TrueNews: Decentralized Photography Verification

Nathalie J. Dittrich, Florian Echtler
Bauhaus-Universität Weimar, Germany
firstname.lastname@uni-weimar.de

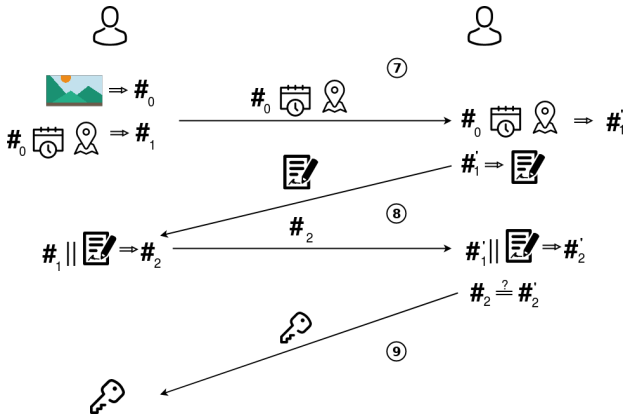

Figure 1: Generation of an image signature (numbers refer to tbl. 1). For a detailed description, see Figure 2.

Abstract

We are currently witnessing an erosion of trust in news media, driven by the narrative of "fake news" and a rise of sophisticated image forgery techniques. Images found online are therefore often dismissed as untrustworthy, even if documenting noteworthy events such as police brutality. We present *TrueNews*, a photography app designed to automatically create and add cryptographic signatures to each image, thereby verifying image's integrity as well as the exact time and location where it was taken. To improve trust in these signatures, nearby mobile devices are automatically contacted via Bluetooth and add their own signatures to the image in question. The signatures are integrated into the image as a watermark, and are thereby preserved when the image is posted online, even if other metadata is removed. We introduce a working prototype of *TrueNews*, show results of our preliminary evaluation with 5 different mobile device models, and discuss potential future extensions.

Introduction & Related Work

False information, described by the recent term of "fake news", is currently one of the biggest threats in social media. According to [11], on US election day 2016, there were in total 8.7 million Facebook engagements for top election stories which actually were false, while there were 7.3 million engagements for mainstream media.

#	Data sent by Host	Data sent by Client
0	Verification Request	ID + Verification mode
1	Verify Start + MAC	-
2-6	Signature + Score	Public Key (random)
7	imageHash	Wi-Fi Start (optional)
8	finalHash	image-Signature
9	-	Public Key + Score

Table 1: Modes used by *TrueNews* during verification (cf. Figure 1).

Additionally, in totalitarian states, even official news is usually flooded with propaganda and access to reliable information is difficult from both in- and outside of the country in question. Independent journalists in such countries often face severe repression, e.g. recently in Turkey. According to the Stockholm Center for Freedom [2], 98 Turkish journalists were convicted, 90 arrested and 167 wanted as of 31/05/19.

Based on these issues, our open-source *TrueNews* Android app follows two goals: enabling independent verification of images in social media, and enabling activists to spread trustworthy information while remaining anonymous. We contribute a protocol to anonymously exchange cryptographic location signatures via Bluetooth Low Energy (BTLE), a method to embed these signatures into pictures uploaded on social media, and a verifier to extract and validate the resulting watermarks.

In related work, one attempt to classify "fake news" was made by Jin et al. [6], who categorized viewpoints of tweets and built a network out of this data in which contradicting tweets can be identified. Stein et al. [9] looked at so-called hyperpartisan news, and trained a machine-learning network to recognize this kind of news based on a corpus of articles which were checked in advance by professional journalists at BuzzFeed. For Twitter, Gupta et al. [5] implemented a browser extension which evaluates tweets by use of machine learning. A credibility score is computed which then is displayed next to the tweets. All those approaches have in common that they try to classify and filter "fake news", but do not prevent the upload of unverified information.

Saroiu et al. [10] focused on the verification of the current location of a mobile device, using wireless infrastructure such as Wi-Fi access points or cell towers to prove the correctness of the location. Public keys are used to identify mobile devices and sign location proofs. Gambs et al. [3] criticize that users are endangering privacy and anonymity when using such location proofs, and introduce *Props*, a system for privacy-preserving location proofs using unique group signatures. Anonymity is preserved as long as the user does not sign the same message twice. *Informacam* also collects information to verify videos and photographs on mobile devices [4]. The file's metadata is enhanced by adding sensor data such as GPS coordinates and also includes PGP signatures. However, it is easy to forge signatures if the user has more than one PGP identity, and the fact that most social networks remove metadata on upload makes independent verification impossible.

Although blockchain technology would in theory be well suited for immutable storage and verification of media files, this approach is currently unsuited for mobile devices due to the significant computing resources required for the proof-of-work approach of validating transactions.

TrueNews

The main purpose of *TrueNews* is to allow verification of timestamp and location for images which are posted online. Nearby smartphones add cryptographic signatures to a picture which are embedded as a digital watermark before upload, thereby signing metadata such as location and time. Once the image arrives in a social network, other users can check its trustworthiness through the *TrueNews* app. Our first challenge is now to ensure the communication between several smartphones (clients) and the host phone. The second challenge is to provide a secure and reliable verification scheme for each image taken with *TrueNews*.

When taking a picture with the *TrueNews* app, the image is first adjusted for watermark-preserving upload into social networks (downscale, add alpha channel). Next, the host phone starts broadcasting a BTLE advertisement to request

Figure 2: Generation of an image signature (cf. Figure 1).

Let M be the raw pixel data of the image to be verified. The image data hash $\#_0 = H(M)$ is concatenated with location and timestamp, sent to the client, and hashed into a verification hash $\#_1$. The client verifies location and timestamp and calculates a signature hash $\#'_1 = H(\#_0||location||time)$. The client then signs $\#'_1$ with its highest-ranked key pair and obtains signature S, which is the corresponding ECDSA signature [7], using the *BrainpoolP160R1* elliptic curve. This curve was selected as it is considered secure for the foreseeable future while producing public keys with a length of 21 bytes that are sufficiently small to be broadcast via BTLE. To verify the generated hash, the host computes a second hash $\#_2 = H(\#_1||S)$ and transmits it to the client. The client also computes $\#'_2 = H(\#'_1||S)$. Iff $\#_2 == \#'_2$, the client reveals its corresponding public key to the host so that the signature can be verified later on.

image verification. Clients answer by broadcasting another advertisement which contains their location and local time stamp. The host checks if time and location match with an allowed deviation of up to 3 minutes and 50 meters.

Valid client responses are gathered for five seconds to create a queue. Afterwards, the host phone sends a start signal to the first client to begin verification. Signing of a photograph takes place between a host and a client only if they both never performed any form of verification before (to preserve unlinkability) and if they are located nearby. The verification procedure consists of two major steps: the signing of the client's keys, and the signing of the photograph. A hash of raw image data and crucial metadata such as date, time, and location is signed by the client using its public key to confirm correctness. Those two steps are repeated for each client. Finally, all information is embedded into the photograph as a digital watermark in the least significant bits of the color channels and uploaded to a social network. We use Twitter because of its widespread usage and open developer API. Figure 1 gives an overview of the whole verification process of a photograph verified by one client.

We use BTLE advertisement broadcasts to enable true peer-to-peer communication without any prior pairing. Advertisements contain a random 16-bit ID for one verification session, a mode indicator for the current verification step, and a cryptographic signature or key. We use undirected scannable advertisements with a maximum available payload size of 51 bytes, requiring careful message design to stay below this limit. As sending multiple advertisements simultaneously will currently fail silently on many Android devices, clients perform the verification process sequentially. Therefore, the host gathers all responses to the verification request for five seconds and forms a client queue. The steps performed during the modes 2-6 refer to the key management which will be discussed in the next section. During steps 7-9 the actual image verification as discussed in sidebar 2 takes place.

As images contain far too much data to be transmitted via BTLE broadcasts, we calculate a hash (SHA256) over the image data which is then sent to clients and signed. Important metadata like date, time, and location are extracted from the host phone's sensors. As a signature algorithm ECDSA with BrainpoolP160R1 curve was used because of its good performance on lightweight devices [7], and because of the small resulting signatures (21 bytes). We also investigated the option to verify the full image via an additional WiFi Direct connection. Unfortunately, our tests have shown that nearly all WiFi Direct implementations on Android are severely broken and currently unusable. All signatures and metadata are embedded directly into the image as a digital watermark (fragile LSB-watermark, to ensure that even minor alterations will cause verification to fail). The following data is embedded into the picture:

`imageHash`: hash created from the raw image, the date, the time and the location; `location`: GPS coordinates of the host phone when the image was taken; `date`: date and time when the image was taken; `hostKey`: the current public key of the host; `finalHashX`: the final hash created from the `imageHash` and the signature from client X (the client contributing to the creation of this `finalHash`; `scoreX`: trustworthiness score of client X; `signatureX`: signature created from client X; `clientKeyX`: public key of client X for verifying the corresponding signature.

When a user wants to check an image, it can be passed to *TrueNews* via the *share intent* which is provided by most social network apps. Figure 3 shows the activity displayed when a user verifies an image taken from social media. Within *TrueNews*, cryptographic keys fulfill two major

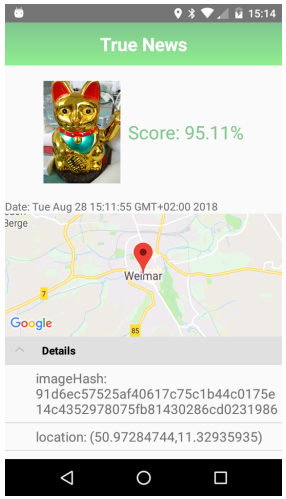

Figure 3: Activity for checking how trustworthy an image taken from social media is.

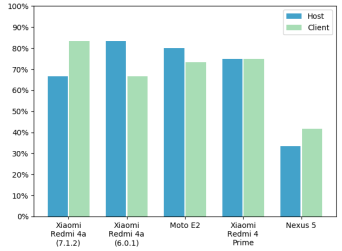

Figure 4: Percentage of succeeded BTLE verifications of the tested devices serving as host and client.

functions, signature creation and a form of identification of the user. Each user's app maintains a set of 20 public/private key pairs, and a trustworthiness score for each pair.

As an exchange for signing the picture (i.e., its hash), the host has to sign five random public keys of the client's key set in turn, which thereby have their trust scores increased. Whenever a public/private key pair has been used for a signature, it will be removed and a new key pair generated. The highest-scoring key from a client's set of 20 will be picked for the next signature, and its score also denotes the trustworthiness of that user. As mentioned above, anonymity is a crucial issue, especially when reporting police brutality or living in an oppressive regime. Consequently, it is necessary to reset the public key of a user as often as possible to maintain unlinkability. However, this approach conflicts with the creation of trustworthiness and the identification of users, leading to a tradeoff between anonymity and security. For signature creation, the key with the highest trustworthiness score is used and discarded after usage. This key cycling keeps the user from being identified or linked to several consecutive photographs.

Connecting the app directly to Twitter could also be considered to threaten anonymity. However, there are several anonymous accounts which can be used to upload sensitive content. Another option would be to use anonymizing proxy servers or the TOR network. Selecting a proper account and maintaining privacy in social networks is currently beyond the scope of this work.

Pilot Study

We conducted an expert review to examine possible flaws in the user interface and the interaction, with five participants as suggested by Nielsen [8]. All experts had a computer science background with at least one higher level degree in computer science or a related major. All participants were in their twenties, with none of them having used *TrueNews* before.

We set two tasks for the experts to accomplish, namely performing a BTLE image verification and checking the trustworthiness of an image from the Twitter app. After completing both tasks, the experts were asked to add additional comments regarding *TrueNews*. Before starting, all experts were informed about the app and its purpose, but to avoid bias, just a brief overview about the purpose of the app was given without details about the internals. For the second task, we showed participants the image from task one inside the Twitter app and let them guess how to access *TrueNews* in this scenario. Notes were taken throughout the review to save the answers, thoughts, and opinions of the experts, who were asked to articulate their thoughts aloud. During the first task, they were asked if they felt informed by the app about what was happening during the verification process. For the second task, they were asked how much they trust the score being displayed. Further details on the results of the pilot study are given in [1].

Discussion & Future Work

Regarding performance, we tested verification 63 times each via BTLE and WiFi Direct, reflecting all host-client combinations possible with the phones listed in figure 4. On average, verification via BTLE took 10.33 s (SD = 2.21) while WiFi Direct took 49.25 s (SD = 17.36), including the fixed delay of 5s for queue generation. BTLE verification failed 20 times out of 63 verifications (68% success rate). In contrast, the WiFi Direct verification succeeded just 7 times and failed 56 times (11% success rate), highlighting the severe implementation issues with WiFi Direct on Android.

Currently, the system is not yet fully secure from a cryptographic point of view. For example, a malicious user

could transmit a manipulated trustworthiness score, or multiple users could mount a Sybil attack. Therefore, some privacy-preserving authentication of the trustworthiness score needs to be integrated.

As a potential solution, we currently investigate the approach to separate the key signing from the image verification, such that public keys are always "cross-signed" automatically when a nearby device is detected. One disadvantage of this solution is that it includes storing the user's location which decreases anonymity.

Additionally, privacy-preserving location proofs could help to avoid false location data being verified, e.g. by embedding a list of currently visible WiFi access points and cell towers into the watermark. This data can later be verified using a geolocation API such as those by Google or Mozilla.

REFERENCES

1. Nathalie Dittrich. 2018. *TrueNews - Decentralized Photography Verification*. Master's thesis. Bauhaus-Universität Weimar.
2. Stockholm Center for Freedom. 2019. Jailed and wanted Journalists in Turkey. `https://stockholmcf.org/updated-list/`. (May 2019). Accessed: 2019-05-31.
3. S. Gambs, M. Killijian, M. Roy, and M. Traoré. 2014. PROPS: A PRivacy-Preserving Location Proof System. In *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*. 1–10. `DOI: http://dx.doi.org/10.1109/SRDS.2014.37`
4. guardianproject. 2015. InformaCore. `https://github.com/guardianproject/InformaCore`. (2015).
5. Aditi Gupta, Ponnurangam Kumaraguru, Carlos Castillo, and Patrick Meier. 2014. TweetCred: A Real-time Web-based System for Assessing Credibility of Content on Twitter. *CoRR* abs/1405.5490 (2014). `http://arxiv.org/abs/1405.5490`
6. Zhiwei Jin, Juan Cao, Yongdong Zhang, and Jiebo Luo. 2016. News Verification by Exploiting Conflicting Social Viewpoints in Microblogs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, 2972–2978. `http://dl.acm.org/citation.cfm?id=3016100.3016318`
7. National Institute of Standards and Technology. 1994. Digital Signature Standard (DSS). FIPS Publication 186. (May 1994).
8. Jakob Nielsen. 2000. Why You Only Need to Test with 5 Users. `https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`. (03 2000).
9. Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *Proceedings of 56th Annual Meeting of the Association for Computational Linguistics (ACL 18)(to appear)*. Association for Computational Linguistics.
10. Stefan Saroiu and Alec Wolman. 2009. Enabling New Mobile Applications with Location Proofs. In *Proceedings of the 10th Workshop on Mobile Computing Systems and Applications (HotMobile '09)*. ACM, New York, NY, USA, Article 3, 6 pages. `DOI: http://dx.doi.org/10.1145/1514411.1514414`
11. Craig Silverman. 2018. This Analysis Shows How Viral Fake Election News Stories Outperformed Real News On Facebook. `https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook`. (July 2018). Accessed: 2018-07-19.