플래시메모리 파일시스템을 위한 안전한 파일 삭제 기법

(Secure Deletion for Flash Memory File System)

선 경 문 † 최 종 무 ††
(Kyoungmoon Sun) (Jongmoo Choi)
이 동 희 ††† 노 삼 혁 ††††
(Donghee Lee) (Sam H. Noh)

요 약 휴대전화, MP3플레이어 및 PMP, USB 메모리 저장장치와 같은 개인용 멀티미디어 및 저장용 이동기기의 사용이 보편화되면서 이동기기에 저장되는 데이타에 대한 안전성이 요구되고 있다. 요구되는 안전성 중 한 가지는 안전한 파일 삭제인데, 이것은 파일의 내용이 완전히 삭제되어 악의적으로 복구될 수 없도록 하는 것이다. 본 논문에서는 이동기기의 저장매체로써 주로 사용되는 플래시 메모리에서 어떻게 안전한 삭제를 할 수 있는지에 대하여 연구한다. 이를 위하여 0으로 덮어쓰기와 가비지 컬렉션을 이용하는 두 가지 안전한 파일 삭제 정책을 고려하였으며, 각 정책들이 플래시 메모리 파일 사스템의 성능에 미치는 영향을 분석하였다. 또한 두 가지 정책들의 장점을 취한 적응적인 파일 삭제 기법을 제안한다. 구체적으로 크기가 작은 파일들에 대해서는 0으로 덮어쓰기 기법을, 크기가 큰 파일들에 대해서는 가비지 컬렉션기법을 적용하였다. 그리고 실제 실험 구현 및 결과를 통해 제안된 기법들이 안전하고 효율적으로 파일을 삭제할 수 있음을 보인다.

키워드 : 개인정보, 플래시메모리, 안전한 파일 삭제

Abstract Personal mobile devices equipped with non-volatile storage such as MP3 player, PMP, cellular phone, and USB memory require safety for the stored data on the devices. One of the safety requirements is secure deletion, which is removing stored data completely so that the data can not be restored illegally. In this paper, we study how to design the secure deletion on Flash memory, commonly used as storage media for mobile devices. We consider two possible secure deletion policy, named zero-overwrite and garbage-collection respectively, and analyze how each policy affects the performance of Flash memory file systems. Then, we propose an adaptive file deletion scheme that exploits the merits of the two possible policies. Specifically, the proposed scheme applies the zero-overwrite policy for small files, whereas it employs the garbage-collection policy for large files. Real implementation experiments show that the scheme is not only secure but also efficient.

Key words : Privacy, Flash Memory, Secure Deletion

1. 서 론

내장형 시스템이나 휴대용 저장 매체에서 플래시 메모리는 저전력과 강한 내구성, 빠른 속도, 용량의 증가, 저렴한 가격 등으로 인하여 하드디스크의 대체 수단으로 각광받아왔다. 플래시 메모리가 대중화되면서 USB플래시 메모리 장치와 같이 개인적인 용도의 저장 장치로의 사용이 증가하고 있으며 이에 따라 플래시 메모리 기반의 파일시스템에 대한 보안 안전성이 요구되고 있다.

플래시 메모리 파일 시스템에서는 한번 기록되었던 파일은 삭제하여도 실제 데이타는 삭제하지 않은 채 메타데이타에 삭제 표시만 하는 경우가 많은데, 이 경우 추후 삭제된 파일의 내용을 복원할 수 있게 되는 문제가 가지고 있다. 예를 들어 개인 및 기업간 전자상거래를 위한 공인 인증서의 경우 절대 외부에 노출되어서는 안될 중요한 파일임에도 불구하고, 한번 플래시 메모리 파일시스템에 기록되게 되면 파일을 삭제하였더라도 가비지 컬렉션(Garbage Collection)이 수행되어 해당 파일의 내용이 전부 소거(Erase)되기 전까지 파일의 내용이 플래시 메모리에 남아있게 된다. 이 경우 악의적인 목적을 가진 자에게 플래시 메모리 장치가 누출되게 되면 중요한 정보의 유출 문제가 발생할 수 있다.

디스크 기반의 파일 시스템에서는 이 문제를 해결하기 위하여 파일 시스템에 의해 할당되었다가 해제된 디스크 블록들의 내용을 삭제하는 기법들이 연구되어 있으나[1-6] 플래시 메모리 기반 파일 시스템에서의 연구

· 이 논문은 2007 한국컴퓨터종합학술대회에서 '플래시메모리 파일시스템을 위한 안전한 파일 삭제 기법'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 단국대학교 정보컴퓨터학부
msg2me@dankook.ac.kr
†† 종신회원 : 단국대학교 정보컴퓨터학부 교수
choijm@dankook.ac.kr
††† 정 회 원 : 서울시립대학교 컴퓨터과학부 교수
dhlee@venus.uos.ac.kr
†††† 종신회원 : 홍익대학교 컴퓨터공학과 교수
samhnoh@hongik.ac.kr

논문접수 : 2007년 10월 2일
심사완료 : 2007년 11월 2일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제13권 제6호(2007.11)

는 아직 미흡한 상태이다. 본 논문에서는 플래시 메모리의 특성, 즉 일반적으로 덮어쓰기는 불가능하며, 데이타의 삭제를 위해서는 소거연산이 필요하다는 특성을 고려한 안전한 파일 삭제를 연구한다. 구체적으로 본 논문에서는 대용량 데이타 저장 매체로써 많이 사용되는 NAND 플래시 메모리 기반에서 YAFFS[7]라는 NAND 플래시 메모리용 파일 시스템을 대상으로 연구를 진행하였다. YAFFS는 비교적 구조가 잘 알려져 있으며 많은 플래시 메모리 파일 시스템에서 채택하는 로그구조 파일 시스템(Log-Structured File System)[8]을 따르고 있다. 이를 대상으로 플래시 메모리에 저장 되었다 삭제된 파일을 복원하는 방법을 보이며, 디스크와는 다른 플래시 메모리만의 특성을 이용하여 효율적이며 안전하게 데이타를 삭제할 수 있는 기법을 제안한다.

2. NAND 플래시 메모리의 구조와 연산

NAND 플래시 메모리는 하나의 비트 값을 저장하는 셀들이 모여 구성되어 있으며 플래시 메모리 용량(Bit 수)만큼의 셀들이 존재하게 된다. 이 셀들이 모여 기본 입출력 단위인 페이지를 이루며, 여러 개의 페이지들이 모여 블록을 이룬다. 그리고 페이지는 주데이타를 저장할 수 있는 데이타 영역과, 부가적인 관리정보를 저장할 수 있는 스페어(Spare) 영역이 존재한다.

NAND 플래시 메모리에서 기본적으로 제공하는 연산으로 페이지 읽기(Page Read), 페이지 쓰기(Page Program), 블록 소거(Block Erase)가 있다. 블록 소거 연산은 블록 내에 모든 셀들의 값을 1로 바꾸게 되며, 플래시 메모리 연산 중 가장 오랜 시간이 소모된다. 셀의 값을 1에서 0으로 바꾸는 연산을 쓰기(Program)라 한다. 쓰기 연산은 페이지 단위로 이루어지게 되며, 한번 페이지가 쓰여지게 되면, 셀의 값이 0에서 1로 다시 바뀔 수 없으므로 대부분의 경우 블록 소거가 이루어져야 다시 사용할 수 있게 된다.

플래시 메모리에 공간을 모두 사용하여 더 이상 사용 가능한 공간이 없을 때에는 불필요한 블록들을 소거하여 페이지를 재생시키는데, 아직 내용이 유효하여 보존되어야 하는 페이지는 다른 블록의 페이지에 복사하게 되며 이러한 작업을 가비지 컬렉션이라 한다.

3. 플래시 메모리 파일 시스템에서의 파일 복원

플래시 메모리에서 이미 쓰여졌던 페이지를 다시 사용하기 위해서는 가비지 컬렉션을 수행해야 하는데 이는 상대적으로 높은 비용을 초래한다. 따라서 많은 플래시 메모리 파일 시스템에서는 가비지 컬렉션 수행 시점을 가능한 나중으로 늦추려 한다. 구체적으로 YAFFS와 같은 많은 플래시 메모리 파일 시스템에서 파일의

```
[root@ez-x5 /root]$ insmod /mnt/nfs/yaffs.o
Using /mnt/nfs/yaffs.o
[root@ez-x5 /root]$ mount -t yaffs /dev/mtdblock2 nand
yaffs: dev is 7938 name is "1f:02"
[root@ez-x5 /root]$ cd nand
[root@ez-x5 nand]$ ls -al
drw-rw-rw-    1 root     root          512 Jan  1 00:06 .
drwxr-x---    3 root     root         1024 Jan  1 00:00 ..
drw-rw-rw-    1 root     root          512 Jan  1 00:06 lost+found
[root@ez-x5 nand]$ cp /mnt/nfs/docs/* ./
[root@ez-x5 nand]$ ls -al
drw-rw-rw-    1 root     root          512 Jan  1 00:06 .
drwxr-x---    3 root     root         1024 Jan  1 00:00 ..
drw-rw-rw-    1 root     root          512 Jan  1 00:06 lost+found
-rw-r--r--    1 root     root       250368 Jan  1 00:06 target.doc
-rw-r--r--    1 root     root      1166357 Jan  1 00:06 target.pdf
[root@ez-x5 nand]$ rm target.doc target.pdf
[root@ez-x5 nand]$ ls -al
drw-rw-rw-    1 root     root          512 Jan  1 00:06 .
drwxr-x---    3 root     root         1024 Jan  1 00:00 ..
drw-rw-rw-    1 root     root          512 Jan  1 00:06 lost+found
[root@ez-x5 nand]$ /mnt/nfs/undelete/undelete /dev/mtd2 target.doc target.pdf
YAFFS File system undelete utility

total size : 54 mega bytes
erase size : 16 killo bytes, write size : 512 bytes, oob size : 16

target.pdf(object id : 261, size : 1166357 bytes -> 1166357 bytes recovered )
target.doc(object id : 260, size : 250368 bytes -> 250368 bytes recovered )
[root@ez-x5 nand]$ ls -al
drw-rw-rw-    1 root     root          512 Jan  1 00:06 .
drwxr-x---    3 root     root         1024 Jan  1 00:00 ..
drw-rw-rw-    1 root     root          512 Jan  1 00:06 lost+found
-rwx------    1 root     root       250368 Jan  1 00:07 target.doc
-rwx------    1 root     root      1166357 Jan  1 00:06 target.pdf
[root@ez-x5 nand]$
```

그림 1 YAFFS에서 삭제된 파일의 복원

삭제 방법은 해당 파일이 점유하고 있는 페이지들의 스페어 영역에 유효하지 않음(invalid) 표시를 해두고 inode를 삭제함으로써 완료된다. 때문에 삭제된 파일의 내용이 오랫동안 플래시 메모리에 남아있게 되며 이를 토대로 복원이 가능하다.

이것을 직접 보이기 위해서 YAFFS Undelete라는 YAFFS파일 시스템용 삭제된 파일 복원 프로그램을 작성 하였다. YAFFS Undelete는 YAFFS에서 inode역할을 하는 Object Header 페이지를 검색한다. 만약 삭제된 파일로 확인 된 Object Header 페이지가 있다면 해당 파일의 고유 ID인 ObjectID와 같은 ObjectID값을 갖고 있는 데이타 페이지들을 검색하여 파일을 복원하게 된다. 그림 1에서 YAFFS Undelete를 이용하여 삭제된 target.doc와 target.pdf라는 두 개의 문서 파일들을 복원하는 것을 보였다.

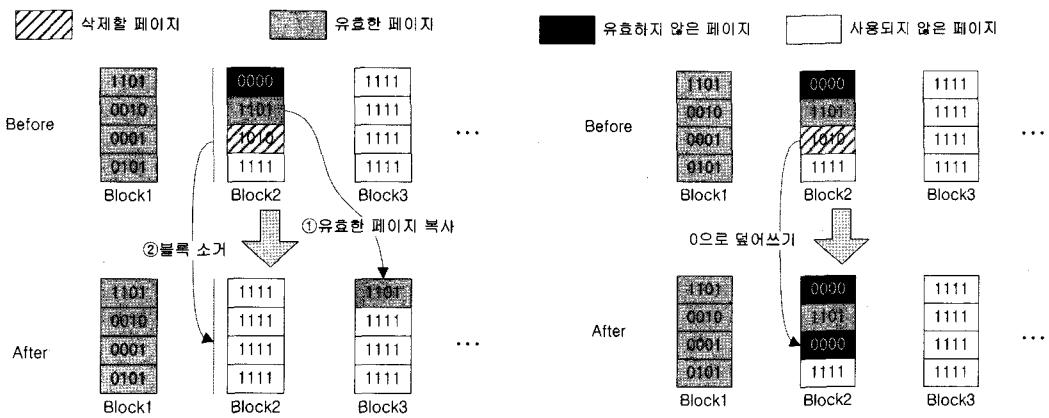
4. 플래시 메모리에서의 안전한 삭제 기법

4.1 가비지 컬렉션에 의한 안전한 삭제

가비지 컬렉션은 비용이 높은 작업이기 때문에 정책에 따라 플래시 메모리에 사용 가능한 공간이 부족할 때 혹은 일정 주기와 같이 전체 성능에 미치는 영향이 가장 적은 시점에 수행되는 것이 일반적이나, 안전한 삭제를 위해서는 파일을 삭제하는 시점에 해당 파일의 내용이 포함된 모든 블록들에 강제로 수행하도록 하여 파일의 내용을 삭제할 수 있다. 블록이 소거되면 블록 내 모든 셀들의 값이 1로 바뀌게 되므로 데이타를 안전하게 삭제할 수 있게 된다(그림 2의 (a) 참조).

4.2 0으로 덮어쓰기에 의한 안전한 삭제

일반적으로 플래시 메모리는 덮어쓰기가 불가능하다. 하지만 실제로는 0에서 1로 비트 변경이 불가능하며, 1



(a) 가비지 컬렉션에 의한 안전한 삭제 (b) 0으로 덮어쓰기에 의한 안전한 삭제

그림 2 NAND플래시 메모리에서 페이지의 내용을 삭제하는 방법

에서 0으로의 비트 변경은 페이지 당 2~3회 정도가 가능하다[9]. 이 기능을 이용하여 삭제하려는 파일과 관련된 모든 페이지들의 데이타 영역과 스페어 영역을 0으로 덮어쓰기 하여 데이타를 소멸시킬 수 있다. 블록 소거는 해당 블록내의 모든 셀 값이 1로 바뀌는데 반해 0으로 덮어쓰기는 해당 페이지 내의 모든 셀 값이 0으로 바뀌게 된다(그림 2의 (b) 참조). 일반적으로 쓰기연산의 시간이 블록연산의 시간보다 작다는 점을 감안하면 블록 내에서 파일의 내용을 포함하는 페이지의 수가 적을 때에 수행하는 것이 유리하다.

4.3 오버헤드 분석 및 최소 오버헤드 선택 방법

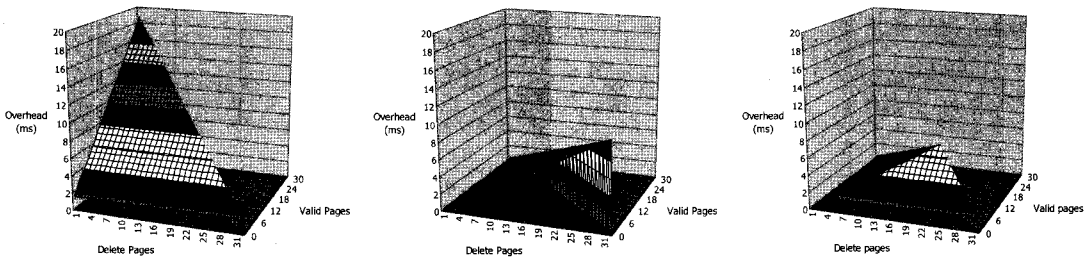
두 기법을 이용하여 파일을 삭제할 때 삭제할 파일의 내용을 포함하고 있는 각 블록에 대하여 비용과 이익, 계산된 오버헤드를 수식으로 나타내면 다음과 같다.

* 표기 R_t: 페이지 읽기 시간 W_t: 페이지 쓰기 시간 E_t: 블록 소거 시간 # *of Delete Pages*: 삭제할 페이지 수 # *of Valid Pages*: 삭제할 블록에 포함된 유효 페이지 수 # *of Invalid Pages*: 삭제할 블록에 포함된 비 유효 페이지 수 # *of Pages Per Block*: 블록 당 페이지 수 # *of New Available Pages*: 새로 얻게 될 사용 가능한 페이지 수 = # *of Delete Pages* + # *of Invalid Pages*
* 0으로 덮어쓰기에 의한 안전한 삭제 $Cost = \#\ of\ Delete\ Pages \times W_t$ *Overhead* = *Cost*
* 가비지 컬렉션에 의한 안전한 삭제 $Cost = \#\ of\ Valid\ Pages \times (R_t + W_t) + E_t$ $Benefit = (E_t / \#\ of\ Pages\ Per\ Block) \times \#\ of\ New\ Available\ Pages$ *Overhead* = *Cost* - *Benefit*

0으로 덮어쓰기에 의한 안전한 삭제 비용은 삭제해야 할 페이지수만큼 쓰기 비용이 발생하므로 $\#\ of\ Delete\ Pages \times W_t$ 만큼의 비용이 발생하였으며, 가비지 컬렉션에 의한 안전한 삭제는 삭제 할 블록 내 유효한 페이지를 복사하고 블록을 소거하는 비용을 포함하므로 $\#\ of\ Valid\ Pages \times (R_t + W_t) + E_t$ 만큼의 비용이 발생한다. 그런데 가비지 컬렉션을 수행하게 되면 내용을 삭제하는 것 이외에 재생되어 사용 가능한 페이지를 얻게 된다. 따라서 플래시 메모리를 사용 도중 유효한 페이지가 부족하여 가비지 컬렉션이 수행되는 빈도를 줄이게 되므로 이것을 이익으로 환산하였다. 따라서 가비지 컬렉션에 의한 안전한 삭제기법은 비용에서 이익을 뺀 만큼의 오버헤드를 갖게 된다. 이익의 계산 방법은 블록을 소거하였을 경우 블록 당 페이지 수만큼의 재생되어 사용 가능한 페이지를 얻는다는 가정 아래에, 새로 얻게 되는 사용 가능한 페이지의 수가 적을수록 0에, 커질수록 블록 소거 비용에 가까워지도록 하였다.

본 연구에서는 두 기법의 오버헤드를 기반으로 최소 오버헤드 선택 방법을 제안한다. 최소 오버헤드 선택 방법은 파일 삭제 요청 시 파일의 내용을 포함하고 있는 모든 블록들을 리스트에 담아두었다가 각 블록 별 0으로 덮어쓰기에 의한 안전한 삭제 기법과 가비지 컬렉션에 의한 안전한 삭제 기법의 오버헤드를 계산하여 오버헤드가 더 작은 기법을 수행한다.

이 기법은 결국 블록 내에 삭제 해야 할 파일의 내용이 많이 포함되어 있거나 유효하지 않은 페이지가 이미 많이 포함되어있는 경우에는 가비지 컬렉션을 수행하고 삭제할 파일의 내용이 포함된 정도가 낮거나 유효한 페이지들이 많이 포함되어 있는 경우에는 0으로 덮어쓰기



(a) 가비지 컬렉션에 의한 안전한 삭제 (b) 0으로 덮어쓰기에 의한 안전한 삭제 (c) 최소 오버헤드 선택 방법

그림 3 삭제 할 페이지와 유효한 페이지의 수에 따른 삭제 기법 별 오버헤드

에 의한 안전한 삭제 기법을 수행하여 성능을 향상시키려 하였다.

그림 3은 삭제할 페이지 수와 블록에 포함된 유효한 페이지 수의 관계에 따른 오버헤드 모델을 나타내고 있다. 플래시 메모리의 각 연산 시 소모시간은 4절의 실제 실험 환경에서 측정된 시간(페이지 읽기: 0.225ms, 페이지 쓰기: 0.323ms, 블록 소거: 1.71ms)을 대입하였다.

그림 3의 (a) 가비지 컬렉션에 의한 안전한 삭제 기법에서는 블록에 포함된 유효한 페이지들이 많을수록 오버헤드가 급격히 상승하는 모습을 보이고 있으며, 그림 3의 (b) 0으로 덮어쓰기에 의한 안전한 삭제 기법에서는 삭제할 페이지의 수에 따라서만 오버헤드가 변화한다. 마지막으로 그림 3의 (c) 최소 오버헤드 선택 방법은 앞서 두 기법의 오버헤드 모델 중 더 작은 오버헤드 영역만 취함을 보이고 있다.

5. 실험 및 성능 측정 결과

제안된 세가지 방법을 이용하여 실제 파일을 삭제할 때의 소모시간을 측정하였다. 실험은 EZ-X5 테스트 보드[10]에서 수행하였는데, 이 시스템은 Intel PXA255 CPU와 SAMSUNG 64MB DRAM, SAMGSUNG 64MB NAND 플래시 메모리가 장착되어 있고 OS는 Linux Kernel 2.4.19를 사용하였다. 실험 방법은 1KB에서 512KB까지, 파일의 크기를 두 배씩 늘려가며 각 크기 별 50개의 파일들을 생성 후 일괄 삭제할 때 평균 삭제 시간을 측정하였다.

단 YAFFS에서의 가비지 컬렉션은 비효율적으로 작성된 부분이 존재한다. YAFFS에서는 가비지 컬렉션을 수행할 블록에 포함된 페이지들의 유효 여부를 알지 못하여 해당 블록의 모든 페이지를 읽어 들인 후 유효 여부를 확인하는데 이는 불필요한 페이지 읽기 비용을 초래하므로, 페이지 상태를 메모리에 저장하도록 하여 플래시 메모리로부터 페이지를 읽어 들이지 않고도 페이지의 유효 여부를 확인할 수 있도록 수정하였다.

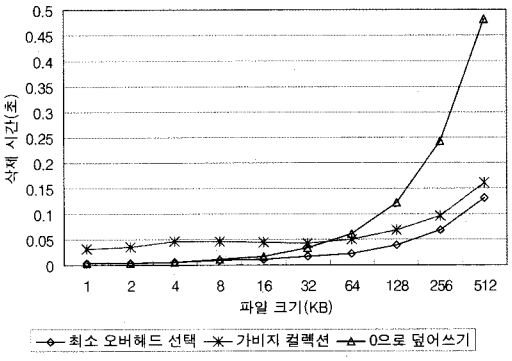


그림 4 플래시 메모리 파일 삭제 기법 별 소모시간

그림 4에서 0으로 덮어쓰기에 의한 안전한 삭제 기법은 파일의 크기에 비례하여 파일의 크기가 작을 때에는 좋은 성능을 보였으나 파일의 크기가 커지면 커질수록 비효율적으로 나타났다. 가비지 컬렉션에 의한 안전한 삭제 기법은 파일의 크기가 커져도 삭제 시간이 비교적 완만한 곡선의 모습 보여주고 있으며 파일의 크기가 클수록 0으로 덮어쓰기에 의한 안전한 삭제 기법보다 뛰어난 성능을 보이지만, 블록 소거의 비용과 유효한 페이지의 복사 비용 등으로 인하여 크기가 작은 파일이라 하여도 일정 이상의 시간을 소모하는 것으로 나타났다. 마지막으로 제안된 최소 오버헤드 선택 방법은 파일의 크기가 작을 때와 클 때 모두 최소의 비용으로 안전하게 파일을 삭제할 수 있었으며, 위 실험을 진행하는 동안 평균적으로 가비지 컬렉션에 의한 안전한 삭제보다는 약 4.4배, 0으로 덮어쓰기에 의한 안전한 삭제보다 약 2.1배 더 빠른 성능을 보였다.

6. 결 론

본 논문에서는 NAND플래시 메모리 기반의 파일 시스템에서 NAND 플래시 메모리의 특성을 이용하여 최소의 비용으로 안전하게 파일을 삭제하여 악의적인 목적의 삭제된 파일복원을 불가능하게 하는 기법을 제안

하였다. 현재 NAND플래시 메모리용 파일시스템인 YAFFS에 제안된 기법을 적용하여 연구하였으며 향후에는 FTL(Flash Translation Layer)에 제안된 기법을 적용하여 대용량 USB메모리 저장 장치와 같은 환경을 목표로 연구를 진행할 예정이며 특성이 조금 다른 MLC(Multi Level Cell) NAND 플래시 메모리에 대해서도 연구를 계획 중이다.

참 고 문 헌

[1] Gopalan Sivathanu, Swaminathan Sundararaman, and Erez Zadok. Type-Safe Disks. In *Proc. of the 7th Usenix Symposium on Operating Systems Design and Implementation*. pp. 15-18. November 2006.

[2] S. Bauer and N. B. Priyantha. Secure Data Deletion for Linux File System. In *Proc. Of the 10th Usenix Security Symposium*, pp. 153-164, Washington, DC, August 2001.

[3] P. Gutmann. Secure Deletion of Data from Magnetic and Solid-State Memory. In *Proc. Of the sixth Usenix UNIX Security Symposium*, pp. 77-90, San Jose, CA, July 1996.

[4] N. Joukov and E. Zadok. Adding Secure Deletion To Your Favorite File System. In *Proc. Of the third international IEEE Security In Storage Workshop*, San Francisco, CA, December 2005.

[5] R. Perlman. Secure Deletion of Data. In *Proc. Of the third international IEEE Security In Storage Workshop*, San Fransisco, CA, December 2005.

[6] M. Sivathanu, L. N. Bairavasundaram, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Life or Death at Block-Level. In *Proc. Of the 6th Symposium on Opearting Systems Design and Implementation*, pp.379-394, San Francisco, CA, December 2004.

[7] Yet Another Flash File System. http://www.aleph1.co.uk/yaffs/

[8] M. Rosenblum and J. Ousterhout. The design and implementation of a log-structured file system. In *Proc. of the 13th Symposium on Operating System Principles*, pp. 1-15, October 1991.

[9] Samsung Electronics. APPLICATION NOTE for NAND Flash Memory. http://www.samsung.com/Products/Semiconductor/Memory/appnote/app_nand.pdf

[10] FA LINUX. EZ-X5 BOARD http://www.falinux.com