

()

☒

AI

- AI

-

() 2 - (),

	2026 6 23
(/)	TUI Music Player C “ ” . : <u>cipher</u> : README.md : Emoji , , Nix . Nix . AI : GPT-5.5 : OpenAI : 2026. 06. 23. URL : https://chatgpt.com 0 255 , XOR . - `P[p]`: `p` - `N[n]`: 16 nonce `n` - `K[k]`: 32 - `S[s]`: 256 - `ROTL8(x, q)`: 8 `x` `q` - `⊕`: XOR $ROTL8(x, q) = ((x \ll q) \mid (x \gg (8 - q))) \bmod 256$, `q = 0` `x` . ### 1. 32 `K` . `D` XOR . ```text K[0] = K[0] ⊕ D

```

D = 0x43 ( )
D = 0x41 ( )
...

`r = 0 ... 8191` 8192

```text
a = (p + r) mod 32
b = (a + 11) mod 32
c = (a + 23) mod 32

m = K[b] + P[p] + N[(p + r) mod 16] + r

K[a] = ROTL8(K[a] ⊕ m, K[c] mod 8) + K[c]
...

```text
K[r mod 32] = K[r mod 32]
               ⊕ ROTL8(K[(r + 17) mod 32], r mod 8)
...

`N`

### 2. 256

`S[n] = n` 1024
nonce

```text
S[n] = n, 0 ≤ n < 256
j = 0

for n = 0 ... 1023:
 i = n mod 256
 j = j + S[i] + K[n mod 32] + N[n mod 16]
 swap(S[i], S[j])
...

`j` `mod 256`

3.

```

	<pre>         `t`                                `Z[t]`          .      ```text     i = i + 1     j = j + S[i] + K[t mod 32]     swap(S[i], S[j])      u      = S[(S[i] + S[j]) mod 256]     Z[t] = u <math>\oplus</math> ROTL8(K[(t + u) mod 32], t mod 8)     ```          `M[t]`      XOR      `C[t]`      .      ```text     C[t] = M[t] <math>\oplus</math> Z[t]     ```      XOR  `x <math>\oplus</math> y <math>\oplus</math> y = x`      .      ```text     M[t] = C[t] <math>\oplus</math> Z[t]     ```      ,      .                                `buffer[4096]`      ,      `t`      .      ### 4.                                  `x`      32      `A`      .      `L`      :      ```text     a = L mod 32     b = (a + 7) mod 32     c = (a + 19) mod 32      A[a] = ROTL8(A[a] <math>\oplus</math> x <math>\oplus</math> A[c], A[b] mod 8) + A[b] + a     A[c] = A[c] <math>\oplus</math> ROTL8(x + A[a], a mod 8)     L      = L + 1     ```                                  `L` 8      little-endian      .      256      . </pre>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<pre>```text for n = 0 ... 255:     v = A[(n + 1) mod 32] + A[(n + 13) mod 32] + n     A[n mod 32] = A[n mod 32] ⊕ ROTL8(v, n mod 8) ```  16 XOR .  ```text TAG[n] = A[n] ⊕ A[n + 16], 0 ≤ n &lt; 16 ```</pre>
( 2 )	1

```

encrypt.c
#include "gerbera.h"

#include <errno.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

static const uint8_t magic[GERBERA_MAGIC_SIZE] =
 {'M', 'I', 'Z', 'U', 'K', 'I', 0xF0, 0x9F, 0x8E, 0x80};

static void encode_u64_le(uint8_t bytes[8], uint64_t value)
{
 unsigned n;
 for (n = 0; n < 8U; ++n) {
 bytes[n] = (uint8_t)(value >> (n * 8U));
 }
}

static int random_nonce(uint8_t nonce[GERBERA_NONCE_SIZE])
{
 static int initialized = 0;

 if (!initialized) {
 srand((unsigned)time(NULL));
 initialized = 1;
 }

 for (size_t i = 0; i < GERBERA_NONCE_SIZE; ++i) {
 nonce[i] = (uint8_t)rand();
 }

 return 1;
}

int encrypt_file(const char *input_path, const char *output_path,
 const char *password)
{
 FILE *input = NULL;
 FILE *output = NULL;
 uint8_t nonce[GERBERA_NONCE_SIZE];
 uint8_t size_bytes[8];
 uint8_t buffer[GERBERA_BUFFER_SIZE];
 uint8_t tag[GERBERA_TAG_SIZE];
 GerberaCipher cipher;
 GerberaAuth auth;
 long input_size;
 size_t count;
 int result = 1;

 input = fopen(input_path, "rb");
 if (input == NULL || fseek(input, 0, SEEK_END) != 0 ||
 (input_size = ftell(input)) < 0 || fseek(input, 0, SEEK_SET) != 0) {
 fprintf(stderr, "Could not open the input file or determine its size.\n");
 goto cleanup;
 }

 if (!random_nonce(nonce)) {
 fprintf(stderr, "Could not generate a secure random nonce.\n");
 goto cleanup;
 }

 output = fopen(output_path, "wb");
 if (output == NULL) {
 fprintf(stderr, "Could not create the output file.\n");
 goto cleanup;
 }

 encode_u64_le(size_bytes, (uint64_t)input_size);
 if (fwrite(magic, 1, sizeof(magic), output) != sizeof(magic) ||
 fwrite(nonce, 1, sizeof(nonce), output) != sizeof(nonce) ||
 fwrite(size_bytes, 1, sizeof(size_bytes), output) != sizeof(size_bytes)) {
 fprintf(stderr, "Could not write the encrypted file header.\n");
 goto cleanup;
 }

 gerbera_cipher_init(&cipher, password, nonce);
 gerbera_auth_init(&auth, password, nonce);
}

```

( )	.
-----	---