암호화 기능을 가진 EDI 메시지의 효율적인 압축 기법에 관한 연구

정 옥진 김 정희 윤 성현 김 태윤
고려대학교 전산과학과

A Study on the Efficient Compression Method of EDI Message with Encryption

Ok-Jin Jeong Jung-Hee Kim Sung-Hyun Yun Tai-Yun Kim
Dept. of Computer Science, Korea University

요 약

현재 통신 자료의 비밀을 위해 여러가지 암호화 방법들이 사용되고 있으며 메시지의 양을 줄이기 위해 다양한 압축 기법들이 활용되고 있다. 그러나 기존에는 암호화와 압축이 서로 독립된 기능으로 구현되어 사용되고 있다.

본 논문에서는 대표적인 압축 기법인 LZW 계열의 알고리즘에 EDI 메시지의 특성을 반영한 새로운 압축 기법을 제안한다. EDI 메시지의 세그먼트 식별자를 사전으로 구성함으로써 압축 효율을 향상시켰다. 또한 통신 자료의 보다 안전한 보호를 위해 암호화 기능을 결합하였다.

1. 서론

통신 정보의 보호는 최근 정보화 사회에서 가장 중요한 문제로 등장하고 있다. 정보의 신속한 교환이라는 측면에서 첨단 정보 통신 기술의 발전은 우리 사회에 획기적인 전환점을 제공해 주었다. 그러나, 지역적으로 분리되어 있는 컴퓨터 네트워크의 특성상 전송중인 디지탈 자료를 안전하게 보호하는 것은 쉬운 일이 아니다. 또한 각종정보 통신 서비스가 증가함에 따라 우리가 저장하고 관리해야 하는 정보의 양도 기하급수적으로 증가하고 있다. 따라서 많은 양의 정보를 보다 안전하고 효율적으로 전송하기 위한 기법이 활발하게 연구되고 있다.

이러한 추세에 힘입어 예전에는 자료의 압축과 암호화가 독립적으로 연구되어 왔지만 지금은 이들 두가지가 서로 결합되어 통신의 효율성과 자료의 안전성을 높이고 있으며 새로운 연구 분야로 자리잡기 시작했다. 그러므로 압축과 암호화를 독립적으로 수행하는 오버헤드가 줄어들고 더 강력하고 효율적인 기능을 제공하게 되었다.

EDI(Electronic Data Interchange)란 표준화된 포맷과 양식을 이용하여 기업 또는 공공 기관 사이의 거래 문서를 컴퓨터 통신을 이용해서 교환하는 새로운 통신 기술이다. 본 논문에서는 EDI 메시지를 교환함에 있어서 EDI 메시지의 압축과 암호화 기법에 대한 내용을 다룬다.

EDI 메시지는 텍스트 자료이므로 압축 또는 암호화를 수행하는 도중에 자료의 변경이나 손실이 발생해서는 안된다. 만약 음성 정보나 그래픽 정보라면 어느정도의 오차를 허용할 수도 있지만 텍스트 자료는 압축과 암호화 과정 전후에 자료의 내용과 형태가 절대적으로 동일해야 한다.

압축이란 자료를 저장하는 데 필요한 공간을 줄이고 전송할 때 시간을 줄이는 방법이다. 따라서 자료의 압축은 사람이 읽을 수 있는 원문에서 중복성을 최대한 제거하고 원문과 같은 의미를 유지하도록 하면서 원문의 양을 줄이게 된다. 즉 전송되는 도중에 자료는 굳이 사람이 눈으로 읽어서 판독할 필요가 없다.

기존의 압축 기법들이 이미 많이 소개되어 있다. 그러나 이러한 기법들은 EDI가 도입되기 전에 또는 활성화되기 전에 개발된 기법들이므로 EDI 메시지에 최적의 압축율을 제공한다는 보장은 할 수 없다.

여러가지 압축 기법 중 Ziv와 Lempel이 제안한 LZ 압축 기법은 많은 응용 영역을 가지고 있으며 특히 텍스트 압축에 널리 사용되고 있다. 이 방법은 Ziv와 Lempel이 1977년에 처음 제안하였고 1984년에 Welch가 실용화시켰으며 보통 LZW 압축 기법이라고 부른다.

본 논문에서는 기존의 LZW 방식에다 사전 기법을 결합한 새로운 압축 기법을 개발하여 제시하였다. 사전은 EDI 메시지에서 자주 사용되는 세그먼트 식별자를 문자열의 앞 부분에 할당함으로써 구성한다. 따라서 사전에 나타난 세그먼트 식별자만큼의 압축이 추가적으로 수행된다. 또한 압축을 수행하면서 동시에 암호화를 수행할 수 있도록 알고리즘을 구성하였다.

2. 압축 모델

압축은 컴퓨터 화일을 저장하는 데 필요한 공간을 줄이고 주어진 대역폭의 채널을 통해서 정보를 전송하는 시간을 줄일 수 있는 방법이다. 이것은 부호화의 형태를 취한다. 부호화의 또다른 목적은 에러의 발견 및 수정과 암호화이다. 에러 발견 및 수정의 과정은 압축의 반대 과정이다. 즉 데이타가 나중에 검사될 수 있도록 데이타에 중복성(redundancy)을 더한다. 압축의 목적은 인간이 읽을 필요가 없을 때 데이타로부터 중복성을 제거하는 것이다. 압축은 텍스트로부터 중복성을 제거하고 침입자에게 유용하게 쓰일 수 있는 통계적 정보와 수단을 줄임으로써 암호화에도 기여한다.

가장 초기에 잘 알려졌던 압축 기법 중의 하나가 Huffman 알고리즘이고, 아직도 많은 연구의 주제가 되고 있다. 그러나, 1970년대 후반에 와서 두가지 중요한 계기로 인해 자리를 거의 빼앗기고 있다. 그 하나가 산술 부호화 기법이다. 이 방법은 Huffman 방식과 유사하지만 더 효율적인 압축을 수행한다. 나머지 하나는 LZ 압축이다. 이 방법은 Huffman 코딩이나 산술 부호화 기법과는 다른 방법을 사용한다. 두가지 압축 방법 모두가 처음 발표되었을 때보다 세련화되고 개선되어 높은 효율을 가지도록 실용화되어 있다.

압축을 수행하는 두가지 일반적인 방법은 통계적 기법과 사전 기법이다. 가장 좋은 통계적 기법은 산술 부호화 기법이고 사전 기법 중에는 Ziv-Lempel 코딩이 좋다. 통계적 압축 기법에서 각 심볼들은 그들이 나타나는 확률에 기초하여 부호어를 할당받는다. 높은 확률의 심볼은 짧은 부호어를 가지며 확률이 낮은 심볼은 긴 부호어를 가지게 된다. 사전 압축 기법에서는 연속되는 문자의 그룹 또는 어절(phrase)을 '사전'에서 검색함으로써 발견할 수 있다.

자료 압축 과정은 두 부분으로 분리될 수 있다. 즉 압축된 비트열을 실제로 생성하는 엔코딩과 정보를 건네주는 모델링으로 이루어진다.

모델링이란 심볼에 확률을 할당하고 코딩은 이 확률을 비트열로 변환시킨다. '코딩'이란 용어는 가끔 모델링을 포함하는 광범위한 의미로 언급되기 때문에 혼동이 생길 수 있다. 광의의 코딩과 협의의 코딩(주어진 모델에 따라 비트열을 생성하는)에는 차이가 있다.

디코더(decoder)는 원문을 정확하게 얻기 위해 압축을 풀 때에도 압축을 할 때와 같은 모델에 접근해야 한다. 이를 달성하기 위한 방법은 정적(static), 반적응적(semi-adaptive), 적응적(adaptive) 모델링의 세가지가 있다.

정적 모델은 모든 텍스트에 같은 모델을 사용한다. 압축될 텍스트 형태의 예제로부터 엔코드가 시작될 때, 적절한 모델이 결정된다. 그 모델의 동일한 복사본이 디코더에 유지된다. 이 방법은 부호화될 텍스트가 모델과 일치하지 않을 때마다 매우 나쁜 압축률을 가져오는 단점이 있으므로, 속도와 단순성이 중요할 때에만 사용된다.

반적응적 모델링은 부호화되는 각 텍스트에 대해 다른 모델을 사용함으로써 이 문제를 해결한다. 압축을 수행하기 전에 텍스트를 미리 읽어서 그 결과를 가지고

모델을 구성한다. 압축된 텍스트가 보내지기 전에 디코더에게 그 모델이 전송되어야 한다. 모델을 전송하는 추가 비용에도 불구하고, 텍스트에 잘 맞는 모델의 선택으로 인해 일반적으로 전체 비용이 줄어들 것이다.

적응적(동적) 모델링은 모델의 전송 오버헤드를 피할 수 있다. 처음에, 엔코더와 디코더는 모든 문자가 같은 확률이라는 혼합 모델을 가정한다. 엔코더는 한 심볼을 전송하기 위해 적응적 모델을 이용하고 디코더는 심볼을 디코드하기 위해 적응적 모델을 사용한다. 엔코더와 디코더는 적응적 모델을 같은 방법(관찰되는 심볼의 확률을 증가시킴으로써)으로 갱신한다. 다음 심볼을 전송하고 이 심볼은 새로운 모델에 의해 디코드되며 모델을 다시 갱신한다. 이런 방법으로 부호화를 계속하면 모델을 갱신하는데 같은 알고리즘을 사용하므로 전송에러가 발생하지 않는다면, 디코더는 엔코더와 동일한 모델을 유지한다. 사용하고 있는 모델은 압축되는 텍스트에 잘 맞는 것이므로, 명시적으로 모델을 전송할 필요가 없는 것이다.

3. 사전 기법

사전 압축 기법은 통계적 압축 기법과는 기본적으로 다르다. 사전 기법은 연속적인 문자열(어절) 그룹을 사전의 색인을 사용해서 치환시킴으로써 압축을 수행한다. 사전은 자주 나타날 것으로 기대되는 어절의 목록으로 구성된다. 색인은 부호화될 어절보다 평균적으로 공간을 덜 차지하도록 선택되며 이 부호어로서 압축을 수행한다.

사전의 색인은 여러 문자의 입력 심볼에 대해 하나의 출력 부호어이므로, 사전 방법은 기계어로 정렬되도록 부호어를 선택할 수 있으므로 보통 빠르다. 사전 모델은 대개 아주 좋은 압축을 수행한다. 대부분의 사전 기법은 유한 문맥 (finite-context) 모델의 한 형태로 모의 실험할 수 있어서 주요 장점은 압축 효율보다 계산 비용(computing resource)의 경제성이라고 할 수 있다.

사전 압축 기법의 설계에서 중요한 선택 사항은 코딩 사전에 포함시키는 표제어(entry)의 선택이다. 일부 설계자들은 저장된 어절의 길이에 제한을 두기도 한다. 예를 들어 digram 코딩에서는 결코 두 문자를 넘지 않는다. 이런 범위내에서, 어절의 선택은 정적, 반적응적 또는 적응적 알고리즘에 의해 이루어진다. 가장 간단한 사전 압축 기법은 단지 짧은 어절을 포함하는 정적 사전을 사용하는 것이다.

일단 사전이 선택되면, 입력 텍스트에 있는 어느 어절를 사전으로의 색인으로 치환한다. 부호화를 하기 위해 텍스트를 어절들로 나누는 것을 'parsing'이라고 부른다. 가장 빠른 방법은 경험적 파싱(greedy parsing)인데, 각 단계에서 엔코더는 텍스트에서 다음 문자와 일치되면 그들을 부호화하기 위해 그에 일치하는 색인을 사용해서 사전에서 가장 긴 문자열을 검색한다.

불행하게도 경험적 파싱이 항상 최적이지는 않다. 엔코더가 어느 정도의 앞을 전망해야 하는지에 대한 제한이 없으므로 최적 파싱을 결정하는 것은 의외로 어려워진다. 경험적 접근법이 최적은 아닐지라도, 정해진 지연시간(delay) 내에 텍스트를 한 번 읽어서 부호화를 수행하므로, 실제로는 널리 사용되었다.

정적 사전 기법은 매우 적은 양의 노력으로 적은 압축을 수행하고자 할 때 유용하다. 다른 형태로 여러번 제안된 빠른 알고리즘의 하나는 digram 코딩인데, 일반적으로 사용되는 digram의 사전 또는 문자쌍을 유지하는 방법이다. 부호화 단계에서 다음 두 문자를 사전에 있는 digram과 일치하는지를 알아보기 위해 조사한다. 만약 일치한다면 그들은 함께 부호화되고, 그렇지 않으면 첫번째 글자만이 부호화된다. 따라서 부호화되는 위치는 한 문자 또는 두 문자씩 적절하게 진행되어 진다.

4. 적응적 사전 압축 기법(Ziv-Lempel Compression)

거의 모든 실용적인 적응적 사전 엔코더들은 Ziv와 Lempel의 연구로부터 유도된 알고리즘의 한 부류에 포함된다. 이 방법의 핵심은 어절이 나왔을 때 텍스트의 앞부분에 나타났던 같은 어절로의 포인터를 대치한다는 것이다. 이런 종류의 알고리즘을 Ziv-Lempel 압축 또는 줄여서 LZ 압축이라고 부른다. 이 방법은 새로운 분야에 빠르게 적용할 수 있으며, 또한 매우 자주 나타나는 짧은 기능 단어를 부호화할 수 있다. 새로운 단어와 어절을 또한 앞선 단어들의 일부로부터 새로 구성할 수도 있다.

이런 방법으로 압축된 텍스트의 디코딩은 간단하다. 디코더는 단지 포인터가 가리키고 있는 이미 디코드된 텍스트의 포인터를 대치하면 된다. 실제로 LZ 부호화는 좋은 압축율을 보이며 디코딩이 매우 빠르다는 것이 주요 특징이다.

LZ78은 적응적 사전 압축의 새로운 접근 방식이고 이론적, 실용적 관점에서 중요하다. 이 방법은 LZ77계열과 병행하여 발전한 기법들 중의 하나이다. 이전에 출현했던 어떤 문자열을 참조하는 포인터를 사용하는 대신, 지금까지 보았던 텍스트를 어절로 parse한다. 여기서 각 어절은 이전에 일치되었던 가장 긴 어절에다 한 문자를 더한 것이다. 각 어절은 접두사(prefix)를 가리키는 가장 긴 색인으로 부호화되고 여분의 한문자를 더한다. 그리고 나서 새로운 어절은 참조될 어절 목록에 추가된다.

입 력	:	a	aa	b	ba	baa	baaa	bab
문자열 번호	:	1	2	3	4	5	6	7
출 력	:	(0,a)	(1,a)	(0,b)	(3,a)	(4,a)	(5,a)	(4,b)

<그림 1> LZ78 방식을 사용한 문자열
"aaabbabaabaaabab"에 대한 부호화

예를 들어, 문자열 "aaabbabaabaaabab"는 <그림 1>에서와 같이 7개의 어절로 나누어진다. 여기서 (i,a)는 현재의 문자열이 문자열 번호 i에 기록된 문자열 뒤에 a가 뒤따른다는 것을 의미한다. 각각은 이전에 나타났던 어절로 부호화되고, 명시적인 문자를 덧붙인다. 마지막 세 문자(bab)는 어절 번호 4("ba")와 뒤따르는 문자 "b"를 사용해서 부호화된다. 어절 번호 0은 빈 문자열이다.

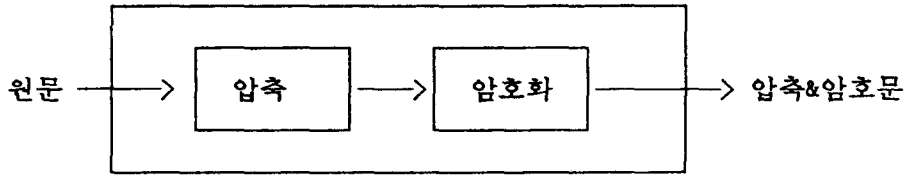
LZ78의 이론적인 중요한 특징은 입력 텍스트가 생성될 때, 입력의 크기가 증가함에

따라 압축율이 최적에 가까와진다는 것이다. 즉 LZ78은 입력 텍스트의 엔트로피(entropy)에 의해 표시된 최소 크기로 무한히 긴 문자열을 부호화할 것이다. 그러나, 입력의 크기가 무한대에 가까와지면 최적 효율을 얻을 수 있지만 대부분의 경우 텍스트는 이보다 훨씬 짧다.

각 어절 이후의 출력에 명시적인 문자를 포함하는 것은 낭비가 된다. LZW는 이런 문자들을 모두 제거해서 출력이 포인터만을 포함하도록 한다. 이 방법은 입력 알파벳에 있는 모든 문자를 포함하도록 어절의 목록을 초기화함으로써 시작된다. 새로운 어절의 마지막 문자는 다음 어절의 첫번째 문자가 되도록 부호화된다. 디코드하는 동안 바로 앞의 어절을 사용해서 어절이 부호화된 경우에는 특별한 주의가 필요하지만, 그것이 해결 불가능한 문제는 아니다.

5. 암호화

기존의 압축 및 암호화의 수행 방법은 형태는 자료에 대해서 압축을 먼저 수행하고 나서 다시 암호화를 수행하였다. 즉 압축과 암호화가 각각 독립적인 기능으로 구현되었으므로 디스크에서 화일을 읽어서 압축을 수행한 다음 디스크에 쓰고 다시 읽어서 암호화를 수행하고 디스크에 쓰는 방식이었다. 그러나 본 논문에서는 자료를 디스크로부터 읽는 시간과 디스크로 쓰는 데 소요되는 시간을 줄일 수 있도록 압축과 암호화를 결합하였다. 또한 원문을 그대로 암호화하는 것이 아니라 압축문을 암호화하므로 암호화에 걸리는 시간을 줄였다.



〈그림 2 〉 압축 및 암호화 순서

암호화에서 가장 중요한 이슈중의 하나는 역시 키의 분배 문제이다. 암호화된 자료의 안전한 보관과 전송을 위해서는 키는 길수록 좋다. 그러나 키의 길이가 길면 키의 분배에 어려움이 가중되므로 키의 길이는 안전성과 키분배의 상반관계를 고려하여 결정해야 한다.

본 논문에서는 키의 분배가 필요하지 않는 자동키 방법을 채택했다. 따라서 이 방법은 키가 암호문 자체에 포함되어 있거나 이미 알려진 상태가 될 수도 있으므로 매우 위험한 방법이 될 수도 있다. 즉, 공격자가 쉽게 키를 얻어서 암호문을 해독할 수 있게 된다.

그러나 본 논문에서는 키의 분배가 안전하게 수행되었다는 가정하에서 암호화를 수행하기 위해서 키의 분배가 필요없는 자동키 방법을 사용했다. 또한 자동키 방법은 미리 주어지는 초기키를 사용하므로 암호문과는 별도의 키를 소유할 수도 있다.

따라서 키의 관리가 완전하다면 자동키 방식도 안전하다고 볼 수 있다.

압축을 수행하는 것만으로도 이미 암호화의 효과를 가지고 있으므로 연속적으로 수행하는 암호화에는 처리 시간의 축소를 위해 다음과 같은 OR 연산으로 암호화를 수행한다.

```
while (1) do
 {
   key = init_key ;
   for ( i=0 ; i < length of key ; i++)
     {
       current = input() ;
       if ( current == EOF ) break ;
       else
         {
           encrypt = current + key ;
           output( encrypt ) ;
           key = encrypt ;
         }
     }
 }
```

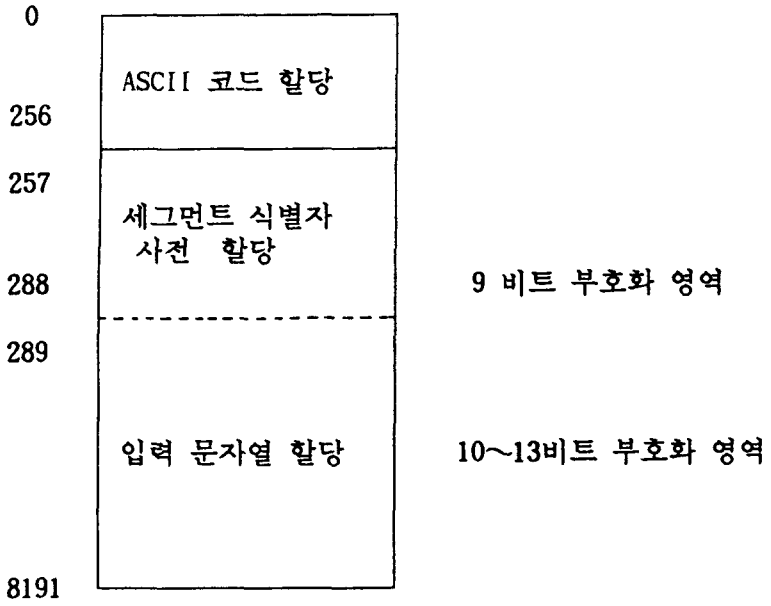
< 그림 3 > 암호화 알고리즘

위의 알고리즘에서 input()은 압축이 끝난 결과를 말하며, EOF의 검사는 입력 화일에서 마지막 문자를 압축하였는지를 검사한다. 그리고 output(encrypt)은 암호화된 결과를 화일에 쓰는 기능이다. 암호화를 수행하는데 사용하는 키는 key이며 이 키는 키의 길이만큼 암호화를 행하고 나서 init_key로 초기화를 행한다.

6. EDI 메시지를 위한 새로운 압축 기법의 설계

EDI 메시지에서는 UNH, UNB, BGM, LIN 등과 같이 3문자로 구성된 세그먼트 식별자가 사용된다. 이 식별자는 EDI 메시지 표준으로 구성되어 있으며, 거래되는 모든 EDI 표준 포맷에는 이런 세그먼트 식별자가 자주 반복된다.

본 논문에서 제안한 방법은 LZ 계열의 압축 기법과 더 긴 문자열을 사전의 표제어로 하는 사전 기법을 접목시켰다. 즉 EDI 메시지에서 사용되는 세그먼트 식별자 32개를 표제어로 한다. 각 표제어에 대해서 257번부터 288번까지의 부호어를 할당한다. 그리고 289번부터는 압축하고자 하는 입력 메시지에 나타나는 문자열에 대한 부호어를 할당하는 데 사용된다.



<그림 4> 부호어의 할당

문자열 테이블의 크기는 8192로 하였으며 테이블이 가득 차면 테이블을 소거하고 다시 작성하도록 하였다.

알고리즘의 수행 절차는 다음과 같다.

1. ASCII 코드 값으로 테이블을 초기화한다.
2. 257번부터는 세그먼트 식별자를 할당한다.
3. 입력 문자를 읽어 스트링 s로 한다.
4. 다음 과정을 입력이 끝날 때까지 반복한다.
   ① 다음 문자를 읽어서 c에 치환한다.
   ② 만일 c가 세그먼트 식별자 태그라면 세그먼트 사전을 검색한다.
   만약 발견되면 그에 해당하는 9비트의 부호어를 할당하고 4로 간다.
   ③ 테이블에서 문자열 sc를 검색한다.
   ④ 만일 문자열이 발견되면 s에 테이블의 주소를 할당하고 4로 간다.
   ⑤ 만일 발견되지 않는다면 문자열 sc를 테이블에 기록하고 s에 대한
   부호어를 출력한다.

압축을 풀 때에는 위의 알고리즘을 역순으로 적용하면 된다. 다음 표는 위에서 제안한 압축 기법과 사전을 도입하지 않은 기법을 사용하지 않은 기존의 LZ 기법의 압축율을 비교한 것이다. 입력 화일은 모두 EDI 메시지를 대상으로 하였다.

$$\text{압축율} = \frac{\text{압축 후의 크기}}{\text{압축 전의 크기}} * 100\ (\%)$$

[표 1] 메시지 크기별 압축율

원문의 크기 (바이트)	기존의 방법		제안한 방법	
	압축 결과	압축율(%)	압축율(%)	압축 결과
1031	685	66.44	57.41	592
2004	1244	62.07	53.79	1078
3036	1694	55.79	48.15	1462
4066	2104	51.74	44.09	1793
5041	2540	50.38	42.33	2134
10081	4295	42.60	35.59	3588
15176	6193	40.80	31.19	4734

7. 결론

정보화 사회에서 컴퓨터 통신은 중요한 비중을 차지하며 많은 양의 자료를 효율적이고 안전하게 전송해야 한다. 컴퓨터 네트워크를 통해서 전송하는 자료의 양은 통신 비용과 직접 연결되므로 기업 활동에서 EDI 시스템을 사용할 때 자료의 압축은 생산성 향상과 비용 절감을 위해서 필수적인 과정이 된다. 또한 거래처에 민감한 자료들은 보안을 유지할 필요가 있으며, 이를 위해서는 암호화가 필요하다.

본 논문에서는 향후 거래량이 급증하게 될 EDI 메시지의 효율적인 압축 기법과 거래 자료의 안전을 위한 암호화 기법을 결합한 알고리즘을 설계하고 구현하였다.

본 논문에서 새롭게 압축 방법은 기존의 LZW 방법을 수용하고 EDI 메시지에서 빈번하게 사용되는 세그먼트 식별자를 표제어로 하는 사전을 구성하여 압축을 보다 효율적으로 수행하였다. 사전의 각 표제어는 문자열 저장 테이블의 시작 부분에 배치하고, 각 표제어에 9비트의 부호어를 할당하였다. 또한 이 표제어는 압축이 진행됨에 따라 문자열 저장 테이블에 기록되면서 더 긴 어절의 일부로 사용된다. 이 기법이 사전을 사용하지 않고 각 문자를 하나의 심볼로 처리하는 기존의 LZ 압축 기법보다 높은 압축율을 가지고 있다는 것을 실험을 통하여 증명하였다. 또한 압축이 끝난 직후 곧바로 암호화를 수행함으로써 화일 입출력에 소비되는 시간을 절약할 수 있고 암호화의 강도를 높이게 되었다. 여기서 암호화를 하는 데 걸리는 추가 시간은 무시될 정도로 거의 없었다.

사전 압축 기법에서는 무엇보다도 사전의 선택이 압축율에 큰 영향을 주기 때문에 향후 연구 과제로서는 압축율을 최대로 할 수 있는 사전의 크기를 조절하는 것과 한글이 포함된 EDI 메시지에 적합하도록 알고리즘을 개선하는 것이다.

EDI 메시지에서는 세그먼트 식별자 이외에도 데이타 엘리먼트에 입력되는 값도 표준에 규정된 코드를 사용한다. 따라서 이 코드값도 사전으로 구축할 경우 좀 더 높은 압축율을 얻을 것으로 기대된다. 또한 이 때 사전의 크기가 무한히 커질 수는

없으며, 사전의 크기가 커지면 오히려 입력 문자열에 대한 적응력이 약해지므로 압축율은 감소할 수도 있으므로 최적의 압축율을 얻을 수 있도록 사전의 크기를 조절해야 한다.

본 논문은 향후 EDI 시스템의 보급과 확산시에 많은 통신 비용을 절감할 수 있을 것으로 기대되며, 암호화 기능이 있으므로 보안 서비스의 기능도 수행할 수 있을 것이다.

<참 고 문 헌>

[1] Timothy Bell, Iva H. Witten, John G. Cleary, "Modeling for Text Compression," ACM Computing Surveys, Vol. 21, No. 4, December 1989.

[2] Fred Halsall, "Data Communications, Computer Networks and Open Systems," Third Edition, Addision-Wesley, 1992.

[3] 박 지환, "Ziv-Lempel 부호와 그 응용에 관한 고찰", 통신정보보호학회지 제 2권, 제 2호, 1992. 6.

[4] 김정희, 김태윤, "EDI 보안과 통제 관리에 관한 연구", 한국통신정보보호학회 논문지, 1991. 11.

[5] 김태윤, 전자거래정보교환(EDI), 집문당, 1991.

[6] 김 태윤, 데이타 통신과 컴퓨터 통신, 집문당, 1990

[7] 조광문, 김태윤, "공중망을 이용한 마이크로 컴퓨터용 EDI 변환 처리 시스템에 관한 연구," 전산활용연구, 제 4 권, 제 1 호, 1991. 12.

[8] 정옥진, 김태윤, "EDI를 위한 메시지 처리 시스템의 설계 및 구현에 관한 연구," 전산활용연구, 제 4 권, 제 1 호, 1991. 12.

[9] 조광문, 김태윤, "EDI를 이용한 기업간 문서 거래 시스템의 설계에 관한 연구," 한국정보과학회 '91 가을 학술발표 논문집, 제 18 권, 제 2 호, 1991. 10.

[10] 정 진욱, 우 치수, "한글 텍스트 압축기법의 구현," 한국정보과학회 논문지, 제 18권, 제 4호, '91. 7.