T-79.159 Cryptography and Data Security **Spring 2004**
Tutorial 4
Thursday 19.2.2004 14.15, room T3.
Markku-Juhani O. Saarinen `<mjos@tcs.hut.fi>`

Collision search

We define a hash function from 4-character strings to 4-character strings using the UNIX `crypt(3)` function. This function is available on all UNIX systems (you may have to link it using `-lcrypto`). Source for the hash function is as follows:

```
#include <crypt.h>
#include <string.h>

/* hash src and place the result to dst */
void chash(char *dst, const char *src)
{
  /* salt is set to ".." and we take chars 2..6 of the result */
  memcpy(dst, crypt(src, "..") + 2, 4);
  dst[4] = 0;
}
```

If your system lacks `crypt(3)`, you can download an implementation from:

`http://www.tcs.hut.fi/~mjos/src/v7crypt.c`

A prototype for this implementation is `char *v7crypt(char *pw, char *salt);` (you will have to change the function name in `chash()` above).

Your task is to devise a program that finds collisions in this hash function. Examples:

```
chash(xanx)=.0Hc  chash(e.sc)=.0Hc
chash(GoR9)=DtHU  chash(AIyS)=DtHU
chash(nn.T)=W08.  chash(f00f)=W08.
chash(p5Tp)=9np9  chash(bUiw)=9np9
```

See page 17 of fourth lecture slides ("Hashes and Message Digests") for one possible algorithm. The running time of this algorithm shouldn't exceed few seconds.

Questions:

1. Each character is actually base-64 encoded and thus a 4-character string contains $2^{4*6} = 2^{24} = 16777216$ possibilities. What is the expected running time (in steps) for collision search ?

2. The full `crypt(3)` takes in 56 bits and produces a 64-bit hash. How difficult would it be to find collisions for this hash ?

If you have problems with the programming task, just please show up on Thursday.