ciphart

key derivation with easier measurable security

caveman

January 9, 2021

i propose *ciphart* — a memory-hard key derivation function with a security gain that's measurable more objectively and more conveniently than anything in class known to date, while maintaining a memory-hardness identical to argon2d[1].

to nail this goal, *ciphart*'s security gain is measured in the unit of *relative entropy bits*. relative to what? relative to the encryption algorithm that's used later on.

therefore, this measure is guaranteed to hold irrespective of adversary's hardware, for as long as the encryption algorithm that's used with *ciphart* is also the same one that's used to encrypt the data afterwards.

my reference implementation is available here[2].

algorithm 1: ciphart version 6

1 **while** *true* **do**
2   **for** $s = 1, 2, \ldots, S$ **do**
3   **for** $l = 1, 2, \ldots, L$ **do**
4   **for** $t = 1, 2, \ldots, T$ **do**
5   **for** $r = 1, 2, \ldots, R$ **do**
6   **if** $t = 1$ **then**
7                         $m_{l,s,t} \leftarrow \mathrm{enc}(m_{l,s,T}, n, k)$;
8   **else**
9                         $m_{l,s,t} \leftarrow \mathrm{enc}(m_{l,s,t-1}, n, k)$;
10                     $n \leftarrow n + 1$;
11                     $k \leftarrow f(m_{l,s,t}[-64:], p, l, s, t)$;
12   **if** $\log_2 n \geq B$ **then**
13   **go to** line 14;
14 **while** *true* **do**
15   **for** $l = 1, 2, \ldots, L$ **do**
16   **if** $\mathrm{len}(k) \geq K$ **then**
17   **return** $k[0:K]$
18         $n \leftarrow n + 1$;
19         $k \leftarrow k \parallel \mathrm{enc}(m_{l,S,T}[1], n, k)$;

1 ciphart

1.1 users' parameters

M	total memory in multiples of 2×64 bytes.
T	maximum number of tasks per lane segment.
R	number of rounds per task.
B	added security in *relative entropy bits*.
L	number of lanes for concurrency.
S	salt.
K	output key size in bytes.
enc	encryption function.
p	passphrase.

1.2 algorithm's parameters

m_i	64 bytes memory for i^{th} task in M-bytes pad.
n	nonce variable with at least 64 bits.

1.3 algorithm's output

k	K-bytes key with $\geq B$ *relative entropy bits*.

1.4 algorithm's steps

shown in algorithm 1.

2 parallelism

3 memory-hardness

4 security interpretation

5 comparison

6 summary

[1] https://github.com/P-H-C/phc-winner-argon2

[2] https://github.com/Al-Caveman/ciphart