Total recall: optimizing recovery process for Windows PINs based on dates

1 Total recall: optimizing recovery process for Windows PINs

1.1 Abstract

PIN stands for **P**ersonal **I**dentification **N**umber. But it is not just a number. It was first introduced in Windows 8 and has become a general sign-in method in Windows 10. PIN has a lot of advantages compared to password authentication. Just read the MS article to get more detailed information on that.

1.2 Recovering Windows PINs

1.2.1 Windows PIN security

For those who are too lazy to refer to the article: one of the main advantages of the PIN code is that it is much much harder to crack. Approximately 100 000 times harder compared to a NTLM password hash. Thus regular methods for recovering Windows passwords are not suitable for PIN decryption. In this article we will show how one can decrease the time required to recover some PINs using the new version of Reset Windows Password.

1.2.2 Recovering PINs based on dates

We noticed that a lot of users create their Windows PIN codes using a date of birth. Their own, relatives, pets, does not matter. Usually such PIN codes consist of either 6 or 8 numbers. For example, 12061999, 05112018, etc. However even such simple but forgotten PINs are very difficult to guess assuming that you can get 200-250 guesses per second max on an average CPU.

So decrypting an 8-digit PIN takes 10^8/200 = 500000 seconds or more than 5 days! Let's reduce this time to some reasonable value, say up to 5 hours. We will need a mask attack for that.

1.2.3 Optimizing the recovery process

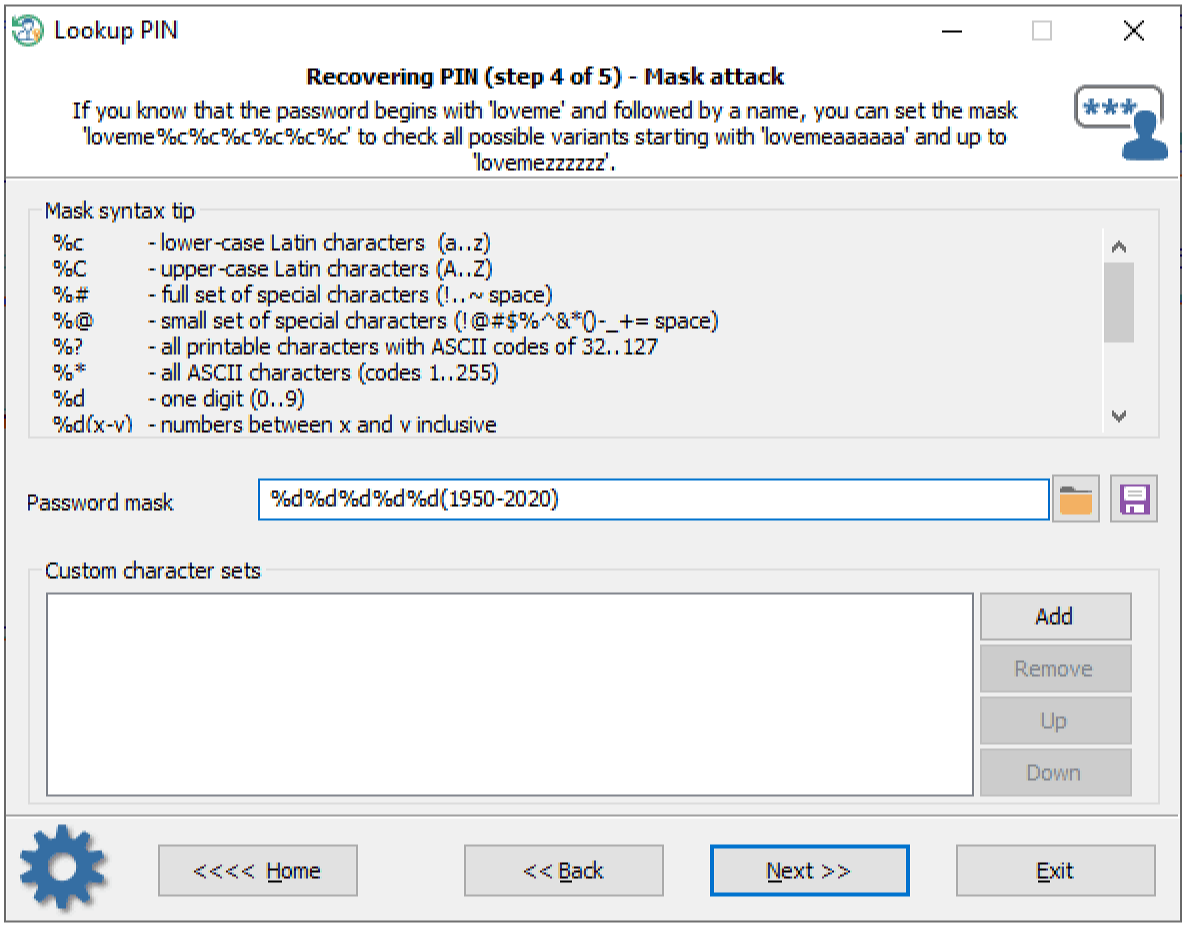
The first thing that comes to mind is setting a digital mask i.e. first character is a digit, second char is a digit too, etc. That is

%d%d%d%d%d%d%d%d

Note that **%d** means one digit in a PIN. 8 digits total.
This mask gives us 100 000 000 combinations. It will take 5 days to check all combinations and that's too much.

Assumption 1: last four digits should be years
What we need: limit the last four digits to some year range
Implementation 1: **%d%d%d%d%d(1800-2030)**
%d(1800-2030) means that we will check numbers from **1800** and up to **2030**

It's hard to imagine that someone could have been born in the 19th century, so what about a stricter restriction?
Implementation 2: **%d%d%d%d%d(1950-2020)**
Just imagine, we have reduced the search range from 100 000 000 to 710 000. Now we can fit into less than a day to go through all PIN combinations.



Recovering PIN (step 4 of 5) - Mask attack

If you know that the password begins with 'loveme' and followed by a name, you can set the mask 'loveme%c%c%c%c%c%c' to check all possible variants starting with 'lovemeaaaaaa' and up to 'lovemezzzzzz'.

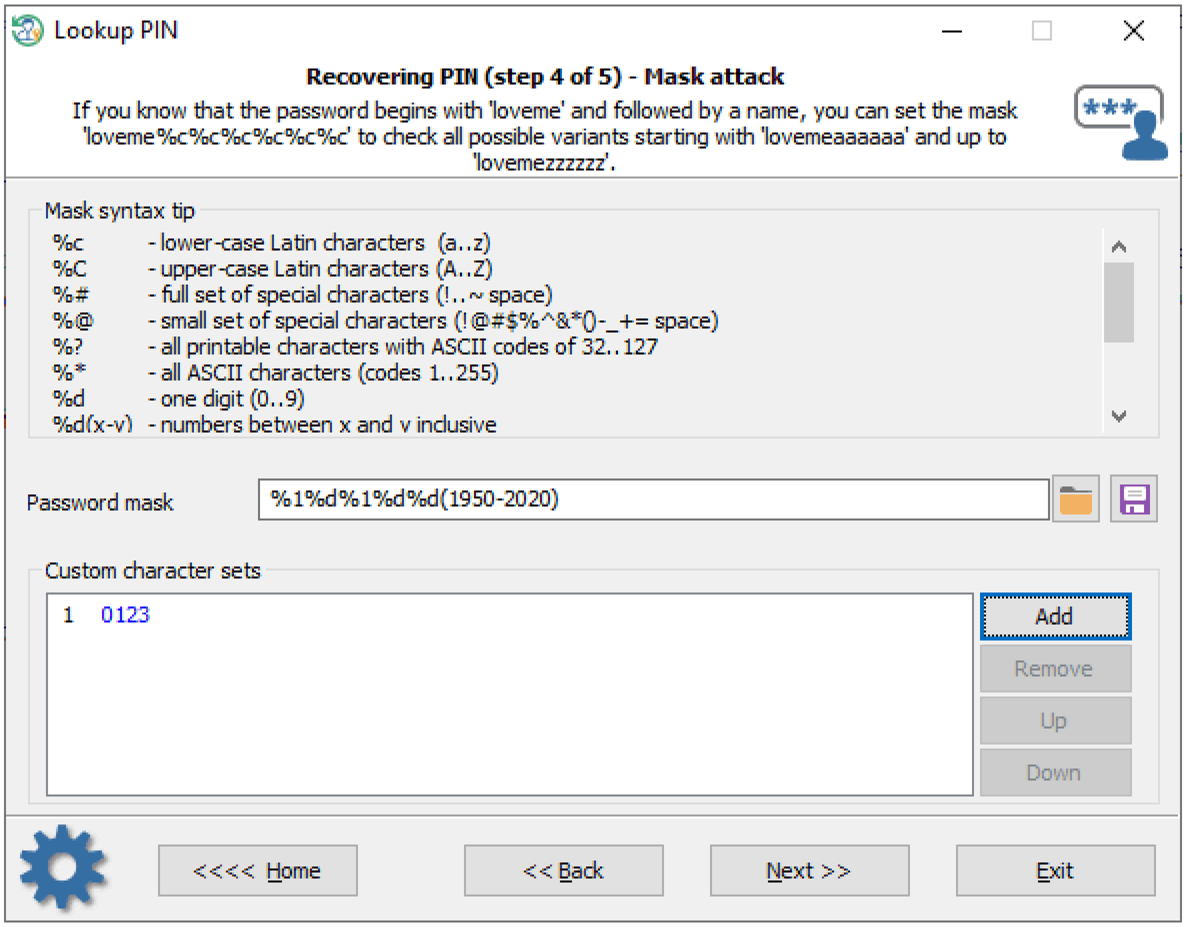
Mask syntax tip

```
%c        - lower-case Latin characters  (a..z)
%C        - upper-case Latin characters (A..Z)
%#        - full set of special characters (!..~ space)
%@        - small set of special characters (!@#$%^&*()-_+= space)
%?        - all printable characters with ASCII codes of 32..127
%*        - all ASCII characters (codes 1..255)
%d        - one digit (0..9)
%d(x-y)   - numbers between x and y inclusive
```

Password mask: %d%d%d%d%d(1950-2020)

Custom character sets

Ok, now something a little bit more difficult.
Assumption 2: first two digits and second two digits should be a day and a month. Or vice versa. In some countries the mmddyyyyy format is in use, in another countries a reversed one i.e. ddmmyyyy. Anyway, both first and third numbers can be only 0, 1, 2 or 3.
Implementation 1: **%1%d%1%d%d(1950-2020)**
where **%1** is the following custom charset: 0123



Recovering PIN (step 4 of 5) - Mask attack

If you know that the password begins with 'loveme' and followed by a name, you can set the mask 'loveme%c%c%c%c%c%c' to check all possible variants starting with 'lovemeaaaaaa' and up to 'lovemezzzzzz'.

Mask syntax tip

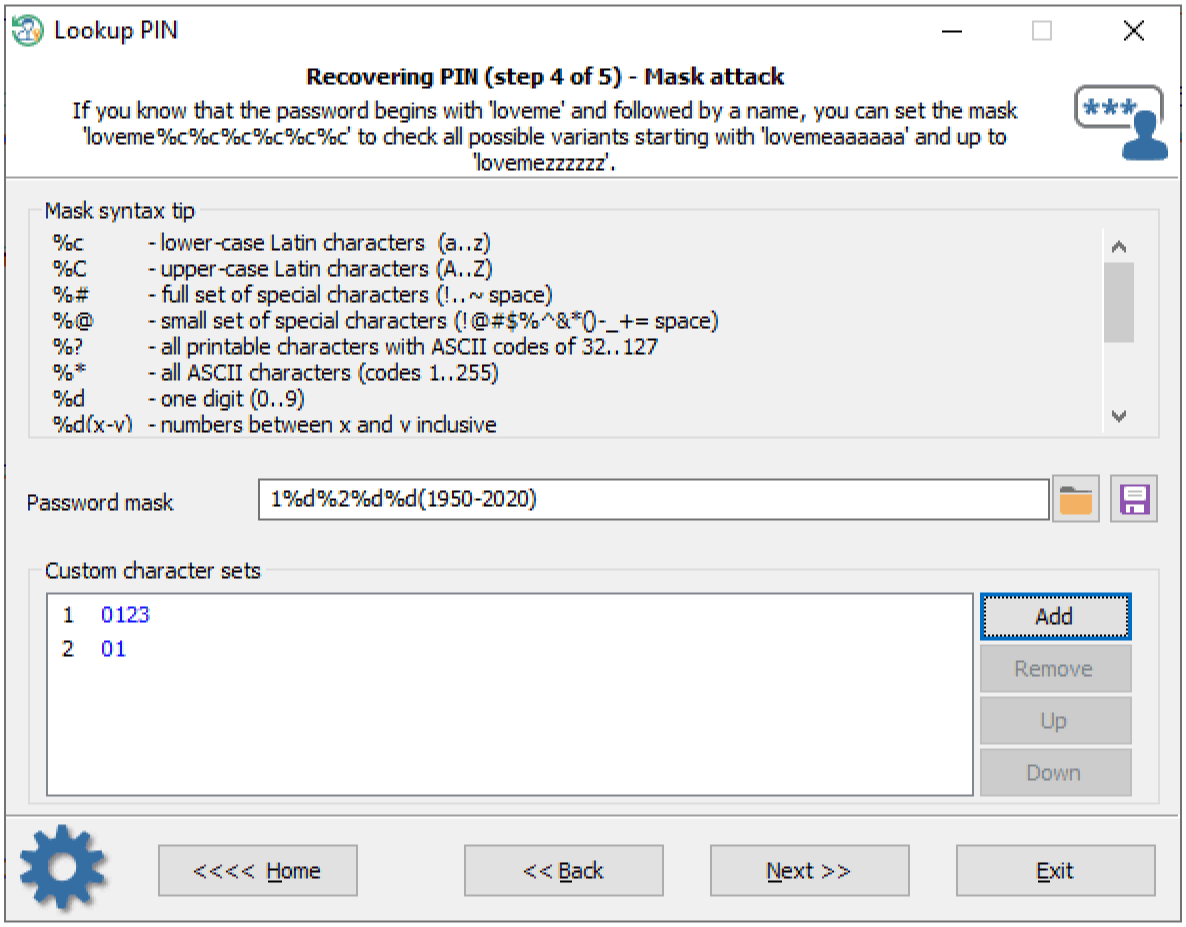
%c	- lower-case Latin characters (a..z)
%C	- upper-case Latin characters (A..Z)
%#	- full set of special characters (!..~ space)
%@	- small set of special characters (!@#$%^&*()-_+= space)
%?	- all printable characters with ASCII codes of 32..127
%*	- all ASCII characters (codes 1..255)
%d	- one digit (0..9)
%d(x-y)	- numbers between x and y inclusive

Password mask: %1%d%1%d%d(1950-2020)

Custom character sets

1 0123

Need more strict limitations?
Implementation 2: **%1%d%2%d%d(1950-2020)**
where **%1** is equal to 0123, **%2** is either 0 or 1



Recovering PIN (step 4 of 5) - Mask attack

If you know that the password begins with 'loveme' and followed by a name, you can set the mask 'loveme%c%c%c%c%c%c' to check all possible variants starting with 'lovemeaaaaaa' and up to 'lovemezzzzzz'.

Mask syntax tip

```
%c       - lower-case Latin characters  (a..z)
%C       - upper-case Latin characters (A..Z)
%#       - full set of special characters (!..~ space)
%@       - small set of special characters (!@#$%^&*()-_+= space)
%?       - all printable characters with ASCII codes of 32..127
%*       - all ASCII characters (codes 1..255)
%d       - one digit (0..9)
%d(x-y)  - numbers between x and y inclusive
```

Password mask: 1%d%2%d%d(1950-2020)

Custom character sets

```
1   0123
2   01
```

Now only 56 800 combinations total.

1.3 Conclusion

Well, you get the idea. The same technique can be applied to restore 6-digit PINs as well.

Take care of yourself, keep your data safe!