1. We shall use the following model for computational cost of breaking a cipher with a 64-bit key:

- A year 2003 computer node that costs 1000 EUR can test $10^7$ keys per second.
- Moore’s “law” will continue to hold; the amount of computing power that can be purchased with 1000 EUR will double every 18 months (exponential growth).
- Significant advances in theoretical cryptanalysis will not occur.

Estimate the time required to break the key (on average case) by the following groups:

a) National Security Agency (http://www.nsa.gov). 5 000 000 000 EUR annual budget for hardware.
b) CSC Oy (http://www.csc.fi), 5 000 000 EUR annual budget for hardware.
c) HUT Krypto Group (http://www.tcs.hut.fi/Research/Crypto/). 5 000 EUR annual budget for hardware.

2. Counter mode (CTR) essentially turns a block cipher into a stream cipher cipher (keystream generator). We shall use a zero IV (initial value):

Encryption:

\[
E_K(0) \oplus P_0 = C_0 \\
E_K(1) \oplus P_1 = C_1 \\
\ldots \\
E_K(n) \oplus P_n = C_n
\]
Decryption:

\[ E_K(0) \oplus C_0 = P_0 \]
\[ E_K(1) \oplus C_1 = P_1 \]
\[ \vdots \]
\[ E_K(n) \oplus C_n = P_n \]

Here \( K \) is the secret key, \( P_i \) is the plaintext block and \( C_i \) is the corresponding ciphertext block.

\( 2^{40} \) bits of keystream is available. Is there any way of distinguishing the keystream from a random sequence without an exhaustive key search?

3. Count the number of different 8-bit block ciphers with an 8-bit key.

Definition. An \( n \)-bit block cipher is a function \( E : V_n \times \mathcal{K} \rightarrow V_n \), such that for each key \( K \in \mathcal{K} \), \( E(K, P) \) is an invertible mapping (the encryption function for \( K \)) from \( V_n \) to \( V_n \), written \( E_K(P) \). The inverse mapping is the decryption function, denoted \( D_K(C) \). \( C = E_K(P) \) denotes that ciphertext \( C \) results from encrypting plaintext \( P \) under \( K \).

Hints: Count ALL variants, good and bad – even identity transform is included. In this case the key space size is \( |\mathcal{K}| = 2^8 \). It might be helpful to think about how much memory would be required to store a general block cipher as a table.

4. Given an RSA public modulus \( n = p \cdot q \), public exponent \( e \) and the private exponent \( d = e^{-1} \mod (p-1)(q-1) \), find factors \( p \) and \( q \).

\[ n = 65837663925249858325414050808539188031 \]
\[ e = 17 \]
\[ d = 154912150412352607810683732878343115217 \]

Testing the variables for correctness (example):

\[ 123456789e \equiv 62733577232488045926157258337023432524 \text{ (mod } n) \]
\[ 62733577232488045926157258337023432524d \equiv 123456789 \text{ (mod } n) \]

Try to find an algebraic solution to the problem that utilizes the knowledge of the private (“secret”) parameter \( d \) and that is faster than direct factorization of \( n \).

Hints: Mathematica and Maple directly support computations on large integers, as do certain programming languages (e.g. BC, Java and Python). Support for C and C++ can be added using the GNU Multiprecision library \texttt{http://swox.com/gmp/}.

5. Find a collision for the first 48 bits (six bytes) of the SHA-1 hash function output. Include a detailed description of the method that you used (with source code if possible).

Example (using hexadecimal notation):

\footnote{Adopted from Definition 7.1 (p. 224) in Menezes et al, Handbook of Applied Cryptography, CRC Press 1996.}
SHA1(77 28 CC 1E 73 0A) =
51 D2 E8 D0 79 11 46 54 A6 00 A7 44 36 2F 17 97 FF E9 93 A9

SHA1(13 DA FC 00 E4 36) =
51 D2 E8 D0 79 11 D8 A8 46 FE 04 79 30 48 A0 6E 50 84 74 FF

Since the first 48 bits of the 160-bit message digest are the same (51 D2 E8 D0 79 11), this is a collision in the sense that is required by the exercise. Note that collision search will take a long time unless you use a $O(\sqrt{n})$ algorithm.

Hints: A reasonable C-language implementation of a collision finding algorithm runs for about one minute on a 1.4 GHz Athlon; Java implementation running on a slow computer may require several hours! Test your algorithm on a smaller problem first (e.g. 32 bits).

More information regarding SHA-1 is available from:

- RFC 3174 contains an implementation of SHA-1: http://www.ietf.org/rfc/rfc3174.txt?number=3174
- The OpenSSL library contains an implementation as well: http://www.openssl.org