The Untrusted Computer Problem and Camera-Based Authentication

Matt Burnside, Dwaine Clarke, Blaise Gassend, Thomas Kotwal,
Marten van Dijk, Srinivas Devadas, Ronald Rivest

March 11, 2002

Abstract

The use of computers in public places is increasingly common in everyday life. In using one of these computers, a user is trusting it to correctly carry out her orders. For many transactions, particularly banking operations, blind trust in a public terminal will not satisfy most users. In this paper the aim is therefore to provide the user with authenticated communication between herself and a remote trusted computer, via the untrusted computer.

After defining the authentication problem that is to be solved, this paper reduces it to a simpler problem. Solutions to the simpler problem are explored in which the user carries a trusted device with her. Finally, a description is given of two camera-based devices that are being developed.

1 Introduction

In this paper we discuss methods for the verification of the trustworthiness of a public computer (e.g., in an Internet cafe or airport lounge). Consider Ursula, on holiday in Peru, who wishes to manage her stocks. She visits the local Internet cafe where she uses a computer to contact her bank's web-site. She uses it to review the stock market and place orders. In doing so she is exposing herself to a host of possible attacks.

Indeed, Ursula has no idea of what is going on inside the computer she is using. Even if the interface looks exactly as she expects, she could in fact be interacting with a Trojan horse. The Trojan horse has many attacks to choose from. It can store Ursula's password for later use. It can tamper with what Ursula is seeing, giving her a misleading idea of the market. It can tamper with Ursula's transaction, changing the amounts, or the stocks that are being bought or sold. A skillfully designed Trojan can completely simulate Ursula's session with her bank, while in fact doing transactions of its own choosing. It could wire Ursula's money to an account in Switzerland or change Ursula's password, thereby preventing her from contacting her bank in the future. Unless suitable measures are taken, Ursula will not even realize that she is being tricked until she checks her account through a trustworthy source. Even if Ursula knows the administrator of the Internet cafe's intentions, the administrator's technical skill may be in doubt. Indeed, a hacker might have overcome the Internet cafe's security and installed malicious software on its computers.

This leaves Ursula in a quandary: She can use the high bandwidth, ergonomic keyboard and mouse, large screen and powerful computing of the Internet cafe and run the risk of being tricked, or she can use her simple mobile device with its low bandwidth and uncomfortable interface to do all her work.

In this paper, we will assume that Ursula is using an untrusted computer to contact a trusted computer (e.g., an Internet banking server) over a network. The goal of this paper is to present methods to provide an authenticated bidirectional channel from the trusted computer to Ursula (the user), through the untrusted computer. Authenticated means that messages received through the channel are guaranteed to be unmodified copies of messages that were sent by the party on the other side of the channel.

After a brief review of related work, section 3 models the untrusted computer problem and shows a reduction to a simpler problem. Section 4 will present several implementations that could solve the simplified problem. Section 5 will focus on the camera based solutions that we have implemented. Finally, section 6 will outline a few areas for further research.

2 Related Work

The difficulties related to using an untrusted terminal are not new. ATMs are one of the most prominent examples. Since the user gives both her card and her PIN to the ATM, she is placing total trust in it. Fake ATMs have been known to simulate a breakdown, simply keeping the user's card after the PIN was entered. Crooks can then use the stolen card and PIN until the user realizes that she has been tricked. To address this attack, many attempts [Mat96, DP00, HB, Mat91] have been made to replace the PIN with a challenge-response mechanism, in which the user knows a secret that allows her to answer the challenge that she is given, by solving a little problem in her head. These systems allow the user to identify herself securely, but they fail to authenticate the rest of the session. Our system guarantees the authenticity of the whole session.

In [ABKL91] a smart-card based system is described that allows secure identification of the user, and that allows her to limit the power that is delegated to the untrusted host, as well as the duration of the delegation. We authenticate each piece of information that is transmitted, without placing any trust at all on the untrusted terminal.

The nearest system to ours that we have found is described in [NP97]. The protocol that they present is similar to ours, but they implement it using simple transparencies, which makes it harder to use than our camera-based authentication system.

3 Model

We now formalize the layout of the untrusted computer problem and illustrate it in figure 1.

- There is a user U. Her abilities are limited to those of a typical person.
- U might be in possession of a personal device D. Its abilities are those of a computer. We will try to restrict them as much as possible to minimize the size, cost, and power requirements of D.
- The combination of D and U will be referred to as DU.
- U will try to communicate with her proxy P, a computer that she trusts[1]. P's abilities are those of a computer, but we will not try to restrict them as strongly as those of D.
- DU and P are connected by a channel C, that embodies an untrusted computer that DU has physical access to, connected to P via an untrusted network. Most of the time C will simply convey messages between DU and P in which case we will say that it is faithful. However, in some cases C can exhibit any behavior that is possible for an arbitrary combination of humans and computers (we will assume that C cannot break cryptographic primitives).
- DU and P can send and receive messages over C, and they can sometimes decide to accept messages that they receive. Messages from DU to P will be called upwards messages, while messages from P to DU will be called downwards messages (as in uploading and downloading).
- In general, DU and P will have a shared secret to perform cryptographic operations. If the cryptographic secret is held by D then it becomes possible for an attacker to steal D and pretend to be U. In the rest of this paper, we will assume that some form of direct identification of U to D takes place, with a PIN number, or a biometric measurement.

Figure 1 illustrates the layout.

Definition 1 *We define a* Unidirectional Authentication with Secure Approval Channel *(UASAC) to be a channel that provides the following primitive operations :*

Downwards Authentication *: P can send a message to U in such a way that U always accepts the message if C is faithful. U will never accept a message that was not sent by P, or that was tampered with.*

Secure Approval *: If C is faithful then U can always inform P that it approves a specific authenticated message from P. P will never consider that a message is approved if U did not actually approve it.*

Upwards Transmission *: U can send a message to P that will be received unmodified if C is faithful.*

[1]More information about proxy-based protocols can be found in [BCM+02].

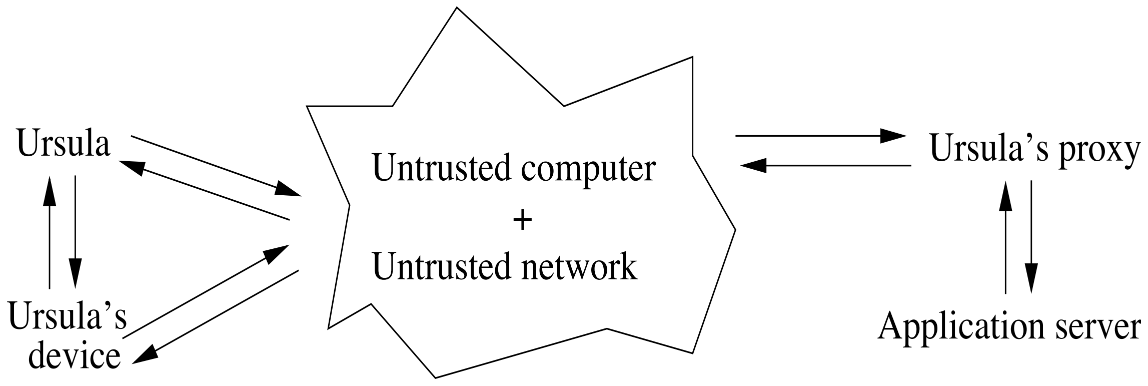

Figure 1: The untrusted computer model : Ursula, equipped with her device, wishes to establish authenticated communication with her proxy over an untrusted channel. Once this is done, she can safely use any application on her proxy, or on some remote application server (such as her bank)

Definition 2 *We define a* Bidirectional Authentication Channel *(BAC) to be a channel that provides the following primitive operations :*

Downwards Authentication *: (same as above)*

Upwards Authentication *: U can send a message to P in such a way that P always accepts the message if C is faithful. P will never accept a message that was not sent by U, or that was tampered with.*

It is noteworthy, that there is no need to send user identifiers or passwords over a BAC. When P receives a message message from DU over a BAC, it knows that it is talking to DU because of the specification of the BAC. All the work needed to identify the parties that are communicating has been done by the algorithm that provides a BAC from an untrusted channel. Adding a password would be redundant. Moreover, since there is no mention of privacy in a BAC, any password sent over C would be compromised.

Theorem 1 *Any UASAC can be used to make a BAC.*

We prove this by showing an algorithm that produces upwards authentication.

1. *U sends a message M to P.*
2. *P receives M' from U, M' might be different from M if C was unfaithful.*
3. *P does an authenticated transmission of M' to U.*
4. *U approves the message M' it received from P if it is identical to M and if U did send M to P.*

U only accepts M' in step 4 if it sent a message M that was identical to M'. Therefore, when P receives approval on M', it can accept it according to the requirements of upwards authentication.

It is of note that a similar result could be obtained by reversing U and P in the definition of the UASAC. However, this seems to lead to more complex and less interesting implementations that have not been studied.

4 The Device

We will now look at implementations of a UASAC. In the definition we have made of a UASAC, no mention was made of D. Though it might be possible to avoid the use of any device at all, we do not know of any convenient way for U alone to authenticate a message that she is receiving.

In the case where D is present, there are two basic designs : U can receive incoming messages through D, or directly from C. We will not consider methods in which U has to compare a non-secure copy of the message obtained directly from C with a secure copy displayed by D, as the non-secure copy is then redundant.

4.1 Relaying Device

In the case where U is receiving incoming messages through the device, there is a simple protocol to obtain a UASAC: Messages are sent from P in encrypted form, along with an encrypted one time password. They are protected by a nonce[2] and a MAC[3]. D receives the message from C, checks its authenticity, and passes it on to U if the checks succeed. This gives us downward authentication. Secure approval is obtained by sending the one time password back to P if U wishes to approve the message. C cannot fake approval because until U decides to approve the message, the one time password is never sent over C in the clear.

In a very straightforward implementation of the relaying device, Ursula comes to the Internet cafe with her PDA. She connects it to the untrusted computer's USB port, and connects to her proxy using her PDA's SSL-capable web browser. She can then do all her interaction with her proxy through her trusted PDA.

In a more complicated implementation Ursula would still have to use her device's screen but she could type her replies on the Internet cafe computer's keyboard, since the up-link of a UASAC need not be secure.

However, all of these solutions are dissatisfying, because Ursula is barely making any use of the untrusted computer's comfortable screen. The Internet cafe's computer is just being used as a network access point. Meanwhile, Ursula has to study the stock market through the tiny screen of her hand-held device.

An original implementation of the relaying device approach is presented in [NP97] where transparencies are used to obtain a secure channel. Here, Ursula has a secret transparency with a random distribution of black and transparent pixels. If Ursula's proxy wants to communicate a message it sends a random-looking black and white pixel pattern, such that the message emerges if Ursula looks at the untrusted computer's screen through her transparency. This is an elegant low-tech solution to our problem. However, it is impractical for a number of technical reasons[4].

4.2 Monitoring Device

With monitoring devices, U gets information directly from C. Some extra information is also sent through C, that U need not concern herself with, but that D uses to verify the authenticity of the information U is getting. If D detects tampering, or if for some technical reason D is unable to authenticate the image (too much noise, the connection from C to D is bad, etc.), it warns U. Monitoring approaches are more convenient as U fully uses the untrusted computer's interface. However, a number of difficulties arise.

First, it is important to realize that D must be authenticating the information that U is getting. It would not be acceptable for D to receive information about what U is seeing on the screen through a USB link, as a malicious computer could send one thing through the USB port, and something completely different to the screen. This means that to authenticate screen content, D must be equipped with a camera.

Even if D and U are both getting their data from the same source (we will consider a screen), caution is still required because of noise. Indeed, it is impossible for D to reconstruct what is being displayed on the screen down to the exact RGB components of each pixel. Variations in screen brightness, camera noise and reflections off the screen all contribute to imprecision in what D can reconstruct. If a rogue message is displayed in grey on a slightly lighter grey, U will be able to read it, while D might see it as uniform grey. It is because of this difficulty that our implementations use black and white (no grey) images.

Thus we see that if D does not perceive as much information as U, then U must be aware of that limitation, and be able to tell when an image might be ambiguous to D. For example, if D can only distinguish between black and white, then a shade of grey should reveal to U that tampering has taken place, even if D does not detect that tampering.

[2] A nonce is a sequence number that helps prevent replay attacks. See [KPS95] for details.

[3] A Message Authentication Code is a cryptographic hash of a message concatenated with a key. It can only be generated and checked by someone bearing the key. See [KBC97] for details.

[4] The main problem is that transparencies cannot be reused, so in order to achieve security, Ursula needs a whole sequence of one-time secret transparencies during each electronic transaction. In [NP97] a method is proposed to use a transparency many times, but it assumes that transparencies can be easily placed at a precise position on the screen that is unknown to the untrusted computer. Our experiments with transparencies on a screen suggest that the on-screen image must be finely scaled and moved before it can be made to line up with the transparency precisely enough; this gives away the transparency's position.

The next section will give details of two camera-based monitoring device implementations.

5 Camera based solution

In this section, we present two implementations of camera-based authentication that are under development. In both cases the user is expected to carry a camera-equipped device that monitors the screen of the untrusted computer she is using. The visual processing involved in extracting on-screen information can be costly in computation resources. The first method we propose tries to minimize this cost, while the second one uses a high bandwidth network connection to move the computation to the proxy.

5.1 Pixel Mapping

In the pixel mapping method, the camera-equipped device is assumed not to move relative to the screen during the authenticated session. An initial calibration phase is used to construct a mapping between screen pixels and camera pixels. The mapping is then used by the device to exactly reconstruct the screen content. A small area at the bottom of the screen is used to transmit a nonce, a one-time password and a MAC.

5.1.1 Protocol

- Downwards Authentication :

  - The Proxy sends (information, encrypted nonce, encrypted one-time password, MAC(information, encrypted nonce, encrypted one-time password)) over the channel. To the user, this appears as an image, with a strip of random-looking data at the bottom.

  - The device exactly reconstructs the screen content from what it sees through its camera.

  - The device checks the nonce, calculates the expected MAC and compares it with the on-screen MAC. If all checks succeed it lights a green light, and displays the decrypted one-time password on a small LCD display.

  - The user reads the screen content.

- Upwards Transmission :

  - The user types commands on the untrusted computer's keyboard. They are sent directly to the proxy.

- Secure Approval :

  - The user reads the one-time password from the device's LCD screen and uses upwards transmission to send it to the proxy.

- Calibration :

  - The untrusted computer displays a predefined sequence of images on its display. The basic property of these images is that each screen pixel flashes its coordinates in binary (with a suitable amount of redundancy to improve robustness).

  - The device records the number that was recorded at each camera-pixel. That number allows the device to know which screen-pixel is seen by each camera-pixel. Camera pixels with invalid numbers are assumed to see multiple screen pixels or to be off-screen; they are excluded from further use. If each screen-pixel was seen by at least one of the valid camera-pixels, a mapping between camera-pixels and screen-pixels can be established. It is later used to reconstruct the on-screen image. The device should check that the relative positions of screen pixels relative to camera pixels are reasonable, in order to detect if the untrusted computer attempts to tamper with the calibration process. Without this check the untrusted computer could perform an arbitrary permutation of the screen pixels.

5.1.2 Evaluation

A preliminary implementation of this protocol has been written. It works with black and white images, and currently requires camera resolution to be about twenty times greater than screen resolution. Current results suggest that this ratio can be reduced by a factor of 5 only by improving the calibration phase. Extensions to limited numbers of colors are also possible. At present the calibration phase displays 27 frames to calibrate a 80 by 50 pixel screen. This takes about 10 seconds. Rethinking the calibration method to take into account simple geometrical constraints should be able to reduce this to under 10 frames for any size of screen.

This approach uses simple algorithms that would be easy to implement in hardware, and does not require large amounts of computation power. Because of its calibration method, no knowledge is needed of the exact screen and camera geometry (curved screens, distorting cameras, etc.). However the geometry is expected not to change with time, which means that the device must be set on a stable surface, instead of being warn by the user (as a badge, for example).

5.2 Optical Character Recognition

In the optical character recognition (OCR) method it is assumed that the user's device is equipped with a camera and an infrared link to the untrusted computer. This link is used to exchange data with the proxy, via the untrusted computer, to take advantage of the large amount of computation available at the proxy.

5.2.1 Protocol

- Downwards Authentication :

  - The Proxy sends information, in the form of an image containing text, to the untrusted computer. This image is displayed on the screen. Note that in order to save transmission time the proxy can send formatted text instead of an image. The semantics of the formatting must be exactly specified.

  - The device takes a picture of the screen, and sends ("verify", picture, encrypted nonce, MAC("verify", picture, encrypted nonce)) to the untrusted computer using the infrared link; the message is then forwarded to proxy.

  - The proxy checks the nonce, calculates the expected MAC and compares it with the received MAC. If all checks pass it verifies that the text displayed on the screen was genuine by performing OCR on the received picture.

  - The result of the OCR is compared with the information from which the image was formed in the first step. If this check passes, the proxy sends (yes, encrypted nonce, MAC(yes, encrypted nonce)) to the device. The device checks the nonce and MAC. If all checks pass the device lights a green light.

- Upwards Transmission :

  - The user types commands on the untrusted computer's keyboard. They are sent directly to the proxy.

- Secure Approval :

  - The user accepts the data on the screen using her device. The device takes a picture of the screen, and sends ("accept", picture, encrypted nonce, MAC("accept", picture, encrypted nonce)) to the proxy. The proxy verifies the message as in downwards authentication. If the tests pass then it considers the picture approved, as only the user's device could have produced the MAC. Note that a race condition exists in which the untrusted computer changes the content of the screen after the user has pressed the accept button on her device, but before the camera has actually taken the picture. To avoid this condition, a proper implementation should consist of two separate buttons: one for capturing the image and one for sending the image to the proxy.

- Image Verification :

  An integral step in the protocol is the proxy's verification of the picture taken of the screen. There are two basic steps in this process. The first is to correct the distortions of the original image that are caused by the camera angle, curvature of the screen, lens deformation, and low camera resolution. An easily identifiable border is placed around the text to facilitate the correction of these distortions.

  The second step in the image verification process is to perform OCR on the processed image. This application differs from most OCR applications because in this case the intended message is known, which allows for a large speed optimization. The OCR algorithm must detect any change in the image that would result in the user seeing a different character from the one that was originally sent by the proxy.

5.2.2 Evaluation

This method's advantages over the pixel-mapping method are that it does not require any calibration, and the camera does not have to be immobile during the session. However it will take longer to verify a given screen, so authentication of every screen during a session would be cumbersome. Instead, only screens containing vital information will be verified.

An implementation of this method is underway. It is expected that the largest text block that can be verified will consist of approximately 100 characters.

6 Possible extensions

In this paper, we have been most concerned with authenticating communication in which the user is receiving visual information. Our protocol could be applied just as well to audio information. Though this is probably not very useful to the average user, it would certainly benefit the visually impaired.

The camera-based system does not provide any privacy, as queries and responses are transmitted in the clear. A limited amount of privacy could be added by allowing the user to point at areas of the screen. Selections made in this way would be visible to the device but not to the UC. The possibilities of such a system are yet to be explored.

For now, all our attention has been restricted to authenticating black and white images (no shades of grey). This is because of the noise-related security concerns that we discussed in section 4.2. Extending our system to a small set of easily distinguishable colors would be easy, as long as the user can be expected to notice if invalid colors are used. This ability of the user is a direction for further study. Ideally, though, it would be nice to extend the system to arbitrary colors, but then we must be sure that the device will perceive the same color areas as the user, which implies a good understanding of human visual perceptions.

7 Concluding remarks

In summary, we have studied how a person using an untrusted terminal can communicate with a trusted computer in an authenticated way with the aid of a trusted device. To do this, we have presented a protocol that allowed us to simplify the initial problem. We then explored ways to implement the reduced problem, considering how convenient each one would be for the user. Finally, we presented our implementations that are based on a camera-equipped device.

References

[ABKL91] Martin Abadi, Michael Burrows, C. Kaufman, and Butler W. Lampson. Authentication and delegation with smart-cards. In *Theoretical Aspects of Computer Software*, pages 326–345, 1991.

[BCM$^+$02] M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas, and R. Rivest. Proxy-based security protocols in networked mobile devices. In *Proceedings SAC*, 2002.

[DP00] Rachna Dhamija and Adrian Perrig. Dejà vu: A user study using images for authentication. In *Proceedings of the 9th USENIX Security Symposium*, 2000.

[HB] Nicholas J. Hopper and Manuel Blum. A secure human-computer authentication scheme.

[KBC97] H. Krawczyk, M. Bellare, and R. Canetti. RFC 2104: HMAC: Keyed-hashing for message authentication, February 1997. Status: INFORMATIONAL.

[KPS95] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network Security, Private Communication in a Public World.* Prentice Hall PTR, 1995.

[Mat91] Tsutomu Matsumoto. Human identification through insecure channel. In *Theory and Application of Cryptographic Techniques*, pages 409–421, 1991.

[Mat96] Tsutomu Matsumoto. Human-computer cryptography: An attempt. In *ACM Conference on Computer and Communications Security*, pages 68–75, 1996.

[NP97] Moni Naor and Benny Pinkas. Visual authentication and identification. In *CRYPTO*, pages 322–336, 1997.