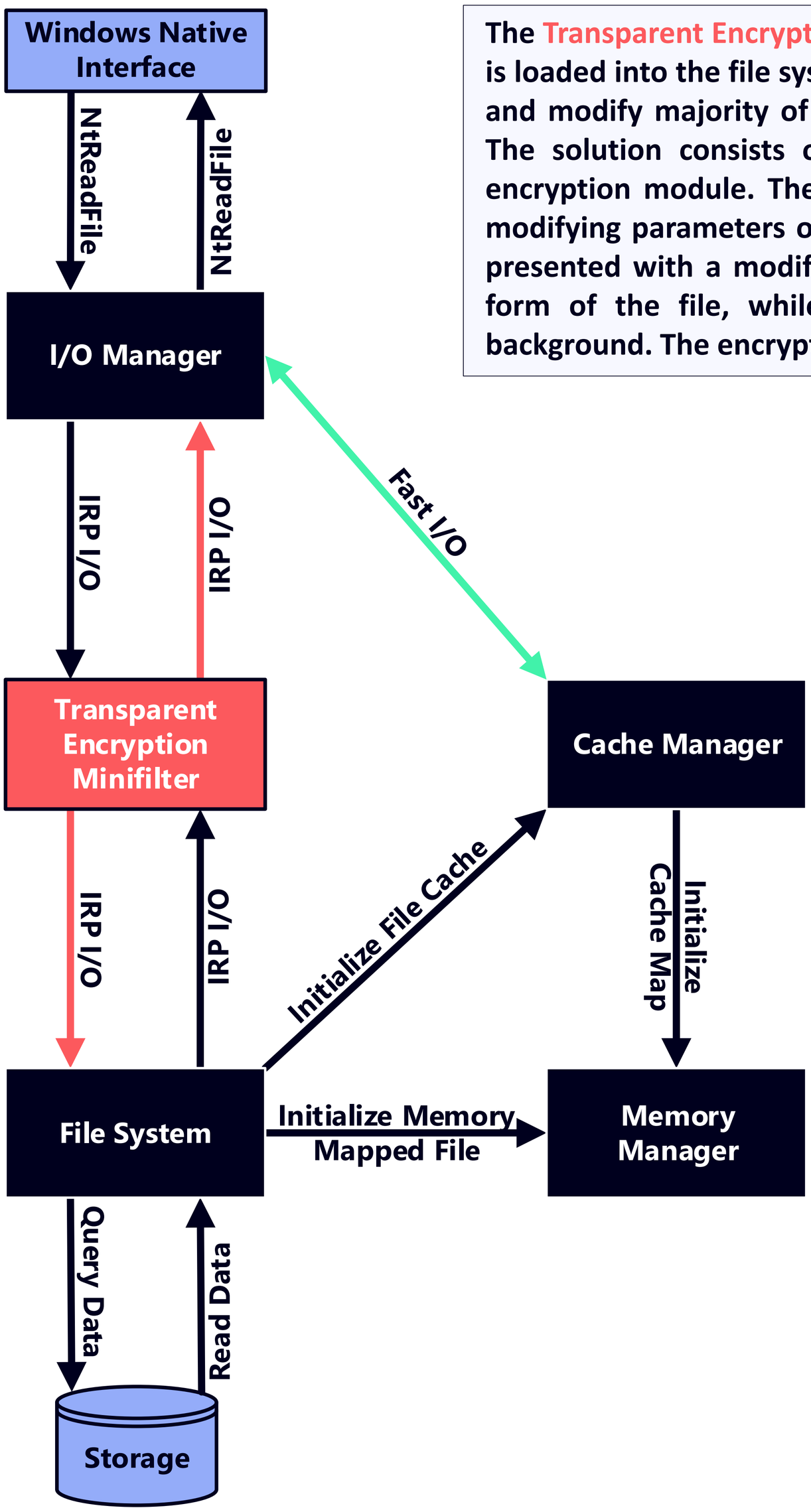
# TRANSPARENT ENCRYPTION
## WITH WINDOWS MINIFILTER DRIVER

#19

Author: Bc. David Pořízek

Supervisor: Doc. Dr. Ing. Dušan Kolář

The Transparent Encryption Minifilter is a Windows minifilter driver written in C. It is loaded into the file system driver stack as a filter driver. This allows it to monitor and modify majority of requests, which are issued towards files on the storage. The solution consists of two core parts – the transparency module and the encryption module. The transparency is ensured by the minifilter driver and by modifying parameters of relevant requests made towards a file. The user is then presented with a modified file view and is virtually working with the decrypted form of the file, while in fact, the file has already been encrypted in the background. The encryption module is described in the second part of this poster.




Fast I/O represents a case, where a file has been already accessed before and thus its cache has been initialized. In these cases, the read or write request does not reach the minifilter or the file system driver, but instead is satisfied immediately by Cache Manager (CM). Since the minifilter is not able to participate in completing the Fast I/O path, it has to preemptively modify requests to initialize the file's cache memory, to make sure the cache contains file's data in its decrypted form.

Input/Output Request Packets (IRPs) are data "blobs" which are used to exchange information between different drivers in the Windows system. There are multiple types of these IRPs, specifically, each type of a request has its own IRP type. File system uses primarily *IRP_MJ_READ* or *IRP_MJ_WRITE* to obtain or write data from or to a file. To first access the file and open its handle a *IRP_MJ_CREATE* must be issued. This also usually triggers initialization of the file's cache. The minifilter driver recognizes this and modifies the initialization parameters in order to append the encryption header to the stored file. The header's structure is described in the figure below.

The encryption module is implemented as a library. This means, that the encryption logic used can be replaced by a different one. By default, the solution uses AES-256 to encrypt the content and MD5 for the checksum. The Content Encryption Key is generated by the driver and uses a true randomness generator provided by the Bcrypt library.