Challenges in Generating Keys for Asymmetric-Key Algorithms

Allen Roginsky

CMVP NIST

November 2015

Overview of the talk

- The use of Random Number Generators
- Key generation for the symmetric-key algorithms
  - AES, Triple-DES, special cases of key usage
  - The applicable NIST publications
- Generation of keys for the asymmetric algorithms
  - What makes the RSA key generation difficult

The use of the RNGs

- Each secret key has to be unpredictable
  - Entropy sources
  - Deterministic Random Generators

- Estimating the amount of randomness

- When can a random number be used directly as a cryptographic key?
  - The key may need to be modified to possess some additional properties

Key Generation for the symmetric-key algorithms

- AES keys
- Triple-DES keys
- Nonces and IVs
- Keys for storage applications
- The dependency among the bits of (certain) keys

NIST Publications

- Implementation Guidance 7.8
  - Six main methods
  - Post-processing
- SP 800-132
- SP 800-133
  - The post-processing is left for SP 800-90A

Generation of Private Keys for the Asymmetric Algorithms

- Key generation for the signature algorithms: DSA, ECDSA and RSA
- The same issues must be addressed when generating private keys for the asymmetric-key-based key agreement and key transport schemes
- Again, it all starts (but not ends) with SP 800-133 / IG 7.8

Digital Signature Algorithm

- First, the domain parameters: p, q, g need to be generated
- If N is the bit length of q, then obtain an N+64 bit random string
- The requirements of IG 7.8 or SP 800-133 apply as if this string were a key
- Convert the bit string into an integer, c
- The private key x is set to

  c (mod (q-1)) + 1.

  This, given the size of c, guarantees a sufficiently uniform distribution of private keys
- The public key is $y=g^x$ (mod p).

Elliptic Curve Digital Signature Algorithm

- First, the domain parameters: the field; the parameters that define the curve; base point G, the prime n which is the order of G, the co-factor h. (The number of points on the curve is n*h.)
- If N is the bit length of n, then obtain an N+64 bit random string
- The requirements of IG 7.8 or SP 800-133 apply as if this string were a key
- Convert the bit string into an integer, c
- The private key d is set to

  c (mod (n-1)) + 1.

  This, given the size of c, guarantees a sufficiently uniform distribution of private keys
- The public key is a point Q = dG on the curve

RSA Signature Algorithm

- More complicated key generation than with the other algorithms due to the different nature of the RSA
- First, generate the seed, in compliance with SP 800-133 or IG 7.8
- Use the seed as a starting point when generating an auxiliary prime p_2
- Use an updated seed to generate p_1 . Consult FIPS 186-4 for the size of these primes and for the required number of the Miller-Rabin tests

RSA Signature Algorithm (continues)

- Use a (further) updated seed to generate p
- The prime may either be generated as a probable prime or as a provable prime. Even with a provable prime there is a (tiny) chance that it will be composite
- Generate the prime q in a similar way and check the conditions on the (p, q) pair

Summary

- Key generation is one of several ways to establish a key
- Key generation requires entropy
- Key generation requires an approved random number generator
- Key generation for the symmetric algorithms is straightforward, but has to comply with SP 800-133 / IG 7.8
- Generating keys for ECDSA is more complicated
- In case of RSA, the compliance with SP 800-133 / IG 7.8 is just a staring point.