Example 127. To (naively) brute-force DES, how much data must we encrypt?

Solution. DES uses 56-bit keys. We need to encrypt 2^{56} times 64 bits.
This is $2^{56} \cdot 8 = 2^{59}$ byte, or 512 pebibyte (binary analog of petabyte) or 576 petabyte (since $2^{59} \approx 5.76 \cdot 10^{17}$).
For comparison. Up to 2012, CERN collected about 200 petabyte of data looking for the Higgs boson.
In 2009, World of Warcraft uses 1.3 petabytes of storage to maintain its game.
How long will this take? Of course, this depends on your machine. Assume our CPU is very fast and can encrypt 500 MB/sec. Then, this will take us about $2 \cdot 5.76 \cdot 10^8$ sec, or about 36.5 years.
Of course, such a brute-force attack can be fully parallelized to quickly bring this number down to less than an hour, for a powerful attacker. Also, the attack can be sped up considerably by careful design (like early aborts).

Review 128. Meet-in-the-middle attack on 3DES.

Example 129. (use as PRG) ANSI X9.17 is a U.S. federal standard for a PRG based on 3DES.

Input: random, secret 64 bit seed s, key k for 3DES (keying option 2)
Produce a random number as follows:

- obtain current time D, compute $t = \mathrm{3DES}_k(D)$
- output as random number $x = \mathrm{3DES}_k(s \oplus t)$
- update the seed to $s = \mathrm{3DES}_k(x \oplus t)$ for future use

Comment. The same approach can be applied to any block cipher.

5 AES

5.1 Finite fields

Example 130. We have already seen xor in several cryptosystems. Note that a single xor operation as in the one-time pad or stream ciphers provides no diffusion (for that reason, we must never reuse the same keystream).

When designing a cipher it may be nice to replace xor of N bit blocks with an operation that does provide some diffusion.

- A tiny amount of diffusion is provided by instead using addition modulo 2^N.
  Due to carries, one bit flip in the input can propagate to more than one bit flipped in the output.
- More diffusion can be achieved using operations (multiplication/inversion) in finite fields like $\mathrm{GF}(2^N)$.
  [We only need to make sure in our design that we don't multiply with zero.]

A **field** is a set of elements which can be added/subtracted as well as multiplied/divided by according to the usual rules.

In particular, a field always has distinguished elements 0 and 1, which are the neutral elements with respect to addition and multiplication, respectively.

Armin Straub
straub@southalabama.edu

Example 131. The rational numbers $\mathbb{Q}$, the real numbers $\mathbb{R}$, and the complex numbers $\mathbb{C}$ all are fields, which you have seen before. They contain infinitely many elements.

Cryptographic applications require finite structures. Correspondingly, our focus will be on **finite fields**, that is, fields consisting of only a finite number of elements.

Example 132. Let p be a prime. The residues modulo p form a field, often denoted as $\mathrm{GF}(p)$.

GF is short for **Galois field**, which is another word for finite field.

Note that we can divide by any element! (Except the zero residue but, of course, we can never divide by 0).

Example 133. The residues modulo 21 (or any other composite number) are not a field.

We can add/subtract and multiply these numbers, but we cannot always divide. Specifically, we cannot divide by elements like $3, 6, 7, \ldots$ even though these are nonzero (we can, of course, never divide by zero).

Note. We have already seen that this seemingly slight deficiency has "terrible" consequences. For instance, the quadratic equation $x^2 = 1$ has more than the two solutions $x = \pm 1$ modulo 21 (namely, ± 8 as well).

AES is built upon byte operations (in contrast to DES, which is built on bit operations). Each of the 2^8 bytes represents one of the 2^8 elements of the finite field $\mathrm{GF}(2^8)$.

Note. We do not yet know what $\mathrm{GF}(2^8)$ is. It cannot be the residues modulo 2^8, because we just observed that the residues modulo n are a field only of n is prime.

We'll meet this field next time...