Digital Investigation 12 (2015) 41–52

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

A mathematical approach to NAND flash-memory descrambling and decoding

Jan Peter van Zandwijk*

Netherlands Forensic Institute, Department of Digital Technology, Laan van Ypenburg 6, 2497 GB The Hague, The Netherlands

ARTICLE INFO

Article history:
Received 27 November 2014
Received in revised form 12 January 2015
Accepted 15 January 2015
Available online 6 February 2015

Keywords:
NAND flash memory
LFSR-based scrambling
Berlekamp–Massey algorithm
Error-correcting code
BCH-code

ABSTRACT

New mathematical techniques for analysis of raw dumps of NAND flash memory were developed. These techniques are aimed at detecting, by analysis of the raw NAND flash dump only, the use of LFSR-based scrambling and the use of a binary cyclic code for error-correction. If detected, parameter values for both LFSR and cyclic error-correcting code are determined simultaneously. These can subsequently be applied to expose the content of memory pages in the raw NAND flash dump and prepare these for further processing with media analysis tools. The techniques were tested on raw NAND flash memory dumps of four different devices and in all cases LFSR-based scrambling and binary cyclic error-correcting codes were in use.

Introduction

Over the years, NAND flash memory has become the dominating storage medium used in consumer electronics such as smartphones, USB-drives, solid state drives and SD cards. NAND flash is a form of non-volatile memory that can be electrically erased and reprogrammed. A typical layout of a device using NAND flash memory is schematically shown in Fig. 1.

The flash storage consists of the flash memory itself and a controller, which handles requests to read or write data and thus provides an interface between the flash memory and the host operating system. NAND flash memory is organized in blocks, each consisting of a number of memory pages. Reading and writing of data to NAND flash memory is done page wise, whilst erasing data from flash memory can only be done block-wise. Blocks typically contain 32-128 memory pages, which each typically consist of 4096–8192 bytes of data. Usually, memory pages in NAND flash are subdivided into a number of smaller data areas, henceforth referred to as 'data chunks', which are typically 1024–2048 bytes of size. Besides data chunks, memory pages contain an area reserved for storage of metadata for that page, generally referred to as the spare area. Information in the spare area is available to the controller only and cannot be accessed from the host. In handling read and write requests, the NAND flash controller performs several tasks aimed at ensuring data integrity, which will be described extensively below. Besides this, the controller executes a wear levelling algorithm in order to spread deterioration of blocks due to erasure evenly across the NAND flash memory, thereby expanding flash memory lifetime. Finally, the controller marks blocks as 'bad' or 'expired' when it has become impossible to erase them and translates also logical block addresses to physical block addresses, i.e. actual locations within the NAND flash memory.

It is well known that NAND flash memory, especially the ones containing multi-level cells (i.e. cells which hold more than one bit of information), is inherently susceptible to bit errors. This means that during normal operation, data

* Corresponding author. Netherlands Forensic Institute, Department of Digital Technology, PO Box 24044, 2490 AA The Hague, The Netherlands. Tel.: +31 70 8886435.
E-mail address: j.p.van.zandwijk@nfi.minvenj.nl.

http://dx.doi.org/10.1016/j.diin.2015.01.003
1742-2876/

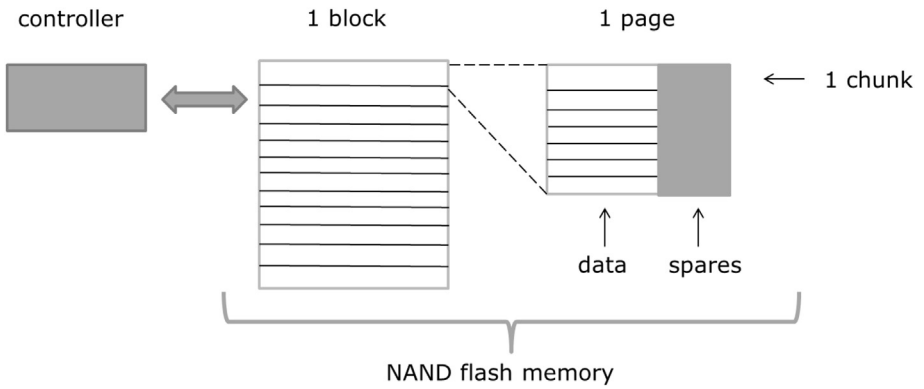

Fig. 1. Typical layout of a device containing NAND flash memory. Controller handles request from the host operating system to read from and write data to the NAND flash device. The NAND flash memory itself is divided into blocks. Each block contains a number of memory pages, each of which contains a number of data chunks. Metadata is stored in the spare area of data chunks.

stored in NAND flash memory can get changed inadvertently, leading to loss of data integrity. Manufacturer of NAND flash memories use encoding and data randomisation techniques to deal with bit errors and hence improve data reliability. In *encoding*, the controller applies an error-correcting code (ECC) to compute parity bits for data to be stored in NAND flash memory and stores these in the spare area of the memory page where the data is written. When the page is read back from the flash memory, the stored ECC parity bits are then used by the controller to verify data integrity. If necessary, they can be used by the controller to correct bit errors in the memory page, a process usually referred to as *decoding*. Due to cell-to-cell interference certain patterns in data can be more prone to generating bit errors when stored in NAND flash memory (see for example Cha and Kang, 2013). *Data randomisation* (often referred to as *data scrambing*) helps prevent this type of bit errors from occurring by reversibly converting the data to a random looking sequence before storage in NAND flash memory, thereby masking any pattern in the data that may give rise to bit errors. Data can be randomised by the controller through addition of a pseudo-random sequence, which indeed removes any pattern that might be present in the data. When data is read back from the flash memory, the same pseudo-random sequence is then subtracted again from the data by the controller before it is presented to the host operating system. This process will henceforth be referred to as *de-randomisation* or *descrambling*. It is important to realise that the purpose of data randomisation is to enhance data reliability and *not* to protect data against unauthorised access. For the latter purpose, it is customary to make use of good cryptographic primitives.

Fig. 2 schematically shows the process of data randomisation and encoding as done by the controller when storing data in NAND flash memory. The order in which both processes can be executed is interchangeable: the controller can either compute parity bits first and then randomise data, or first randomise data and then compute parity bits for this randomised data. After these processes, randomised data and parity bits are written to NAND flash memory. In doing so, some slight reformatting of data might occur. For instance, randomised data can be stored in reversed byte order, or bits in the data might be inverted before storage. Due to the reformatting process, randomised data and parity bits as found in NAND flash memory might not correspond directly to data coming from the encoding and randomisation processes, as described above. When data is read from NAND flash memory, the reverse processes are executed by the controller: data from flash memory is de-randomised and decoded, spares are removed and the resulting data is presented to the host operating system.

In a forensic investigation, data from a NAND flash device can be secured by making a high-level copy of the file system on the device using forensic imaging tools such as EnCase, dd or FTK Imager. In doing so, data present on the NAND flash device is accessed through the controller. In such a case, the processes of decoding and de-randomisation data remain invisible because these are dealt with by the controller. Also, the controller takes care of presenting the content of memory pages in the right order to the host. As an alternative to high-level data acquisition, it is possible to acquire data from a NAND flash device at a lower level by making a raw dump of the content of the NAND flash memory itself. This is technically more involved than imaging the high-level file system and can require some special expertise. An overview of techniques for low-level data acquisition of NAND flash memory can be found in Breeuwsma et al. (2007). Low-level data acquisition may be the only possibility to secure data from a NAND flash device when it is impossible to use the controller to access data on the device, as may be the case when the controller is damaged or when access to the device is protected by an unknown password. An advantage of low-level data acquisition with respect to high-level acquisition is that also information which is hidden by the controller such as for example memory pages marked as bad and spare areas of memory pages, becomes available to investigation. Data obtained in a low-level acquisition is unsuitable for direct further processing with media analysis tools because it contains randomisation, needs to be decoded and memory pages need to be put in the right order to reconstruct the high-level file system.