https://doi.org/10.2352/ISSN.2470-1173.2019.5.MWSF-536

StegoAppDB: a Steganography Apps Forensics Image Database

***Jennifer Newman*[+], *Li Lin*[+], *Wenhao Chen*[†], *Stephanie Reinders*[+], *Yangxiao Wang*[▷], *Min Wu*[⋆], *and Yong Guan*[†]**
[+] ***Department of Mathematics, Iowa State University, Ames, Iowa, USA***
[†] ***Department of Electrical & Computer Engineering, Iowa State University, Ames, Iowa, USA***
[▷] ***Department of Computer Science and Software Engineering, University of Washington Bothell, Bothell, Washington, USA***
[⋆] ***Department of Electrical & Computer, University of Maryland, College Park, MD, USA***

Abstract

In this paper, we present a new reference dataset simulating digital evidence for image (photographic) steganography. Steganography detection is a digital image forensic topic that is relatively unknown in practical forensics, although stego app use in the wild is on the rise. This paper introduces the first database consisting of mobile phone photographs and stego images produced from mobile stego apps, including a rich set of side information, offering simulated digital evidence. StegoAppDB, a steganography apps forensics image database, contains over 810,000 innocent and stego images using a minimum of 10 different phone models from 24 distinct devices, with detailed provenanced data comprising a wide range of ISO and exposure settings, EXIF data, message information, embedding rates, etc. We develop a camera app, Cameraw, specifically for data acquisition, with multiple images per scene, saving simultaneously in both DNG and high-quality JPEG formats. Stego images are created from these original images using selected mobile stego apps through a careful process of reverse engineering. StegoAppDB contains cover-stego image pairs including for apps that resize the stego dimensions. We retain the original devices and continue to enlarge the database, and encourage the image forensics community to use StegoAppDB. While designed for steganography, we discuss uses of this publicly available database to other digital image forensic topics.

Introduction

Creating effective tools for forensic practitioners requires that developers have access to standardized sets of data, as recognized by the National Academy of Sciences [31]. As forensic processing of digital photographs becomes an increasingly important part of criminal investigations, the young field of digital image forensics must cultivate carefully designed and populated datasets. One area of digital image forensics is steganalysis, the analysis of a photograph for hidden content. Image steganography is the process to hide a *message* or *payload* in an *innocent* image, producing a *stego* image. See [39] or [25] for an introduction to steganography.

Stego apps on mobile devices are popular, easy to use, and stealthy. Table 1 displays the number of installs for 6 of over 100 apps that are available today. Development of techniques to discover steganography content where the image is "in the wild," that is, representative of what a practitioner may see while investigating a forensic case, is quite different from detection of a stego image in a controlled academic setting. We use "in the wild" to describe unconstrained scenarios, involving many apps, many different source devices, and unknown processing to images such as photo editing. Access to simulated digital evidence images can permit benchmark of current steg detection software and advance improved solutions, as well as introduce academic researchers to unanticipated questions. Until StegoAppDB, no current image database provided data that is reflective of that found in mobile stego cases.

Table 1: Real-world stego apps used for cover-stego image generation.

App Name	Platform	# Installs	Embedding Domain	Open Source
PixelKnot	Android	100,000+	JPEG	Yes
Steganography_M	Android	10,000+	Spatial	No
Pocket Stego	Android	1,000+	Spatial	No
MobiStego	Android	1,000+	Spatial	Yes
Passlok Privacy	Android	1,000+	JPEG	Yes
Pictograph	iOS	-	Spatial	Yes

While image datasets have been used successfully for benchmarking academic steganalysis algorithms [4], [20], [17], we identify some drawbacks for their use in benchmarking steganalysis tools on data closer to "in the wild" as encountered by forensic practitioners. For example, the commercial software StegoHunt [38] advertises capabilities to analyze image data. However, we know of no publicly available datasets containing stego images on which to benchmark performance errors. Further, as the use of stego apps on mobile devices becomes more prevalent, large datasets containing examples of images from these sources will provide benchmarking capabilities for current and future software, allowing more realistic detection of steganography "in the wild." Thus, creation of datasets that addresses some of these shortcomings is a welcome addition to the forensic community. Tellingly, the prevalence of steganography use occuring in forensic settings is unknown: there is no existing software designed for steg detection on suspect imags from a mobile device, nor any studies published detailing the population use rate of steganography, to the authors' knowledge.

If a forensic practitioner would like to test unknown images for steganograpy, she appears to have limited choices. First, three off-the-shelf software packages−StegoHunt [38], DC3 StegDetect [18] and Provos' StegDetect [33]−cannot detect, with any reliability, stego images produced with recent stego algorithms [15]. Second, if a forensic image analyst would like to develop or benchmark any new steg detection software beyond what these are designed to detect, there are no publicly available datasets (at least known to us). Other steg detection approaches can be used, such as searching for evidence of auxiliary installation files on the computer [40], which is proposed as field triage.

We observe that when faced with a similar situation, the face recognition community turned to unconstrained datasets that challenged solutions beyond constrained datasets, such as drivers licenses [32], [9]. In a similar manner, by using data sets that foster detection challenges with "in the wild" image data, the steganalysis community can pursue questions that are applicable to real-life scenarios, such as the transfer-learning problem of cover-source mismatch [24].

Motivated by this challenge, we propose that a database satisfy the following criteria for developing and benchmarking solutions to practical steg detection.

1. **Authentication**. Each image is provided with pedigree of origin, including camera device, meta data (including EXIF data), acquisition app, etc.
2. **Representation.** The data includes representatives of practical scenarios encountered in crime cases.
3. **Evaluation.** The data is effective for evaluating and benchmarking standard algorithms including commercial software and academic algorithms, and allows reliable, reproducible, and measurable results that may be used in a court of law.
4. **Public access and free of copyright or privacy issues.** Communities require low-cost or free access to a standard data set without encountering copyright or privacy issues.

A review of several popular data sets used in digital image forensics research finds they have varying degrees of agreement with the four criteria, revealing a need for such data. Our goal is to provide an image database suitable to researchers in both crime lab and academic settings, allowing for performance evaluation on both academic steganalysis algorithms and commercial steganalysis software. Data in the database should allow simulation of "in the wild" digital evidence, as well as data appropriate for academic researchers to pursue problems related to real data created from real-world mobile stego apps. Should steganography evidence achieve the penultimate goal of being presented in a court case, performance evaluation on a standardized data set is in line with Daubert's requirement that scientific expert testimony be assessed for its evidentiary reliability [36].

In this paper we introduce a publicly available data set that agrees with much of 1-4 above. The database consists solely of images from mobile phones and mobile stego apps, representing data that certainly forensic practitioners see with increasing frequency. With over 800,000 stego and non-stego images created from stego apps on mobile phones, the data is offered in several file formats and with a wide variety of: scenes, exposure settings, embedding algorithms (software apps), embedding rates, devices (24) and models (10). A user-friendly web page provides detailed information on the content of the database and how to query and download. The data in StegoAppDB has extensive relational information for each data item. We continue to add to the database, and we invite suggestions to improve the contents or access. We anticipate access to a database with such varied and richly notated corpora will provide opportunities for motivated developers and researchers to create more practical solutions for steganography detection, and perhaps be useful for other digital image forensic purposes.

The remaining sections of the paper are as follows. In "Related Works," we review popular data sets and current software for steg detection. In "Creation of the Database," we discuss the acquisition procedure for original images and the generation of stego and other images for the database. In "Descriptive Statistics, Substantiation and Evaluation," we give descriptive statistics and provide results of several experiments to substantiate our claims of the database′s investigatory nature. The section "User Interface to Query for Data" describes how the database can be queried. We conclude with remarks for forensic scientists from academic and practicing communities, including potential uses of our database in other areas of digital image forensics.

Related Works

While both academics and practitioners pursue forensic analysis of digital images, the two communities have very different goals. Academics seek innovative methods that advance the state-of-the-art in focused areas of conceptual performance; for steganalyisis, this can mean improving detection error, or introducing a new framework to improve performance, such as the relationship of embedding changes to syndrome coding [16]. At the other end of the spectrum, forensic practitioners expect their results to be interpreted in context of legal matters, and require outcomes that are validated by well-established and reproducible scientific procedures supporting quantitative analysis of uncertainty. Since our objective is to provide a set of data suitable to both communities, this section discusses issues from each community, including datasets, software and algorithms used to develop and benchmark steg detection. Datasets used in the development of algorithms or software can, of course, influence the performance of the software.

One typical job of an academic steganalyst is to create a new embedding algorithm (examples are WOW [22] or J-UNIWARD [21]), and then test its security (ability to be detected) using steganalysis techniques. Testing unknown images for hidden content with no prior information occurs only in stego challenges, the last of which completed in 2010 [4], and another which is currently ongoing. As a standard practice, academic steganalyzers use a set of innocent images and create their own stego images using their code, and, for example, data from BOSSbase [3], and more recently, for data-hungry convolutional neural networks, the BOWS-2 data set [5]. It is well-known that the peculiarities of a data set combined with features and machine learning classifier influences detection performance, e.g., see [34], where the authors verify different security performances based on compression or downsampling rates of RAW cover sources.

While other image forensics are not the focus of our work, we note that two data sets created explicitly for image forensics - the popular and publicly available data set for forgery detection RAISE [17] and the excellent Dresden database for camera identification [20] - have been used by researchers for steganalysis experiments. They are included below in our comparison with the four criteria. For descriptions of additional data sets, we refer the reader to the extensive review in [17].

1. BOSSbase [4] was created for an academic steganalysis competition in 2010 and by hindsight, introduced a clear example of the cover-source mismatch problem [10]. BOSSbase has been used successfully to evaluate the performance of hundreds of academic steganalysis algorithms. It contains 10,000 RAW images from 7 different still camera de-

vices. However, it was not designed to benchmark commercial programs. The images have EXIF data, whose authentication, copyright and privacy statuses are unknown. The scenes reflect real-life scenarios, are collected in auto-exposure settings or half-auto settings, with the intent to produce high visual-quality images; thus, exposure settings do not cover more extreme lighting conditions.

2. The RAISE database contains 8156 RAW images, constructed primarily for forgery detection [17], in auto-exposure modes from three different still cameras over a period of several years. RAISE does not contain any JPEG images. It provides a challenging real-world data set They are authenticated, and the scene content reflects real-life scenarios. They are copyright-free, and most likely do not have privacy issues.
3. The Dresden Image Database [20] was also created for the forensic community, with the main purpose for camera identification. It has almost 17,000 images, mostly in high-quality JPEG, from 73 still cameras devices representing 25 distinct models, with many different camera settings. However, the scene content is limited to a relatively small number of different scenes in order to replicate same scene/different camera scenarios for intense camera-specific forensic processing. Thus, it does not provide the wide range of scene content required for reliable stego detection. The data are authenticated and many images reflect real-life scenarios. They are copyright-free, and are taken with auto-exposure settings.

Both RAISE and Dresden image data sets were not designed with steganography in mind, and so they lack stego images, naturally. Thus, their data cannot be used directly for evaluation of commercial steg detection software. While undoubtedly stego images could be produced and made available, currently this has not been done.

From an analysis of existing data sets and our requirements for creating a database to meet much of the four criteria, we propose our plan. The challenges were to meet as much of the four forensic criteria as possible, and generate hundreds of thousands of images. Our data set consists of images produced from mobile phones and mobile apps, and retains provenanced side information for all images. The data is copyright-free and has no privacy issues, and is free and open to the public. To generate the images, we created a fast and efficient method using program analysis and reverse engineering. Each app has unique characteristics and required individual inspection using apk tools to generate the intermediate images and specific embedding rates. With these problems solved, StegoAppDB was populated. It is available online at *https://forensicstats.org/stegoappdb/* [12] and we encourage the forensic community to access, download and use the data, and contact us with any suggestions.

Creation of the Database

To populate our database, we use the four criteria as a guideline to develop appropriate data acquisition methods, data source choices, and auxiliary data authentication information. Our procedure was designed to collect a large amount of image data, to represent a reasonable number of different mobile phone cameras, and to include stego images from apps that are native to mobile devices. During the initial phase of data collection, we observed the well-known phenomenon that exposure settings of the images—related to image noise—impact the error rates of steganalysis algorithms [28], [29]. Therefore, to make the image data representative, we collect the original photos with large diversities in exposure settings, which includes both ISO value and exposure time. To acquire such large amounts of photos from the phone's camera, we created our own research camera app that allows the photographer to collect 20 images automatically. Then, in order to create the large number of stego images from the stego apps on the phone, we reverse engineered each app individually using manual methods so that we could run the stego app directly on the phone that was cabled to the computer. This allowed us to produce stego images on the phone much faster than was possible by entering the same information on the app on the phone by a human. This section describes the process by which we create images that are put into our database.

Collecting Original Images using Cameraw

Exploiting the comprehensive range of features available for camera APIs on Android and iOS platforms, we create a camera app called "Cameraw" to capture images on our lab's smartphones [13]. Our main goal with Cameraw is to create a standardized process for the image acquisition procedure so that the app is simple to use, takes large amounts of photos quickly at acceptable visual fidelity but with varied exposure settings, and reduces the number of screen touches. By the press of one button, Cameraw automatically captures 20 images of one scene. After the "capture" button is pressed, the following steps ensue:

1. The auto focus and auto exposure pre-capture sequence is triggered.
2. After a short time, the focus is locked and exposure settings converge.
3. Two auto exposure (AE) images are captured, one JPEG and one DNG, and 9 manual exposure settings are calculated using the AE values.
4. The camera switches to manual exposure mode, and captures 9 pairs (one JPEG and one DNG) of additional images at the 9 manual settings, for a total of 20 images with 10 different exposure settings, within 15 seconds.

Although the Android and iOS camera libraries provide auto exposure bracketing functionalities, we choose not to use them because they do not provide a wide enough range of ISO and exposure time values. Instead, we implement a customized bracketing method using the auto ISO and exposure time values that retain fairly good fidelity image quality. Let i be the auto ISO, and let e be the auto exposure time. We calculate 3 ISO values: $0.5*i$, $1.75*i$, $3.0*i$, and 3 exposure time values: $2.0*e$, $1.25*e$, $0.5*e$. From these values, we generate 9 distinct pairs of ISO and exposure time settings for capture in "manual" mode of the camera API. The other camera parameters are chosen by the built-in camera firmware. We lock all camera parameters during capture (except ISO and exposure time), to ensure all 20 photos have the same capture settings.

Cameraw is implemented with the camera2 API [1] in Android, and the AVFoundation framework [2] in iOS. We installed Cameraw on all devices, 10 Android phones and 14 iPhones. Sev-

eral photographers were hired to capture a minimum of 100 indoor scenes using each device, resulting in total of 2412 scenes and 48,240 original images. We use the term "original" to describe those images that are captured using Cameraw on the mobile devices, with no further processing applied, including cropping. Our database contains other types of images that are created by processing methods, which we discuss in the next section.

Generating Cover-Stego Images

To facilitate benchmarking capabilities for steg detection of stego images generated from mobile stego apps, the database provides a large selection of app-generated stego images. We discovered that some apps embed a distinctive "signature" into the image, while others do not. We provide stego images for both types of apps.

We select six apps and create a coding environment on a computer, using a mobile phone connected to the computer, and batch-produce stego images with very specific embedding rates much quicker than possible by hand on the device. This process allowed us to save intermediate images where needed, specifically "cover" images. A cover image can be viewed as a 0% embedded stego image, that is, a cover image is the image just prior to embedding and having the same pixel dimensions as the stego image. For many stego apps we observed, the image input to the app through the GUI user interface may undergo image resizing, among other alterations. Since academic steganalysis algorithms require cover-stego image pairs of the same dimension, machine learning classifiers in the traditional sense cannot be constructed from "original" image-stego image pairs as used by mobile stego apps. See Fig. 1, where the data flow for a generic mobile stego app is displayed on the left hand side. From the GUI of the stego app, the user inputs the payload, or message, as well as an input image, typically from the gallery or using the camera. The user may also enter a password as well. These data are passed to the internal code of the app, and any image processing such as resizing, or additional image creation, such as a cover image, can be obtained only through program analysis. Since the cover image is not known, feature extraction and subsequent machine learning development cannot be completed, as displayed on the right-hand side of Fig. 1.

Cover images are a currently a necessity for machine learning algorithm development. This is not the case for certain stego apps that add an app-specific signature in the stego image that can be detected using other means. We next describe the process to generate cover and stego images for all apps.

Stego Apps.

To generate cover and stego images, we select six real-world stego apps: five Google Play Store apps and one Apple App Store app. See Table 1. In the column "Embedding Domain," we see that two apps embed payload in JPEG domain while the remaining embed in the spatial domain. The "Open Source" column indicates that source code for two apps is not publicly available. Inaccessible source code increases the difficulty of studying the apps' embedding process. For these two apps, we use reverse engineering tools to analyze the binary code and to delineate the steps that generate a stego image. We refer the interested reader to [15] for case studies of reverse engineering applied specifically to mobile stego apps.

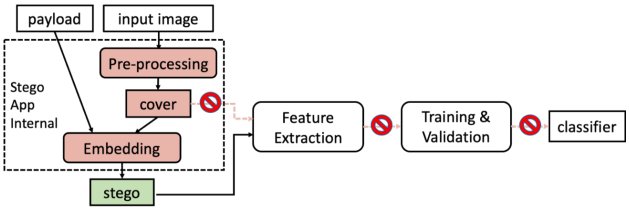


Figure 1: Data and processing flow in a generic mobile stego app (left-hand side). On the right-hand side are steps to create a machine classifier, which cannot be completed for a stego app image due to unknown cover image.

We define the images in our database in the following way: An *original image* is the image acquired by the mobile phone camera. An original image is used in many ways, such as an input image to a mobile stego app; or pieces are cropped to produce smaller-sized images to be used as cover images to embedding algorithms. A *cover image* is the image that is paired to the stego image used for machine learning classifiers. A cover image has the same pixel dimensions (width x height) as the corresponding stego image, and can be viewed as a 0% stego image. We define an *input image* to be the image that is selected by the user on the GUI of the stego app (on the phone), usually from the phone's gallery or by using the phone's camera. In academic steganalysis algorithms that use cover-stego image pairs, the input image as defined here is identical image to the cover image. In mobile stego apps, the input image is the photo selected by the user and subsequently processed by the app developer to create the final stego image. This input image can differ from the cover image due to pre-processing or resizing by the app prior to embedding.

Batch Image Generation.

By manual code analysis of the apps, we determine how the embedding path is selected, which embedding method was used, pre-processing of input images and messages, etc. With knowledge of the data modification and processing steps, we create a script to batch generate cover−stego image pairs. Using source code modification and binary code instrumentation, we add two necessary functions to get complete side information for each stego image generated from a stego app:

1. We save the intermediate cover image (see Fig. 1). This ensures that the database contains the corresponding cover image for each stego image, and is a critical step for stego apps that resize the input images prior to embedding.
2. We generate stego images with specific embedding rates. This is achieved by analyzing the stego app's embedding procedure, including how the payload is processed. To generate a stego image with a specific embedding rate, the modified app first calculates the cover image capacity and then calculates the necessary length of the embedded payload to achieve that rate. We correct for additional length added for auxiliary information.

As machine learning steganalysis classifiers prefer smaller images due to computational constraints, we process the original DNG and JPEG images to create (symmetrically) center-cropped grayscale PNG images (512x512). We use the term "cropped" to label these innocent PNG images. The PNG images are used as

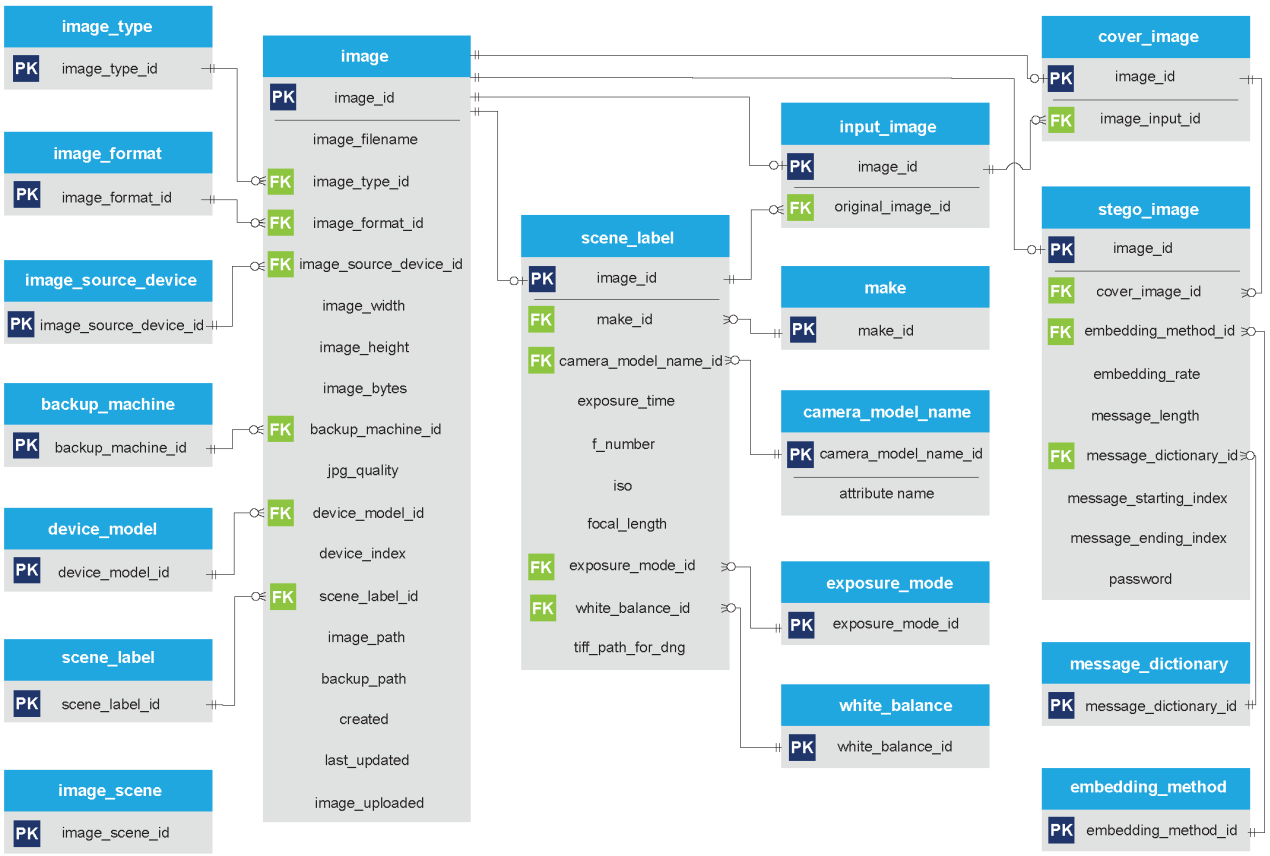


Figure 2: The entity-relation (ER) diagram for the database StegoAppDB.

input images to the four spatial domain embedding apps to generate stego images. For the two frequency domain embedding apps, original-sized images are used as input images. Both DNG and JPEG images are used to generate *PixelKnot* stegos, while only JPEG images are used to generate *Passlok* stego images due to Passlok's inability to accept DNG files. All information is stored in the database for each stego image. We remark that all other variable being the same, the type of image used, DNG or JPEG, can affect the steg detection error rate [34]. The set of 10 JPEG and 10 DNG images from one scene (20 total images) differ only in file format and allow for experiments that involve compressed versus raw data comparisons.

The input messages embedded into the stego images are actual text messages selected from a set of dictionaries containing 34 Shakespeare plays [35], ensuring variability by randomly selecting the start line for each text message from the dictionaries. The message dictionaries are provided with each download from StegoAppDB.

After this extensive coding process, we produce a large number of innocent images and corresponding stego images, at a variety of embedding rates, from a variety of phone models and devices, using a variety of currently-available mobile stego apps. The information used to generate each stego image is stored in data fields in StegoAppDB, including the associated cover image. Each cover image generated internally from an app is also stored in StegoAppDB, along with its information. The availability of this extensive side information for each image provides provenance and clarity about the generation process for each image.

The data collected and populated in StegoAppDB is arranged in a relational database. The entity relation (ER) diagram is shown in Fig. 2. This details the relation between cover images, input images, original images, and stego images, as well as the other information stored for each type of image.

Descriptive Statistics, Substantiation, and Evaluation

In this section, we present a quantitative summary of the data in StegoAppDB. We design experiments and give results that show how the use of different data in several current steg detection tools can substantiate strengths or weaknesses of the tool. We also select some academic machine-learning-based detection algorithms and evaluate their performance using the data from StegoAppDB.

Table 2: Database summary: smartphone models, camera specifications, and number of images of different types.

Device Model	# Devices	ISO Range	Exposure Time Range	# Scenes	# Original Images	# Cropped Images	# Covers	# Stegos
Google Pixel 1	4	50 ∼ 3735	1/1258 ∼ 1/7	402	8040	8040	36180	180900
Google Pixel 2	2	50 ∼ 1708	1/9358 ∼ 1/9	201	4020	4020	18090	90450
Samsung Galaxy S8	2	56 ∼ 2097	1/2643 ∼ 1/12	208	4160	4160	18720	93600
One Plus 5	2	100 ∼ 3200	1/2777 ∼ 1/8	201	4020	4020	18090	90450
iPhone 6s	2	25 ∼ 1000	1/60 ∼ 1/3	200	4000	4000	4000	20000
iPhone 6s Plus	2	25 ∼ 1250	1/67 ∼ 1/3	200	4000	4000	4000	20000
iPhone 7	4	20 ∼ 1250	1/67 ∼ 1/3	400	8000	8000	8000	40000
iPhone 7 Plus	2	20 ∼ 1000	1/60 ∼ 1/5	200	4000	4000	4000	20000
iPhone 8	2	20 ∼ 800	1/60 ∼ 1/3	200	4000	4000	4000	20000
iPhone X	2	20 ∼ 800	1/62 ∼ 1/3	200	4000	4000	4000	20000
total	24	20 ∼ 3735	1/9358 ∼ 1/3	2412	48240	48240	119080	595400
total images								810960

Descriptive Statistics of the Database Contents

The images in StegoAppDB comprise original images, grayscale PNG images, and stego images and their corresponding cover images. Table 2 displays a summary of camera models and image data for the 24 devices. Each smartphone device acquired at least 100 indoor scenes of images, where one scene has 20 images.

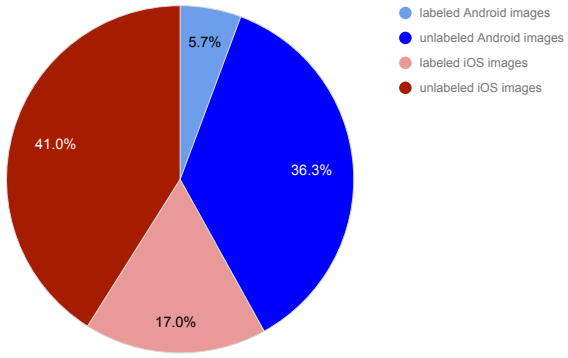


Figure 3: Percentages of labeled vs. unlabeled images, identified by operating system.

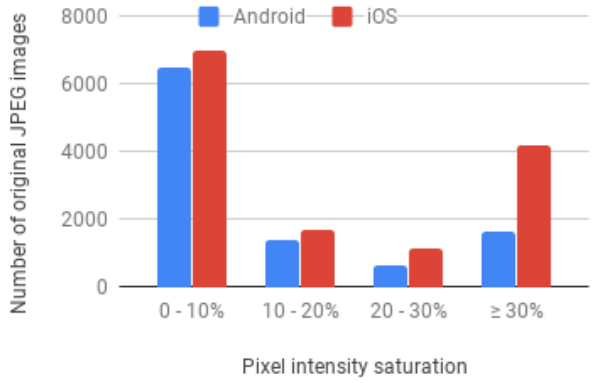


Figure 4: Distribution of saturation of intensity values.

During photo acquisition, the photographer assigns a scene one of ten labels, or is unlabeled: *books*, *apple*, *orange*, *chair*, *stairs*, *backpack*, *clock*, *keyboard*, *bottle*, and *keys*.Figure 3 shows

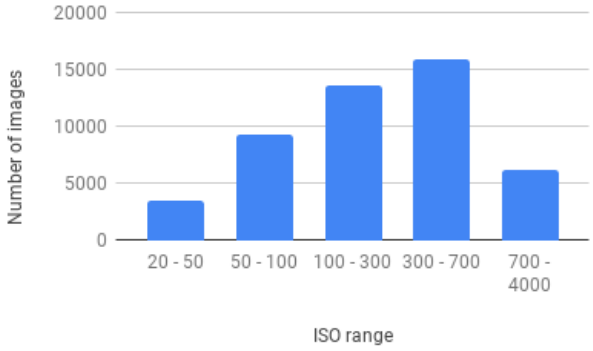


Figure 5: Distribution of ISO values of original images.

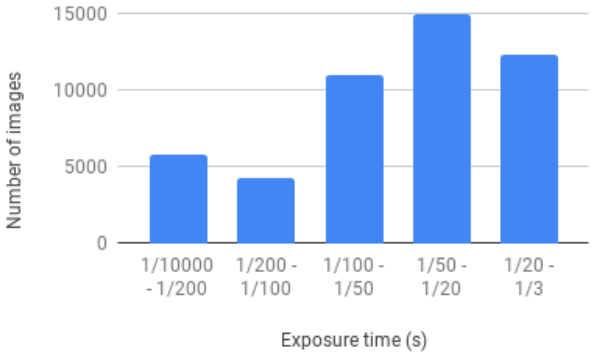


Figure 6: Distribution of exposure times of original images.

the number of labeled and unlabeled original images. Out of the 48240 original images, 10940 have labels.

In Fig. 4, we give a distribution of the number of images having saturated pixel intensity values. For an original JPEG image, we compute the proportion of intensity values across all three RBG planes that are below 5 and above 250. Each image falls into one of the four categories as given in Fig. 4, which shows their distribution across all original images.

As the exposure settings varied for 18 of the 20 original images in a scene set, these images cover a wide range of ISO and exposure time values. Figures 5 and 6 show the distribution of original images at different ranges of exposure settings.

As shown in Table 3, our database contains a total of

810,960, of which 595400 are stego and 119080 are cover. For each cover image, five stego images are generated at the embedding rates 0.05, 0.10, 0.15, 0.20, and 0.25. Parameters relevant to the embedding process such as embedding rate, change rate, input message, password, etc., for each stego image are included in the csv files downloaded with each set of images.

Table 3: Number of cover and stego images by Stego Apps.

Stego App	Operating System	# cover images	# stego images
PixelKnot	Android	20240	101200
Steganography (by Meznik)	Android	20240	101200
Pocket Stego	Android	20240	101200
MobiStego	Android	20240	101200
Passlok Privacy	Android	10120	50600
Pictograph	iOS	28000	140000
total		119080	595400

Commercial or free programs assert capabilities of identifying steganography in some files or work environments. For example, the embedding algorithm F5 [37] has been widely used to hide messages in JPEG images, and three software programs − StegoHunt[38], DC3-StegoDetect[18] and Provos−StegoDetect [33] − claim to detect stego images created by the standard F5 algorithm. To verify this, we randomly select 2000 original JPEG images from our database, implement the standard F5 algorithm to create 2000 stego images with 10% embedding rate, and present all 2000 cover-stego pairs to the programs. The result is provided in Table 4.

Table 4: Error of detection on images generated by standard F5

Error Type	Stego Hunt	DC3−StegDetect	Provos−StegDetect
False Alarm	0%	0%	24.6%
Misdetection	5.2%	0%	47.4%
Avg. Error	2.6%	0%	36.0%

As we can see from the Table 4, both Stego Hunt and DC3−StegDetect have very good performance in detecting the stego images generated by the standard F5, while the Provos−StegDetect has an unacceptable error rate in our experiment.

However, images generated by stego apps are very different from the images created by standard academic embedding algorithms. In this case, we use the app PixelKnot as an example, since it also implements the F5 steganography algorithm with some minor modifications. Again, we perform a different random selection of 2000 cover-stego pairs from our database that are created by PixelKnot, and present them to the three software programs. The result is presented in Table 5.

Table 5: Error of detection on images generated by PixelKnot

Error Type	Stego Hunt	DC3−StegDetect	Provos−StegDetect
False Alarm	0%	0%	24.6%
Misdetection	100%	100%	75.4%
Avg. Error	50%	50%	50%

As we can see from the Table 5, all three programs fail to detect the stego images created by PixelKnot. The developers' code modification of F5 implemented in their version of PixelKnot includes an omission of a signature of the F5 algorithm when writing the stego output images. The signature is found in stego images output from the standard F5 code. This shows that the rise of mobile stego apps can bring new challenges to the forensic image analyst. Indeed, if a digital forensic tool fails in a particular scenario, it can be argued that those results should be used to correct the tool weakness [8]. With our comprehensive stego database, we are eager to work with research teams or companies that have strong interest in this new challenge and opportunity − detecting stego images from mobile stego apps.

Evaluation of Machine Learning Detection Algorithms

It is well-known that machine learning is a very powerful tool in detecting stego images created by academic steganography methods. To show that it is also very effective in detecting stego images by apps, we design the following experiment.

Here, we target classification of stego images created by four Android apps: PixelKnot, Steganography, Pocket Stego, and Passlok Privacy. To that end, we select original JPEG images from four different devices: one Google Pixel 1, one Google Pixel 2, one Samsung Galaxy S8, and one OnePlus 5, and all corresponding stego images. With embedding rate fixed at 10%, for each device and each app, we have 1000 cover-stego pairs of images from the database, in which 500 pairs are used for training, and the remaining 500 pairs are used for testing.

For the machine learning methods, we implement the CC-JRM [26] for feature extraction on JPEG images (stego images created by PixelKnot and Passlok) and SRM [19] for feature extraction on PNG images (stego images created Steganography and Pocket Stego). The FLD ensemble classifier [27], which is essentially a random forest method, performs the classification. The classification accuracy is present in Table 6.

Table 6: Classification accuracy of detecting cover-stego pairs by ML algorithms

Apps	Pixel 1	Pixel2	Samsung S8	One Plus 5	Mix of four devices
PixelKnot	97.5%	97.6%	97.6%	98.3%	99.0%
Steganography	98.0%	97.8%	99.4%	97.7%	98.6%
Pocket Stego	96.8%	97.3%	99.5%	98.3%	98.4%
Passlok Privacy	99.0%	97.1%	98.3%	98.3%	98.6%

As we can see in Table 6, machine-learning-based detection algorithms have very impressive performance in detecting stego images, provided that we know which app create them. However, we point out that, in the above experiment, we only use JPEG images as the original source. In the case where RAW images are used as the original source to generate the stego images, the accuracy drops significantly [14]. Embedding rate can also affect the detection rate, and in realistic scenarios for small messages, the effective embedding rate can be less than 3%, making detection even harder. In this experiment, only a few machine learning algorithms are testified. We welcome others to develop better detection algorithms for stego images from apps and test using our database.

User Interface to Query for Data

The design of the webpage for public access was created to facilitate as simple a process as possible for data queries, given the richness of the data. The database can be accessed through a link from CSAFE's webpage [12]. On the database's main webpage,

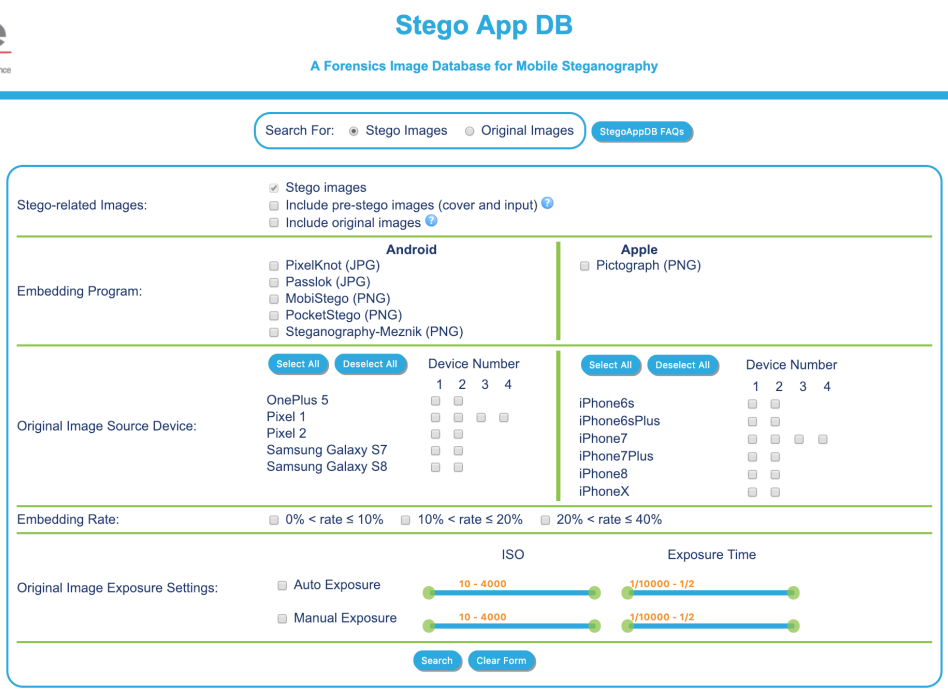


Figure 7: Search options for stego images in StegoAppDB.

we give descriptions of the information the user is able to query, listing the query parameters to search on a wide range of image characteristics.

There are two main types of data:

- "Original" images, which are the images captured using Cameraw on mobile devices, with native pixel dimensions as determined by the camera;
- "Stego" images, which are images embedded with messages and produced using the stego apps on mobile devices.

Original images or portions thereof may be used for academic steganalysis algorithms, forgery, or camera identification problems. Original images can be downloaded and portions cropped out by the user, or downsized to produce images appropriate for academic embedding algorithms. When querying for stego images, options are available to download other images that are associated with the stego images, such as cover images or input images. Downloading a cover image that generated the stego image would be useful in constructing a steganalysis machine learning classifier where cover images are required for training. (Recall that some apps downsize the input image prior to embedding, and the downsized image is not available to the app user.)

From Fig. 7, a picture of the stego search webpage, note that the "Original Image Source Device" data is grouped by mobile operating system: Android and iOS (Apple). The mobile app data is also separated by operating system, because each app is written specific to the operating system.

To find "stego" images, a user can query with parameters related to both the stego images and the corresponding source (original) images. A search is premised on including data satisfying the checked options. For stego searches, there are five options:

- **Stego-related Images.** A user can choose whether or not to include the input and cover images associated with the stego images.
- **App Embedding Program.** Select to include from six different stego apps.
- **Original Image Source Device Model.** Select to include stego images created from corresponding original images on specific device models.
- **Embedding Rate.** Select to include from different ranges of embedding rates.
- **Original Image Exposure Settings.** Select to include stego images created from corresponding original images with specific exposure settings.

The search for original images is similar to searching for stego images. To find "original" images, a user can query the database by selecting boxes from the following options:

- **Device Model.** Select to include from any of the 10 device models and 24 devices.
- **Original Image Format.** Select to include from JPEG and JPEG quality, and/or DNG.
- **Scene Content.** Select to include from 10 indoor scene labels, and/or select "unlabeled."
- **Exposure Settings.** Select to include from auto exposure and/or specific ranges of ISO and exposure times.

When a set of images are downloaded, a zip file is provided. The contents of the zipped file are: folders in which the images reside, one folder for each type of image (stego, cover, input, and originals); a csv file for each image type downloaded that contains the side information for each image; a README text file that describes the search parameters for that particular search query; and a zip file containing the message dictionaries.

With each image identified in a query, all meta information and side information are also provided. For original images, this

includes EXIF data, device and label information. For stego images, all embedding parameters including stego app, input message, password, etc. are provided. EXIF data is not included with any image except "original" type, and the EXIF is included in an original image as part of the Cameraw acquisition process.

Conclusion and Potential Uses

In this paper, we announce the new database StegoAppDB, the first data set consisting of hundred of thousands of images from mobile phones. The database contains images from 24 different devices of 10 different models, and includes stego images generated by stego apps from the phones. It has a variety of different types of data that make it amenable not only for steganography, but for forgery and camera identification. Each image has a rich set of annotated side information, free from copyright and privacy issues, and is publicly available. The database continues to include additional data, and updates to the database are located on its homepage *https://forensicstats.org/stegoappdb/* [12].

Recently it was shown that additional large steganography datasets can help solve other digital image forensic including detection of multiple image manipulations using Convolutional Neural Networks (CNNs) [41]. Here, learned parameters from a CNN trained for steganalysis were "transferred" to a new CNN, the latter of which was trained successfully on a small amount of data from a new database to detect median filtering, JPEG compression, etc. Having more data sets available to the digital image forensics community can be a valuable resource.

Other digital image forensic topics may find our database suitable. The Dresden and RAISE forensic data sets are used for algorithm development in many digital image forensic research works, and include image resampling detection [6], camera device identification[23], camera model identification [7], generative adversarial network (GAN) attacks to photo-response non-uniformity-based (PRNU) [11], and identifying forensic traces of images created by GANs [30], among many other forensic works. The StegoAppDB database can also be used in similar research experiments for creating and testing these types of algorithms.

We invite the forensic image analysis community to use StegoAppDB and offer suggestions for improvements and future content.

Acknowledgement

We are grateful to the following undergraduate students for helping us acquire images for our database: Yiqiu Qian; Joseph Bingham; Chase Webb; and Mingming Yue. This work was partially funded by the Center for Statistics and Applications in Forensic Evidence (CSAFE) through Cooperative Agreement #70NANB15H176 between NIST and Iowa State University, which includes activities carried out at Carnegie Mellon University, University of California Irvine, and University of Virginia.

References

[1] Android Developers. Android Camera2 API. `https://developer.android.com/reference/android/hardware/camera2/package-summary`, 2018.

[2] Apple Inc. AV Foundation. `https://developer.apple.com/av-foundation/`, 2018.

[3] P. Bas, T. Filler, and T. Pevnỳ. Bossbase. `http://agents.fel.cvut.cz/boss/index.php?mode=VIEW&tmpl=materials`, 2010.

[4] P. Bas, T. Filler, and T. Pevnỳ. Break Our Steganographic System: The Ins and Outs of Organizing BOSS. In *Information Hiding*, pages 59–70. Springer, 2011.

[5] P. Bas and H. . `http://bows2.ec-lille.fr`. T. . B. M. . J. Y. . . Furon, Teddy.

[6] B. Bayar and M. C. Stamm. On the robustness of constrained convolutional neural networks to jpeg post-compression for image resampling detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 2152–2156. IEEE, 2017.

[7] B. Bayar and M. C. Stamm. Towards open set camera model identification using a deep learning framework. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2007–2011. IEEE, 2018.

[8] J. Beckett and J. Slay. Digital forensics: Validation and verification in a dynamic work environment. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 266a–266a. IEEE, 2007.

[9] L. Best-Rowden, S. Bisht, J. C. Klontz, and A. K. Jain. Unconstrained face recognition: Establishing baseline human performance via crowdsourcing. In *IEEE International Joint Conference on Biometrics*, pages 1–8, Sept 2014.

[10] R. Böhme. An epistemological approach to steganography. In *International Workshop on Information Hiding*, pages 15–30. Springer Berlin Heidelberg, June 2009.

[11] N. Bonettini, L. Bondi, S. Mandelli, P. Bestagini, S. Tubaro, and D. Güera. Fooling prnu-based detectors through convolutional neural networks. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 957–961. IEEE, 2018.

[12] Center for Statistics and Applications in Forensic Evidence. StegoAppDB Homepage. `https://forensicstats.org/stegoappdb/`, 2018.

[13] W. Chen. Cameraw, an Android camera app for digital image forensics. Technical report, CSAFE, Iowa State University, Oct. 2017.

[14] W. Chen, L. Lin, M. Wu, Y. Guan, and J. Newman. Tackling android stego apps in the wild. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2018*, 2018.

[15] W. Chen, Y. Wang, Y. Guan, J. Newman, L. Lin, and S. Reinders. Forensic analysis of android steganography apps. In G. Peterson and S. Shenoi, editors, *Advances in Digital Forensics XIV*, pages 293–312, Cham, 2018. Springer International Publishing.

[16] R. Crandall. Some notes on steganography. *Posted on steganography mailing list*, pages 1–6, 1998.

[17] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. RAISE: a raw images dataset for digital image forensics. In *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 219–224. ACM, 2015.

[18] DC3. Stegdetect. personal correspondence with William Eber, 2017.

[19] J. Fridrich and J. Kodovsky. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, 2012.

[20] T. Gloe and R. Böhme. The 'Dresden Image Database' for Benchmarking Digital Image Forensics. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1584–1590, New York, NY, USA, 2010. ACM.

[21] V. Holub, J. Fridrich, and T. Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014(1):1, 2014.

[22] V. Holub and J. J. Fridrich. Designing steganographic distortion using directional filters. In *WIFS*, pages 234–239, 2012.

[23] Y. Huang, L. Cao, J. Zhang, L. Pan, and Y. Liu. Exploring feature coupling and model coupling for image source identification. *IEEE Transactions on Information Forensics and Security*.

[24] A. D. Ker, P. Bas, R. Böhme, R. Cogranne, S. Craver, T. Filler, J. Fridrich, and T. Pevnỳ. Moving steganography and steganalysis from the laboratory into the real world. In *Proceedings of the first ACM workshop on Information hiding and multimedia security*, pages 45–58. ACM, 2013.

[25] G. Kessler. An overview of steganography for the computer forensics examiner. `https://www.garykessler.net/library/fsc_stego.html`, 2015. Last Accessed: 2018-07-31.

[26] J. Kodovskỳ and J. Fridrich. Steganalysis of jpeg images using rich models. In *Media Watermarking, Security, and Forensics 2012*, volume 8303, page 83030A. International Society for Optics and Photonics, 2012.

[27] J. Kodovsky, J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7(2):432–444, 2012.

[28] L. Lin, W. Chen, Y. Wang, S. Reinder, Y. Guan, J. Newman, and M. Wu. The impact of exposure settings in digital image forensics. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 540–544. IEEE, 2018.

[29] L. Lin, J. Newman, S. Reinders, Y. Guan, and M. Wu. Domain adaptation in steganalysis for the spatial domain. *Electronic Imaging*, 2018(7):319–1–319–9, 2018.

[30] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi. Do gans leave artificial fingerprints? *arXiv preprint arXiv:1812.11842*, 2018.

[31] National Research Council and others. *Strengthening forensic science in the United States: a path forward*. National Academies Press, 2009.

[32] P. J. Phillips, W. T. Scruggs, A. J. O'Toole, P. J. Flynn, K. W. Bowyer, C. L. Schott, and M. Sharpe. Frvt 2006 and ice 2006 large-scale experimental results. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):831–846, 2010.

[33] N. Provos and P. Honeyman. Detecting steganographic content on the internet. In *Network and Distributed System Security Symposium*, San Diego, California, Feb 2002. Internet Society.

[34] V. Sedighi, J. Fridrich, and R. Cogranne. Toss that BOSSbase, Alice! *Electronic Imaging*, 2016(8):1–9, 2016.

[35] W. Shakespeare. The Complete Works of William Shakespeare. `http://shakespeare.mit.edu/`, 1993.

[36] H. S. Stern. Statistical issues in forensic science. *Annual Review of Statistics and Its Application*, 4:225–244, 2017.

[37] A. Westfeld. F5—a steganographic algorithm. In I. S. Moskowitz, editor, *Information Hiding*, pages 289–302, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[38] WetStone Technologies. Stego hunt. https://www.wetstonetech.com/product/stegohunt/, 2017.

[39] C. Woodford. Encryption and steganography. `https://www.explainthatstuff.com/encryption.html`, 2018. Last Accessed: 2018-07-31.

[40] R. Zax and F. Adelstein. Faust: Forensic artifacts of uninstalled steganography tools. *Digital Investigation*, 6(1-2):25–38, 2009.

[41] Y. Zhan, Y. Chen, Q. Zhang, and X. Kang. Image forensics based on transfer learning and convolutional neural network. In *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, pages 165–170. ACM, 2017.

Author Biography

Dr. Jennifer Newman received her BA in Physics from Mount Holyoke College and her PhD in Mathematics from the University of Gainesville, FL. She is an Associate Professor of Mathematics at Iowa State University in the Department of Mathematics, her research focusing on image processing, stochastic modeling, steganalysis and image forensics. She is a member of IEEE, IS&T, SIAM, and IAI.

Li Lin received his B.S. degree in Mathematics from Capital Normal University, Beijing, China. He is currently pursuing the Ph.D degree in Applied Mathematics at Iowa State University, Ames, Iowa. His research interests include statistical image forensics, steganalysis, and statistical learning.

Wenhao Chen received his B.E. and M.S. degree in Computer Science from Xi'An Jiaotong University. He is currently a Ph.D candidate in Computer Engineering at Iowa State University. His research interests include program analysis, malware detection, and steganalysis.

Stephanie Reinders received her BA in Journalism and Asian Languages and Literatures from the University of Minnesota (2005). After working for several non-profit organizations as an administrative assistant, she returned to school to earn a graduate degree in mathematics. She received a post-baccalaureate certificate in Mathematics from Smith College (2013) and currently is pursuing a PhD in Applied Mathematics and Computer Engineering at Iowa State University.

Yangxiao Wang received his B.E. in Computer Engineering from Iowa State University. He is currently pursuing the M.S. degree in Computer Science and Software Engineering at University of Washington Bothell. His research interests include program analysis and image forensics.

Dr. Min Wu is a Professor of ECE and Distinguished Scholar-Teacher at the University of Maryland, College Park. She received her Ph.D. degree in electrical engineering from Princeton University. She leads the Media and Security Team (MAST), with main research interests on information security and forensics, and multimedia/multimodal signal and data science. She was elected as an IEEE Fellow and an AAAS Fellow for contributions to signal processing, multimedia security and forensics.

Dr. Yong Guan is a Professor of Electrical and Computer Engineering, the Associate Director for Research of Information Assurance Center, and the cyber forensics coordinator for NIST-CSAFE at Iowa State University. He received his Ph.D. degree in Computer Science from Texas A&M University. Supported by NSF, NIST, IARPA, ARO and Boeing, his research focuses on security and privacy issues, including digital forensics, network security, and privacy-enhancing technologies for the Internet.

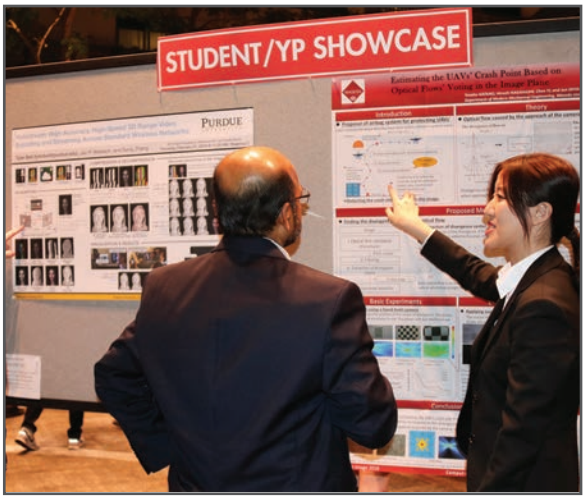
JOIN US AT THE NEXT EI!

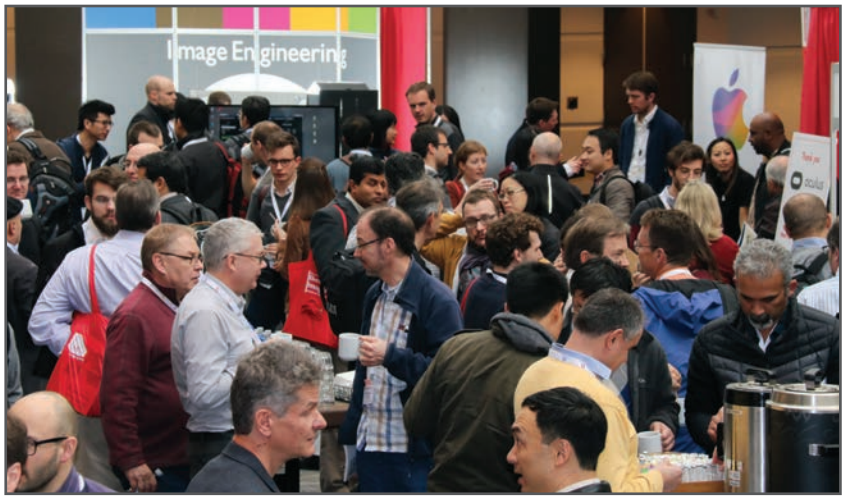
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org