# hymnMark: Towards Efficient Digital Watermarking on Android Smartphones

***Nai Miao*, Yutao He**, Jane Dong****

*Department of Electrical and Computer Engineering
California State University, Los Angeles
5151 State University Drive, Los Angeles, CA 90032 USA
**Jet Propulsion Laboratory/California Institute of Technology
4800 Oak Grove Drive, Pasadena, California 91109
miaonai1229@gmail.com,Yutao.He@jpl.nasa.gov,jdong2@calstatela.edu

**ABSTRACT**

*Fast growth in ubiquitous use of digital-camera-equipped smartphones in our daily life has generated large amount of multimedia data such as images, audio, and video clips that need to be processed, stored, and transmitted on battery-powered mobile devices. Yet little research has been done to protect those multimedia data on smartphone platforms. This paper presents design and implementation of an efficient digital watermarking application, called hymnMark, to perform watermark embedding and detection for digital images on the Android platform. Preliminary evaluation shows that hymnMark can successfully embed in color images different types of watermarks with good resistance to noise as well as a number of digital signal processing attacks, in the meantime entail low power consumption.*

**Keywords:** Digital Watermarking, Power efficiency, Smartphone, Android, Discrete Cosine Transform.

## 1. INTRODUCTION

With the widespread of mobile networks, smartphone applications become more and more popular in recent years. The high mobility of the smartphones makes them ideal end platforms for multimedia applications such as web video, image browsing, photo sharing, etc. Since these digital media are highly subject to attacks including content modification, it is critical to better protect data integrity.

Digital watermarking is an effective technology to achieve authentication, copyright protection, and integrity of multimedia data and has been extensively studied in the past decades [1]. However, digital watermarking algorithms are computation-intensive and power hungry. How to develop an efficient digital watermarking application on resource-constrained mobile devices like smartphones requires a decent balance between performance and power consumption yet it has received little research attention. The research described in this paper aims to tackle the problem by developing a robust and power-efficient digital watermarking application targeted to smartphone platforms.

In our research, first, characteristics of wide range of digital watermarking algorithms have been investigated in the context of the energy-constrained Android smartphone platform. As a result of the study this set has been down-selected to one algorithm with optimal power efficiency for smartphone platforms. Second, the selected algorithm has been further optimized to reduce the computation cost with respect to Android computing environment. As a proof-of-concept, it is then implemented in an Android app called *hymnMark*. It features a user-friendly GUI to allow easy watermark generation, embedding and detection in Android-powered smartphones. Third, comprehensive empirical evaluation has been conducted to measure the algorithm's performance (i.e., robustness against various attacks) and power consumption.

In summary, our research makes the following main contributions:

1. We have developed the first Android app that performs digital watermarking completely on the smartphone. Our research has shown that effectiveness of a digital watermark algorithm for smartphones not only depends on its performance but also its power efficiency, since it will limit its sustained performance to protecting multimedia data.
2. We have identified a set of practical optimization techniques that proves to be effective in the smartphone environment. We believe that they can be applied to other digital watermarking applications on smartphones.
3. We have developed a micro-level instrumentation methodology that allows measurement of power consumptions inside one application. It enables fine-grained power profiling which in turns helps pinpoint the *hotspot* of one application for further optimization.

The rest of the paper is organized as below. Section 2 describes the related work. Section 3 provides an overview of digital watermarking technologies and the hardware configuration of Android environment. Section 4 presents in details the design of the hymnMark system. Complete

evaluation and results are given in Section 5, followed by conclusions and future in Section 6.

## 2. RELATED WORK

The computation complexity of watermarking algorithms varies significantly with different embedding approaches. However, to achieve good resistance to noise, compression and other signal processing attacks, a common practice is to embed the watermark in the transform domain [1]. Some robust watermarking approaches also require spread spectrum analysis. Hence, the computational cost can be fairly high, which will also lead to high power consumption. Among the existing efforts of developing good watermarking system on low power devices, Arun Kejariwal at el made valuable contributions via evaluating a number of existing watermarking approaches in embedded environment. Their research provided a good perspective of the power consumption of various watermarking algorithms [2]. According to their experimental results, the Koch and Bruyn approach has the lowest power consumption and shortest execution time, especially for host images with high resolutions.

Takao Nakamura [3] described a fast and robust watermarking detection scheme on cellular phones. However, it only worked with 16-bit watermark and images with resolution 288*352. In 2011, J. Jeedella and H. Al-Ahmad [4] at Khalifa University of Science, Technology & Research proposed an algorithm for watermarking mobile phone color images using BCH code. This algorithm demonstrated good robust level through benchmark tests for attacks and the watermarked image had high PSNR. However, this method required the watermark to be in the format of numbers. Particularly, the implemented algorithm utilized cell-phone numbers as the watermark.

It is worthwhile to mention two available applications on Android platform for watermark detection, namely Digital Space [5] and Digimarc Discover [6]. These two applications are very similar and allow the user to hold the camera mounted on the smart-phone about 5-7" away from the image until cell-phone "bee" to finish the detection. After "Bee", the application will tell the user whether there is a watermark in the image. Users of these applications need to register online, embed watermarks into images, and save them in their accounts. Only watermark detection on the registered images is performed on a smart-phone.

## 3. BACKGROUND INFORMATION

### 3.1 Overview of Digital Watermarking

Digital watermarking is the process of embedding information into a digital signal, like audio, image, and video, which can be detected for authentication and identification. The embedded watermark can be number, characters, image, or any other identification information [1].

Digital watermarking systems can be categorized into different types. In terms of perceptibility, there are visible and imperceptible digital watermark. Since invisible watermarks are typically used for authentication and data integration, we only consider this type of watermarks in our research. In terms of robustness, there are three types of digital watermark, namely robust, semi-fragile, and fragile watermark. Robust watermark are widely used for copyright protection, while the other two are used for data integrity and authentication. Specifically, due to its ability to detect attacks as well as its good resistance to channel noise and compression, semi-fragile watermark has become a desirable approach for authentication.

The embedding process of digital watermark also varies a lot. In general, the watermark can be embedded in spatial domain, transform domain, or both. Embedding approaches involving transform domain analysis usually provide better resistance against compression. DCT (discrete cosine transformation) DWT (discrete wavelet transformation) are two widely used transformations in watermark embedding. Both have their own advantages and disadvantages. Since DCT is used in compression standards such as JPEG and MPEG, DCT domain embedding offers significant convenience for JPEG images, while the multi-resolution nature of DWT offers good means for spread-spectrum analysis and thus enhance the robustness of the embedded watermark, or provides support to localize the regions being attacked.

### 3.2 Smart-phone configuration requirements and constraints

To design a digital watermarking application on an Android smart-phone platform, it is important to take into accounts its hardware constraints. Table 1 lists features of three different Android smart-phones. Specifically, the screen resolution, the processor power, the memory size and the power efficiency are critical in a design. Ideally, a developed watermarking system on an Android platform should be fast and responsive, power efficient, uses less memory space, and provides seamless user experience to achieve the target security functions. To meet the above design goal, an appropriate watermarking algorithm needs to be selected that for implementation under hardware constraints listed in Table 1 [7].

## 4. DESIGN OF HYMNMARK

### 4.1 Watermarking algorithm selection

The first step of our research is to study existing watermarking algorithms and identify suitable algorithms with good performance-computation balance that can be implemented in low power devices. In comparison with characteristics of a number of algorithms, Koch's algorithm [8] has been selected as the baseline algorithms, due to its

lowest power consumption, shortest execution time, and greater robustness against common attacks.

The embedding process of Koch's algorithm can be described briefly as follows: First, DCT transformation is applied to the entire image. The next step is to generate position sequence that maps the watermark bit to the pixel locations. Next, Randomly Sequenced Pulse Position Modulated Code (RSPPMC) [8] will be embedded into the locations in image blocks that are selected in the first two steps. Lastly, inverse DCT and de-quantization is applied to the embedded blocks. Among all the steps, the largest computation power is spent in the RSPPMC embedding part.

There are some limitations of Koch's algorithm. First, it is non-blind watermark algorithm, that is, watermark cannot be detected without side information. In particular, detection process of Koch's algorithm needs a key file to indicate the location sequence and watermark length in order to detect watermark. Secondly, Koch's algorithm does not support multi-resolution images because of nature of the DCT transformation. Thirdly, the watermark length is also limited. Since only one bit is embedded into one 8*8 block of an image, the watermark length is bounded by the number of 8x8 blocks in a host image. As a result, modifications are required to further improve its performance. To the best of our knowledge, our work is the first effort to implement and test Koch's algorithm in Android platform. Therefore, design and implementation of the proposed watermarking application together with the evaluation results will provide useful insights and guidelines for future research in the area.

| | HTC Sensation 4G | Samsung Galaxy S II | HTC EVO 3D |
|---|---|---|---|
| Android version | Gingerbread (2.3) | Gingerbread (2.3) | Gingerbread (2.3) |
| Skin | Sense 3.0 | TouchWiz 4.0 | Sense 3.0 |
| US carrier | T-Mobile | TBD | Sprint |
| Display | 4.3-inch Super LCD | 4.3-inch Super AMOLED Plus | 4.3-inch Super LCD with glasses-free 3D |
| Resolution | 540 x 960 | 480 x 800 | 540 x 960 |
| Dual-core processor | 1.2GHz Qualcomm MSM8260 | 1.2GHz Samsung Exynos 4210* | 1.2GHz Qualcomm MSM8660 |
| RAM | 768MB | 1GB | 1GB |
| Storage | 1GB internal with 8GB MicroSD card | 16GB or 32GB internal with MicroSD slot | 4GB internal with MicroSD slot** |
| Front camera | VGA | 2 megapixel | 1.3 megapixel |
| Rear camera | 8 megapixel,dual LED flash | 8 megapixel,LED flash | 2x 5 megapixel, 3D images and video |
| Video | 1080p at 30fps | 1080p at 30fps | 1080p at 24fps (2D), 720p at 30fps (3D) |
| 4G internet | 14.4Mbps HSPA+ | 21.1Mbps HSPA+ | WiMAX |
| Accelerometer | Yes | Yes | Yes |
| Gyroscope | Yes | Yes | Yes |
| Battery | 1520mAh | 1650mAh | 1730mAh |
| Thickness | 11.3mm | 8.49mm | 12.1mm |
| Weight | 148g / 5.22 ounces | 116g / 4.09 ounces | 170g / 6 ounces |

Table.1 Three Typical Smart-phone Hardware Configurations

**4.2 hymnMark Architecture**

The hymnMark conceptual flow-chart is shown in Figure.2. In the top level hymnMark includes two processes: watermark embedding and detection. The embedding process includes four major functions: 1) import image; 2) select watermark; 3) RSPPMC embedding; and 4) save/export watermarked image. In our current implementation, host images can only be imported from local memory. A watermarked image can be saved to a file on the SDCard of a smart-phone. The detection process includes four essential functions: 1) import image; 2) import a key file; 3) watermark detection; 4) display the retrieved watermark.

**4.3 Implementation**

**4.3.1 Overview**

hymnMark is implemented as an app on Android operating system using JAVA with Android Plug-In. It has been tested on a real Android Phone NEXUS One with Android system 2.3.3. Its detailed configuration is: Android SDK 2.2, Eclipse jdk-6u24-linux-x64, eclipse-jee-helios-SR2-linux-gtk-x86_64, and Ubuntu 10.10 Linux.

**4.3.2 GUI Front-End**

hymnMark features a GUI-based front-end. It uses the popular MVC (Model-View-Control) framework for GUI applications. In this framework, user interface and the models, which are usually called "classes" in Java, are separately designed to cooperate with each other through controllers. Generally, an "XML" layout is a view interacting with users; a "Java" class is a model that will be called by a controller when needed; an "activity" is a "View and Controller", so as a "View", it corresponds with an XML-layout as a "controller".

**4.3.3 Digital Watermark Kernel**

Details of the watermark embedding and detection processes of Koch's algorithm are described as follows.

4.3.3.1 Embedding Process:

The embedding process takes three inputs, namely *Host image*, *selected Watermark*, and *JPEG Quality factor*, and produced the watermarked image. After the host image is loaded, color space transformation will be applied such that the image will be represented in YUV color space instead of RGB. In hymnMark, the digital watermark is only embedded in Y component for a true color host image.

After color space transform, block-based DCT is performed followed by JPEG-alike quantization to each 8x8 block of DCT coefficients. To embed RSPPMC, two DCT coefficients will be randomly selected in the low-middle frequency range per block while the MSE of the original block and embedded block meets the minimum requirement, which is $1e^{-3}$.

$$MSE = \frac{1}{mn}\sum_{y=1}^{m}\sum_{x=1}^{n}[I(x,y) - I'(x,y)]^2$$

where *I(x,y)* and *I'(x,y)* are the Y component values at the index of (x,y) of the original and the embedded images, respectively.

After the embedding process, de-quantization and inverse DCT are conducted to each block. Then, the embedded blocks are multiplexed to create the Y layer. YUV to RGB color space conversion will be conducted to get the watermarked image.

4.3.3.2 Detection Process:

The detection process takes two inputs, *host image* and *key*. Like the embedding process, the detection process also requires color space transformation, DCT transformation, and JPEG-like quantization prior to actual detection.. Based on the key stored in a file, essential information for watermark detection, such as location sequence, watermark length, watermark type, and quality factor, is obtained. The selected DC coefficients blocks and the coefficients in each block can be recovered in order of the "seed" from the key file. Then, inverse RSPPMC is conducted to each pair of coefficients in each block; the process is repeated until the full length of watermark is recovered. The result of inverse RSPPMC is a bitstream that is further converted to a text or an image with respect to the watermark type. The detected watermark can also be saved in a new file on the SD Card of a smart-phone.

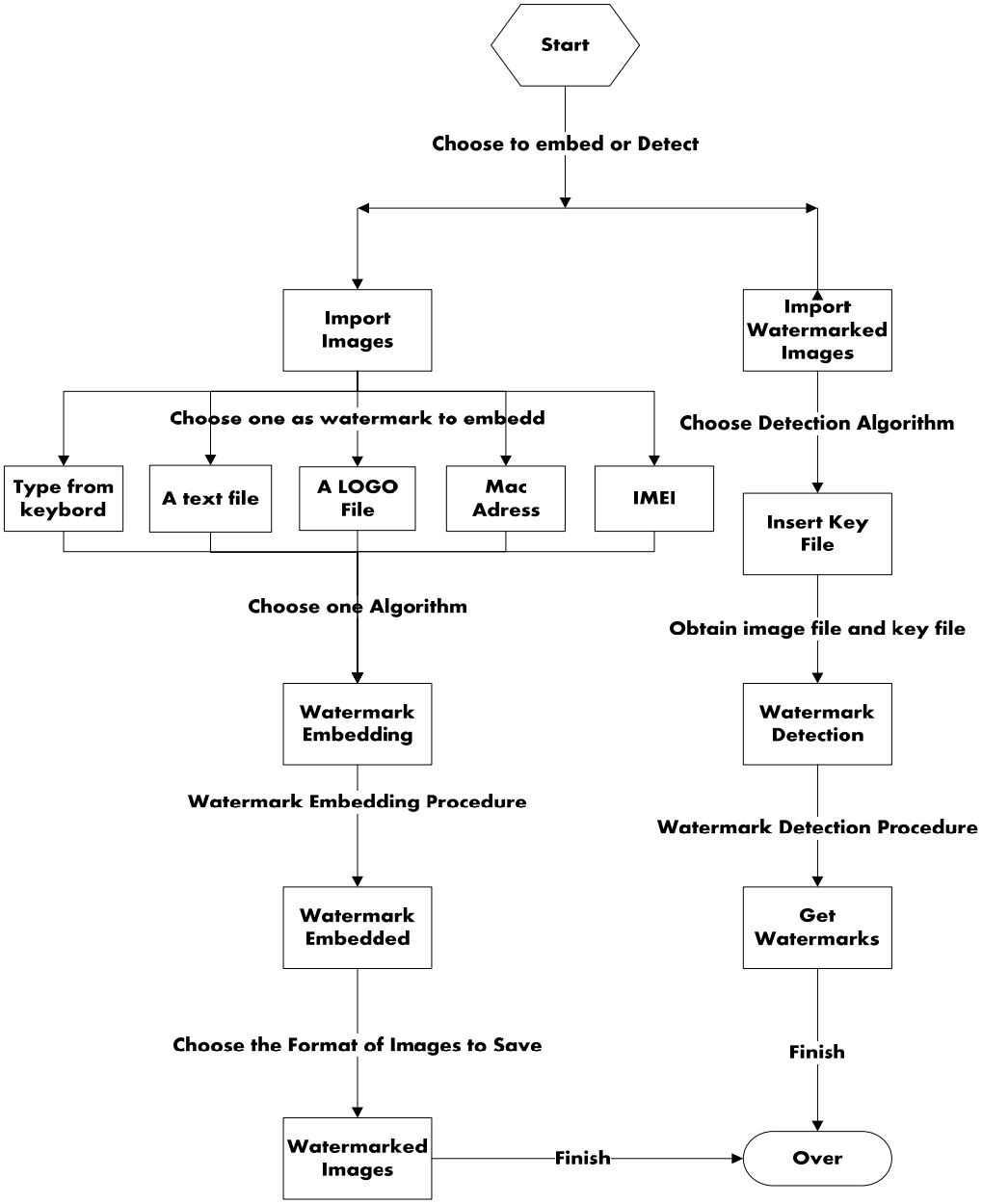
**4.4 Optimizations to the Koch Algorithm**

In order to reduce the computation cost, save power consumption, and to accommodate color images, the following modifications have been made in the implementation.

First, the order of DCT transformation and the selection of image blocks has been swapped. In the original Koch's algorithm, DCT transformation is applied to the entire picture, which is unnecessary since we only embed the watermark bits in a subset of DCT blocks. A disadvantage of the original Koch's algorithm is that if the image is big, the DCT transformation will cause large amount of computation. By switching the order, we only need to apply the DCT transformation to the selected 8*8 blocks. As a result, computation cost is reduced, and so does the power consumption.

Secondly, our implementation narrows down the range of pixel selection for watermark embedding within each block. Watermark bits are supposed to be embedded into the low-middle frequency of DC coefficients. In an 8*8 block, the random selection range could be shrank to {3,4,5,10,11,12,14,17,18,19,20,24,25,26,27,28,32,33,34,35, 40,41} instead of {0-63}. In the optimized implementation, 34.38% of computational power for DC coefficients selection is saved compared to the original Koch's Algorithm.

Thirdly, we add an image size adjustment approach after a host image is read from SD Card to avoid the out-of-memory (OOM) problem that is common for Android applications. To work with the memory constraints for an Android application (16MB), hymnMark automatically reduces the image size based on the device screen. For example, Android Nexus One's screen size is 600*480; then a host image of 2096*2096 will be resized to 480*480. In this way, we are able to control the memory usage of the application while handling multiple images




Figue.2 hymnMark Application User Flow-Chart

In addition to reducing computation, we have modified the algorithm to support color images. In particular, to achieve good perceptual performance, color space conversion is performed first and the watermark is embedded in Y components of a color image.

## 5 RESULTS AND ANALYSIS

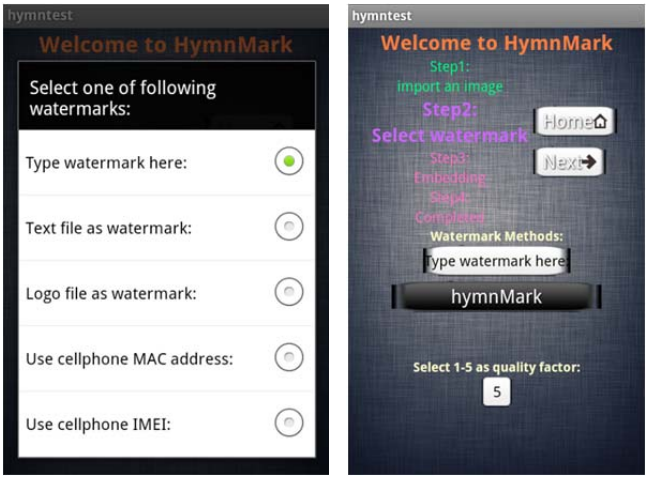
### 5. 1 User Interface

hymnMark has a user-friendly GUI. Figures 3 to 6 illustrate the key steps of using hymnMark to embed and detect watermark on a smart-phone.

As shown in Figure 3a, a user can click ImageView to import a host image. Five types of watermark are supported in hymnMark: plain text, a text file, a logo image file, the MAC address of the smart-phone, and the IMEI number of the smart-phone. A user can select the preferred watermark types through the GUI. In addition, a user can input quality factor (in range of 1 to 5) to indicate the watermarking robustness. Figure 3b shows that a user types text message *hymnMar*k as the watermark and selects 5 as the quality factor (the most robust). Figure.4a shows the embedding progress and the watermarked image after the embedding process is completed. If a user is satisfied with the

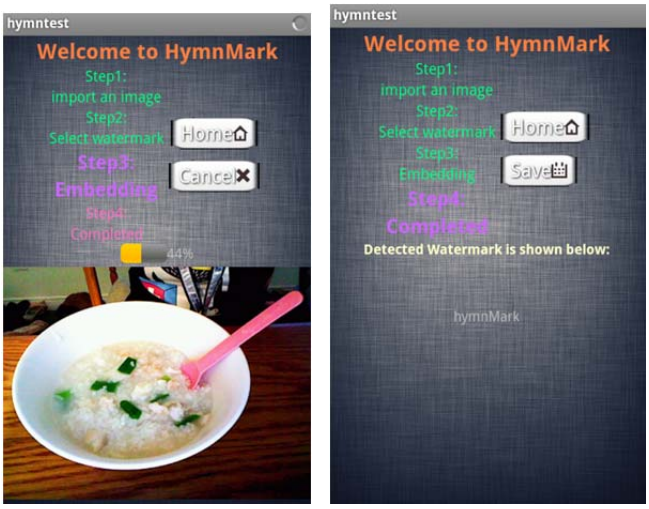
watermarked image , the user can click on “save” to save it onto the SD Card.

From the main page, if a user chooses “detect”, a detection page will show up. The embedded image will be displayed in the “ImageView” and the application will ask for a key file in order to detect the watermark in the image. After the key file is loaded, the detection process will start. Figure 4b displays the detected watermark. If the watermarked image has been severely damaged, the detection may not be successful, and the detected watermark may contain errors or may be illegible.




(a) (b)

Figure.3 hymnMark Interface illustration: a) Five types of watermark message Selection; b) Typing Message as Watermark




(a) (b)

Figure.4 hymnMark Interface illustration :a) Embedding Page; b) Detected watermark shown page

### 5.2 Performance Analysis

To evaluate the performance of the hymnMark app with respect to quality of watermarked images, a number of tests have been conducted. Table 2 lists the quality analysis results for host images of different sizes. It is obvious that if the watermark is short (1 bit), the impact is small and the resulted watermarked image has higher quality. Quality factor indicates the robustness of the watermark. To make the watermark more robust, it needs to be embedded into the DCT coefficients of lower frequency, which will consequently have more impact on the image quality. Our test results show that even for quality factor of 5, the watermarked image still has excellent quality (PSNR > 35 dB).

| Host Image Size | Watermark Length | Quality Factor | PSNR |
|---|---|---|---|
| 32kb | 1b | 1 | 50.55 |
| 32kb | 1b | 2 | 50.42 |
| 32kb | 1b | 3 | 50.27 |
| 32kb | 1b | 4 | 50.01 |
| 32kb | 1b | 5 | 49.97 |
| 200kb | 3.1kb | 1 | 37.47 |
| 200kb | 3.1kb | 2 | 37.33 |
| 200kb | 3.1kb | 3 | 37.11 |
| 200kb | 3.1kb | 4 | 37.03 |
| 200kb | 3.1kb | 5 | 36.87 |

Table.2 PSNR for Various Quality Factors

### 5.3 Robustness Analysis through Multiple Attacks

In [9], Johnson C. Lee analyzed the attacks on common watermark techniques. Following his analysis, we have evaluated the performance of hymnMark system under some common attacks including rotation, cropping, scaling, mosaic, Gaussian, contrast, chrominance, luminance and compression. All the attacks are executed through Adobe Photoshop CS4 version 11.0.

Table 3 summarizes the results of the robustness tests. In the table, *robust range* means that among the specified parameter settings, the watermark can be detected correctly. Take compression as an example, the max setting in Photoshop for users to modify is 0-12, which 0 stands for the worst quality, while 12 means the best quality. The watermarked images are tested in the Photoshop and be detected for the watermark message. When the compression range is in 4 to 12, the watermark can be detected completely. Therefore, its correspondent robust range is 4 to 12. The testing results show that hymnMark has no resistance towards attacks such as rotation, cropping and scaling, but has fairly good resistance against contrast change and compression.

| Attacks: | Robust Range | Max setting in PS |
|---|---|---|
| Rotation | None | 0 to 360 |
| Cropping | None | Any |
| Scaling | None | Any |
| Mosaic | None | 2 to 200 |
| Gaussian | 0.0 to 5 | 0.0 to 250 |
| Luminance | -25 to 25 | -50 to 150 |
| Chrominance | -10 to 10 | -180 to 180 |
| Contrast | -50 to 50 | -50 to 100 |
| Compression | 4 to 12 | 0 to 12 |

Table.3 Robustness Analysis

**5.4 Power Consumption Analysis**

The power consumption of hymnMark can be measured at two different levels, *macro-level* and *micro-level*. Macro-level measurement shows total power consumption of hymnMark compared to other applications on the smart-phone. Micro-level measurement, on the other hand, provides a close-up view of power profiling of hymnMark, which is capable of showing the hotspot of hymnMark, that is, where it consumes most power.

5.4.1 Smart-phone power model:

Efficient energy management requires good understanding of where and how power is consumed, including how much the whole system uses and how much each component uses. In [12], Aaron Carroll and Gernot Heiser tested energy consumption of CPU/RAM, screen display, GSM (Global System for Mobile Communication, originally from Group Special Mobile), flash storage, network and GPS through different applications. The results in Table.4 show that the majority power consumption is used in GSM module and screen display. In these experiments, it is easy to see that brightness of display is the most significant factor that affects the power consumption of a mobile device. , followed by the CPU power consumption.

On the other hand, smart-phones are considered as personal portable computers nowadays and their users expect fast-responsive time of apps. For example, people who get lost want to find a right direction fast when they ask for help from a map application on smart-phone.

| | Power (% of total) | | | | | | | Battery life |
|---|---|---|---|---|---|---|---|---|
| Workload | GSM | CPU | RAM | Graphics | LCD | Backlight | Rest | [hours] |
| Suspend | 45 | 19 | 4 | 13 | 1 | 0 | 19 | 49 |
| Casual | 47 | 16 | 4 | 12 | 2 | 3 | 16 | 40 |
| Regular | 44 | 14 | 4 | 14 | 4 | 7 | 13 | 27 |
| Business | 51 | 11 | 3 | 11 | 4 | 11 | 10 | 21 |
| PMD | 31 | 19 | 5 | 17 | 6 | 6 | 14 | 29 |

Table.4 Daily energy usage and battery life under a number of usage patterns [12]

5.4.2 Power Measurement Methodology

The most popular power consumption measurement tools are PowerTutor and PowerProfile. PowerTutor is an Android app working on Google phones that calculates the power consumption of CPU, display, Wi-Fi, and user applications running on the platform. To access the power consumption measurement, it uses Android inner resources such as:

- *android.content.Context;*
- *com.android.internal.util.XmlUtils;*
- *org.xmlpull.v1.XmlPullParser.*

However, PowerTutor and PowerProfile only calculate the power usage based on components not within each application. As a result, they fail to provide fine-grained measurement and insights on power consumptions of functions such as *read-image*, *read-watermark*, color space conversion, block selection, *DCT transformation*, *quantization*, and *embedding/detection*, *de-quantization*, *iDCT transformation*, *inverse color space conversion*, *store images/watermark.*.

Research described in [13] has developed a fine-grained power measurement tool called *Eprof* but it is not available to the community. As a result, we have developed the micro-level power consumption analysis method based on Android EXTRA_LEVEL and EXTRA_SCALE APIs EXTRA_LEVEL measures the current power level, and EXTRA_SCALE measure the maximum level of the smart-phone. The methodology details are described as follows:

1. Set up a test project to evaluate “EXRTA_LEVEL” and “EXTRA_SCALE” variables, and verify their functions;.
2. Once the verification is passed, the micro-level power consumption analysis should be conducted following the steps illustrated in Figure 9.

The current power level of the smart-phone is measured before and after the execution of each code block. This allows us to measure the power consumption of each part of the application; and we can also study the impact of application parameters (such as the quality factor) on the power usage. Therefore, the micro-level power consumption analysis is very useful to optimize the implementation of each part of the application under power constraint.

5.4.3 Power consumption of hymnMark

To analyze the power consumption of hymnMark in the macro-level, “PowerTutor”[10] is utilized. During our tests, we have found that the total power of the fully charged cell-phone, a Nexus one, was 1144mAh (4.12 Volt). The watermark embedding and detection process have been measured through continuous execution. On average, the embedding process consumes about 66.7mW power, and the detection process consumes 38.4mW. Hence, the average of the entire hymnMark is (66.7+38.4)/2=52.55mW.

Using the same experiment setting, we have also measured an Android default browser’s power consumption, which is around 282mW. In comparison, our developed hymnMark system consumed less power than a regular web browsing application.

Currently we are in the process of tuning the micro-level instrumentation, and we hope to report the preliminary results of the micro-level power analysis for hymnMark during the conference presentation.

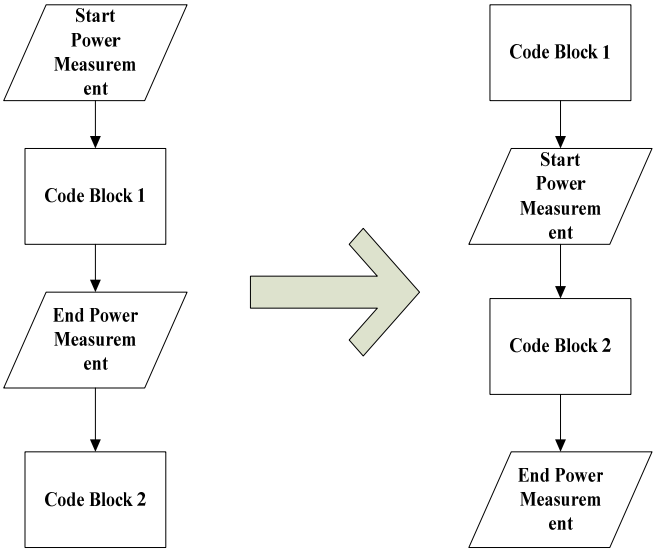



Figure.9 Procedure of Micro-level power consumption analysis

## 6. CONCLUSION AND FUTURE WORK

This paper presents the first Androi SmartPhone watermark app. The core of the system is the watermark embedding and detection processes based on Koch's algorithm. Optimizations have been made to reduce the computation cost and power consumption on the Android SmartPhone platform. Comprehensive testing has been conducted to evaluate quality, robustness, and power consumption of the implementation. Experimental results demonstrate that the watermarked images have excellent visual quality; and the power consumption is lower than a web browser app on a smart-phone platform. In the future, we will further optimize the algorithm to reduce the power consumption and execution time. More power consumption analysis will be conducted to gauge the energy efficiency of each internal function block.

## References

[1]. Watermarking digital image and video data, A-State-of-the-Art Overview, Gerhard C. Langelaar, Iwan Setyawan, and Reginald L. Lagendijk, IEEE image processing magazine, 2000

[2]. Energy Analysis of Multimedia Watermarking on Mobile Handheld Devices, Arun Kejariwal, Sumit Gupta, Alexandru Nicolau, Nikil Dutt, Rajesh Gupta, School of Information and Computer Science Tensilica Inc. Dept. of Computer Science and Engineering University of California at Irvine, IEEE Conference Publications,2005

[3]. A Fast and Robust Digital Watermark Detection Scheme for Cellular Phones, Takao Nakamura, Atsushi Katayama, Ryo Kitahara, and Kenji Nakazawa, NTT Cyber Space Laboratories Yokosuka-shi, 239-0847 Japan, 2006

[4]. An Algorithm For Watermarking Mobile Phone Color Images Using BCH Code, J. Jeedella and H. Al-Ahmad, Khalifa University of Science, Technology & Research, IEEE Conference Publications, 2011

[5]. Digital Space, https://play.google.com/store/apps/developer?id=Digital+Space

[6]. DigiMarc Discover, https://play.google.com/store/search?q=digimarc+discover&c=apps

[7]. Android System Smart-Phone Hardware Configuration. http://www.engadget.com/2011/04/15/htc-sensation-versus-the-rest-of-the-dual-core-world-smartphone/

[8]. Towards Robust and Hidden Image Copyright Labeling, E. Koch & J. Zhao, Fraunhofer Institute for Computer Graphics Wilhelminenstr. 7, 64283 Darmstadt, Germany, Proc. of 1995 IEEE Workshop on Nonlinear Signal and Image Processing (Neos Marmaras, Greece, June 20-22, 1995)

[9]. Analysis of Attacks on Common Watermarking Techniques, Johnson C. Lee, Student Member, IEEE Electrical and Computer Engineering Department University of British Columbia 2356 Main Mall, Vancouver, BC Canada V6T 1Z4, IEEE Conference Publications, 2011

[10]. Power Tutor, http://ziyang.eecs.umich.edu/projects/powertutor/

[11]. How Fast Is Your Mobile App? Gomez Knows, Charles Babcock, InformationWeek November 04, 2011.

[12]. An Analysis of Power Consumption in a Smartphone, Aaron Carroll, NICTA and University of New South Wales, Gernot Heiser NICTA, 2010 USENIX Annual Technical Conference, 2010.

[13] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang, *Where is the energy spent inside my app? Fine Grained Energy Accounting on Smartphones with Eprof* , Eurosys 2012.