electronics

MDPI

Article

Linguistic Secret Sharing via Ambiguous Token Selection for IoT Security

Kai Gao [1], Ji-Hwei Horng [2,*], Ching-Chun Chang [3] and Chin-Chen Chang [1,*]

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 407102, Taiwan; p0968862@o365.fcu.edu.tw
[2] Department of Electrical Engineering, National Quemoy University, Jinning, Kinmen 892009, Taiwan
[3] Information and Communication Security Research Center, Feng Chia University, Taichung 407102, Taiwan; ccc@fcu.edu.tw
* Correspondence: horng@email.nqu.edu.tw (J.-H.H.); ccc@o365.fcu.edu.tw (C.-C.C.)

Abstract: The proliferation of Internet of Things (IoT) devices has introduced significant security challenges, including weak authentication, insufficient data protection, and firmware vulnerabilities. To address these issues, we propose a linguistic secret sharing scheme tailored for IoT applications. This scheme leverages neural networks to embed private data within texts transmitted by IoT devices, using an ambiguous token selection algorithm that maintains the textual integrity of the cover messages. Our approach eliminates the need to share additional information for accurate data extraction while also enhancing security through a secret sharing mechanism. Experimental results demonstrate that the proposed scheme achieves approximately 50% accuracy in detecting steganographic text across two steganalysis networks. Additionally, the generated steganographic text preserves the semantic information of the cover text, evidenced by a BERT score of 0.948. This indicates that the proposed scheme performs well in terms of security.

Keywords: ambiguous token selection; Galois field; linguistic secret sharing; language modeling

Citation: Gao, K.; Horng, J.-H.; Chang, C.-C.; Chang, C.-C. Linguistic Secret Sharing via Ambiguous Token Selection for IoT Security. *Electronics* **2024**, *13*, 4216. https://doi.org/10.3390/electronics13214216

Academic Editor: Yazan Otoum

Received: 29 September 2024
Revised: 21 October 2024
Accepted: 25 October 2024
Published: 27 October 2024

1. Introduction

Steganography conceals a secret message in cover media. A classic scenario illustrating linguistic steganography is the "Prisoner's problem". Consider that Alice and Bob were two inmates in a prison, planning to escape. To facilitate their plan, they decided to exchange secret messages via short notes. However, all message exchanges must be checked by Eve. If Eve successfully detects any concealed message, she retains the authority to terminate any further communication. In this scenario, Alice employed an embedding rule to conceal the secret message within the cover note, which was then sent to Bob under the surveillance of Eve. Upon receiving the note, Bob employed an extraction rule to retrieve the concealed message from the note.

In the rapidly evolving landscape of the Internet of Things (IoT), the surge of connected devices brings new security challenges. Ensuring secure and confidential data transmission is crucial in this context. Traditional encryption methods may be unsuitable due to the limited computational and storage capacities of IoT devices. Thus, steganography emerges as a potential alternative for protecting information. With the IoT now deeply embedded in everyday life, securing user privacy, data integrity, and infrastructure is essential [1,2]. Despite advances in the IoT, security and privacy concerns remain major hurdles. Innovative solutions like steganography are vital for building robust security frameworks and supporting the IoT's sustainable growth.

As time has advanced, except for text, various cover media have been employed in steganography including images [3,4], audio [5], 3D mesh models [6], and videos [7]. However, due to the development of deep neural networks, linguistic steganography

attracts much attention again [8–27]. Linguistic steganography can be primarily categorized into two types based on whether the steganographic text maintains the semantics of the cover text, namely generation-based linguistic steganography [8–19] (GLS) and modification-based linguistic steganography [20–27] (MLS).

GLS primarily embeds secret bits during the generation of high-quality steganographic text, leveraging neural network-based language models such as RNNs [8], LSTM [9–11], auto-encoders [12], and GANs [13]. This flexibility in word selection for each position based on secret data is especially beneficial in the IoT context, where devices generate large volumes of text that can serve as cover messages. However, this mechanism results in significant differences in features between the cover text, generated through semantic predictions from the language model, and the steganographic text. To address these discrepancies, recent advancements have focused on enhancing both the security and quality of steganographic text. Zhou et al. [14] utilized a GAN to overcome issues such as exposure bias and embedding deviation, which can compromise the security of steganography. This scheme dynamically adjusts probability distributions to maintain diversity and embeds information during the model's training process to enhance security. Yan et al. [15] proposed a secure token selection principle to further improve security and resolve ambiguity, which ensures the sum of selected token probabilities correlates with statistical imperceptibility. Similarly, Zhang et al. [16] proposed an adaptive dynamic grouping scheme to embed secret information by grouping tokens based on their probabilities from a language model recursively, addressing the statistical differences between the probability distributions of steganographic text and natural text. To maintain the semantic expression of the generated steganographic text, some schemes [17–19] try to incorporate semantic constraints. Yang et al. [17] utilized context as the constraint, aiming to preserve a strong semantic correlation between the steganographic text and the cover text. Wang et al. [18] encoded secret data and refined the generated steganographic text through multiple rounds, improving the text's quality and reducing the negative effects of steganographic encoding. Wang et al. [19] enhanced the controllability of steganography generation by analyzing the discourse features of the cover, which serve as the inputs to the steganography generator. However, since these GLS schemes operate at the word level, they can still easily lead to significant distortion of the local semantics. Furthermore, state-of-the-art steganalysis models use deep neural networks to extract multidimensional statistical features, enhancing their ability to detect steganographic text. These models integrate temporal features derived from spatial features [28] or continuous text sequences [29], improving their effectiveness in detecting steganographic text. This significantly reduces the inherent embedding capacity of GLS.

MLS primarily embeds secret bits by modifying part of the cover text at the word [20–22], phrase [23,24], and sentence levels [25–27]. Word- or phrase-level MLS schemes generally utilize synonym substitutions to embed the secret. For instance, Chang and Clark [20] employed the Google n-gram corpus to verify the contextual applicability of synonyms, ensuring more accurate and contextually appropriate substitutions. Xiang et al. [21] introduced a method that combines arithmetic coding with synonym substitution, analyzing synonyms based on their relative frequencies and quantizing them into a binary sequence. This sequence is then compressed using adaptive binary arithmetic coding to create space for additional data. The compressed data, along with the secret data, are embedded into the text using synonym substitutions. Dai and Cai [22] proposed a steganographic technique using a patient Huffman algorithm, which generates text by combining ciphertext-driven token selection with language model-based sampling.

At the phrase level, Wilson and Ker [23] introduced distortion measures specific to linguistic steganography, which helps determine the optimal embedding strategy, balancing text quality with embedding capacity. Qiang et al. [24] emphasized the importance of preserving meaning by using paraphrase modeling to generate suitable substitute candidates.

At the sentence level, MLS schemes typically convert sentences into alternative forms that maintain the original meaning, using techniques such as syntactic analysis [25] and sentence translation [26,27]. For instance, Xiang et al. [25] developed a syntax-controlled

paraphrase generation model to automatically modify the expression of cover text, using a syntactic bins coding strategy for embedding secret information within the generated syntactic space. Yang et al. [26] pivoted the text between two languages and embedded secret data using a semantic-aware encoding strategy, which modifies the expression of the text while maintaining its original meaning, thereby allowing for a larger payload. Ding et al. [27] introduced a scheme that integrates semantic fusion and language model reference units into a neural machine translation model. This approach generates translations that embed secret messages while preserving the text's semantics and context. Additionally, they implemented a new encoding scheme that combines arithmetic coding with a waiting mechanism, enhancing embedding capacity without compromising semantic consistency.

In the context of the IoT, where device-generated texts often follow specific structures or patterns, MLS can be used to subtly modify these texts to embed secret data. However, these MLS schemes suffer from a low embedding capacity.

In summary, GLS schemes offer impressive embedding capacity but may introduce semantic ambiguity and increase suspicion by steganalysis. While MLS schemes successfully maintain the overall semantics, their limited embedding capacity poses a significant challenge. Moreover, previous linguistic steganography schemes typically involve encrypting the secret data, recording additional information for recovery, and managing the secret key. Specifically, in GLS, to ensure that the receiver generates the same text as the sender, the sender must transmit the latent space vectors used to initialize the model, ensuring consistent text output. Furthermore, both GLS and MLS require the encryption of secret data, necessitating the transmission of a secret key for decryption, which can be impractical in certain applications.

In order to solve the above problems, we propose a novel linguistic secret sharing scheme for IoT security. In our scheme, only the most ambiguous word in each sentence is substituted to embed secret data, thereby preserving the semantic integrity of each sentence. Moreover, the receiver can also easily identify this specific word during the extraction process. Additionally, we employ a secret sharing mechanism to encrypt secret data. Instead of relying on a secret key, secret sharing distributes the secret into multiple shares, ensuring that a single share alone cannot restore the original secret. Our contributions are summarized as follows:

1. We propose a token selection algorithm that enables both the sender and the receiver to identify the same most ambiguous word in each sentence.
2. Data embedding and extraction can be performed without the need to share any secret key.
3. The proposed scheme maintains the semantic coherence of the steganographic text.
4. Secret sharing over a Galois field is first introduced to linguistic steganography.

2. Preliminary Work

We first review the concept of $(k,\ n)$-threshold secret sharing, which serves as the foundational framework of the proposed scheme for ensuring security. Then, a masked language model called "RoBERTa" is introduced, which will be modified to embed data in our scheme.

2.1. $(k,\ n)$-Threshold Secret Sharing over $GF(2^m)$

In 1979, Shamir [30] proposed a cryptographic algorithm known as $(k,\ n)$-threshold secret sharing, which improves security by distributing the secret to multiple participants. The concept of (k, n)-threshold secret sharing is to divide a secret into n shares and distribute to n participants. The original secret can only be reconstructed when at least k shares are available in recombination.

The fundamental operation of $(k,\ n)$-threshold secret sharing over $GF(2^m)$ involves the construction of a set of polynomial equations over a Galois field, provided $x_1, x_2, \ldots, x_n \in GF(2^m)$ denote n distinct binary polynomials. Given a secret $s \in GF(2^m)$, we can construct a $k-1$ degree polynomial as follows:

$$f(x) = \left[s \oplus (a_1 \odot x) \oplus \left(a_2 \odot x^2\right) \oplus \ldots \oplus \left(a_{k-1} \odot x^{k-1}\right)\right] \bmod p(x), \tag{1}$$

where "$\oplus$" represents the exclusive or operation; "$\odot$" represents the Galois field multiplication in $GF(2^m)$, where each element is an m-bit binary polynomial, and the multiplication follows the rules of the Galois field; and $\{a_1, a_2, \ldots, a_{k-1}\} \in GF(2^m)$ and $p(x)$ represent a group of random binary polynomials and an irreducible polynomial, respectively. After secret share generation, each participant holds a share $(x_i, f(x_i))$, where x_i is the owner's private key and $f(x_i)$ is the corresponding secret share.

When at least k participants contribute their private keys and secret shares, the coefficients $a_1, a_2, \ldots a_{k-1}$ and secret s can be restored via the Lagrange interpolating polynomial.

$$f(x) = \sum_{q=1}^{k} \{f(x_q) \odot \prod_{\substack{1 \le w \le k \\ w \ne q}} \left[(x \oplus x_w) \odot \left(x_q \oplus x_w\right)^{-1}\right]\} \bmod p(x). \tag{2}$$

$$s = \sum_{q=1}^{k} \{f(x_q) \odot \prod_{\substack{1 \le w \le k \\ w \ne q}} \left[x_w \odot \left(x_q \oplus x_w\right)^{-1}\right]\} \bmod p(x), \tag{3}$$

where $(\cdot)^{-1}$ represents the multiplicative inverse in the Galois field. For any element $a \in GF(2^m)$, the inverse is an element $b \in GF(2^m)$ such that $a \odot b = 1$ under the field's multiplication. When at least k participants contribute their private keys and secret shares, the coefficients $a_1, a_2, \ldots a_{k-1}$ and secret s can be restored via the Lagrange interpolating polynomial.

To provide a clearer understanding of the proposed scheme, we present a mathematical example. Assume that $k = 2$, $n = 3$; the $k - 1$ degree polynomial can be constructed by

$$f(x_i) = s \oplus (a_1 \odot x_i) \bmod p(x). \tag{4}$$

Let $s = 111$, $a_1 = 101$, $x_1 = 001$, $x_2 = 010$, $x_3 = 011$, and $p(x) = x^3 + x + 1$. Three shares can be calculated as

$$f(x_1) = s \oplus (a_1 \odot x_1) \bmod p(x) = (111 \oplus 101 \odot 001) \bmod 1011 = 010, \tag{5}$$

$$f(x_2) = s \oplus (a_1 \odot x_2) \bmod p(x) = (111 \oplus 101 \odot 010) \bmod 1011 = 110, \tag{6}$$

$$f(x_3) = s \oplus (a_1 \odot x_3) \bmod p(x) = (111 \oplus 101 \odot 011) \bmod 1011 = 011. \tag{7}$$

By collecting any two out of three shares, s can be recovered by Equation (3) as

$$\begin{aligned} s &= \left\{f(x_1) \odot \left[x_2 \odot (x_1 \oplus x_2)^{-1}\right] \oplus f(x_2) \odot \left[x_1 \odot (x_2 \oplus x_1)^{-1}\right]\right\} \\ &= (010 \odot (010 \odot 110) \oplus 110 \odot (001 \odot 110)) = 101 \oplus 010 = 111. \end{aligned} \tag{8}$$

Similarly, a_1 can also be recovered by Equation (2).

2.2. RoBERTa-Masked Language Modeling

RoBERTa (Robustly Optimized Bidirectional Encoder Representations from Transformers Pre-training Approach) [31] is an NLP model that is built upon a variant of the transformer architecture [32] and is specifically designed to handle language comprehension tasks. Its layout is depicted in Figure 1. The key objective of the RoBERTa model is to improve the pre-training process in order to effectively leverage large-scale unlabeled text data for model training. Compared to the earlier BERT [33], RoBERTa incorporates a larger model size and a longer training time while introducing several technical improvements such as dynamic masks, continuous text paragraph training, and larger batch sizes.

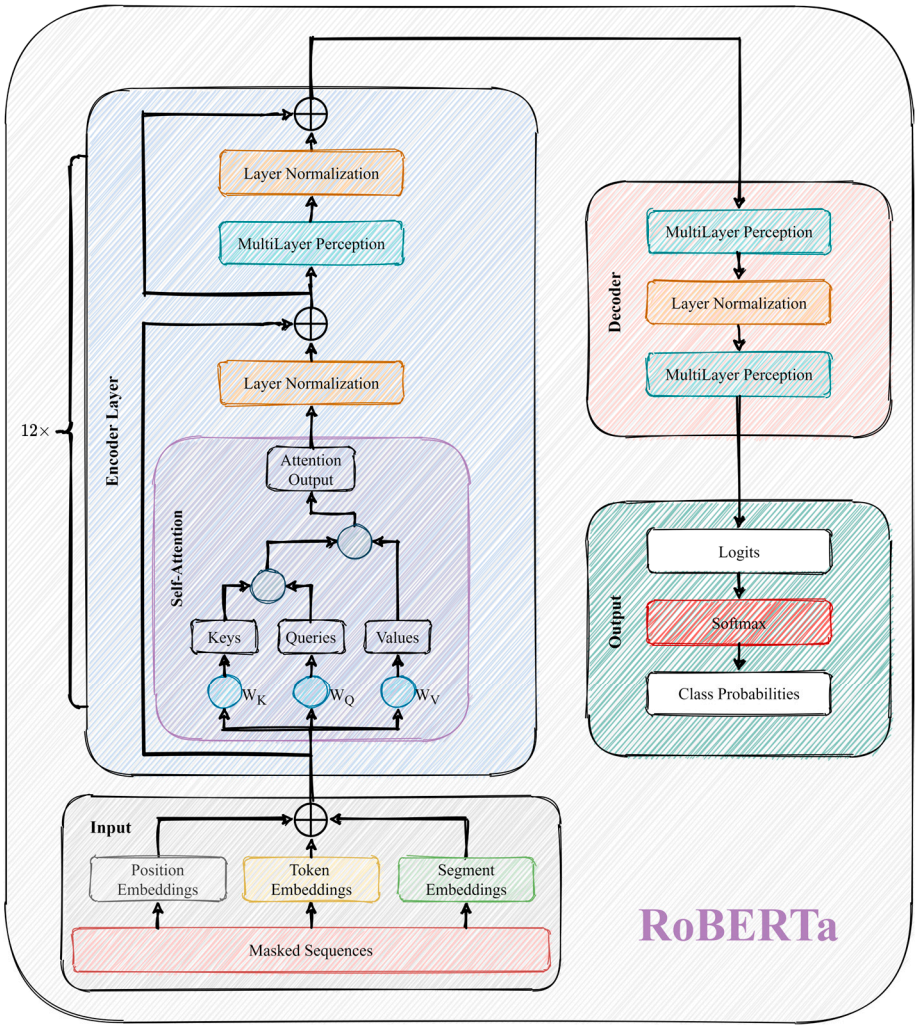


Figure 1. Architecture of RoBERTa.

Masked language modeling is the fundamental task of RoBERTa, aiming to predict a masked word based on the context of surrounding words. Tokenization is an initial step, where a text sequence is split into tokens. An input sentence containing one or more mask tokens is fed into the model, which estimates the probability distribution of each mask token across the entire vocabulary. The predicted results can then be used to replace the masked tokens for data embedding.

3. Proposed Linguistic Secret Sharing

Consider a scenario in which a company produces advanced equipment that is restricted for use in certain areas or by specific companies. This equipment has an associated secret code to activate it. The equipment is delivered by a logistics company, while the secret code is distributed among multiple participants with a secret sharing scheme to ensure authorized usage of the equipment.

As illustrated in Figure 2, during the share generation stage, the secret code is transformed into n distinct shares using a polynomial secret sharing technique. These shares are then concealed within the regular messages intended for the participants with an open-source pre-trained model. These steganographic messages are then transmitted over the IoT devices of the participants.

In the secret recovery stage, the system enables any k authorized personnel (where $k < n$) to collaboratively extract the activation code of the equipment. By applying the same token selection principle and data embedding rule, participants can extract the secret shares

from the messages on their own IoT devices and combine them back into the activation code. This mechanism ensures authorized usage of the protected equipment by preventing unauthorized personnel or an insufficient number of participants from activating it.

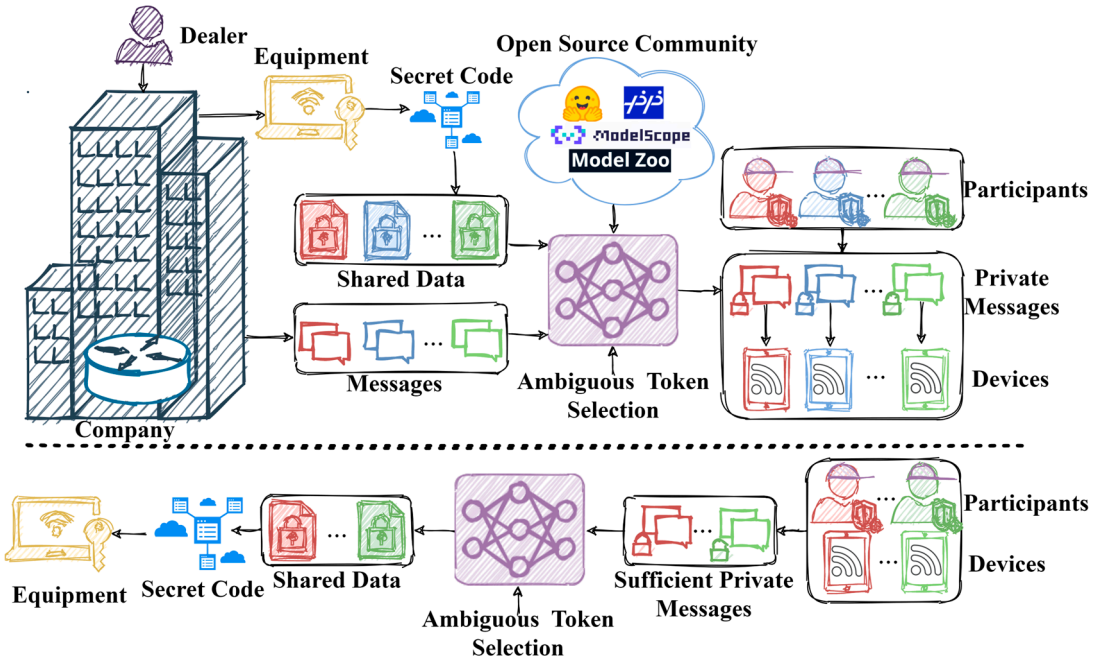


Figure 2. Flowchart of proposed scheme.

3.1. Text Share Generation

Theoretically, the proposed scheme can employ any type of text as its carrier and the three carrier texts can be completely different in content. However, in our case, we choose to utilize texts generated by deep learning-based models. By utilizing generated texts, their uniqueness can prevent attackers from comparing the texts to existing content on social media or the Internet in order to decipher the secret. Additionally, the generated texts offer greater control to tailor the contents according to the user's requirements. Users can generate texts suitable for linguistic secret sharing by providing appropriate prompts. For implementing the text generation stage, GPT-4 [34] is applied as the text generator. It is worth mentioning that we do not specifically address grammar or syntax, so the generated text is used directly without modification.

3.2. Token Selection Algorithm and Data Embedding Rule

In the proposed scheme, an ambiguous token is selected and masked for each sentence first. The masked sentence is fed into a token predictor, which gives the prediction results for each masked token. Finally, the masked token is replaced with one of the prediction results according to the data embedding rule.

To select the target token, assume a sentence consists of l tokens denoted as $T = (t_1, t_2, \ldots, t_l)$. Each token t_i can be masked individually and predicted using masked language modeling. For a token belonging to the vocabulary pool $\mathcal{V}$, the initial candidate pool for prediction is denoted as $\left[c_1, c_2, \ldots, c_{|\mathcal{V}|}\right]$ with corresponding probabilities $\left[p_1, p_2, \ldots, p_{|\mathcal{V}|}\right]$, where $\sum_{j=1}^{|\mathcal{V}|} p_j = 1$. To quantify the ambiguity of a token's prediction, we define the probability difference indicator as

$$\mathcal{D}_t = \sum_{j=1}^{m-1} \left(p_{\sigma(j)} - p_{\sigma(j+1)}\right), \tag{9}$$

where $p_{\sigma(j)}$ represents the j-th greatest prediction probability. A lower $\mathcal{D}_t$ value indicates that the top m prediction probabilities for this token are close together, making it harder for the model to confidently predict the token. Altering this token does not significantly affect the overall semantics of the sentence. The Algorithm 1 for ambiguous token selection is given as follows:

Algorithm 1: Ambiguous Token Selection

Input: A sentence $T = (t_1, t_2, \ldots, t_l)$.
Load: Token predictor $\mathcal{P}$.
Output: ambiguous token $\boldsymbol{t}$.
1: **for** $i = 1, 2, \ldots, l$ **do**

$$\boldsymbol{t_i} = < \text{mask} >$$
$$T' = (t_1, t_2, \ldots, \boldsymbol{t_i}, \ldots t_l)$$
$$\left[p_1, p_2, \ldots, p_{|m|}\right] = \mathcal{P}(T')$$

compute $\mathcal{D}_{t_i}$.
2: **end for**
$c = \underset{i}{\text{argmin}}(\mathcal{D}_{t_i})$
3: **return** $\boldsymbol{t} = t_c$

This algorithm ensures that the token with the highest prediction ambiguity (i.e., the smallest $\mathcal{D}_t$) is selected for data embedding. Since this token is the most difficult for the model to predict, replacing it with one of the top m candidates will introduce minimal semantic distortion, thus allowing for the embedding of $\log_2 m$ bits of secret data.

After selecting the ambiguous token $\boldsymbol{t}$, its top m prediction candidates $[c_1, c_2, \ldots, c_m]$ can be mapped into m different binary codes. Therefore $\log_2 m$ bits of data can be embedded into the sentence T by replacing $\boldsymbol{t}$ with one of its top m prediction candidates. To illustrate the token selection algorithm and data embedding rule, an example is given in Figure 3. Suppose the cover text is "I am very happy." and m is set to four; the predictor can provide the top four prediction results with the highest probability for each masked token. By masking each token individually and predicting, we can obtain the prediction results along with their respective probabilities. After calculating the probability difference for each token's prediction results, we select the token with the lowest value. In this case, "very" is selected as the ambiguous token because it has the lowest probability difference indicator compared to the other tokens. The top four prediction candidates for this token are then mapped in descending order of probability to represent the secret data (i.e., "very": 00; "so": 01; "really": 10; and "extremely": 11). Therefore, $\log_2 4 = 2$ bits of data can be embedded by replacing "very" with one of the top four candidates.

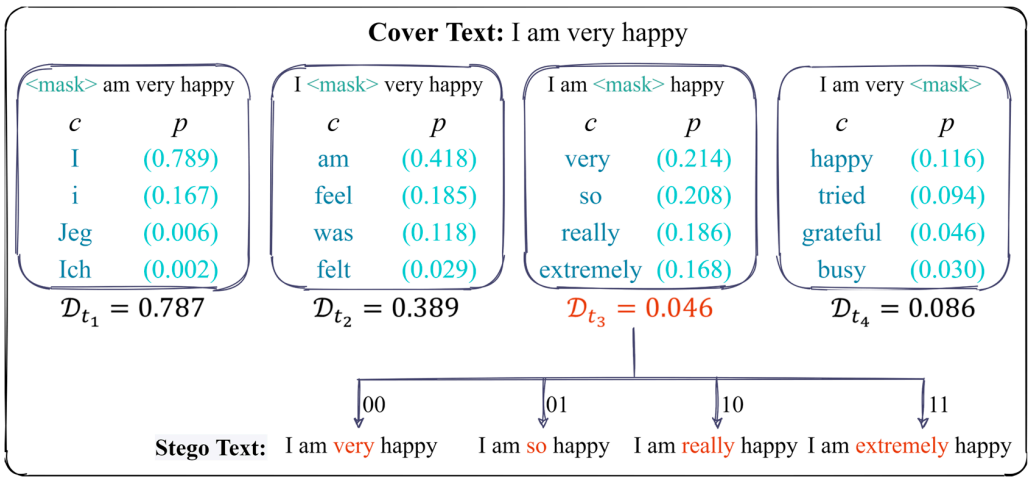


Figure 3. An example of token selection and data embedding.

Note that when applying the token selection and data embedding rule to a text, as shown in Figure 4, the modified sentence is considered as the preceding context for the current sentence. This ensures that the embedding process maintains coherence and consistency throughout the entire text.

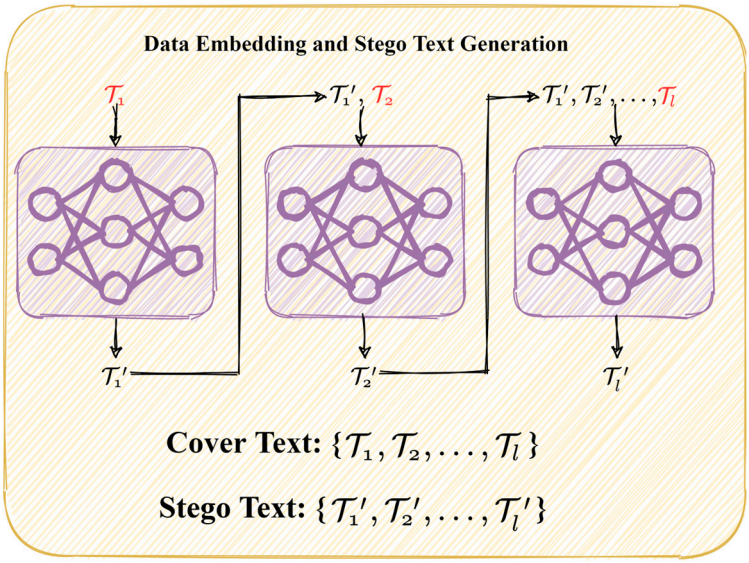


Figure 4. Embedding rule.

3.3. Secret Share Generation

Referring to Figure 2 again, the dealer adopts the polynomial secret sharing over $GF(2^m)$ to distribute the secret data into secret shares and embeds the shares into distinct generated texts correspondingly using the token selection algorithm and the data embedding rule. The procedures are given as follows:

Step 1: Convert the secret data into a sequence of binary segments $\mathcal{S} = \{\mathcal{s}_1, \mathcal{s}_2, \ldots, \mathcal{s}_k\}$, where $\mathcal{s}_i$ is $\log_2 m$ bit in length.

Step 2: Generate n secret shares $e_i^1, e_i^2, \ldots, e_i^n$ using Equation (1) by replacing s and the coefficients $a_1, a_2, \ldots a_{k-1}$ with $\mathcal{s}_1, \mathcal{s}_2, \ldots, \mathcal{s}_k$.

Step 3: Generate n texts $T_1, T_2, \ldots, T_n$ using a text generator with proper prompts.

Step 4: Embed the n secret shares into the n texts correspondingly using the token selection algorithm and the data embedding rule to generate n text shares $T_1', T_2', \ldots, T_n'$.

3.4. Secret Data Recovery

By collecting any k out of n text shares, a combiner can restore the secret data. The procedures are summarized as follows:

Step 1: Collect any k text shares.

Step 2: Split each text into sentences and identify a marked token for each sentence using the token selection algorithm.

Step 3: Retrieve and collect the embedded secret bits from marked tokens according to the data embedding rule.

Step 4: Combine k secret shares to recover the original secret data using Equations (2) and (3).

4. Experimental Results

In this section, we introduce our experimental settings, give a demonstrative example, and evaluate the performance of our scheme in terms of sentiment and semantic analyses.

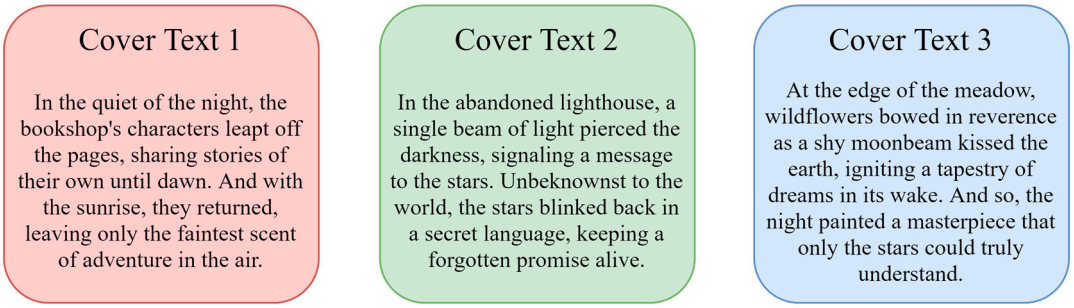
4.1. Experimental Setting

Model: Our experiments make use of the GPT-4 networks, a substantial language model developed by OpenAI for the purpose of generating cover texts. GPT-4 is a state-of-the-art model with a remarkable capacity for comprehending and producing both natural language and code. Furthermore, we employ RoBERTa as our token predictor in this study.

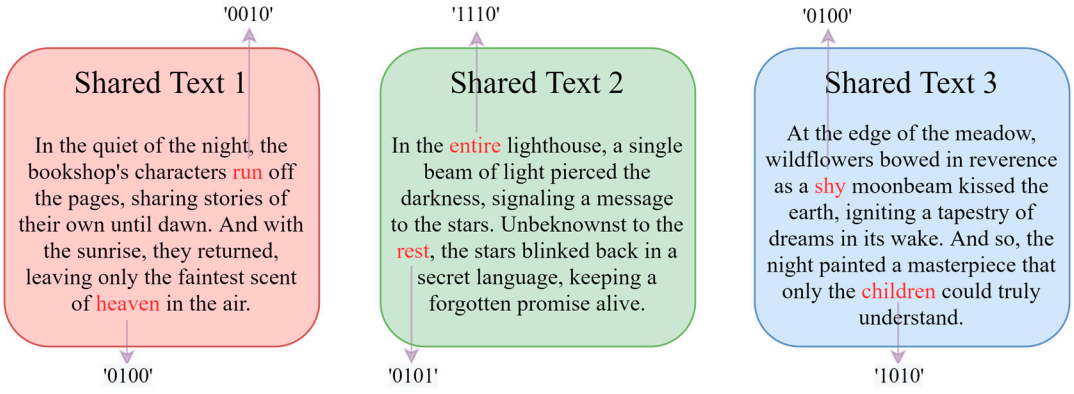
Implementation: Our experiments are implemented using Python 3.8 and rely on PyTorch 1.7.1 as the foundational framework. Acceleration is achieved through the utilization of Nvidia 3090 and CUDA 11.2. The secret messages applied are binary pseudo-random bitstreams.

4.2. Applicability Demonstration

An example of (2, 3)-linguistic secret sharing is provided. Suppose the original secret data are an 8 bit binary random bitstream "11010011"; after secret sharing, three shared bitstreams "00100100", "11100101", and "01001010" are generated. Three cover texts and their corresponding text shares are shown in Figures 5a and 5b, respectively. It is noteworthy that each sentence within the share text is capable of being embedded as 4 bit shared data when the number of candidate pools of each selected token is set to 16. During the secret recovery process, the ambiguous token within each sentence can be easily identified through the proposed token selection algorithm. Subsequently, the shared data can be extracted according to the order of selected tokens in the candidate pool. Following this step, any two out of three shared data can be combined to recover the original secret bitstream.



(**a**) Cover share text.



(**b**) Marked share text.

Figure 5. An example of (2, 3)-linguistic secret sharing.

4.3. Performance Analysis

We evaluate the performance of steganographic text using sentiment and semantic analyses. The sentiment of a text is determined by its emotional nature, where "positive" refers to an optimistic or favorable emotion, while "negative" refers to a pessimistic or

unfavorable emotion. The sentiment of a text can be assessed using a pre-trained BERT-based sentiment classifier [33]. In our experiments, text share 1 (474 words) and text share 2 (248 words) are generated from texts classified with positive (P: 99.84%) and negative (N: 96.54%) emotions, respectively. Two strategies are used to segment the text into sentences; strategy 1 segments the text with periods, while strategy 2 segments the text with punctuation marks, including commas and periods. Thus, the number of resulting sentences for strategy 1 is fewer than that for strategy 2. Data embedding is executed by selecting only one word to replace within each sentence. As shown in Table 1, the steganographic texts generated with different m values successfully preserve the sentiment classification results (CRs) of the cover text.

Table 1. Sentiment classification.

Top m/CR	Text Share 1 (P: 99.84%)		Text Share 2 (N: 96.54%)	
	Strategy 1	**Strategy 2**	**Strategy 1**	**Strategy 2**
8	P: 99.76%	P: 99.26%	N: 95.52%	N: 90.86%
16	P: 99.78%	P: 99.54%	N: 95.45%	N: 87.41%
32	P: 99.71%	P: 99.81%	N: 97.35%	N: 96.45%
64	P: 99.75%	P: 99.59%	N: 96.16%	N: 96.39%
128	P: 99.85%	P: 99.42%	N: 96.76%	N: 99.25%

To validate the preservation of semantic fidelity between the cover text and modified text, we employ BERTScore [35] to analyze the texts. BERTScore captures the deep semantic representations of both the cover text and the modified text. We then use cosine similarity to compare these representations, ensuring that the intrinsic meanings remain consistent between the two texts. As shown in Table 2, the modified text effectively maintains the semantic essence of the cover text.

Table 2. Semantic similarity.

Top m/CS	Text Share 1		Text Share 2	
	Strategy 1	**Strategy 2**	**Strategy 1**	**Strategy 1**
8	0.954	0.935	0.963	0.928
16	0.954	0.935	0.962	0.935
32	0.952	0.933	0.960	0.931
64	0.953	0.934	0.959	0.933
128	0.953	0.932	0.961	0.932

In summary, the modified text generated by the proposed scheme successfully retains the information from the cover text, as only one ambiguous word is replaced in each sentence. Although the text produced using segment strategy 2 exhibits slightly diminished quality compared to that generated through segment strategy 1, this trade-off yields a higher embedding capacity. However, this slight degradation in text fidelity also increases the risk of the text being detected by steganalysis. Therefore, for subsequent experiments, we select strategy 1 with $m = 8$. While the embedding capacity is relatively smaller, the proposed scheme ensures the highest text fidelity, preserving the integrity and meaning of the original cover text.

4.4. Comparison

We further compared the proposed scheme with two state-of-the-art schemes [25,26] using a testing set of 1000 sentences. We set m to eight and employed strategy 1 to execute (2, 3)-threshold linguistic secret sharing. As shown in Table 3, the average embedding capacities (ECs) in bits per word (bpw) for Xiang et al.'s [25] scheme and Yang et al.'s [26] scheme are higher than that of our scheme, with values of 0.333 bpw. However, the proposed scheme exhibits better resistance to steganalysis, with detection accuracies of

0.514 and 0.504 for LSCNN [28] and TSRNN [29] models, respectively, which is lower than those of the compared schemes. Additionally, our scheme demonstrates a BERTScore of 0.948, indicating better preservation of the semantic information of the cover text.

Table 3. Characteristic comparisons.

Schemes	EC	LSCNN	TSRNN	BERTScore	Tolerance	Metadata
[25]	0.333	0.526	0.513	0.676	No	Yes
[26]	0.333	0.594	0.559	0.825	No	Yes
Proposed	0.162	0.514	0.504	0.948	Yes	No

Although the EC could be enhanced by selecting a larger m or by applying strategy 2, this would inevitably introduce a trade-off between security and text fidelity, potentially distorting the naturalness of the generated text and making it more susceptible to detection or steganalysis. Moreover, our scheme is tolerant of data loss or user failure and requires no additional information or secret key management (metadata), which is more practical than the compared schemes.

4.5. Theoretical Analysis for IoT Implementation

Due to limited access to large-scale real-world IoT environments, our study currently focuses on theoretical analysis and simulations. The proposed scheme leverages pre-trained models such as GPT-4 and RoBERTa, which can be deployed without additional training, substantially lowering the implementation threshold. However, the computational complexity of these models presents significant challenges in IoT contexts. RoBERTa's complexity is $O(n^2 d)$, where n is the input sequence length and d is the model dimension. Assuming an input length of 128 and approximately 300 million parameters, the computational requirement is about 4.9×10^{12} FLOPs (floating point operations). In comparison, GPT-4, with around 175 billion parameters, requires approximately 2.9×10^{15} FLOPs to generate a sentence of length 128. This scale of computation is suitable for high-performance cloud computing environments but may impact the real-time responsiveness of IoT devices. In practical IoT networks, participants could pre-agree on specific model versions for deployment in local or cloud-based environments. For IoT devices with constrained computational capabilities, we recommend a cloud deployment strategy to mitigate these computational demands and ensure efficient operation.

4.6. Limitations

Although the proposed scheme improves the security of secret data through a secret sharing mechanism, which is unconditionally secure [36], several limitations still need to be addressed. First, while secret data can be recovered without the need for additional data, the scheme lacks an authentication mechanism to detect cheating by individual participants. Additionally, secret sharing requires more storage space because the secret data must be distributed across multiple shares. Moreover, the proposed ambiguous token selection algorithm imposes constraints on the embedding capacity of each sentence, potentially limiting the overall data throughput. Finally, the computational overhead introduced by neural network operations could become a challenge for resource-constrained IoT devices. Balancing the trade-off between enhanced security and computational efficiency remains a critical area for future research and optimization. Our future work will focus on addressing these issues.

5. Conclusions

In this paper, we introduce a novel linguistic secret sharing scheme via an ambiguous token selection algorithm for IoT security. Unlike previous schemes, the proposed scheme does not require sharing additional information for correct data extraction. Moreover, we employ a secret sharing mechanism to improve security. Experimental results show that

the steganographic text generated by the proposed scheme can effectively preserve the sentiment and semantic information of the cover text. In the future, we will explore how to improve the data embedding capacity of the proposed scheme.

Author Contributions: Conceptualization, K.G.; methodology, K.G.; software, K.G.; validation, K.G., J.-H.H., and C.-C.C. (Ching-Chun Chang); formal analysis, K.G. and J.-H.H.; investigation, K.G., J.-H.H., and C.-C.C. (Ching-Chun Chang); resources, C.-C.C. (Chin-Chen Chang); data curation, K.G.; writing—original draft preparation, K.G.; writing—review and editing, J.-H.H.; visualization, K.G.; supervision, C.-C.C. (Chin-Chen Chang); project administration, C.-C.C. (Chin-Chen Chang). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All data can be downloaded from https://huggingface.co, accessed on 25 November 2016.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Litoussi, M.; Kannouf, N.; Makkaoui, K.E.; Ezzati, A.; Fartitchou, M. IoT Security: Challenges and Countermeasures. *Procedia Comput. Sci.* **2020**, *177*, 503–508. [CrossRef]
2. Mohanty, J.; Mishra, S.; Patra, S.; Pati, B.; Panigrahi, C.R. IoT Security, Challenges, and Solutions: A Review. In *Progress in Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2019*; Springer: Singapore, 2021; Volume 2, pp. 493–504.
3. Gao, K.; Chang, C.-C.; Horng, J.-H.; Echizen, I. Steganographic Secret Sharing via AI-generated Photorealistic Images. *EURASIP J. Wirel. Commun. Netw.* **2022**, *2022*, 1–23. [CrossRef]
4. Dong, L.; Zhou, J.T.; Sun, W.W.; Yan, D.Q.; Wang, R.D. First Steps Toward Concealing the Traces Left by Reversible Image Data Hiding. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *67*, 951–955. [CrossRef]
5. Dutta, H.; Das, R.K.; Nandi, S.; Mahadeva Prasanna, S.R. An Overview of Digital Audio Steganography. *IETE Tech. Rev.* **2020**, *97*, 632–650. [CrossRef]
6. Gao, K.; Horng, J.-H.; Chang, C.-C. Reversible Data Hiding for Encrypted 3D Mesh Models with Secret Sharing over Galois Field. *IEEE Trans. Multimed.* **2023**, *26*, 5499–5510. [CrossRef]
7. Zhang, J.; Ho, A.T.S.; Qiu, G.; Marziliano, P. Robust Video Watermarking of H.264/AVC. *IEEE Trans. Circuits Syst. II Express Briefs* **2007**, *54*, 205–209. [CrossRef]
8. Yang, Z.; Guo, X.; Chen, Z.; Huang, Y.; Zhang, Y. RNNStega: Linguistic Steganography Based on Recurrent Neural Networks. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1280–1295. [CrossRef]
9. Fang, T.; Jaggi, M.; Argyraki, K.J. Generating Steganographic Text with LSTMs. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 100–106.
10. Kang, H.; Wu, H.; Zhang, X. Generative Text Steganography Based on LSTM Network and Attention Mechanism with Keywords. *Electron. Imaging* **2020**, *32*, 1–8. [CrossRef]
11. Guo, Y.; Wu, H.; Zhang, X. Steganographic Visual Story with Mutualperceived Joint Attention. *EURASIP J. Image Video Process.* **2021**, *2021*, 1–14. [CrossRef]
12. Yang, Z.; Zhang, S.; Hu, Y.; Hu, Z.; Huang, Y. VAE-Stega: Linguistic Steganography Based on Variational Auto-Encoder. *IEEE Trans. Inf. Forensics Secur.* **2020**, *16*, 880–895. [CrossRef]
13. Yang, Z.; Wei, N.; Liu, Q.; Huang, Y.; Zhang, Y. GAN-TSTEGA: Text Steganography Based on Generative Adversarial Networks. In Proceedings of the 18th International Workshop on Digital-Forensics and Watermarking, Chengdu, China, 2–4 November 2019; pp. 18–31.
14. Zhou, X.J.; Peng, W.L.; Yang, B.Y.; Wen, J.; Xue, Y.M.; Zhong, P. Linguistic Steganography Based on Adaptive Probability Distribution. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 2982–2997. [CrossRef]
15. Yan, R.Y.; Yang, Y.T.; Song, T. A Secure and Disambiguating Approach for Generative Linguistic Steganography. *IEEE Signal Process. Lett.* **2023**, *30*, 1047–1051. [CrossRef]
16. Zhang, S.; Yang, Z.; Yang, J.; Huang, Y. Provably Secure Generative Linguistic Steganography. *arXiv* **2021**, arXiv:2106.02011.
17. Yang, Z.; Xiang, L.; Zhang, S.; Sun, X.; Huang, Y. Linguistic Generative Steganography with Enhanced Cognitive-imperceptibility. *IEEE Signal Process. Lett.* **2021**, *28*, 409–413. [CrossRef]
18. Wang, R.; Xiang, L.Y.; Liu, Y.F.; Yang, C.F. PNG-Stega: Progressive Non-Autoregressive Generative Linguistic Steganography. *IEEE Signal Process. Lett.* **2023**, *30*, 528–532. [CrossRef]
19. Wang, Y.; Song, R.; Zhang, R.; Liu, J.; Li, L. LLsM: Generative Linguistic Steganography with Large Language Model. *arXiv* **2024**, arXiv:2401.15656.
20. Chang, C.-Y.; Clark, S. Practical Linguistic Steganography Using Contextual Synonym Substitution and a Novel Vertex Coding Method. *Comput. Linguist.* **2014**, *40*, 403–448. [CrossRef]

21. Xiang, L.; Li, Y.; Hao, W.; Yang, P.; Shen, X. Reversible Natural Language Watermarking Using Synonym Substitution and Arithmetic Coding. *Comput. Mater. Contin.* **2018**, *55*, 541–559.
22. Dai, F.Z.; Cai, Z. Towards Near-Imperceptible Steganographic Text. *arXiv* **2019**, arXiv:1907.06679.
23. Wilson, A.; Ker, A.D. Avoiding Detection on Twitter: Embedding Strategies for Linguistic Steganography. *Electron. Imaging* **2016**, *28*, 1–9. [CrossRef]
24. Qiang, J.; Zhu, S.; Li, Y.; Zhu, Y.; Yuan, Y.; Wu, X. Natural Language Watermarking via Paraphraser-Based Lexical Substitution. *Artif. Intell.* **2023**, *317*, 103859. [CrossRef]
25. Xiang, L.Y.; Ou, C.F.; Zeng, D.J. Linguistic Steganography: Hiding Information in Syntax Space. *IEEE Signal Process. Lett.* **2023**, *31*, 261–265. [CrossRef]
26. Yang, T.; Wu, H.; Yi, B.; Feng, G.; Zhang, X. Semantic-preserving Linguistic Steganography by Pivot Translation and Semantic-aware Bins Coding. *IEEE Trans. Dependable Secur. Comput.* **2024**, *21*, 139–152. [CrossRef]
27. Ding, C.; Fu, Z.; Yang, Z.; Yu, Q.; Li, D.; Huang, Y. Context-Aware Linguistic Steganography Model Based on Neural Machine Translation. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2023**, *32*, 868–878. [CrossRef]
28. Wen, J.; Zhou, X.; Zhong, P.; Xue, Y. Convolutional Neural Network Based Text Steganalysis. *IEEE Signal Process. Lett.* **2019**, *26*, 460–464. [CrossRef]
29. Yang, Z.; Wang, K.; Li, J.; Huang, Y.; Zhang, Y.-J. TS-RNN: Text Steganalysis Based on Recurrent Neural Networks. *IEEE Signal Process. Lett.* **2019**, *26*, 1743–1747. [CrossRef]
30. Shamir, A. How to Share a Secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
31. Liu, Y.H.; Ott, M.; Goyal, N.; Du, J.F.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
32. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All You Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
33. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
34. OpenAI. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774.
35. Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K.Q.; Artzi, Y. BERTScore: Evaluating text generation with BERT. *arXiv* **2019**, arXiv:1904.09675.
36. Rashmi, K.V.; Shah, N.B.; Ramchandran, K.; Kumar, P.V. Information-theoretically Secure Erasure Codes for Distributed Storage. *IEEE Trans. Inf. Theory* **2018**, *64*, 1621–1646. [CrossRef]