PROTOTYPE MODEL LOGIC

UBI Encryption

우리만의 비밀코드: UBI 암호화
C언어 기반 모델 ProtoType
Presentation explaining the UBI Encryption Logic

Ahn Sung Hyun
(shacoding.com)

CONTENTS

01 Structure
- 프로그램 설명
- 프로그램 기능
- 인코딩 및 디코딩

02 Encoding
- 인코딩 함수
- 인코딩 과정

03 Decoding
- 디코딩 함수
- 디코딩 과정

04 Others
- 문자 수 세기
- 사용자 경고문
- 추후 발전방향
- 업데이트 사항

01 | Structure

프로그램 설명

우리만의 비밀코드 : UBI 암호화

우비 암호화는 RSA기반 암호화에 UBI(우비) 컨셉을 결합해서 'UBBUBI' 와 같은 조합의 암호를 만드는 것을 말한다.

암호라는 것이 이미지가 딱딱해서 중요성을 모르는 사람들이 많다고 생각한다.

따라서 우비같이 비를 연상케 하는 친근한 이미지와 편리하게 암호를 만들 수 있는 시스템이 있으면 개선될 것이라고 기대하였다.

또한 기존에 암호문을 만들어 보려 했으나 복잡해서 포기한 사람들한테도 도움이 될 것이라고 생각한다.

Hello, My name is SungHyun.

BUUBUBUBUBBUBUIBBUUUUBUBBBUIBBUUBUUUBBUBBUIBBBUBBBBBUUB
UIBBUBBUBBUUBBUUIBUBUUUUUBBUBBBIBUBBBUBBBBBUBIBUUBUUBUBB
BUUBIBUBUUUBUUBUBIBUBBUBUUBUUUIBUBBUBBBUBUBBBIBBUBUUBUUU
BBBUIBBBUBBUUBUBUUUIBBUUBUBBUBUBUUI

실제 프로그램으로
암호화한 내용입니다!

01 | Structure

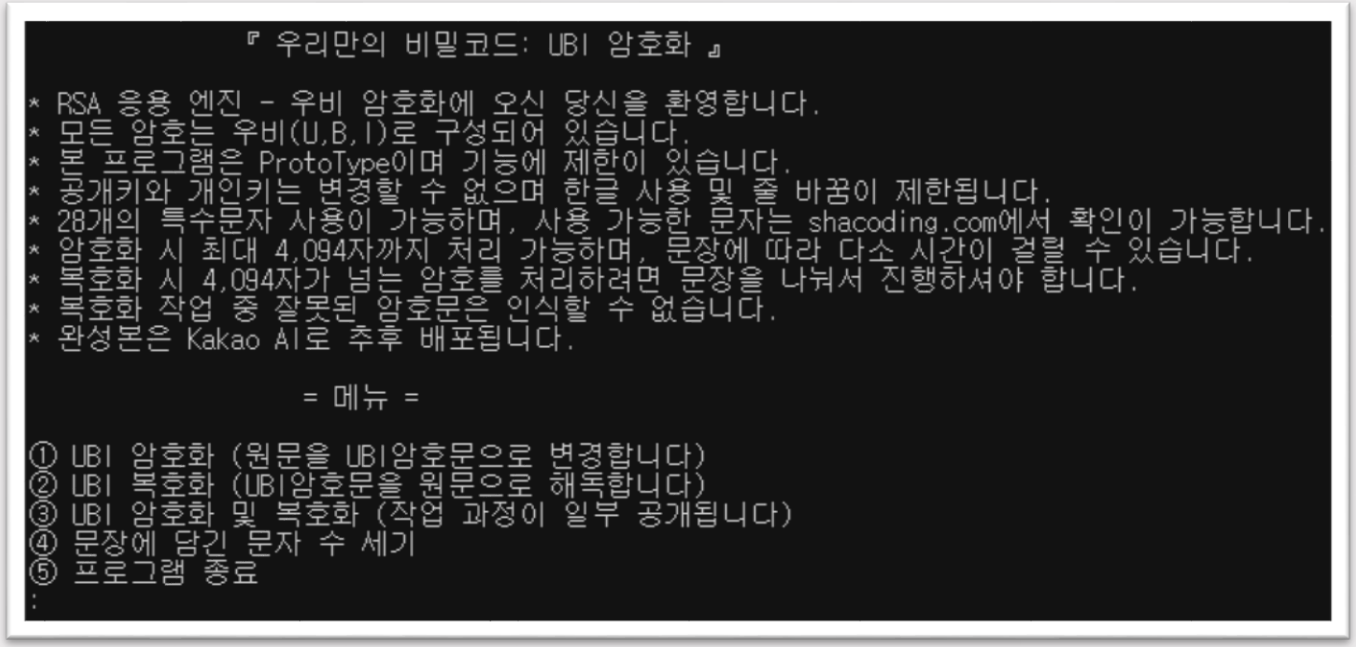
프로그램 기능

C언어를 기반으로 콘솔창에서 동작하는 ProtoType모델이다.

RSA기반으로 공개키와 개인키를 사용하며 현재, 프로그램 내에서 변경은 제한돼있다.

암호화, 복호화, 암호화 및 복호화(연속), 문자 수 세기 기능을 제공한다.

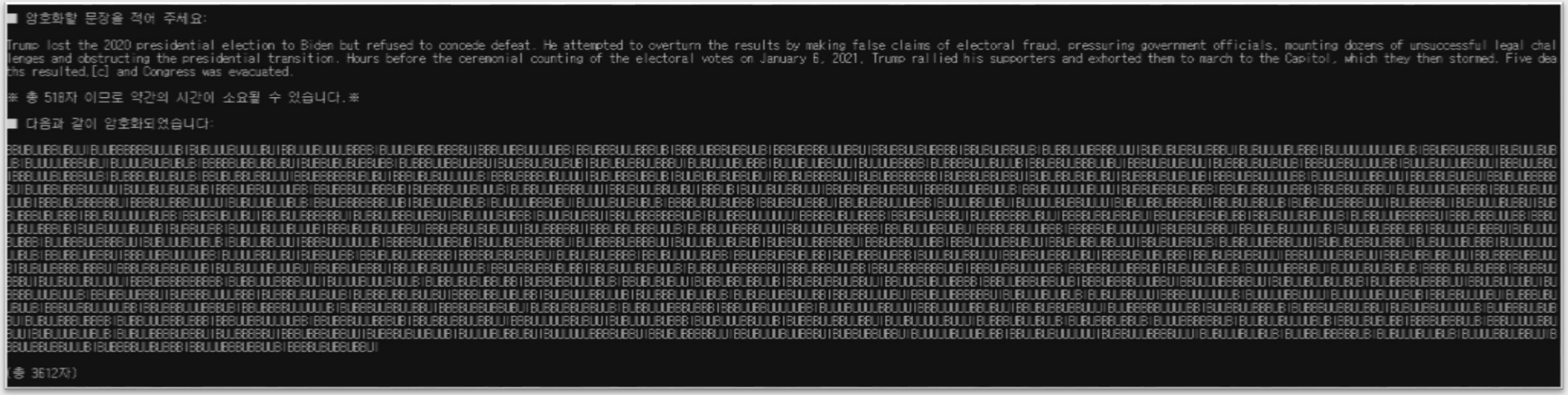
영어 소문자 및 대문자, 숫자를 포함해서 총 90개의 문자를 처리할 수 있다.



(**구현된 특수문자(28개)**: +,-,*,/,%,=,|,&,<,>,!,?,@,#,$,~,(,),[,],^,콜론, 세미콜론, 작은 따옴표, 큰 따옴표, 마침표, 콤마, 공백)

01 | Structure

프로그램 주요 기능



메뉴(1) – 암호화

< 콘솔창 입력 최대 범위: 4,094자 >

01 | Structure

프로그램 주요 기능

메뉴(2) – 복호화

< 콘솔창 입력 최대 범위: 4,094자 >
- 암호문이 원문에 비해 매우 기니 나눠서 진행 권장 -

01 | Structure

프로그램 주요 기능

```
■ 암호화할 문장을 적어 주세요:

Trump lost the 2020 presidential election to Biden but refused to concede defeat. He attempted to overturn the results b
y making false claims of electoral fraud, pressuring government officials, mounting dozens of unsuccessful legal challen
ges and obstructing the presidential transition.
```

```
■ 다음과 같이 암호화되었습니다:

BBUBUUBBUBUUIBUUBBBBBBUUUUBIBUBUUUBUUUUBUIBBUUUBUUUUBBBBIBUUUBUBBUBBBBUIBBBUUBBUUUUUBBIBBUBBBUUUBBBUBIBBBUUBBBUBBUUBIBB
UBBBBUUUBBUIBBUBBUUBUBBBBIBBUBUUBBUBIBUBBUUUBBBBUUUIBUBUBUBBUUBBUIBUBUUUUBUBBBIBUUUUUUUUUUBUBIBBUBBUUBBBUIBUBUUUBUBUB
BUUUUUBBBUBUIBUUUBUUBUBUBIBBBBBUBBUBBUBUIBUBBUBUBUBBUBBIBUBBBUUBBUUBBUIBUUBBUUBUUBUBIBUBUBUBBUUBBBUIBUBUUUBUBBBIBUUUB
UBBUUUIBUUUUBBBBBIBUBBBBUUUBUUUBIBBUBBUUBBBUUBUIBBBUUBUBUUUIBUBBBUBUUBUUBIBBBUUBBUUUUUBBIBUUUBUUUUBBUUUIBBBUUUBBUIBBBU
UBUBBBUUBIBUBBBUBUUBUUBIBBUBUBBUBBUUUIBBUBBBBBBUBUBUIBBBUBUBUUUUUUBIBBBUBBBUBUUUUIBUBUBBBUBBUBIBUUBUBUBUBBUBUIBBUBUBBB
```

```
■ 다음과 같이 복호화되었습니다:

Trump lost the 2020 presidential election to Biden but refused to concede defeat. He attempted to overturn the results b
y making false claims of electoral fraud, pressuring government officials, mounting dozens of unsuccessful legal challen
ges and obstructing the presidential transition.

(총 289자)
```

메뉴(3) – 암호화 및 복호화

< 암호화 -> 복호화 연속 진행 > - 진행 과정 일부 공개 (RSA 해독 과정) -

```
e:14453, n:16637
원원원원원원원원원원 숫자 4618 => 암호 숫자 3380
원원원원원원원원원원 숫자 2113 => 암호 숫자 10209
원원원원원원원원원원 숫자 1663 => 암호 숫자 5186
원원원원원원원원원원 숫자 1215 => 암호 숫자 12559
원원원원원원원원원원 숫자 1920 => 암호 숫자 8926
원원원원원원원원원원 숫자 6320 => 암호 숫자 14723
원원원원원원원원원원 숫자 805 => 암호 숫자 14109
원원원원원원원원원원 숫자 6355 => 암호 숫자 14809
원원원원원원원원원원 숫자 5355 => 암호 숫자 15302
원원원원원원원원원원 숫자 5363 => 암호 숫자 6959
원원원원원원원원원원 숫자 1618 => 암호 숫자 1689
원원 숫자 519 => 암호 숫자 11384
```

```
d:17, n:16637
원원원원환 숫자 3380 => 복호 숫자 4618
원원원원환 숫자 10209 => 복호 숫자 2113
원원원원환 숫자 5186 => 복호 숫자 1663
원원원원환 숫자 12559 => 복호 숫자 1215
원원원원환 숫자 8926 => 복호 숫자 1920
원원원원환 숫자 14723 => 복호 숫자 6320
원원원원환 숫자 14109 => 복호 숫자 805
원원원원환 숫자 14809 => 복호 숫자 6355
원원원원환 숫자 15302 => 복호 숫자 5355
원원원원환 숫자 6959 => 복호 숫자 5363
원원원원환 숫자 1689 => 복호 숫자 1618
원원환 숫자 11384 => 복호 숫자 519
```

01 | Structure

인코딩 구조 (hi->BUBUUBBBUBBBBI)

FIRST STEP	SECOND STEP	THIRD STEP	FOURTH STEP	FIFTH STEP
문자열 -> 문자 조각	문자 조각 -> 전처리 코드	전처리 코드 -> 원 숫자	원 숫자 -> 암호 숫자	암호 숫자 -> 암호 문자열
hi -> h와 i로 분해!	h, i -> 8과 9로 변경 a->1, b->2, ... -> z->26 A->27, B->28, ...	8, 9-> 809로 결합 1, 2 -> 102로 결합 10, 20 -> 1020으로 결합	809-> 5359로 암호화 C(암호 숫자)= M(원 숫자)^e mod n <RSA 암호화 공식>	5359-> 10100111101111 변환 (10진수->2진수) 10100111101111 -> BUBUUBBBUBBBBI 변경 (1->B, 0->U, 구분점->I)

분해 → 변경 → 결합 → 암호화 → 변환

전처리 코드 : RSA 암호화 전처리 코드

e:14453, n:16637

01 | Structure

디코딩 구조 (BUBUUBBBUBBBBI ->hi)

FIRST STEP
암호 문자열 -> 변환 숫자

BUBUUBBBUBBBBI
-> 10100111011111 **변경**
(B->1, U->0, I->**구분점으로 이용**)

10100111011111 -> 5359 **변환**
(**2진수->10진수**)

SECOND STEP
변환 숫자 -> 복호 숫자

5359-> 809**로 복호화**

M(복호 숫자)=
C(변환 숫자)^d mod n
<RSA **복호화 공식**>

THIRD STEP
복호 숫자 -> 전처리 코드

809-> 8, 9**로 분해**

102 -> 1, 2**로 분해**
1020 -> 10, 20**으로 분해**

FOURTH STEP
전처리 코드 -> 문자 조각

8, 9 -> h**와** i**로 변경**

1->a, 2->b, ... -> 26->z
27->A, 28->B, ...

FIFTH STEP
문자 조각 -> 문자열

h**와** i -> hi**로 결합!**

변환 → **복호화** → **분해** → **변경** → **결합**

변환 숫자 = 암호 숫자

복호 숫자 = 원 숫자
n:16637, d:17

전처리 코드 : RSA
암호화 전처리 코드

02 | Encoding

인코딩 함수

* **code =**「 **RSA** 암호화 전처리 코드 」집합 저장 **(ex) a**는 **1**로, **b**는 **2**로)
* **bundle =** 원 숫자 집합 저장 **(ex) 102,203,1020,...)**
* **encode =** 암호 숫자 집합 저장 **(ex) 2120,3366, ...)**

문자열 -> 문자 조각

```
get_symbol(char* tmp)
```

문자열(string)을 조각으로 나눠서 리턴 (ex) ab-> 1회 시행:'a', 2회 시행:'b')

문자 조각 -> 전처리 코드

```
make_Code(char* string, list* code)
```

문자 조각으로 「 RSA 암호화 전처리 코드 집합(code) 」 생성 (ex) a->1, b->2)

전처리 코드 -> 원 숫자

```
get_integer(list* code)
```

전처리 코드 2개로 원 숫자 만들기 (ex) 1,2 -> 102)

```
make_Bundle(list* code, list* bundle)
```

원 숫자 집합 생성 <bundle> (ex) 102,1021,1200,..)

원 숫자 -> 암호 숫자

```
make_C(int M)
```

RSA 암호화 <원 숫자 -> 암호 숫자로 변경> (ex) 102 -> 3366)

```
make_Encode(list* bundle, list* encode, int num)
```

암호 숫자 집합 생성 <encode> (ex) 3366,3450,9800,..)

암호 숫자 -> 암호 문자열

```
make_Estring(list* encode)
```

암호 숫자로 암호 문자열(Estring) 생성 (ex) 3366 3450 -> 110100100110 110101111010 -> BBUBUUBUUBBUIBBUBUBBBBUBUI)

02 | Encoding

인코딩 과정 (hello-> BBUBBBUUUBBBUBIBBUUUUBUBBBUIBBUBUBBUBBBUUBI)

문자열 -> 문자 조각

```
문자 조각:  h(104) | e(101) | l(108) | l(108) | o(111) |
```

문자 조각 -> 전처리 코드

```
Code(전처리 코드): 8 | 5 | 12 | 12 | 15 |
```

전처리 코드 -> 원 숫자

```
Bundle(원 숫자 집합):  805 | 1212 | 1500 |
```

원 숫자 -> 암호 숫자

```
e:14453, n:16637
원 숫자 805 => 암호 숫자 14109
원 숫자 1212 => 암호 숫자 3118
원 숫자 1500 => 암호 숫자 13753
```

암호 숫자 -> 암호 문자열

```
2진수 변환 배열(1): 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
2진수 변환 배열(2): 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
2진수 변환 배열(3): 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
```

'I' 가 구분점

```
BBUBBBUUUBBBUBIBBUUUUBUBBBUIBBUBUBBUBBBUUBI

(총 43자)
```

03 | Decoding

디코딩 함수

* code2 = 「 RSA 복호화 전처리 코드 」 집합 저장
  (ex) B는 1로, U는 0으로)
* c_code = 변환 숫자(암호 숫자) 집합 저장 (ex) 2120,3366, ...)
* decode = 복호 숫자(원 숫자) 집합 저장 (ex) 102,203,1020,...)

암호 문자열 -> 변환 숫자

RSA 복호화 전처리 코드 ≠
전처리 코드 =
RSA 암호화 전처리 코드

```
get_symbol2(char* tmp)
```
Estring의 문자열을 조각으로 나눠서 리턴 (ex) BU-> 1회 시행:'B',2회 시행:'U')

```
make_code2(char* Estring, list* code2)
```
암호 문자 조각으로 「 RSA 복호화 전처리 코드 집합(code2) 」 생성 (ex) B->1, U->0)

```
make_c_code(list* code2, list* c_code)
```
code2 이용해서 변환 숫자(암호 숫자) 집합 생성 (ex) 11000011100 -> 1564)

변환 숫자 -> 복호 숫자

```
make_M(int C)
```
RSA 복호화 <변환 숫자(암호 숫자) -> 복호 숫자(원 숫자)로 변경> (ex) 1564 -> 102)

```
make_Decode(list* c_code, list* decode, int num)
```
복호 숫자(원 숫자) 집합 생성 <decode> (ex) 102,1021,1200,..)

복호 숫자 -> 전처리 코드

```
get_Code(list* decode)
```
복호 숫자(원 숫자) 분할해서 「 RSA 암호화 전처리 코드 」 반환 (ex) 102 -> 1과 2 반환)

전처리 코드 -> 문자 조각

문자 조각 -> 문자열

```
decode_Effect(int* decode)
```
「 RSA 암호화 전처리 코드 」 를 아스키코드로 변환 후 문자 출력
(전처리 코드를 다 쓸 때까지 반복) (ex) 1->97->'a'출력, 2->98->'b'출력, ...)

03 | Decoding

디코딩 과정 (BBUBBBUUUBBBUBIBBUUUUBUBBBUIBBUBUBBUBBBUUBI -> hello)

암호 문자열 -> 변환 숫자

```
문자 조각: B | B | U | B | B | B | U | U | U | B | B | B | U | B | I | B | B | U | U | U | U | B | U | B | B | B | U | I | B | B | U | B | U | B | B | U | B | B | B | U | U | B | I |
Code2(복호화 전처리 코드): 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | -1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | -1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | -1 |
C_code(변환 숫자 집합): 14109 | 3118 | 13753 |
```

'I' -> '-1'

'-1' 기준으로 나눠서 변환

변환 숫자 -> 복호 숫자

```
d:17, n:16637
변환 숫자 14109 => 복호 숫자 805
변환 숫자 3118 => 복호 숫자 1212
변환 숫자 13753 => 복호 숫자 1500
```

복호 숫자 -> 전처리 코드

```
8 | 5 | 12 | 12 | 15 |
```

전처리 코드 -> 문자 조각

```
문자 조각: h(104) | e(101) | l(108) | l(108) | o(111) |
```

문자 조각 -> 문자열

```
hello
(총 5자)
```

04 | Others

문자 수 세기

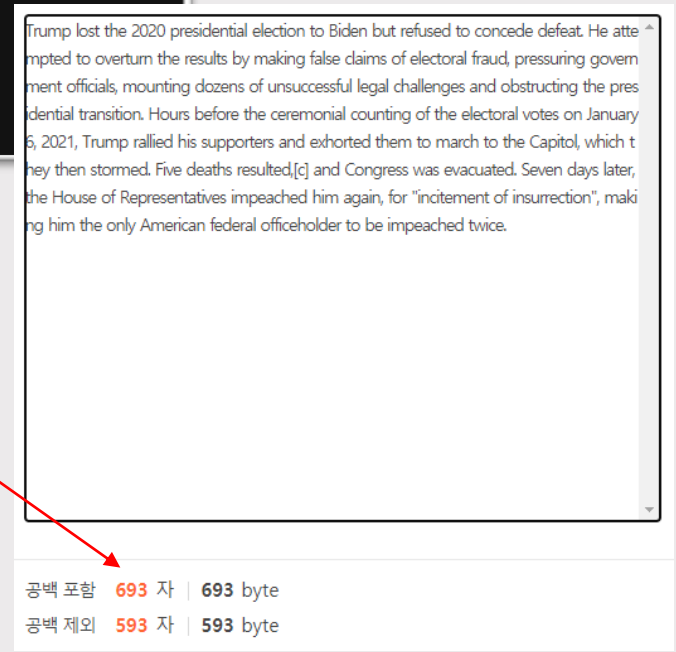
```
■ 문자 개수를 셀 문장을 적어 주세요 (최대 4,094자):

Trump lost the 2020 presidential election to Biden but refused to concede defeat. He attempted to overturn the results b
y making false claims of electoral fraud, pressuring government officials, mounting dozens of unsuccessful legal challen
ges and obstructing the presidential transition. Hours before the ceremonial counting of the electoral votes on January
6, 2021, Trump rallied his supporters and exhorted them to march to the Capitol, which they then stormed. Five deaths re
sulted,[c] and Congress was evacuated. Seven days later, the House of Representatives impeached him again, for "inciteme
nt of insurrection", making him the only American federal officeholder to be impeached twice.


( 문자의 개수는 693개 입니다! [공백 포함] )
( 문자의 개수는 593개 입니다! [공백 제외] )
```

(**메뉴**.4를 통해 수 세기)

(**전문 프로그램이랑 수치 일치**)




```
hello
(총 5자)
```

(**암호화 하기 전, 복호화한 후**)

```
BBUBBBUUUBBBUB|BBUUUUBUBBBU|BBUBUBBUBBBUUB|
(총 43자)
```

(**암호화한 후**)

04 | Others

사용자 경고문

```
■ 암호화할 문장을 적어 주세요:

Trump lost the 2020 presidential election to Biden but refused to concede defeat. He attempted to overturn the results b
y making false claims of electoral fraud, pressuring government officials, mounting dozens of unsuccessful legal challen
ges and obstructing the presidential transition. Hours before the ceremonial counting of the electoral votes on January
6, 2021, Trump rallied his supporters and exhorted them to march to the Capitol, which they then stormed. Five deaths re
sulted,[c] and Congress was evacuated. Seven days later, the House of Representatives impeached him again, for "inciteme
nt of insurrection", making him the only American federal officeholder to be impeached twice.

※ 총 693자 이므로 약간의 시간이 소요될 수 있습니다.※
```

200자 이상의 문자를 암호화할 시 딜레이가 발생할 수 있으므로 <경고문> 을 출력!

```
ve deaths resulted,[c] and Congress was evacuated. Seven days later, the House of Representatives impeached him again, f
or "incitement of insurrection", making him the only American federal officeholder to be impeached twice.Trump lost the
2020 presidential election to Biden but refused to concede defeat. He attempted to overturn the results by making false
claims of electoral fraud, pressuring government officials, mounting dozens of unsuccessful legal challenges and obstruc
ting the presidential transition. Hours before the ceremonial counting of the electoral votes on January 6, 2021, Trump
rallied his supporters and exhorted them to march to the Capitol, which they then stormed. Five deaths resulted,[c] and
Congress was evacuated. Seven days later, the House of Representatives impeached him again, for "incitement of insurrect
ion", making h

※ 해당 문장은 4,094자를 초과해서 콘솔창에서 일부분이 삭제됐을 가능성이 있습니다!! ※
※ 총 4094자 이므로 약간의 시간이 소요될 수 있습니다.※
```

콘솔창에 4,094자를 초과해서 입력하면 내용이 삭제되므로, 입력한 문자가 4,094자 이상일 때 <경고문> 을 출력!

04 | Others

추후 발전방향

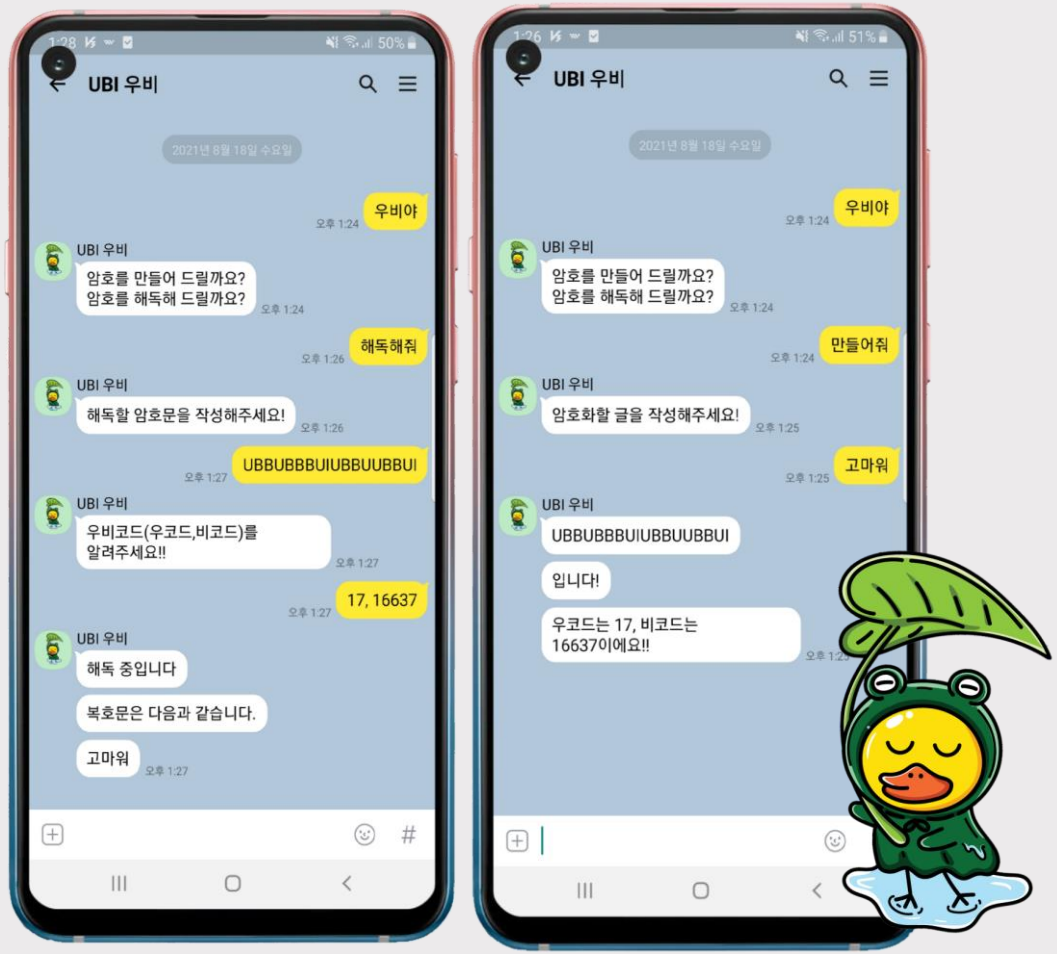
'친근하고 편리한 암호화'를 목표로 하므로
대중화된 '카카오톡'을 이용해서 제작!

암호화할 때 필요한 공개키(e,n)는 랜덤 생성돼서
사용자가 입력하는 부담을 줄이도록 구상함.
또한 (e,n)을 (우코드,비코드)로 이용해서
수학의 딱딱한 이미지를 탈피하고자 함.

복호화(해석)를 할 때는 사용자가 우비코드를 입력 시
검증 후 서버에 저장된 개인키로 진행함.

<e,d,n 상수: 랜덤 생성/ Bot이 암호화한 후 e,n 제공>

(카카오 오픈빌더 API이용 & Node.js로 구현 계획)



04 | Others

업데이트 사항 (2/8)

1. 사용자의 우비코드 사용!

-> RSA의 e값은 우코드, n값은 비코드로 지정함.
UBI 암호화 시 사용자에게 우비코드를 제공함.
UBI 복호화 시 우비코드를 입력해야 실행됨.

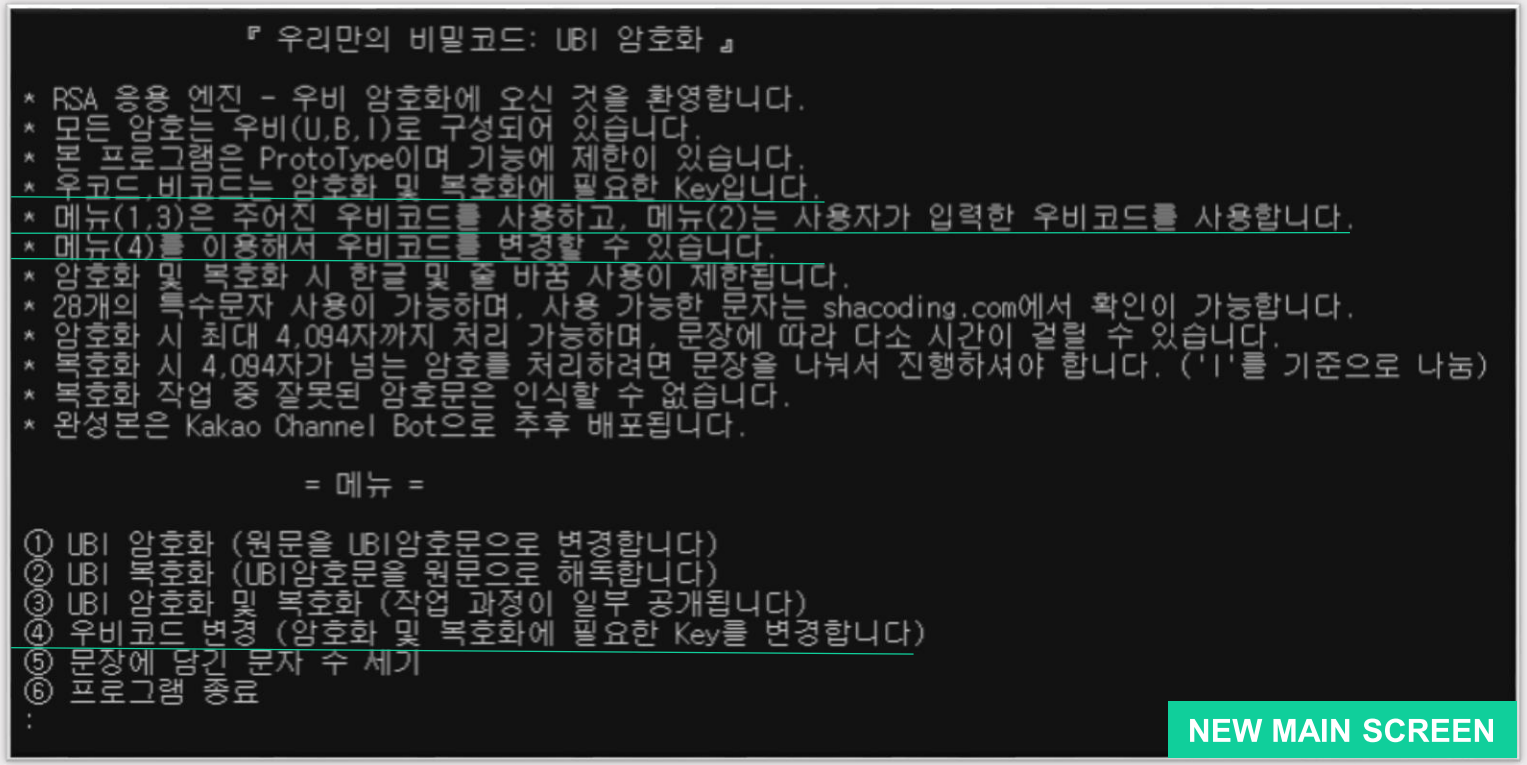
2. RSA 랜덤 키 생성으로 인한 보안강화!

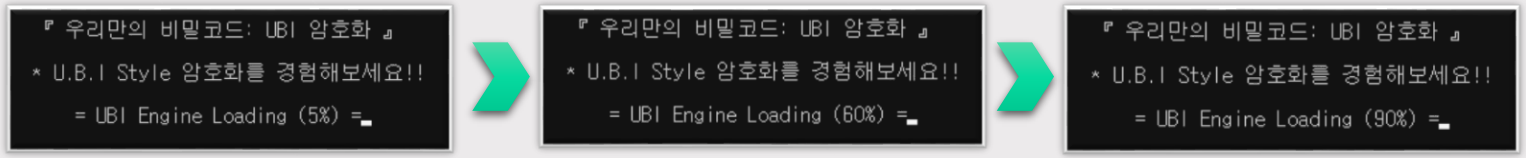
-> 시작 시, RSA의 개인키와 공개키를 랜덤으로 생성함.
역추적 시 170,500가지의 경우의 수를 따져야 함.
프로그램이 새로 실행될 때마다 키 값은 변경됨.

3. 프로그램에서 Key 값 변경 가능!

-> 프로그램 실행 후 우비코드(e,n) 및 d값 변경 가능.
메뉴(4)를 이용해서 순식간에 값이 변경됨.

(RSA 키 생성 중 로딩 화면 생성)





04 | Others

업데이트 확인(I)

(암호화 하고 우비코드를 알림)

```
■ 암호화할 문장을 적어 주세요:
abcdefg
(총 7자)
■ 다음과 같이 암호화되었습니다:
BUUUUUUUBUBBBUB|BBUBUUBBUUBUBUU|BUBUBUBBBBUUBBU|BBUBUUUUUUUBBBU|
(총 64자)
[ 우코드(e), 비코드(n) ] = [ 781, 36811 ]
```

(복호화를 위해 우비코드를 작성)

```
■ 우코드(e)를 입력해주세요:781
■ 비코드(n)를 입력해주세요:36811
■ 복호화할 문장을 적어 주세요:
BUUUUUUUBUBBBUB|BBUBUUBBUUBUBUU|BUBUBUBBBBUUBBU|BBUBUUUUUUUBBBU|
(총 64자)
■ 다음과 같이 복호화되었습니다:
abcdefg
(총 7자)
```

사용자의 우비코드 사용!

-> RSA의 e값은 우코드, n값은 비코드로 지정함.
UBI 암호화 시 사용자에게 우비코드를 제공함.
UBI 복호화 시 우비코드를 입력해야 실행됨.

04 | Others

업데이트 확인(Ⅲ)

프로그램에서 Key 값 변경 가능!

-> 프로그램 실행 후 우비코드(e,n) 및 d값 변경 가능.
메뉴(4)를 이용해서 순식간에 값이 변경됨.

```
* 우코드(e)는 777, 비코드(n)는 83371로 변경되었습니다!

[ Random Key -> Public(n,e):83371,777, Private(d):71817 ]
[ Random Key Test -> [Original] 10000, [Encoded] 4348, [Decoded] 10000 ]
```

(메뉴(4)를 입력 시 다른 Key로 즉시 변경됨!)

// 프로그램 시작 시 **약 2500개의 키가 생성됨**
// 2500개 중 하나의 키로 우비코드를 재지정!
// 중복 키가 나오지 않게 설정함

(우비코드가 변경됐으니 당연히 암호문도 바뀜)

```
■ 암호화할 문장을 적어 주세요:

abcdefg

(총 7자)

■ 다음과 같이 암호화되었습니다:

BBBBBBUUBBBUUUUB|BUBBUBBUBBUBBUUU|BUBBBUUBUBUBUBUU|BUUBUBUUUBBBBBUBU|

(총 69자)

[ 우코드(e), 비코드(n) ] = [ 777, 83371 ]
```

U B I E n c r y p t i o n P r o t o t y p e

THANK YOU FOR WATCHING