D.) Specifications of the Academic Signature-PRNG or "Fleas PRNG"

The Fleas PRNG is a component of utmost importance to security. It is organized as a class in C++. The security of the PRNG depends on the internal code in the class as well as on the way the public functions are called from other modules. The instance of the class used in Academic Signature is the global variable "zuf". The code of the class is contained in the module "random_clx.cpp".

As in the specification of the Fleas Pseudorandom Functions the verbal description below may be hard to follow and readers familiar with the C++ syntax are encouraged to additionally look at the source code.

Initialization

When Academic Signature is called for the first time, it detects the absence of a stored random pool of name "randpool.rnpl" in the current "x-secrets" folder and enters in the initialization procedure. The user is prompted to give a reference to an ephemeral file which ideally is a volatile file just created e.g. a sound recording or a webcam snapshot just recorded, which should be deleted immediately after usage to prime the PRNG. The ephemeral file is hashed to a hash value of 2011 bytes length with a Hash function from the fleas family, Fleas_lb in the most recent version. This hash value is immediately enciphered with the users login key and stored on hard disk as failsafe recovery pool for the case the program did not exit properly. (The normal exit procedure will always overwrite the pool with an encrypted fresh version.)

Knowing that the sloppy user might easily replace that file by a permanent, conveniently located file, the user is additionally prompted for a line of arbitrary keystrokes. Additionally this serves to cryptographically separate the currently used pool from the stored failsafe pool. This string as well as the hash value will be fed to a less recent pseudorandom function of the fleas family not described here. The result is 2011 bytes long and is used as the primary random pool.

Storage of the Random Pool

On several occasions during Academic Signature execution, a copy of the PRNG pool is co-hashed again with unique and fresh information (usually system time) and the stored encrypted pool value is replaced by the encrypted version of this value to block the threat of pool reuse.

During the regular exit procedure of Academic Signature the pool is co-hashed with the current system time, enciphered with the current login key and stored on disk. In case the login key is changed during program execution, the stored pool is not recoded since future deciphering of the pool with a wrong key acts as a desirable randomization.

Recovery of the Random Pool

On start of Academic Signature the pool is read in, checked for consistency and deciphered. The pool is then copied, the copy is co-hashed with current system time and the encrypted result overwrites the past stored pool.

Immediately afterwards the pool just read is co-hashed with the pool just written to cryptographically separate current pool from the stored pool and all other past pools. The result of this operation is used as primary random pool in the current instance of Academic Signature.

This primary pool is now copied and the copy is fed to an independent pseudo random function "develop_ls" to produce a consumable pool of 2011 random bytes.

Usage of the Random Pool

Upon request of random bytes from the consumable pool it may occur that the consumable pool of initially 2011 bytes is used up. In this case the primary pool will be fed to a pseudo random function "develop_o", the result will serve as new primary random pool. The pools age parameter will be incremented. A new secondary, consumable pool will be derived by subsequently applying the pseudorandom function "develop_ls" to a copy of the new primary pool. This pool will then be used to satisfy remaining random byte requests.

In case the age parameter reaches a critical value, the user will be prompted to supply a new line of random keystrokes to rejuvenate the pool. Note that this will not happen in the lifetime of our universe with a pool of 2011 bytes. I introduced this limit for reasons of logical soundness. It may come in handy, however, if someone abuses the Fleas PRNG by selecting a ridiculously small pool size like in NSA influenced standards.