Tradeoff between Energy Savings and Privacy Protection in Computation Offloading

Jibang Liu, Karthik Kumar, and Yung-Hsiang Lu
School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA
{liu285,kumar25,yunglu}@purdue.edu

ABSTRACT

Offloading can save energy on mobile systems for computation-intensive applications. The mobile systems send programs and data to grid-powered servers where computation is performed. Offloading, however, causes privacy concerns because sensitive data may be sent to servers. This paper investigates how to protect privacy in computation offloading. We use steganography to hide data before sending them to servers. This paper evaluates the tradeoff between energy savings and privacy protection for content-based image retrieval with different steganographic techniques. We implement these methods on a PDA and compare their energy consumption, performance, and effectiveness of protecting privacy.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; C.4 [**Performance of Systems**]: Design studies

General Terms

Algorithms, Design, Measurement, Performance

Keywords

computation offloading, privacy protection, energy saving, image retrieval, steganography.

1. INTRODUCTION

Computation offloading migrates computation from battery-powered mobile systems to grid-powered servers and can extend the former's battery life [1, 2, 3, 4]. Offloading sends data and programs to servers; hence, the servers may have private information about the mobile users. Previous studies focus on determining whether to offload [1, 3, 4] and what computation to offload [2]. Some studies investigate how to protect privacy in data outsourcing [5, 6] but no study is devoted to evaluating the energy consumption in protecting offloaded data from mobile systems.

ISLPED'10, August 18–20, 2010, Austin, Texas, USA.

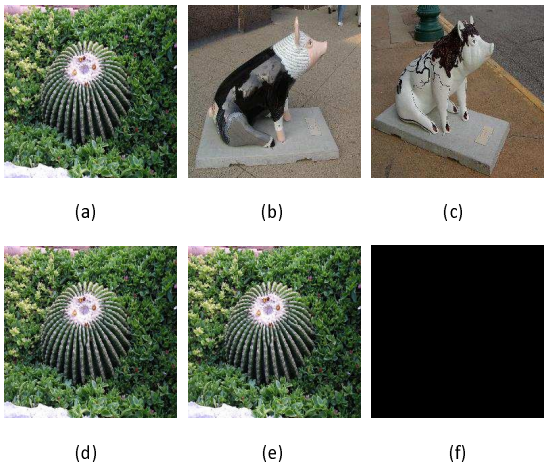

Figure 1: Two examples of steganography. (a) is the cover image. (b), (c) are hidden in (a) and their corresponding stego images are (d) and (e). (f) is the difference of (d) and (e).

Encryption is commonly used when transmitting confidential data through insecure channels. Encryption, unfortunately, is inapplicable in computation offloading because data are decrypted by the server and the server has complete knowledge about the data. To protect privacy from unauthorized use by the server, we may adopt a different technique called *steganography* [7, 8, 9]. Steganography *hides* data so that the server is unaware of the existence of information. Image processing is computation-intensive and a good candidate for offloading. Figure 1 shows two examples of steganography. A *cover image* is used to disguise the *data image* so that the data image is hard to recognize. The combined image is called a *stego image*. Many steganographic techniques have been proposed; Ella shows that "blind steganalysis" is difficult [10]. In other words, a server cannot easily detect hidden data if the server does not know what steganographic technique is adopted. A key challenge is to allow offloaded computation to be performed on steganographic data because the computation must remain meaningful on stego images. Suppose we want to compare the images in Figure 1 (b) and (c), Figure 1 (d) and (e) are sent to the server instead. Figure 1 (f) shows the pixel-wise difference between (d) and (e). Since the cover image is never sent to server, the server cannot detect hidden data

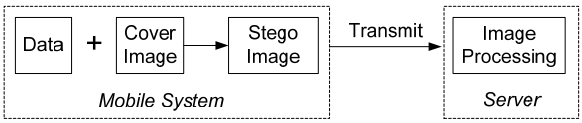

Figure 2: Offloading image computation protected by steganography.

by subtracting the cover image from the stego images. The difference of two stego images in (f) does not reveal the hidden data in (b) or (c). Even though traditional techniques can provide high embedding capacity and stego image quality, hidden data cannot be directly processed before being extracted from the stego images. We have to find hiding techniques that allow processing hidden data without extracting them to protect privacy.

This paper examines the energy consumption of hiding data through steganography for offloading image processing from mobile systems to servers. Before images are sent to servers, they are hidden inside a cover image. The created stego images are then transmitted to a server for processing, as shown in Figure 2. This paper describes the requirements of hiding techniques such that meaningful processing can be performed on protected data. We demonstrate that content-based image retrieval (CBIR) can benefit from offloading because similarities can be maintained in stego images. Different steganographic techniques protect privacy to different degrees and also consume different amounts of energy on the mobile system. This paper evaluates different steganographic techniques implemented on an HP iPAQ and compares their energy consumption, retrieval performance, and hiding quality. We believe this is the first paper considering the tradeoff of energy savings and privacy protection for computation offloading.

2. RELATED WORK

There have been many studies on computation offloading to save energy. Rong and Pedram [1] use stochastic models to determine whether to offload. Li et al. [2] use cost graphs generated from program analysis to decide which parts to offload. Wolski et al. [3] and Hong et al. [4] decide whether to offload based on network bandwidths. Xian et al. [11] use timeout: if a program runs longer than a threshold value, the computation is offloaded. Kumar et al. [12] consider offloading as a service and detect whether servers perform the computation as promised. Previous studies focus on determining whether to offload and what computation to offload without protecting privacy. Content-based image retrieval (CBIR) [13] finds images similar to a query by extracting image features and comparing the features. ImgSeek [14] is an open-source CBIR program; it performs two-dimensional Haar wavelet decompositions on the images and the coefficients with large magnitudes are used as features. Lu et al. [15] protect privacy by extracting image features and encrypting the features. However, extracting features is computation-intensive and should be performed on servers [4].

Steganography [7, 8, 9] hides data so that they are difficult to detect. Image steganography is commonly applied in spatial or frequency domain. In the spatial domain, a widely known algorithm embeds data in the least significant bit (LSB) [7]. In the frequency domain, data are hidden in the transformation coefficients. Wu et al. [8] propose a method based on pixel value differencing (PVD) to improve embedding capacity by hiding more data in edges and less data in smooth regions. Most of the existing methods can improve capacity and stego image quality. However, these methods are inapplicable to offloading because data are hidden irregularly and the data must be extracted *before* further processing.

Visual inspection and statistical steganalysis are often used to detect hidden data. The invisibility of a steganography algorithm is the first and foremost requirement [16]. If a stego image shows artifacts, people suspect secret data are hidden. To prevent suspicion, a stego image should have a high peak signal-to-noise ratio (PSNR). Steganograpy often leaves "signatures" in the stego images and they can be detected by RS steganalysis [17]. RS steganalysis classifies all pixels of a stego image into three groups using masks M and $-M$: (1) the regular group (R_M and R_{-M}), (2) singular group (S_M and S_{-M}), and (3) unusable group. For an image without hidden data, $R_M \cong R_{-M}$ and $S_M \cong S_{-M}$ are satisfied. When data are hidden, the values ($R_{-M} - R_M$) and (S_M - S_{-M}) increase with the amounts of hidden data. If such results are detected, one may suspect that the image has hidden data.

This paper hides images before sending them to a server. The program executed at the sever never explicitly extracts the hidden data and all bits are treated uniformly; hence, the server cannot easily detect the hidden data. Our methods can save mobile systems' energy by offloading without losing privacy because hidden data can resist detection.

3. PRIVACY-PRESERVING COMPUTATION OFFLOADING

As shown in Figure 2, before sending the data to the server, the images are processed using steganography. The stego images are sent to the server for further processing. The adopted protection techniques must ensure the computation performed at the server remains meaningful. Meanwhile, the hidden data must be difficult for the server to detect. This section first provides a general view of data protection for offloading. We then characterize energy consumption models for offloading and analyze the requirements of data protection techniques. We propose a block-based steganographic method to hide data for offloading image retrieval. Our method can tradeoff between energy consumption and privacy protection by choosing different parameters in data hiding.

3.1 A General Framework on Data Protecting for Offloading

Computation offloading may save energy on mobile systems. However, before offloading computation, we must adopt protection schemes for privacy. To protect data, different protection schemes can be used in different types of programs. For example, encryption can be used for protecting important data in a storage service. Watermarking can be used to prevent unauthorized changes on spreadsheet. Steganography can be used to protect images for image retrieval. A general approach integrating different protection approaches is described in Figure 3 and the symbols are defined in Table 1. Without protection, original data D are

sent to the server directly. The result R is obtained with program C and this result is returned to the mobile system. With protection, original data D are first protected by protection scheme P. The protected data D' are sent to the server and processed by the program C' to generate the result R'. This result R' is returned to the mobile system and finally the result R'' is produced using the inverse protection P^{-1}. To process the protected data, some modification to the program may be required. Hence, the program C' is possibly different from C. The final result R'' must be acceptable compared with R. Ideally, R'' is the same as R. However, due to various reasons, they may be different; as a result, the quality of the program may be degraded when data are protected.

Table 1: Symbols and their definitions.

D : original data	C : program
R : intended result	P : protection scheme
D' : protected data	δ : protection key
C' : program using D'	R' : result by using D'
P^{-1} : protection inverse	R'' : obtained result

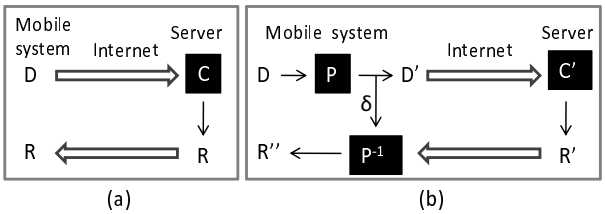

Figure 3: (a) Offloading a program without protection. Original data D are sent to the server directly. The result R is sent back to the mobile system. (b) Offloading a program with protection. The data are protected by P and D' are sent to the server. R' is returned and produces the final result R''

3.2 Energy Consumption Model

We analyze energy consumption in three scenarios: (1) running a program without offloading, (2) offloading a program and data without protection, and (3) offloading with protection. Table 2 lists the symbols and their meanings. The energy consumption at the mobile system includes three parts: computation energy, idle energy waiting for the results from the server, and transmission energy. We use $comp(.)$ to denote the amount of computation and $size(.)$ to denote the size of transmitted data. In scenario (1), the program is entirely performed at the mobile system. The energy consumption is:

$$P_c \times \frac{comp(C)}{U} \tag{1}$$

In scenario (2) as shown in Figure 3 (a), data D are offloaded and the program C is performed at the server. The total energy includes idle energy when the server runs C, and transmission energy for sending data D and receiving R:

$$P_l \times \frac{comp(C)}{S} + P_t \times \frac{size(D+R)}{B} \tag{2}$$

In scenario (3) as shown in Figure 3 (b), data D are protected by P; D' are offloaded and C' is performed at the server; R' is returned to the mobile system and processed by the inverse protection P^{-1}. Hence the total energy consumption includes computation energy for performing P and P^{-1}, idle energy when the server runs C', and transmission energy for sending data D' and receiving R':

$$P_c \times \frac{comp(P+P^{-1})}{U} + P_l \times \frac{comp(C')}{S} + P_t \times \frac{size(D'+R')}{B} \tag{3}$$

Offloading saves energy if (1) is larger than (3). To save energy, we have to find P and P^{-1} that do not require excessive amounts of computation, and the sizes of transmitted data D' and R' are small. Meanwhile, R'' must be sufficiently close to R.

Table 2: Parameters for offloading

P_c	Computation power	U	Speed of mobile system
P_t	Network power	S	Speed of server
P_l	Idle power	B	Bandwidth of network

3.3 Data Protection in Image Processing

We have to choose protection techniques P so that meaningful processing can still be performed on protected data D'. In a program C, a processing procedure can be considered as a transformation T performed on data D with the property of Y: $Y(T(D)) = \text{true}$. A desirable protection technique P on D first transforms the data D to D'. After applying the original processing T, the same property is still satisfied: $Y(T(D')) = \text{true}$. This property does not always hold. A simple example is intensity histogram of images. Let P be steganographic techniques; $T(D)$ be the histogram of image D; $T(D')$ be the histogram of the steganographic image D'. Figure 4 shows the histograms in one color channel of Figure 1 (b) and (d). The histograms change substantially. We can make Y as "the peak of a histogram is between 100 and 150." $Y(T(D))$ is true and $Y(T(D'))$ is false. This example suggests that choosing a suitable protection technique P for a specific transformation T is non-trivial.

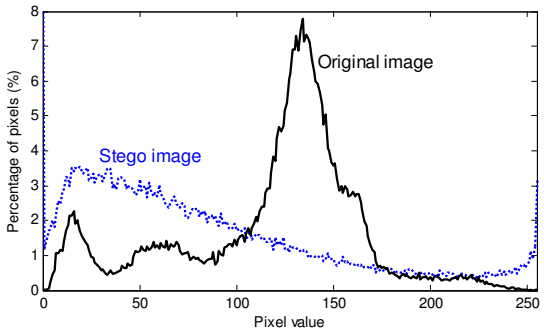

Figure 4: Intensity histograms of an original image and its stego image. The original image is Figure 1 (b) and the stego image is Figure 1 (d).

In traditional steganographic techniques, images are hidden in a cover image and meaningful operations may not be directly performed on the stego images. The hidden data are extracted from stego images before applying additional processing. However, the offloaded program cannot explicitly extract the hidden data at the server; otherwise, the hid-

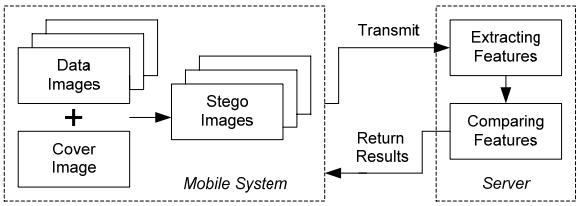

Figure 5: CBIR offloading flow.

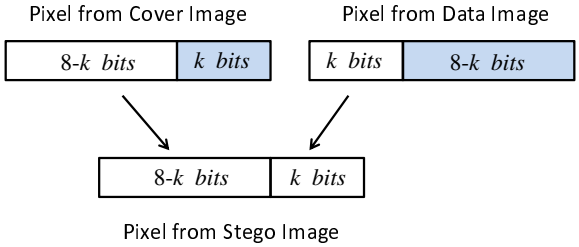

Figure 6: Hiding image by substituting k LSBs of the cover for k MSBs of the data image.

den data are revealed at the server. We have to find hiding techniques P that allow processing at the server.

3.4 Data Hiding Method

Content-based image retrieval (CBIR) converts images into numerical representations called features. The features are compared to find similar images. Figure 5 shows our procedure for offloading CBIR: (1) Original images D are hidden in a cover image and their stego images D' are created. (2) Stego images D' are transmitted to the server. (3) Features of the stego images are extracted at the server. (4) Features are compared and similar images are identified. (5) Finally, the indexes of similar images R' are returned. This procedure allows the mobile system to achieve results directly without performing any inverse protection P^{-1}. The server can access only the stego images and thus the original images are protected. We use ImgSeek based CBIR as the program C. ImgSeek ranks image similarities by first finding features with large magnitudes. However, in stego images, the features' magnitude is dominated by the cover image instead of the hidden images. Extracting features based on magnitude is not effective for steganographic images any more. In our program C', we compute the distance of features as the metric to compare two images. Let $f(D)$ be the features of image D. Transformation T computes the similarity between two images based on distance of their features: $T(D_1, D_2) = |f(D_1) - f(D_2)|$. Suppose D_1, D_2 and D_3 are three images. D_1 and D_2 are similar; D_1 and D_3 are dissimilar. This property Y can be expressed as $|f(D_1) - f(D_2)| < |f(D_1) - f(D_3)|$, namely $T(D_1, D_2) < T(D_1, D_3)$. Our goal is to find a steganographic technique P such that the property can be maintained with following condition:

$$\begin{array}{l} |f(D_1) - f(D_2)| < |f(D_1) - f(D_3)| \Rightarrow \\ |f(D_1') - f(D_2')| < |f(D_1') - f(D_3')|. \end{array} \quad (4)$$

Figure 6 shows a simple and commonly used steganographic technique. Suppose one pixel has 8 bits. The stego image is created by substituting the k LSBs of every pixel in the cover by the k MSBs of every pixel in the data image. The data image is approximately represented by its k MSB planes. Let x be the value of $(8-k)$ MSBs from cover image and y be the value of k MSBs from data image. The value of this pixel in the stego image is $x \cdot 2^k + y$. This technique uses a cover image with the same resolution as the data image. ImgSeek extracts features using Haar decomposition which satisfies linear property, so the similarity of two stego images can be expressed as:

$$\begin{array}{l} T(D_1', D_2') = |f(D_1') - f(D_2')| \\ = |[f(x \cdot 2^k + y_1)] - [f(x \cdot 2^k + y_2)]| \\ = |f(y_1) - f(y_2)| = T(y_1, y_2). \end{array} \quad (5)$$

The term $f(x \cdot 2^k)$ is canceled because we use the same cover image. The distance of two data images is approximately represented by the distance of the k MSBs of the pixels. The accuracy depends on k's value. A larger k improves the accuracy. When $k = 8$, the accuracy is 100%; i.e. no steganography is used. However, as k increases, the method causes serious image degradation with maximum error of one pixel up to $2^k - 1$.

We improve the qualities of stego images by hiding data in one *block* instead of only one pixel and thus reducing errors. The cover image is divided into non-overlapping blocks, each with $m \times n$ pixels; both m and n are powers of 2. The k MSBs in a pixel from the data image is partitioned into $m \times n$ parts and each part is hidden in one pixel of the block. In order to maintain quality, y is *evenly* distributed among the $m \times n$ pixels in the block. The maximum difference among the pixels in the same block is only one. Every pixel contributes either $\lfloor \frac{y}{mn} \rfloor$ or $\lceil \frac{y}{mn} \rceil$. For example, if y is 13 and a 2×2 block is used, 13 is partitioned into $13 = 4+3+3+3$. Three of the pixels are assigned value 3 and one pixel is assigned value 4. Then, the $8 - k$ MSBs from the cover image are prepended. The maximum change from the cover image is $\lceil \frac{2^k - 1}{mn} \rceil$. It can be proved that hiding data in one pixel and in one block can obtain the same coefficients, thus this block hiding method still satisfies the requirement in Equation (5). The proof is omitted here due to the page limit. Figure 7 shows four ways of hiding 13 with four different block sizes: 1×2, 2×2, 2×4, and 4×4.

3.5 Tradeoff between Privacy and Energy

There is a tradeoff between privacy protection and energy consumption when we use different block sizes. A larger block has higher hiding quality because the hidden data are more difficult to detect. A larger cover image is needed so more energy is consumed for creating the stego image; also, a larger stego image is transmitted to the server and more energy is consumed. Conversely, a smaller block consumed less energy for hiding and transmission but the hidden data are easier to detect.

4. EXPERIMENTAL RESULTS

We implement different hiding techniques in $C\#$ on an HP iPAQ 6954 PDA as the mobile system and a computer with 2GHz CPU and 3GB memory as the server. In order to measure the power consumption on PDA, we connect a 0.25Ω resistor to its battery and use a National Instruments data acquisition card to read the voltages at a sampling frequency of 10 kHz. We compare the following methods:

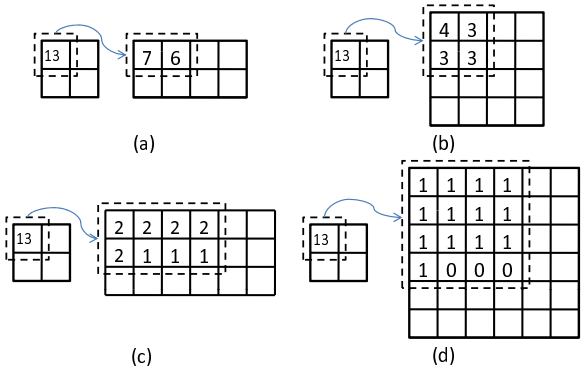

Figure 7: Hide value 13 in three different block sizes: (a) 1×2, (b) 2×2, (b) 2×4, and (d) 4×4.

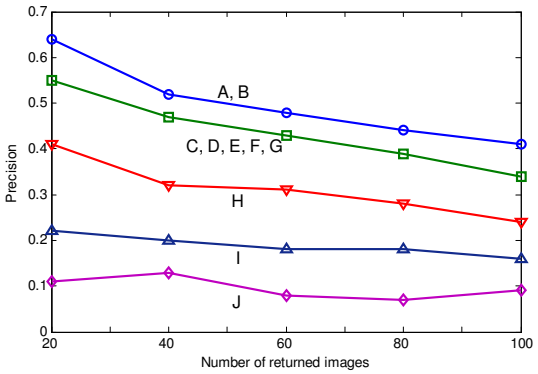

Figure 8: Retrieval accuracy comparison. A higher curve indicates better retrieval accuracy.

(A) Running CBIR entirely on the mobile system without hiding nor offloading.

(B) Offloading CBIR to the server without hiding images.

(C) Replacing LSBs shown in Figure 6, $k = 4$.

(D) Using 1×2 block shown in Figure 7(a).

(E) Using 2×2 block shown in Figure 7(b).

(F) Using 2×4 block shown in Figure 7(c).

(G) Using 4×4 block shown in Figure 7(d).

(H) Replacing LSBs shown in Figure 6, $k = 3$.

(I) Replacing LSBs shown in Figure 6, $k = 2$.

(J) Pixel value differencing presented in [8].

We do not consider extracting features at mobile system and send features to server for comparing. Extracting features requires much more computation than comparison [4]; hence, this method provides negligible energy savings at the mobile system. The test images are obtained from the Corel dataset [18]. These images include 10 categories and each category includes 100 images.

4.1 Retrieval Accuracy

The retrieval accuracy is evaluated through comparing the precision with different numbers of returned images. Suppose t images are returned. Among the t images, if r images belong to the same category as the query image, *precision* is r/t. A high precision indicates a better matching algorithm. From every category we select 10 images as the query and apply all methods for comparison. Figure 8 shows the retrieval accuracy comparison of methods (A)-(J). Precision is calculated when the numbers of returned images are 20, 40, 60, 80, and 100. (A) and (B) represent the original implementation of CBIR; they have higher precision since data images are processed directly. Other methods involve hiding; the accuracy is reduced because of the loss of details in the LSBs of hidden images. In (C)-(G), $k = 4$; they can obtain above 80% accuracy if normalized to (A) and (B). (C)-(G) have the same precision, because they generate the same Haar coefficients for comparing similarity. Though their cover images have different sizes, the retrieval results are independent of cover images as discussed in Equation (5). In (H) and (I), the accuracy is significantly lower. (J) hides data irregularly and thus the similarity properties are not preserved. Due to the low accuracy of (H), (I), and (J), they will not be further considered in the following analysis.

4.2 Privacy Protection

The degree of privacy protection is evaluated using PSNR and RS steganalysis. A higher PSNR is preferred because such a stego image makes hidden data difficult to detect by visual inspection. We hide every image into the same cover through different methods. The average PSNRs of (C)-(G) are shown in Table 3. From the table, hiding data in one pixel (C) provides the lowest quality and hiding in one block can improve PSNR values. PSNR values increase with the block size from (C) to (G), because a larger block can reduce hiding errors than a smaller block.

Table 3: Average PSNR values with different hiding methods. Higher PSNRs are better.

Method	Average PSNR (dB)
(C)	32.0
(D)	36.2
(E)	39.1
(F)	44.3
(G)	50.2

To evaluate the stego images using the RS steganalysis, we calculate $(R_{-M} - R_M)$ and $(S_M - S_{-M})$ with masks $M = [0\ 1\ 1\ 0]$ and $-M = [0\ -1\ -1\ 0]$ for the 1000 stego images. The results show that in our methods $(R_{-M} - R_M)$ and $(S_M$ - $S_{-M})$ are both in the range of (-0.1, 0.1). As discussed in [19], RS steganalysis cannot conclude there are hidden data in the stego images.

Considering both PSNR metric and RS analysis, the method based on a larger block can provide a better hiding quality and thus better privacy protection.

4.3 Energy Consumption

We compare the energy consumption in (A)-(G). (A) runs entire image retrieval on PDA; (B) offloads the program directly without steganography; (C)-(G) offload the program with protection by steganography. Energy consumption in (A) is the baseline for comparison. The values of P_c, P_t, and P_l in our measurement are about 814 mW, 1110 mW,

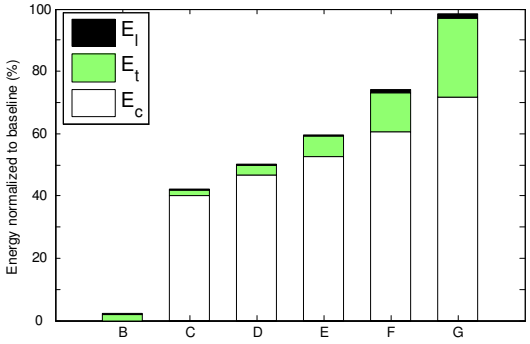

Figure 9: Energy normalized to baseline for different offloading methods. E_l, E_t and E_c are idle energy, transmission energy, and computation transmission energy. A higher bar indicates more energy consumption.

and 266 mW. We measure computation energy, idle energy, and transmission energy. The results are shown in Figure 9. The amounts of energy in (B)-(G) are normalized to (A).

In all these methods, energy consumption is less than 100% of the baseline, which means they all save energy by offloading. (B) consumes the least amount of energy, but it does not protect privacy. (C)-(G) consume more energy than (b) due to data protection. In (C)-(G), the computation energy E_c accounts for the most part of total energy consumption, because hiding data is run on PDA. The idle energy E_l is nearly negligible because the server is much faster than the PDA, resulting in short idle time. From (C) to (G), the energy E_c, E_l, and E_t all increase as the block size increases. The total amount of energy consumption increases from about 43% in (C) to about 98% in (D). The method based on a larger block always results in more energy consumption.

4.4 Tradeoff Analysis

We can choose different block sizes as a tradeoff between protecting privacy and saving energy. (A) performs CBIR without offloading and consumes more energy than all the other methods. However, it provides the best protection of privacy because the images are always on the mobile system. (B) offloads CBIR directly without data hiding and consumes the least amount of energy, but it does not provide any protection. (C)-(G) hide images to protect privacy before offloading CBIR. They achieve different degrees of privacy protection and consume different amounts of energy by choosing different block sizes. With a larger block, privacy is better protected but more energy is consumed. Conversely, hiding data in a smaller block degrades stego image quality and increases the risk of losing privacy; it also reduces energy consumption.

5. CONCLUSION

We present a steganographic technique to protect privacy in computation offloading for retrieving similar images. The technique uses a block-based method to hide data and then offloads computation to the server. We implement our method on a PDA and the results show that our method can tradeoff between energy savings and privacy protection by choosing different block sizes to hide data.

6. ACKNOWLEDGMENTS

This work is supported in part by NSF grants CNS-0716271 and CNS-0541267. Any opinions, findings, and conclusions or recommendations in the paper are those of the authors and do no necessarily reflect the views of the sponsor.

7. REFERENCES

[1] Rong and Pedram. Extending the Lifetime of a Network of Battery-Powered Mobile Devices by Remote Processing: a Markovian Decision-Based Approach. In *DAC*, pages 906–911, 2003.

[2] Li et al. Computation Offloading to Save Energy on Handheld Devices: a Partition Scheme. In *CASES*, pages 238–246, 2001.

[3] Wolski et al. Using Bandwidth Data to Make Computation Offloading Decisions. In *IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, 2008.

[4] Hong et al. Energy Efficient Content-Based Image Retrieval for Mobile Systems. In *ISCAS*, pages 1673–1676, 2009.

[5] S. Pearson. Taking Account of Privacy When Designing Cloud Computing Services. In *Workshop on Software Engineering Challenges of Cloud Computing*, pages 44–52, 2009.

[6] De Capitani di Vimercati et al. Preserving Confidentiality of Security Policies in Data Outsourcing. In *ACM Workshop on Privacy in the Electronic Society*, pages 75–84, 2008.

[7] Chandramouli et al. Image Steganography and Steganalysis: Concepts and Practice. In *Digital Watermarking*, pages 35–49, 2004.

[8] Wu and Tsai. A Steganographic Method for Images by Pixel-value Differencing. In *Pattern Recognition Letters*, pages 1613–1626, 2003.

[9] Nguyen et al. Multi Bit Plane Image Steganography. In *International Workshop on Digital Watermarking*, pages 61–70, 2006.

[10] W. Ella. Detecting Steganography on a Large Scale. In *Crossroads*, volume 15, pages 3–6, 2008.

[11] Xian et al. Adaptive Computation Offloading for Energy Conservation on Battery-powered Systems. In *ICPADS*, pages 1–8, 2007.

[12] Kumar et al. Ranking Servers Based on Energy Savings for Computation Offloading. In *ISLPED*, pages 267–272, 2009.

[13] Datta et al. Image Retrieval: Ideas, Influences, and Trends of the New Age. In *ACM Computing Surveys*, pages 34–94, 2008.

[14] Jacobs et al. Fast Multiresolution Image Querying. In *International Conference on Computer Graphics and Interactive Techniques.*, pages 277–286, 1995.

[15] Lu et al. Secure Image Retrieval through Feature Protection. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1533–1536, 2009.

[16] Morkel et al. An Overview of Image Steganography. In *Annual Information Security South Africa Conference*, pages 1–12, 2005.

[17] Fridrich et al. Detecting LSB Steganography in Color and Gray-scale Images. In *IEEE Multimedia*, volume 8, pages 22–28, 2001.

[18] Corel test set. Available online: http://wang.ist.psu.edu/ jwang/test1.tar.

[19] Zhang et al. Steganography with Least Histogram Abnormality. In *Computer Network Security*, pages 395–406, 2003.