Digital Check Forgery Attacks on Client Check Truncation Systems

Rigel Gjomemo[1], Hafiz Malik[2], Nilesh Sumb[1], and V.N. Venkatakrishnan[1] and Rashid Ansari[1]

[1] University of Illinois at Chicago {rgjome1,nsumb2,venkat,ransari}@uic.edu
[2] University of Michigan-Dearborn hafiz@umich.edu

Abstract. In this paper, we present a *digital check forgery* attack on check processing systems used in online banking that results in check fraud. Such an attack is facilitated by multiple factors: the use of digital images to perform check transactions, advances in image processing technologies, the use of untrusted client-side devices and software, and the modalities of deposit. We note that digital check forgery attacks offer better chances of success in committing fraud when compared with conventional check forgery attacks. We discuss an instance of this attack and find several leading banks vulnerable to digital check forgery.

Keywords: digital check forgery, financial applications, remote deposit

1 Introduction

Remote check deposit is one of the most recent internet-based practices introduced as an alternative to traditional paper-based check deposit and clearing, which required customers to physically go to the banks and banks to physically meet to exchange checks. This practice was enabled in the US by the Check 21 Act in 2008 [1], which established the equivalence between paper checks and their electronic representations (typically images), and regulated the practice of *check truncation*, which removes a paper check from the clearing process by using one of its electronic representations. This practice largely reduces costs related to physical exchange of paper checks among financial institutions.

To remotely deposit a check, a customer uses a *client truncation system* (outlined in Figure 1) and comprises: 1) a scanning device, which creates an image of the front and back of the check (step 1), 2) a processing software module (e.g., computer program or mobile app), which processes those images (step 2), and 3) a communication system to transmit the images over the internet to the bank servers (step 6). On the server side, the check image is recovered and processed by optical recognition software to determine the amount along with the routing and account numbers. The extracted information is further processed to clear the check. Common *client check truncation systems* in use today include scanners and computers (businesses) and smartphones (end customers).

This work was partially supported by National Science Foundation grants CNS-1065537, CNS-1069311, and CNS-0845894

The convenience of remote check deposit using a *client check truncation system* has made this feature very popular among financial institutions and their customers. According to recent statistics, millions of private and business bank customers are using it on a daily basis in the United States, and several governments and financial institutions worldwide have already introduced it or are projected to introduce it in the near future [2–6].

In this paper, we demonstrate that this convenience comes with an increased risk of check forgery, especially so when compared with the more traditional paper-based check deposit. This is especially significant given that (paper-based) checks remain the payment type most vulnerable to fraud attacks, with frauds amounting to 69% of all payment frauds [7] and the revenue losses due to check fraud in the U.S. alone amount to approximately $645 million [8].

This paper examines the risk of check forgery associated with remote check deposit and is based on the following observations about the changes introduced in check transactions by *client check truncation systems*: 1) digital image processing enables sophisticated forgeries on check images with an unprecedented precision, 2) functions such as check acceptance previously executed by trusted and well-guarded entities (bank tellers, ATMs) have been delegated to untrusted entities (users) that use the *client check truncation system*, 3) substitution of the paper-based checks with image-based checks has rendered well-established, decades-old anti-forgery techniques mostly useless and 4) the paper trail is eliminated since the physical check remains in the hands of the fraud perpetrator.

Based on these observations, we devise a class of attacks that demonstrate the feasibility of successful digital check forgery aided by untrusted *client check truncation systems*. These attacks are based on client device and software tampering to inject forged images in the transaction and on a library of image processing modules that we created to digitally alter check images. One particular instance of this class of attacks is outlined in Figure 1, where in addition to the normal truncation steps, a check image is extracted at some point along the path to the server, for instance before it reaches the processing software module (step 3), digitally forged by the attacker using custom-made or off the shelf tools (step 4), and replaced with a forged image before being sent to the bank's servers (step 5). Another instance of this class of attacks includes creation of a forged check from scratch, without possessing an existing check.

To demonstrate the practicability of these attacks, we describe specific attack instances performed on banking applications belonging to three Fortune 500 banks where the *client check truncation system* is implemented on Android smartphones. We carefully designed the experiments to avoid any harm to actual banks or customers. We also followed responsible disclosure practices, where we shared our findings with the vulnerable banks more than five months prior to this submission, to give the banks sufficient time to develop and deploy appropriate countermeasures. In our conversations, all banks acknowledged the vulnerability and the underlying issues raised by our research.

Contributions. The scientific purpose of this paper is three-fold: 1) To examine the threat on client check truncation systems and understand their inherent

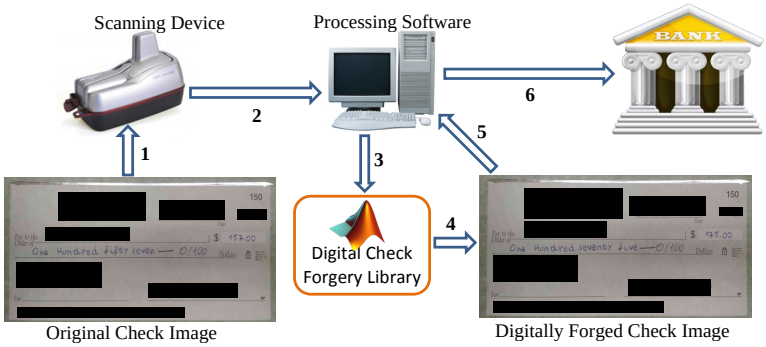


Fig. 1: Check Truncation System and Attack Description

weaknesses, 2) to analyze the possible ways by which a criminal could construct advanced check forgery attacks and 3) to shed light on appropriate countermeasures that would thwart such attacks. We make the following contributions:

- We highlight the easiness of carrying out digital check forgery through long established and powerful image processing techniques (§ 3).
- We compare classic physical check forgery techniques with digital check forgery techniques and highlight the ineffectiveness of classic anti-forgery mechanisms in preventing digital check forgery (§ 2).
- We describe a framework and techniques that can be used to digitally tamper check images (§ 3, § 4).
- We describe an instance of an attack that targets the *client check truncation* systems of three major banks implemented on Android smartphones (§ 4, § 5).
- Based on our insights and experience, we provide in § 6 some guidelines and suggestions for possible countermeasures against such attacks.

2 Current Check Transactions and Anti-forgery Measures

In this section, we provide a short background of check transactions and survey the history of digital check processing as well as common forgery techniques and anti-forgery countermeasures developed to prevent check forgery.

Prior to the 90s, a check issued by a bank that is deposited into an account of another bank required physical exchange of the paper check between the two banks before the money transfer took place. To avoid delays in such exchanges, central clearing facilities were developed, wherein banks met each day, where the paper copies were exchanged and the money credited and debited from the relevant accounts. To avoid forgeries, checks had to be examined manually by several bank employees along their path (teller of the receiving bank, often teller's supervisor, as well as employees of the settling bank). This process was necessarily labor-intensive, costly and slow.

As check transactions became common and the volume of exchanged checks increased, magnetic ink routing and account numbers enabled machines to read and sort the deposited checks much faster. However, the clearing process was still

Techniques	Usage	Digital Checks
Paper-based	Paper changes visible properties if tampered	Ineffective
Ink-based	Ink changes visible properties if tampered	Ineffective
Print-based	Printed patterns visible on original check only	Camera-dependent

Table 1: Common Techniques to Combat Check Forgery

dependent on physical exchange of checks at a central clearing house, somewhat still slowing the clearing process.

Check Truncation. To overcome these limitations, the Check 21 Act came into effect in the U.S. in October 2004, establishing the legal equivalence between paper and substitute checks (paper representations of checks with the same information as the original checks), and their electronic representations [9]. This Act expedited check clearing by regulating the preexisting practice of *check truncation*, in use by some banks. As a result, older practices of paper-based check clearing could be used together with the newer practice of check truncation.

The next development included the widespread use of *client check truncation systems*. These systems brought check truncation facilities to bank customers via a flood of technologies for remote check processing. Such systems include dedicated check scanners, PC clients, as well as smartphones. This development brought the benefits of electronic check processing to the end customers by providing valuable savings efforts related to physically going to the bank. In addition, the original paper check remained with the end customers.

2.1 Traditional Check Forgery

Check forgery is executed by physically altering the information written on a check. Alterations may involve amounts, payee names, routing and account numbers, dates, and so on. Check forgery may be executed in many ways, most commonly by: 1) *photocopying* an original check using image processing tools and printing devices, 2) *check washing*, where the ink on the original check is erased using chemical compounds and new information is written on the check, and 3) *check fabrication*, where a completely new check is created.

Table 1 outlines some common techniques currently used to combat paper-based check forgery. (We omit the techniques that can be used on the back-end, such as account reconciliation, as they are common to both paper and digital checks.) The goal of these techniques is to make *physical check forgeries* more difficult and to detect forgeries when checks are submitted. They include: 1) paper-based ones focused on the paper material of the check, which is produced by highly specialized and difficult to replicate technologies and is often sensitive to chemical materials, 2) Ink-based ones, such as bleeding ink and magnetic ink character recognition (MICR), which focus on the ink used in the original checks, and 3) Print-based ones, such as ultra-violet (UV) printing, void pantographs, watermarks, and microprints, which rely on printed patterns that are destroyed or become visible on photocopied checks. These countermeasures have improved

detection of check forgery considerably. However, even if the reported success rate of these countermeasures is close to 84%, check forgery continues to be a widespread problem causing large financial losses every year [7].

However, the recent remote check truncation practice has completely bypassed these protection mechanisms by removing the very foundations they rely on – paper and ink. In particular, only print-based techniques, which rely on visual properties rather than on chemical and physical ones, may be potentially adapted as protection mechanisms, since those properties are preserved to some extent in digital check images. These techniques may depend on several factors, such as resolution and image quality, camera quality, and pattern quality. However, even though image forensics research to detect forged JPEG images exists [37, 23, 17, 33, 29, 28], the numerous challenges that need to be faced to adopt these ideas to digital check images have not received sufficient attention from the image processing community. Additionally, due to the recency of this practice, the development of new methods that exploit features of the digital domain have not received sufficient attention either.

3 Attack Description

In this section, we describe the advantages of digital check forgery over physical check forgery, which render the former much more likely to succeed than the latter, and a framework that leverages a wide range of image processing methods that can be used by an attacker to perform sophisticated forgeries.

3.1 Digital Check Forgery Advantages

The attacker's goal is to gain monetary gain via remote check deposit by either digitally modifying an existing check or by digitally forging a new check. In this paper, we do not consider the (trivial) case where checks may be modified physically and then remotely deposited. In fact, we believe that digital forgery is a lower hanging fruit for an attacker than physical check forgery, since it provides several advantages over physical forgery. These advantages are described next.

Precision. Digital image processing provides an attacker with the opportunity to manipulate an image at the pixel level with a level of precision unrivaled by physical forgery. Consider for example the amount area of an actual check shown in Figure 2 and produced by a (5MPixel) camera. The area measures approximately 3×0.8 cm in the physical check while the corresponding image measures 296×87 pixels for a total of 25,752 pixels at a bit depth equal to 24. Using digital image processing, an attacker can assign to each of those pixels any of the 16.8 million colors available at that bit depth. In reality, an attacker can only choose from a smaller set of colors that comprises only dark ones for the amount to show, however that subset is still a large one. To reach a similar precision level in the physical check, an attacker would have to be able to select and manipulate a region equal to $0.93 * 10^{-4} mm^2 = 93\mu^2$. With a scanning device of higher resolution, digital forgery can be even more precise. Furthermore, digital forgeries do not destroy the physical check. In particular, even though a physical attacker may not need the level of precision available in the digital

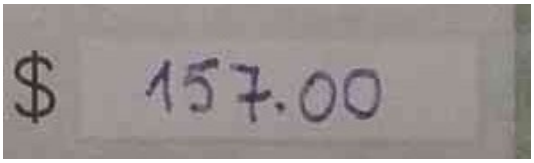


Fig. 2: Amount Area (Magnified 3x times.)

domain, a physical forgery may trigger countermeasures such as bleeding ink and chemically sensitive paper that would make the paper check unusable.

Unlimited Trial and Error. Since all forging operations are performed in the digital domain, an attacker has an unlimited power to revert them and return the image to the original state. Alternatively, by keeping copies of a check image file, an attacker has an unlimited number of images on which to perfect the forgery before submitting the check to the bank, thus minimizing the risk of detection. In the physical domain however, forgeries cannot be attempted more than once or twice on the same physical check without damaging it.

Absence of high fidelity trails for forgery detection. Recall that both the traditional check and the ATM transactions leave a paper trail that can facilitate forgery detection either in real-time (in case of traditional check transaction) or during post-clearance audit. As the remote deposit transaction does not leave a paper trail at the financial institution, however, none of the anti-forgery countermeasures described earlier can be used to detect forgeries.

Use of untrusted client check truncation systems. In the recent deployments of remote check deposit the *check truncation systems* have changed from trusted, tamper resistant, and well protected entities (e.g. teller centers or video surveilled ATMs) to untrusted (vulnerable to tampering) entities under an attacker's control. By modifying these components and their software, a determined attacker can interpose at any point along the path from the scanning device to the network device that sends the images and extract or inject forged images.

We assume that the attacker does not have any prior knowledge about specific image forgery detection mechanisms that may be in place on a target server. However, the attacker has a good knowledge about common forgery detection and counter-forensics techniques [19]. These techniques rely on the fact that almost all image forgeries leave characteristic artifacts in the resulting image. These artifacts may be detected by several passive detection techniques such as bispectral analysis, JPEG compression-based methods, etc [24, 34, 38, 27, 18]. To increase the chances for avoiding detection by any of these techniques, an attacker must use sophisticated forgery methods depending on the type of modification. These methods are described in §3.2.

In summary, the availability of *powerful*, *sophisticated*, and often *easy-to-use* digital image processing tools, the elimination of the *paper trail*, and the use of untrusted *client check truncation systems* contribute to the feasibility of this attack.

3.2 An Image Processing Framework for Digital Check Forgeries

The objective of an attacker is to conduct digital check forgery. To do this, the attacker will desire to introduce as few modifications to the original image as possible during the forging process. Therefore, the modifications must be carried

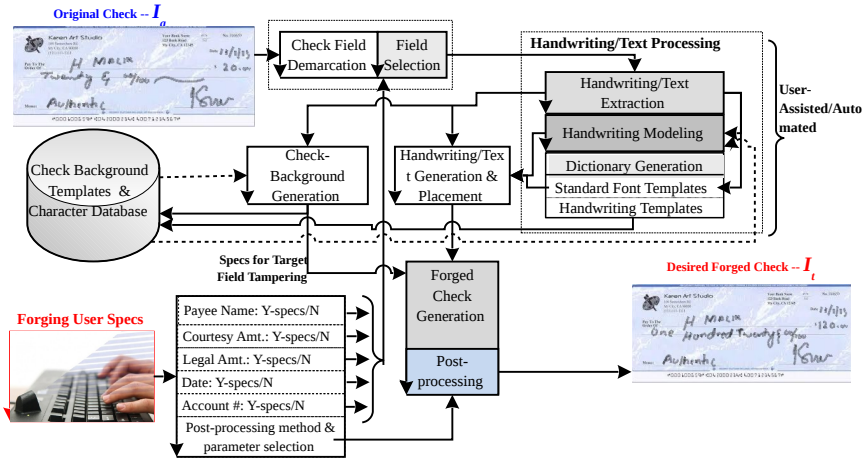


Fig. 3: A conceptual block diagram of digital forged check generation framework.

out in such a way that the "background" remains intact in the forged image, and only the fields targeted for tampering are isolated and altered. The design of a framework, outlined in Figure 3, is motivated by these objectives. In addition, although a variety of regions of interest exist on the check, the framework focuses on content alteration of five check fields: Payee name, Courtesy amount, Legal amount, Date, and Check number. The content of these fields consists of either handwritten or printed text.

Attack framework. The input to the framework consists of a rectangular image I_a of an original check and of forging user specifications, while the output is a forged image, I_t. These specifications identify the target fields, the type of alterations, and postprocessing method. For example, the Courtesy Amount of "*20.00*" and Legal Amount of "*Twenty & 00/100*" in the original image I_a may be targeted with a specification for alteration to "*120.00*" and "*One Hundred Twenty & 00/100*", respectively.

Field Demarcation and Field Selection (FDFS): This unit analyzes the input image I_a and demarcates the boundaries of the target check fields using a graphical interface and user input. For automatic field demarcation, an attacker can also take advantage of an automatic check reading system similar to [31].

Handwriting/Text Processing (HTP): This unit analyzes the text in the target check fields for text extraction, handwriting style modeling, and dictionary construction from handwriting and standard font templates. This unit is divided into the following three subunits:

Handwriting/Text Extraction (HTE) Subunit: This unit analyzes the selected fields for text extraction using methods based on digital image morphology [22] and attacker feedback. For example, a series of *dilation* and *erosion* operations along with user feedback are used for handwriting/text extraction process.

Handwriting Modeling (HM) Subunit: For handwritten target fields, preserving the handwriting style may help an attacker bypass eventual handwriting

Forgery Type	Processing Units/Subunits Involved
Check #	FDFS → HTP(HTE & DC)→ TGP → CBG → FCG
Date	same as above
Legal- & Courtesy-amount	FDFS → HTP (HTE, HM, DC)→ TGP → CBG → FCG
Payee Name	same as above
Signature	FDFS → HTP (HTE & DC)→ TGP → CBG → FCG
Fake Check Generation	HTP (DC)→ TGP → CBG → FCG

Table 2: Check Forgeries and Their Realizations using the Framework Units

verifications. To this end, this unit models the handwriting extracted from the input check image using active shape modeling as discussed in [21].

Dictionary Construction (DC) Subunit: This unit processes the target fields to extract a template for each character with the purpose of reusing them later. In particular, a series of image processing operations such as attacker-assisted segmentation, slant correction, size normalization, and thickness normalization is used for this task [31]. Character dictionaries for each victim and check type are stored in the database and later used to generate the text in the forged check.

Check-Background Generation (CBG): This unit "washes the check" by interpolating the pixels corresponding to the extracted text and filling them with values similar to the surrounding background. This operation can be executed with varying levels of sophistication, by using background check images stored in the database and employing a variety of super-resolution interpolation methods to make the washed pixels as similar to the background as possible [32, 36, 16].

Text Generation and Placement (TGP): The task of this unit is to generate and place new text in the target fields. The new text can be composed using existing characters saved previously in the dictionary or by using a template-based active shape modeling technique [21] backed by the handwriting model learned by the *HM* unit, thus preserving the consistency with the handwriting and fonts in the original check. In addition, other operations such as resizing, rotation, and spacing can be employed.

Check Background Templates and Character Database (CBTCD): The database stores the estimated check background templates, issuer-specific handwriting style models and text dictionaries. During the forged check generation processing, the TGP and FCG units request the database unit to provide information not readily available from the input image of the check, such as character templates previously extracted and check backgrounds.

Forged Check Generation and Post-processing(FCG): This unit is responsible for suppressing artifacts such as text or field boundary imperfections. The type of post-processing method (e.g. type of smoothing filter used) is provided in the attacker's input. It is worth highlighting that post-processing operations such as linear or nonlinear smoothing are likely to leave (statistical) traces themselves [19]. To get around such issues, an attacker can take advantage of counter-forensics methods, as discussed in [19].

Employing the Framework for Attacks. The proposed framework enables the attacker to perform a wide range of simple and sophisticated forgeries. Each forgery can be realized by using various features and framework units. For in-

stance, to modify specific fields of an existing check, an attacker can use the units *FDFS*, *HTP*, *CBTCD TGP*, *CBG*, and *FCG*, in that sequence.

Backed by a rich database of check and character templates, which can be populated over time, and by post-processing counter-forensic capabilities, more sophisticated forgery attacks are possible, e.g., generating a fake check digitally from scratch. We depict in Table 2 some specific instances of forgeries and how they may be executed by using the units of this framework.

4 Implementation

In this section, we describe the implementation of an instance of our attack for three *client check truncation systems* that run on the Android platform.

4.1 Library Instrumentation

The objective of library instrumentation is to achieve transparent interposition between the point where the check image is acquired and the point where it is sent over the network. The instrumentation described here is Android-specific but similar instrumentation may be applied to other implementations of the *client check truncation systems*. More specifically, it includes: 1) software modification with the purpose of analyzing the communications between the different application components, 2) identification of the interposition points where the original images can be extracted and where the forged images can be injected, and 3) implementation of the actual extraction and injection operations.

We highlight at this point that we deliberately treated the bank applications as black boxes for several reasons. First, we wanted to prove the generality of this attack and did not want to rely on particular implementation details of a specific application. Second, the EULAs of those applications specifically prohibit decompilation or modification of the binary or source code.

The *client truncation system* in Android lies entirely inside the device and includes the full software and hardware stack from the camera hardware to the bank application as depicted in Figure 4.a. As can be noted, the bank applications rely on the camera and network APIs during a check transaction.

In Android, the camera subsystem is implemented by the Java `android.hardware.camera` package and related C/C++ code residing in the lower layers, while the network APIs are implemented by several libraries, among which the Java Apache HttpClient library. To capture the operations during a check transaction, we introduced DEBUG output messages in several key points inside these libraries. Each message prints out the line of the source code that is being executed as well as the stack trace at that point. Using these instrumentations we gained a clear picture of how these libraries interact with the applications in the different steps of a check deposit transaction.

To take a picture, an application issues a request to the class `android.hardware.camera.Camera`. Inside this class, another private class is responsible for handling callbacks and ultimately forwarding the (JPEG) image data to the application inside an array of bytes. Next, the application processes the image and sends it to the network APIs to be delivered to the bank servers.

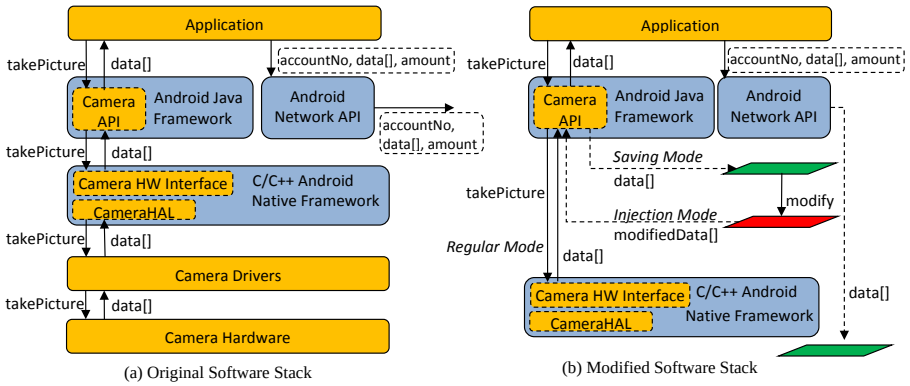


Fig. 4: The Original and Modified Camera Subsystems

Further instrumentation of the Camera and HttpClient classes allowed us to extract the original images being delivered to the bank applications and the processed images being sent over the network.

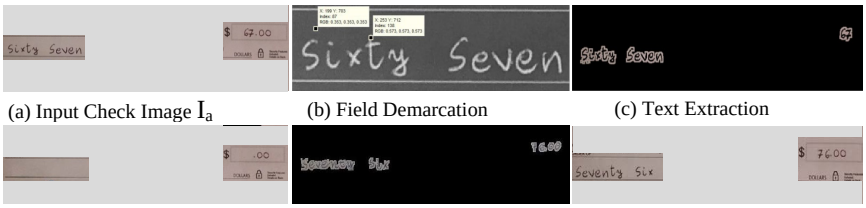
The previous analysis suggests two alternatives for the modified image injection point: 1) in the camera subsystem before the image is received and processed by the application, and 2) in the network subsystem, after the image is received and processed by the application and before it is encrypted and sent over the network. The latter alternative however poses a greater risk, since it may interfere with eventual image processing inside the application. In addition, not all applications use the Apache HttpClient library. Therefore, we chose to instrument the camera subsystem for injecting the forged image. The resulting system is depicted in Figure 4.b in the Appendix using dashed arrows.

Our instrumentation provides three different modes of operation for the Camera subsystem: 1) *Saving mode*, where a copy of the image data is saved as a JPEG file on local storage and the image data are forwarded to the application, 2) *Injection mode*, where the the image data are retrieved from a file on local storage rather than from the underlying layers, and 3) Regular mode, which is the original mode of the camera subsystem, where the image data are forwarded to the applications that use the camera. These modes can be enabled/disabled using a simple configuration file saved on the local storage of the phone.

We chose not to interfere with the applications' operations, in order to introduce as little disturbances as possible in the data received by those applications. For instance, since the applications request JPEG data rather than RAW data, we decided not to change the option to RAW. In fact, even though the RAW data returned by the camera may provide the original dataset to perform the forgery, a subsequent JPEG compression is still needed to pass the modified image to the application. If we do not know the parameters used for compression by the camera the subsequent compression (done by our framework) may be different from that performed by the camera, thus potentially disturbing the data.

4.2 Digital Check Forgery

For our proof-of-concept implementation, we decided to perform a light-weight forgery (due to the sensitivity of the experiments) by tampering only with the Legal- & Courtesy-Amount fields. This forgery is realized using a MATLAB



(a) Input Check Image $\mathbf{I_a}$ (b) Field Demarcation (c) Text Extraction

(d) Check Background Generation (e) User-assisted TGP (f) Forged Image $\mathbf{I_t}$

Fig. 5: Digital Check Forgery Steps

implementation of approximately 1100 LoC of the framework units *FDFS*, *HTP*, *TGP*, and *FCG* described in §3.2. A GUI was also developed to assist the *FDFS*, *HTE*, and *CBG* units with user input. The GUI visualizes the check and allows the user to provide an input vector consisting of the locations of the target fields and the post-processing method to be used along with its parameters.

More specifically, starting from the original check (Figure 5.a), the user-assisted *FDFS* unit selects the two fields (Legal-amount shown in Figure 5.b). Next, the *HTP* unit uses background subtraction and relative thresholding to identify the handwritten text in those fields (Figure 5.c). Next, assisted by the developed GUI, the *HTE* subunit directs the user to select portions of the field representing single characters and ultimately build a character dictionary of the text in the check. Next, for each target field, the user-assisted *TGP* unit sequentially selects the desired set of characters from the dictionary and places them in the selected field (Figure 5.e illustrates how the extracted characters for numerals '6' and '7' are used to generate '76' and how the words '**six**', '**ty**', and '**seven**' are used to generate '**seventy six**').

For each target field, the *CBG* unit "digitally washes" the check by replacing the pixels corresponding to the text with pixel values estimated from the neighborhood pixels (Figure 5.d). Next, the *FCG* unit merges the background image with the TGP-produced image to obtain the final image (Figure 5.f). Next, post-processing based on an averaging filter of $3x3$ pixels is used to mitigate boundary artifacts of the forgery. The final post-processing step uses Exiftool [10] to copy and update as necessary the JPEG Exif metadata from the original file to the forged file. For instance, the creation timestamp is modified to coincide with the injection time rather than with the capture time.

5 Experiments and Results

Experimental Setup and Application Descriptions. The experiments were performed on a Galaxy Nexus phone running Android 4.1.2 (Jelly Bean). The Android source files were downloaded from the official Google Android repos [11] and those implementing the camera and network APIs were modified as described in the previous section. Next, a *userdebug* build, which provides root access, was flashed into the phone.

Banking Applications Description. After a user logs in, each application presents a screen with the check and instructs the user to take pictures of its front and back. Next, the user is required to select the account # where the

	Captured(quality/size dimensions/metadata)	**Transmitted**(size dimensions/metadata)	Obfuscated Code
Bank 1	95/700KB/1600×1200px/Exif	80KB/1600×1125px/JFIF	No
Bank 2	70/290KB/1600×1200px/Exif	290KB/1600×1200px/Exif	Yes
Bank 3	30/210KB/1600×1200px/Exif	80KB/ N/A	Yes

Table 3: Preliminary Analysis Results

check must be deposited, type in the amount, and finally submit the check. Up to the final submission step, the transaction may be canceled at any time by the user. The application transmits the two images and the data submitted by the user to the server using the network APIs. On the server side, optical character recognition (OCR) software is used on the check's areas of interest and a confirmation message is sent back to the user.

5.1 Preliminary Experiments and Analysis

Before carrying out the actual attacks, several preliminary experiments were performed to gain an understanding of: 1) the properties of the images captured by the camera and of those sent to the server as well as applications' features, 2) the server side operations, in particular tolerance to errors, picture quality, OCR capabilities, and possible human involvement in the clearing process.

Image Properties and Application Features. Using the instrumented libraries, we initiated several transactions with untampered checks, most of which were aborted by us before the final submission. Four transactions were instead brought to completion targeting two banks (two transactions per bank).

The results of these experiments are outlined in Table 3. In this table, the first column represents the bank, the second represents the JPEG quality, approximate file size, dimensions, and the metadata format of the images (Exif or JFIF) captured by the camera while the third column represents the same information about the images transmitted to the servers. Finally, the fourth column shows the applications that use code obfuscation (discovered by inspection of the stack traces). As can be noted, Bank 1 compresses the image before sending it to the server, presumably to save bandwidth. In addition, its Exif metadata are replaced by JFIF metadata. Bank 3 instead retrieves a low quality image from the camera from the start. We could not capture the transmitted images for Bank 3 using our Apache HTTP instrumentation. However, we observed a total encrypted network traffic equal to approximately 80KB per image, suggesting that the images are sent over the network via some other mechanism. In addition, we discovered that OCR is performed on the smartphone as well.

The practices of sending low quality images have important consequences on the server's side ability to detect forged images. In fact, while the regions of interest can still be processed successfully by OCR software, the loss of information in the high frequencies present in low JPEQ quality images makes detection of artifacts introduced by forgeries hard to detect. Indeed, pixel values tend to be more uniform across larger regions giving images blocky features.

Server Operations and OCR. In a first experiment, a different amount from the one written on the check was entered during the user interaction step. In

	Preliminary Experiments	Transformation Type	Success
Bank 1	Wrong Amount/names	Block Swapping (amount)	YES
Bank 2	Wrong Amount	Block Swapping (amount)	YES
Bank 3	Wrong Amount	Block Swapping (amount)	YES

Table 4: Attack Results

this case, the server recognizes the mismatch and prompts the user to enter the amount again. If a different amount is entered again, the server temporarily accepts the amount entered by the user. Ultimately, however, the amount written on the check is deposited in the user's account. The results of this first experiment suggest that no OCR is being performed on the client and that in the case of a disagreement between the amount entered by the user and the amount recognized by OCR, there is human intervention in the *check clearing system*. In a second experiment, misspelled names were written in the handwritten name portion of the check. In this case, the transaction proceeded without glitches, suggesting that OCR is not performed on the handwritten name portion of the check.

The results of these experiments suggest that the *check clearing system* is configured to be tolerant towards errors, to the advantage of attackers. Indeed, given the wide variety of checks, lighting conditions in which pictures may be taken and cameras, it would be difficult to set strict parameters for picture quality and JPEG characteristics on the server side.

5.2 Forging Attacks and Results

Three checks with small amounts were modified and injected in each application. The modifications reused the characters of the original check as outlined in Table 4 and described in § 4. In our experiments, the *payer* and the *payee* were the same person and the accounts were different accounts of that person on different banks or within the same bank. The forged checks that were injected into the three applications were cleared without glitches within a business day.

By connecting the phone to a computer before a check transaction and switching among the different modes of operation described in the implementation section, the attack proceeds as follows.

Acquisition. The camera subsystem is set in *saving* mode. A remote check transaction is started in the banking application and a picture of the check front is taken. The byte array with the image data is saved as a file on the local file system, in addition to being forwarded to the application. At this point, the transaction is canceled to avoid sending the original image to the server.

Digital Forgery. The saved image is pulled from the phone using `adb` and, using the procedure described in the implementation section, it is modified in Matlab and pushed back on the local file system of the phone.

Injection. The camera subsystem is placed in *injection* mode and a check transaction is started in the app. When taking the picture of the check front, the modified image is loaded from the local file system as a byte array, which is forwarded to the application, while the byte array corresponding to the real image is blocked. Next, the camera subsystem is placed in *regular* mode and the picture of the check back is taken. Finally, the images are submitted to the server.

This attack takes approximately ten minutes, most of which spent by the user in pushing and pulling the image from the smartphone and in providing input specifications to Matlab's framework. In our experiments, the percentage of changed pixels was in average equal to 0.43% of the total number of pixels.

We note that due to the sensitive nature of the evaluation, the nature of various experiments we conducted were "light-weight" and our results have to read in that light. More experiments and different forgeries are technically possible (forging account numbers, creating checks from scratch, using larger amounts), but have not been tested against any possible mitigation strategies currently employed by the banks due to their sensitive nature. More such experiments are needed to be done in collaboration with the banks to study the feasibility of these advanced attacks.

After the attacks, we contacted the banks and provided them with an earlier draft of this paper, nearly 5 months before submission of this paper. The banks have acknowledged the problem of digital forgery and are actively working to design countermeasures. We also shared our preliminary ideas regarding countermeasures which we discuss below.

6 Countermeasures

In this section, we discuss some countermeasures that can be employed on the client and server sides of a remote check transaction system, to prevent or detect digital check forgery attacks. We intend to provide a high-level discussion, and note that our treatment of this topic is not comprehensive due to space reasons.

Trusted Computing. Trusted computing solutions have been proposed and implemented against client side tampering in a wide range of devices [25, 12, 13]. Using a trusted computing platform, the image data or the sensitive portions of the image (amount, account number) may be digitally signed by a hardware-based tamper-resistant and trusted module on that platform before being released to the upper layers of the OS. As an example, the OMAP 4 hardware platform on the Galaxy Nexus phone used in our experiments provides capabilities to implement a trusted computing module on this phone [14]. To take full advantage of these capabilities, however, the applications must be modified to interface with the trusted module. In addition, such a solution would not protect against attacks that modify the physical check before scanning it.

Check Reconciliation, Positive Pay and Transaction Limits. The oldest technique that is an effective method for consumers to prevent check fraud is reconciliation, wherein a consumer-kept record of the check issue is matched against the actual transaction record. However, it appears that a large fraction of users do not reconcile their accounts [15]. Motivated by this, *Positive Pay* is a common countermeasure used to protect against check forgery, developed primarily for businesses. In positive pay, the *payee* sends copies of the issued checks to the bank daily, thus providing a reference copy of the original check. However, due to its cost this countermeasure is currently used only by corporations and companies and even then, according to [7], 23% out of 5700 surveyed companies do not use it due to the costs involved.

A current countermeasure to reduce risk is placement of check, daily, and monthly amount limits on remote check transactions for end users. However, this measure, while reducing eventual harm, does not prevent the attack from occurring. In addition, this measure seems to be applied in practice only to individual customers and not to business customers.

Digital Image Forensics Techniques. Digital image forensics techniques may be utilized to detect check forgeries and raise the difficulty bar for attackers. We list some of these techniques below:

- **Camera fingerprinting**. Recent research shows that device sensor irregularities affect the pixel values of the images and the pattern of these irregularities is unique to each device [30, 26]. These unique camera characteristics may serve as or be used to derive unique watermarks to add to the target check fields and thus detect images that are not produced by the devices. This method, however, requires the servers to obtain a set of digital images produced by a device to derive the watermarks (for instance, by taking a set of pictures and sending them when a check truncation application is first started). A skilled attacker, however, may defeat this countermeasure by providing an initial set of pictures with the same watermarks as (future) forged images.
- **Copy-evident images**. This technique introduces special markings in JPEG files that are invisible to the naked eye but become visible when an image is recompressed under certain conditions [28]. The special markings, however, need to be introduced before an attacker extracts the image, ideally by the camera hardware. Combined with a list of server-approved JPEG quantization tables to reduce the freedom of attackers in manipulating the images, this technique may significantly raise the bar of difficulty for attackers.
- **Double JPEG Compression**. Several techniques have been proposed to detect JPEG artifacts due to a second JPEG compression [23, 17, 20, 33, 29]. However, the existence of double compression alone is not sufficient to detect forgeries and, as seen in our experiments, images may be recompressed by some of the apps. Furthermore, recent research on anti-forensics shows that some of these techniques can be defeated [35].

High quality images. To further improve the chances of detection on the server side, high quality images may be sent by the applications. This countermeasure is simple to implement, however, it must be accompanied with the deployment of appropriate forgery detection mechanisms on the server.

7 Conclusion

In this paper, we presented and analyzed a digital check forgery attack that can be used against client check truncation systems. This attack is enabled by the delegation to untrusted entities of critical operations performed by trusted entities. We demonstrate the feasibility of an instance of this attack with experiments on three banking applications running on Android smartphones. We also discussed countermeasures that can be employed against this attack.

References

1. http://www.fdic.gov/consumers/consumer/alerts/check21.html.
2. http://www.marketsandmarkets.com/PressReleases/merchant-remote-deposit-capture.asp.
3. http://www.pymnts.com/briefing-room/mobile/playmakers/2013/how-popular-is-mobile-check-deposit/.
4. http://www.forrester.com/The+State+Of+Mobile+Banking+2012/fulltext/-/E-RES75582.
5. http://www.theguardian.com/money/2013/dec/26/banks-smartphone-photos-cheques.
6. http://www.canadianlawyermag.com/legalfeeds/1640/canadian-banks-get-ok-to-move-ahead-with-image-cheque-deposit.html.
7. http://www.afponline.org/fraud/.
8. http://www.aba.com/Products/Surveys/Pages/2013DepositAccount.aspx.
9. http://www.ffiec.gov/exam/check21/faq.htm#General.
10. http://www.sno.phy.queensu.ca/ phil/exiftool/.
11. http://source.android.com/.
12. http://www.trustedcomputinggroup.org/.
13. http://nelenkov.blogspot.com/2012/07/jelly-bean-hardware-backed-credential.html.
14. http://ti.com/m-shield.
15. http://www.moebs.com/AboutUs/Moebsinthenews/tabid/57/ctl/Details/mid/484/ItemID/26/Default.aspx.
16. Gholamreza Anbarjafari and Hasan Demirel. Image super resolution based on interpolation of wavelet domain high frequency subbands and the spatial domain input image. *ETRI J*, 32(3):390–394, 2010.
17. Tiziano Bianchi and Alessandro Piva. Detection of nonaligned double jpeg compression based on integer periodicity maps. *IEEE Transactions on Information Forensics and Security*, 7(2):842–848, 2012.
18. Gajanan K. Birajdar and Vijay H. Mankar. Digital image forgery detection using passive techniques: A survey. *Digital Investigation*, 10(3):226 – 245, 2013.
19. R. Bohme and M. Kirchner. *Digital Image Forensics: There is More to a Picture Than Meets the Eye*, chapter Counter-forensics: Attacking Image Forensics, pages 327–366. Springer-Verlag, 2013.
20. Chunhua Chen, Yun Q. Shi, and Wei Su. A machine learning based scheme for double jpeg compression detection. In *ICPR*, pages 1–4. IEEE, 2008.
21. A. Chowriappa, R.N. Rodrigues, T. Kesavadas, V. Govindaraju, and A. Bisantz. Generation of handwriting by active shape modeling and global local approximation (gla) adaptation. In *International Conference on Frontiers in Handwriting Recognition (ICFHR), 2010*, pages 206 – 211, 2010.
22. Nick Efford. *Digital Image Processing: A Practical Introduction Using Java^{TM}*. Pearson Education, 2000.
23. H. Farid. Exposing digital forgeries from jpeg ghosts. *Information Forensics and Security, IEEE Transactions on*, 4(1):154–160, 2009.
24. H. Farid. A survey of image forgery detection. *IEEE Signal Processing Magazine*, 2(26):16 – 25, 2009.
25. Eimar Gallery. *An Overview of Trusted Computing Technology*. IEEE, 2005.
26. Miroslav Goljan and Jessica Fridrich. Sensor-fingerprint Based Identification of Images Corrected for Lens Distortion. In *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security and Forensics 2012*, 2012.

27. R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, 3rd edition, 2007.
28. Andrew Lewis and Markus Kuhn. Towards copy-evident jpeg images. In Stefan Fischer, Erik Maehle, and Rdiger Reischuk, editors, *GI Jahrestagung*, volume 154 of *LNI*, pages 1582–1591. GI, 2009.
29. Bin Li, Y.Q. Shi, and Jiwu Huang. Detecting doubly compressed jpeg images by using mode based first digit features. In *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, pages 730–735, 2008.
30. J. Lukas, J. Fridrich, and M. Goljan. Digital camera identification from sensor pattern noise. *Information Forensics and Security, IEEE Transactions on*, 1(2):205–214, 2006.
31. R. Palacios and A. Gupta. A system for processing handwritten bank checks automatically. *Journal Image and Vision Computing*, 10(10):1297–1313, 2008.
32. Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *Signal Processing Magazine, IEEE*, 20(3):21–36, 2003.
33. Toms Pevn and Jessica J. Fridrich. Detection of double-compression in jpeg images for applications in steganography. *IEEE Transactions on Information Forensics and Security*, pages 247–258, 2008.
34. A. Piva. An overview on image forensics. *ISRN Signal Processing*, 2013:22 pages, 2013. Article ID 496701.
35. M.C. Stamm and K.J.R. Liu. Anti-forensics of digital image compression. *Information Forensics and Security, IEEE Transactions on*, 6(3):1050–1065, 2011.
36. Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *Image Processing, IEEE Transactions on*, 19(11):2861–2873, 2010.
37. Hang Yu, Tian-Tsong Ng, and Q. Sun. Recaptured photo detection using specularity distribution. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 3140–3143, 2008.
38. L. Zhou, D. Wang, Y. Guo, and J. Zhang. Blur detection of digital forgery using mathematical morphology. In *Lecture Notes in Computer Science*, volume 4496, pages 990–998. Springer Berlin / Heidelber, 2007.