**1. Key struct Initialization:**

***gnutls_privkey_t privkey;***
***ret =gnutls_privkey_init(&privkey);***

*In cryptography, X.509 is a standard that defines the format of public key certificates.*

**2. Call generation function:**

**gnutls_privkey_generate2(gnutls_privkey_t pkey, gnutls_pk_algorithm_t algo, unsignedint bits, unsignedint flags, const gnutls_keygen_data_st *data, unsigned data_size)**

**3. Call x.509 generation function:**

**gnutls_x509_privkey_generate2(gnutls_x509_privkey_tkey, gnutls_pk_algorithm_t algo, unsignedint bits, unsignedint flags, const gnutls_keygen_data_st *data, unsigned data_size)**

**3. RSA key generation:**

**_gnutls_pk_generate_keys(...)**
**?????**

**3. provable-prime.c :**

staticconstuint16_t primes[] = {
3, 5, 7, 11, 13, 17, 19, ...,
65521 }

---

/* seed is handled as mpz_t instead of simply using INCREMENT
* for the few (unlikely) cases where seed overflows. */
**nettle_mpz_set_str_256_u(s, seed_length, seed);**

---

/* c = Hash(seed) XOR Hash(seed+1) */
**nettle_mpz_get_str_256(tseed_length, tseed, s);**

---

/* c = 2^(bits-1) + (c mod 2^(bits-1)) */
**highbit = 1L << (bits - 1);**

---

Show rest of the flow on :
https://github.com/psekan/gnutls/blob/master/lib/nettle/int/rsa-keygen-fips186.c