KIPS Tr. Comp. and Comm. Sys.
Vol.6, No.1 pp.31~42 pISSN: 2287-5891

https://doi.org/10.3745/KTCCS.2017.6.1.31

# Study for Android Smartphone's Gallery Thumbnail Forensic Analysis

Daeho Yun† · Sang Jin Lee††

## ABSTRACT

Thumbnail, the small sized graphic file such as JPEG or GIF, serves to help the users to be recognized as a rapidly helps to make it easier recognize while browsing the large sized graphic file. Gallery application, which is installed in a later version of the 4.4.x(Kitkat) Android smartphone records the generated time of graphic file in thumbnail metadata. Thumbnail can be used to draw up the timeline of user action about user's action such as creation, modification, deletion with original graphic file analysis. Also, take advantage of the features thumbnails are stored sequentially in a single thumbcache file, even if the thumbcache is deleted, we can restore the thumbnails. This paper illustrates the feature of thumbnail created by Android OS basic gallery app and methods for utilization in digital forensics.

Keywords : Thumbnail, Graphic File, Smartphone, Mobile Forensic

# 안드로이드 스마트폰의 갤러리 썸네일(Thumbnail)에 대한 포렌식 분석 방법에 관한 연구

윤 대 호† · 이 상 진††

## 요 약

썸네일(thumbnail)은 JPEG, GIF 등 그래픽 파일의 축소판이며 그래픽 파일을 탐색하면서 알아보기 쉽도록 만들어주어 대용량 그래픽 파일이 빠른 속도로 사용자에게 인식될 수 있도록 도와주는 역할을 수행한다. 국내 안드로이드 스마트폰의 4.4.x(Kitkat) 이후 버전에서 설치되는 갤러리 애플리케이션은 썸네일의 metadata에 그래픽 파일의 생성시각을 기록한다. 이것을 기존 원본 그래픽 파일과 함께 병행하여 분석하면 생성, 수정, 삭제 등의 사용자 행위에 대한 타임라인을 구성할 수 있다. 또한 썸네일이 썸캐시라는 단일 파일 안에 순차적으로 저장되는 특징을 활용하면 썸캐시가 삭제된다고 하더라도 썸네일을 복원할 수 있다. 본 논문에서는 안드로이드 스마트폰의 기본 갤러리 애플리케이션이 생성하는 썸네일의 특성을 분석하고 삭제된 썸네일을 복원하는 방법과 디지털 포렌식 관점에서 활용할 수 있는 방안을 제시한다.

키워드 : 썸네일, 그래픽 파일, 스마트폰, 모바일 포렌식

## 1. 서 론

범죄 수사에 있어서 그래픽 파일의 분석은 매우 중요한 역할을 수행한다. 도촬이나 아동포르노과 같은 범죄는 그래픽 파일의 존재 유무가 범죄 혐의 입증에 필수적인 요소로서 작용한다[1].

범죄혐의의 입증에 필요한 주요 증거는 범죄자에 의해 삭제되는 경우가 많아 삭제된 데이터를 복구하는 것이 가장 중요한데 안드로이드에서 사용자의 데이터를 저장하는 파티션은 Ext4 파일시스템을 사용하고 있어 삭제된 파일을 복구하는 것이 제한된다[2]. 이 경우 그래픽 파일 포맷의 특성에 따라 카빙해야 하나 Exif 정보가 남아있는 그래픽 파일이 아닌 경우 그래픽 파일의 메타데이터를 추출할 수 없다는 한계점이 있다.

한편, 각종 OS에서는 대용량 그래픽 파일을 사용자가 신속하게 인식할 수 있도록 썸네일을 생성하는데, 원본 그래픽 파일에 비해 작은 용량으로도 원본 그래픽 파일의 형태를 인식할 수 있게 해 주어 범죄 수사에 많은 도움이 되어 왔다[3].

그 중에서도 안드로이드 OS 4.4.x(Kitkat) 버전 이후부터 갤러리 애플리케이션이 생성하는 썸네일에는 사용자가 스마트폰에 원본 그래픽 파일을 최초 생성한 시각이 기록되며, 이것을 원본 그래픽 파일과 교차 분석할 경우 원본 그래픽 파일에 대한 사용자 행위정보를 분석할 수 있다. 또한 그

† 준 회 원 : 고려대학교 정보보호대학원 정보보호학과 석사과정
†† 종신회원 : 고려대학교 정보보호대학원 교수
Manuscript Received : August 8, 2016
Accepted : September 29, 2016
* Corresponding Author : Sang Jin Lee(sangjin@korea.ac.kr)

Table 1. Path and Content of Thumbcache

| Model | | Android Version | Path | Content |
|---|---|---|---|---|
| SAMSUNG | SM-G930 (Galaxy S7) | 6.0.1 | /sdcard/emulated/Android/data/com.sec.android.gallery3d/cache/ | · imgcache<br>· micro<br>· mini<br>· nano |
| | SM-N920 (Galaxy Note5) | 6.0.1 | | |
| | SHV-E330 (Galaxy S4) | 5.0.1 | | · imgcache<br>· imgcacheMicro<br>· imgcacheMini |
| | | 4.4.2 | | |
| | SM-N900 (Galaxy Note3) | 4.4.2 | | |
| LG | LG-F460 (LG G3) | 6.0.1 | /sdcard/emulated/Android/data/com.android.gallery3d/cache/ | · imgcache<br>· imgcache_screen<br>· smallimgcache |
| | LG-F320 (LG G2) | 5.0.1 | | · imgcache |
| | | 4.4.2 | | |
| | LG-F240 (LG GPro) | 4.4.2 | | |

외에 다른 애플리케이션이 그래픽 파일에 대해 생성하는 썸네일 역시 그 특성을 분석함에 따라 특정 그래픽 파일이 사용자의 스마트폰에 저장되어 있었음을 알 수 있는 경우가 있다. 따라서 썸네일을 분석하는데 있어서 그래픽 파일에 대한 뷰어 및 편집 기능을 제공하는 갤러리 애플리케이션에 대한 분석이 매우 중요하다.

본 논문에서는 국내에서 사용되는 주요 안드로이드 스마트폰에 기본적으로 설치된 갤러리 애플리케이션이 생성하는 썸네일의 특성을 분석하고 디지털 포렌식 측면에서의 활용 방안을 제시한다.

## 2. 관련 연구

현재까지 PC용 운영체제에서 생성되는 썸네일에 대한 연구는 운영체제의 버전별로 계속 진행되어 왔다. 반면 안드로이드 스마트폰에서 생성되는 썸네일에 대한 연구는 Ming Di Leom의 논문[3] 외에는 없으며 Igor Mikhaylov와 Oleg Skulkin이 자신의 포렌식 관련 블로그[4]에 4.4.x (Kitkat) 버전의 썸캐시에 대해 언급하고 있다.

그러나 Ming Di Leom의 논문[3]에서는 삼성 넥서스 S (GT-I9020T) 1개 기종 및 4.1.2 (Jelly Bean) 버전의 갤러리 애플리케이션이 생성하는 썸네일만을 다루고 있으며 원본 그래픽 파일의 축소판인 썸네일의 단편적인 특징만을 다루고 있다. 또한 삭제된 썸네일을 복구하는데 있어서 썸네일의 특징을 고려하지 않았다. 또한 Igor Mikihaylov 등의 블로그[4]에서는 추가된 썸캐시가 있으니 모바일 포렌식간 함께 분석해야 한다는 정도만 기술되어 있으며 그 특징에 대하여 다루지 않고 있다.

따라서 보다 많은 기종의 안드로이드 스마트폰과 4.1.2 (Jelly Bean) 버전 이후 출시된 안드로이드 OS의 갤러리 애플리케이션이 생성하는 썸네일의 특징에 대해 새롭게 분석하고, 분석된 특징에 고려하여 삭제된 썸네일을 복원하는 방법과 디지털 포렌식 관점에서 활용 방법에 대한 연구가 필요하다.

## 3. 안드로이드 스마트폰 갤러리 썸네일 및 썸캐시 (Thumbcache)의 종류 및 특성

안드로이드 스마트폰의 기본 갤러리 애플리케이션이 생성하는 썸네일은 파일 단위로 저장되는 다른 애플리케이션의 썸네일과는 달리 썸캐시(thumbcache)라는 특정 파일 안에 모아서 연속적으로 저장되어 있다는 특징이 있다. 따라서 썸네일의 특성을 분석하는데 있어서 썸캐시의 구조 및 특성도 함께 분석해야 한다.

### 3.1 썸캐시의 구성

안드로이드 스마트폰에서 생성되는 썸캐시는 제조사 및 안드로이드 버전별로 파일명을 다른 방식으로 생성하지만 기본적인 구조는 썸캐시의 종류별로 [content].0, [content].1, [content].idx 3개의 파일로 구성되어 있으며 제조사별, 버전별 썸캐시의 위치 및 구성은 Table 1과 같다. .0 및 .1 파일은 썸캐시로서 썸네일을 사용자가 갤러리 애플리케이션을 확인하는 순서대로 저장하며 .idx 파일은 썸캐시의 정보와 썸캐시 내에서 특정 썸네일을 검색하기 위한 Index 등을 기록하고 있다.

삼성 스마트폰의 경우 imgcache, micro, mini는 썸네일을 저장하는 해상도에 의해서 구분되며 imgcache, micro, mini 순으로 높은 해상도를 가지고 있으나 정확한 해상도는 기기별로 상이하다. LG 스마트폰은 6.0.x (Marshmallow) 이전 버전까지는 imgcache에 해상도별로 구분하여 모두 저장하였으나 6.0.x (Marshmallow) 이후 버전부터는 해상도별로 구분하여 저장하고 있다. 기기별 일반적인 썸네일 해상도는 Table 2와 같다. 제조사별, 모델별로 해상도에 있어서 차이가 있기 때문에 단편화된 썸네일을 블록 단위로 추출하였을 때 Fig. 1과 같이 해상도가 작을수록 원본 그래픽 파일 대비 복원율이 높아진다. 따라서 해상도가 낮은 썸네일을 우선적으로 추출하는 것이 유리하다.

Table 2. Thumbnail Resolution

| Model | | Identifier | | |
|---|---|---|---|---|
| | | 0x31 | 0x32 | 0x33 |
| SAMSUNG | SM-N920 (Galaxy Note5) | 960×540 | 474×267 | 175×98 |
| | SHV-E330 (Galaxy S4) | 640×360 | 480×270 | 72×72 |
| | SM-N900 (Galaxy Note3) | 640×360 | 320×240 | 120×120 |
| LG | LG-F460 (LG G3) | 640×454 | 219×219 | 1920×1080 |
| | LG-F320 (LG G2) | 640×360 | 144×144 | 1920×1080 |
| | LG-F240 (LG GPro) | 640×360 | 200×200 | 1920×1080 |

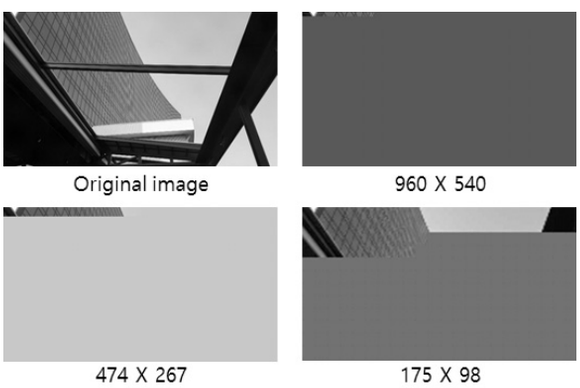


Fig. 1. Extracted thumbnail by block size (4KB)

### 3.2 .0 및 .1 썸캐시 내부 구조

하나의 썸캐시는 3개의 파일(.0, .1, .idx)로 구성된다. 확장자 .0, .1은 썸네일을 직접 저장하며 .idx는 .0, .1로부터 특정 썸네일을 찾을 수 있도록 index와 논리적 위치 정보를 기록하고 있다. .idx를 기기 및 버전과 무관하게 동일한 구조를 가지는 반면 .0, .1은 썸네일의 metadata를 표현하는 방식에서 기기마다, 안드로이드 버전마다 상이한 구조를 가지고 있다.

.0 또는 .1 파일은 사용자가 갤러리 애플리케이션에서 사진을 확인한 순서대로 썸네일을 기록하는 썸캐시 파일로, header 4바이트(0x108524BD)에 이어서 썸네일의 메타데이터가 저장되는 description과 실제 썸네일 content가 반복적으로 기록되는 구조를 가지고 있다[3].

썸캐시는 초기화 상태에서 .0 썸캐시에 먼저 기록하고 용량이나 썸캐시의 양이 일정 수준에 도달할 경우 .1 썸캐시에 기록된다. 만약 .1 썸캐시도 일정 수준에 도달할 경우 .0 썸캐시의 썸네일을 삭제하고 새로 기록한다.

썸캐시 header 및 썸네일 content는 모든 기종 및 안드로이드 버전에서 동일하게 사용 중이며 썸네일 description만 기종별로 상이한데 .0 썸캐시의 세부적인 구조는 Table 3과 같다.

4바이트의 썸캐시 header 이후로 썸네일의 index가 8바이트로 기록되어 있으며 start offset는 썸캐시 내에서 썸네일이 시작되는 논리적 주소이다. thumbnail size는 썸네일의 실제 사이즈로, thumbnail metadata의 시작위치부터 계산되어 thumbnail size만큼 이동하면 다음 썸네일이 등장한다.

thumbnail metadata는 기종과 무관하게 'local/image(동영상일 경우 video)/item'이 유니코드로 인코딩되어 있으며 그 외에 그래픽 파일의 생성 순번(counter), 해상도 식별자(identifier), 원본 그래픽 파일의 생성시각이 unix time으로 기록되어 있다. 여기서 그래픽 파일의 생성 순번은 스마트폰 내에서 썸네일의 원본 그래픽 파일이 생성된 순번을 의미한다. 즉, 같은 원본 그래픽 파일이 여러 종류의 썸캐시에 생성되더라도 순번은 항상 동일하다. 또한 해상도 식별자는 썸네일이 현재 어떠한 썸캐시에 생성되었는지를 의미한다.

원본 그래픽 파일의 생성시각은 말 그대로 썸네일의 생성시각, 즉 원본 그래픽 파일을 갤러리 애플리케이션으로 본 시각이 아닌 원본 그래픽 파일이 스마트폰에 생성된 시각을 말한다. 만약 원본 그래픽 파일이 스마트폰 카메라로 찍은 사진이라면 그 사진을 찍은 시각이 될 것이며 인터넷에서 다운로드 받은 그래픽 파일이라면 실제 다운로드 받은 시각이 된다. 이는 스마트폰을 이용한 사용자의 행위를 분석할 때 중요한 요소로 적용될 수 있다.

thumbnail metadata 부분은 Table 4와 같이 기종별로 다양한 형태를 가지고 있으며 LG 스마트폰에서는 Table 4 및 Fig. 2와 같이 원본 그래픽 파일의 경로 및 파일명을 함께

Table 3. Structure of .0 Thumbcache

| Name | | Offset | Size | Description |
|---|---|---|---|---|
| Header | | 0x0~0x3 | 4byte | 0x10 85 24 BD |
| Thumbnail description | Index | 0x4~0xB | 8byte | Index of thumbnail |
| | CRC | 0xC~0xF | 4byte | CRC of thumbnail |
| | Start offset | 0x10~0x13 | 4byte | Start offset |
| | Thumbnail size | 0x14~0x17 | 4byte | Size of thumbnail |
| | Thumbnail metadata | 0x18~ | variable | Thumbnail count, resolution identifier and created time of original graphic file. |
| Thumbnail content | Thumbnail header | · | 4byte | 0xFF D8 FF E0 |
| | Thumbnail footer | · | 2byte | 0xFF D9 |

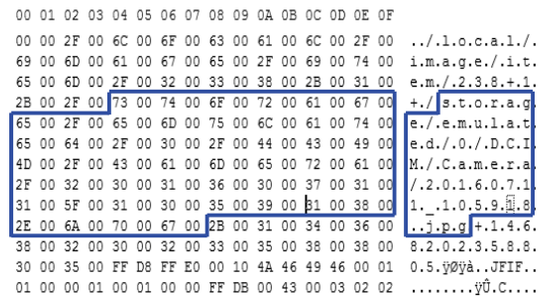


Fig. 2. Thumbnail of LG Smartphone

Table 4. Thumbnail Metadata

| Model | | Version | Metadata |
|---|---|---|---|
| SAMSUNG | SM-G930 (Galaxy 7) | 6.0.1 | /local/image/item/count +Unix Time(Dec)+identifier |
| | SM-N920 (Galaxy Note5) | | |
| | SHV-E330 (Galaxy S4) | 5.0.1 | /local/image/item/count +identifier\|Unix Time(Hex) |
| | | 4.4.2 | |
| | SM-N900 (Galaxy Note3) | 4.4.2 | |
| LG | LG-F460 (LG G3) | 6.0.0 | /local/image/item/count+identifier+/%Image Path% /%Image Name% +Unix Time(Dec) |
| | LG-F320 (LG G2) | 5.0.1 | |
| | LG-F240 (LG GPro) | 4.4.2 | |

기록하고 있다. 마지막으로 thumbnail content는 썸네일의 header(0xFF D8 FF E0)와 footer(0xFF D9)를 signature로 식별할 수 있도록 하고 있다.

### 3.3 .idx 파일

.idx 파일은 썸캐시 내에서 썸네일을 직접 찾아가기 위한 index와 썸캐시 내에서의 위치를 저장하는 역할을 중점적으로 수행하지만 좀 더 면밀히 분석하면 썸캐시에 대한 많은 정보를 얻을 수 있다. .idx 파일의 구조는 Table 5와 같다.

max .idx amount는 .idx 파일에서 보유할 수 있는 index의 최대 수량이다. 2절에서 설명한 바와 같이 한 가지 해상도의 썸네일은 .0과 .1 썸캐시 2개의 파일에 나누어 저장되기 때문에 여기서 명시된 값은 .0 또는 .1 썸캐시 중 하나의 썸캐시에 대하여 저장할 수 있는 index의 최대 수량이다. max thumbcache size는 1개의 썸캐시 파일이 가질 수 있는 최대용량이다. 하나의 썸캐시 파일은 이 값만큼의 데이터를 저장할 수 있고 만약 썸캐시가 이 용량에 도달할 경우 다른 썸캐시(.0 썸캐시일 경우 .1로, .1 썸캐시일 경우 .0으로)에 저장되기 시작한다.

앞서 언급한 max .idx amount와 연관지어 정리하면 썸네일은 .0 썸캐시에 먼저 기록되다가 .idx에 기록되는 index가 max .idx amount를 초과하거나 .0 썸캐시에 저장된 썸네일의 용량이 max thumbcache size를 초과할 경우 .1 썸캐시에 기록된다. 반대로 .1 썸캐시가 max .idx amount나 max thumbcache size에 도달하게 되면 .0 썸캐시를 삭제하고 새롭게 쓰기 시작한다.

active thumbcache는 현재 활성화되어 썸네일을 기록하고 있는 썸캐시를 가리킨다. 이 값이 0x0이면 .0 썸캐시가 활성화된 상태라는 것을 의미하고 0x1이면 .1 썸캐시가 활성화된 상태라는 것을 의미한다. 따라서 가장 최근에 생성된 썸네일을 확인하고 싶다면 이 값을 참조하여 현재 활성화된 썸캐시가 어느 것인지 확인해야 한다. thumbnail count는 활성화되어 있는 썸캐시에 생성되어 있는 썸네일의 개수를 의미하며 thumbcache size는 활성화되어 있는 썸캐시의 현재 용량을 표시한다.

.idx 파일의 metadata 이후에는 썸네일별로 index가 기록되는데 index는 thumbnail index와 thumbnail offset으로 구성되어 있다. thumbnail index는 동일한 원본 그래픽 파일에 동일한 index를 갖도록 8바이트 값으로 생성되며 썸캐시 내에서 특정 썸네일을 검색하기 용이하도록 도와준다. thumbnail offset은 thumbnail index가 가리키는 썸네일이 썸캐시 내에서 저장되어 있는 상대적인 주소를 가리킨다. 이 값을 통하여 다른 해상도의 썸캐시에 저장되어 있는 썸네일과 구별할 수 있다.

### 3.4 사용자의 행위에 따른 썸네일의 변화

안드로이드 스마트폰의 썸네일에 대해 연구한 Ming Di Leom의 논문[3]에서 사용자의 행위에 따른 썸네일의 변화에

Table 5. Structure of .idx

| Name | | Offset | Size | Description |
|---|---|---|---|---|
| Header | | 0x0~0x3 | 4byte | 0x30 30 27 B3 |
| Meta-data | Max .idx amount | 0x4~0x7 | 4byte | Maximum index amount |
| | Max thumcache size | 0x8~0xB | 4byte | Maximum size of thumbcache |
| | Active thumbCache | 0xC~0xF | 4byte | Referencing thumbcache |
| | Thumbnail count | 0x10~0x13 | 4byte | Thumbnail amount |
| | Thumbcache size | 0x14~0x17 | 4byte | Thumbcache's current size |
| | CRC | 0x1C~0x1F | 4byte | CRC of thumbcache .idx |
| Index | thumbnail index | · | 8byte | Index of thumbnail |
| | Thumbnail offset | · | 4byte | Logical offset of thumbnail |

Table 6. Condition of Experiment

| No. | | User action |
|---|---|---|
| #1. Create | 1-1 | take 10 pictures |
| | 1-2 | take 1 picture and preview the picture |
| | 1-3 | download 3 graphic files from internet |
| #2. Execute | 2-1 | execute gallery app |
| | 2-2 | enter 'Camera' folder |
| | 2-3 | reduce previewed thumbnails |
| | 2-4 | preview 200 graphic files |
| | 2-5 | open 1 graphic file |
| | 2-6 | open 5 graphic files |
| | 2-7 | edit 1 graphic file and save |
| | 2-8 | use slideshow function |
| | 2-9 | open graphic file from sdcard |
| | 2-10 | copy graphic file to another folder |
| | 2-11 | move graphic file to another folder |
| #3. Delete | 3-1 | delete 1 graphic file in gallery app |
| | 3-2 | delete 1 folder including graphic files in gallery app: |
| | 3-3 | delete 1 graphic file using file explorer app |
| | 3-4 | delete previewed graphic file using camera app |

대해 연구가 진행되었으나 단일 스마트폰 기종 및 안드로이드 버전에 대한 연구만 진행되어 보편적인 결론에 도달하지 못하였다. 따라서 본 장에서는 삼성 및 LG 스마트폰에서 다양한 안드로이드 버전별로 사용자의 행위에 따라 썸네일이 어떻게 변화하는지 알아보았다.

사용자의 행동에 따른 썸네일의 변화를 정확하게 확인하기 위하여 스마트폰은 실험 전에 공장초기화를 실시하고 다른 애플리케이션의 구동없이 Table 6에 명시된 행동만을 실시하였다.

실험에 사용된 스마트폰은 삼성 갤럭시 노트5 (SM-N920, 6.0.1), 삼성 갤럭시 S5 (SHV-E330, 4.4.2), LG G3(LG F400, 4.4.2) 총 3대를 사용하였다.

1) 그래픽 파일 생성 실험 결과

Table 7의 결과와 같이 삼성 갤럭시 노트5나 LG G3는 카메라 애플리케이션으로 사진을 촬영하여도(실험 #1-1) 사진의 썸네일이 생성되지 않았다. 그러나 삼성 갤럭시 S4는 사진 촬영 후 갤러리 애플리케이션을 실행하지 않아도 썸네일이 생성되었는데 이때 생성된 썸네일은 갤러리 애플리케이션 실행시 폴더별로 가장 최근에 생성된 그래픽 파일의 썸네일이었다.

즉, 다른 기종에서는 갤러리 애플리케이션을 실행해야만 썸네일을 생성하는 반면, 삼성 갤럭시 S4는 갤러리 애플리케이션의 실행과 무관하게 OS에서 자체적으로 갤러리 애플리케이션의 실행에 대비하여 썸네일을 생성한다. 그러나 카메라 애플리케이션의 미리보기 기능(실험 #1-2)은 갤러리 애플리케이션과 연동되어 있어 썸네일이 생성되는 것을 알 수 있었다.

한편 인터넷에서 그래픽 파일을 다운로드하는 행위(실험 #1-3)만으로는 모든 기종에서 썸네일이 생성되지 않았다.

Table 7. Result of Experiment

( ○ : generated, × : not generated )

| No. | | SAMSUNG | | LG |
|---|---|---|---|---|
| | | Gallaxy Note 5 (SM-N920) | Gallaxy S4 (SHV-330) | LG G3 (LG F400) |
| | | 6.0.1 | 4.4.2 | 4.4.2 |
| #1. Create | 1-1 | × | ○ | × |
| | 1-2 | ○ | ○ | ○ |
| | 1-3 | × | × | × |
| #2. Execute | 2-1 | ○(1 thumbnail per folder) | ○(1 thumbnail per folder) | ○(1 thumbnail per folder) |
| | 2-2 | ○(0x32 Size) | ○(0x32 Size) | ○(0x32 Size) |
| | 2-3 | ○(0x33 Size) | ○(0x33 Size) | × |
| | 2-4 | ○ | ○ | ○ |
| | 2-5 | created 3 thumbnails | created 7 thumbnails | created 1 thumbnail |
| | 2-6 | plus 4 thumbnails | plus 2 thumbnails | plus 4 thumbnails |
| | 2-7 | ○(result graphic file) | ○(result graphic file) | ○(result graphic file) |
| | 2-8 | × | × | × |
| | 2-9 | ○ | ○ | ○ |
| | 2-10 | ○ | ○ | ○ |
| | 2-11 | ○ | ○ | ○ |
| #3. Delete | 3-1 | no change | no change | no change |
| | 3-2 | no change | no change | no change |
| | 3-3 | no change | no change | no change |
| | 3-4 | no change | no change | no change |

즉, 사용자가 사진을 촬영하거나 그래픽 파일을 다운로드 받는 등 스마트폰에 그래픽 파일을 생성한다고 해서 반드시 썸네일이 생성되는 것은 아니다.

2) 갤러리 애플리케이션 실행 실험 결과

사용자가 갤러리 애플리케이션을 실행하면(실험 #2-1) 애플리케이션은 폴더별로 가장 최근에 생성된 그래픽 파일의 썸네일을 생성하며 이때 해상도는 0x32에 해당하는 해상도를 가진다. 즉, 사진을 촬영하여 썸네일이 생성되지 않았더라도 촬영 직후 사용자가 갤러리 애플리케이션을 실행하는 순간 가장 마지막에 촬영한 사진의 썸네일이 생성되는 것이다.

갤러리 애플리케이션에서 폴더로 들어가면 보이는 그래픽 파일별 썸네일의 크기를 조작(실험 #2-2, #2-3)하는 것으로 크기별 썸네일을 생성할 수 있다. 폴더 내에 한 화면으로 다 확인할 수 없을 정도의 많은 그래픽 파일이 있는 폴더에 들어갈 경우(실험 #2-4) 폴더의 모든 그래픽 파일에 대한 썸네일이 생성되는 것이 아니라 화면상에 보이는 그래픽 파일 및 그 다음화면의 그래픽 파일에 대한 썸네일만 생성되었다.

그래픽 파일 중 하나를 직접 터치하여 미리보기 화면을 실행할 경우(실험 #2-5, #2-6) 0x31에 해당하는 해상도의 썸네일이 생성되었는데 이때 직접 본 썸네일뿐만 아니라 사용자가 사진을 넘기는 행동에 대비하여 미리보기를 실행한 그래픽 파일의 전후로도 썸네일이 생성되었다.

또한 갤러리 애플리케이션에서 자체적으로 제공하는 편집 기능을 이용하여 그래픽 파일을 수정하고 저장할 경우(실험 #2-7) 마지막에 저장된 수정본의 미리보기 화면이 나오기 때문에 썸네일이 생성되었으나, 슬라이드 쇼 기능을 수행하는 경우(실험 #2-8) 모든 그래픽 파일을 슬라이드 쇼 기능으로 미리보기를 하였더라도 썸네일은 생성되지 않는다.

스마트폰 내의 그래픽 파일이 아닌 외장형 sd카드에 저장된 그래픽 파일을 열어본 경우(실험 #2-9) 역시 썸네일이 생성되었는데 삼성 스마트폰의 경우 내장 메모리인지 외장 메모리인지 구분할 수 없었지만 LG 스마트폰의 경우 원본 그래픽 파일의 경로를 저장하고 있기 때문에 해당 그래픽 파일이 외장 메모리에 저장되어 있던 그래픽 파일이라는 것을 알 수 있었다.

갤러리 애플리케이션 내에서 그래픽 파일을 복사(실험 #2-10) 또는 이동(실험 #2-11)하는 경우 새롭게 복사되거나 이동되면서 생성된 그래픽 파일의 썸네일이 생성되며 이때 썸네일에 기록된 시각은 복사 또는 이동이 실행된 당시의 시각이라는 것을 알 수 있었다.

즉, 사용자가 스마트폰에 저장되어 있는 그래픽 파일을 직접 열어보지 않더라도 다양한 행위의 결과에 의해 그래픽 파일의 썸네일이 생성된다는 것을 알 수 있으며 그래픽 파일을 복사 또는 이동시킬 경우 그러한 행동을 실시한 시각을 알 수 있다.

3) 그래픽 파일 삭제 실험 결과

그래픽 파일을 삭제하는 다양한 행동을 모두 실험한 결과(실험 #3-1 ~ #3-4) 원본 그래픽 파일의 변화와는 무관하게 썸네일에는 전혀 변화가 없는 것을 확인할 수 있었다. 앞선 실험 #2-11에서 그래픽 파일을 다른 폴더로 이동시키는 경우에도 원리는 원본 그래픽 파일을 삭제하고 새로운 곳에 그래픽 파일을 생성하는 것이나 이동 전, 이동 후 썸네일이 모두 유지되었다.

즉, 사용자가 스마트폰의 그래픽 파일을 삭제한다고 하더라도 썸네일에는 삭제된 그래픽 파일의 썸네일이 계속 남아 있게 된다.

### 3.5 썸네일의 특성

지금까지 살펴본 썸네일에 대한 실험 결과와 같이 썸네일은 그래픽 파일의 미리보기 기능을 위하여 생성되기 때문에 갤러리 애플리케이션을 실행하는 것만으로도 Fig. 3과 같이 최초 실행시 보이는 그래픽 파일들에 대한 미리보기 기능을 제공하기 위해 썸네일이 생성된다. 이러한 썸네일들은 미리보기 화면의 크기에 따라 micro나 mini와 같은 작은 해상도의 썸캐시에 생성된다.

또한 썸네일은 원본 그래픽 파일로부터 생성되지만 원본 그래픽 파일의 수정, 삭제로부터 직접 영향을 받지 않기 때문에 썸네일이 생성된 상태에서 원본 그래픽 파일이 삭제되었다고 하더라도 썸네일은 삭제되지 않는다는 것도 확인하였다. 이러한 특성으로 인하여 사용자가 원본 그래픽 파일을 삭제하여 카빙으로 복구하기 힘들어진다고 하더라도 썸네일을 분석함으로써 어떠한 그래픽 파일이 삭제되었고 그 모습이 어떠한 형태인지 쉽게 알 수 있다.

그리고 하나의 그래픽 파일에 대하여 다양한 해상도로 썸네일이 생성되기 때문에 하나의 썸네일이 단편화되어 복원이 되지 않더라도 다른 해상도의 썸네일이 복원될 가능성이 높다. 따라서 사용자가 썸캐시를 모두 삭제해도 다른 해상

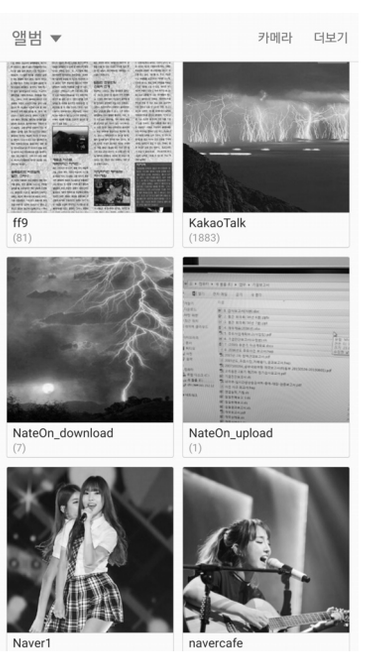


Fig. 3. Initial Screen of Gallery App

도에서는 단편화되지 않고 복원될 확률이 더 높다. 특히 해상도가 낮은 썸네일은 단편화되더라도 복원율이 높기 때문에 원본 그래픽 파일의 형태를 추측하는데 있어서 용이하게 활용될 수 있다.

또한 파일 단위로 저장되는 다른 썸네일들과는 달리 썸캐시라는 파일 안에 연속적으로 저장되며 썸캐시 안에서 metadata를 별도로 기록하고 있기 때문에 썸네일 구조를 분석하면 동일한 header와 footer를 사용하는 썸네일과 구별하여 갤러리 애플리케이션이 생성하는 썸네일만을 추출할 수도 있을 것이다.

## 4. 썸네일 추출 방법

기존의 그래픽 파일이나 썸네일의 카빙 방법은 header와 footer를 이용하여 파일을 복원하는 방법을 사용해 왔다. 그러나 이러한 카빙 방식은 그래픽 파일의 형태만을 확인할 수 있을 뿐, 카빙된 그래픽 파일이 일반 그래픽 파일인지 썸네일인지 구분할 방법이 없었다.

따라서 썸네일의 특징을 고려하여 관련 정보를 추출하고 그래픽 파일의 형태를 확인할 수 있도록 Fig. 4와 같은 알고리즘을 사용하였다.

먼저 썸네일의 추출 방법을 썸네일의 상태에 따라 논리적 추출과 물리적 추출로 구분하였으며 물리적 추출간 단편화된 썸네일을 복원하는 방법을 다루고 이를 구현한 도구를 소개한다.

### 4.1 논리적 추출

논리적인 복원 방식은 스마트폰 내부의 파일시스템 정보가 온전하게 유지되어 있고 사용자에 의하여 썸네일이 제거되지 않은 상태에서 썸네일을 완전하게 스마트폰으로부터 추출 가능한 상태에서 사용하는 방식이다.

수사관은 썸네일을 완전히 추출할 수 있기 때문에 모든 썸네일은 단편화되지 않은 상태이다. 따라서 Fig. 4의 logical extract 부분의 순서도를 통하여 썸캐시 내에 저장되어 있는 썸네일을 모두 완전히 추출할 수 있다.

먼저 썸네일의 header를 확인하여 추출할 썸네일의 수량

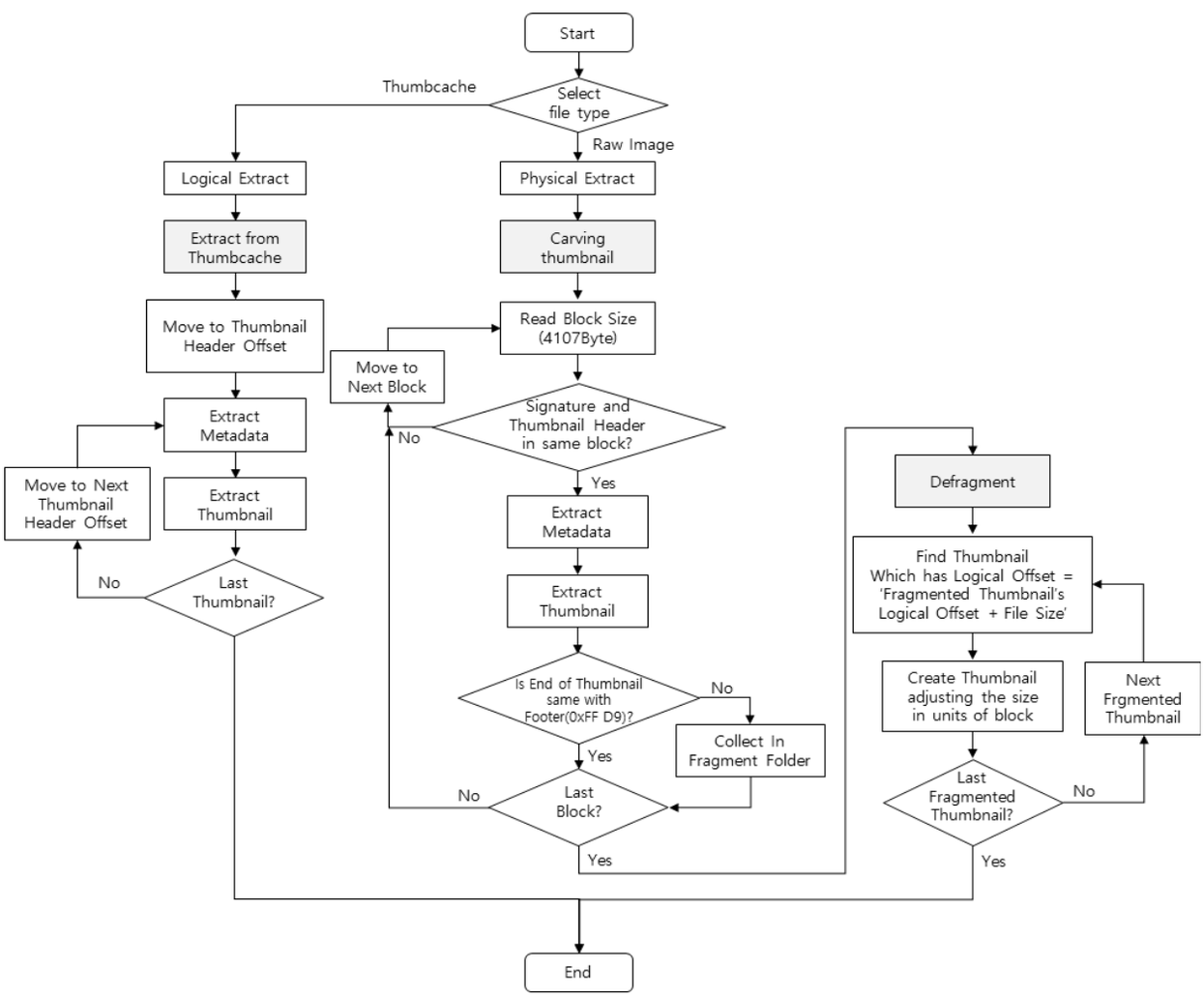


Fig. 4. Algorithm of Thumbnail Caving

Table 8. Result of Thumbnail Carving

| Model | | Version | Origin thumbnail | | | | Extracted thumbnail | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | total | 0x31 | 0x32 | 0x33 | total | 0x31 | 0x32 | 0x33 |
| SAMSUNG | SM-N910 | 6.0.1 | 604 | 158 | 408 | 38 | 598 | 155 | 405 | 38 |
| | SM-G900 | 4.4.2 | 109 | 12 | 58 | 39 | 108 | 12 | 57 | 39 |
| LG | LG F400 | 4.4.2 | 26 | 2 | 18 | 6 | 26 | 2 | 18 | 6 |

을 확인하고 header에 앞서 기록되어 있는 썸네일 metadata를 추출한 후 썸네일을 추출하여 출력한다.

썸네일은 썸캐시 내에서 동일한 형식이 반복되는 구조를 가지고 있기 때문에 썸네일을 온전하게 추출할 수 있는 논리적 추출 방식은 이를 구현하고 적용하는데 큰 문제가 없다.

### 4.2 물리적 추출

물리적 추출 방식은 썸네일을 저장하는 썸캐시의 metadata가 손상되어 썸캐시를 온전하게 복원할 수 없거나 사용자에 의하여 썸캐시가 제거되어 Inode를 이용한 썸캐시의 추출이 불가능한 상태에서 썸네일의 특징을 활용, Raw 이미지를 분석하여 썸네일을 추출하는 방식이다.

지금까지 살펴본 썸네일의 주요 구조적인 특징은 ① 블록의 header 부분이 아니더라도 썸네일의 header가 등장, ② 안드로이드 스마트폰의 기종을 무시하고 모든 썸네일은 반드시 '/.l.o.c.a.l.(0x2F 00 6C 00 6F 00 63 00 61 00 6C 00)'이라는 signature를 반드시 보유, ③ '/.l.o.c.a.l.' signature와 Header의 사이에 썸네일의 metadata가 저장, ④ 하나의 썸캐시 안에서 모든 썸네일은 순차적으로 등장한다는 것이다.

썸네일을 보관하고 있는 썸캐시의 metadata가 손상되어 있기 때문에 썸네일 또는 썸캐시의 고유한 signature를 통하여 썸네일을 찾아 추출해야 하는데 본 논문에서는 썸네일을 찾는 signature로 '/.l.o.c.a.l.'을 사용하였다.

썸네일의 header인 '0xFF D8 FF E0'는 갤러리 썸네일 외에도 다수의 애플리케이션에서 썸네일을 저장하는 용도로 사용하기 때문에 갤러리 썸네일만을 추출하는 용도로 사용하는 데에는 적절하지 않다. 그러나 위에서 언급한 '/,l,o,c,a,l.' signature와 함께 조합하면 갤러리 썸네일만을 정확하게 추출할 수 있다.

Fig. 4와 같이 먼저 Raw 이미지로부터 블록 단위로 읽으면서 '/.l.o.c.a.l.' signature를 찾는다. 단, signature가 블록에 걸쳐 있으면 검색간 누락될 수 있으므로 검색용 블록 크기는 signature의 크기를 고려하여 4107바이트(블록 크기인 4096바이트 + signature 크기 12바이트 - 1바이트)로 설정한다.

블록 내에서 signature가 발견되면 signature의 뒤에 썸네일의 header가 존재하는지를 확인하고 만약 signature가 없거나, 있더라도 같은 블록 내에 썸네일의 Header가 없으면 다음 블록을 검색하는 방식을 반복하며 썸네일을 검색한다.

signature와 썸네일 header가 존재하는 블록을 발견하면 1MB 크기로 블록을 읽어온 후 썸네일의 구조를 이용하여 metadata 및 썸네일 이미지를 추출한다.

썸네일 metadata 중에서 추출할 것으로는 ① counter, ② identifier, ③ file creation time, ④ file size, ⑤ logical offset, ⑥ physical offset이다. 이 모든 metadata는 썸네일을 추출하는 과정에서 같은 블록 내에 있다면 모두 추출할 수 있다. 특히 LG 스마트폰의 경우 원본 그래픽 파일의 저장경로 및 파일명을 저장하고 있기 때문에 반드시 추출해야 한다. 이때 signature의 8바이트 앞에 4바이트 길이로 thumbnail size가 기록되어 있기 때문에 이 부분을 읽어낼 수 있다면 정확한 크기만큼의 썸네일을 추출할 수 있다.

또한 같은 방식으로 thumbnail size만큼을 signature로부터 읽었을 때 썸네일 footer가 존재하지 않는다면 이 썸네일은 단편화되었다는 것을 알 수 있으며, metadata 중 logical offset과 함께 분석하면 단편화된 썸네일을 복원하는데 활용할 수 있다. 단편화된 썸네일을 복원하는 방법은 다음 절에서 다루겠다.

한편 썸네일이 단편화되었다고 하더라도 단편화되기 이전까지의 썸네일 데이터가 남아있기 때문에 원본 썸네일의 일부를 확인할 수 있다.

이 방식을 이용하여 Table 8과 같이 삼성 갤럭시 노트4 (SM-N910, 6.0.1)의 썸네일 604개 중 598개를 추출하였고 삼성 갤럭시 S5 (4.4.2)의 썸네일 109개 중 108개를 추출하였으며 LG G3 (F400, 4.4.2)의 썸네일 26개 중 26개를 추출하여 평균 99.0%의 추출률을 기록하였다.

### 4.3 단편화된 썸네일 복원

복원된 썸네일 총 732개 중 단편화된 썸네일은 30개로 전체 대비 4.1%를 차지하고 있었으며 이를 복원하는데 썸네일의 특징을 활용하여 Fig. 5와 같은 알고리즘을 적용하였다.

앞서 다룬 바와 같이 최초 썸네일을 추출하면서 metadata로부터 file size만큼을 읽었을 때 썸네일의 footer가 아닌 경우 해당 썸네일은 단편화되어 있는 상태이기 때문에 관련 metadata를 'Frag_Storage'라는 별도의 리스트 변수에 저장한다. 전체적으로 Raw Image로부터 썸네일의 추출이 끝난 이후부터 Frag_Storage에 저장되어 있는 단편화된 썸네일 metadata를 바탕으로 추출된 모든 썸네일과 비교하며 자신의

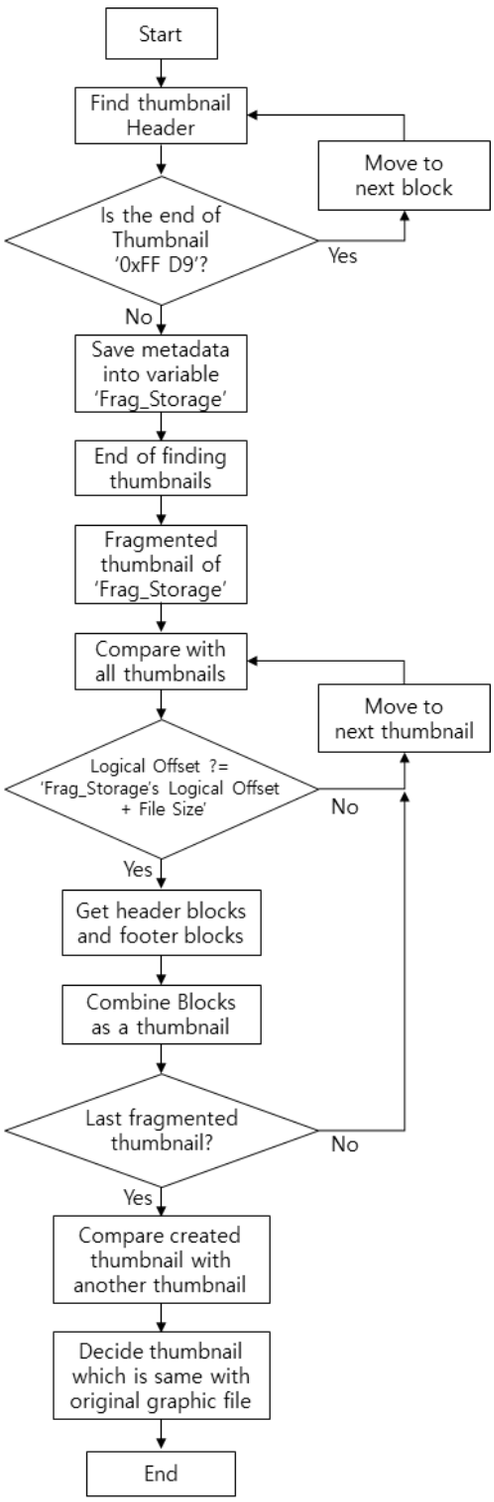


Fig. 5. Algorithm of Defragment

footer를 가지는 썸네일을 찾는다.

썸네일은 metadata와 썸네일 이미지가 연속하여 기록되기 때문에 하나의 썸네일의 앞부분에는 이전 썸네일의 뒷부분이 연속적으로 저장되어 있다. 또한 썸네일별로 metadata에는 썸캐시 내에서의 logical offset와 thumbnail size를 기록하고 있기 때문에 특정 썸네일의 logical offset에서 thumbnail size를 더한 위치에는 다음 썸네일이 존재한다. 이를 단편화된 썸네일을 복원하는데 활용하여 단편화된 썸네일의 logical offset과 thumbnail size를 더한 값이 logical offset이 되는 썸네일을 찾는다면 그 썸네일은 단편화된 썸네일의 다음 썸네일이 되며 다음 썸네일의 앞에 기록되어 있는 데이터가 단편화된 썸네일의 footer 이전 데이터들이 된다.

썸네일은 일반적으로 200KB 이하의 데이터로 구성되어 있으며 Ext4 파일시스템의 특징상 단편화가 최소화되기 때문에 두 조각으로 단편화되었을 경우 복원이 용이하다[5].

단편화된 썸네일이 두 조각으로 단편화되어 있다고 가정하면 앞서 단편화된 썸네일은 header 부분을 보유하고 있고 다음 썸네일의 앞부분은 footer 부분을 보유하고 있다. 여기서 단편화된 썸네일의 metadata로부터 썸네일의 크기를 알 수 있으므로 Fig. 6과 같이 header와 footer를 블록 단위로 유지하면서 header 블록의 뒷부분과 footer 블록의 앞부분을 블록 단위로 더하고 빼면서 각각을 그래픽 파일로 추출한다.

썸네일의 용량이 작을수록 추출되는 썸네일의 양도 적으며 반대로 썸네일의 용량이 크면 추출되는 썸네일의 양도 많아진다. 그러나 썸네일이 두 조각으로 단편화되어 있다고 가정하면 추출된 썸네일 중 하나는 원본 썸네일과 일치할 것이다. 특히 같은 counter값을 가지는 다른 해상도의 썸네일과 직접 대조하면 정확하게 원본 썸네일이 어떤 것인지 알 수 있다. 원본 썸네일이 맞는지 여부는 사실상 수사관의 육안으로 직접 확인하는 것이 가장 정확하기 때문에 마지막 결정은 자동화가 아닌 수사관이 직접 확인해야 한다.

앞서 언급한 3대의 스마트폰에서 추출된 썸네일 중에서 단편화된 썸네일 30개에 대하여 위 방식을 사용한 결과 30개 모두 원본 썸네일로 추정 수 있는 썸네일을 복원할 수 있었다.

### 4.4 썸네일 추출 도구(Thumbnail Carver)

지금까지 본 장에서 설명한 썸네일 복원 방식을 적용하여 썸캐시 또는 Raw image로부터 썸네일을 추출하는 도구를 Fig. 7과 같이 개발하였다.

이 도구는 안드로이드 스마트폰의 썸캐시 또는 Raw image를 입력받아 썸네일 및 그 metadata를 추출하여 출력해 주며 Raw image일 경우 단편화된 썸네일도 추출하여 출력한다.

metadata를 추출하여 출력해 주기 때문에 사용자는 원본 그래픽 파일의 생성 시각이나 생성 순번을 알 수 있고 같은 counter에 다른 해상도를 가지는 썸네일을 동시에 보여주어 사용자가 쉽게 비교하고 원본 그래픽 파일의 모습을 추측하기 용이하게 하였으며 모든 metadata 정보는 Fig. 8과 같이 csv 파일로 출력하여 사용자에 의해 수동으로 정렬·검색 및 분석이 가능하도록 하였다.

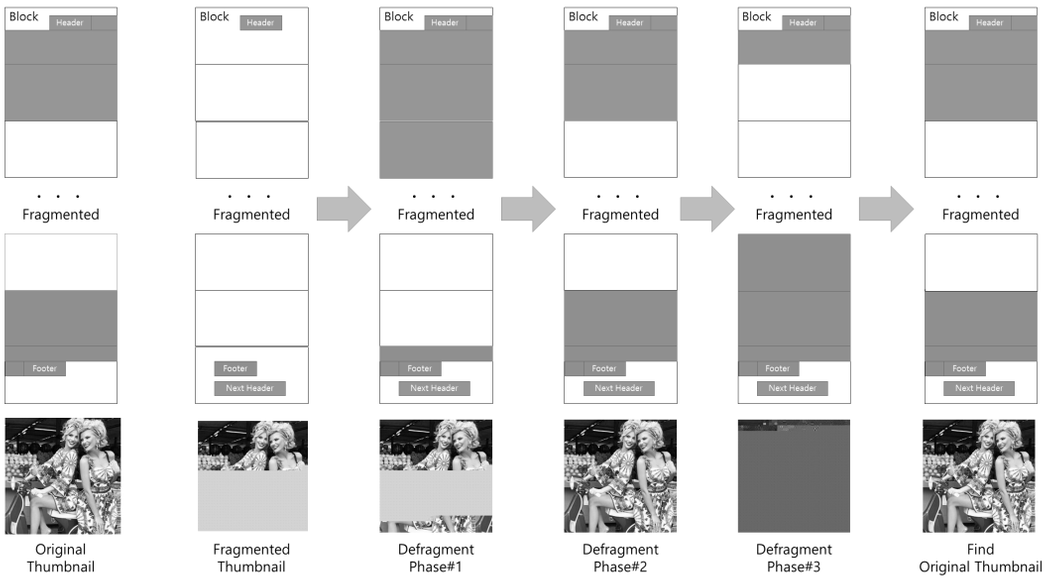


Fig. 6. Defragment of Two-piece Fragmented Thumbnail

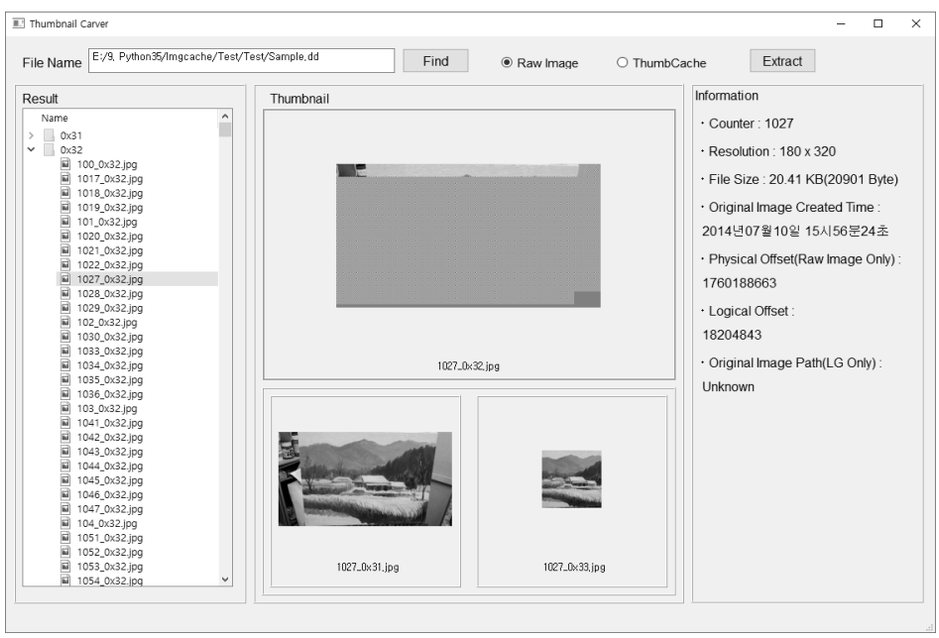


Fig. 7. Thumbnail Carver

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Count | Counter | Identifier | Resolution | File Size | Created Ti | Original P | Physical C | Logical Of | Fragment | |
| 2 | 0 | 3249 | 0x32 | 320×180 | 17.69 KB(1 | 2015년05월 | Unknown | 5.37E+08 | 1397604 | 0 | |
| 3 | 1 | 3248 | 0x32 | 240×320 | 26.24 KB(2 | 2015년05월 | Unknown | 5.37E+08 | 1415798 | 0 | |
| 4 | 2 | 3227 | 0x32 | 180×320 | 23.40 KB(2 | 2015년05월 | Unknown | 5.37E+08 | 1442745 | 0 | |
| 5 | 3 | 3226 | 0x32 | 180×320 | 20.12 KB(2 | 2015년05월 | Unknown | 5.37E+08 | 1466782 | 0 | |
| 6 | 4 | 3224 | 0x32 | 320×180 | 20.16 KB(2 | 2015년05월 | Unknown | 5.37E+08 | 1487458 | 0 | |
| 7 | 5 | 3225 | 0x32 | 180×320 | 22.06 KB(2 | 2015년05월 | Unknown | 5.37E+08 | 1508176 | 0 | |
| 8 | 6 | 3223 | 0x32 | 180×320 | 21.31 KB(2 | 2015년05월 | Unknown | 5.37E+08 | 1530842 | 0 | |
| 9 | 7 | 3221 | 0x32 | 180×320 | 23.82 KB(2 | 2015년05월 | Unknown | 5.37E+08 | 1552744 | 0 | |
| 10 | 8 | 3220 | 0x32 | 180×320 | 18.38 KB(1 | 2015년05월 | Unknown | 5.37E+08 | 1577213 | 0 | |
| 11 | 9 | 3219 | 0x32 | 180×320 | 15.41 KB(1 | 2015년05월 | Unknown | 5.37E+08 | 1596109 | 0 | |
| 12 | 10 | 3217 | 0x32 | 180×320 | 14.88 KB(1 | 2015년05월 | Unknown | 5.37E+08 | 1611967 | 0 | |

Fig. 8. Metadata Saved as .csv

또한 단편화된 썸네일은 어떠한 썸네일이 원본 썸네일 파일인지 확신할 수 있는 알고리즘이 없기 때문에 동일 썸네일에 대한 복원된 썸네일을 모두 사용자에게 보여 주어 사용자가 복원된 썸네일을 보고 직접 판단할 수 있도록 하였다.

블록 단위로 탐색하므로 대용량 Raw image에 대해서도 별도의 이미지 분할 없이 추출이 가능하며 이미 metadata에 저장된 용량만큼만 읽어 썸네일을 생성하므로 불필요한 저장 공간 낭비를 최소화할 수 있다.

## 5. 썸네일의 디지털 포렌식 분석 방법

안드로이드 스마트폰에서 생성되는 썸네일은 카빙으로 추출된 그래픽 파일들보다 범죄수사에 있어서 훨씬 효과적으로 활용할 수 있다. 본 장에서는 디지털 포렌식 관점에서 썸네일의 분석 방법에 대하여 알아본다.

### 5.1 타겟 사진의 존재 유무 확인

용의자의 입장에서는 그래픽 파일 중에 범죄와 관련된 것은 반드시 삭제하려고 노력할 것이기 때문에 수사관이 용의자의 스마트폰을 압수한 시점에서 주요 증거가 될 그래픽 파일은 삭제되어 있을 가능성이 크다.

또한 Ext4 파일시스템의 특성상 원본 그래픽 파일이 삭제되면 카빙을 한다고 하더라도 그래픽 파일이 단편화되어 있을 경우 복원이 어렵다[2].

그러나 본 논문에서 제시한 도구를 활용할 경우 썸네일이 삭제되고 단편화되었더라도 높은 확률로 복원할 수 있으며 복원에 실패하더라도 같은 counter를 가지는 다른 해상도의 썸네일과 비교하면 충분히 원본 그래픽 파일을 추정할 수 있다.

지금까지 살펴본 썸네일의 특성을 적용하면 원본 그래픽 파일이 삭제되었다고 하더라도 스마트폰에 썸네일이 생성되어 있다면 그에 해당하는 원본 그래픽 파일이 스마트폰 내에 저장되어 있었다는 것을 의미한다. 따라서 용의자는 자신이 해당 그래픽 파일을 보유하고 있었다는 사실을 부인할 수 없게 된다[3].

예를 들어 아동 포르노 소지 혐의에 대한 수사관 아동 포르노의 썸네일이 용의자의 스마트폰에서 발견된다면 해당 썸네일에 대한 원본 그래픽 파일이 용의자의 스마트폰에 저장되어 있거나 저장되어 있었음을 알 수 있을 것이다. 따라서 아동 포르노를 소지하고 있는 것만으로도 처벌 대상이 되는 이 범죄 유형에 있어서 수사의 결정적인 단서가 될 수 있다.

### 5.2 특정 사진의 생성 시각 확인

4.4.x(Kitkat) 버전 이후의 안드로이드 스마트폰의 갤러리 애플리케이션에서 생성된 썸네일은 다양한 형태로 원본 그래픽 파일의 생성 시각을 기록하고 있다. 삼성 계열 스마트폰의 경우에는 1초 단위, LG 계열 스마트폰에서는 1천분의 1초 단위까지 구체적으로 생성 시각을 기록하고 있어 원본 그래픽 파일이 언제 생성되었는지 알 수 있다.

예를 들어 지하철 도촬범을 현행범으로 체포하였을 경우 용의자가 자신이 촬영한 사진을 삭제하였을 경우 삭제된 사진의 Inode에는 삭제된 시각만이 기록되기 때문에 사진을 카빙 방식으로 추출한다고 하더라도 언제 촬영했는지 입증할 수 없다. 이 경우 해당 사진의 썸네일로부터 사진의 생성 시각을 확인할 수 있으므로 용의자의 위치정보와 함께 분석하면 특정 시각에 지하철 내에서 해당 사진을 촬영했다는 것을 입증할 수 있다.

### 5.3 원본 사진 파일과의 연계성 입증

앞서 언급했듯이 갤러리 썸네일의 메타데이터에 기록되어 있는 시각은 원본 그래픽 파일의 생성 시각과 일치하기 때문에 원본 그래픽 파일이 삭제되지 않았다면 썸네일의 메타데이터에 기록된 생성시각을 통해 원본 그래픽 파일을 찾을 수 있다.

먼저 썸네일을 추출한 후 썸네일의 metadata에 기록된 Unix 시간값을 10진수로 변환한 후 초단위로 버림한 뒤 16진수로 전환한다.

16진수로 전환된 unix time은 4바이트의 값이 되며 이 값을 Ext4 파일시스템의 Inode상의 시간 값과 일치하는 Inode를 탐색하여 해당 데이터를 추출하면 Fig. 9와 같이 원본 그래픽 파일을 확인할 수 있다. 또한 만약 그래픽 파일을 생성하는 애플리케이션의 특징으로 인해 생성된 시각이 기록된다면 보다 수월하게 원본 그래픽 파일을 찾을 수 있다.

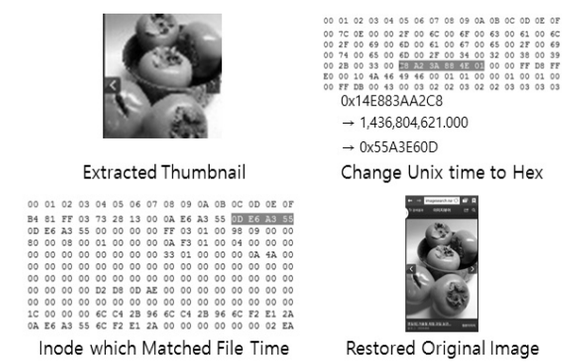


Fig. 9. Find Original Graphic File from Thumbnail

한편 LG 스마트폰은 썸네일 metadata에서 썸네일의 원본 그래픽 파일에 대한 경로가 직접 저장되어 있어 썸네일과 원본 그래픽 파일을 대조하는 것이 훨씬 용이하다. 특히 저장경로가 'Download' 폴더인지 카메라로 직접 촬영했을 때 저장되는 'DCIM'이나 'Camera' 폴더인지와 직접 촬영시 부여되는 파일명의 특징 등을 교차하여 분석하면 원본 그래픽 파일과의 대조뿐만 아니라 해당 그래픽 파일이 용의자의 스마트폰에서 촬영되었는지까지도 정확하게 확인할 수 있다.

### 5.4 그래픽 파일의 생성 순번 및 시각 추정

썸네일이 생성되지 않은 상태에서 갤러리 애플리케이션을 이용하지 않고 파일탐색기와 같은 애플리케이션을 이용하여

그래픽 파일을 삭제할 경우 그 그래픽 파일의 생성시각을 알 수 없다.

그러나 이후 새로운 그래픽 파일을 생성한 후 그에 대한 썸네일이 생성되었을 경우 기존에 저장되어 있는 썸네일 간에 counter에서 공백이 발생하게 된다.

이때 스마트폰으로부터 그래픽 파일을 카빙한 후 썸네일과 대조를 통해 썸네일에는 생성되지 않았으나 그래픽 파일은 존재할 경우 해당 그래픽 파일이 공백이 발생한 counter에 생성된 것이라고 추정할 수 있고 앞뒤로 생성된 썸네일의 생성 시각을 통하여 삭제된 그래픽 파일의 생성시각 역시 추정할 수 있다.

## 6. 결론 및 향후 계획

지금까지 썸네일은 디지털 포렌식 관점에서 삭제된 원본 그래픽 파일의 형태를 추정하는데 사용되며 범죄수사에 도움이 되어 왔으나 안드로이드 스마트폰에서 생성되는 썸네일에 대한 분석은 미비한 상황이었다.

그러나 본 논문에서 살펴본 바와 같이 원본 그래픽 파일과 썸네일을 함께 분석할 경우 원본 그래픽 파일의 생성시각과 같은 시간정보 뿐만 아니라 그와 관련된 사용자의 행위에 대한 정보도 얻을 수 있다. 또한 용의자가 고의로 원본 그래픽 파일 뿐만 아니라 썸네일까지 삭제한다고 하더라도 본 논문에서 제시한 도구를 사용하여 삭제된 썸네일을 복원할 수 있음을 보였으며 이를 통하여 원본 그래픽 파일의 형태를 확인할 수 있다.

특히 그래픽 파일이 범죄 혐의 입증에 중요한 증거가 될 경우에는 썸네일을 함께 분석하면 보다 효율적으로 수사할 수 있을 것이다.

향후에는 안드로이드 스마트폰에서 기본적으로 제공하는 갤러리 애플리케이션 외의 뷰어 애플리케이션이나 메신저에서 생성하는 썸네일의 특성을 분석할 계획이다.

## References

[1] Heun-jae Lee, “The problem of criminal policy and criminal law of pornography possession crime in children and juveniles,” *Korean Criminological Review*, pp.123-152, 2014.

[2] Kevin D. Fairbanks, “An analysis of Ext4 for digital forensics,” *Digital Investigation*, Vol.9, pp.118-130, 2012.

[3] Ming Di Leom, Christian Javier D’Orazio, Gaye Deegan and Kim-Kwang Raymond Choo, “Forensic Collection and Analysis of Thumbnails in Android,” *2015 IEEE Trustcom/BigDataSE/ISPA*, pp.1055-066, Aug., 2015.

[4] Igor Mikhaylov and Oleg Skulkin, Do not miss : New thumbnail databases in Android OS [Internet], http://www.weare4n6.com/do-not-miss-new-thumbnail-databases-in-android-os/.

[5] A. Mathor, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier, “The new ext4 filesystem : current status and future plans,” *2007 Linux Symposium Volume Two*, pp.21-33, Jun., 2007.

윤 대 호

e-mail : knightem@hanmail.net
2007년 육군사관학교 일본어과(학사)
2016년~현 재 고려대학교 정보보호대학원 정보보호학과 석사과정
관심분야 : Digital Forensic, Mobile Forensic

이 상 진

e-mail : sangjin@korea.ac.kr
1987년 고려대학교 수학과(학사)
1989년 고려대학교 수학과(석사)
1994년 고려대학교 수학과(박사)
1989년~1999년 ETRI 선임연구원
1999년~현 재 고려대학교 정보보호대학원 교수
2008년~현 재 고려대학교 디지털포렌식연구센터 센터장
관심분야 : Digital Forensic, Steganography, Hash Function