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 g = 0, 1, ..., G − 1 do
3          for l = 0, 1, ..., L − 1 do
4              for t = 0, 1, ..., T − 1 do
5                  for r = 0, 1, ..., R − 1 do
6                      i ← gT + t;
7                      if t = 0 then
8                          j ← i + T − 1;
9                      else if t = T − 1 then
10                         j ← i + 1 − T;
11                     else
12                         j ← i + 1;
13                     m_j ← enc(m_i, n_l, k);
14                     n_l ← n_l + 1;
15                     k ← f(m_j, p, l, s, t);
16         if f = true and log_2(n_1 L) ≥ B then
17             go to line 19;
18     f ← true;
19 while true do
20     for l = 1, 2, ..., L do
21         if len(k) ≥ K then return k[0 : K];
22         n ← n + 1;
23         k ← k ‖ enc(m_{l,S,T}[1], n, k);
```

1 ciphart

1.1 parameters

M	total memory in multiples of 2×64 bytes.
T	number of 64-byte 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 internal variables

G	$\leftarrow M/64/T/L$; total number of segments.
m_i	64 bytes memory for i^{th} task in M-bytes pad.
n_l	$\leftarrow lG$; nonce variable for l^{th} lane with at least 64 bits.
f	$\leftarrow$ false; a flag indicating whether the M-byte pad is filled.

1.3 output

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

1.4 steps

shown in algorithm 1.

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

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

2 parallelism

3 memory-hardness

4 security interpretation

5 comparison

6 summary