A User-Friendly Approach to Human Authentication of Messages

Jeff King and Andre dos Santos

College of Computing, Georgia Institute of Technology[⋆],
Atlanta GA 30332, USA
{peff, andre}@cc.gatech.edu

Abstract. Users are often forced to trust potentially malicious terminals when trying to interact with a remote secure system. This paper presents an approach for ensuring the integrity and authenticity of messages sent through an untrusted terminal by a user to a remote trusted computing base and vice versa. The approach is both secure and easy to use. It leverages the difficulty computers have in addressing some artificial intelligence problems and therefore requires no complex computation on the part of the user. This paper describes the general form of the approach, analyzes its security and user-friendliness, and describes an example implementation based on rendering a 3-D scene.

Keywords: Authentication, Human Cryptography.

1 Introduction

Security protocols often require their participants to perform complex computations. While computers are built for such tasks, human users must trust computers to faithfully perform the operations on their behalf. Unfortunately, this trust is often misplaced or misunderstood. The ubiquity of public computers is convenient, but those computers act not in the interests of their users, but of programs running inside them. Even if the owner of a computer is trusted, there is no guarantee that the system is not actually under the control of a virus, worm, or other attacker.

Consider a user who wishes to purchase something from a store. In most modern stores, the user swipes a credit card through a machine provided by the store. The machine shows the amount of the transaction on a visual display, and the user confirms the transaction by pressing a button on the machine. There are two threats to the user in this scenario. First, the store's machine may have misrepresented the amount of the transaction, showing one value but charging another to the account. Second, the user intends for only one transaction to occur. However, the machine has been provided with the credit card number, which is the only information necessary to make an unbounded number of transactions.

[⋆] This work was supported by a grant from Microsoft Corporation.

A.S. Patrick and M. Yung (Eds.): FC 2005, LNCS 3570, pp. 225–239, 2005.

To combat the latter threat, many systems have proposed the use of a trusted computing platform which performs sensitive operations in a secure way. For example, if the user in the previous example had been carrying a smart card, it would have been possible for the trusted smart card to produce a digital signature for a single transaction of a certain amount without revealing any additional information to the store's machine. Such a trusted platform must always be available to the user (either as a mobile device, or on a network) and must be resistant to external attacks.

Because of security, cost, and physical constraints, trusted platforms are often unable to interact directly with the user. For example, most smart cards rely on an expensive, non-portable reading device to display output and receive input from the user. In a point-of-sale setting such as the one described above, the reading device would likely be provided by the store. In the case of a trusted platform on a computer network, the user is clearly dependent on a local computing system to transmit and receive messages across the network.

The result of this lack of interactivity is that systems utilizing a trusted platform may be vulnerable to attack by an untrusted system which sits between the user and the trusted platform. Consider again the example of the shopping user, with one change: he now connects his smart card to the store's machine. The store's machine sends the amount of the transaction to the smart card. The card shows the amount to the user by asking the store's machine to display it. The user confirms the transaction by pressing a button on the store's machine, and the card finishes the transaction. Because all interaction has gone through the store's machine, which has the ability to alter messages in transit, the user cannot be sure that the amount displayed was the same as the amount sent to the smart card.

One way to solve this problem is to devise a protocol which allows a user to exchange messages with a trusted platform in a way that ensures the integrity and authenticity of the exchanged data. Furthermore, it must be possible for the actions required for one of the protocol's participants to be performed by a human without the aid of a computing device.

This paper introduces a novel approach for exchanging messages between a trusted computing platform and a human over an untrusted medium that is both secure and easy to use. The security of the approach is based on the difficulty that computers have in solving certain hard artificial intelligence (AI) problems. The user-friendliness is based on the natural ability of humans to solve the same problems. Section 2 introduces the concept of a keyed AI transformation, a general primitive for providing integrity and authenticity checks for messages with a human recipient. It also describes and analyzes a keyed AI transformation based on the problem of recognizing three dimensional (3-D) objects in a rendered scene. Section 3 describes a protocol for sending secure messages from a human to the trusted platform based on the keyed AI transformation primitive. Section 4 discusses the performance characteristics of the 3-D example, while sections 5 and 6 discuss related and future work.

2 Keyed AI Transformation

The notion of using hard artificial intelligence problems as security primitives has recently received a great deal of attention[1]. In short, a hard AI problem is a problem that can be solved by a majority of humans but not by a computer using state of the art AI programs (in the consensus of the AI research community). Security protocols can leverage the hardness of these problems just as cryptography has made use of hard problems in number theory, such as factoring.

A CAPTCHA is an automated test for telling humans and computers apart[1]. Many CAPTCHAs are based on a simple formula: randomly select a message, transform it in such a way that only a human can understand it, and ask the user to decipher all or part of the original message. If the response matches the original message, then the user is deemed human. For example, Yahoo wants to prevent automated programs from signing up for free email accounts. When creating an account, a user is presented with an image containing distorted text. To pass the test, the user must type one or more words that are represented in the image.

A trusted platform that wishes to protect a message being sent to a human receiver through a hostile medium can use a CAPTCHA transformation to provide confidentiality and integrity against attacking computers. Instead of randomly generating a message as above, the computer transforms the specific message that it wishes to send and transmits the result through the hostile medium. An attacking computer is unable to extract the original message from the transmitted message due to the properties of the transformation. Since any human can extract the message, the confidentiality provided is very weak. However, if a human attacker is not available to see the message in real-time, then the message can be considered temporarily confidential. This primitive is discussed further in section 3. An attacking computer will also have difficulty making a meaningful but undetectable modification to the message. Without the ability to determine which parts of the message are meaningful and which parts are obfuscation, modifying an existing image is difficult.

An attacking computer can, however, simply replace the entire message. Because all parameters of the transformation procedure are publicly known, there is nothing to uniquely identify the sending computer. A given message could have been created by any computer, not only by the trusted platform. To solve this problem, we introduce the notion of a keyed AI transformation, which is a function taking as input a message m and a secret key k, and returning a new message t. Both the sender and the receiver of the message have knowledge of the key, but nobody else does. The sender's transformation incorporates the key, and the receiver verifies that the resulting message was made by a program with knowledge of the key. Because the receivers are humans, it must be easy for them to remember the key and to verify its usage in the message.

We define the security and usability properties of a transformation with the following parameters: A transformation $t = T(m, k)$ is an $(\alpha, \beta, \gamma, \delta, \epsilon, \tau)$-keyed transformation if and only if:

- the probability that a human can extract m from t is at least α
- the probability that a human with knowledge of k can correctly verify whether k was used to create t is at least β
- there does not exist a computer program that runs in time τ such that the probability of the program extracting m from t is greater than γ
- there does not exist a computer program that runs in time τ such that the probability of the program extracting k from t is greater than δ
- let A be a computer program that modifies t such that a human will extract m' from t (with $m' \neq m$); there does not exist an A that runs in time τ such that the probability of a human failing to detect the modification is greater than ϵ

To be useful, a keyed transformation must be both user-friendly and secure. The parameters α and β represent the ability of humans to use the system and would ideally be maximized. The values of α and β for a specific transformation can be determined empirically.

The security of the system depends on the values of γ, δ, and ϵ. For maximum security, a transformation would ideally minimize all three values. A high value of γ indicates that the message data can be intercepted by a computer, destroying even the temporary confidentiality provided by the system. Although this does not directly affect the integrity or authenticity of a message, there may be some security impacts. These are discussed in section 3. A high value of δ indicates that a computer attacker will have good success at learning the key. Since the key is the only feature distinguishing forged messages from authentic ones, if it is compromised, then arbitrary messages can be forged. Finally, ϵ represents the integrity of the transformation; if it is high, then a legitimate encoding can be undetectably changed to carry a different message. The values of γ, δ, and ϵ can be determined by using computer programs to perform the stated tasks. The programs should represent state of the art techniques in the area of the transformation.

2.1 3-D Keyed Transformation

One example of a keyed transformation is based on the rendering of a three dimensional scene. The message is a 3-D object model to be sent from a trusted computer to a human, and the key is a set of 3-D object models. The message to be sent may be a model of a physical object, or of 3-D extruded text. The user and the trusted computer both know the key. The trusted computer knows the model data of the objects, while the user simply knows the appearance of the objects. The security of the transformation stems from the hardness of extracting components from the rendered scene, and from the difficulty in seamlessly modifying the image.

To encode a message, the trusted computer first creates an empty three dimensional scene. It then inserts a set of randomly colored planes to act as a background. The message model is placed in the scene at a random location. The coloration, rotation, and size of the model are determined randomly. Similarly, one or more instances of each key object are placed in the scene, each

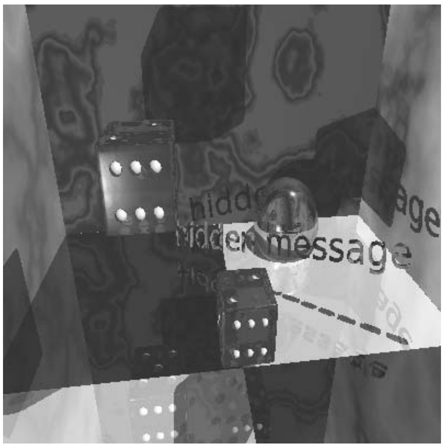

Fig. 1. 3-D Transformation with message "hidden message" and key "dice"

with a randomly selected rotation and size. Next, several objects with reflective surfaces are placed randomly throughout the scene. A camera location and light source are selected randomly, and the result is raytraced to create a two dimensional image, which is the final transformed output. An example is shown in figure 1. A human receiver of such an image would look for the presence of his key ("dice") and read the message contents ("hidden message").

Note that the random selection of the scene parameters presents a tradeoff between security and usability. Clearly some combinations of values will result in images in which the message or key objects are difficult for the user to see, or which are easy for a computer program to manipulate. For example, the message model must not be occluded by the key objects and vice versa. The colors and brightness used must provide sufficient contrast to make the models visible to a human viewer.

Unfortunately, there is no simple way to determine the best method for selecting these parameters. An implementation must use manually-coded heuristics to constrain the parameters; these heuristics are based on empirical measurements of human reaction to the scenes. The ranges for our proof-of-concept system were reached through a trial-and-error development process in which the parameter constraints were manually changed, a large number of scenes were randomly rendered, and the resulting images were evaluated for their usability and security properties. This process has led to an implementation that can input arbitrary text (within a certain length constraint) and key object models to produce scenes which have a high probability of being readable by humans, but from which current computer programs will have difficulty extracting the text and keys.

2.2 Attacks on 3-D Transformation

The security of the transformation is based on an assumption of the hardness of certain tasks. In this section, we analyze some possible attacks and show that their probabilities of success are bounded by the parameters of the transformation, including γ, δ, and ϵ.

Suppose that Carol is a computer that wishes to send a short text message m to a human, Harry. Harry is sitting at a workstation named Wanda, which is able to connect to Carol across the Internet. Harry doesn't trust Wanda, but he wants to be sure that the message he receives was sent by Carol and that it was not modified in transit. Carol and Harry have previously established the secret key $(apple, orange)$ and Carol is storing a 3-D model of an apple and one of an orange. She renders m along with the apple and orange models; the result is t, which she sends to Harry. Wanda is capable of discarding or modifying messages in transit, as well as creating new messages and displaying them to Harry. Wanda's goal is to convince Harry that the message is actually m'.

Key Guessing. Wanda can simply discard the message sent by Carol and instead send her own message. Suppose that Wanda has stored beforehand a set of object models. Harry will be looking for an image containing a message model (either text or a meaningful object) and all of his key objects (apples and oranges). Harry will tolerate the presence of other objects in the scene, since they may have been included to confuse an attacker. Wanda randomly selects n objects from her set, renders them in a scene along with m', and sends the result to Harry. If Wanda's image includes both apples and oranges (or objects that look similar enough to fool Harry), then Harry will accept the forged message as authentic.

Assume that Harry has selected k object models from a finite list of N models. Wanda includes n models in her attempted forgery. If her set of n models is a superset of Harry's set of k objects, her forgery is successful. Assume that $n < N$ and $k < N$ (neither Wanda nor Harry can choose more objects than exist in the list). The probability P of Wanda's success is

$$\begin{array}{l} \text{if } n < k \text{ then } P = 0 \\ \text{if } n \geq k \text{ then } P = \prod_{i=1}^{k} \frac{n-i+1}{N-i+1} \end{array}$$

It is therefore in Wanda's best interest to make n as large as possible. However, n is bounded by the amount of space in the image. In practice, she probably can't fit more than five or six objects into the scene before it gets too cluttered for Harry to identify anything. In order to lower Wanda's probability of success, Harry can increase either k or N. Increasing k has the side effect that Harry must remember and verify a larger number of objects; it is probably inconvenient for Harry to have more than three key objects. Increasing N to be several orders of magnitude larger than n gives Wanda a very low probability of successfully guessing the key objects. Ideally, N would be unbounded or impossible for Wanda to enumerate. If each individual user can have an arbitrary model, then Wanda will not be able to store a pre-determined set of possible objects.

In the previous analysis, it was assumed that Harry picked a model from the N choices using a uniform distribution. In reality, Harry's selection will be influenced by the appearance of the models and his particular interests. For example, a model of a very cute puppy will likely be chosen more frequently than one of asparagus. Wanda can weight her selections to achieve a higher success rate. This is similar to using a dictionary attack against easy-to-guess text passwords. People are more likely to choose a password based on their login name, pet's name, or words from a dictionary than they are to choose a random string. Such easy to guess passwords should be avoided. One solution is to randomly generate a password. With text passwords, this frequently leads to passwords that are difficult to remember. However, randomly assigning an object model is much more user-friendly; the user only needs to remember the appearance of a few objects.

Convert 2-D to 3-D. Wanda can intercept t (Carol's 2-D image) and attempt to "reverse render" it back to a 3-D scene description. Once she has the original scene description, she can replace m with m', re-render the image, and send it to Harry. The probability of Wanda successfully performing the reverse render has an upper bound of $\min(\gamma, \delta)$. This follows intuitively from the fact that recreating the original scene would allow extracting both the message and the key from Carol's image. The probabilities of performing those tasks are bounded by γ and δ respectively. The values of γ and δ are discussed in the next two attacks.

Extract Key Objects. Wanda can try to extract the key objects, render a new scene with the extracted objects and m', and send the result to Harry. Her probability of success in extracting the keys is bounded by δ. It is also likely that key objects will be partially occluded. Wanda will have to guess at the missing portion or attempt to cover it up with text or other objects in her new scene.

Moreover, even if she is able to locate the objects within the image, Wanda will have only a single perspective on a three dimensional object. When placing the object in her newly rendered scene, she may attempt either to render the object from the same perspective or to extrapolate other perspectives on the object. Rendering from the same perspective requires that the camera angle, camera direction, and lighting be identical; it is difficult to calculate these parameters from only the 2-D rendered image. If Wanda chooses instead to extrapolate other perspectives on the object, she will have to guess at the appearance of the hidden sides of the object. Her guesses must be accurate enough to fool Harry, which should be difficult for non-trivial objects.

Modify 2-D Image. Rather than attempting to extract the 3-D information from the image, Wanda can simply attempt to insert, delete, or modify the text of the 2-D image. The probability of her changes not being detected by Harry has an upper bound of ϵ.

The parameter ϵ is kept low by the "reality" of the 3-D scene. Harry has a probability of detecting odd reflections, out-of-place colors, or letters with unmatched angles. His ability to do so is expressed by ϵ. For example, consider the case of deleting text. Wanda must not only identify the text and locate its exact boundaries, but she must extrapolate the objects and background that were not visible behind the text. Furthermore, deleting one image of the text isn't enough. The same text may appear as a reflection in other places, using different colors or shapes.

Insertion of text is somewhat easier, since it is covering up sections of the image rather than revealing them. However, to make a believable result, Wanda will have to account for added shadow and reflection in the scene. Without knowing the coordinates of the original light source or camera, accurately calculating those features is very difficult.

2.3 Human Adversary

The security parameters of the keyed transformation are based on foiling a non-human attacker. In some situations, this may be a reasonable assumption. An untrusted environment may be physically secured from other humans and blocked from accessing a network. However, in many cases human cooperation with a malicious computer is a possibility. It is assumed that a human attacker can easily extract the message text and key from a transformation. We will call such an attacker Robert, since he can "Read" the data in the transformation.

Consider the 3-D keyed transformation and the example given. Once a message is transmitted, Wanda can send it to Robert. Robert extracts the key and sends it back to Wanda, who renders a new scene with an alternate message and sends it to Harry. One problem for Wanda is that Robert may have trouble communicating the key objects to her. Robert may know the objects are apples and oranges, but he might not have a 3-D model of either fruit on hand. If there is a finite list of models, he can probably choose the correct one. However, if the model is not taken from a public list, he will have to find or construct models of apples and oranges. This significantly increases Robert's work, and makes it difficult for him to cooperate in real-time.

It is, in fact, feasible for a keyed transformation to have a key whose features are practically impossible for humans to articulate. An important point is that, due to the AI domain, Harry doesn't need to be able to describe the key. He only needs to be able to recognize the key's presence in a transformation. When Harry and Carol initially agree on a key, Carol can randomly generate a key and show it to Harry. Harry learns the key well enough during this procedure to verify messages later.

2.4 Other Keyed Transformations

Keyed transformations are intended as a general security primitive. The example given is meant to be illustrative, and is by no means the only keyed transformation available. Other transformations can be created using different hard AI problems.

For example, there has been some exploration of speech as a security primitive [2]. It may be possible to construct a keyed transformation by synthesizing text to speech using a particular voice (and obscuring the voice with audible noise to make automated analysis difficult). The shared key would be the parameters used to generate the voice. Instead of memorizing the vocal parameters, the human verifier would simply recognize the voice. This system is very user-friendly; recognizing voices is already a well-established security mechanism used between humans. Furthermore, the speech transformation would be more resistant to human attempts at disclosing the key or modifying the audio stream, potentially making it a better choice than the 3-D transformation.

3 Protocol Description

The previous section has shown that a keyed transformation can provide for the authenticity of messages sent from a trusted computer to a human. However, it may be the case that other operations are of equal interest. For example, a human may want to send a message to a computer, ensuring that it arrived intact, or a computer may want to send a message to a human and ensure that the human received it. This section describes a protocol based on keyed transformations for sending reliable, authenticated messages between a human and a computer through an untrusted intermediary.

The participants in the protocol are again Harry (the human) and Carol (the computer). Wanda (Harry's workstation) is able to modify or delete messages between Harry and Carol, and may introduce arbitrary messages. Harry and Carol have previously agreed upon a shared key k and a keyed transformation T.

Imagine Harry wants to send a message m to Carol. He wants to know whether she received the message without modification, and he wants her to know whether the message was genuine. The protocol works as follows:

1. Harry transmits the message m to Carol without any security features
2. Carol computes $t = T(m, k)$ and transmits it to Harry
3. Harry verifies the authenticity of t based on the shared key k
4. Harry extracts the message text from t and confirms that it is equivalent to the original m
5. Harry transmits a single secure bit to Carol indicating whether the message was correctly received

Similarly, imagine that Carol has a message m that she wants to send to Harry. She uses the following protocol:

1. Carol computes $t = T(m, k)$ and transmits it to Harry
2. Harry verifies the authenticity of t based on the shared key k
3. Harry extracts the message from t
4. Harry transmits a single secure bit to Carol indicating that he has received the message

Note that Wanda can send an arbitrary message m to Carol, who will return $T(m, k)$. If Harry and Carol are using the same key to send messages in both directions, then the resulting transformation can be used by Wanda to spoof arbitrary messages from Carol to Harry. It is therefore critical in such a bidirectional system that Carol and Harry agree upon separate keys for original messages and confirmation messages.

The protocol is built on the concept that given trusted data flow in one direction (provided by the keyed transformation) and a single bit of trusted flow in the other direction, arbitrary trusted input can be constructed[3]. It is not immediately obvious how the final step in each procedure, the sending of the single secure bit, is performed. The decision will vary from system to system, depending on the capabilities of Harry and Carol, the desired level of security, and the desired ease of use of the system. This section describes two methods with very different characteristics and analyzes their properties.

Physical Interaction. If Carol is a mobile device with which Harry can physically interact, then he may be able to send a signal to her directly. Such mobile devices may not have buttons or other direct input features. In this case, some procedure must be devised, such as interrupting the interaction between the device and the host system. For example, if Carol is a USB device, Harry can unplug her. If she is a smart card, he can remove her from the reader.

If disconnecting Carol sends a "0" bit (indicating that the message was tampered with), then what sends a "1" bit (confirmation that the message was valid)? Since there are no other signals, Carol can assume that the failure to be disconnected within a certain time period constitutes a confirmation. Therefore Carol waits N seconds after sending a response before believing the authenticity of a received message. If she is disconnected before her timer has elapsed, then the message is discarded.

The intuitiveness of this scheme is appealing. If Harry detects that Wanda is cheating, he simply disconnects his device and walks away. There is an obvious problem with this scheme, however: the confirmation is implicit. That is, without any interaction from Harry, Carol assumes that a message is valid. To exploit this, Wanda could wait for a time when Harry is not interacting with Carol. She sends Carol a message, who responds with a transformation. Wanda discards the transformation. Harry, unaware that any of this is happening, does not disconnect Carol. After Carol's timer expires, she accepts the message as valid. One easy solution to this problem is to keep Carol disconnected when she is not in use. The practicality of connecting Carol only when needed depends on Harry's situation. If his only operation with the malicious terminal is performing a single interaction (such as making a financial transaction in the checkout line of a store), then he can simply connect his device when he's ready to use it. On the other hand, if Harry is using a workstation for several hours, he doesn't want to repeatedly connect and disconnect his device every time it is used.

Another way to combat this problem is to remove the implicit confirmation by reversing the meaning of the messages. That is, Carol requires an explicit disconnection within a certain time period to confirm the message. Because some tamper resistant mobile devices are powered by the host computer, Harry may have to reconnect Carol for her to do useful work on the message. There are two problems with this. One is that Carol may not be able to keep sufficient state across a disconnect; once reconnected, the context and data of the original message may have been lost. The other problem is that Harry's actions are not intuitive for humans. In order to confirm a message, he must disconnect and reconnect his mobile device. In order to reject it, he must leave his device connected and wait for Carol's timer to expire.

One-time Secret. A much more general approach is to assume that Harry can send only insecure messages to Carol through Wanda. In this case, Harry and Carol pre-arrange a particular confirmation word. After receiving a message, Carol sends the response and waits for input from Harry. When she receives the expected word, the message is confirmed as genuine. A lockdown mechanism prevents Wanda from using a brute-force attack to guess the word (e.g., Carol might accept only a few incorrect attempts before considering the message a forgery). Harry is responsible for typing the word to confirm the message. He and Carol may have agreed beforehand upon the confirmation word, or Carol may randomly generate a word and embed it in her transformed response.

Clearly the confirmation word must be kept secret until it is used. Furthermore, in using it, the word is revealed to Wanda. Thus, each word can be used only once. This may create a usability problem for Harry if the words are established with Carol beforehand. Rather than simply remembering that his key is "apples and oranges" he now must remember a unique word for each transaction. With a long list, he will almost certainly need to carry a memory aid such as a paper list. With a short list, there is a limit to the number of transactions he can perform.

Including the confirmation word in the transformed response from Carol increases the user-friendliness of the system but may decrease the security. In this case, Harry is not required to remember anything; he simply must type a word that he sees embedded in the transformation. However, the secrecy of the word relies on the temporary confidentiality provided by the transformation. If Wanda can extract the confirmation word, either herself or with the help of Robert, then she can send and confirm arbitrary messages.

4 Performance

Besides the security and usability of the proposed system, performance is also of interest. Keyed transformations rely on hard problems which are often resource intensive (e.g., the 3-D transformation). Furthermore, the system is targeted towards areas of secure computation, including mobile devices which are often severely constrained in terms of processing and storage resources.

This section examines the resource requirements of the 3-D transformation and looks at the implications of offloading processing and storage requirements to a third-party platform.

4.1 Computational Resources

Our current unoptimized implementation of the 3-D keyed transformation requires between two and ten seconds per message on a modern workstation, depending on the key object models used. This makes the system usable on a small scale, but may be a problem for high-volume servers. Such servers may be able to increase performance through the use of specialized graphics hardware.

This processing requirement is completely unacceptable for low-powered mobile devices. For example, current smart cards typically operate at speeds of eight megahertz or less. The only option in this case is to use a separate computation server to do the rendering. The computation server is assumed to be accessible over a potentially hostile network; it never directly interacts with Harry. Communication between the trusted platform and the server is secured using traditional cryptographic methods. Carol's requests contain a scene to be rendered and are signed and encrypted so that only the server can read them. The response, containing the transformed image, is similarly authenticated.

From a security perspective, the ideal situation is that the computation server can be trusted not to disclose the key. If the server cannot be trusted, then forgeries may be possible. Because the server never interacts directly with Harry, it cannot forge or modify messages between Carol and Harry in the same way that Wanda can. However, if the server and Wanda cooperate, then arbitrary forgeries can be created. When Harry sends a message m, Wanda can send Carol an alternate message m' but ask the server to do the transformation using m. Carol thus returns to Harry a rendering with the message he expects. However, Carol has received Wanda's alternate message. When Harry sends the confirmation to Carol, he is confirming the wrong message. Alternatively, Wanda can ask the server to disclose the key to her, and she can do the rendering herself. Carol cannot display the returned image directly to Harry; she must ask Wanda to do it for her. Wanda can discard Carol's image and show her own rendering instead.

This attack relies on Wanda and the server being able to match messages to rendering requests. If Wanda knows how to contact the server directly, then she can simply send m and m' to the server. The server finds a rendering request that includes m' and replaces the message with m. If Carol is relying on Wanda to send messages through the network (as is the case with many mobile devices), then Wanda knows the address of the computation server and can communicate directly with it.

To prevent direct communication, Carol can send the rendering request through a mix-net[4] or other anonymizing system. In this case, the server doesn't know Wanda's address, so it cannot contact her directly. The anonymizing system can also deliver the rendering request to a random computation server, preventing Wanda from knowing the server's address. The two malicious parties (the server and Wanda) may still be able to simply guess each other's addresses. To reduce the probability of successful guessing, the list of possible servers should be large, as should the list of possible "Wandas." Furthermore, it may be useful for legitimate computation servers to keep track of requests from malicious terminals looking for malicious computation servers. By tracking these terminals, a blacklist of malicious terminals can be created to warn users.

Anonymizing requests to the computation servers prevents direct communication, but there may still be a covert channel. The transformed image is created by the server, sent to Carol, and then given to Wanda to display. The server can use steganography to encode information about the key in the returned image[5]. When Carol sends the image to Wanda, Wanda extracts the key, renders an alternate scene, and displays her new scene in place of the one Carol sent. To prevent the use of this covert channel, either Carol or the mix-net should attempt to strip steganographic information from the returned image[6].

4.2 Storage Resources

The 3-D models in our current implementation are described by the construction of geometric solids, and are thus on the order of a few kilobytes (stored in a human-readable form). Using polygonal mesh models would make the size much larger. It should be trivial for modern servers to store models of either type.

For resource constrained systems, however, even a few kilobytes of data may be pushing the limit. For many models, compression is a good solution (our human-readable models showed 75% compression using standard Lempel-Ziv encoding).

Another solution is to simply store a pointer to a model, rather than the model itself. Many constrained devices will have to offload the processing anyway; in this case, there is no need to store the model directly. If there are a finite number of models, then the trusted platform can simply store an index into the list of models (e.g., storing the word "dice" rather than the actual model; the computation server has a model that matches "dice"). Providing such a master list of models may make the key guessing attack described in section 2.2 easier. Wanda can create her list of guesses based on the list of models.

One final solution would be to have a separate storage location that is not trusted, but used only to store encrypted data. The constrained system would store only the key; the encrypted data and the key (encrypted so that only the computation server can read them) would be sent to the computation server. If each trusted platform provides its own model, then an arbitrary number of models can be used, making the key guessing attack more difficult. However, the user now has an additional responsibility to provide the storage (perhaps by carrying a business-card CD, USB storage device, or other medium).

5 Related Work

Previous work in the area of human-computer security protocols has frequently focused on authenticating a human user to a computer[7, 8]. Hopper and Blum provide a discussion of such human authentication protocols in [8]. However, the systems described provide only authentication of the involved parties, not the authenticity and integrity of individual messages. Furthermore, the practicality of such systems is in question.

Naor and Pinkas propose the use of visual cryptography for message authentication [9]. The user carries a printed transparency as a key; the cipher text is an image of the same size. When a user places the transparency over the image, the plaintext is revealed (with some amount of random noise). This system both requires the user to carry a physical transparency and can only be used a limited number of times.

Gobioff et al describe protocols for providing trusted input and output with a smart card[3]. However, their suggestions rely on installing low-bandwidth (e.g., single-bit) trusted input and output paths to the card. To date, this has not been done.

Stabell-Kulø et al define a protocol for authenticating messages to be signed by a smart card[10]. Their proposal relies on the user computing a one-time pad substitution cipher over the message. This not only requires the user to carry a memory aid of the substitution, but it is a computational burden to the user. They do, however, introduce the concept of a confirmation word as used in section 3 of this paper.

6 Conclusions and Future Work

The use of hard AI problems as security primitives is a relatively new field. This paper explains the concept of using AI problems to check the integrity and authenticity of messages, as well as introducing metrics for evaluating individual problems. The construction of the 3-D keyed transformation highlights some of the difficulties of this approach, both in terms of security and usability.

The next step in this line of research is to develop and evaluate individual keyed AI transformations. The 3-D transformation described has been subjected to very limited usability testing. Similarly, the difficulty in breaking the transformation has been evaluated only on a small scale. The true values for γ, δ, and ϵ can only be determined by the consensus of the AI and security research communities. Even if the 3-D transformation proves useful, other transformations may have desirable properties. For example, an audio-based transformation may be useful alongside a visual transformation to provide greater accessibility.

Furthermore, complete systems that use keyed transformations need to be developed and deployed. Many of the details of the protocols are specific to individual situations (e.g., the capabilities of the trusted computing device).

Using hard AI problems as security primitives is a little-explored but worthwhile pursuit. In particular, many of the techniques map directly to ways in which humans intuitively ensure security in the real world. The 3-D transformation attempts to make a coherent scene so that cut-and-paste forgeries are impossible. Similarly, nobody would trust a paper contract in which a piece of paper with text had been glued on top the original document. People authenticate each other over the telephone using the unique sound of their voices. The speech transformation attempts to do exactly this by giving the trusted computer a unique, audible voice.

This paper takes a clear step in the direction of human-computer authentication. It is hoped that researchers, both in the security community and without, will help to advance this field both by developing schemes based on specific problems, and by trying to break existing schemes by solving the underlying problems.

References

1. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: Using hard AI problems for security. In: Advances in Cryptology – Eurocrypt 2003. Volume 2656 of Lecture Notes in Computer Science., Springer-Verlag (2003)
2. Kochanski, G., Lopresti, D., Shih, C.: A reverse turing test using speech. In: Proceedings of the International Conferences on Spoken Language Processing, Denver, Colorado (2002) 1357–1360
3. Gobioff, H., Smith, S., Tygar, J.D., Yee, B.: Smart cards in hostile environments. In: Proceedings of the Second USENIX Workshop on Electronic Commerce. (1996)
4. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24** (1981) 84–88
5. Smith, J.R., Comiskey, B.O.: Modulation and information hiding in images. In: Workshop on Information Hiding. Volume 1174., Isaac Newton Institute, University of Cambridge, UK (1996) 207–226
6. Johnson, N.F., Jajodia, S.: Steganalysis of images created using current steganography software. In: Second Workshop on Information Hiding. Volume 1525 of Lecture Notes in Computer Science., Portland, Oregon, USA (1998) 273–289
7. Matsumoto, T.: Human-computer cryptography: an attempt. In: Proceedings of the 3rd ACM conference on Computer and communications security, ACM Press (1996) 68–75
8. Hopper, N.J., Blum, M.: Secure human identification protocols. In: Advances in Cryptology – Asiacrypt 2001. Volume 2248 of Lecture Notes in Computer Science., Springer-Verlag (2001) 52–66
9. Naor, M., Pinkas, B.: Visual authentication and identification. In: Advances in Cryptology – Crypto '97. Volume 1294 of Lecture Notes in Computer Science., Springer-Verlag (1997) 322–336
10. Stabell-Kulø, T., Arild, R., Myrvang, P.H.: Providing authentication to messages signed with a smart card in hostile environments. In: USENIX Workshop on Smartcard Technology. (1999)