An Authenticated Camera

John Kelsey Bruce Schneier Chris Hall
Counterpane Systems, 101 East Minnehaha Parkway, Minneapolis, MN 55419
{kelsey,schneier}@counterpane.com
hallc@cs.colorado.edu

Abstract

We develop protocols for an authenticated camera that allows people to verify that a given digital image was taken by a specific camera at a specific time and specific place. These protocols require interaction between the camera and base station both before and after a series of images are taken.

1 Introduction

Analog and digital cameras are potentially very useful devices for proving that some event happened. The most obvious application for this is in gathering evidence for a trial. We develop a protocol to certify digital images involving the Camera, the Policeman, and one or more trusted Servers. The Policeman uses the Camera to take pictures. These pictures, along with some additional cryptographic data, are authenticated in various ways by the Servers and the Camera. Eventually, a Jury may review the images, and should be convinced of certain things about the images: their the approximate range of time in which they were taken, etc.

1.1 Authenticating an Image

The most basic trait we want in an image is authenticity. That is, we want proof that an image is what was actually captured by the camera at a given time. We would also like to know that no images from the camera have been withheld: i.e., that a sequence of images represents all the images taken by the camera between the first image and the last image. Image authenticity is a problem with current digital and analog cameras: it is possible to digitally alter an image so that is it difficult to tell that it has been altered. We use digital signatures [6] to ensure the authenticity of an image from our camera, and chained hashes [8] to ensure completeness (i.e., that all pictures taken at a site are given to the Jury).

There are two basic attacks to this system, both non-cryptographic:

a. Generate some altered image and then to project it into the camera's lens. This could be quite difficult to counter, though some possible methods are discussed below.

b. Stage some event, using actors and props as needed, to create a valid-looking image which can be taken by the Camera. This will only be distinguishable from the real thing only by carefully examining the picture for signs of fraud. There are some technical measures that can make this harder, discussed below, although the biggest barrier is the amount of expertise and expense required to stage a picture convincingly enough that it will survive any scrutiny.

1.2 Binding an Image and a Time

Another trait we may want is an authenticated timestamp. This would allow us to make strong statements about when an image was taken. To ensure valid timestamps, we must have access to some system whose clock we trust not to be subverted. In some situations, we may simply interact with a digital timestamping service [4] to note when we took each image, or to note when we left with the camera, and when we came back. In other situations, we will have a trusted base station with an accurate clock, or a trusted clock in the camera.

There are two basic attacks on this idea:

a. The clock can be tampered with. Assuming a really wealthy and capable opponent, this could be very difficult to stop. Interacting with other secure clocks provides significant resistance to this kind of attack.

b. It is easy to prove that some image existed before time T: simply hash the image and register it with a digital timestamping service. It is much harder to prove that an image was not taken before time T. We will discuss some methods for ensuring this below.

1.3 Binding an Image and a Location

One of the more effective ways to guard against many kinds of framing with this kind of Camera is to bind each image, not only to a time, but also to a physical location. This can be difficult, but if it's done properly it will give a Jury very compelling evidence. This can be done in two simple ways:

a. The Camera can use a Global Positioning System (GPS) receiver to get its current location. This can be included with the image data, and digitally signed and timestamped.

b. The Camera can interact with some fixed system that is known to be in a specific location. It can then be allowed a short time (perhaps half an hour) to take images. This might even be done with a police car (which might have a GPS locator on it already), to make the Camera less expensive.

The problem with using the existing GPS system for location checks is no particular security available for GPS signals. It is relatively easy to control the Camera's inputs and outputs, and to use this to feed the Camera incorrect GPS location information.

There are ways to make this more difficult, as will be discussed below.

1.4 Binding an Image and a Person

In many cases, it may be important that a given person was there when the pictures were taken. (For example, this may be the case when a person is being called as a witness.) The simplest way to do this is to have the person allow a picture of himself to be taken, perhaps with good enough resolution to make out fingerprints or retina prints. Alternatively or additionally, biometric devices may be used.

To attack this system, it may be possible to capture biometrics and to somehow superimpose them upon dishonest Policemen to make them appear to be someone else. (Imagine a pair of gloves that have someone else's fingerprints on them.) Conceptually, biometrics are just like passphrases: they can be overheard and replayed. There are techniques to make this difficult, as will be discussed below, but nothing can make this impossible.

1.5 Pitfalls in the Real World

When this kind of Camera is actually used, it is important for Policemen and Juries to understand what can and cannot be proven by the camera. For example, an authenticated image is what appeared to the camera. There is no cryptographic way to prove that the event that was captured in that image is an accurate portrayal of what actually happened at some place. For example, no amount of cryptography can prevent a wealthy adversary from hiring actors to play out some event that never happened. A bunch of pictures of a flying saucer landing from this camera are excellent evidence that what is in the image is what could have been seen at that time; it is not evidence that the flying saucer is actually a space ship instead of a prop. These issues are almost exactly the same as the ones that occur when using a conventional camera, but it's important for users and/or jurors to realize this.

2 Techniques, Data Structures, and Notation

In this section we introduce the techniques, basic data structures, and notation that we will use throughout the paper.

2.1 Cryptographic and Antitampering Techniques

The following sections introduce the cryptographic and hardware techniques used in the camera.

2.1.1 The Internal CSP

A Cryptographic Support Processor is a tamper-resistant microprocessor which supports cryptographic (and possibly some non-cryptographic) functions. It has internal data, which is secret, and it is very difficult to alter the CSP's internal operations. In this document, we will assume that the CSP is capable of calculating hashes and signing/verifying using public-key algorithms such as RSA [6]. Mathematical descriptions of the cryptographic operations used in this system can be found in [7]. The CSP is also able to maintain some internal state.

Except where we state otherwise, we will assume that the CSP is truly tamper-resistant. The CSP is also assumed to have a large amount of nonvolatile, nonsecure memory. Most of this is used to record digitized pictures.

2.1.2 The Secure Perimeter

The Camera includes several components, including a lens with auto-focus, the CSP and nonvolatile memory, a power source, an A/D converter, a processor to do image compression, a light sensor, a flash, etc. All of this is enclosed in a tamper-evident case of some kind. It should be quite difficult and expensive to open a Camera up without leaving an obvious trail. There may also be internal sensors that are monitored by the CSP to leave a trail if the Camera is opened. Optimally, the CSP, A/D converter, and compression processor would all be on the same chip, and would all be tamper-resistant.

2.2 Resisting Replays of Images

An image replay occurs when a dishonest Policeman captures an image at one time, and then tries (perhaps after altering it in some subtle way) to "replay" this image into the camera. If this can be done, the authenticity of the system is compromised.

There are three places that the image can be sent into the CSP:

a. through the lens,

b. into the A/D converter's input, or

c. into the CSP directly.

The secure perimeter of the camera case should make it quite difficult to carry out (b) or (c). No cryptographic techniques can prevent (a) entirely, but we can apparently make it very difficult.

Traditionally, replay attacks are resisted by some kind of challenge-response or time-dependent input. The Camera may use many measures to get some control over the image that comes in. Included in this are changes in focus and orientation of the lens, and changes in color, intensity, and precise direction of the flash. (The flash will always go off, regardless of light conditions.) Other possible measures include integrating digital audio recording into the system (which then allow for an entire new range of sound-based challenge-response techniques), integrating some kind of range-finding mechanism (using light, radar, sonar, or whatever) to determine actual distances, etc. Note that almost all of these will have to be analyzed offline for evidence of image replay. However, this could probably be done by computer software (important, since fraud will probably be rare enough that humans who review images and associated data will probably not expect to see anything odd).

2.3 Data Structures and Operations

The basic data structure of the image is:

- Image data
- Additional information
- Cryptographic information.

2.4 Notation

a. $\text{Sign}_{SK_t}(X)$ represents the digital signature of X under private key SK_t.

b. $\text{hash}(X)$ represents the one-way hash of X. We use SHA [5] in our system, although any other secure hash function would work.

c. X, Y represents the concatenation of X with Y.

3 The Basic System

The basic system works as follows:

1. The Camera is authorized at the Base. The Base interacts with the Camera to ensure resistance to replay attacks. The Base and the Camera agree on a 160-bit number, X_{-1}.[1] Additionally, the Base's location and current internal timestamp are noted and digitally signed by both the Base and the Camera.

   This step is necessary to establish a specific time and place where the camera appeared, and to establish an audit-trail based upon chained hashes denoted X_i. In addition the chained hashes are used to generate "challenge" values (the camera's autofocus, flash, and other settings). When we're done with this step, both the Base and the Camera have agreed upon X_{-1}, which is the initial hash in the chain.

2. The Camera takes several images on-site. The ith image is denoted as P_i. The authenticated image, A_i, is formed.

$$
\begin{aligned}
Z_i &= X_{i-1}, P_i, \text{InternalInfo}, \text{ChallengeInfo} \\
A_i &= Z_i, \text{Sign}_{SK_C}(Z_i) \\
X_i &= \text{hash}(A_i).
\end{aligned}
$$

   Note that ChallengeInfo is partially determined by X_{i-1}. The chain of X_i values provide us with an auditable hashing chain, which provide resistance to alteration of the camera's data even if the camera's private key is somehow leaked. If the Camera's private key is not leaked, then we have additional assurances about the difficulty of predicting the X_i values, since they are based partially on signatures.

3. The Camera is returned to a Base. There, the Base interacts with it to ensure resistance to replay attacks, and the initial X_{-1}, final X_i, Base's location, and current internal timestamp are noted and digitally signed by both the Base and the Camera.

If the Camera interacted with the base both before and after taking images, we have strong verification that there has been no alteration of image values, even by someone who

[1] This assumes that SHA [5] is the one-way hash function. Other functions are possible.

knows the camera's private key. We also are able to bound the camera in time and space: we know where and when it was, on two separate occasions. If the total time elapsed is only an hour or two, this gives us a significant amount of information about its possible locations, and bounds the images in time.

3.1 Initial Authorization

The Base and the Camera perform the following protocol to initially authorize secure use of the Camera:

1. The Base forms

   R_0 = a random number

   T_{-1} = a current timestamp

   C_{B_1} = certificate of PK_{B_1}, this Base's public key,

   and sends to the Camera

   $M_0 = C_{B_1}, R_0, T_{-1}, \text{Sign}_{SK_{B_1}}(R_0, T_{-1})$, where SK_{B_1} is the Base's private key.

   R_0 is used by the base to prevent any replay attacks. For example, using R_0 protects against someone else impersonating the Base and using a replay attack to start a signed, yet bogus, hash chain with the Camera.

   T_{-1} presents a specific time, which can later be used to verify that the Camera interacted with the base at this time.

   C_{B_1} simply allows the Camera to know this Base's public key.

   Note that this protocol has one serious potential flaw: if the Base and the Camera don't carefully time each others' responses, then it will be possible for an attacker to interact with the Camera in one location, and the Base in another, and to transmit their messages over a phone line or radio link. This is known as the "grandmasters' problem" [1] and there is not a cryptographic solution to it; instead, all we can do is time the responses, and time out very quickly.

2. The Camera verifies the certificate and the Base's signature, and then forms

   R_1 = a random number

   C_C = certificate of PK_C, the Camera's public key,

   and sends to the Base

   $M_1 = C_C, R_1, \text{Sign}_{SK_C}(M_0, R_1)$, where SK_C is the Camera's private key.

   By using R_1, the Camera also prevents replay attacks. For example, using R_1 prevents someone from trying to pose as the Camera and then trying to "register" it in a time and place it isn't in.

   C_C lets the Base know the Camera's public key. By including the previous message in the signature, we make a wide class of attacks impossible, and ensure that there is a cryptographic audit trail [8].

3. The Base forms

   $$X_{-1} = \text{hash}(T_{-1}, R_0, R_1)$$

   and sends to the Camera

   $$M_2 = \text{Sign}_{SK_{B_1}}(X_{-1}).$$

   The Camera already knows T_{-1}, R_0, and R_1 so it can derive X_{-1}. This step simply leaves the Camera with a digitally signed value based on R_0, R_1, and the timestamp.

4. The Camera verifies the signature in M_2 and saves it for use in the final authentication protocol.

All these steps are recorded by both Camera and Base. The Camera should be especially careful to keep track of X_{-1} and C_{B_1} as they will be required for later parts of the protocol.

At this point the Base has a fix on the Camera's location in both space and time. The Base has a fix on the former because the Camera interfaces with the Base and the latter because the protocol is interactive. The Camera has a signed timestamp which can be used later to prove when the pictures may have been taken.

There are two basic attacks to consider in this part of the protocol: someone trying impersonate a Camera and someone trying to impersonate a Base. In carrying out the former attack someone may try to "register" the Camera in a bogus place and/or time. By sending R_0 and then forcing the Camera to sign it (in the signature of M_0), the Base can be assured that it is interacting with a Camera rather than seeing replayed packets. In the latter attack someone may try to create a bogus timestamp for their Camera. By sending R_1 and forcing the Base to sign it, the Camera can be assured that it is interacting with the Base.

3.2 Taking an Image

The Camera performs the following set of operations to take an image:

1. The Camera captures, digitizes, and optionally may compress the image taken into P_i. This is the standard functionality of any digital camera, though we

note that all of these functions take place inside the secure perimeter.

2. The Camera collects the internal state of the Camera, based on intrusion detection sensors and any optional devices (as discussed below), into Y_i. By collecting internal sensory data, and making the image rely upon it, we may be able to detect suspicious readings that indicate that some kind of attack on the Camera has taken place.

3. The Camera forms

   $W_i = \text{hash}(\text{Sign}_{SK_C}(X_{i-1}))$, where W_i is the ChallengeInfo discussed in the protocol outline.

   and uses W_i to determine its specific auto-focus setting, flash intensity, F-stop, etc. Note that this is a value that can't be known ahead of time by an attacker, even one of the Servers, without knowledge of the Camera's private key.

4. The Camera forms

   $$Z_i = X_{i-1}, P_i, Y_i, W_i$$

   and then forms and stores the authenticated image

   $$A_i = Z_i, \text{Sign}_{SK_C}(Z_i)$$

   and calculates

   $$X_i = \text{hash}(A_i)$$
   $$i = i + 1.$$

   By making X_i dependent upon A_i, we make it dependent upon the Camera's private key (because of the signature), and also on all the information available to the Camera for all previous images. Because A_i also includes X_{i-1}, we also have a continuous hash chain.

3.3 Final Authentication

A Base, not necessarily the same as in the first part of the protocol, and the Camera perform the following protocol to authenticate the images in the camera. (If the Bases are different, then it may raise some synchronization questions in some cases.)

1. The Camera forms

   R_2 = a random number

   C_C = certificate of PK_C, the Camera's public key,

   and sends to the Base

   $$M_3 = C_C, R_2, \text{Sign}_{SK_C}(R_2).$$

   Again, the random number R_2 is used to prevent replay attacks.

2. The Base verifies the certificate and signature. If they are OK, it forms

   R_3 = a random number

   T_n = a timestamp, where n is the number of images taken.

   C_{B_2} = certificate of PK_{B_2}, the Base's public key.

   and sends to the Camera

   $$M_4 = C_{B_2}, R_3, T_n, \text{Sign}_{SK_{B_2}}(M_3, R_3, T_n).$$

   By including the random number and the timestamp, we prevent replay attacks, and also bound the camera in space and time. Just as with the first protocol, this is vulnerable to the grandmaster's problem.

3. After verifying the signature in M_4 the Camera forms

   $X_n = C_{B_1}, M_2, A_1, ..., A_n$, where n is the number of images taken.

   and sends to the Base

   $$M_5 = X_n, \text{Sign}_{SK_C}(M_4, X_n).$$

   This will allow this Base to verify everything that has happened.

4. The Base reviews M_5, verifies signatures, hash chains, timestamps, and other internal and external values. When satisfied that this set of images is authentic, it forms

   $$M_6 = M_2, \text{Sign}_{SK_{B_2}}(M_5).$$

M_6 is stored permanently in the base. Eventually the base could forward a copy of the M_6 to a central repository.

This leaves us with a chain of messages that trace a sequence of digitally-signed images from the time the Camera was taken out of the police station, until it was returned.

At this point the person who take the pictures is responsible for them. The point to this protocol is to help to photograph prove the legitimacy of the photographs should the need ever arise.

3.4 What Can Go Wrong

- The Camera's tamper-resistance could fail. If the outer perimeter fails, an opponent will likely be able to intercept the CSP's commands to the auto-focus and flash, and may be able to make a believable image to feed into the A/D converter. If the CSP's tamper-resistance fails, then an attacker can create and authenticate whatever fraudulent images he likes.
- The Camera could have images sent into it, and have its protective measures defeated that should have prevented this.
- The Base could be subverted, either completely or partially (such as messing up the random number source, or the internal secure clock).
- The signature algorithm could be broken, or one of the keys could somehow be compromised.
- The hashing algorithm could be broken.

4 Bounding the Image in Time

In the basic system, there is some minimal bounding of the images taken in time. For many applications, this will be enough time information. However, some applications may need more precise measurements of time on each image.

We can easily include an internal secure clock in the Camera. This clock will produce a timestamp, T_i, as needed. In this case, we simply include T_i in the internal state information, Y_i. This is recorded, signed, and hashed. Note that because this information is now used in creating the images that follow (by altering the specific auto-focus and flash settings), each image is strongly dependent on the timestamp of the previous image. This allows bounding in time both ways.

During the final Authentication phase, the Base now verifies that $T_{-1} < T_0 < T_1 < \ldots < T_{n-1} < T_n$.

Along with the items discussed above, there are other potential problems with this:

- If the attacker can alter the camera's internal timekeeping twice (once to alter it for use in a fraudulent set of images, once to fix it so that the Base doesn't notice during final Authentication), then this system can be broken.
- If the Base times are altered somehow, we have big problems. A solution to this might be to require the Base to interact with a digital timestamping service immediately after each Initialization and Authorization step.

5 Bounding the Image in Space

In the basic system, the time-bounding effectively bounds the system in space as well: given an idea of the maximum speed possible for a Policeman between Initial Authorization and Final Authentication, we are able to determine a radius of possible locations. Unfortunately, this is not very limiting. If we suspect that the image was taken a few miles away from the location claimed, this gives us little help. In this section, we develop some methods to attempt to bound the location of the Camera at the time the image was taken.

It is possible to outfit the Camera with a GPS locator. This provides location data which can be included into the internal state data, Y_i, when signing and hashing an image. Unfortunately, GPS by itself isn't secure.

It would be possible to build a variant of GPS which is secure, in the sense that precise times between challenge and response can be used to rule out attempts to carry out a "rerouting" attack, in which the GPS signal is transmitted to some other place, and that place's response is sent back. If we have authenticated location data, it can be included in the internal state data, Y_i.

The simplest way to do this is to have some local location-finder based on the Camera's radio transmissions, which finds the direction to the Camera at any given time. To make this resistant against interception and rebroadcast of the signals, we can continually issue challenge/response queries, and require response in a fixed amount of time. Other methods, such as frequency hopping, random noise broadcasts, spread spectrum techniques, and directional transmissions, may also be used.

Some work has been done work towards making GPS secure, including making distributible receivers which a person can be used to authenticate a their location.[2][3]

Along with the items discussed above, there are other potential problems with this:

- GPS isn't secure against active attacks.
- It may be very difficult to measure timing of challenges and responses as finely as would be needed to detect all rerouting attacks.

6 User Authentication

In the basic system, the Initial Authorization and Final Authentication may include some additional data about the Policeman who is checking out the camera, but no additional information is kept about who is actually using the Camera.

The simplest way to do this is to use the camera to take images of the people who need to be authenticated, at close enough range that biomectrics like fingerprints and retina prints can be read out of the picture. This takes almost no additional cost.

It is also possible to use some kind of biometric (such as a fingerprint) to operated the camera at all. The readout from the fingerprint scanner is put into Y_i. This could be used to keep the camera from being used at all unless the right fingerprint was on the camera.

Along with the items discussed above, there are other potential problems with this:

- If image replay is possible with the Camera, 6.1 doesn't work. (On the other hand, neither does the Camera.)
- Nothing ensures that the person verified as being present is actually around for any of the pictures he isn't actually in.
- Biometrics are very similar conceptually to passphrases and images: they can be intercepted and replayed. This means that there may be ways to bypass the kind of control over who is allowed to use the Camera.

7 Conclusions

A secure camera can be built using the simple cryptographic primitives of digital signatures and one-way hash functions, a cryptographic processor inside a secure perimeter, and clever protocol design. However, the most dangerous attack against this system—feeding a falsified image through the camera lens—cannot be prevented cryptographically. Like many systems, the digital portion can be secured, but the analog portion cannot.

8 Acknowledgments

The authors would like to thank Steve Bellovin, James Jorasch, David Wagner, and Jay Walker for their helpful comments and suggestions.

References

[1] Y. Desmedt, C. Goutier, and S. Bengio, "Special Uses and Abuses of the Fiat-Shamir Passport Protocol," *Advances in Cryptology—CRYPTO '87 Proceedings*, Springer-Verlag, 1988, pp. 21-39.

[2] D. Denning, P. MacDoran, "Location-Based System Delivers User Authentication Breakthrough,", *http://all.net/csi/csi-96-01.html*, Computer Security Institute, Jan 96, pp. ??.

[3] D. Denning, P. MacDoran, "Location-Based Authentication: Grounding Cyperspace for Better Security," *Computer Fraud and Security Bulletin*, Feb 96, pp. ??.

[4] S. Haber and W.S. Stornetta, "How to Time-Stamp a Digital Document," *Journal of Cryptology*, v. 3, n. 2, 1991, pp. 99-112.

[5] National Institute of Standards and Technology, NIST FIPS PUB 180, "Secure Hash Standard," U.S. Department of Commerce, May 93.

[6] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, v. 21, n. 2, Feb 1978, pp. 120-126.

[7] B. Schneier, *Applied Cryptography, 2nd Edition,* John Wiley & Sons, 1996.

[8] B. Schneier and J. Kelsey, "Automatic Event-Stream Notarization Using Digital Signatures ," *Proceedings of the Cambridge Protocols Workshop*, to be published by Springer-Verlag, 1996.