정보보호개론

-Breaking a Vigenere variant cipher-

2017310100 김수종

1. 개발환경

해당 과제는 인의예지 서버 내에서 linux 환경을 기반으로 C언어로 진행하였다. 라이브러리로 c언어에서 기본적으로 제공하는 <stdio.h>, <stdlib.h>, <string.h>를 사용하였다. 컴파일을 위해서 소스코드와 같은 디렉토리에 ascillFrequency.txt 파일이 존재하여야 한다.

1. 코드 설명

이 소스코드의 전체 구성은 크게 암호의 key length 찾기, key의 각 index 별로 가장 많이 나오는 값 찾기, 암호의 index 별로 값 추측하기 세가지로 이루어져 있다. 우선 hw1\_input.txt 파일을 입력 받아 모두 buffer라는 한 char 배열에 넣어주었다. 입력 값의 범위가 1000~5000 bytes 이므로 한 버퍼에 넣어주었다.

Key length를 결정하는 방식은 다음과 같다.

1. Key length를 1부터 10까지 증가시키며 각 key length 별로 input의 index 0부터 key length 만큼씩 더해가며 각 ascii code의 빈도수를 체크한다.
2. 각 key length에서 아스키 코드의 을 계산하여 poss\_key 배열에 저장해 놓는다.
3. Key length를 10까지 검사 후 가장 큰 poss\_key 값을 찾는다.
4. 가장 큰 poss\_key의 key\_length의 공약수들 중 target key length가 있을 수 있으므로 가장 큰 poss\_key – 0.05 이상의 값을 가진 key\_length 중 가장 작은 값을 final key length로 결정한다.

Key length를 이렇게 결정한 후 이후에 더 빠르게 key 값을 찾기 위해 각 key의 index 별로 가장 많이 나오는 값들을 찾는다. 이를 모두 biggestlist에 넣어주었다.

그 후 각 index 별로 byte값을 추측해야 한다. 이를 위해 기존 ASCII 코드의 평문에서의 빈도수 값을 asciiFrequency.txt 파일로부터 ascillFre 배열로 받아온다. 여기서 추후에 좀 더 빠르게 해당 byte 값을 찾기 위해 가장 빈도수가 높은 값부터 frequency 배열로 받아온다. 이러한 정보들을 가지고 key의 후보지들을 만들고 이를 검사하는 과정은 다음과 같다.

1. 각 index 별로 가장 많이 나오는 값과 asciiFrequency.txt 상에서 가장 많이 나오는 값이 같다고 추측하고 XOR 연산을 통해 key 후보를 만든다.
2. Check 함수를 통해 실제 ascii 코드의 빈도 값 와 이 ciphertext와 key 후보의 XOR 연산을 통해 나온 빈도 값 의 곱들의 합을 계산한다.
3. 만약 이 값이 0.06을 넘는다면 해당 추측이 확실하므로 break를 하고 해당 index의 key 값을 최종 값으로 결정한다. 0.06을 넘지 않는다면 ascillFrequency.txt 상에서 그 다음으로 많이 나오는 값과 같다고 추측하고 다시 I번 부터 진행한다.
4. 만약 모든 256가지 값에서 0.06을 넘는 값이 나오지 않는다면 이 중에서 가장 큰 값을 최종 값으로 결정한다.

이렇게 얻게 된 key 값들로 기존의 ciphertext 값에 XOR 연산을 통해 plaintext를 얻는다.

1. 시간 복잡도 분석

기존에는 이러한 Breaking a Vigenere variant cipher을 진행할 때 Key length가 1과 L 사이 일 때 Key length를 결정하는데 256L 만큼의 시간 복잡도가 발생하고 각 index 별로 key 값을 결정하는데 256^2\*L 만큼의 시간 복잡도가 발생한다고 알려져 있다. 해당 코드에서도 Key length를 결정할 때 256L 만큼의 시간 복잡도가 발생한다. 각 index 별 key 값을 결정할 때는 최악의 경우에는 마찬가지로 256^2\*L 만큼의 시간 복잡도가 발생한다. 하지만 해당 코드에서는 가장 자주 나오는 값을 ASCII 코드의 빈도수가 가장 높은 것부터 대조하며 대부분의 경우에서 256개를 모두 검사하지 않고 제일 빈도수가 높은 ‘ ‘과 ‘e’ 에서 결정된다. 그러므로 평균적인 경우 시간 복잡도는 256\*L에 수렴한다.