

Forum: [Hack Forums](#)

Thread: [Improved RSA implementation](#)

putnik1313 - May 26, 2025, 10:22 am

new function replacement:

----- Modification 1 -----

old function: modiv

new version: extended_gcd_modinv

Code // Extended Euclidean Algorithm for Modular Inverse

```
int extended_gcd_modinv(mpz_t rop, const mpz_t a, const mpz_t m) {
    mpz_t r, newr, t, newt, quotient, tmp;
    mpz_inits(r, newr, t, newt, quotient, tmp, NULL);
    mpz_set(r, m); mpz_set(newr, a);
    mpz_set_ui(t, 0); mpz_set_ui(newt, 1);

    while (mpz_cmp_ui(newr, 0) != 0) {
        mpz_fdiv_q(quotient, r, newr);
        mpz_set(tmp, newr);
        mpz_mul(newr, quotient, newr); mpz_sub(newr, r, newr);
        mpz_set(r, tmp);
        mpz_set(tmp, newt);
        mpz_mul(newt, quotient, newt); mpz_sub(newt, t, newt);
        mpz_set(t, tmp);
    }
    if (mpz_cmp_ui(r, 1) > 0) {
        mpz_clears(r, newr, t, newt, quotient, tmp, NULL);
        return 0; // No inverse
    }
    if (mpz_cmp_ui(t, 0) < 0)
        mpz_add(t, t, m);
    mpz_set(rop, t);
    mpz_clears(r, newr, t, newt, quotient, tmp, NULL);
    return 1;
}
```

[code]

----- Modification 2 -----

old function: [b]ct_mpowm[/b]

new version: [b]montgomery_ladder_powm[/b]

[code]

// Montgomery Ladder for Modular Exponentiation (side-channel resistant)

```

void montgomery_ladder_powm(mpz_t rop, const mpz_t base, const mpz_t exp, const mpz_t mod)
{
    mpz_t R0, R1;
    mpz_inits(R0, R1, NULL);
    mpz_set_ui(R0, 1);
    mpz_set(R1, base);
    size_t nbits = mpz_sizeinbase(exp, 2);
    for (ssize_t i = nbits - 1; i >= 0; --i) {
        if (mpz_tstbit(exp, i) == 0) {
            mpz_mul(R1, R0, R1); mpz_mod(R1, R1, mod);
            mpz_mul(R0, R0, R0); mpz_mod(R0, R0, mod);
        } else {
            mpz_mul(R0, R0, R1); mpz_mod(R0, R0, mod);
            mpz_mul(R1, R1, R1); mpz_mod(R1, R1, mod);
        }
    }
    mpz_set(rop, R0);
    mpz_clears(R0, R1, NULL);
}

```

----- Modification 3 -----

old function: rsa_decrypt

new version: rsa_decrypt_blinded

Code // --- RSA blinding ---

```

void rsa_decrypt_blinded(mpz_t out, const mpz_t in, const rsa_private_key *priv) {
    INFO("rsa_decrypt_blinded: Starting blinded RSA decryption");
    gmp_randstate_t state;
    gmp_randinit_mt(state);
    gmp_randseed_ui(state, (unsigned long)time(NULL) ^ (unsigned long)rand());

    mpz_t r, r_inv, blinded, tmp, gcd;
    mpz_inits(r, r_inv, blinded, tmp, gcd, NULL);

    // Generate random r with 1 < r < n, gcd(r, n) == 1
    int attempts = 0;
    do {
        mpz_urandomm(r, state, priv->n);
        mpz_gcd(gcd, r, priv->n);
        attempts++;
        INFO("Blinding attempt #%d: r generated", attempts);
        dump_mpz(r, "r");
        dump_mpz(gcd, "gcd(r, n)");
    } while (mpz_cmp_ui(r, 1) <= 0 || mpz_cmp_ui(gcd, 1) != 0);

    OK("Blinding factor r found in %d attempts", attempts);

    // blinded = in * r^e mod n
    INFO("Blinding input and computing r^e mod n");
    montgomery_ladder_powm(tmp, r, priv->e, priv->n);
    dump_mpz(tmp, "r^e mod n");
    mpz_mul(blinded, in, tmp);
}

```

```

mpz_mod(blinded, blinded, priv->n);
dump_mpz(blinded, "blinded input");

// decrypted = blinded^d mod n
INFO("Decrypting blinded value");
montgomery_ladder_powm(tmp, blinded, priv->d, priv->n);
dump_mpz(tmp, "blinded^d mod n");

// r_inv = r^-1 mod n
if (mpz_invert(r_inv, r, priv->n) == 0) {
    ERROR("Failed to invert blinding factor r modulo n (r has no inverse)");
} else {
    OK("Computed modular inverse of r");
    dump_mpz(r_inv, "r_inv");
}

// out = decrypted * r_inv mod n
mpz_mul(out, tmp, r_inv);
mpz_mod(out, out, priv->n);
dump_mpz(out, "final plaintext output");

mpz_clears(r, r_inv, blinded, tmp, gcd, NULL);
gmp_randclear(state);
OK("rsa_decrypt_blinded completed successfully");
}

```

Successfully improved security and speed.

Updated OAEP encoder/decoder:

```

Code __attr_nodiscard __attr_hot int oaep_encode(
    const uint8_t * __restrict__ in, size_t inlen, uint8_t * __restrict__ out, size_t k,
    const uint8_t * __restrict__ label, size_t labellen) __noexcept
{
    size_t hLen = 32;
    DBG("oaep_encode: Entry (inlen=%zu, k=%zu, labellen=%zu)", inlen, k, labellen);

    if (k < 2 * hLen + 2) {
        ERROR("oaep_encode: Output buffer too small for OAEP: k=%zu", k);
        return 0;
    }
    if (inlen > k - 2 * hLen - 2) {
        ERROR("oaep_encode: Input too large: inlen=%zu, max=%zu", inlen, k - 2 * hLen - 2);
        return 0;
    }

    uint8_t lHash[32];
    sha256(label ? label : (const uint8_t *)"", labellen, lHash);
    dump_hex(lHash, 32, "oaep_encode: lHash");

    size_t ps_len = k - inlen - 2 * hLen - 2;
    uint8_t *DB = (uint8_t *)malloc(k - hLen - 1);
    if (!DB) {
        ERROR("oaep_encode: malloc failed for DB (%zu bytes)", k - hLen - 1);
    }
}

```

```

    return 0;
}
memcpy(DB, lHash, hLen);
memset(DB + hLen, 0, ps_len);
DB[hLen + ps_len] = 0x01;
memcpy(DB + hLen + ps_len + 1, in, inlen);

dump_hex(DB, k - hLen - 1, "oaep_encode: DB before masking");

uint8_t *seed = (uint8_t *)malloc(hLen);
if (!seed) {
    ERROR("oaep_encode: malloc failed for seed");
    free(DB);
    return 0;
}
if (!__unlikely(rsa_get_secure_random_bytes(seed, hLen))) {
    ERROR("oaep_encode: secure random failed, using weak fallback");
    for (size_t i = 0; i < hLen; ++i)
        seed[i] = rand() & 0xFF;
}
dump_hex(seed, hLen, "oaep_encode: seed");

uint8_t *dbMask = (uint8_t *)malloc(k - hLen - 1);
if (!dbMask) {
    ERROR("oaep_encode: malloc failed for dbMask");
    free(DB); free(seed);
    return 0;
}
mgf1_sha256(seed, hLen, dbMask, k - hLen - 1);
for (size_t i = 0; i < k - hLen - 1; ++i)
    DB[i] ^= dbMask[i];
dump_hex(DB, k - hLen - 1, "oaep_encode: masked DB");

uint8_t *seedMask = (uint8_t *)malloc(hLen);
if (!seedMask) {
    ERROR("oaep_encode: malloc failed for seedMask");
    free(DB); free(seed); free(dbMask);
    return 0;
}
mgf1_sha256(DB, k - hLen - 1, seedMask, hLen);
for (size_t i = 0; i < hLen; ++i)
    seed[i] ^= seedMask[i];
dump_hex(seed, hLen, "oaep_encode: masked seed");

out[0] = 0x00;
memcpy(out + 1, seed, hLen);
memcpy(out + 1 + hLen, DB, k - hLen - 1);

dump_hex(out, k, "oaep_encode: OAEP encoded output");

free(DB);
free(seed);
free(dbMask);

```

```

free(seedMask);
OK("oaep_encode: Success");
return 1;
}

__attr_nodiscard __attr_hot int oaep_decode(
    const uint8_t * __restrict__ in, size_t k, uint8_t * __restrict__ out, size_t * __restrict__ outlen,
    const uint8_t * __restrict__ label, size_t labellen) __noexcept
{
    size_t hLen = 32;
    DBG("oaep_decode: Entry (k=%zu, labellen=%zu)", k, labellen);
    dump_hex(in, k, "oaep_decode: input");

    if (k < 2 * hLen + 2) {
        ERROR("oaep_decode: k too small for OAEP: k=%zu", k);
        return 0;
    }
    if (!in || !out || !outlen) {
        ERROR("oaep_decode: null argument(s): in=%p out=%p outlen=%p", (void*)in, (void*)out,
(void*)outlen);
        return 0;
    }
    if (in[0] != 0x00) {
        ERROR("oaep_decode: leading byte is not 0x00");
        return 0;
    }

    const uint8_t *maskedSeed = in + 1;
    const uint8_t *maskedDB = in + 1 + hLen;

    uint8_t seed[32];
    memcpy(seed, maskedSeed, hLen);
    dump_hex(seed, hLen, "oaep_decode: maskedSeed");

    uint8_t *DB = (uint8_t *)malloc(k - hLen - 1);
    if (!DB) {
        ERROR("oaep_decode: malloc failed for DB");
        return 0;
    }
    memcpy(DB, maskedDB, k - hLen - 1);
    dump_hex(DB, k - hLen - 1, "oaep_decode: maskedDB");

    uint8_t *seedMask = (uint8_t *)malloc(hLen);
    if (!seedMask) {
        ERROR("oaep_decode: malloc failed for seedMask");
        free(DB);
        return 0;
    }
    mgf1_sha256(DB, k - hLen - 1, seedMask, hLen);
    for (size_t i = 0; i < hLen; ++i)
        seed[i] ^= seedMask[i];
    dump_hex(seed, hLen, "oaep_decode: seed");

```

```

uint8_t *dbMask = (uint8_t *)malloc(k - hLen - 1);
if (!dbMask) {
    ERROR("oaep_decode: malloc failed for dbMask");
    free(DB);
    free(seedMask);
    return 0;
}
mgf1_sha256(seed, hLen, dbMask, k - hLen - 1);
for (size_t i = 0; i < k - hLen - 1; ++i)
    DB[i] ^= dbMask[i];
dump_hex(DB, k - hLen - 1, "oaep_decode: DB");

uint8_t lHash[32];
sha256(label ? label : (const uint8_t *)"", labellen, lHash);
dump_hex(lHash, 32, "oaep_decode: lHash");
// Use constant time comparison in production for security
if (__likely(ct_memcmp(DB, lHash, hLen))) {
    ERROR("oaep_decode: lHash mismatch");
    free(DB); free(seedMask); free(dbMask);
    return 0;
}

// OAEP decode (ending section)
size_t i = hLen;
while (i < k - hLen - 1 && DB[i] == 0x00)
    ++i;
if (i == k - hLen - 1 || DB[i] != 0x01) {
    ERROR("oaep_decode: 0x01 separator not found in DB");
    free(DB);
    free(seedMask);
    free(dbMask);
    return 0;
}

size_t mLen = k - hLen - 1 - i - 1;
memcpy(out, DB + i + 1, mLen);
*outlen = mLen;

dump_hex(out, mLen, "oaep_decode: Decoded output");

free(DB);
free(seedMask);
free(dbMask);
OK("oaep_decode: Success");
return 1;
} E/D Status:      Success
OAEP Encoding:     Success
OAEP Decoding:     Success
PEM Loader:        Success
Serializer:        Success
Deserializer:      Success
SCA CM Status:     Success
Satisfied: 70%

```

webdev77 - May 26, 2025, 2:27 pm

This is entirely useless ai spam if you don't post the full code. Bunch of functions in here don't exist.

putnik1313 - May 26, 2025, 2:31 pm

(May 26, 2025 02:27 PM)webdev77 Wrote: [\[CITE\]](#) This is entirely useless ai spam if you don't post the full code. Bunch of functions in here don't exist.

my friend, full code is 2k lines, you can find it here [full code](#).

webdev77 - May 26, 2025, 2:50 pm

(May 26, 2025 02:31 PM)putnik1313 Wrote: [\[CITE\]](#) my friend, full code is 2k lines, you can find it here [full code](#).

This isn't even what you posted. Next time be more careful about what you're posting.

putnik1313 - May 26, 2025, 2:54 pm

(May 26, 2025 02:50 PM)webdev77 Wrote: [\[CITE\]](#) This isn't even what you posted. Next time be more careful about what you're posting.

can you elaborate please? what is not what i posted? im working on it even right now, since i posted it, i've made more then 5 updates to it, of course is different ?

?

thank you anyway for your instructions on how to post... very helpful

webdev77 - May 26, 2025, 3:01 pm

(May 26, 2025 02:54 PM)putnik1313 Wrote: [\[CITE\]](#) can you elaborate please? what is not what i posted? im working on it even right now, since i posted it, i've made more then 5 updates to it, of course is different ?

?

thank you anyway for your instructions on how to post... very helpful

Exactly none of the code you posted comes from that repo. Commit history is public. It was never part of that repo. Care to explain?

putnik1313 - May 26, 2025, 3:04 pm

(May 26, 2025 03:01 PM)webdev77 Wrote: [\[CITE\]](#) Exactly none of the code you posted comes from that repo. Commit history is public. It was never part of that repo. Care to explain?

my friend you are out of your mind, either you're blind either you talking nonsense, what should i profit from posting non-sense?

(May 26, 2025 03:01 PM)webdev77 Wrote: [\[CITE\]](#) Exactly none of the code you posted comes from that repo. Commit history is public. It was never part of that repo. Care to explain?

you just made me double check just to make sure, everything is there, so you talking non-sense, sorry my friend.

are you sure you looked the right file? there are 2 files, you need to check here

https://github.com/MrkFrcsl98/RSA_Algorithm/blob/main/src/minimal/rsa.h