

A Method to Ensure the Confidentiality of the Compressed Data

M. Oğuzhan Külekci

National Research Institute of Electronics & Cryptology, Turkey.

kulekci@uekae.tubitak.gov.tr

Abstract—The usual way of ensuring the confidentiality of the compressed data is to encrypt it with a standard encryption algorithm such as the AES. However, the cost of encryption not only brings an additional computational complexity, but also lacks the flexibility to perform pattern matching on the compressed data, which is an active research topic in stringology. In this study, we investigate the secure compression solutions, and propose a practical method to keep contents of the compressed data hidden from unauthorized people. The method is based on the Burrows–Wheeler transform (BWT) such that a random ordering of the input symbols are used as the lexicographical ordering during the construction. The motivation is the observation that on BWT of an input data it is not possible to perform a successful search nor construct any part of it without the knowledge of the character ordering used in the construction. Capturing that secret ordering from the BWT is hard. The proposed method is more elegant than a standard encryption process, with an additional advantage that it supports the compressed pattern matching methods, while still pertaining the confidentiality. When the input data is homophonic, where the frequencies of the symbols are flat and the alphabet is sufficiently large, it is possible to unify compression and security in a single framework with the proposed technique instead of the two–level compress–then–encrypt paradigm.

Keywords–secure compression, Burrows–Wheeler transform, compressed pattern matching

I. INTRODUCTION

Compression aims to represent an input data with least number of bits. The reduced size of the input yields efficient storage. Another benefit is the better use of the available bandwidth in a network while transmitting the data. As an example, it would be really difficult to store/share/exchange movies or songs without the contemporary multimedia compression techniques.

On the other hand, security and privacy are vital in today’s digital life as data is vulnerable to attacks while being transmitted over a network or stored in a multi–user environment. Hence, the users demand for compression combined with security, which gives rise to secure compression paradigm.

The naive and most widely used technique in practice is to apply a standard encryption algorithm to the compressed data. Most of the compression software used today include an option to encrypt the output with a password specified by the user. As an example WinZip¹ and WinRAR² packages

provide encrypting the compressed data with the AES [1] algorithm. That compress–then–encrypt approach provides high level of security in case of proper usage, such as selecting strong passwords. Several attacks [2], [3], [4], [5], [6] on encrypted compressed data has been published, which are mainly because of improper injection of cryptographic primitives into the compression software.

The computational load of the compress–then–encrypt methods become more serious when mobile environments with limited resources are considered [7]. Another disadvantage of this approach is the fact that the encryption level lacks the opportunity to perform text processing on compressed data, which means any operation doable on the underlying text without decompressing the whole file [8]. Pattern matching on compressed data [9], [10] is such an operation that aims to search strings on the compressed data without decompressing it. When encryption level is appended over compression, the compressed pattern matching methods may be applied only after decryption. Since decrypt/encrypt algorithms are not computationally cheap, it becomes not very feasible to follow this path.

A second approach in securing the compressed data is to unify compression and encryption. Creating multiple statistical models for the compression and selecting one of them according to a key stream generated via a password is one of the ways for such an integration. If the target is a multimedia file such as a video stream, the coefficients used in the compression algorithm may be encrypted also [11]. Some alterations in the entropy coders have also been proposed such as the interval splitting in arithmetic coding [8] or using multiple code–trees in Huffman coding [7]. Using encrypted dictionaries [12] in dictionary based compression algorithms is yet another alternative. Unifying compression and security seems more elegant than a pure encryption scheme in a sense that both compression and security requirements are fulfilled in a single step. However, the main drawback here is the sacrifice of compression ratio as a consequence of alterations performed in the original compression technique. Although it is possible to compress data more effectively, we are losing a bit of space with the aim of injecting security. Another disadvantage is again the lack of efficient methods to perform compressed pattern matching on concurrently compressed and secured data.

This study proposes a new alternative to provide security of a compressed file. The main idea is based on

¹www.winzip.com

²www.rarlab.com

the observation that the reconstruction of the original data from its Burrows-Wheeler transform (BWT) [13] or the backwards search [14] of a pattern require the knowledge of the lexicographical ordering used in the construction of the BWT. When a random permutation of the symbols is used as the lexicographic ordering in BWT, it is hard to deduce this ordering especially when the data is homophonic with a large alphabet. The reason we emphasize the homophonic property is the fact that if the underlying data has a statistical bias, the ordering used in the BWT is vulnerable to a statistical attack (as described in section 4).

Since a compressed sequence is homophonic with all possible 256 bytes in practice, if we apply BWT with such a secret ordering to the output of a compression algorithm, we can ensure the security of the compressed data. This is an alternative method to compress-then-encrypt method as compress-then-BWT. The computational load of BWT is lighter than an encryption process. Despite its easier construction, the main advantage is the ability of BWT to reconstruct any part of the text or to search a pattern of length m in $O(m)$ steps via backwards search. This property enables pattern matching on the compressed data without decompressing it, which is not possible when an encryption scheme is applied.

If the distribution of the characters in the input data is flat (homophonic), the proposed method can be directly applied, which results a *secure-BWT* compression achieving compression and security simultaneously. In such a case the proposed method becomes an alternative to the second approach, where compression and security are unified. An advantage here is the possibility to deploy pattern matching on compressed data. Note that we do not need to run a concurrent stream cipher besides the BWT compression as oppose to other techniques of this genre. The details are described in section 4.

The outline of the paper is as follows. We investigate the previous studies in section 2 followed by the review of the BWT and backwards searching in section 3. The proposed method is introduced and discussed in section 4. We conclude with the summary and further opportunities.

II. PREVIOUS WORK ON SECURING THE COMPRESSION

Previous studies on secure compression generally focus on two approaches as encrypting the output of the compression or altering the compression algorithms to provide security.

The first approach, which we refer as compress-then-encrypt, is mostly preferred by commercial compression solutions such as the WinRAR and WinZip, where both use AES algorithm since 2002 and 2004 respectively. Prior to AES usage, the security features of WinZip were investigated and some early attacks were proposed by Biham&Kohen [2] in 1994 followed by the study of Stay [3] in 2002. In 2004 Kohno [4], [5] showed that there were still vulnerabilities after the inclusion of the AES on version 9.0 of the WinZip

software. It worths here to note that those attacks were not attempting to break the AES cipher, but benefitting from some weaknesses originating from the improper integration of compression software and the encryption. Similar attacks on WinRAR software were also introduced by Yeo&Phan [6] in 2006.

The compress-then-encrypt methods provide high level confidentiality with the additional computational cost of the encryption algorithms that are proved to be secure as a second layer. Although this load is arguably affordable on today's powerful processors, it is more limiting especially on mobile environments, where the computational resources are less. A second drawback comes with the lack of opportunities to perform search on the compressed data. There exists methods [15] to run pattern matching on compressed data without decompressing it. However, the encryption level lacks this opportunity. It is arguable that there are also some studies aiming searchable encryption [16]. Unfortunately, the cryptographic primitives used in those searchable encryption algorithms are far more slow and resource demanding than any standard algorithm, which makes them infeasible in practice.

The second approach in securing the compressed data is to integrate compression and encryption in a single framework. One of the earliest studies on this area by Witten&Cleary [17] in 1988 proposed keeping the underlying probability model hidden in an adaptive arithmetic coder for security. Bergen&Hogan in 1992 [18] and 1993 [19] showed that this was not appropriate by introducing an attack capturing the secret statistical model of the coder, where Liu *et al.* [20] in 1997 introduced an alternative method to defer the Bergen&Hogan attack. Cleary *et al.* [21] decried that arithmetic coding is not secure in 1995 and Lim *et al.* [22] gave a complete cryptanalysis in 1997. Barbir [23] proposed an alteration in arithmetic coding to make it secure. The idea was to update the statistical model not after the compression of every symbol, but only at some specific points defined by a concurrently running stream cipher. This semi-adaptive time-variant modeling scheme degraded the compression efficiency for security. Randomized arithmetic coding by Grangetto *et al.* in 2004 [24] and 2006 [25] again incorporated a stream cipher such that the probability intervals of characters were shuffled during the compression according to the bits generated by the cipher. A similar study in 2007 by Kim *et al.* [8] combined shuffling with interval splitting, where the probability interval of a symbol was defined no more to be contiguous, but splitted into several pieces. Recently, Zhou *et al.* [26] in 2009 described a chosen-ciphertext attack on that secure arithmetic coding. Despite numerous efforts to make arithmetic coding secure, there has been also some attempts focusing on other compression techniques. For example, Wu&Kuo [7] in 2005 incorporated multiple code-trees in Huffman coding that are based on multiple statistical models used in a secret ordering and

In 2008 Zhou *et al.* [27] concentrated on dictionary based compression and proposed a method to secure Lempel–Ziv–Welch compression.

Unified compression and encryption methods usually include a stream cipher that is run concurrently with the compression algorithm. The entropy coders are tuned to execute on multiple models in a secret ordering that is specified by the output of the stream cipher. In general, altering a compression scheme to provide security causes a decrease in compression performance and speed. The disadvantage of missing flexibility to perform pattern matching on the compressed data is still a problem of this genre as is the case for the compress–then–encrypt algorithms.

III. BURROWS–WHEELER TRANSFORM AND BACKWARDS SEARCHING PARADIGM

A. The Burrows–Wheeler Transform

Let the input text be denoted by $T = t_1 t_2 \dots t_n$, where $t_i \in \Sigma$ for $1 \leq i \leq n$ and Σ represents the alphabet of characters occurring in T as $\Sigma = \{\epsilon_1, \epsilon_2, \dots, \epsilon_k\}$. The cardinality of Σ is $|\Sigma| = k$. While computing the BWT of T , a hypothetical end-of-sequence character $\$$, which is not in Σ and lexicographically smaller than any other symbol $\epsilon_i \in \Sigma$, $1 \leq i \leq k$, is assumed to be inserted at the end of T .

The cyclic–right–shift s of T is defined as $CRS_s(T) = t_s t_{s+1} \dots t_n \$ t_1 t_2 \dots t_{s-1}$. We lexicographically sort all $CRS_s(T)$ strings for $1 \leq s \leq (n + 1)$. The BWT of T is formed by taking the corresponding rightmost characters of each $CRS_s(T)$ in the sorted order, which is shown in Figure 1 on a sample input text $T = \text{mississippi}$. The last column $L = \text{ipssm\$pissii}$ depicted in Figure 1 is the resulting $BWT(T)$.

B. Reconstructing Original Text from Its BWT

The procedure to reconstruct the original text T from its BWT is given in Algorithm 1. The initial step is to compute the first column F from the input $L = BWT(T)$. Column F is simply the lexicographical ordering of the characters appearing in L as can be observed at figure 1 c). Once the ordering among the symbols of Σ is known, F can be calculated easily by sorting the characters of L . We need to define the functions $range_F(\epsilon_i)$ and $rank_L(pos, ch)$ to continue with the reconstruction.

- $[begin, end] \leftarrow range_F(\epsilon_i)$: $begin$ and end are the beginning and ending indices of ϵ_i on F , e.g., $range_F(p) = [7, 8]$ in Figure 1.
- $q \leftarrow rank_L(pos, ch)$: q is the number of characters that are equal to ch up to position pos in L , e.g., $rank_L(8, i) = 1$ in Figure 1.

Since $CRS_{n+1} = \$t_1 t_2 \dots t_n$ appears at top of the sorted list regarding to the assumption that $\$$ is the least significant symbol, the first character of L is always the last character of T as $t_n = \ell_1$. Thus, we set the beginning position on L

Input: $BWT(T) = L = \ell_1 \ell_2 \dots \ell_{n+1}$

Output: $T = t_1 t_2 \dots t_n$

```

1  $F \leftarrow \text{construct}F(L)$ ;
2  $j \leftarrow 1$ ;
3 for  $i = n$  down to 1 do
4    $t_i \leftarrow \ell_j$ ;
5    $[begin, end] \leftarrow \text{range}_F(\ell_j)$ ;
6    $q = \text{rank}_L(j, \ell_j)$ ;
7    $j \leftarrow begin + q$ ;
8 end

```

Algorithm 1: ReconstructText

as $j = 1$. We continue by finding the range of ℓ_j on the F column and move forward from the beginning of this range equal to the number of ℓ_j characters observed up to point j on L . We update the value of j equal to this position's index. The next character (in reverse order) of the text is the corresponding symbol ℓ_j on L column. Same procedure is repeated until the complete text is reconstructed.

C. Searching a Pattern via BWT

Input: Pattern $P = p_1 p_2 \dots p_m$,

$BWT(T) = L = \ell_1 \ell_2 \dots \ell_{n+1}$

Output: How many times P appears in T ?

```

1  $F \leftarrow \text{construct}F(L)$ ;
2  $i \leftarrow m$ ;  $[begin, end] \leftarrow \text{range}_F(p_m)$ ;
3 while  $i > 1$  do
4   if  $(end - begin) < 1$  then return 0;
5    $q_1 \leftarrow \text{rank}_L(begin, p_{i-1})$ ;
6    $q_2 \leftarrow \text{rank}_L(end, p_{i-1})$ ;
7   if  $\ell_{end} = p_{i-1}$  then  $q_2 \leftarrow q_2 + 1$ ;
8    $[begin, end] \leftarrow \text{range}_F(p_{i-1})$ ;
9    $end \leftarrow begin + q_2 - 1$ ;
10   $begin \leftarrow begin + q_1$ ;
11   $i = i - 1$ ;
12 end
13 return  $end - begin + 1$ ;

```

Algorithm 2: FindPattern

We use the same $rank_L$ and $range_F$ primitives in searching for a given pattern $P = p_1 p_2 \dots p_m$ as indicated in Algorithm 2. After computing the F column from the given L column, we proceed on P in reverse order by setting the initial value i to m , and the initial range to $range_F(p_m)$. The initial range on F is narrowed down according to the number of p_{i-1} characters observed on L up to the $begin$ and end positions of this range. The main reason is the property of BWT that the h^{th} occurrence of a symbol on the L column corresponds to its h^{th} appearance on F column. We continue this process until the range is empty, which indicates P is not observed in T , or we reach to the beginning of P with this traversal, which means number of occurrences of P is

s	$CRS_s(T)$	s	$CRS_s(T)$	i	F	L	i	
1	mississippi\$	12	\$mississippi	1	\$	mississipp	i	1
2	ississippi\$m	11	i\$mississipp	2	i	\$mississip	p	2
3	ssissippi\$mi	8	ippi\$mississ	3	i	ppi\$missis	s	3
4	sissippi\$mis	5	issippi\$miss	4	i	ssippi\$mis	s	4
5	issippi\$miss	2	issippi\$mi	5	i	ssissippi\$	m	5
6	ssippi\$missi	1	mississippi\$	6	m	issippi	\$	6
7	sippi\$missis	10	pi\$mississip	7	p	i\$mississi	p	7
8	ippi\$mississ	9	ppi\$mississi	8	p	pi\$mississ	i	8
9	ppi\$mississi	7	sippi\$missis	9	s	ippi\$missi	s	9
10	pi\$mississip	4	sissippi\$mis	10	s	issippi\$mi	s	10
11	i\$mississipp	6	ssippi\$missi	11	s	sippi\$miss	i	11
12	\$mississippi	3	ssissippi\$mi	12	s	sissippi\$m	i	12

a) Cyclic right shifts

b) Sorted CRS

c) $BWT(T) = L$

Figure 1. The BWT of sample text mississippi.

equal to the length of the range. More details on backwards searching can be obtained from [14] and [28].

IV. THE METHOD

The $range_F$ and $rank_L$ queries described in section 3 require the knowledge of the lexicographical ordering. We propose to use Burrows-Wheeler transform with a secret lexicographical ordering to support security in data compression. Assume user \mathcal{U} wants to compress file \mathcal{F} and keep it secure from others. \mathcal{U} selects a random permutation $\mathcal{O} = o_1 < o_2 < \dots < o_k$, where for all $1 < i \leq k$, $o_i \in \Sigma$ and $o_i \neq o_j$ while $i \neq j$. We will investigate the possible solutions of achieving compression and security with the proposed method under two sections as direct application of *secure-BWT* and *compress-then-BWT*.

A. Secure-BWT

Assume that (U) first computes the BWT of T according to (O) , followed by the classical move-to-front and run-length encoding to achieve the compression. It is possible to perform search on BWT compressed text [29]. Thus, we have the ability to perform efficient pattern matching on compressed data in such a scheme. The use of the secret lexicographical ordering \mathcal{O} preserves the confidentiality of the data as it is not possible to open the archive without \mathcal{O} .

The only way of a malicious adversary is to capture \mathcal{O} . There exists $|\Sigma|!$ possible distinct permutations of symbols in Σ . The brute-force attack requires $|\Sigma|!$ decompression and verification according to a stopping criteria. For sufficiently large $|\Sigma|$ (e.g. $20! > 2^{61}$, $25! > 2^{83}$, $100! > 2^{526}$) this becomes impractical. If the alphabet size is not large enough to provide enough security, say less than 20, it is always possible to represent the data with a super alphabet by combining consecutive symbols. However, if there exists a statistical relationship between the characters of the input

text, as is the case for natural languages, there exists a convenient attack.

It is possible to get the number of occurrences, $freq_i$, of each character $\epsilon_i \in \Sigma$ by simply counting on the L column. The problem is than to correctly determine the F column, which also means capturing the secret \mathcal{O} since the characters are sorted according to their lexicographic order on F . For any $\epsilon_i \in \Sigma$, assume r_i indicates its starting position on F . Thus, the range of ϵ_i on F is $[r_i, r_i + freq_i - 1]$. Remembering from BWT that l_i and f_i symbols are consecutive characters in T , and assuming that the adversary has the knowledge of the digram probability distribution of T , it is easy to guess the range of ϵ_i by traversing the possible ranges and calculating the digram distributions. The value of r_i can be any value between 1 and $n - freq_i + 1$, and for each possible interval the digram statistics of $l_{r_i}\epsilon_i$ are computed. The interval that has the closest match to the original digram probabilities is the correct one. This procedure requires $n - freq_i + 1$ digram frequency calculations. If we sum over all symbols in Σ , $\sum_{i=1..k} (n - freq_i + 1) = n \cdot (k - 1) + k$, the complexity of constructing correct F is $O(n \cdot |\Sigma|)$.

This type of a statistical attack can be applied if and only if the digram statistics of T is known or easy to guess. When the input data is homophonic as the frequency distribution of symbols is flat, the adversary cannot run such a statistical attack. Hence, *secure-BWT* can be of choice, if file F is homophonic. Else, the solution is to first turn F into a homophonic sequence, and then compute the BWT with (O) again as described in the following *compress-then-BWT* section.

B. Compress-then-BWT

If the distribution of characters in \mathcal{F} is not flat, as is the case for natural language texts, we suggest to apply first a suitable compression scheme of choice, and then take the BWT of the compressed sequence according to the selected

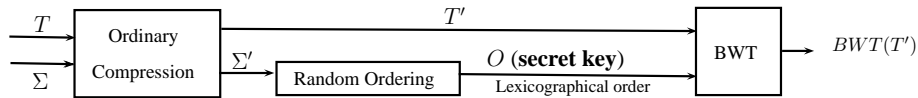


Figure 2. Compress-then-BWT solution.

secret ordering \mathcal{O} . The compression algorithms remove the redundancy in the data, and hence, the compressed data is homophonic. There is no way to apply statistical attacks on encrypted homophonic sequences and actually the encryption algorithms are deployed after compression in practical usage even if the aim does not require to compress.

In our case, if file F is not homophonic, or the alphabet size is small, or pure BWT is not powerful enough on the source data for good compression, it is proposed to use BWT after compression. This type of secure compression solution is depicted in Figure 2.

The compression algorithm may be chosen as any possible solution fitting to the underlying data. If the user wants to keep the ability to perform pattern matching on the compressed data, it would be useful to select a compression scheme on which compressed pattern matching is defined, e.g. [29], [15]. When such an algorithm is used followed by the BWT of the compressed text according to the secret \mathcal{O} , \mathcal{U} has the capability of reconstructing any part of the compressed text that gives the flexibility to run the related search procedure of the used algorithm. Note that an adversary without the knowledge of (\mathcal{O}) cannot execute the $rank_L$ and $range_F$ queries, and hence, the compressed data is kept confidential.

C. Discussion

The secure-BWT proposal corresponds to integrating compression and security in a single framework and compress-then-BWT matches with the compress-then-encrypt approaches in the literature.

When we compare secure-BWT with the works such as the interval splitting or randomized arithmetic coding, or multiple Huffman coding, the first difference is that we do not need to run a stream cipher along with the compression. Once the secret lexicographical ordering is specified, the rest of the process is the classical BWT compression without any sacrifice from compression ratio for the security. Thus, secure-BWT provides an elegant way of confidentiality in practice by removing the necessity of a concurrent cipher. The second contribution is preserving the capability to perform compressed pattern matching with methods allowing efficient search on BWT compressed text [29].

The compress-then-BWT method replaces the encryption process in compress-then-encrypt approaches with the BWT. Since BWT is a lighter operation than a modern cipher, the computational load is practically less in the proposed method. However, we need to state that a standard

encryption using cryptographic primitives provides provable security. On the other hand, the security in BWT with a secret ordering is more in the practical domain than theory. Note that this is actually the case for most of the secure compression schemes we investigated in section 2. Thus, for highly confidential data, it might be more appropriate to use an AES encryption. Unlike compress-then-encrypt methods, compress-then-BWT preserves the opportunity to run compressed pattern matching as long as the selected compression scheme supports.

V. CONCLUSIONS

This study proposed an alternative way to preserve the confidentiality of the compressed data via BWT constructed with a secret ordering of the symbols. Alphabet reordering was previously applied to enhance the compression ratio [30], but to the best of our knowledge it is the first time used to ensure the security, where a major amount of literature on secure compression had focused on preserving privacy on arithmetic coding with some limited attempts targeting Huffman or dictionary methods.

The key observation is the fact that it is not possible to reconstruct the original text nor perform any search without the correct lexicographical order used in the construction. Based on this fact, two schemes as secure-BWT and compress-then-BWT are introduced. Compress-then-BWT is especially useful in keeping the content of the compressed files hidden from the unauthorized, such as establishing the privacy of the JPEG images. Secure-BWT unifies compression and security without any loss in the efficiency of the compression. Compressed pattern matching is possible on both of the proposed techniques, where prior studies did not pay much attention. Further investigations of provably secure techniques in Burrows-Wheeler transform is an open problem, which may lead to significant contributions in practice on privacy preserving text indexing.

REFERENCES

- [1] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—the Advanced Encryption Standard*. Springer, 2002.
- [2] E. Biham and P. Kocher, “A known plaintext attack on the PKZIP stream cipher,” in *Proceedings of Fast Software Encryption’94*. Springer, 1995, pp. 144–153.
- [3] M. Stay, “ZIP attacks with reduced known plaintext,” in *Fast Software Encryption*. Springer, 2002, pp. 411–429.

- [4] T. Kohno, "Attacking and repairing the WinZip encryption scheme," in *Proceedings of the 11th ACM conference on computer and communications security*. ACM, 2004, pp. 72–81.
- [5] —, "Analysis of the winzip encryption method," Citeseer, Tech. Rep., 2004.
- [6] G. Yeo and R. Phan, "On the security of the WinRAR encryption feature," *International Journal of Information Security*, vol. 5, no. 2, pp. 115–123, 2006.
- [7] C. Wu and C. Kuo, "Design of integrated multimedia compression and encryption systems," *Multimedia, IEEE Transactions on*, vol. 7, no. 5, pp. 828–839, 2005.
- [8] H. Kim, J. Wen, and J. Villasenor, "Secure arithmetic coding," *Signal Processing, IEEE Transactions on*, vol. 55, no. 5, pp. 2263–2272, 2007.
- [9] A. Amir, "GaryBenson, and Martin Farach. Let sleeping files lie: Pattern matching in z-compressed files," in *Proceedings of the 5th ACM/SIAM Symposium on Discrete Algorithms*, 1994.
- [10] Y. Shibata, T. Kida, S. Fukamachi, M. Takeda, A. Shinohara, T. Shinohara, and S. Arikawa, "Byte Pair encoding: A text compression scheme that accelerates pattern matching," 1999.
- [11] W. Zeng and S. Lei, "Efficient frequency domain selective scrambling of digital video," in *IEEE Transactions on Multimedia*. IEEE, March 2003, vol. 5, no. 1, p. 118.
- [12] V. Govindan, "An Intelligent Text Data Encryption and Compression for High Speed and Secure Data Transmission over Internet," in *IIT Kanpur Hackers Workshop IITKHACK04*. Citeseer, 2004.
- [13] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," DEC SRC, Tech. Rep. 124, 1994.
- [14] P. Ferragina and G. Manzini, "Opportunistic data structures with applications," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, 2000, pp. 390–398.
- [15] D. Shapira and S. Klein, "Compressed pattern matching in jpeg images," *International Journal of Foundations of Computer Science*, vol. 17, no. 6, pp. 1297–1306, 2006.
- [16] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," in *Advances in Cryptology—CRYPTO 2005*. Springer, 2005, pp. 205–222.
- [17] I. Witten and J. Cleary, "On the privacy afforded by adaptive text compression," *Computers & Security*, vol. 7, no. 4, pp. 397–408, 1988.
- [18] H. Bergen and J. Hogan, "Data security in a fixed-model arithmetic coding compression algorithm," *Computers & Security*, vol. 11, no. 5, pp. 445–461, 1992.
- [19] —, "A chosen plaintext attack on an adaptive arithmetic coding compression algorithm," *Computers & Security*, vol. 12, no. 2, pp. 157–167, 1993.
- [20] X. Liu, P. Farrell, and C. Boyd, "Resisting the Bergen-Hogan attack on adaptive arithmetic coding," *Cryptography and Coding*, pp. 199–208, 1997.
- [21] J. Cleary, S. Irvine, and I. Rinsma-Melchert, "On the insecurity of arithmetic coding," *Computers & Security*, vol. 14, no. 2, pp. 167–180, 1995.
- [22] J. Lim, C. Boyd, and E. Dawson, "Cryptanalysis of adaptive arithmetic coding encryption scheme," in *Processing of ACISP*, 1997, pp. 216–227.
- [23] A. Barbir, "A methodology for performing secure data compression," in *Proceedings of the 29th Southeastern Symposium on System Theory*, 1997, p. 266.
- [24] M. Grangetto, A. Grosso, and G. Olmo, "Selective encryption of jpeg2000 images by means of randomized arithmetic coding," in *Proceedings of IEEE 6th workshop on Multimedia Signal Processing*, Siena, Italy, September 2004, pp. 347–350.
- [25] M. Grangetto, E. Magli, and G. Olmo, "Multimedia selective encryption by means of randomized arithmetic coding," *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 905–917, 2006.
- [26] J. Zhou, O. Au, and P. Wong, "Adaptive chosen-ciphertext attack on secure arithmetic coding," *Signal Processing, IEEE Transactions on*, vol. 57, no. 5, pp. 1825–1838, 2009.
- [27] J. Zhou, O. Au, X. Fan, and P. Wong, "Secure Lempel-Ziv-Welch (LZW) algorithm with random dictionary insertion and permutation," in *Multimedia and Expo, 2008 IEEE International Conference on*. IEEE, 2008, pp. 245–248.
- [28] G. Navarro and V. Mäkinen, "Compressed full-text indexes," *ACM Computing Surveys*, vol. 39, no. 1, 2007.
- [29] T. Bell, M. Powell, A. Mukherjee, and D. Adjeroh, "Searching BWT compressed text with the boyer-moore algorithm and binary search," in *Proceedings of the Data Compression Conference*, 2002.
- [30] B. Chapin and S. R. Tate, "Higher compression from the bwt by modified sorting," in *Proceedings of the Data Compression Conference*, 1998.