

**Algorithm type:** Linear congruential generator

**Modulus:**  $2^{31}$

**Multiplier:** 1103515245

**Increment:** 12345

**Random generator C code:**

```
static inline uint32_t _sclprng_rand(uint64_t *seed, uint32_t upper_limit)
{
#define PRNG_INCREMENT (12345)
#define PRNG_MULTIPLIER (1103515245)

    *seed = (*seed) * PRNG_MULTIPLIER + PRNG_INCREMENT;
    return (uint32_t)((*seed) / (upper_limit * 2)) % upper_limit;
}
```

**Seed generation:**

Input:

- ⑩ 32-byte secret encryption key
- ⑩ 16-byte secret initialization vector (IV)

1. Allocate 16-byte array “seed\_key” and copy content of IV to this array.
2. Allocate 16-byte array “seed\_buf” and copy first 16 bytes of key to this array.
3. Perform AES-128 ECB operation where “seed\_buf” is data and “seed\_key” is key. Copy the result 16 bytes back to “seed\_buf”.
4. Copy last 16 bytes of key to “seed\_key” array.
5. Repeat step 3.
6. 16-byte “seed\_buf” consists of four 4-byte words. Make XOR operation between them. As a result, you will have a 4-byte seed. Example on C:

```
seed1 = (uint32_t*)(seed_buf + 0);
seed2 = (uint32_t*)(seed_buf + 4);
seed3 = (uint32_t*)(seed_buf + 8);
seed4 = (uint32_t*)(seed_buf + 12);

seed = (uint32_t)((*(seed1) ^ *(seed2) ^ *(seed3) ^ *(seed4));
```