UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Risto Pärnapuu

Verifiable Photo Snapshots

Master's Thesis (30 ECTS)

Supervisor: Ahto Truu, MSc
Supervisor: Sven Laur, PhD

Tartu 2020

Verifiable Photo Snapshots

Abstract:

In the age of digital media, photos, videos, and various other media types are widely available and accessible by anyone. Each capture of such media usually includes metadata about when the media was captured, the location of the capture, etc. On many occasions, the capture date of the media becomes a relevant factor in determining whether or not the media can or can not be used to base some claims on. However, the media itself and the associated metadata is easily modifiable by anyone.

In case of rental services, e.g., motor vehicle rental, apartment renting, and various others, it is in the interest of both the owner and the renter that the rented item doesn't get damaged. If, however, it does happen that the item gets damaged, it is not straightforward for a third party to say who caused it. The owner can claim that the renter caused the damage, and the renter can argue that the damage was there before. Such a dispute could be avoided if either the owner or the renter made a photo of the rented item during the handover.

In this thesis, we will analyse how to prove the capture date of the photo media type. We will present existing partial and alternative ways of achieving this goal and offer a new solution that would provide a reasonable cryptographic guarantee so that owner nor the renter could claim a false statement.

Keywords:

Verifiable Photo Snapshot, Photo Integrity, Blockchain, Randomness Beacon, Trusted Timestamping

CERCS: P175 Informatics, systems theory

Tõestatavad Foto Hetktõmmised

Lühikokkuvõte:

Digitaalse meedia ajastu on võimaldanud laialdase ligipääsu erinevat tüüpi meediale. Fotode, videote ja heli jäädvustamine on võimalik lihtsalt ja kiirelt. Jäädvustamise hetkel lisatakse meediale tihti ka metaandmed selle jäädvustamise aja, koha ja muu kohta. Meedia jäädvustamiskuupäev võib mitmel juhul muutuda oluliseks teguriks, kui on vaja meedia põhjal mingit otsust vastu võtta. Meediafail ja sellega kaasa pandud metaandmed on aga lihtsasti võltsitavad.

Näiteks renditeenuste (mootorsõidukite rent, korteri üürimine ja mitmesugused muud) korral on nii omaniku kui ka üürniku huvides, et üüritav ese ei kahjustuks. Kui eset siiski kahjustatakse, ei ole kolmandal osapoolel lihtne öelda, kas selle põhjustas üürnik või oli ese kahjustatud juba üleandmise ajal. Omanik võib väita, et kahju tekitas üürnik, ja üürnik võib väita, et ese oli kahjustatud juba üleandmise ajal. Antud lahkheli saaks vältida, kui omanik või üürnik oleks üleandmise ajal teinud esemest pildid nii, et piltide jäädvustamis aeg oleks usaldatav.

Selles lõputöös analüüsime, kuidas tõestada pildi jäädvustamise aega, toome välja olemasolevad osalised ja alternatiivsed lahendused selle eesmärgi saavutamiseks ning pakume välja uue lahenduse, mis annab mõistliku krüptograafilise garantii jäädvustamisaja õigsuse kohta.

Võtmesõnad:

Tõestatavad Foto Hetktõmmis, Foto terviklus, Plokiahel, Juhuslikkuse Majakas, Ajatembeldus

CERCS: P175 Informaatika, süsteemiteooria

Contents

1 Introduction

In any rental service, it is always a possibility that the rented item gets damaged. Thus it's quite a common practice to grab a photo of the rented item before renting it out. This process is done to prevent the other party from claiming false statements in case the rented item gets damaged and parties don't agree on when the damage occurred. Taking a photo could potentially be beneficial to both parties. A renter may claim that the damage was there before the handover, and the owner may claim it wasn't. In such a situation, a photo taken at the time of the handover would help to prove either renter's or owner's statement.

However, just taking the photo is not enough to conclusively prove either statement. There are multiple ways either party could cheat in this situation. If the damage occured during the renting period, then the renter could claim that it was there before, and the photo showing the undamaged property is either modified or was taken before the property was rented out. If the property was damaged before it was rented out, then the owner could claim that the renter caused the damage, and the photo is either modified or was taken after the property was rented out. In section 2.2, we will identify and analyse distinct ways either party could cheat and claim a false statement about the photo.

There exists various partial and alternative solutions to the problem of proving a photo's capture date. In section 2.3, we will offer some of the partial and alternative solutions and analyse each one explicitly. In section 3, we will present some of the existing solution platforms that relate to the problem of proving the photo's capture date.

This thesis aims to offer a generic solution to the problem of the photo's capture date. The proposed solution breaks down into three distinct parts, each one specified in solving one of the attack vectors described in section 2.2. We will explain the function of each part and how they are all coupled into a single solution in section 4. Problems with the proposed solution and things left for future work are discussed in paragraphs 4.6 and 4.7 respectively.

In section 5, we shall describe a more concrete implementation for the solution proposed in section 4.

2 Overview

2.1 Motivation

In the age of digital media, many media types like photos, videos and audio recordings are widespread and accessible to anyone. It can be said that each capture of such media represents a snapshot of the time when it was taken. However, that time-point is not easily identifiable later when the media is being analysed. One could analyse the media itself and base their claims of time on the data recorded in the media. In most cases, such claims are not provable in any way, as the counterparty could easily make their claims and there is no standard way of saying who is right.

Most digital media types do add metadata to the media files and capture time is usually included. In most cases, the metadata is included as plain text data which is easily modifiable by anyone willing to put enough time into it. This raises demand for a more secure way of capturing media where whoever owns it could prove precisely when it was taken, and a false statement could not be made. In this thesis, we will focus specifically on creating such a solution for photo media type.

Taking a photo of an item is an easy way to take a snapshot of the physical state of the object at a specific time point. This is most commonly done during a renting process where both parties want to have some insurance when either party is malicious. However, this system falls apart when the malicious party challenges the photo claiming that the photo is modified or not taken during the handover. In such a situation, there is no definite proof that the owner of the photo could present to show that the photo is indeed the original and taken at the correct time. This is because photos are easily modifiable, especially now where photo modification software is widely available and can produce photos that look as they are unmodified. Similarly, the metadata included in the digital file of the photo can also be edited with minimal effort.

Renting service is the main use case that we will present as an example in this thesis.

There are however, many more applications where verifiable photo snapshots could be used. Some examples include:

- In case when someone sees a crime being committed, a verifiable photo snapshot would give irrefutable evidence for the case.
- Photo-based insurance claims [CKR17].
- Journalists could include a verifiable photo along with the published article to add credibility to the article.
- A photography contest where the photos must be taken between specific dates [sta].
- An orienteering competition where the participants have to take photos of the visited checkpoints.
- And last and somewhat different example: a hostage situation where the validity of the photo is crucial.

2.2 Attack Vectors of Non-Verifiable Solution

By non-verifiable photo insurance, we mean the process where one or both parties take a photo of the rented item to resolve potential future disputes over the visible damages that the item might have when returned to the owner. In this section, we shall describe and analyse what are the possible ways either party (renter or the owner) could claim a false statement about a damaged item in such a situation. We will only present the attack scenarios that involve photo integrity. Each attack vector will be marked with the letter **A** followed by a number.

To add context to the following attack vectors, we will first describe how well-meaning parties could benefit from having taken photos of the rented item during the handover.

- If the renter damages the property unknowingly and the damage gets discovered when the item is handed back to the owner, the owner can show pictures of an undamaged property to prove that the renter caused it.
- When handing back the item, if the owner discovers damages that he didn't know about (but were there before the initial handover), the renter can show the pictures taken during the handover to prove that the damage was there before.

Additionally, we are working under the assumption that if the renter damages the rented item, the item will stay damaged and won't be fixed before it gets handed back to the owner.

A1. Back-dating. Back-dating means putting an earlier date to the data than the actual one. In the case of a fraudulent renter, there are two possible ways of cheating by using back-dating. When the renter damages the rented item, he could take a new photo of the already damaged item and claim the photo was taken at the time of handover. Secondly, the renter can modify the metadata of the original handover photo to change the time value so that it would appear that the original handover photo was taken before the actual handover. In both cases, the false statement claimed by the renter would be *"I did not damage the item, it was already damaged during the handover"*.

A fraudulent owner can not benefit from back-dating as the original handover photo that the owner already has would show the same state in a later time.

A2. Forward-dating. Forward-dating means putting a later date to the data than the actual one. In the case of a fraudulent owner, there are two possible ways of cheating by using forward-dating. The owner could claim that the renter damaged the property even if the damage was already present. He could do that if he possesses a photo of the item taken when it was undamaged and modify the metadata of that photo to change the time value to the time of the handover, and claims that the renter caused the damage by

presenting the older photo. Secondly, the owner can modify the metadata of the original handover photo to change the time value so that it would appear that it was taken after the actual handover. In both cases, the false statement claimed by the owner would be *"The item was undamaged during the handover and the damage occurred when the item was in possession of the renter"*.

A fraudulent renter does not benefit from forward-dating as the aim of the renter is to show that the damage was present before or during the handover.

A3. Photo modification. Photo editing has evolved to a level where the photo forgeries are very difficult to detect [NWW17]. This means either party could take photos taken during the handover and adjust them to their needs. A fraudulent renter could then claim a scenario A1 without needing to take a new photo. Similarly, a fraudulent owner could claim scenario A2 without being in possession of a photo taken when the item was undamaged.

2.3 Alternative And Partial Solutions

The problem of photo integrity and verifiable photo timestamp is not a new one. Thus there are many existing partial and alternative solutions to it. However, most of them do not give a reasonable guarantee against all the attack vectors or have some other problems that may render them impractical. We will look at some of those solutions and present their benefits and shortcomings. Each example will be marked with the letter **S** followed by a number.

S1. Time-sensitive item in the photo. Probably the most widely used way to bind a photo to a specific date is to include an item in the photo that provably did not exist before. The most common example of such an item would be the days' newspaper. This method does nothing to protect against photo modification attack vector A3. Even if you

would assume that the photo was not modified, there are problems with this approach. The time-sensitive item in the photo would prove that it wasn't taken before the item existed. However, it says nothing about the period after the creation of the time-sensitive object. A malicious party could easily obtain the same item and take a new photo any time after. Thus we can conclude that such an approach would only avoid attack vector A2, but would still fail against A1 and A3.

S2. Black box camera. A perfect solution to protect against all the aforementioned attack vectors would be an implementation of a camera that allows absolutely no interference and signs and timestamps each photo it takes. By no interference, we mean that the user has no control over the internals of the camera and could only point the camera to the subject and take a photo by pressing a button. This would mean that the camera would act as a black box. Such a solution would protect against all attack vectors A1, A2 and A3. A big drawback of such a solution is that the users would have to trust the manufacturer of the camera and believe that the camera is indeed tamper-proof, which is very hard to prove. Additionally, having a dedicated black box camera solution would greatly lower the availability of the service, meaning that very few people would have access to it. It's also worth mentioning that such cameras can be costly.

S3. Agreeing on the photo and signing it during the handover. Another approach would be to require both parties to sign the photo during the handover. This approach doesn't say anything about the actual photo being signed. It would mean that both parties agree on the photo being a proper representation. Although this approach doesn't directly protect against any of the aforementioned attack vectors, it is sufficient to resolve the underlying issue in renting service. A considerable drawback of this approach is that it requires actions from both parties.

3 Related Work

3.1 Truepic

Truepic is a blockchain-based image verification platform. Truepic is mostly a server-side solution where users of the platform use Truepic mobile application to capture photos and forward them to the Truepic server for processing. In addition to the photo itself, the application also collects and sends additional metadata, such as GPS location. Once the photo is sent, the server performs over 20 verification tests on the photo. These tests include capture device integrity, capture date and time verification, location and checking if the photo is not a recapture of an existing one. Trupic doesn't go into specifics of how these tests are performed. Thus we can not definitively conclude the trustworthiness of the system. Each photo is also stored in Bitcoin blockchain which is not controlled or hosted by Truepic to make sure the record of the photo is immutable. For each photo sent to the Truepic server, Truepic also creates a separate verification page where recipients can view the photo and see the results of the performed verification tests [tru].

3.2 YouPic Blockchain

YouPic Blockchain is a decentralized photography platform dedicated to giving photographers and content creators more control over their work and providing the tools they need to monetize it. YouPic uses a blockchain implementation built by themselves to provide copyright tools to help creators to attach licences to their content, manage smart contracts and track unauthorized usage of their content [You].

Although the problem that YouPic aims to solve is not the same as covered in this thesis, both problems share the key similarities which depend on proving the integrity of a photo.

3.3 Picture-Based Crop Insurance (PBI)

PBI is a solution that aims to improve the accessibility and coverage of high-quality crop insurance for smallholder farmers who are living under a thread of crop damage by unpredictable weather events. By leveraging the increasing smartphone ownership among smallholder farmers, the PBI system uses a time-lapse of the pictures taken by farmers themselves to verify the insurance claims. The system has been rolled out and tested in some parts of Africa and India. At the time of writing this thesis, the PBI system doesn't use any cryptographic tools to verify the validity of the photos. Some measures, like GPS location restrictions, were put in place to prevent malicious activity [CKR17].

Even though this solution currently doesn't use strong protective measures against malicious farmers, the system has shown promising results to make crop insurance accessible and affordable. However, for a production-ready solution, further protective measures, like the ones discussed in this thesis, would be necessary.

4 Proposed Solution

The solution aims to solve all three problems with the non-verifiable solution mentioned in section 2.2. One of the goals is also to find a compromise between making the system completely tamper-proof and ease of use. We will describe a generic solution which, if implemented, would provide the users with reasonable guarantee that the photos taken by the application are unmodified and indeed taken when claimed. A more specific implementation for this solution will be given in section 5 as an example.

The proposed solution composes of three distinct parts, each part solving one of the issues discussed in section 2.2. We use the concept of randomness beacons to prevent forward-dating, trusted timestamping to prevent back-dating and more specially designed camera session to ensure that the actual photo was taken at that specific moment and not injected into the process maliciously. The time bounds set by the solution are illustrated in figure 1.

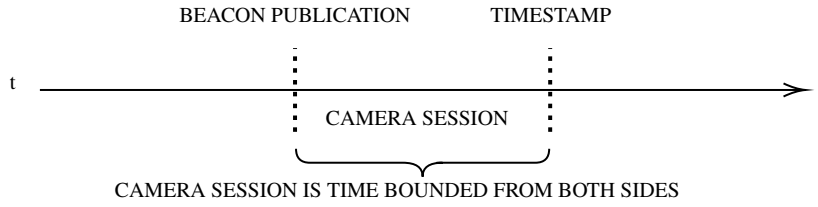

Figure 1. Time-bound restrictions on the camera session.

4.1 Randomness Beacon

The concept of randomness beacons was first proposed in 1981 by M. Rabin [Rab81]. A randomness beacon is a public utility that produces fresh public randomness at timed intervals. Each publication (pulse) would generally also contain metadata about the publication. Metadata could include a time value when it was published, a reference to the previous value, etc. National Institute of Standards and Technology (NIST) started

the Randomness Beacon project in 2011. The project aimed to offer an implementation of randomness beacon and promote trusted public randomness utility [CSD19]. Randomness beacons have many usages like assigning random officials for financial audits, randomized clinical trial and legal metrology [BGKP19].

A useful feature of randomness beacons is that each publication is verifiability tied to a specific point in time (when it was published), which means we can use it to set a time restriction in our solution to prevent forward-dating. We describe two implementations of randomness beacons in this section. The exact use of randomness beacons in our solution will be further explained in section 4.2.

4.1.1 NIST Randomness Beacon

NIST develops and manages the NIST Randomness Beacon implementation. As of writing this thesis, NIST has published two versions of their randomness beacon design, the latest one (version 2) being in a draft state [KBPB19]. We will be using version 2 when describing the NIST implementation of randomness beacon. The NIST beacon implementation periodically outputs a publication which contains 512 fresh random bits. All of the bits of the randomness are timestamped, signed and put into a hash-chain as seen in figure 2.

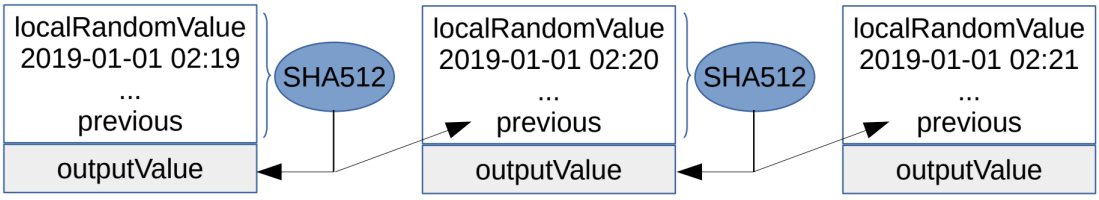

Figure 2. Hash chain that is generated from subsequent pulse publications (figure taken from [KBPB19]).

This ensures the publication can't be changed afterwards. NIST also specifies the beacon protocol which enables users to interact with the beacons to obtain information

about past publications. A general overview of NIST service beacon is visualized in figure 3. The pulses are formed in the beacon engine, which includes an internal clock, a random number generator (RNG) and all the capabilities that are needed to run the beacon. A frontend application communicates with the database of the beacon engine to answer user's requests. The database contains all information about past pulses.

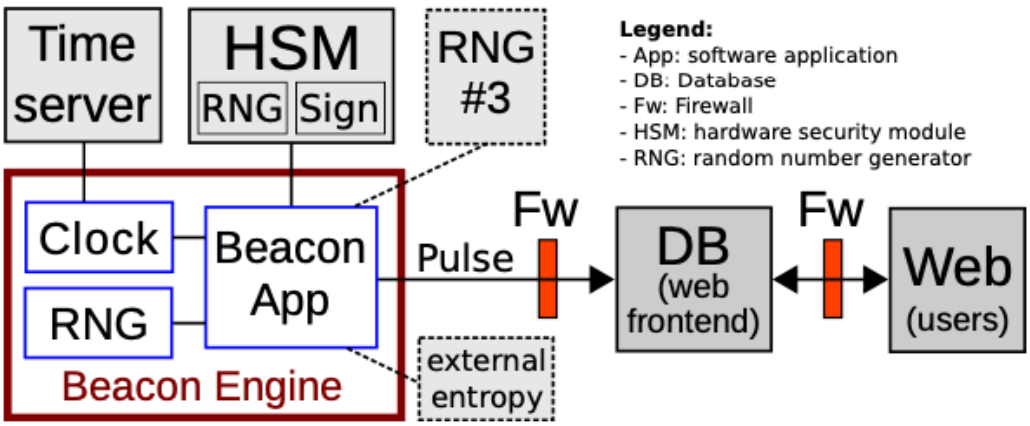

Figure 3. Beacon service beacon (figure taken from [KBPB19]).

4.1.2 Non-Deterministic Time System

Not to be confused with the "non-deterministic time" in complexity classes, a non-deterministic time (NDT) system is a way of generating a representation of time in which the value of the next time point is non-deterministic. NDT is an alternative use case for blockchain-based timestamping. For context, an example of a deterministic time system would be any type of digital or mechanical clock where the value associated with the next time point is known in advance. Non-deterministic property of the system means that the time value of a specific moment won't be revealed before the moment actually occurs and can't be predicted. If the NDT value corresponding to the time when a photo was taken were to be organically included in the photo, then the validity against forward-dating could be checked by comparing the NDT value included in the photo

against the NDT value corresponding to the time when the photo was taken. If the values do not match, the photo can be assumed to be altered.

A possible implementation of an NDT system is a system of globally distributed hash-tree document authentication infrastructure. In such a system all the client's documents are hashed and a global hash tree is built periodically at specific intervals. The root hash value of the global tree is made public and represents a new time value [GTB$^+$]. Such NDT system is a specific type of randomness beacon. The randomness of each publication is guaranteed as long as there are enough users of the system. A general overview of such a system is visualized in figure 4.

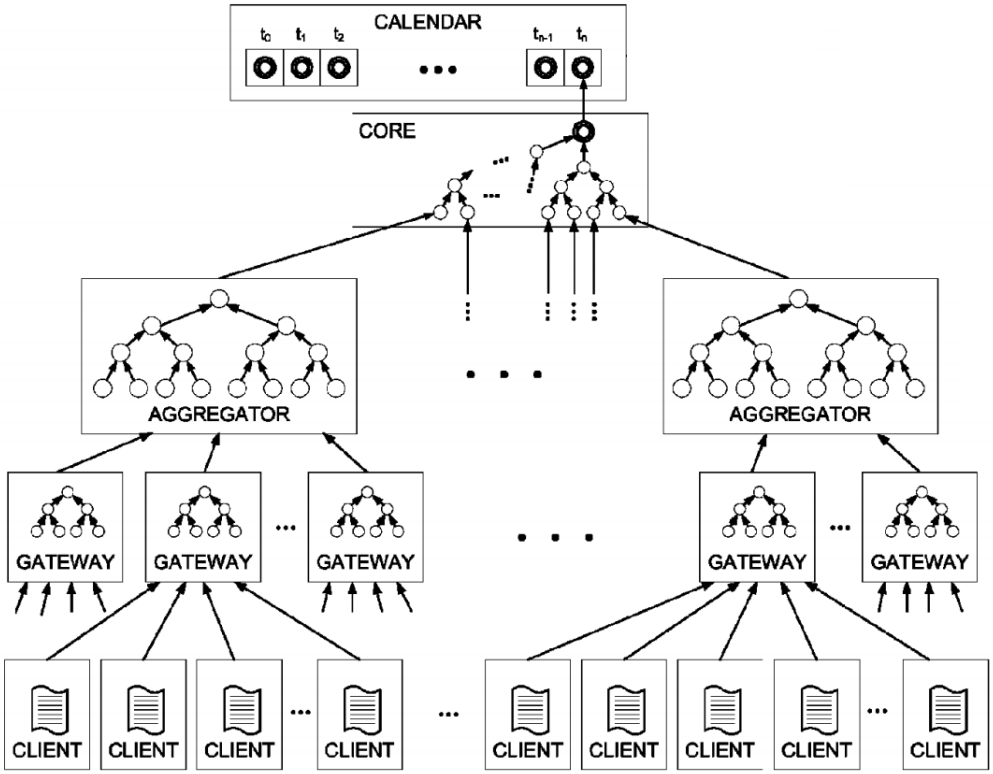

Figure 4. NDT system based on globally distributed hash-tree document authentication infrastructure (figure taken from [GTB$^+$]).

4.2 Camera Session

The main purpose of the camera session is to ensure that the session is occurring live. To achieve this, we use a similar idea to the one discussed in "S1" paragraph where an object which could not have existed before a specific time-point was present in the photo. Instead of using a physical object like presented in the example, we will be using the beacon publication value of a given time-point, which satisfies the same property that it could not have existed before that time-point.

In the solution S1, a time-sensitive object was always included in the actual camera feed. The same solution would be possible with the given beacon publication value. The user of the application would need to write down the beacon publication value on a piece of paper and make it visible to the camera. Although doable, it would be highly impractical. In our proposed solution the beacon publication value itself won't be included in the actual video feed. Instead, we will be using an augmented reality solution where the user would need to interact with the augmented object projected on the feed and navigate it through a challenge that depends on the beacon publication value.

The challenge could be anything from simple commands to move the camera up, down, left or right to a more complex augmented reality solution where the user has to pass through a generated maze. The only requirement is that the generation of the challenge is a function of the beacon publication value and depends on it. Once the user passes the challenge, he can be allowed to take the photo. The challenge session feed ends once the user completes the session by pressing the capture button. The captured photo must be set as the last frame of the challenge feed, which ensures that the photo is linked to the challenge feed. After the session ends, a proof container for the photo shall be created.

The length of the session should be made as short as possible to narrow down the time when the photo was taken as much as possible and to reduce the size of the proof container. For a specific implementation the maximum session length will vary depending

on the specific methods.

4.2.1 Proof Container

To later verify that the photo was taken after the challenge has passed, we need to create a container that includes all the data needed for verification. The container verification process is discussed in section 4.5.2. The contents of the container must contain the following items:

- Beacon publication value.
- Time value corresponding to the beacon publication value.
- Tracking data of the challenge session.
- The video feed of the challenge session.
- The photo itself (can be omitted and extracted as the last frame of the video).

4.2.2 Verifiable Video Snapshots

As an alternative, a verifiable video could be used instead of the photo. This would mean modifying the challenge session so that instead of ending the session when the user presses the capture button, the session continues until the user stops the recording. This change would require no further modifications to the solution and could be easily implemented as a replacement or additional feature depending on the use case.

4.3 Video Feed Analysis

The trustworthiness of the solution relies on the effectiveness of the video feed analysis. In this section we will discuss different kind of video feed verification techniques that

would improve the trustworthiness of the solution. An alternative solution for the video feed analysis will be discussed in section 4.3.3.

When and where the video feed analysis should be performed is also an important question. The analysis could be performed during the capture process or verification process:

- **During capture.** Performing the video feed analysis during the capture process would mean a significant time increase for the duration of the capture process. On the positive side, it would help to identify an invalid session early and prevent creating invalid containers.

- **During verification.** This option increases the verification process time but allows the usage of the most up to date verification techniques.

Considering the previous arguments, we recommend performing the video feed analysis during the verification process. The analysis could be either performed either on device or offloaded to a cloud environment:

- **On device.** Running the analysis on the device itself can be costly as the processing power of the device is limited.

- **Offloading to a cloud environment.** We can offload the work to a cloud environment and relay on the results provided by it. Offloading works under the assumption that we can trust the cloud provider and the communication with it.

Even if the video feed analysis is run during the capture phase, we would still recommend running it during the verification process so that the verifier doesn't have to rely on the capturer. If the analysis requires a significant amount of computing power, then it should be offloaded to a cloud environment.

4.3.1 Continuous Feed Detection

The described camera session challenge solution only works under the assumption that the video feed is continuous. Otherwise it would be possible for the adversary to concatenate the challenge session video with any photo. One possible solution for this problem is to apply a technique called shot transition detection. It is an automated process of detecting transitions between different shots in a video feed [GS12]. This process can be applied to the whole camera session feed. If the detection identifies any transitions, the session can be assumed to be fraudulent.

The speed of the transition detection is also important as the user would like the results as soon as possible. In recent years new research has been done into faster and more efficient shot transition detection techniques like TransNet [SML19] which uses a deep network for detecting the transitions. If the shot transition detection time becomes an issue when the photo is being taken, then the detection can be run afterwards (when validating the proof container).

4.3.2 Augmented Feed Detection

If an adversary were able to build the video feed themselves, then it would be possible to generate the challenge session video that would pass the challenge. Such an attack would require more effort from the adversary, but is theoretically possible. An additional feature of the camera session that would prevent this would be to detect if the session is augmented or not. The solution for this will not be covered in this thesis and is left for future work.

4.3.3 Manual Inspection

An alternative to the automated video feed analysis would be a manual inspection of the video feed during the verification process. In a real world setting this would only be required when an actual dispute needs to be settled and certainty in the integrity of the

proof container needs to me maximised. A manual inspection would enable detection of fraudulent session feed that are distinguishable by the human eye. Granted, fraudulent feeds that are generated by more advanced methods would go undetected. However, on the positive side, the adversary would only have limited time to generate the fraudulent challenge feed since the generation has to happen between the beacon publication time and the time of the timestamp.

4.4 Trusted Timestamping

Trusted (digital) timestamping is a process of tracking and managing the modifications to data in such a way that no one should be able to modify the data and keep the validity of the timestamp. A timestamp proves the existence of the data before the specific date which solves the problem of back-dating, since it is impossible to obtain a timestamp for a date that has already passed.

There are multiple examples of how to achieve the existence of a digital timestamp. The most common way is to use a Time Stamping Authority (TSA). TSA is a trusted third party that issues timestamps [ACPZ01]. To timestamp a document, the client uses hashing to calculate the fingerprint for the document. That fingerprint (hash of a document) is sent to the TSA and is signed along with the time value using the private key of the TSA. The client will receive the timestamp for the document.

The solution presented in this thesis will use trusted timestamping to sign the proof container generated by the camera session to bound the time of the session and make the proof container immutable. figure 5 visualizes the sequence of messages shared by the solution application, TSA and randomness beacon provider.

4.4.1 Timestamping with Decentralized Blockchain

Blockchain is a data structure containing a list of records. Each record included in the blockchain is cryptographically linked to the previous record. Linking is done by

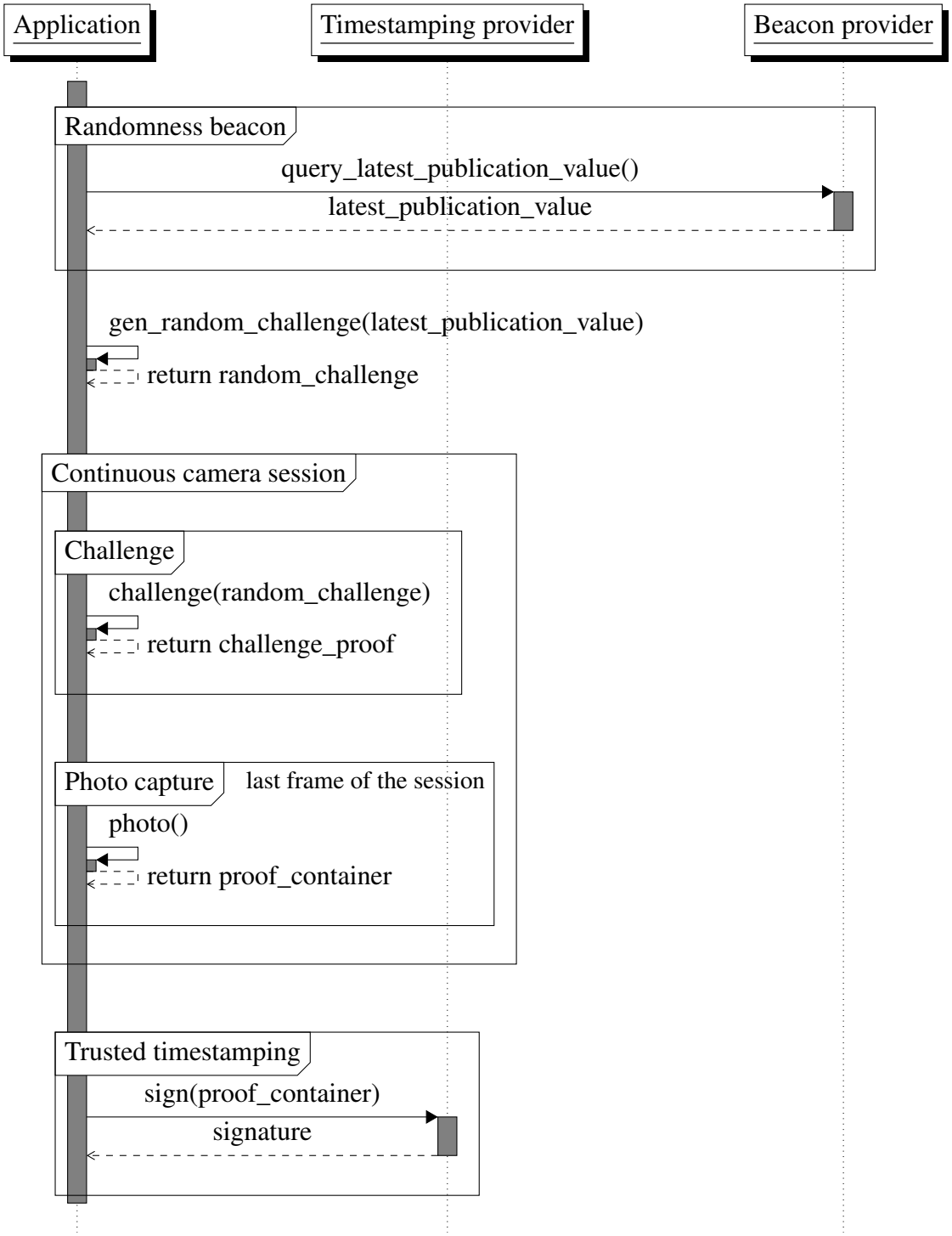

Figure 5. High level overview of the proposed solution illustrating the three distinct parts of the photo capturing solution.

including the hash and the timestamp of the previous block to the added block. One of the key characteristics of blockchain is that once the data is persisted, it can not be changed. This feature comes from the fact that the blocks are cryptographically linked, and changing the data of a single record would invalidate the next, as the hash of the previous block changed.

An alternative way of implementing timestamping is by using blockchain [HS91]. Similarly to TSA solution, the hash of the document is sent to the blockchain. But instead of signing the hash using a private key, the hash is published in a globally distributed chain. From the inherent property that the records in a blockchain can not be changed, the existence of the document hash in the blockchain will prove the existence of the data before the specific date. An added benefit of using blockchain for timestamping is decentralization which means no trusted third party is required to prove the timestamp in the future. Timestamping data on blockchain already has many use cases in securing the integrity of media files, e.g. video coming from dashboard cameras [GKB16].

4.5 Verification

Once the application completes the camera session and the proof container is time-stamped, the user will gain access to a signed proof container for the photo. To verify that container is valid, a series of verification steps must be performed. These steps are independent and can be performed in no particular order. We will present the steps in a logical order which would enable detecting tampered container as soon as possible. High-level overview of this system represented as a sequence diagram is seen in figure 6.

4.5.1 Timestamp Verification

Before the contents of the container can be validated, one must first check if the timestamp for the container is valid. I the timestamp doesn't correspond to the container, we can not be sure of the integrity of the container and should assume it to be tampered. Only after

the timestamp verification succeeds, the container verification results can be trusted.

4.5.2 Container Verification

Container verification involves three key parts.

1. Firstly, the beacon publication value inside the container must be compared against the real publication value corresponding to the time value in the container. If the publication value from the beacon system calendar doesn't match the publication value included in the container, the container can be assumed to be tampered. Only when the container publication value is verified, the verifications depending on this value can be trusted.

2. Secondly, the challenge for the challenge session must be regenerated using the publication value and the generating description which are both included in the container. The regenerated challenge must be checked against the video feed challenge tracking data included in the container. If the regenerated challenge differs from original or user had failed the challenge session, verification fails.

3. Lastly, as discussed in section 4.3, video feed must be continuous to prevent a malicious user from altering the video feed. This process is the most time-consuming part of the verification process and theoretically could be pre-processed on the cloud. Similarly, any kind of additional video feed analysis techniques could be applied to improve the trustworthiness of the system. This also means that the verifications performed by the system could improve in time, even for pre-existing containers as new analysis techniques can be applied to challenge videos.

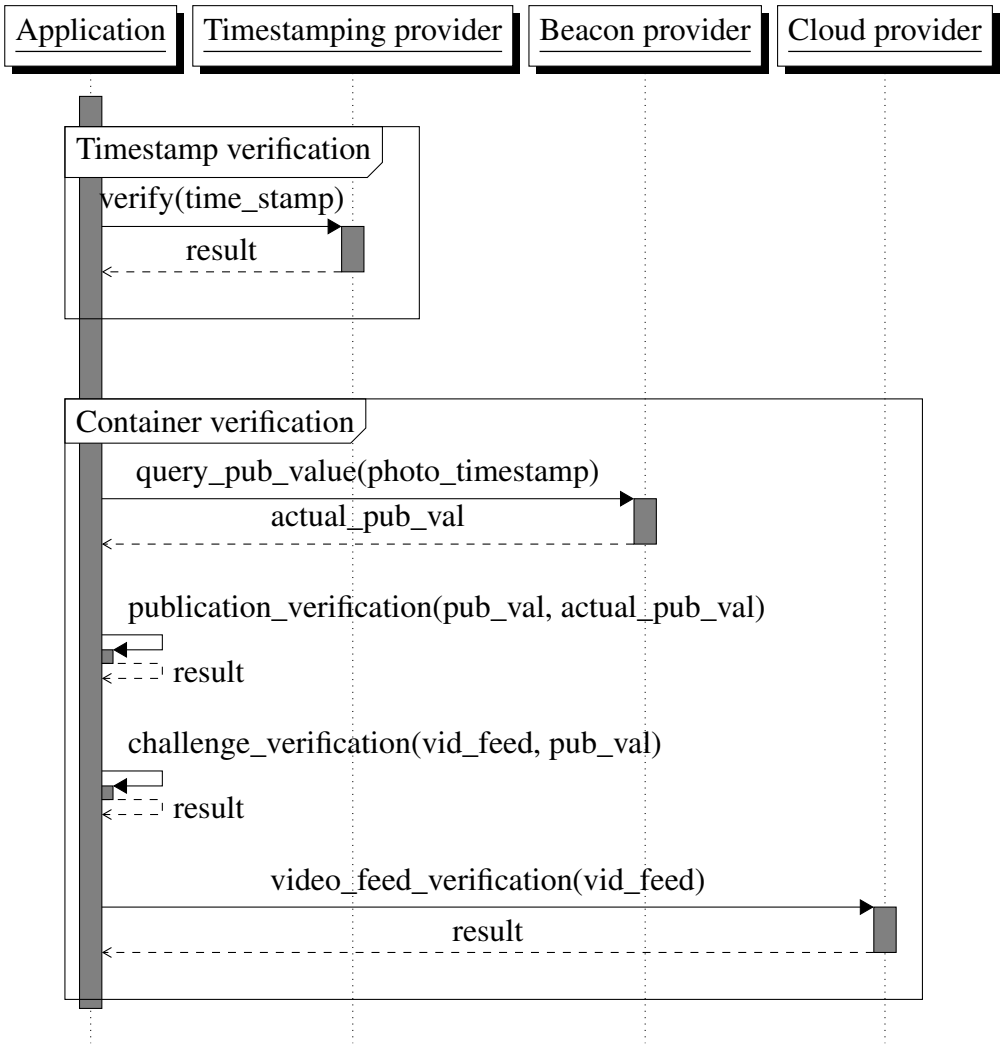

Figure 6. High-level overview of the container verification process.

4.6 Problems

Size of the proof container. A drawback of this solution is the size of the generated proof container. This is because, in addition to the photo itself, it has to contain the video of the camera session which can be quite large compared to the photo itself. One possible solution for this would be to limit the time the camera session can take place, thus lowering the size of the video file. This however doesn't solve the issue completely as we can not use a very tight time window for the session because solving the challenge by user can take some time.

Dependency on a continuous feed. As mentioned in section 4.3, the challenge session solution depends on the fact that the session itself is continuous and the adversary should not be able to "paste" together the challenge video. It would be highly beneficial if this dependency could be avoided. Otherwise the strengthening of the solution boils down to making the shot transition detection as accurate as possible.

Longer than usual photo capturing process. Compared to the typical "point and shoot" type of scenario, the proposed solution takes more time since the user has to pass the challenge session. The time of taking a single photo will depend on how long it takes the user to solve the challenge and is limited to however long implementation sets the maximum length of the session. The speed of the session also depends on the usability of the application itself, thus a good user experience (UX) design for the challenge session is crucial.

4.7 Future Work

Further video feed analysis techniques. As mentioned in section 4.2, the camera session should also check if the session occurring in the feed is augmented or not. This however would require much more in-depth solution involving machine learning and

other techniques to detect such sessions. The specification of this solution is left for future work.

Support for other media types. The presented solution is designed specially for the photo media type and won't work for most other types. A potential future work can be done by designing similar solutions for other media types like audio and video where the randomness beacon value is encoded into the captured data some other way.

Additional metadata. To improve the number of use cases the solution supports, further data could be added to the proof container. Some examples include geographical information, data about the phone, data about the user, etc. For each piece of such metadata, security analysis should be done to make sure that an adversary would have hard time tampering with the data.

5 Implementation

The aforementioned solution description leaves multiple parts of the solution unspecified. These are parts that are left for the implementer to describe. There are five main points that the implementation must cover:

1. What is the maximum length of the challenge session?
2. What kind of timestamping provider to use?
3. What kind of randomness beacon system to use?
4. What does the camera session challenge look like?
5. How does the challenge depend on the beacon publication value?

In this section we will provide a more specific description of the aforementioned solution design described in section 4. Keep in mind that this is only one of the possible implementations and many others can be designed.

For both randomness beacon and timestamping we will be using a single system called KSI Blockchain that provides both of the utilities. This also eases the implementation process as communication is done with a single third party.

5.1 KSI Blockchain

KSI Blockchain is a globally distributed system which provides services for creating server-supported digital signatures and timestamping [BKL13]. The system is based on cryptographic hash functions and special kind of binary trees called hash trees [Mer88]. A binary hash trees built by KSI are data structures where all leaf nodes are labelled with a hash of some sort of data block and all non-leaf nodes are labelled with the hash of its child nodes.

A KSI aggregator is a machine that periodically builds the binary hash tree. KSI uses multiple levels of aggregators where down level aggregators get their input hashes from clients and subsequent aggregators get their input hashes from down level aggregators and so on. This process is done to support a very large number of clients at the same time. A round finishes when the root hash value of a top-level aggregator is put to the calendar blockchain as illustrated in figure 7. Each client will receive a signature for the hash which contains the necessary hash values needed to replicate the aggregation tree path from their input hash to the top of the root hash value. Only the calculated root hash will be persisted in the public calendar blockchain and the rest of the tree is discarded. This process is done once every second.

The calendar blockchain is a tree where each leaf node corresponds to a specific second since 1970-01-01 00:00:00 UTC and each leaf is the root hash value of a global aggregation tree that was built on that second. Guardtime OÜ, the maintainer of the KSI Blockchain, periodically publishes the hash calendar's root hash value in public physical and electronic media to improve the reliability of the signature verification service.

KSI Blockchain is a decentralized blockchain timestamping service, thus we can use it as a trusted timestamping provider for our implementation.

Since the root hash value of an aggregation round depends on all of the clients' inputs, it will be completely unpredictable as long as enough clients are using KSI. This property along with the per-second update makes it perfect for using it as a non-deterministic time system where new time value is published every second. For this implementation we shall use KSI Blockchain as a randomness beacon provider which is an alternative use case for KSI Blockchain. Since we are using a single service for both the randomness beacon and timestamping, it simplifies the implementation process as we only have to communicate with a single provider.

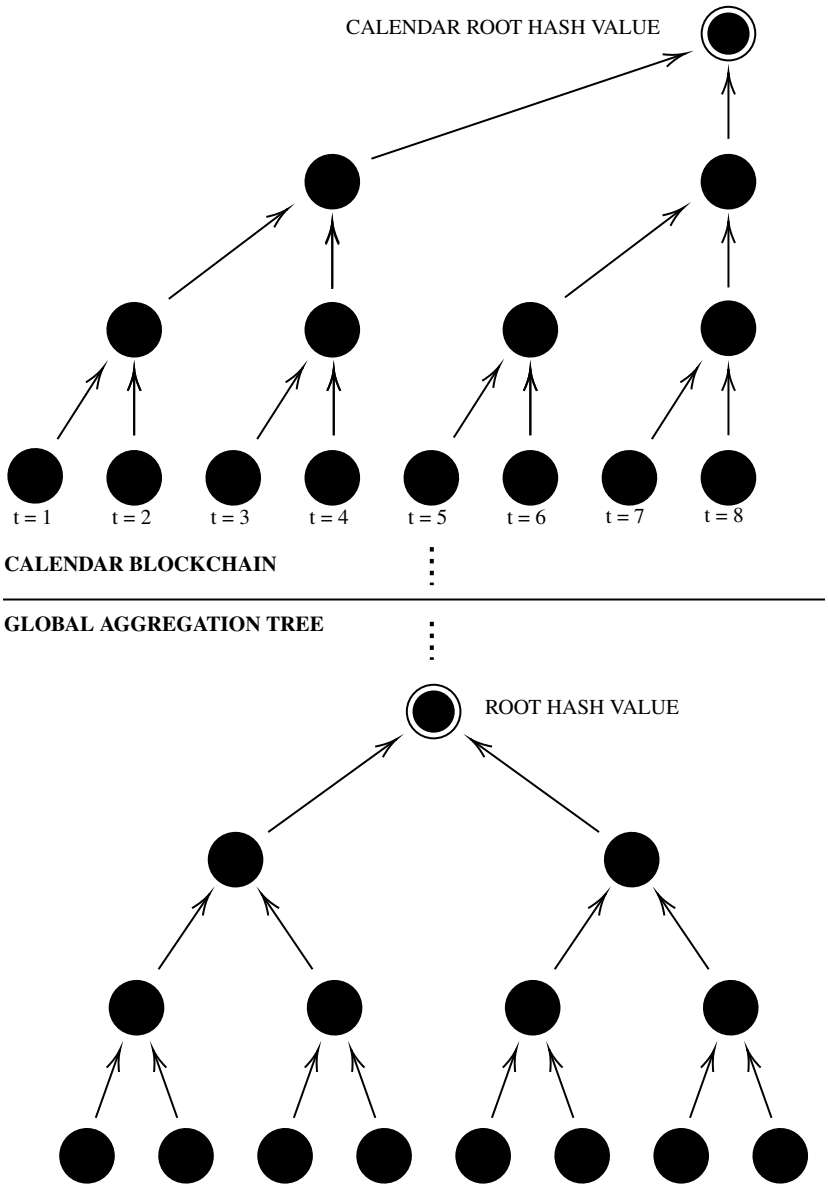

Figure 7. High-level structural overview of how the KSI blockchain calendar is built.

5.2 Camera Session Challenge

We shall provide an example of a challenge system that uses an AR based system. The system shall scan the challenge feed and fix a point on the feed where an augmented point will be placed. The user has to navigate the augmented point either on the left or right side of the barrier as illustrated in Figures 8 and 9 respectively. The augmented point (represented as a black circle on the figures) has to be fixed to a point on the camera feed so that the user could change its position by moving the camera device.

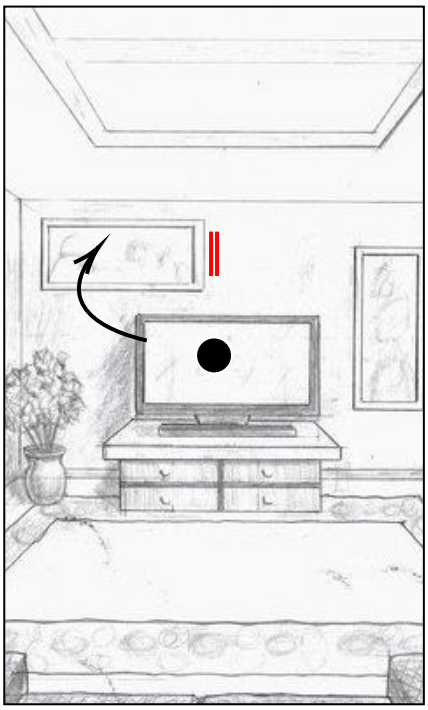

Figure 8. Application guiding the user to move the AR point past left of the barrier.

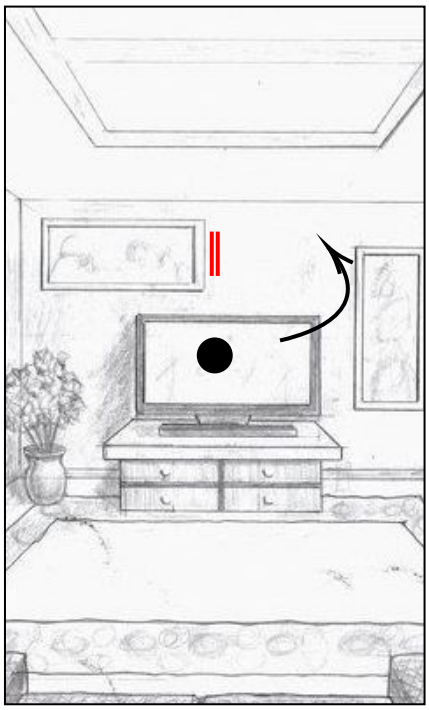

Figure 9. Application guiding the user to move the AR point past right of the barrier.

Taking figure 8 as a starting example, if the user passes the barrier on the correct side the application would respond positively and if the user passes the barrier on the incorrect side the application would respond negatively as demonstrated in figures 10 and 11 respectively.

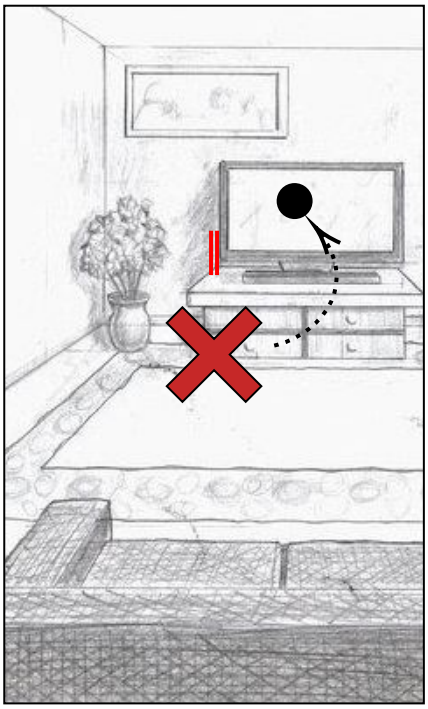

Figure 10. An example of how the application reacts to a correct movement.

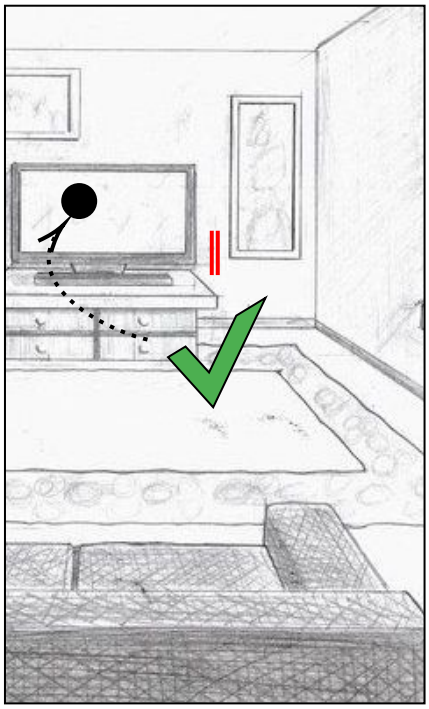

Figure 11. An example of how the application reacts to a incorrect movement.

The binary gate solution was chosen as an example because it's easiest to implement and its security is easy to analyse. If we set the challenge as a single binary gate then the pre-recorded session would have an exactly 50% chance of passing the challenge. If we increase the number of gates to 2, the adversary will have a 25% chance of passing. It's easy to see that as we increase the number gate iterations t, the success of the adversary goes down as seen in table 1. Thus the number of iterations acts as a security parameter which could be adjusted according to the required security level.

Number of iterations (t)	**Probability of success**
1	0.5
2	0.25
3	0.125
4	0.0625
5	0.03125
6	0.015625
7	0.0078125
8	0.00390625
9	0.001953125
10	0.0009765625

Table 1. Adversary's probability of success depending on the number of iterations in the challenge session.

There are multiple ways to design the function that generates the gate passage sequence from the beacon publication value. The function itself does not have to be fixed and could be left configurable from a validated set of functions. The simplest of such functions would be to use the first t bits of the beacon publication value and generate left passing gate when the bit is set to 0 and right passing gate when the bit is set to 1.

Once the user passes the t challenge rounds, we can say that the session is live with

confidence $1 - \frac{1}{2^t}$ and the user can be allowed to end the session. Once the user ends the session, the last frame of the session will be set as the snapshot photo. After the session ends, a proof container can be created.

The maximum length of the challenge session shall depend on the number of challenge rounds t. Each round should be completed within 2 seconds which gives us a total maximum time $2t$.

6 Conclusion

We described a solution for designing a mobile application that enables the user to take a photo in such a way that the capture time of the photo is provably included in a narrow timeframe. The timeframe starts when the camera session starts and ends when the camera session ends.

The starting time of the camera session is fixed by using a random beacon publication value for the given time. The ending time of the session is set by using a trusted timestamping solution. We can guarantee that the camera session took place between the two time points because trusted timestamping ensures that the signed proof container existed in that state at the given time and since the proof container contains a random beacon publication value, the proof container must have been created after the publication value was released.

It is important to note that the random beacon publication and trusted timestamping proves that the session took place in that time period and does not ensure that the session that took place wasn't fraudulent. If we had presented this as a solution, anybody would have been able to inject whatever media into the process. To solve this issue, we took the initiative from an example provided in section 2.3 and attached a time-sensitive object to the session. The object had to be organically included in the capture process so that it would be as difficult as possible to add it to pre-existing media. We chose the beacon publication value as the time-sensitive object since it satisfies the criteria that it could not have existed before that time point.

The only way to include the publication in the actual photo would be to physically write down the value and take the picture with it. This, however, would be highly impractical. Instead, we generate capture session challenges from the beacon publication values. Only if the user passes the challenge, the session can be considered valid.

The challenge video feed works under the assumption that the feed is not augmented and is continuous. Otherwise, it would be possible for the adversary to pass the challenge

session and include a pre-existing photo in the container. Thus we can conclude that the security of the system depends on how well the video analysis is able to detect malicious content. The positive side of this dependency is that the security of the system can improve in time when new analysis techniques are developed to counter new adversarial capabilities. Additionally, the validity of the camera session feed could always be manually inspected for extra confidence.

References

[ACPZ01] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Rfc3161: Internet x.509 public key infrastructure time-stamp protocol (tsp), 2001.

[BGKP19] Luís T. A. N. Brandão, Carlos Galhardo, John Kelsey, and René Peralta. Usages of public randomness. https://csrc.nist.gov/CSRC/media/Presentations/usages-of-public-randomness/images-media/20191106-NIST-ITL-Science-Day-poster-Beacon--ts-20191118.pdf, 2019.

[BKL13] Ahto Buldas, Andres Kroonmaa, and Risto Laanoja. Keyless signatures' infrastructure: How to build global distributed hash-trees. volume 8208, pages 313–320, 10 2013.

[CKR17] Francisco Ceballos, Berber Kramer, and Miguel Robles. The feasibility of picture-based insurance (pbi): Smartphone pictures for affordable crop insurance, 12 2017.

[CSD19] Information Technology Laboratory Computer Security Division. Interoperable Randomness Beacons | CSRC, June 2019.

[GKB16] Bela Gipp, Jagrut Kosti, and Corinna Breitinger. Securing video integrity using decentralized trusted timestamping on the bitcoin blockchain. In *MCIS*, 2016.

[GS12] Lakshmi Priya G.G. and Domnic S. Modified color layout descriptor for gradual transition detection. In P. Balasubramaniam and R. Uthayakumar, editors, *Mathematical Modelling and Scientific Computation*, pages 421–428, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[GTB+] Michael Gault, Ahto Truu, Ahto Buldas, Martin Ruubel, and Jeffrey Pearce. Non-determininistic time systems and methods. http://www.freepatentsonline.com/9178708.html.

[HS91] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *J. Cryptology*, 3(2):99–111, 1991.

[KBPB19] John Kelsey, Luis T. A. N. Brandao, Rene Peralta, and Harold Booth. A Reference for Randomness Beacons, May 2019.

[Mer88] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[NWW17] Sophie J. Nightingale, Kimberley A. Wade, and Derrick G. Watson. Can people identify original and manipulated photos of real-world scenes? *Cognitive Research: Principles and Implications*, 2(1):30, Jul 2017.

[Rab81] M. Rabin. Transaction protection by beacons. Technical report, Aiken Computation Laboratory, Harvard University, 1981.

[SML19] Tomáš Souček, Jaroslav Moravec, and Jakub Lokoč. Transnet: A deep network for fast detection of common shot transitions. *ArXiv*, abs/1906.03363, 2019.

[sta] How to detect if a photo's metadata has been changed? https://photo.stackexchange.com/questions/43150/how-to-detect-if-a-photos-metadata-has-been-changed. [Online; accessed 09-August-2020].

[tru] Truepic | Technology. https://truepic.com/technology. [Online; accessed 04-April-2020].

[You] YouPic. YouPic Blockchain – The world's first decentralized photography platform. https://youpic.com/blockchain. [Online; accessed 05-April-2020].

Appendix

I. Glossary

Renter Person who rents the item from Owner.

Owner Person who is the legal owner of the item and rents it to the renter.

Item The item handed over from owner to the renter.

Handover Time frame or point when the item was handed over from owner to the renter.

Snapshot A photo representation of the item at a specific time point.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Risto Pärnapuu**,

*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Verifiable Photo Snapshots,

   *(*title of thesis)

   supervised by Sven Laur and Ahto Truu.

   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Risto Pärnapuu

15/05/2020