Remote Proofs of Video Freshness for Public Spaces

Junia Valente, Alvaro A. Cardenas
Erik Jonsson School of Engineering & Computer Science
The University of Texas at Dallas
Richardson, TX
{juniavalente, alvaro.cardenas}@utdallas.edu

ABSTRACT

We propose the use of trusted and verified social media feeds as visual challenges to increase our confidence that video footage from public spaces is fresh and authentic. Our work is related to recent advances in a growing area dealing with ways to prove physical statements to a digital (or even human) verifier, where a verifier sends a physical (real-world) challenge to the prover and the prover (usually a sensor) takes measurements of the physical property and submits the response to the verifier. Our proposal can be used to automatically verify the video feed from a (possibly untrusted) camera monitoring a public space.

CCS CONCEPTS

• **Computing methodologies** → **Scene anomaly detection**; • **Security and privacy** → *Authentication*; • **Applied computing** → *Surveillance mechanisms*;

KEYWORDS

IoT; Visual Challenges; Video Authentication

1 INTRODUCTION

Surveillance cameras are used in a variety of settings, including the monitoring of uranium enrichment facilities to verify that countries abide by the Nuclear Non-Proliferation Treaty [37], monitoring access to certificate vaults protecting secret keys [8], monitoring access to the computers generating random numbers for the lottery [9], and monitoring electricity substations [30]. As the sensitivity of usage scenarios and pervasiveness of security cameras increase, we need to ensure they are protected by defense-in-depth mechanisms to ensure that captured images are fresh and authentic.

In a typical Hollywood bank heist film, attackers hack the security cameras and replay old footage so that security guards are not able to see the attack taking place. While Hollywood films are not known for their authenticity, this particular camera hacking scenario is becoming more realistic as the interest for hacking cameras and the expertise of attackers continue to increase. In a recent example, the cameras that monitored access to computers generating random numbers for a lottery system recorded only one second per minute rather than running continuously like normal and prosecutors argue that the defendant tampered with the camera equipment to have an opportunity to insert a thumb-drive into the computers without detection [9]. Similar Hollywood-style attacks have been demonstrated in practice [11], showing how to freeze the current frame on the video administrator panel.

These examples show the need for having multiple-defense mechanisms for security cameras. We argue that once an attacker has compromised the secret keys or the root-of-trust of an embedded device, the attacker can bypass most of the traditional integrity mechanisms that the receiver of the message can use to verify its authenticity. In addition, traditional security mechanisms based on secret keys are not equipped to detect physical attacks, like moving the cameras to point to a different place [48].

In this paper we extend our previous work leveraging random challenges in the visual field of the camera [49] with the design of minimally invasive, frequently-updated challenges for cameras in public spaces. In particular, we propose and implement a prototype of the use of tweets by a large pool of verified users in order to provide the continuous challenge to be displayed in the screen.

Contributions. Our contributions over our previous work include: (1) the design of a minimally invasive system to the public that deploys a continuous stream of trustworthy-generated challenges used to verify the freshness of a video stream, (2) an improved description of the adversary threats to this system and applicable to the more general problem of physical challenges (as discussed in Section 2), and (3) an analysis showing how our design improves the security of other proposals that use physical challenges.

Note that because we are not sending the challenge to the device but to the physical world, our approach works for legacy systems. In addition our proposal is useful for non-security situations, such as for providing tools to automatically detect when a digital signage is malfunctioning or gets damaged.

While our proposal requires only two devices to be added, a display and a verifier, and it can work without the need to change any parameter in an already deployed system; our main scenario is to partially borrow already deployed display or digital signages (such as those shown in Figure 1) to send the random challenge at periodic intervals.

CPS-SPC'17, November 3, 2017, Dallas, TX, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5394-6/17/11...$15.00
https://doi.org/10.1145/3140241.3140242

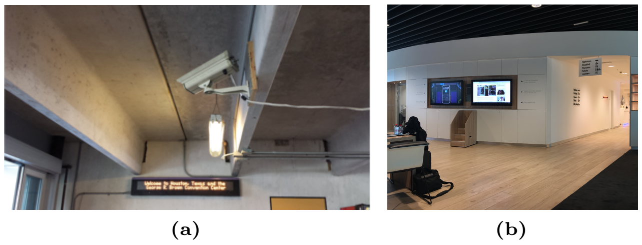


Figure 1: Figure 1a shows an LED display in the field of view of a camera in a parking structure. Figure 1b shows displays in the field of vision of surveillance cameras in an airport.

The paper is organized as follows: In Section 2 we summarize related work. In Section 3 we introduce the possible threats against our system. Section 4 describes the high-level design of our system, including how to guarantee two security requirements for our system: (1) how to obtain a pool of Twitter-verified trustworthy tweets, and (2) how to keep this pool of tweets continually active, because we need to send continuously challenges to guarantee freshness. Section 5 shows experimental results showing the performance of our system against the first four types of threats considered in our model. Section 6 describes the security analysis of our proposal and its limitations, and finally Section 7 concludes the work and discusses future research.

2 RELATED WORK

Liveness Checks: Our work is related to ongoing efforts for using camera phones for user identification or authentication purposes [2, 15, 16]. A new feature called *liveness check* enabled Android Jelly Bean users to unlock their devices by blinking while unlocking. Screen lock features such as "Face Unlock" are categorized as providing low security in comparison to pin numbers or passwords, so it was no surprise when researchers showed a simple solution to bypass the liveness checks [23]. More recently, companies are investing on innovative ways to use facial recognition technology to replace passwords with selfies and videos [15, 16]. One of the proposed verification process is based on having a device prompt the user to perform certain actions, motions, or gestures (e.g., smile, blink, tilt head) [32]. This is similar to the intuition behind our verification system: we propose that if the output of a camera does not reflect changes introduced in front of the camera, then there is an indication that the camera is not truthfully reporting fresh footage. In their case, the verification assumes the camera to be trustworthy and simply fails to authenticate the user; in our case, we are able to detect tampering in the images captured by the camera. To spoof a liveness detector, an adversary only needs a small number of images (e.g., an image of the person it is trying to spoof with both eyes open, and then an image of the person it is trying to spoof with a wink) to launch the attack. One of our goals in using random tweets is that they provide a reasonably amount of randomness with each new challenge, preventing these types of attacks.

Visual Channels to Share Authentication Information: Our research is superficially related to work that leverages side-channels to convey security relevant information to users [18, 29]. In these cases a photo or a video used to exchange information needs to be trusted, and this trustworthiness rests on the premise that humans are present (i.e., they can see with their eyes the same image the camera is capturing) and will be able to spot any spoofs on the image. Our premise focuses precisely on the opposite assumption: we assume no human in charge of verification is co-located with the camera (in fact, for most of the time our verifier is an algorithm and not a human), and furthermore, our threat model assumes the attacker is able to compromise the visual channel to send false image frames and launch replay attacks so that the footage might not look suspicious even to security guards. Also, because we do not send the challenge to the camera (but to the physical world in front of the camera) the camera does not even need to know about the attestation protocol nor stop its normal operations to perform any attestation code (as in the case of McCune et al. [18]). Our proposal can be used to improve security protocols that rely on a trusted visual channel to be secure, but our proposal is also more general and applicable to other cases.

Proving Physical Statements: The closest related work to our own contributions is the growing area of research leveraging physical challenges to prove properties about a system. The ability to prove *physical* statements to a digital verifier is an area of research that has attracted recent attention [5, 7, 14, 20, 26, 27, 31, 33]. In its general form, the problem is a challenge-response protocol, where the *verifier* sends a challenge to the *prover*, the prover (usually a sensor-equipped device) takes measurements of a physical property, and then submits a response to the verifier with information to help convince the verifier of the validity of a physical property captured by the sensor. The interaction between a verifier and a prover can have one, or multiple rounds of challenges and responses. In most cases these mechanisms do not rely on classical secret keys or any tamper-resistant security hardware and can therefore serve as backup in situations where the secret keys have been compromised, or in cases where provers do not have secret keys readily available to them [27].

While all these contributions have advanced our understanding of proofs of physical properties, they have some limitations. Some of them impose overhead for continuous monitoring of a sensor signal, for example zero knowledge proofs have overhead that might not be needed in most practical settings where we just want to verify that a sensor reading is correct, and practical proposals built with Physical Unclonable Function (PUF) structures still need to pre-record and privately store operations of the PUF under a variety of sensor conditions. Proposals focusing on continuous monitoring of a sensor signal [20, 31] generally assume a weaker form of adversary model that has not compromised the proving device or cannot serve as a man in the middle [31] (they only

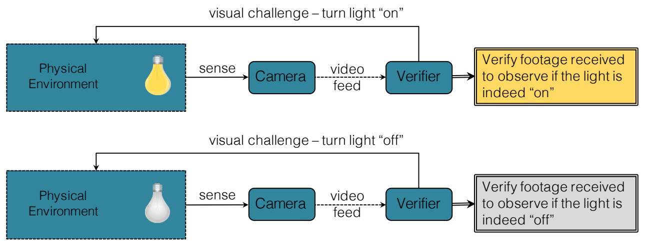

visual challenge – turn light "on"
Physical Environment
sense
Camera
video feed
Verifier
Verify footage received to observe if the light is indeed "on"
visual challenge – turn light "off"
Physical Environment
sense
Camera
video feed
Verifier
Verify footage received to observe if the light is indeed "off"

Figure 2: An intuitive idea of a *challenge* to verify the integrity of camera devices includes: have a light on sight of camera that turns on or off at the discretion of verifier. Then, the verifier can use image processing techniques to observe if the light indeed is on (or off) in the next image frame.

consider physical attacks) or that do not capture the specific instantiation of the challenge sent by verifier [20].

In this paper we focus on providing tools to improving the authenticity of video footage. Our **proposal has a unique combination of properties:** (1) it can be used to automatically verify sensor data continuously (in real-time), (2) humans can also verify the physical challenge seen in the video (this property is unique to our system as all previous work assumes that the verifier is trusted, but in our case the human can check–although not continuously–if the verifier is working properly), (3) the verification can be done remotely, (4) our system can be deployed in public spaces as the physical challenge is designed in a way that is minimally disruptive to potential bystanders, (5) our results show that our proposal is resilient to a large class of adversaries with the same resources generally assumed in previous work [20, 31, 49] (including physical and cyber-attacks) but we also discuss stronger adversaries that can bypass ours and previous related work, and (6) the prover does not need to be aware that the verifier is challenging it, because the challenge is sent to the physical world rather than to the prover itself, our approach can be used in "legacy" sensors (provers).

In particular we propose the use of fresh Twitter feeds to display in a digital signage located in the field of view of the camera whose video feed we want to verify remotely. In addition to security purposes, our proposal can also be used to automatically detect as soon as possible when digital signage malfunctions, and to monitor that ads (i.e., ads instead of tweet messages) have been shown in digital signage locations at the appropriate times and according to a previously agreed contract.

3 THREAT MODEL

To illustrate the basic intuition of our approach and to motivate discussion on the adversary model, let us consider that a remote verifier can turn on and off a light bulb in a room where a camera is located, as illustrated in Fig. 2. If the verifier turns the light on, the verifier can check if the video received has the light on, and similarly, if the verifier turns the light off, it can check if the video received has the light off.

While this approach might prevent simple physical attacks or Hollywood-style attacks where the attacker replays old footage, it cannot thwart an attacker that knows our system is in place and who can tap to our channel sending the challenge. To bypass our system the attacker only needs to capture one frame with the light bulb on, and one frame with the light bulb off, and then replay one of these frames based on the challenge it observes.

This leads us to the discussion of adversary models for continuous monitoring proposals. While previous work has focused on physical and analog-only attacks [31], or on attacks where the adversary cannot adjust their response to each particular challenge [20], in this work we also consider an attacker like the one described in the previous paragraph who will try to respond with a frame corresponding to the challenge that was sent.

In particular, we consider the following adversary threats in increasing severity: **Physical Attacks:** This is an attacker that can attempt to damage the camera physically, cover it, or move the camera to point to a different place [48]. Notice that these types of attacks can be launched even if the adversary does not have the cryptographic material the camera uses to authenticate itself and the video it sends. Notice also that while a human-verifier can also detect these attacks, our goal is to develop automated video processing algorithms to detect these type of attacks automatically.
Spoofing Attacks: This corresponds to an attacker that has compromised the authentication credentials of the camera and can authenticate itself to the prover, but sends a video feed of a different location. Notice that the detection of this attacker is equivalent to the detection of physical attacks, as our video processing algorithms will look for the display of the challenge in the video feed but will not find it.
Replay Attacks: This attacker has compromised authentication credentials of the camera, and decides to send a replay attack (e.g., an attack of the video from the day before).
Smart Replay Attacks: This attacker has compromised the authentication credentials of the camera, and knows our system is in place. This smart replay attack will keep replaying an old image frame until we send a new challenge. After capturing a real image frame of the new challenge it will then continue replaying that image until the next challenge.
Compromised Verifier Attack: This attack assumes that the attacker instead of compromising the prover (as it is traditionally assumed) compromises the algorithm that processes the video, and reports that no attack is present regardless of the video footage shown.
Anti-forensics Attacks: This attacker knows our system is in place, and in addition, it attempts to create a forged image containing the random challenge. Moreover, it attempts to create the video footage with anti-forensic tools to prevent the verifier from using video forensic algorithms.

In this paper we show that our proposal is strong against the first four types of attacks, it is better than related work using physical challenges that has never considered compromised verifiers, and it is weak (and potentially broken) against the last attack, the final score will depend on advances in multimedia forensics and anti-forensics (however we argue that the powerful anti-forensic attacker has never been considered before in previous related work and that previous proposals in that space are also broken against this type of adversary; in their papers they consider this adversary outside of their scope).

4 DESIGN CONSIDERATIONS

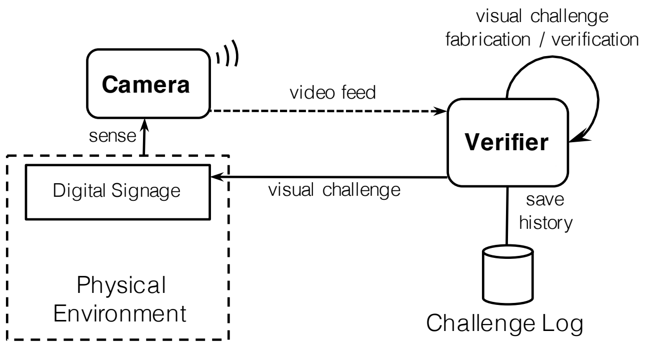


Figure 3: (1) A visual challenge is sent to a display, (2) the camera captures a video of a scenery including the display, (3) the verifier retrieves the next video frame, and (4) verifies the challenge in the video frame: if the verifier confirms the challenge in the frame just received, then it gains confidence that the camera is transmitting fresh and authentic footage.

As discussed in the previous section, we cannot send binary challenges (like light bulb on or off) because these would be fairly easy to bypass by the *smart replay attacker.* In particular, we want to minimize the chances that the same challenge is sent repeatedly to the field of vision of the camera, and we also need a device in the field of vision of the camera that can be used to display diverse challenges.

A digital signage or a LED display can be used to display text (or images) sent as challenges by the verifier. Our system design is illustrated in Fig. 3. The verifier can send random passphrases to be displayed in the digital signage. However, this approach would not work well for a video camera monitoring a public space. In order to implement our system in a public location, we would need the digital signage or LED display to show meaningful information to people in the area.

To address these concerns we propose to study challenges composed by intelligible texts generated and extracted by various sources online such as social media content. One of the reasons why we are considering social media content is because they are informative, popular, and look visually more appealing than random texts or barcodes. Also, social media content has been increasingly used to broadcast fresh, authentic information (e.g., consider trending topics, news updates): on Twitter, there are 500 million tweets on average per day, corresponding to about 5,700 tweets per second [47]; and there are 80 million new photos each day on Instagram [13] and 350 million on Facebook [3].

In this paper we explore the use of *tweets* as challenges. In our basic idea the verifier (1) selects a **random, recent** tweet from a **verified** (and reputable) source (to prevent social media attacks [12]) and sends this tweet to be displayed in the digital signage, (2) receives the video feed from the camera (prover), (3) using optical character recognition, it extracts the displayed tweet from expected display location in the image frame, and (4) verifies whether the extracted tweet is the correct one (or close enough to the correct one).

In the next subsections we will discuss (1) how to obtain a pool of Twitter-verified trustworthy tweets, (2) how to guarantee there is enough randomness, and (3) how to increase the refresh rate of the challenge to improve the freshness guarantees while providing the same level of usability in public spaces.

4.1 Trustworthy Tweet Selection

We assume the Twitter server to be trusted. For example, we assume the attacker cannot manipulate the Twitter user base to return only tweets known in advance to the attacker, and we assume the attacker is not able to manipulate the source of tweets to send only tweets from malicious Twitter users. To enforce this in practice, we consider using tweets only from trusted sources such as Twitter verified users. While attackers may compromise individual verified users, our design will present multiple tweets from multiple verified users with the hope that not all of the randomly-selected verified users used in our challenge have been compromised.

We propose to use Twitter APIs [44] to collect real-time tweets and provide the verifier with a large pool of fresh tweets. New tweets are continuously pushed to the pool of tweets such that there is always enough tweets for the verifier to randomly choose from as the next visual challenge.

While the verification process is running, the tweet pool needs to be maintained (or monitored by itself or another entity) to ensure the freshness and randomness properties for visual challenges. Thus, tweets are temporarily stored in a *self-maintained* storage where the tweet pool is periodically updated to delete unused tweets as they become old and to add new tweets as they become available.

Preliminary Requirements. The requirements necessary for fabricating tweet visual challenges include: (1) `User list`: creating a list of Twitter users who frequently post new tweets. Our user list consists of 337 verified users—mainly news accounts because they show content with wide appeal and tweet out frequently. (2) `Pool creation`: automatically creating a tweet pool by storing tweets from users (from the user list) as they tweet. (3) `Self-maintaining pool`: add new tweets as they become available and auto expire old unused tweets to maintain the pool with enough, fresh tweets. We use MongoDB's time to live feature to auto expire data.

Architecture. We show our proposed architecture in Fig. 4. The most important component is the verifier. The verifier is responsible for verifying the camera feed in real-time as footage is captured by the camera. The verifier retrieves the

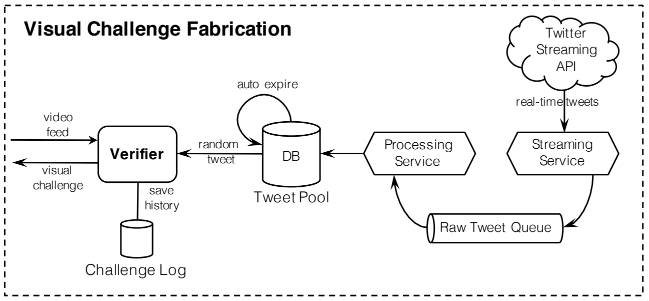


Figure 4: We use the Twitter API to collect real-time tweets and use them as visual challenges.

next random visual challenge (e.g., tweet), then the visual challenge is displayed in the digital signage. When the verifier then receives the next image frame, it verifies that the image frame contains the correct visual challenge. If the feed does not contain the correct challenges over a time window, the verifier will raise an alarm.

Another important component is the tweet pool database. This pool is dynamic, and is constantly updated by either adding or deleting tweets. New tweets are added in real-time as new tweets arrive for users from our list of users. Tweets are deleted in two possible ways: (1) tweets auto expire after a specified amount of time; or (2) tweets are automatically deleted each time they are used as a visual challenge by the verifier. Other components such as the processing and streaming services are responsible for receiving tweets from the Twitter APIs, and processing and saving the tweets to the tweet pool where the verifier can query on demand.

4.2 Aesthetics vs. Security

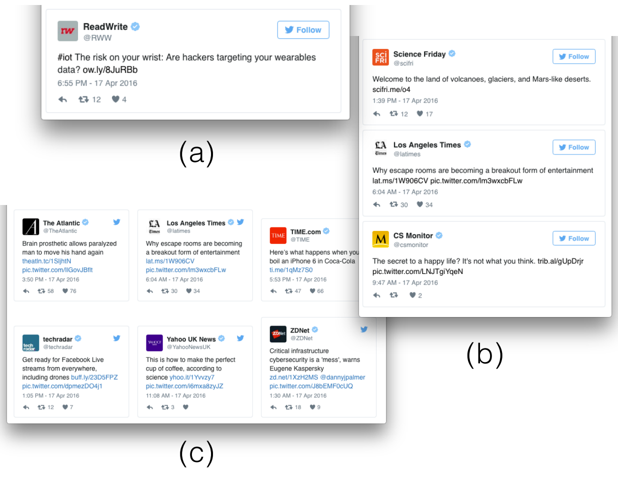
Bystanders close to the digital signage may notice the visual challenges in the digital monitor and may briefly pay attention to them as they are displayed and refreshed in the display. For this reason, we consider looking at the trade-offs between the usability of displaying tweets (visual aesthetics and look-and-feel of the visual design) and the security for preventing attacks (increasing the refresh rate of tweets).

An advantage of using social content such as tweets is that tweets by themselves already convey a message and look familiar to most people. In our implementations we display tweets using best practices from the Twitter brand standards [45] (e.g., use of the elegant and approachable Gotham font family, leaving a 150% safety white space around the Twitter logo) and we acknowledge that in practice the selection of tweets should be carefully tailored to a particular audience to improve the acceptability of this technology. Next we provide a brief discussion on the pros and cons for each proposed visual design shown in Fig. 5.

Single Tweet. One of the design considerations we study, is the *refresh rate* for displaying a single visual challenge at a time. In particular, we aim to keep a low refresh rate to allow more time for a bystander to read the tweets being refreshed in the monitor. *Security question:* what happens when we hold the visual challenge longer? An attacker can use the created image for the next time steps of the camera footage. Therefore, an attacker can have enough time to learn, and launch attacks for the remaining time the challenge remains in the monitor. The following designs tackle this limitation.

List of Tweets. Displaying a list of tweets might be the most natural way for users to engage with the digital monitor. An advantage is that we can decrease the time a tweet stays up. In particular, when we use multiple tweets and update only one at a time: the overall visual challenge (e.g., the composition of all tweets) will be frequently updated, while it will still take longer for an individual tweet to disappear. This gives more time for people to read the tweets (if they wish to do so) while sending new challenges more frequently. Both security and usability are improved over the last design. One advantage is that the attacker will have less time to create and display fake visual challenges at run-time (though this might not disrupt the attack creation), and will certainly prevent an attacker from creating offline fake images because the attacker must now rely on the current $v - 1$ tweets being displayed when creating the fake images. The biggest advantage with this design is that it significantly decreases the time that the overall visual challenge stays on the signage, and at the same time it increases the time for users to read the tweet. Further, it feels more dynamic than displaying only one tweet at a time.

Grid of Tweets. Similar to the previous design, a grid layout is also another natural way of displaying multiple tweets. The difference is that there is no clear indication on the order on which the users should read the tweets. While it is an advantage to be able to refresh multiple tweets at a time, for users it might be annoying when they might not know when and which tweets will be refreshed next (some users might be overwhelmed if the number of tweets increases). This should not be a problem considering cases when it is not meant for


ReadWrite @RWW
#iot The risk on your wrist: Are hackers targeting your wearables data? ow.ly/8JuRBb
6:55 PM - 17 Apr 2016
12 4

Science Friday @scifri
Welcome to the land of volcanoes, glaciers, and Mars-like deserts. scifri.me/o4
1:39 PM - 17 Apr 2016
12 17

Los Angeles Times @latimes
Why escape rooms are becoming a breakout form of entertainment lat.ms/1W906CV pic.twitter.com/Im3wxcbFLw
6:04 AM - 17 Apr 2016
30 34

CS Monitor @csmonitor
The secret to a happy life? It's not what you think. trib.al/gUpDrjr pic.twitter.com/LNJTgiYqeN
9:47 AM - 17 Apr 2016
2

The Atlantic @TheAtlantic
Brain prosthetic allows paralyzed man to move his hand again theatln.tc/1SljhtN pic.twitter.com/IlGovJBflt
3:50 PM - 17 Apr 2016
58 76

Los Angeles Times @latimes
Why escape rooms are becoming a breakout form of entertainment lat.ms/1W906CV pic.twitter.com/Im3wxcbFLw
6:04 AM - 17 Apr 2016
30 34

TIME.com @TIME
Here's what happens when you boil an iPhone 6 in Coca-Cola ti.me/1qMz7S0
5:53 PM - 17 Apr 2016
47 66

techradar @techradar
Get ready for Facebook Live streams from everywhere, including drones buff.ly/23D5FPZ pic.twitter.com/dpmezDO4j1
1:05 PM - 17 Apr 2016
12 7

Yahoo UK News @YahooNewsUK
This is how to make the perfect cup of coffee, according to science yhoo.it/1Yvvzy7 pic.twitter.com/i6mxa8zyJZ
11:08 AM - 17 Apr 2016
3

ZDNet @ZDNet
Critical infrastructure cybersecurity is a 'mess', warns Eugene Kaspersky zd.net/1XzH2MS @dannyjpalmer pic.twitter.com/J8bEMF0cUQ
1:30 AM - 17 Apr 2016
18 9

Figure 5: Design strategies for displaying visual challenges: (a) single tweet, (b) grid layout, and (c) list of tweets.

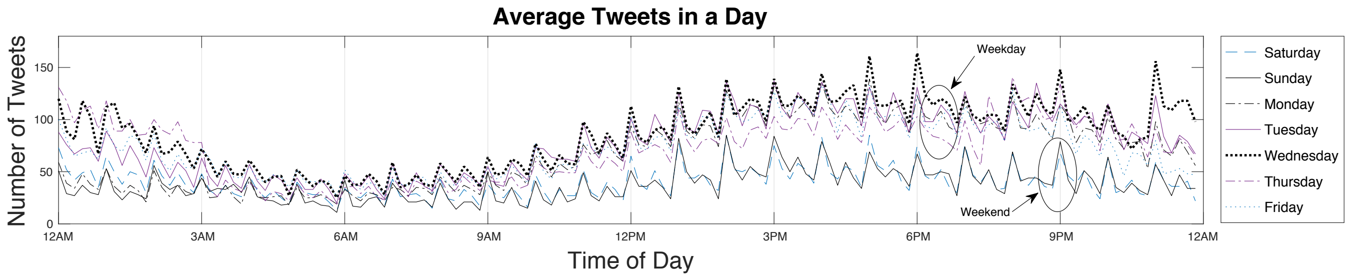


Figure 6: The average number of tweets in a day—sampled at 10 minutes for 24 hours.

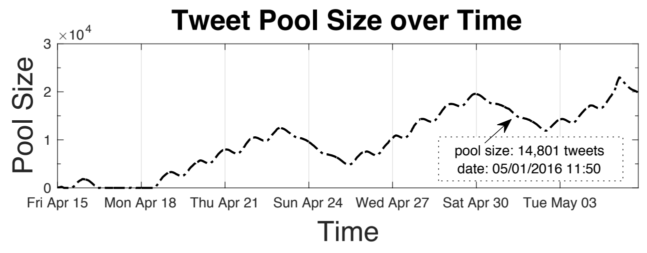


Figure 7: Number of tweets in the pool of tweets over time. The pool size is calculated based on the accumulative number of tweets over time and the subtraction of tweets as they are used as the next visual challenge. This graph reflects the selection and deletion of 6 tweets per minute.

users to read every single tweet being displayed. In terms of security, we want to make it harder for an attacker to correctly guess which tweets will be refreshed next and also to create fake image frames containing the correct tweets.

One usability disadvantage for all three proposed designs is that tweets are used only once, and a user might attempt to wait until tweets start repeating (as it is in most signages found in public places). However, never repeating the same tweets is a big security advantage. Otherwise, an attacker could simply save the sequence of video frames (containing the tweet visual challenges) and replay when the tweet visual challenges start repeating in the monitor.

5 EXPERIMENTAL RESULTS

User Pool Selection: We selected a total of 337 users to collect tweets from, based on criteria such as: users' location, verified users, and influential / popular users. For location, we selected users from the USA, Canada, and the UK; for popular users, we selected news channels (i.e., users whose description had the word *news*), and for influential users, we included popular politicians, business leaders, and people who made to The 100 Most Influential People list.

We used search queries like "`news filter:verified lang:en`" on Twitter and handpicked the final user list. This search query shows an example of how to find users that are verified, tweets using the English language, and has the word news on the account description. To limit users from a particular location we can include: "`near:location within:15mi`" in the search query.

Tweeting Frequency Pattern: Fig. 6 shows the average number of tweets in a day for each day of the week. As expected, throughout a 24-hour period, the largest number of tweets belong to weekdays and the lowest to weekends. The lowest number of tweets occurs at 5:50 a.m. on Sunday with 11 tweets on a 10-minute interval. Nonetheless, on average the lowest numbers occur between 5 a.m. and 7 a.m. for all days of the week. This is different from the highest numbers of tweets. The highest number of tweets occurs at 6 p.m. on Wednesday with 164 tweets for the same time interval. On average there is a larger number of tweets between 4 p.m. and 6 p.m. for all days. It is worth noting that the results directly reflect our choice of selected users: most users are from the U.S. and it is understandable that they are not tweeting during night hours. These trends could highly change when we select more users from different time zones and users from outside the U.S. such as from the UK or Australia.

Tweet Pool Size: By analyzing the tweeting frequency pattern of 337 verified users, it is clear that it is not enough to pick the latest tweet as the next visual challenge. There are times (such as weekends and late hours) that the tweet rate substantially decreases. For this reason, we instead create a pool of tweets to use over time. We do this by collecting and storing tweets (as they come) to a pool of tweets. Then at each verification step, instead of using the latest tweet as the next visual challenge, we pick a *random* tweet out of a pool of tweets. The pool continuously updates with new tweets (as they become available) and in parallel to the visual challenge selection operation. After the tweet is selected to be used as the next visual challenge, it is deleted from the pool to prevent used tweets from being re-used.

Fig. 7 shows the tweet pool size over time (for 15 days). Even when there are fewer tweet activities during late and early hours, the overall number of new tweets during the day is large enough to keep populating a large pool of tweets. Further, these observations can be used to design better update strategies with higher selection/deletion rate, and auto-expiring of tweets to prevent the size of the pool to grow infinitely while ensuring enough tweets at all times in the pool.

One advantage we have from picking the next random tweet out of a large pool-of-fresh-tweets (versus selecting a random latest tweet published by one of the verified users) is that it helps ensuring the pool of tweets is never empty. Another advantage is that if an attacker is able to gain access to the pool of tweets, the attacker will not be able to correctly guess what tweet will be used next.

User Tweeting Frequency Pattern: The users we selected have a different tweeting frequency pattern. Fig. 8 shows the number of tweets for all 337 users. While the plot reflects all the users, only of subset of the users (every 10 user) is labeled in the x-axis. Users are sorted by the number of tweets they have tweeted over a period of 20 days. Users like `@nytimes` tweeted 2,758 times, whereas others like `@TwitterSF` only 8 times. It turns out that most users (269 of them) tweeted overwhelmingly under 1,000 versus the 22 users that tweeted more than 2,000 and 7 users that tweeted more than 3,000.

Word and Character Frequency vs. Reading Speed: In our studies we found that on average the tweets we gathered where composed of 12 words (11.88) and 95 characters (94.76). Studies suggest that the average reading speed of an adult ranges from 200 to 250 words-per-minute (WPM) [19] and more precisely 228(+/-30) WPM [41] and 863(+/-234) characters per minute (CPM). Therefore, in our case a reasonable refresh rate would range from 2.79 to 3.64 seconds to leave each tweet in the monitor (based on our average number of words).

In the case we use multiple tweets, we can increase the refresh rate while still maintaining a reasonable time for each individual tweet. For example, consider that we use three tweets at a time. Then, if we leave the overall image in the monitor (containing three tweets) for 1 second, the longest time an individual tweet will be displayed would be 3 seconds which is within a good range for a person to read and we would have minimized the overall time of each unique combination of tweets in the monitor. Therefore, decreasing the maximum duration of a smart replay attack.

5.1 Live Video Capture and Recognition

Tweets Captured for Recognition. We have continuously displayed a total of **11,494** visual challenges in the monitor

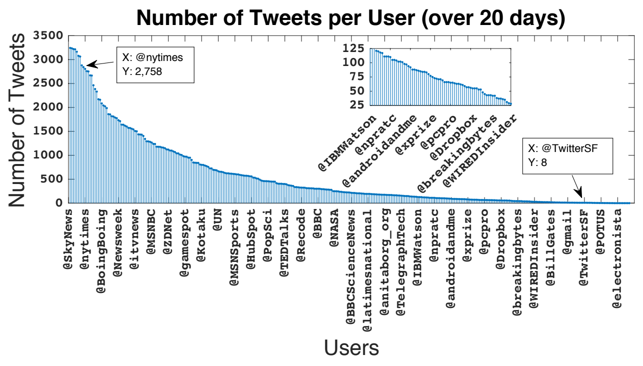
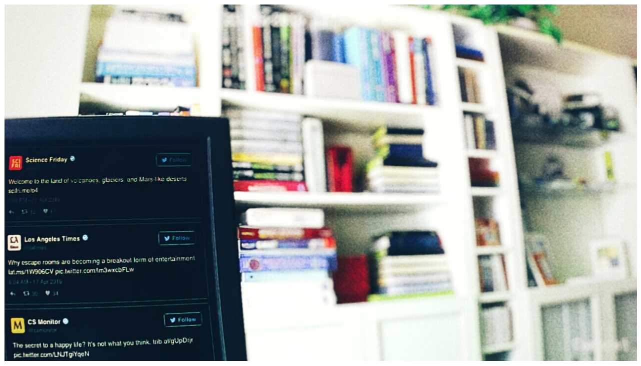


Figure 8: The number of tweets per user for a period of 20 days. Users are sorted by order of most number of tweets to users with the least number of tweets.

(DELL E156FP 15" LCD monitor) placed in the field-of-view of the camera; and we have captured a video frame for each time a new visual challenge was displayed. In our experiments, we have refreshed the visual challenge every 4 seconds. We captured video frames both during day time and at night time, to test the accuracy of recognition for different lighting settings. Fig. 9a shows our deployment setup for conducting experiments. In this photo we show the Swann NVW-470 surveillance camera system we use in all our experiments, and we show a digital monitor (slightly in front of the camera) displaying a Tweet visual challenge on the upper right hand-corner of the screen. Here, the laptop (not pictured) runs the verifier code generating the visual challenge (by retrieving tweets via the Twitter API), and displays the visual challenge in the monitor. The verifier also runs the recognition algorithm to extract a digital form of the Tweet text from the visual challenge and compare it with the desired text as described next.

(a)



Welcome to the land of volcanoes, glaciers, and Mars-like deserts scifri.me/o4

Why escape rooms are becoming a breakout form of entertainment lat.ms/1W906CV pic.twitter.com/lm3wxcbFLw

The secret to a happy life? It's not what you think. trib.al/gUpDrjr pic.twitter.com/LNJTgiYqeN

(b)

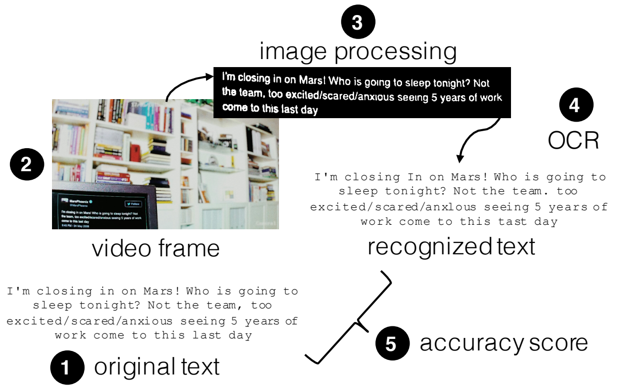
Figure 9: (a) A photo of our setup where the monitor is placed in the field-of-view of the camera. (b) An example of an image frame captured by the surveillance camera.

Fig. 9b shows our implementation of the multiple tweet challenges. The figure is of an image frame captured by the camera—part of the monitor displaying the visual challenges can be seen in the figure. The tweets are aligned using a list design. As we show in our results later, we perform our image processing step for each tweet individually, and then calculate an average of the accuracy score for multiple tweets. The results we get by using the list-of-tweets design is similar to using the grid design (not shown in the photo).

5.1.1 Recognition Accuracy of Visual Challenges. Our visual challenge recognition process is described in Fig. 10. As described before, first the verifier picks a random tweet to display in the monitor. Upon receiving the next video frame, our image processing includes the following: (1) detection of visual challenge in the video frame, (2) cropping the visual challenge from the frame, (3) thresholding the image frame, (4) sharpening, and (5) resizing by enlarging the image. Once we have prepared the image frame, we recognize the visual challenge using an optical character recognition (OCR) engine. In our experiments we use `tesseract` [39] as the OCR engine. We use the following metric (based on [25]) to calculate the *recognition accuracy score* of our visual challenges:

$$\text{accuracy}_{\text{OCR}} = \frac{l - dist}{l} \in [0, 1]$$

where l is the length of the correct string and $dist$ is the edit distance—a metric to tell how far apart the extracted string is from the correct one based on the minimum number of edit operations necessary to convert one string to another, i.e., the number of errors between the recognized and original text. To compute the edit distance (also known as the Levenshtein distance [21]) we use the `python-Levenshtein` [10] library in our implementations.



I'm closing in on Mars! Who is going to sleep tonight? Not the team, too excited/scared/anxious seeing 5 years of work come to this last day

I'm closing In on Mars! Who is going to sleep tonight? Not the team. too excited/scared/anxlous seeing 5 years of work come to this tast day

I'm closing in on Mars! Who is going to sleep tonight? Not the team, too excited/scared/anxious seeing 5 years of work come to this last day

Figure 10: Visual challenge recognition process: (1) original text is used as visual challenge in the display; (2) camera captures the next video frame; (3) image processing includes: detection of visual challenge in the frame, cropping the visual challenge, thresholding, sharpening, and resizing the image; (4) visual challenge is recognized with an OCR algorithm that is trained with characters from the original text; and (5) the extracted text is compared with the original text using the edit distance to calculate the accuracy score.

In Fig. 11 we show the recognition accuracy score per user. There are a total of 5,052 blue dots in the figure depicting the accuracy score for each frame recognized (from over 11 thousand image frames collected). We show a subset sample of character recognition accuracy scores for each visual challenge, grouped by the corresponding Twitter user. The y-axis shows multiple scores (blue dot) for each user in the x-axis; thus, multiple dots per user are aligned vertically. The brown line shows the average accuracy score per user. The average accuracy is high for all but a few users (mostly those who tweeted short URLs).

Several of the users who had a low accuracy score tweeted short URLs, and we found a discrepancy on how Twitter displays URLs on tweets (most URLs on the tweet text retrieved from the API are a shortener to a short URL like `http://t.co/4SR5N1D3GB` whereas embedded tweets contain the original URLs) causing low accuracy scores because of mismatched URLs. In order to obtain better results we could potentially filter users who tweet URLs.

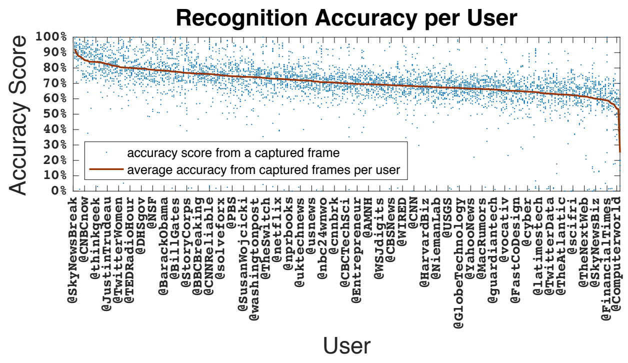


Figure 11: Accuracy score of captured frame per user.

Fig. 12 shows the accuracy for recognizing visual challenges in over 1,025 captured frames where the challenge was composed of a list of *multiple* tweets. The visual challenge in each frame is different from the challenge in the previous frame by just one new tweet; and from the fact that the tweets were slightly shifted up to make room for the new tweet to appear. For each frame, we have calculated the accuracy of individual tweets (blue dot) and the average score among the multiple tweets (brown line). We ordered the x-axis from high-accuracy to low accuracy.

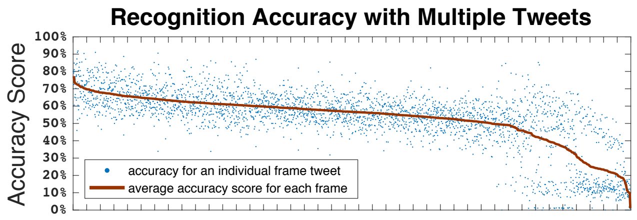


Figure 12: Accuracy score for multiple tweets. We ordered the x-axis from high-accuracy to low accuracy.

While the accuracy of our multiple-tweet proposal is good, it is far from perfect. However, we do not need to have perfect accuracy for our system performance, we only need to have more confidence that the challenge is in the video frame we received, instead of another (perhaps older) challenge.

To see how the error rate (e.g., $1 - accuracy$) behaves before and after an attack, we experimented with sending old frames to the verifier starting at frame 177. The results are illustrated in Fig. 13.

As we can see, in order to decide if the challenge is the correct one or not in the frame, we only need to select a threshold close to 50%. We explore in more detail how good is our binary classification accuracy under a variety of thresholds in the next section.

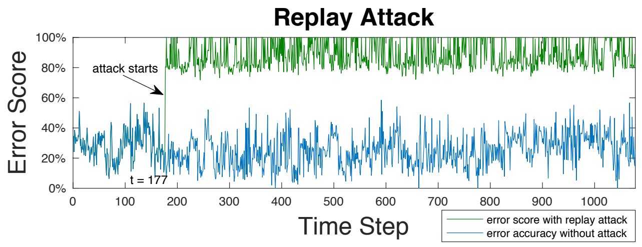


Figure 13: Error score under a replay attack.

There has been an increasing interest in leveraging computer vision to help visual impaired people to access visual information [4, 6, 17, 24, 38]. Some of these works look at the problem of recognizing digital signage with the goal of parsing the information transmitted by the signage to the user. This problem is much more challenging than ours. One obvious advantage of our system is that the verifier knows where the visual challenge is located in the image frame, and also what the visual challenge should be, whereas the related approaches in fact try addressing the text localization problem. The second advantage of our problem is that we do not need to get 100% OCR accuracy recognition. We only need to be fairly certain (e.g., 60% accuracy) that our expected text is in the image.

5.2 Binary classification accuracy

We classify each frame into either one of two classes as follows:

$$\text{class} = \begin{cases} 1, & \text{accuracy}_{\text{OCR}} > thresh \\ 0, & \text{otherwise} \end{cases}$$

The first class consists of frames that have the correct challenge the verifier sent, and the other one consists of frames that have a different challenge. Table 1 shows possible threshold values that we found to work well in our experiments. For example, when we use 1,078 correct frames (the positive class) and 1,078 replayed frames (the negative class) and compare their performance we find the following results: (1) By using *thresh=27.5%* (shown in bold in the table) we correctly classify 1,066 (out of 1,078) frames belonging to the positive class (true-positive) and only 12 false negatives; and (2) for the incorrect frames (from a replay-attack) we have 2 false positives and 1,076 true negatives. Fig. 14 shows the Receiver operating characteristic (ROC) curve to show the performance of our binary classifier system.

Table 1: Binary classification on different thresholds

Threshold	TP	FN	FP	TN	TPR	FPR
15.0%	1,070	8	506	572	0.993	0.4694
17.5%	1,069	9	368	710	0.992	0.3414
20.0%	1,068	10	200	878	0.991	0.1855
22.5%	1,067	11	65	1,013	0.990	0.0603
25.0%	1,067	11	12	1,066	0.990	0.0111
27.5%	**1,066**	**12**	**2**	**1,076**	**0.989**	**0.0019**
30.0%	1,064	14	0	1,078	0.987	0.0000
32.5%	1,063	15	0	1,078	0.986	0.0000
37.5%	1,062	16	0	1,078	0.985	0.0000
40.0%	1,061	17	0	1,078	0.984	0.0000

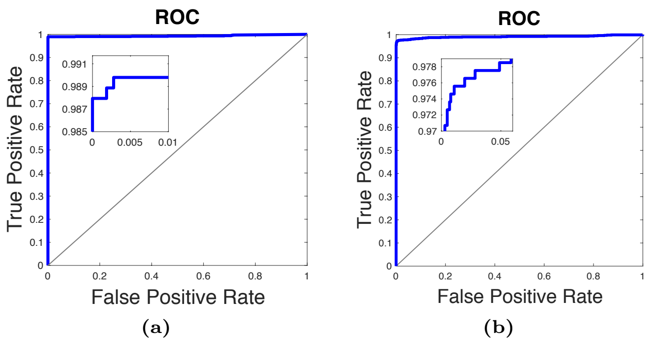


Figure 14: ROC curve using data from a replay attack. We simulate a replay attack by recording previous frames and replaying them at a later time. Results from using (a) single-tweet design and (b) list-of-tweets design as visual challenge.

These results show a robust performance against (1) physical, (2) spoofing, and (3) replay attacks. To mitigate the problem of (4) smart replay attacks, we have designed a challenge system that contains multiple tweets, in order to minimize the time a smart replay attacker will have between replaying a valid frame and receiving the next challenge.

6 SECURITY ANALYSIS

As shown in the previous section, our implementation prevents several attacks from our threat model:

Physical Attacks: If the camera is covered or destroyed, the video frames received by the verifier will not have the tweet, and verification of the tweet will fail. If the camera is moved to point to a different location, the tweet will appear in a different location of the image frame, and thus the verifier can detect that the tweet is not being displayed in the correct location of the frame. **Spoofing Attacks:** As in the previous case, if the video sent by the attacker does not correspond to a video that contains the display of a visual challenge, then the attack will be detected. **Replay Attacks:** If the attacker replays video containing old challenges (tweets) the optical character recognition algorithm used by the verifier will detect that the challenge displayed is not the same as the challenge that was sent. The ability to detect this depends of course on how often we refresh challenges. This leads us to the next attack. **Smart Replay Attacks:** This attacker will only display a still image of the most recent challenge. While the challenge is not refreshed, this image will pass the test of the verifier, therefore if we want to prevent the smart replay attackers to have a long time to show us an old frame, we would need to refresh the challenge frequently enough. This attack is the reason we selected Tweets as visual challenges instead of other less frequent media updates like headline news. The ability to tap to hundreds or thousands of verified Twitter users in our analysis, shows that there is always a fresh, recently posted Tweet we can use to satisfy our challenge refresh rate requirements.

In order to minimize the amount of time that the smart replay attacker can replay an old frame and pass the verifier test, we need to refresh the verification challenge frequently enough. However we now need to take into consideration design issues related to human factors. If we refresh a tweet frequently enough, people in the area attempting to read it will be frustrated by the refresh rate, however, if we update tweets for example, every minute, then the smart replay attacker will have a minute where it can replay the first video frame and then attempt some malicious action in the field of vision of the camera for one minute without being detected. To mitigate these concerns we designed multiple interleaving challenges. So instead of showing only one tweet, we had have several of them and then only update the oldest one each time. Assuming we have enough tweets in the display, allowing us to refresh the challenge rate we can minimize the time a valid smart replay attack is valid.

Our final two attacks (a compromised verifier and an anti-forensic forgery attack) are more difficult to defend against. We argue however that previous work most related to our own setting has not considered these advanced attacks, and in this paper we want to give some initial ideas on how we can tackle these attacks.

In particular the security of previous proposals always assumes the verifier is trusted (the system fails if the verifier is compromised). In our case, if an attacker compromises the verification procedure, it can always report a good match between the challenges and the video, even though the image can be completely fabricated.

Compromised Verifier Attack: Notice, that given our design choices, the verification of the challenge can also be easily done by a human operator. The human operator cannot do this verification continuously as our automated algorithms (it is not a perfect solution), but it can do so occasionally, or if other indicators of compromise are present. To check if the verifier is working correctly the human operator can check the received video feed, identify the author of the tweet (a trusted reputable and popular Twitter user in our design) and then using a different (hopefully uncompromised) computer check the Twitter feed of the claimed author, and check whether the tweet is fresh (the tweet needs to be less than a couple of minutes old in our design). While this may not a perfect solution to the problem, we believe our work is the first to propose a partial solution to the problem of having the digital verifier compromised.

The final adversary is the most challenging to protect against. If an attacker has remote access to the camera, and also has the necessary credentials to be a man-in-the-middle between the camera and the verifier, it can capture the image of the camera, crop the area where the current verifier appears, and paste it over the image frame the attacker wants to spoof before sending it to the verifier. This is a powerful adversary whose analog has rarely been considered in other similar proof of reality work [20, 31].

Anti-Forensic Attack: Our previous experiments in image forensics [49] and those of others [22, 36] can attempt to minimize this threat by detecting so called copy-and-paste (or copy and move) image forgery attacks [1]. Unfortunately, for pretty much any image forensic tool, there is an anti-forensic attack (i.e., a forged image attack that tries to bypass attempts from forensic tools to identify if the image has been tampered with) [35]. The state of the art in the forensic vs. anti-forensic research community considers this problem a cat and mouse game [34]. At the end of the day the success of an anti-forensic attack will depend on the power and resources of the adversary (how much does it know about the forensic tools we use) and on some luck (it needs to generate a continuous stream of successful anti-forensic images and we only need one mistake or lighting condition advantage to detect one forgery).

Note also that by relaying on Twitter services to provide our challenges, our system will depend on an infrastructure that is not in our control. Twitter can change their services, or our connection to Twitter can even be attacked by our adversary in order to prevent us from receiving fresh tweets, forcing us to stop sending new challenges. We can always try to mitigate these concerns by using several services (Twitter, Instagram, Facebook, News Headlines etc.) or if we are completely offline, by asking our verifier to generate its own random phrases to be sent as physical challenges.

Finally, recall that we are using Twitter-verified accounts, and not just random Twitter accounts, as such the security of our scheme also depends on the ability of Twitter to provide trustworthy verified accounts. In particular, by selecting high-profile Twitter-verified accounts (instead of random accounts), we minimize the risk of generic attacks against Twitter [28, 40], including mass account hijacking, and fake accounts. Further, for the purposes of our verification mechanism, we do not discriminate Twitter accounts as those propagating *fake news* or not. Our approach is independent to tweet content. So, as long as the account from where the tweet is coming from is verified and belongs to a trusted, reputable source we include those tweets to be used as visual challenges.

When we performed our experiments, Twitter verification was not open to the public, but recently, Twitter opened their verification process to the public [46]. While this might increase the risk of having malicious Twitter-verified accounts, our design selects high-profile news organizations, which should remain trustworthy users. Having said that, it will be interesting to monitor the behavior of Twitter-verified accounts, and see if opening this verification to the public will increase the level of malicious yet Twitter-verified accounts.

7 CONCLUSIONS

We have proposed a new way to verify the integrity of video footage from cameras deployed in public spaces. Our proposal has a unique new combination of properties: (1) it can be used to automatically verify sensor data continuously (in real-time), (2) humans can also verify the physical challenge seen in the video (this property is unique to our system as all previous work assumes that the verifier is trusted, but in our case the human can check—although not continuously—if the verifier is working properly), (3) the verification can be done remotely, (4) our system can be deployed in public spaces as the physical challenge is designed in a way that is minimally disruptive to potential bystanders, (5) our results show that our proposal is resilient to a large class of adversaries, and (6) the prover does not need to be aware that the verifier is challenging it, because the challenge is sent to the physical world rather than to the prover itself, our approach can be used in "legacy" sensors (provers).

Perhaps the largest challenge that remains open is the problem of forensic and anti-forensic tools for detecting image forgeries like the copy-and-paste or copy-and-move attack. This is a line of work that is highly relevant for the security of our system against stronger adversaries, but it is also an orthogonal line of work. The unique property that our proposed system has when compared to traditional image forensic analysis is that in the standard case, the forgery does not need to be created in real-time, while in our case, if the attacker wants to bypass the verifier test, the forgery needs to be created in real-time. Unfortunately copy-and-paste forgeries can be created efficiently. The only question remaining would be on the real-time anti-forgery tools.

We plan to continue this line of research in future work by finding analogies of our proposal to other sensor applications.

ACKNOWLEDGMENTS

This material is based upon work supported by NSF CNS #1547502 and by the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.

REFERENCES

[1] Irene Amerini, Lamberto Ballan, Roberto Caldelli, Alberto Del Bimbo, and Giuseppe Serra. 2011. A sift-based forensic method for copy–move attack detection and transformation recovery. *Information Forensics and Security, IEEE Transactions on* 6, 3 (2011), 1099–1110.

[2] Martin Anderson. 2016. Google testing facial recognition payments on Android and iOS in San Francisco. (March 2016). https://thestack.com/cloud/2016/03/03/google-hands-free-facial-recognition-trial-san-francisco

[3] Business Insider. [n. d.]. Facebook Users Are Uploading 350 Million New Photos Each Day. http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9. ([n. d.]).

[4] James Coughlan and Roberto Manduchi. 2013. Camera-based access to visual information. *Assistive technology for blindness and low vision* (2013), 219–246.

[5] Ben Fisch, Daniel Freund, and Moni Naor. 2014. Physical zero-knowledge proofs of physical properties. In *Advances in Cryptology (CRYPTO'14)*. Springer, 313–336.

[6] Giovanni Fusco, Ender Tekin, Richard E Ladner, and James M Coughlan. 2014. Using computer vision to access appliance displays. In *Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility*. 281–282.

[7] Alexander Glaser, Boaz Barak, and Robert J Goldston. 2014. A zero-knowledge protocol for nuclear warhead verification. *Nature* 510, 7506 (2014), 497–502.

[8] Dan Goodin. 2012. A Fort Knox for Web crypto keys: Inside Symantec's SSL certificate vault. (Nov. 2012). https://arstechnica.com/security/2012/11/inside-symantecs-ssl-certificate-vault/

[9] Dan Goodin. 2015. Prosecutors suspect man hacked lottery computers to score winning ticket. (April 2015). https://arstechnica.com/tech-policy/2015/04/prosecutors-suspect-man-hacked-lottery-computers-to-score-winning-ticket/

[10] Antti Haapala. [n. d.]. Levenshtein Python C. https://github.com/ztane/python-Levenshtein/. ([n. d.]).

[11] Craig Heffner. 2013. Exploiting Network Surveillance Cameras Like a Hollywood Hacker. https://youtu.be/B8DjTcANBx0. (Nov. 2013).

[12] Muhammad Ikram, Lucky Onwuzurike, Shehroze Farooqi, Emiliano De Cristofaro, Arik Friedman, Guillaume Jourjon, Mohammad Ali Kaafar, and M Zubair Shafiq. 2015. Combating Fraud in Online Social Networks: Detecting Stealthy Facebook Like Farms. (2015).

[13] Instagram Press. [n. d.]. Instagram Stats. https://www.instagram.com/press/. ([n. d.]).

[14] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. 2015. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound.. In *24th USENIX Security Symposium*. 483–498.

[15] Alice MacGregor. 2016. Amazon wants to replace passwords with selfies and videos. https://thestack.com/security/2016/03/15/amazon-wants-to-replace-passwords-with-selfies-and-videos/. (March 2016).

[16] Alice MacGregor. 2016. MasterCard rolls out 'selfie' verification for mobile payments. https://thestack.com/iot/2016/02/22/mastercard-rolls-out-selfie-verification-for-mobile-payments/. (Feb. 2016).

[17] Roberto Manduchi and James Coughlan. 2012. (Computer) vision without sight. *Commun. ACM* 55, 1 (2012), 96–104.

[18] J.M. McCune, A. Perrig, and M.K. Reiter. 2005. Seeing-is-believing: using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy (S&P)*. 110–124.

[19] James McNair. 2011. What is the average reading speed and the best rate of reading. (2011).

[20] Yilin Mo, R. Chabukswar, and B. Sinopoli. 2014. Detecting Integrity Attacks on SCADA Systems. *Control Systems Technology, IEEE Transactions on* 22, 4 (July 2014), 1396–1407.

[21] Vreda Pieterse and Paul E. Black. 2015. Levenshtein distance. In *Dictionary of Algorithms & Data Structures*. (2015).

[22] Alessandro Piva. 2013. An overview on image forensics. *ISRN Signal Processing* 2013 (2013).

[23] Angelo Racoma. 2012. Android Jelly Bean Face Unlock 'liveness' check easily hacked with photo editing. http://www.androidauthority.com/android-jelly-bean-face-unlock-blink-hacking-105556/. (Aug. 2012).

[24] Irati Rasines, Pedro Iriondo, and Ibai Díez. 2012. *Real-Time display recognition system for visually impaired*. Springer.

[25] Stephen V Rice, Frank R Jenkins, and Thomas A Nartker. 1996. *The fifth annual test of OCR accuracy*. Information Science Research Institute.

[26] Kurt Rosenfeld, Efstratios Gavas, and Ramesh Karri. 2010. Sensor physical unclonable functions. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. 112–117.

[27] Ulrich Rührmair, JL Martinez-Hurtado, Xiaolin Xu, Christian Kraeh, Christian Hilgers, Dima Kononchuk, Jonathan Finley, and Wayne Burleson. 2015. Virtual proofs of reality and their physical implementation. In *IEEE Symposium on Security and Privacy (S&P)*. 70–85.
[28] Carl Sabottke, Octavian Suciu, and Tudor Dumitras. 2015. Vulnerability disclosure in the age of social media: exploiting twitter for predicting real-world exploits. In *24th USENIX Security Symposium*. 1041–1056.
[29] N. Saxena, J. Ekberg, K. Kostiainen, and N. Asokan. 2006. Secure device pairing based on a visual channel. In *IEEE Symposium on Security and Privacy (S&P)*.
[30] Santa Clara County Sheriff. 2013. PG&E Substation Surveillance Video. https://www.youtube.com/watch?v=RQzAbKdLfW8. (June 2013).
[31] Yasser Shoukry, Paul Martin, Yair Yona, Suhas Diggavi, and Mani Srivastava. 2015. PyCRA: Physical Challenge-Response Authentication For Active Sensors Under Spoofing Attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1004–1015.
[32] Michael Andrew Sipe, Henry Will Schneiderman, and Michael Christian Nechyba. 2013. Facial recognition. (June 4 2013). US Patent 8,457,367.
[33] Ivo Sluganovic, Marc Roeschlin, Kasper B Rasmussen, and Ivan Martinovic. 2016. Using Reflexive Eye Movements for Fast Challenge-Response Authentication. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1056–1067.
[34] Matthew Stamm, Sabrina Lin, and KJ Liu. 2012. Forensics vs. anti-forensics: A decision and game theoretic framework. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1749–1752.
[35] Matthew Stamm, Sabrina Lin, and KJ Liu. 2012. Temporal forensics and anti-forensics for motion compensated video. *IEEE Transactions on Information Forensics and Security* 7, 4 (2012), 1315–1329.
[36] Matthew Stamm, Min Wu, and KJ Ray Liu. 2013. Information forensics: An overview of the first decade. *Access, IEEE* 1 (2013), 167–200.
[37] Ariane Tabatabai. 2015. How much monitoring of Iranian nuclear facilities is enough? http://thebulletin.org/how-much-monitoring-iranian-nuclear-facilities-enough7923. (Jan. 2015).
[38] Ender Tekin, James M Coughlan, and Huiying Shen. 2011. Real-time detection and reading of LED/LCD displays for visually impaired persons. In *IEEE Workshop on Applications of Computer Vision (WACV)*. 491–496.
[39] Tesseract OCR. [n. d.]. Tesseract Open Source OCR Engine. https://github.com/tesseract-ocr/tesseract. ([n. d.]).
[40] Kurt Thomas, Frank Li, Chris Grier, and Vern Paxson. 2014. Consequences of connectivity: Characterizing account hijacking on twitter. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 489–500.
[41] Susanne Trauzettel-Klosinski and Klaus Dietz. 2012. Standardized Assessment of Reading Performance: The New International Reading Speed Texts IReST. *Investigative ophthalmology & visual science* 53, 9 (2012), 5452–5461.
[42] Twitter. [n. d.]. API Rate Limits Per User or Per Application. https://dev.twitter.com/rest/public/rate-limiting. ([n. d.]).
[43] Twitter. [n. d.]. The Streaming APIs Overview. https://dev.twitter.com/streaming/overview. ([n. d.]).
[44] Twitter. [n. d.]. Twitter API. https://dev.twitter.com. ([n. d.]).
[45] Twitter. [n. d.]. Twitter Brand Policy. https://about.twitter.com/press/twitter-brand-policy. ([n. d.]).
[46] Twitter. 2016. Announcing an Application Process for Verified Accounts. https://blog.twitter.com/2016/announcing-an-application-process-for-verified-accounts-0. (July 2016).
[47] Twitter Blog. 2013. New Tweets per second record, and how! https://blog.twitter.com/2013/new-tweets-per-second-record-and-how. (Aug. 2013).
[48] Lisa Vaas. 2015. Man arrested after posting Facebook videos of him tampering with traffic cameras. https://nakedsecurity.sophos.com/2015/08/28/man-arrested-after-posting-facebook-videos-of-him-tampering-with-traffic-cameras/. (Aug. 2015).
[49] Junia Valente and Alvaro A. Cárdenas. 2015. Using Visual Challenges to Verify the Integrity of Security Cameras. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC'15)*. ACM, 141–150.

A TWITTER APIS

The Twitter REST API allows us to retrieve tweets users have tweeted in the past while the Streaming API returns new tweets as they become available. Fig. 15 shows an example of an application using the REST API [43]: the application accepts user requests, sends one or more requests to the Twitter's API, and then sends back the results to the user.

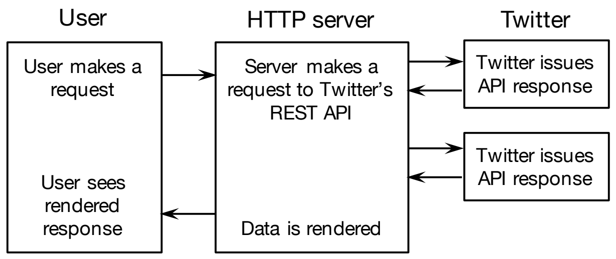

User
HTTP server
Twitter
User makes a request
User sees rendered response
Server makes a request to Twitter's REST API
Data is rendered
Twitter issues API response
Twitter issues API response

Figure 15: Example of an application using the REST API.

We experimented with the REST API; however, we found out that the tweet retrieval is rate limited to 450 queries per 15 minute window [42]. We therefore turned to the Streaming APIs instead to avoid rate limits. Fig. 16 shows how the Streaming API works [43].

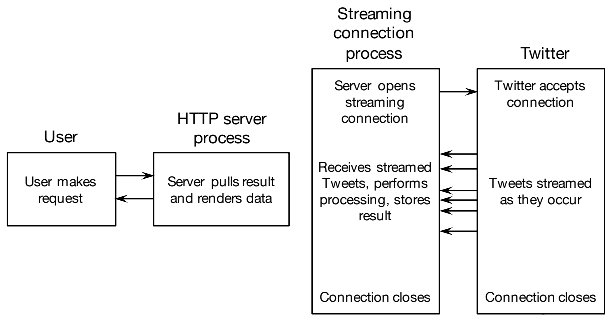

Streaming connection process
Twitter
User
HTTP server process
User makes request
Server pulls result and renders data
Server opens streaming connection
Receives streamed Tweets, performs processing, stores result
Connection closes
Twitter accepts connection
Tweets streamed as they occur
Connection closes

Figure 16: Example of an application using the Streaming API.

By default, the Streaming API delivers a stream of tweets with (1) tweets from a list of users (we specified), (2) tweets retweeted by those users, (3) replies to tweets created by the users, (4) retweets (RTs) of tweets created by the users, and (5) replies created without pressing the reply button (e.g., `@MarsCuriosity hello`). Upon receiving new tweets, the verification system filters and processes all incoming tweets to select only those that make good candidates for visual challenges: we discard tweets that start with `"RT @"` (because they are essentially repeated tweets and can compromise the freshness property of our system) and any tweet that does not come from a user specified in our application (e.g., tweets that were RTed by pressing the retweet button on Twitter).