


An Ambiguous Coding Scheme for Selective Encryption of High Entropy Volumes

M. Oğuzhan Külekci¹

Informatics Institute, Istanbul Technical University

Istanbul, Turkey

kulekci@itu.edu.tr

 <https://orcid.org/0000-0002-4583-6261>

Abstract

This study concentrates on the security of high-entropy volumes, where entropy-encoded multimedia files or compressed text sequences are the most typical sources. We consider a system in which the cost of encryption is hefty in terms of some metric (e.g., time, memory, energy, or bandwidth), and thus, creates a bottleneck. With the aim of reducing the encryption cost on such a system, we propose a data coding scheme to achieve the data security by encrypting significantly less data than the original size without sacrifice in secrecy. The main idea of the proposed technique is to represent the input sequence by *not* uniquely-decodable codewords. The proposed coding scheme splits a given input into two partitions as the *payload*, which consists of the ambiguous codeword sequence, and the *disambiguation information*, which is the necessary knowledge to properly decode the payload. Under the assumed condition that the input data is the output of an entropy-encoder, and thus, on ideal case independently and identically distributed, the payload occupies $\approx \frac{(d-2)}{d}$, and the disambiguation information takes $\approx \frac{2}{d}$ of the encoded stream, where $d > 2$ denotes a chosen parameter typically between 6 to 20. We propose to encrypt the payload and keep the disambiguation information in plain to reduce the amount of data to be encrypted, where recursive representation of the payload with the proposed coding can decrease the to-be-encrypted volume further. When $2 \cdot 2^d \leq n \leq \tau \cdot d \cdot 2^d$, for $\tau = \frac{d-1.44}{2}$, we show that the contraction of the possible message space 2^n due to the public disambiguation information is accommodated by keeping the codeword set secret. We discuss possible applications of the proposed scheme in practice.

2012 ACM Subject Classification Information systems → Data encryption, Information systems → Multimedia databases, Mathematics of computing → Combinatorics, Mathematics of computing → Coding theory, Security and privacy → Management and querying of encrypted data

Keywords and phrases Non-prefix-free codes, selective encryption, massive data security, multimedia data security, high-entropy data security, source coding, security in resource-limited environments

Digital Object Identifier 10.4230/LIPIcs.SEA.2018.7

1 Introduction

Achieving security of massive data volumes with less encryption makes sense on platforms where the cost of encryption defines a bottleneck as being heavy according to some metrics, e.g., time, memory, energy, or bandwidth. For example, let us assume a system at which the items in a queue are waiting for the encryption/decryption unit to get processed, and consequently delays occur. One simple solution might be to increase the number of the

¹ This work has been supported by TUBITAK-ARDEB-1001 program grant number 117E865.



© Muhammed Oğuzhan Külekci;

licensed under Creative Commons License CC-BY

17th International Symposium on Experimental Algorithms (SEA 2018).

Editor: Gianlorenzo D'Angelo; Article No. 7; pp. 7:1–7:13

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

serving units, but on the other hand usually the congestion only appears at certain times, and the expense of the additional unit may not be feasible. While waiting on the queue, the items can be encoded such that the amount of data to-be-encrypted is reduced, and the items are processed more quickly in the system. Such a reduction can simply improve the throughput of a security pipeline without a need to upgrade the infrastructure.

Similarly, in battery-constrained environments such as mobile devices [9], sensor networks [3], or unmanned aerial vehicles [18], performing less encryption may also help to increase the battery life. It had been shown that symmetric security algorithms roughly doubles the energy consumption of normal operation in those environments, and asymmetric security algorithms increase the energy usage per bit in order of magnitudes (around 5 fold) [14].

Previously, selective encryption schemes [12] have been proposed to reduce the encryption load, particularly on transmission of video/image files [19, 11, 8]. In selective encryption, segments of the data, which are assumed to include important information, e.g., the I-frames in a video stream, are encrypted, while rest of the data is kept plain. We introduce an alternative approach to reduce the amount of encryption required to secure a source data. As opposed to the partial security provided by the selective encryption schemes, we aim to provide the security of the whole data by benefiting from the intrinsic ambiguity of non-prefix-free (NPF) coding.

In NPF coding a codeword may be a prefix of some others, and thus, the nice self-delimiting property of the prefix-free schemes [2] does not apply. Therefore, the codeword boundaries on the encoded stream should be explicitly specified for correct decoding. In other words, the disambiguation information required to decode an NPF codewords stream is the identification of the codeword boundaries on that sequence.

The NPF coding has not been addressed much in the literature except a few studies [4, 10, 1] due to that unique decodability problem, which limits, if not totally removes, its possible usage in practical applications, particularly in data compression. However, we consider that this lack of unique decodability in NPF coding may provide us an interesting opportunity in terms of security. It is noteworthy that the hardness of decoding an encoded data without the knowledge of the used codeword set had been addressed as early as in 1979 [15], and later by others [6, 7]. More recently, non-prefix-free codes have also been mentioned [13] in that sense.

The main idea of the proposed technique here is to represent the input sequence by *not* uniquely-decodable codewords, which can be summarized as follows. We process the n bit long input bit sequence in blocks of d bits according to a predetermined d parameter such that $d \cdot 2^d \leq n$. Due to some limitations that will be described in the paper, typically d is expected to be between 6 and 20. We create 2^d non-prefix codewords by using a *secret* permutation of the numbers $[1 \dots 2^d]$, and then replace every d -bits long symbol in the input with its corresponding NPF codeword of varying bit-length in between 1 to d . We call the resulting bit stream the payload since it includes the actual information of the source. This sequence is not decodable without the codeword boundaries. Therefore, we need to maintain an efficient representation of the codeword boundaries on the payload. This second stream is referred as the *disambiguation information* throughout the study. The total space consumed by the payload and the disambiguation information introduces an overhead of $2(d-1)/d \cdot 2^d \approx \frac{1}{2^{d-1}}$ bits per each original bit, which becomes negligible as d increases, for example, it is less than 7 bits per a thousand bit when $d = 8$. Thus, proposed scheme actually splits the input into two partitions, which occupy almost the same space.

We prove that the *payload* occupies $\approx \frac{(d-2)}{d}$, and the *disambiguation information* takes $\approx \frac{2}{d}$ of the final volume. When the payload, which is the main source information, is

encrypted and disambiguation information is stored in plain, the amount of to-be-encrypted volume decreases by $2/d$ of the original size, e.g., for $d = 8$, 25% of the data is not required to be encrypted. The payload can be subject to the same process recursively, which gives us the opportunity to tune the size of the encrypted volume with processing power. For instance, in case $d = 8$, a second level of encoding of the first level's payload increases the gain in encryption from 25% to 43%.

In this scenario, it is important to analyze the information leakage by the plain disambiguation information. Since the payload is encrypted, it should not be easier for an attacker to guess the payload from the disambiguation information rather than breaking the ciphered payload. When $2 \cdot 2^d \leq n \leq \tau \cdot d \cdot 2^d$, for $\tau = \frac{d-1.44}{2}$, we show that the contraction of the possible message space 2^n due to the public disambiguation information is accommodated by keeping the codeword set secret.

It might have captured the attention of the reader that the analysis assumes the input bit stream to be uniformly i.i.d., which seems a bit restrictive at a first glance. However, the target data types of the introduced method are mainly the sources that have been previously entropy encoded² such as the video files in mpeg4 format, sound files in mp3, and similar others. The output of the compression tools squeezing data down to its entropy actually is actually quite nice input for our proposal. We support this observation by the experiments performed on various compressed file types showed that the results on real data is very close to the theoretical bounds computed by the uniformly i.i.d. assumption.

The outline of the paper is as follows. In Section 2 we introduce the proposed ambiguous encoding method based on the non-prefix-free codes, and analyze its basic properties mostly focusing on the space consumption of the partitions. We also provide verification of the theoretical claims based on uniformly i.i.d. assumption on some files that are already entropy-encoded. Section 3 focuses on using the ambiguous coding to reduce the number of encryption operations, and investigates the information leakage by the disambiguation information, which is proposed to be stored in plain format without encryption. We finalize our study by summarizing the results and discussing further related research avenues.

2 Ambiguous Data Coding

Let $\mathcal{A} = a_1 a_2 \dots a_n$ denotes a uniformly independently and identically distributed bit sequence, and $d > 1$ is a predetermined block length. Without loss of generality we assume n is divisible by d . Otherwise, it is padded with random bits. \mathcal{A} can be represented as $\mathcal{B} = b_1 b_2 \dots b_r$, for $r = \frac{n}{d}$ such that each d -bits long b_i in \mathcal{B} is from the alphabet $\Sigma = \{0, 1, 2, \dots, 2^d - 1\}$.

We will first define the *minimum binary representation* of an integer, and then use this definition to state our encoding scheme.

► **Definition 1.** The **minimum binary representation (MBR)** of an integer $i \geq 2$ is its binary representation without the leftmost 1 bit.

As an example, $MBR(21) = 0101$ by omitting the leftmost set bit in its binary representation as $21 = (\mathbf{1}0101)_2$.

² Any lossless data compression scheme, where each symbol is represented by minimum number of bits close to the entropy of the symbol according to Shannon's theorem [17].

7:4 Ambiguous Coding For Selective Encryption

$\Sigma =$	0	1	2	3	4	5	6	7
$\Sigma' =$	6	0	5	1	7	2	4	3
	ϵ_1	ϵ_2	ϵ_3	ϵ_4	ϵ_5	ϵ_6	ϵ_7	ϵ_8
	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
$W =$	{000,011,100,111}	0	11	1	{001,010,101,110}	00	10	01
$\mathcal{A} =$	001	110	101	011	010	111	100	000
$\mathcal{B} =$	1	6	5	3	2	7	4	0
$NPF(\mathcal{A}) =$	w_2	w_7	w_6	w_4	w_3	w_8	w_5	w_1
$NPF(\mathcal{A}) =$	0	10	00	1	11	01	101	000
$DisInfo(\mathcal{A}) =$	01	1	1	01	1	1	00	00

■ **Figure 1** A simple sketch of the non-prefix-free coding of an input bit sequence \mathcal{A} , where \mathcal{B} is the representation of \mathcal{A} with the block length $d = 3$. Σ' is a random permutation of the corresponding alphabet Σ , and W is the non-prefix-free codeword set generated for Σ' according to Definition 2. The disambiguation information $DisInfo(\mathcal{A})$ is computed according to Lemma 5.

► **Definition 2.** Let $\Sigma' = \{\epsilon_1, \epsilon_2, \dots, \epsilon_{2^d}\}$ be a permutation of the given alphabet $\Sigma = \{0, 1, 2, \dots, 2^d - 1\}$, and $W = \{w_1, w_2, \dots, w_{2^d}\}$ is a codeword set such that

$$w_i = \begin{cases} MBR(2 + \epsilon_i) & \text{,if } \epsilon_i < 2^d - 2 \\ \{MBR(2^d + \zeta) : \forall \zeta \in \{0, 1, \dots, 2^d - 1\}, \text{where } \zeta = \{0, 3\} \pmod{4}\} & \text{,if } \epsilon_i = 2^d - 2 \\ \{MBR(2^d + \zeta) : \forall \zeta \in \{0, 1, \dots, 2^d - 1\}, \text{where } \zeta = \{1, 2\} \pmod{4}\} & \text{,if } \epsilon_i = 2^d - 1 \end{cases}$$

The representation of the input $\mathcal{A} = \mathcal{B} = b_1 b_2 \dots b_r$ with the *non-prefix-free* codeword set W is shown by $NPF(\mathcal{A}) = c_1 c_2 \dots c_r$ such that $c_i = w_{1+b_i}$. When a codeword c_i has multiple options, a randomly selected one among the possibilities is used.

The NPF coding of a sample sequence according to the Definitions 1 and 2 with the parameter $d = 3$ is shown in Figure 1. The codewords w_1 and w_5 are *sets* as their corresponding $\epsilon_1 = 6$ and $\epsilon_5 = 7$ values are greater than or equal to $6 = 2^3 - 2$. Thus, when $c_i = w_1$ or $c_i = w_5$, a randomly selected codeword respectively from sets w_1 or w_5 , is inserted.

► **Proposition 3.** In a codeword set W that is generated for a block length $d > 1$ according to Definition 2, the lengths of the codewords in bits range from 1 to d , where the number of ℓ -bits long codewords for each $\ell \in \{1, 2, \dots, d - 1\}$ is 2^ℓ , and for $\ell = d$ there exist 2 sets of codewords each of which includes 2^{d-1} elements.

Proof. According to Definition 2, the entities in W are minimum binary representations of numbers $\{2, 3, \dots, 2^{d+1} - 1\}$. Since the MBR bit-lengths of those numbers range from 1 to d , there are d distinct codeword lengths in W .

Each codeword length $\ell \in \{1, 2, \dots, d - 1\}$ defines 2^ℓ distinct codewords, and thus, total number of codewords defined by all possible $\ell < d$ values becomes $\sum_{i=1}^{d-1} 2^i = 2^d - 2$. The remaining 2 codewords out of the $|W| = 2^d$ items require d -bits long bit sequences.

For example, when $d = 3$, the W includes $2(= 2^1)$ codewords of 1-bit long, $4(= 2^2)$ codewords of length 2, and $2(= 2^3 - 6)$ codeword *sets* of length 3-bits as shown in Figure 1. ◀

► **Lemma 4.** The $NPF(\mathcal{A})$ is expected to occupy $n \cdot (1 - \frac{2}{d} + \frac{2(d+1)}{d \cdot 2^d})$ bits space for a uniformly i.i.d. input \mathcal{A} of length $n = r \cdot d$ bits.

Proof. The total bit length of the NPF codewords is simply $\sum_{\ell=1}^d C_\ell \cdot \ell$, where C_ℓ denotes the number of occurrences of the b_i values represented by ℓ -bits long codewords in \mathcal{B} . Assuming the uniform distribution of \mathcal{B} , each $b_i \in \{0, 1, 2, \dots, 2^d - 1\}$ appears $\frac{r}{2^d}$ times. The number

Codelength	# of occurrences	represented by	space consumption
$d - 1$	$\frac{r}{2} = \frac{r}{2^d} \cdot 2^{d-1}$	1	$1 \cdot \frac{r}{2}$
$d - 2$	$\frac{r}{4} = \frac{r}{2^d} \cdot 2^{d-2}$	01	$2 \cdot \frac{r}{4}$
$d - 3$	$\frac{r}{8} = \frac{r}{2^d} \cdot 2^{d-3}$	001	$3 \cdot \frac{r}{8}$
...
1	$\frac{r}{2^{d-1}} = \frac{r}{2^d} \cdot 2$	00...1	$(d - 1) \cdot \frac{r}{2^{d-1}}$
d	$\frac{r}{2^{d-1}} = \frac{r}{2^d} \cdot 2$	00...0	$(d - 1) \cdot \frac{r}{2^{d-1}}$
Total space occupied:			$r(2 - \frac{1}{2^{d-2}})$

■ **Figure 2** The representation of the codeword lengths to specify the codeword boundaries on the NPF stream.

of distinct b_i values represented by a codeword of length ℓ is 2^ℓ for $1 \leq \ell < d$, and two of the b_i values require $\ell = d$ bit long codewords as stated in Proposition 3. Thus, $C_\ell = \frac{r}{2^d} \cdot 2^\ell$ for $1 \leq \ell < d$, and $C_d = \frac{r}{2^d} \cdot 2$. The length of the $NPF(\mathcal{B})$ bit-stream can then be computed by

$$|NPF(\mathcal{A})| = \frac{r}{2^d} \cdot (1 \cdot 2 + 2 \cdot 2^2 + \dots + (d - 1) \cdot 2^{d-1} + d \cdot 2) \quad (1)$$

$$= \frac{r}{2^d} \cdot (2d + \sum_{i=1}^{d-1} i \cdot 2^i) = \frac{r}{2^d} \cdot (2d + 2^d \cdot (d - 2) + 2) \quad (2)$$

$$= \frac{r}{2^d} \cdot (2^d(d - 2) + 2(d + 1)) \quad (3)$$

$$= r \cdot d - r \cdot (2 - \frac{d + 1}{2^{d-1}}) \quad (4)$$

$$= r \cdot (d - 2 + \frac{d + 1}{2^{d-1}}) \quad (5)$$

$$= \frac{n}{d} \cdot (d - 2 + \frac{d + 1}{2^{d-1}}) \quad (6)$$

$$= n \cdot (1 - \frac{2}{d} + \frac{2(d + 1)}{d \cdot 2^d}) \quad (7)$$

While computing the summation term in equation (2), we use the formula from basic algebra that $\sum_{i=1}^p i \cdot 2^i = 2^{p+1}(p - 1) + 2$, and substitute $p = d - 1$. ◀

The input sequence \mathcal{A} is originally n bit long, and the NPF coding reduces that space by $n \cdot (\frac{2}{d} - \frac{2(d+1)}{2^d})$ bits. However, since non-prefix-free codes are not uniquely decodable, $NPF(\mathcal{A})$ cannot be decoded back correctly in absence of the codeword boundaries. Therefore, we need to represent these boundary positions on $NPF(\mathcal{A})$. Lemma 5 states an efficient method to achieve this task.

► **Lemma 5.** *The expected number of bits to specify the codeword boundaries in the $NPF(\mathcal{A})$ is $n \cdot (\frac{2}{d} - \frac{4}{d \cdot 2^d})$, where $|\mathcal{A}| = n = r \cdot d$.*

Proof. Due to Proposition 3 there are 2^ℓ distinct codewords with length ℓ for $\ell \in \{1, 2, \dots, d - 1\}$ and 2 codewords (sets) are generated for $\ell = d$. Since each d -bits block has equal probability of appearance on \mathcal{A} , the number of occurrences of codewords having length $\ell \in \{1, 2, \dots, d - 1\}$ is $\frac{r}{2^d} \cdot 2^\ell$. The most frequent codeword length is $(d - 1)$, which appears at half of the r codewords as $\frac{r}{2^d} \cdot 2^{d-1} = \frac{r}{2}$. It is followed by the codeword length $(d - 2)$ that is observed

7:6 Ambiguous Coding For Selective Encryption

■ **Table 1** The payload, disambiguation information, and overhead bits per each original bit introduced by the proposed ambiguous coding for some selected d values.

$d =$	4	6	8	10	12	14	16	20
Overhead per bit $\frac{d-1}{d \cdot 2^{d-1}} \approx$	0,094	0,026	0,007	0,002	$1.1 \cdot 10^{-4}$	$4.4 \cdot 10^{-4}$	$2,8 \cdot 10^{-5}$	$1.8 \cdot 10^{-6}$
Payload per bit $1 - \frac{2}{d} + \frac{2(d+1)}{d \cdot 2^d} \approx$	0.656	0.703	0.759	0.802	0.834	0.857	0.875	0.900
Dis.Info. per bit $\frac{2}{d} - \frac{4}{d \cdot 2^d} \approx$	0,438	0.323	0.248	0.200	0.167	0.143	0.125	0.100

$\frac{r}{4}$ times. When we examine the number of codewords with length $\ell \in \{1, 2, \dots, d-1\}$, we see that this distribution is geometric, as depicted in Figure 2. The optimal prefix-free codes for the codeword lengths are then $\{1, 01, 001, \dots, 0^{d-2}1, 0^{d-1}\}$, which correspond to codeword lengths $\{d-1, d-2, d-3, \dots, 1, d\}$ respectively. Thus, codeword length $\ell = (d-i) \in \{1, 2, \dots, d-1\}$, which appears $\frac{r}{2^d} \cdot 2^{d-i}$ times on \mathcal{A} , can be shown by i bits. We use $(d-1)$ consecutive zeros to represent the codeword length $\ell = 2$ as the number of occurrences of d -bits long codewords is equal to the number of 1 bit long codewords on \mathcal{A} . Notice that the representation of the codeword lengths are prefix-free that can be uniquely decoded.

Total number of bits required to represent the individual lengths of the codewords can be computed by

$$\frac{r}{2^d} (2(d-1) + \sum_{i=1}^{d-1} i \cdot 2^{d-i}) = r \cdot \left(\frac{2(d-1)}{2^d} + \sum_{i=1}^{d-1} i \cdot 2^{-i} \right) \quad (8)$$

$$= r \left(\frac{d-1}{2^{d-1}} + \frac{2^d - d - 1}{2^{d-1}} \right) \quad (9)$$

$$= r \left(2 - \frac{1}{2^{d-2}} \right) \quad (10)$$

$$= \frac{n}{d} \cdot \left(2 - \frac{1}{2^{d-2}} \right) \quad (11)$$

$$= n \left(\frac{2}{d} - \frac{4}{d \cdot 2^d} \right) \quad (12)$$

◀

► **Theorem 6.** *The ambiguous encoding of n bit long uniformly i.i.d. input \mathcal{A} sequence is achieved with $\frac{2(d-1)}{d \cdot 2^d}$ bits overhead per each original bit.*

Proof. Total overhead can be computed by subtracting the original length n from the sum of the space consumption described in Lemmas 4 and 5. Dividing this value by the n returns the overhead per bit as shown below.

$$\frac{1}{n} \cdot \left[n \cdot \left(1 - \frac{2}{d} + \frac{2(d+1)}{d \cdot 2^d} \right) + n \cdot \left(\frac{2}{d} - \frac{4}{d \cdot 2^d} \right) - n \right] = \frac{d-1}{d \cdot 2^{d-1}} = \frac{2(d-1)}{d \cdot 2^d} \quad (13)$$

◀

Table 1 summarizes the amount of extra bits introduced by the proposed encoding per each original bit in \mathcal{A} . A large overhead, which seems significant for small d , e.g., $d < 8$, may inhibit the usage of the method. However, thanks to the to the exponentially increasing denominator (2^d) in the overhead amount that the extra space consumption quickly becomes

very small, and even negligible. For instance, when $d = 8$, the method produces only 6.8 extra bits per a thousand bit. Similarly, the overhead becomes less than 3 bits per 100K bits, and less than 2 bits per a million bits for the values of $d = 16$ and $d = 20$, respectively. Thus, for $d \geq 8$, an input uniformly i.i.d. bit sequence can be represented with a negligible space overhead by the proposed ambiguous encoding scheme.

2.1 Experimental Verification

During the calculations of the payload and disambiguation information sizes as well as the overhead, the input data has been assumed to be independently and identically distributed. In practice, the input to the proposed method is supposed to be the output of an entropy coder, where the distribution of d -bits long items in such a file may deviate from the perfect assumptions. We would like to evaluate whether such entropy-encoded files still provide enough good uniformity close to the theoretical claims based on uniformly i.i.d. assumption. Therefore, we have conducted experiments on different compressed files to observe how much these theoretical values are caught in practice.

We have selected 16 publicly available files³, where the first ten are *gzip* compressed data from different sources and the remaining six are multimedia files of *mp3*, *mp4*, *jpg*, *webm*, *ogv*, and *flv* formats. The first $d \cdot 2^d$ bits of each file is inspected for distinct values of $d = \{8, 12, 16, 20\}$, and the corresponding observed payload and disambiguation information sizes are computed as well as the overhead bits in each case.

Table 2 includes the comparisons of the observed and theoretical values on each analyzed dimension. The payload size, which is the total length of the concatenated NPF codewords, and the disambiguation information size, which is the total length of the prefix-free encoded codeword lengths, are both observed to be compatible with the theoretical claims. This is also reflected on the overhead bits as a consequence. Thus, in terms of space consumption, the experimental results on compressed data support the theoretical findings based on perfect uniformly i.i.d. input data assumption.

3 Data Security with Reduced Encryption Operations

Given a high-entropy input bit-stream \mathcal{M} , two secret keys \mathcal{K}_1 and \mathcal{K}_2 , and a properly chosen d parameter, the data security scheme $\mathcal{S}(\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}, d)$ aiming reduced encryption operations starts with generating the permutation $\Sigma' = \{\epsilon_1, \epsilon_2, \dots, \epsilon_{2^d}\}$ via a cryptographically secure pseudo-random number generator seeded with the secret key \mathcal{K}_1 . The input data \mathcal{M} is then encoded with the ambiguous coding described in previous section. This encoding generates the *payload*, which is the concatenated NPF codewords, and the *disambiguation information*, which simply specifies the lengths of individual codewords via an optimal prefix-free code. The payload is encrypted with some selected encryption algorithm by using the key \mathcal{K}_2 , where the disambiguation information is kept in plain. Hence, we need to analyze how much information is revealed by the public disambiguation information, and show that the leakage by the disambiguation information section does not provide an advantage for an attacker to break the cipher on the payload.

³ First ten files are available from <http://corpus.canterbury.ac.nz>. and <http://people.unipmn.it/~manzini/lightweight/corpus/>. The multimedia files are from <https://github.com/johndyer/mediaelement-files>.

7:8 Ambiguous Coding For Selective Encryption

■ **Table 2** Verification of the theoretical claims on selected files for $d = \{8, 12, 16, 20\}$.

File name	Payload Size		Disambiguation Information Size		Overhead Bits	
	Thr.	Obs.	Thr.	Obs.	Thr.	Obs.
chr22		1555		500		7
etext99		1515		533		0
gcc		1491		578		21
howtobwt		1510		538		0
howto		1551		511		14
jdk		1522		540		14
rctail		1535		527		14
rfc	1554	1522	508	526	14	0
sprot34		1538		524		14
w3c2		1529		519		0
mp3		1470		585		7
jpg		1426		636		14
mp4		1415		654		21
webm		1496		566		14
ogv		1456		592		0
flv		1571		484		7
<hr/>						
chr22		40961		8213		22
etext99		41103		8060		11
gcc		40764		8410		22
howtobwt		41058		8127		33
howto		41079		8095		22
jdk		41074		8122		44
rctail		41075		8088		11
rfc	40986	40769	8188	8394	22	11
sprot34		41049		8125		22
w3c2		41016		8158		22
mp3		41021		8131		0
jpg		40819		8344		11
mp4		41373		7779		0
webm		40835		8317		0
ogv		40985		8189		22
flv		40796		8367		11
<hr/>						
chr22		917579		131027		30
etext99		917289		131302		15
gcc		917397		131209		30
howtobwt		917518		131088		30
howto		917812		130794		30
jdk		917412		131179		15
rctail		917139		131437		0
rfc	917538	917346	131068	131275	30	45
sprot34		918158		130433		15
w3c2		917707		130899		30
mp3		914926		133695		45
jpg		915821		132770		15
mp4		905887		142689		0
webm		917075		131561		60
ogv		916558		132108		90
flv		915335		133286		45
<hr/>						
chr22		18873040		2098575		95
etext99		18872686		2098853		19
gcc		18879975		2091602		57
howtobwt		18875332		2096207		19
howto		18875502		2096037		19
jdk		18873190		2098406		76
rctail		18876497		2095042		19
rfc	18874410	18873175	2097148	2098364	38	19
sprot34		18878705		2092891		76
w3c2		18876613		2094945		38
mp3		18863837		2107721		38
jpg		18878914		2092625		19
mp4		18898789		2072826		95
webm		18873348		2098210		38
ogv		18875407		2096208		95
flv		18909861		2061735		76

► **Lemma 7.** *The number of distinct messages that can be generated from a given disambiguation information is $2^{n - \frac{2n}{d} + \frac{4n}{d^2}}$ by assuming the codeword set W , parameter d , and message length n are known.*

Proof. A codeword length $\ell = d - i$ appears $\frac{r}{2^i}$ times in the disambiguation information for $i = 1$ to $d - 1$, and represents 2^ℓ distinct symbols. The d bit long codewords appear $\frac{r}{2^{d-1}}$ times, and represent two distinct symbols. Thus, the total number of distinct sequences that can be generated from a known disambiguation information can be counted by

$$2^{\frac{r(d-1)}{2}} \cdot 2^{\frac{r(d-2)}{4}} \cdot \dots \cdot 2^{\frac{r}{2^{d-1}}} \cdot 2^{\frac{r}{2^{d-1}}} = 2^{rd \sum_{i=1}^{d-1} 2^{-i}} \cdot 2^r \sum_{i=1}^{d-1} i 2^{-i} \cdot 2^{\frac{r}{2^{d-1}}} \quad (14)$$

$$= 2^{\frac{rd(2^{d-1}-1) - r(2^d - d - 1) + 1}{2^{d-1}}} \quad (15)$$

$$= 2^{r(d-2 + \frac{4}{2^d})} \quad (16)$$

$$= 2^{n - \frac{2n}{d} + \frac{4n}{d^2}} \quad (17)$$

◀

The result of Lemma 7 is consistent with previous Lemma 5 such that the disambiguation information is not squeezing the possible message space by more than its size. In other words, when the codeword set W is known, plain disambiguation information reduces the possible 2^n message space to $2^{n-\epsilon}$, where $\epsilon = n(\frac{2}{d} - \frac{4}{d \cdot 2^d})$.

However, in the proposed scheme, W is private, and we need to investigate whether that secrecy of W accommodates the loss of information by the public disambiguation data. Lemma 8 shows that for an attacker using the knowledge revealed by the disambiguation information does not provide an advantage over breaking the encryption on the payload as long as the codeword set W is kept secret.

► **Lemma 8.** *The shrinkage in the possible message space due to public disambiguation information can be accommodated by keeping the codeword set W secret in the ambiguous coding of $n \leq \tau \cdot d \cdot 2^d$ bit long data for $\tau = \frac{d-1.44}{2}$.*

Proof. W is a secret permutation of the set $\{0, 1, 2, \dots, 2^d - 1\}$ containing 2^d numbers. Thus, there are $2^d!$ distinct possibilities, which corresponds to $\log 2^d!$ bits of information. On the other hand, the amount of revealed knowledge about the n bit long input by the disambiguation information is $n(\frac{2}{d} - \frac{4}{d \cdot 2^d})$ bits. The advantage gained by keeping W secret should accommodate the loss by making disambiguation information public. This simply yields the following equation.

$$\log(2^d!) \geq n \cdot \left(\frac{2}{d} - \frac{4}{d \cdot 2^d} \right) \quad (18)$$

$$\frac{\ln(2^d!)}{\ln 2} \approx \frac{2^d \cdot \ln 2^d - 2^d}{\ln 2} \geq n \cdot \frac{2}{d} \quad (19)$$

$$\frac{2^d \cdot d \cdot \ln 2 - 2^d}{\ln 2} \geq n \cdot \frac{2}{d} \quad (20)$$

$$2^d(d - 1.44) \geq n \cdot \frac{2}{d} \quad (21)$$

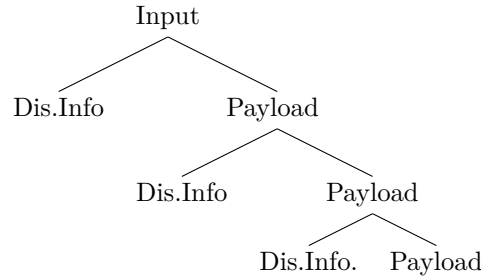
$$d \cdot 2^d \cdot \left(\frac{d - 1.44}{2} \right) \geq n \quad (22)$$

◀

7:10 Ambiguous Coding For Selective Encryption

■ **Table 3** The minimum ($d \cdot 2^d$) and maximum ($d \cdot 2^d \cdot (\frac{d-1.44}{2})$) block sizes that are appropriate according to the proposed ambiguous coding scheme for selected d values.

d	Block Size in bits		d	Block Size in bits	
	min	max		min	max
6	384	875	14	230K	144K
8	2K	6.7K	16	1M	7M
10	10K	43K	18	4.7M	39M
12	49K	256K	20	21M	194M



■ **Figure 3** Sketch of three rounds recursive application of the proposed scheme for further reduction in encryption amount.

Yet another point not to neglect in practice is to choose d such that $2^{d!}$ is no smaller than $2^{\mathcal{K}_1}$. This is to make sure that it should not be easier to try all possible permutations than breaking the secret key (\mathcal{K}_1) of the pseudo-random number generator used in creating the permutation. Assuming the keys used in symmetric encryption schemes are at least 256 bits, $d \geq 6$ seems a lower bound for a security level provided by a 256 bit symmetric encryption since $2^{6!} > 2^{295} > 2^{256}$.

Due to Lemma 8, the choice of d creates an upper bound on the size of the input data that will be subject to the proposed ambiguous coding scheme. On the other hand, it would be appropriate to select d such that the input size is at least $d \cdot 2^d$ bits to confirm with the computations in the size arguments of the payload and the disambiguation information, which assumed all possible 2^d symbols are uniformly i.i.d. on the input. The minimum and maximum block sizes defined by the d parameter are listed in Table 3 considering these facts. Therefore, given an input bit string \mathcal{A} , the ambiguous encoding to be achieved in blocks of the any preferred size in between these values seems appropriate in practice.

The value of d plays a crucial role both in the security and in the to-be-encrypted data size. It is good to choose large d for better security with less (even negligible when $d > 8$) overhead. On the other hand, the payload size is inversely proportional with d , and thus, the reduction in the data volume to be encrypted decreases when d increases.

Thus, to achieve better reductions in the encryption amount, the ambiguous coding can be recursively applied on the payload generated after the first round. Figure 3 sketches this by considering three rounds and Table 4 lists the percentage of gain for each level. For instance, when $d = 8$, the gain in to-be-encrypted volume is around 25 percent. If the payload, which is roughly 75 percent of the original data at the end of this round, is again encoded with the proposed scheme, then gain is improved to more than 40 percent. Even one more round reaches near 60 percent less encryption requirement. Notice that in case of multiple application of the ambiguous coding, the latest payload is encrypted and the remaining disambiguation information are kept plain.

■ **Table 4** Percentages of the disambiguation and payload data with 3 rounds of recursion for various d values calculated according to the lemmas 4 and 5. The bold values represents the percentage of the data to be encrypted.

$d :$		6	8	10	12	14	16
1st Round	Dis. Info.	32.29	24.80	19.96	16.66	14.28	12.50
	Payload	70.31	75.88	80.21	83.39	85.73	87.50
	Overhead	2.60	0.68	0.18	0.04	0.01	0.00
2nd Round	Dis. Info.	22.71	18.82	16.01	13.89	12.25	10.94
	Payload	49.44	57.58	64.34	69.53	73.49	76.57
	Overhead	4.44	1.20	0.32	0.08	0.02	0.01
3rd Round	Dis. Info.	15.96	14.28	12.84	11.58	10.50	9.57
	Payload	34.76	43.69	51.61	57.98	63.00	67.00
	Overhead	5.72	1.60	0.43	0.11	0.03	0.01

4 Conclusions

We have presented an ambiguous coding scheme based on variable-length non-prefix-free codes that splits an input bit-stream into two as the payload and the disambiguation information. We have proved that the overhead at the end of this coding becomes negligible, particularly when $d \geq 8$. The encryption of the payload is supposed to be performed by standard ways, and the disambiguation information is kept plain. Thus, there appears a gain in the amount to be encrypted, which is equal to the disambiguation information size. Proposed ambiguous coding can be applied recursively on the payload generated as depicted in Figure 3 to increase that gain in encryption amount.

We assumed that the input to the ambiguous encoder is uniformly i.i.d. in ideal case, and empirically verified that the compressed volumes ensures the mentioned results. Actually, applying entropy coding before the encryption is a common daily practice, which makes the proposed method to be directly integrated. The *mpeg4* video streams, *jpg* images, compressed text sequences, or *mp3* songs are all typical data sources of high-entropy. Related previous work [5, 16] had stated that although the perfect security of an input data requires a key length equal to its size (one-time pad), high-entropy data can be perfectly secured with much shorter keys. This study addresses another dimension and investigates achieving security of such volumes by encrypting less than their original sizes by using the introduced ambiguous coding scheme.

Reducing the amount of data to-be-encrypted can make sense in scenarios where the encryption process defines a bottleneck in terms of some metrics. Ambiguous coding becomes particularly efficient on securing large collections over power-limited devices, where the cost of encryption becomes heavy in terms of energy. This reduction also helps to increase the throughput of a security pipeline without a need to expand the relatively expensive security hardware. For instance, let's assume a case where the data is waiting to be processed by a hardware security unit. When the amount of data exceeds the capacity of this unit, a bottleneck appears, which can be resolved by increasing the number of such security units. However, adding and managing more security units is costly, particularly when the bottleneck is not so frequent, but only appearing at some time. An alternative solution is to use the proposed ambiguous coding, where instead of expanding the security units, data can be processed appropriately while waiting in the queue, and the amount to be encrypted can be reduced up to desired level by applying the scheme recursively if needed. Notice that as

opposed to previous selective encryption schemes, ambiguous coding supports the security of the whole file instead of securing only the selected partitions. Besides massive multimedia files, small public key files around a few kilobytes that are used in asymmetric encryption schemes are also very suitable inputs for the ambiguous coding. The exchange of public keys via symmetric ciphers can also benefit from the reduction introduced.

The non-prefix-free codes have not received much attention in the literature due to their intrinsic decodability problem. However, such a disadvantage may turn to be an advantage in terms of security systems as investigated in this study. Further security applications based on such ambiguous codes have the potential to be out-of-the box solutions particularly in privacy preserving information retrieval and secure text processing applications.

References

- 1 Boran Adaş, Ersin Bayraktar, and M Oğuzhan Külekci. Huffman codes versus augmented non-prefix-free codes. In *Experimental Algorithms*, pages 315–326. Springer, 2015.
- 2 Norman L Biggs. Prefix free codes. In *Codes: An Introduction to Information Communication and Cryptography*, pages 1–14. Springer, 2008.
- 3 R. Chandramouli, S. Bapatla, K. P. Subbalakshmi, and R. N. Uma. Battery power-aware encryption. *ACM Trans. Inf. Syst. Secur.*, 9(2):162–180, 2006. doi:10.1145/1151414.1151417.
- 4 Marco Dalai and Riccardo Leonardi. Non prefix-free codes for constrained sequences. In *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, pages 1534–1538. IEEE, 2005.
- 5 Yevgeniy Dodis and Adam Smith. Entropic security and the encryption of high entropy messages. In *Theory of Cryptography Conference*, pages 556–577. Springer, 2005.
- 6 Aviezri S. Fraenkel and Shmuel T. Klein. Complexity aspects of guessing prefix codes. *Algorithmica*, 12(4-5):409–419, 1994.
- 7 David W Gillman, Mojdeh Mohtashemi, and Ronald L Rivest. On breaking a huffman code. *IEEE Transactions on Information theory*, 42(3):972–976, 1996.
- 8 Marco Grangetto, Enrico Magli, and Gabriella Olmo. Multimedia selective encryption by means of randomized arithmetic coding. *IEEE Transactions on Multimedia*, 8(5):905–917, 2006.
- 9 Fadi Hamad, Leonid Smalov, and Anne James. Energy-aware security in m-commerce and the internet of things. *IETE Technical review*, 26(5):357–362, 2009.
- 10 Muhammed Oğuzhan Külekci. Uniquely decodable and directly accessible non-prefix-free codes via wavelet trees. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 1969–1973. IEEE, 2013.
- 11 Shiguo Lian, Zhongxuan Liu, Zhen Ren, and Haila Wang. Secure advanced video coding based on selective encryption algorithms. *IEEE Transactions on Consumer Electronics*, 52(2):621–629, 2006.
- 12 Ayoub Massoudi, Frédéric Lefebvre, Christophe De Vleeschouwer, Benoit Macq, and J-J Quisquater. Overview on selective encryption of image and video: challenges and perspectives. *EURASIP Journal on Information Security*, 2008(1):1, 2008.
- 13 Rashmi Bangalore Muralidhar. Substitution cipher with nonprefix codes. Master’s thesis, San Jose State University, 2011.
- 14 Nachiketh R Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on Mobile Computing*, 5(2):128–143, 2006.
- 15 Frank Rubin. Cryptographic aspects of data compression codes. *Cryptologia*, 3(4):202–205, 1979.

- 16 Alexander Russell and Hong Wang. How to fool an unbounded adversary with a short key. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 133–148. Springer, 2002.
- 17 Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.
- 18 Balemir Uragun. Energy efficiency for unmanned aerial vehicles. In *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, volume 2, pages 316–320. IEEE, 2011.
- 19 Marc Van Droogenbroeck and Raphaël Benedett. Techniques for a selective encryption of uncompressed and compressed images. *ACIVS Advanced Concepts for Intelligent Vision Systems, Proceedings*, pages 90–97, 2002.