MysteryTwister C3

THE CRYPTO CHALLENGE CONTEST

Fastest in the West

Author: Matthias Minihold, ECRYPT-NET

February 2018

Introduction (1/3)

The transportation company FTW, Fastest Trains in the West, also sells tickets on the Internet. After scrolling through endless tailored ads, eventually stumbling upon the credit card payment option,

1. your surname,
2. a journey confirmation number, and
3. a slightly obfuscated credit card number

are emailed to you, along with all kinds of additional information. As FTW does not accept your particular credit card, in the end, a friend, kindly re-typing all 'required' fields in the form, buys the train ticket for you. Anyway – done!

Introduction (2/3)

On the day you leave, your attention is drawn to the fact that, despite full payment, it's necessary to turn a "journey *reservation* confirmation" into an actual acceptable ticket. To do so you have to re-input all the data at one of the machines at the train station. The queue in front of the machine is suspiciously short – as it sadly is out of order. Collecting a coupon, waiting for your time-slot at the counter, you realize you do not physically have the credit card – even worse, the full credit card number is partially unknown to you. As your paranoid friend is impossible to reach, and would be reluctant to share the credit card details over the phone anyway, you nevertheless decide to claim the ticket at the counter.

Introduction (3/3)

There, it is necessary to "prove" you are the rightful owner of the ticket by providing the surname, a journey confirmation number (both sent via email), and the full credit card number you actually don't have.
You remember a recent forum entry providing details on the company's use of deprecated cryptography: It seems as if they use the subset-sum problem to protect the credit card numbers.
Subset-sum problems are often sub-problems of modern cryptographic schemes, however stand-alone usage is very unusual.

Forum excerpt about FTW's encryption details

```
n = 80
N = [92118164865905878700455 2, 11845070455954523230034 98,
11316528264440999722858 7, 35528881685407115857532 5, 26527867818322757048187 4,
10373390696965678004605 3, 36789847914243044344006 4, 20106684512570097689394 0,
55510331038633869529914 3, 97285422301532317534832 1, 84438844758592566642448 6,
98858961634645186511121 5, 25947229035918392948147 0, 12306283211253899164679 3,
23472931483639701717928 4, 59885239107604242222621 1, 11157675584539390777278 31,
95643630004931996306386 2, 58394357846204828032640 6, 75371389731423199078594 2,
99726989200436797567622 2, 80946079411568520574881 5, 11514260649989665006321 19,
91780250689134568006960 3, 45626255807069723562793 4, 99416949823436900023061 3,
66559001851832264361162 1, 10008137033300775310133 08, 10822216147950544684738 09,
86407569949115404262234 0, 67571414286080886852415 4, 17428820549683551498360 9,
11551731830480769569045 03, 31373248446199131849426 2, 85379184125058771092336 7,
85325669700572585931221 7, 53445262758719358969211 5, 66527096104873027061960 1,
84716444970913487860059 9, 83332655730310696690267 4, 84922429311875509023761 3,
48776273397174277736215 0, 20500130511829850019482 9, 40972599629008216735276 0,
11589868312410298226240 55, 11072700498451717423357 9, 38672715894073406007944 6,
19919523882762125146012 , 49229611307950513682409 , 99462980345887581187342 1,
22913629103050712075364 3, 73494139818101997124577 5, 78517042137487894699048 ,
11764891182641983459443 12, 11159048791090580509768 21, 22240055451910635946370 2,
18068011146658400529648 5, 27893592415817944695712 6, 49959363946498299206463 1,
11633840974588584510178 01, 10987463663580627052456 20, 17620452062048272124787 4,
18870177537260853475102 , 85273173141457155192018 4, 48751786051303149771164 0,
58019836469627414797824 1, 11608130538984432767302 14, 10483978888106534631907 33,
83053221606236884423656 7, 35627506618510875385863 9, 61713555757844557749 66,
10121623597008120572416 93, 14353464769170037441786 4, 46670667011446336937319 8,
10527806046410498768647 88, 55574285742333806035198 7, 60918123961316274921402 0,
44174593519017094247718 0, 88880763402725435457187 3, 66372415321314499219729 0]
```

Challenge (1/2)

Having just your laptop at your disposal, you give it a try and ... find out the full credit card number within a few minutes – just in time when it is your turn at the ticket counter.
Without further identity check, you "prove" the rightful order by providing all the data, and eventually get a ticket!
As you enter the high-security high-speed train, you wonder why companies often use deprecated cryptography to secure sensitive information sent over the Internet.

Challenge (2/2)

Suppose that the confirmation number $c = 65761213830692761304809 17$ is an encryption of the 16-digit credit card number from which you know the first two and the last two digits: "22**********65".

For $n = 80$ find the set I with $|I| = \frac{n}{10} = 8$ such that $c = \sum_{i \in I} N_i$ to solve the instance!

The solution consists of the complete credit card number. Please enter the solution without any spaces or additional characters.

Hints

1. If your approach follows the 'Meet in the Middle' paradigm, then combining $I_1 \subseteq \{1, \ldots, \frac{n}{2}\}$ with $I_2 \subseteq \{\frac{n}{2}+1, \ldots, n\}$ such that $|I_1| = |I_2| = \frac{n}{20}$ gives the set $I = I_1 \cap I_2$.
2. A split-set, e.g. $I_1 \subseteq \{1, \ldots, \frac{n}{2}\}$ with $|I_1| = \frac{n}{20} = 4$, can be enumerated with $\texttt{Subsets}(\texttt{range}(0, n/2), n/20)$, in case you implement your solution in SageMath.
3. The use of try-except mechanisms can be helpful, too:

Listing 1: Demo of try-except mechanism

```
N = 6
for i in range(N):
try:
inv = power_mod(i,-1,N)
print "    \%d^(-1) mod \%d = \%d"\%( i, N, inv )
except:
print "x = \%d is not invertible"\%i
```

Informal definition of the subset-sum problem

The subset-sum problem (also knapsack problem) is as follows:

You may already have had a knapsack problem when travelling:

- Your luggage may weight at most S [kg] at check-in.
- Not squandering, the bag WILL weigh exactly S [kg].
- You own n different, equally beautiful items of weight: N_1, N_2, ... , N_n.

Task: Pack a knapsack, too small to contain all your items, with a suitable subset, fulfilling a precise weight constraint.