PSWD v3 Algorithm

Step 1: User Token Generation

User Provides

User Name → SHA256 → User ID

Master Password

check cache for saved User Token

User Token NOT FOUND → Generate User Token

Generate User Token

User Name + Master Password → User Token → SHA256 (Repeat Hash K1 Times)

This step is designed to take a signficant amount of time so K1 should be relatively large, ie 10^7.

User Token FOUND → User Token → Continue to Step 2

User Token → Save to Cache, sha256(User Name) as Key

Step 2: Password Generation

From Step 1:
- User Name
- Master Password
- User Token

Password Options:
- Domain
- Length
- Caps (T/F)
- Symbols (T/F)
- Symbol Chars
- K2 (# Hashes)

Domain + Master Password + User Token = KEY

pswd → SHA256 (Repeat K2-1 Times)

After K2-1 hashes:

pswd + "1" → SHA256 → pswd1

pswd + "2" → SHA256 → pswd2

Last Hash: concatenate two separate hashes of the result of the previous hash + "1" and "2". This is done to lengthen the result of the hashes.

pswd1 + pswd2 → Base64

Base64 encoding the result of the last SHA256 hash returns a String that can potentially contain all numbers and letters (upper- and lower-case).

Remove Special Characters: '+', '/' and '='

Trim String to Length = pswd

Add Symbols, if requested:

Create a reproducible set of numbers to use as character indices to determine which characters should be replaced by symbols and which symbols to use.

KEY + "numbers1" → SHA256 ... KEY + "numbers4" → SHA256

remove all non-digit characters

nums = result: a list of numbers that will be the same given the same set of start parameters

Determine the number of symbols to add:
- div = nums[0]
- if div <= 3; div = 4
- n = Length / div

For each symbol to add:
- location = (next 2 digits in nums) MOD Length
- sym_location = (next 2 digits in nums) MOD length(Symbol Chars)
- pswd[$location] = Symbol Chars[$sym_location]

If Uppercase letters are NOT requested:

convert pswd to lowercase

Example Password

Step 1: User Token Generation

User Name = dave

Master Password = topS3cret

Before Hash:
User Token = davetopS3cret

After 1 Hash:
User Token = sha256(davetopS3cret) = 0ee174fd149328aca71ba762c1db24...

After 2 Hashes:
User Token = sha256(0ee174fd149328aca71ba762) = 97991d897c3c4d0103633123c7265e...

After 10000000 Hashes:
User Token = e3af81751373ce6eed667e85abf680b...

Step 2: Password Generation

User Name = dave

Master Password = topS3cret

User Token = e3af81751373ce6eed667e85abf680b8...

Domain	= example.com	Symbols	= T
Length	= 24	Symbol Chars	= !@$*-_.?
Caps (T/F)	= T	K2 (# Hashes)	= 250

Before Hash:
KEY = example.comtopS3crete3af81751373ce6ee...

After 1 Hash:
pswd = 3649a452852a065d618e5a711ef9be0d1f2...

After 249 Hashes:
pswd = 4a8b1b1c0784dd0fc6fe7ce07de10e46f39...

Last Hash:

4a8b1b1c0784d{...}1 → SHA256 → f770d7295fd4232fb7dc...

4a8b1b1c0784d{...}2 → SHA256 → 5ef8420826fbca0ba...

f770d7295fd4232fb7dc... + 5ef8420826fbca0ba...

Base 64 encode the hex output of the last pair of hashes remove the special characters + / = :

pswd = 93DXKVUIy33HOJ6UkFqQoje3lBqfDM2zG...

Trim to length of 24:

pswd = 93DXKVUIy33HOJ6UkFqQoje3

If CAPS = T and SYMBOLS = F, this would be the final password

Add Symbols:

example.comtopS3crete3af{...}numbers1 ... → SHA256

f6d200071eaa24645e5e316906196e2324bdb2e...

remove all non-digit characters

62000712464553169061962324279173999999...

Determine the number of symbols to add:
- div = 6
- if div <= 3; div = 4
- n = 24 / 6 = 4

For the first symbol:
- location = 20 MOD 24 = 20
- symbol_location = 00 MOD 8 = 0
- pswd[20] = SYM_CHARS[0] # replaces 'o' with '!'

For the second symbol:
- location = 71 MOD 24 = 23
- symbol_location = 24 MOD 8 = 0
- pswd[23] = SYM_CHARS[0] # replaces '3' with '!'

FINAL PASSWORD:

pswd = 93DXKVU_y33HOJ6U?FqQ!je!