2002.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucléic acids research*, 46(D1):D1074–D1082, 2018.
Steganography of Complex Networks

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020.

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, et al. String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, 47(D1):D607–D613, 2019.

K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. The extreme classification repository: Multi-label datasets and code, 2016. URL [http://manikvarma.org/downloads/XC/XMLRepository.html](http://manikvarma.org/downloads/XC/XMLRepository.html).

Lawrence Bassham, Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Stefan Leigh, M Levenson, M Vangel, Nathanael Heckert, and D Banks. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2010-09-16 2010. URL [https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762).

Shmuel T. Klein and Dana Shapira. On the randomness of compressed data. *Information*, 11(4), 2020. ISSN 2078-2489. doi:10.3390/info11040196. URL [https://www.mdpi.com/2078-2489/11/4/196](https://www.mdpi.com/2078-2489/11/4/196).

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

Milad Taleby Ahvanooey, Qianmu Li, Jun Hou, Ahmed Raza Rajput, and Yini Chen. Modern text hiding, text steganalysis, and applications: a comparative analysis. *Entropy*, 21(4):355, 2019.

Anna D Broido and Aaron Clauset. Scale-free networks are rare. *Nature communications*, 10(1):1–10, 2019.

Michael Haenlein. A social network analysis of customer-level revenue distribution. *Marketing Letters*, 22(1):15–29, 2011. ISSN 09230645, 1573059X. URL [http://www.jstor.org/stable/41488518](http://www.jstor.org/stable/41488518).

Eyal Carmi, Gal Oestreicher-Singer, Uriel Stettner, and Arun Sundararajan. Is Oprah contagious? the depth of diffusion of demand shocks in a product network. *MIS Q.*, 41(1):207–221, 2017.

Justin M Drake, Evan O Paull, Nicholas A Graham, John K Lee, Bryan A Smith, Bjoern Titz, Tanya Stoyanova, Claire M Faltermeier, Vladislav Uzunangelov, Daniel E Carlin, et al. Phosphoproteome integration reveals patient-specific networks in prostate cancer. *Cell*, 166(4):1041–1054, 2016.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi:10.1038/s41586-020-2649-2. URL [https://doi.org/10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2).

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi:10.25080/Majora-92bf1922-00a.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL [https://igraph.org](https://igraph.org).

Scott Griffiths. bitstring, 2020. URL [https://github.com/scott-griffiths/bitstring](https://github.com/scott-griffiths/bitstring). bitstring is a pure Python module designed to help make the creation and analysis of binary data as simple and natural as possible.