

A Method to Ensure the Confidentiality of the Compressed Data

M. Oğuzhan Külekci

kulekci@uekae.tubitak.gov.tr

TÜBİTAK-BILGEM-UEKAE

National Research Institute of Electronics&Cryptography, Turkey

**International Conference on
Data Compression, Communication, and Processing**
Palinuro, Italy, June 23, 2011

The problem: Securing Compression

Compression mainly helps...

- Efficient storage
- Efficient transmission

Security and Privacy in Compression

Compressed data is vulnerable to attacks as well

- in local multi-user environments
- in remote systems (e.g., clouds)

Users demand for compression combined with security!

Main Approaches

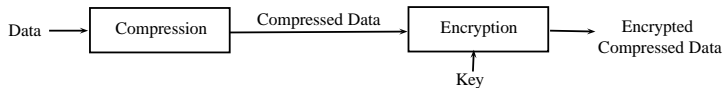
Compress–then–encrypt

Encrypt the compressed data with a standard encryption algorithm.

Unifying compression and security

Perform compression in such a way that the compressed data cannot be decoded without a secret knowledge.

Compress-then-encrypt



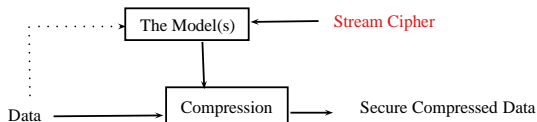
- Provides high level of secrecy (via AES, RC4, ...)
- Used by popular compression software (e.g., WinZip, WinRar, ...)
- Several attacks introduced (Biham&Kohen'94, Stay'02, Kohno'04, Yeo&Phan'06,...)
- Attacks do not break the encryption algorithm, but rather make use of improper integration between compression and encryption

Compress-then-encrypt

Some remarks..

- Additional cost of encryption algorithm (especially prohibiting in mobile environments with less computational resources)
- Lack of opportunities to perform operations on compressed data (e.g., compressed pattern matching)
 - *How about searchable encryption?*
 - Unfortunately, not yet practical*

Unifying compression and security



Securing Arithmetic Coding

- Semi-adaptive time-variant modeling (Barbir'97)
- Randomized AC: shuffle the intervals (Grangetto'04)
- Interval splitting: shuffle + split (Kim *et al.*'07)

Securing Huffman Coding

- Multiple code-trees (Wu&Kuo'05)

Securing Dictionary-Based Coding

- Secure LZW compression (Zhou'08)

Unifying compression and security

Some remarks...

- Requires a concurrent stream cipher
- Sacrifice of compression ratio due to alterations in the original schemes
- Lack of opportunities to perform operations on compressed data (e.g., compressed pattern matching)

Motivation

- There are studies to secure AC, HC, and dictionary based LZ schemes
- Securing BWT has not been addressed yet...
- Can we do something to ensure the confidentiality of the compressed data via BWT

Burrows–Wheeler Transform

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

Cyclic right shifts

Sorted CRS

Burrows–Wheeler Transform

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

$$BWT(T) = L$$

Burrows–Wheeler Transform

	1	2	3	4	5	6	7	8	9	10	11	12
$T =$	m	i	s	s	i	s	s	i	p	p	i	\$

	1	2	3	4	5	6	7	8	9	10	11	12
	t_{12}	t_{11}	t_8	t_5	t_2	t_1	t_{10}	t_9	t_7	t_4	t_6	t_3
$F =$	\$	i	i	i	i	m	p	p	s	s	s	s
$L =$	i	p	s	s	m	\$	p	i	s	s	i	i
	t_{11}	t_{10}	t_7	t_4	t_1	t_{12}	t_9	t_8	t_6	t_3	t_5	t_2

Important properties of BWT

- F is the lexicographically sorted character array of L .

Burrows–Wheeler Transform

	1	2	3	4	5	6	7	8	9	10	11	12
$T =$	m	i	s	s	i	s	s	i	p	p	i	\$

	1	2	3	4	5	6	7	8	9	10	11	12
	t_{12}	t_{11}	t_8	t_5	t_2	t_1	t_{10}	t_9	t_7	t_4	t_6	t_3
$F =$	\$	i	i	i	i	m	p	p	s	s	s	s
$L =$	i	p	s	s	m	\$	p	i	s	s	i	i
	t_{11}	t_{10}	t_7	t_4	t_1	t_{12}	t_9	t_8	t_6	t_3	t_5	t_2

Important properties of BWT

- F is the lexicographically sorted character array of L .
- f_j is the next character of l_j in T : $l_j f_j \rightarrow t_z t_{z+1}$.

Burrows–Wheeler Transform

	1	2	3	4	5	6	7	8	9	10	11	12
$T =$	m	i	s	s	i	s	s	i	p	p	i	\$

	1	2	3	4	5	6	7	8	9	10	11	12
	t_{12}	t_{11}	t_8	t_5	t_2	t_1	t_{10}	t_9	t_7	t_4	t_6	t_3
$F =$	\$	i	i	i	i	m	p	p	s	s	s	s
$L =$	i	p	s	s	m	\$	p	i	s	s	i	i
	t_{11}	t_{10}	t_7	t_4	t_1	t_{12}	t_9	t_8	t_6	t_3	t_5	t_2

Important properties of BWT

- 1 F is the lexicographically sorted character array of L .
- 2 f_i is the next character of l_i in T : $l_i f_i \rightarrow t_z t_{z+1}$.
- 3 The i^{th} occurrence of a character in L corresponds to same text position in T with its i^{th} occurrence in F .

Reconstructing Text from BWT

T = ?

	F	L
1		i
2		p
3		s
4		s
5		m
6		\$
7		p
8		i
9		s
10		s
11		i
12		i

Reconstructing Text from BWT

$T = \dots i$

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

1 Compute F from L .

2 $t_n = l_1 = i$

Reconstructing Text from BWT

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

$$T = \dots \mathbf{p}i$$

- 1 Compute F from L .
- 2 $t_n = l_1 = i$
- 3 $rank_L(1, i) = 0$, $range_F(i) = [2, 5]$
 $L \rightarrow F : l_1 \rightarrow f_2, t_{n-1} = l_2 = p$

Reconstructing Text from BWT

$$T = \dots ppi$$

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

- 1 Compute F from L .
- 2 $t_n = l_1 = i$
- 3 $rank_L(1, i) = 0$, $range_F(i) = [2, 5]$,
 $L \rightarrow F : l_1 \rightarrow f_2, t_{n-1} = l_2 = p$
- 4 $rank_L(2, p) = 0$ and $range_F(p) = [7, 8]$
 $L \rightarrow F : l_2 \rightarrow f_7, t_{n-2} = l_7 = p$

Reconstructing Text from BWT

$$T = \dots\text{ippi}$$

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

- 1 Compute F from L .
- 2 $t_n = l_1 = i$
- 3 $\text{rank}_L(1, i) = 0$, $\text{range}_F(i) = [2, 5]$,
 $L \rightarrow F : l_1 \rightarrow f_2, t_{n-1} = l_2 = p$
- 4 $\text{rank}_L(2, p) = 0$ and $\text{range}_F(p) = [7, 8]$
 $L \rightarrow F : l_2 \rightarrow f_7, t_{n-2} = l_7 = p$
- 5 $\text{rank}_L(7, p) = 1$ and $\text{range}_F(p) = [7, 8]$
 $L \rightarrow F : l_7 \rightarrow f_8, t_{n-3} = l_8 = i$

Reconstructing Text from BWT

$T = \text{mississippi}$

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

- 1 Compute F from L .
- 2 $t_n = l_1 = i$
- 3 $rank_L(1, i) = 0$, $range_F(i) = [2, 5]$,
 $L \rightarrow F : l_1 \rightarrow f_2, t_{n-1} = l_2 = p$
- 4 $rank_L(2, p) = 0$ and $range_F(p) = [7, 8]$
 $L \rightarrow F : l_2 \rightarrow f_7, t_{n-2} = l_7 = p$
- 5 $rank_L(7, p) = 1$ and $range_F(p) = [7, 8]$
 $L \rightarrow F : l_7 \rightarrow f_8, t_{n-3} = l_8 = i$
.....
- 12 $rank_L(12, i) = 3$ and $range_F(i) = [2, 5]$
 $L \rightarrow F : l_{12} \rightarrow f_5, t_{n-3} = l_5 = m$

Backwards Searching

Search P = sip

	F	L
1		i
2		p
3		s
4		s
5		m
6		\$
7		p
8		i
9		s
10		s
11		i
12		i

Backwards Searching

Search $P = sip$

1 Compute F from L .

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

Backwards Searching

Search $P = \text{si}\underline{p}$

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

- 1 Compute F from L .
- 2 $\text{range}_F(p) = [7, 8]$
Candidate Interval = $[7, 8]$

Backwards Searching

Search $P = \text{sip}$

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

- 1 Compute F from L .
- 2 $\text{range}_F(p) = [7, 8]$
Candidate Interval = $[7, 8]$
- 3 $\text{rank}_L(7, i) = 1$, $\text{rank}_L(9, i) = 2$, $\text{range}_F(i) = [2, 5]$

Backwards Searching

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

Search P = sip

- 1 Compute F from L .
- 2 $range_F(p) = [7, 8]$
Candidate Interval = $[7, 8]$
- 3 $rank_L(7, i) = 1$, $rank_L(9, i) = 2$, $range_F(i) = [2, 5]$
Candidate Interval = $[2 + 1, 2 + 2 - 1] = [3, 3]$

Backwards Searching

Search $P = \underline{s}ip$

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

- 1 Compute F from L .
- 2 $range_F(p) = [7, 8]$
Candidate Interval = $[7, 8]$
- 3 $rank_L(7, i) = 1$, $rank_L(9, i) = 2$, $range_F(i) = [2, 5]$
Candidate Interval = $[2 + 1, 2 + 2 - 1] = [3, 3]$
- 4 $rank_L(3, s) = 0$, $rank_L(4, s) = 1$, $range_F(s) = [9, 12]$

Backwards Searching

	F	L
1	\$	i
2	i	p
3	i	s
4	i	s
5	i	m
6	m	\$
7	p	p
8	p	i
9	s	s
10	s	s
11	s	i
12	s	i

Search $P = \underline{\text{sip}}$

- 1 Compute F from L .
- 2 $\text{range}_F(p) = [7, 8]$
Candidate Interval = $[7, 8]$.
- 3 $\text{rank}_L(7, i) = 1$, $\text{rank}_L(9, i) = 2$, $\text{range}_F(i) = [2, 5]$
Candidate Interval = $[2 + 1, 2 + 2 - 1] = [3, 3]$.
- 4 $\text{rank}_L(3, s) = 0$, $\text{rank}_L(4, s) = 1$, $\text{range}_F(s) = [9, 12]$
Final Interval = $[9 + 0, 9 + 1 - 1] = [9, 9]$.

$P = \text{sip}$ occurs in $T = \text{mississippi}$ once.

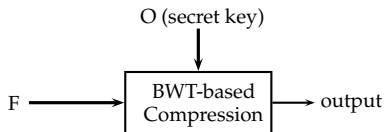
Functions in Reconstruction and Backwards Search

- $ComputeF$: Sort the characters in L according to the lexicographic ordering.
- $Range_F(c)$: The beginning and ending positions of character c in F .
- $Rank_L(i, c)$: Number of c characters observed up to position i in L . Note that it can be answered in $O(\log |\Sigma|)$ time via a wavelet tree constructed over L .

Key Observation

$ComputeF$ and $range_F$ functions require the knowledge of lexicographical ordering of the alphabet.

sBWT: Scrambled-BWT



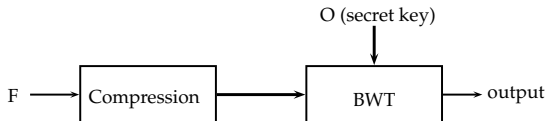
- Compute BWT of input F with a secret lexicographical ordering \mathcal{O} .
- $|\Sigma|!$ possible distinct orderings.
- Large key space, when Σ is sufficiently large, e.g., $20! > 2^{61}$, $25! > 2^{83}$, $100! > 2^{526}$.
- Possible to perform pattern search on BWT-compressed text (Bell *et al.* DCC'02).
- Small alphabets ? \rightarrow represent F with a super-alphabet, and use geometric-BWT (Chien *et al.* DCC'08) for effective search

sBWT

A statistical attack...

- A malicious adversary \mathcal{M} can run a statistical attack, when
 - F is not homophonic and 2-gram statistics of F are available
 - There exists an intelligibility criteria to decide on the original text
- Instead of $|\Sigma|!$ trials, it might be possible to capture \mathcal{O} in $n \cdot |\Sigma|$ steps
- \mathcal{M} makes use of the property that $l_i f_i$ pair in BWT is an ordinary 2-gram of T (please see the paper for details).

Compress-then-sBWT



- Compression removes redundancy, so F becomes homophonic with 256 character alphabet in practice.
- Thus, more safe against the statistical attack
- Still preserves the ability to run compressed pattern matching unless the preferred compression scheme supports.

Conclusions

- sBWT ← unifying compression and security in BWT
- Compress-then-sBWT ← alternative to compress-then-encrypt
- Benefits :
 - No need to include encryption algorithms in compression software
 - Better handling of pattern matching in **compressed and secured** data

Conclusions

- An initial attempt to secure BWT, a major component in compression and text indexing
- Compressed pattern matching on **secure compressed** data
- Some immediate applications :
 - Integration into bzip2 package as an additional security option ?
 - Securing the lossless JPEG images via a second layer sBWT ?

Thanks...