Defeating Malicious Terminals in an Electronic Voting System

Daniel Hanley [1], Jeff King [1], André dos Santos [1]

[1]College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30332-0280
USA

Abstract. *The advent of electronic voting gives rise to a new threat: Adversaries may execute undetectable, automated attacks against the system. Elections are often secured through complex policies, which may be difficult to enforce; Completely Automated Public Turing Tests to Tell Computers and Humans Apart (CAPTCHAs) provide an inexpensive alternative. The goal of this study is to introduce a unique application of CAPTCHAs that allows a human to transmit a message securely across an untrusted medium, and this has direct implications in the domain of electronic voting. We assume that the voter is equipped with a trusted voting device capable of digitally signing the vote. A trusted tallier generates a CAPTCHA-encrypted ballot, which contains a one-time pad, a mapping of candidates to values. This CAPTCHA is sent to the user across an untrusted voting terminal. The user transmits to the trusted device a value corresponding to his chosen candidate, which is signed using a blind signature scheme and transmitted to the tallier. Finally, the tallier then translates this value into the voter's selected candidate. All steps of such a protocol must be defined such that they are usable by all voters, and we will consider the usability of some example CAPTCHA-based voting systems.*

1. Introduction

A system that fails to limit its dependency on trusted parties is generally prone to abuse. This is why a democratic system aims to restrict trust such that those in power are held accountable for their actions. Unfortunately, some electronic voting systems deployed recently in the United States and elsewhere may extend trust to a much wider and less accountable set of machines and individuals.

In a traditional voting system, the voter directly records a vote on some physical medium, and places it in the care of voting administrators and talliers who are trusted with the task of securely storing the votes. A set of checks and balances, coupled with the physical nature of the vote record, complicates attacks against this system. It is simply inefficient to physically modify a significant number of votes, and such attacks are easy to detect due to oversight and auditing by multiple non-cooperating parties. For instance, a voting administrator may attempt to attack this system by discarding or modifying ballots, yet this attack is expensive, as a large number of ballots must be modified in order to have a substantial impact on the election's outcome. Any physical attack – particularly a large-scale attack – will necessarily generate a fair amount of physical evidence. Typically, processes are in place which detect and respond to such attacks: Administrators of varied political interests monitor the elections, ballots are audited afterward, and so forth. This traditional system is not absolutely secure, of course, since its security is no better than the design and implementation of the policies that govern the elections. Security is therefore burdened by human shortcomings and the expense of training voting administrators, who are often volunteers. Furthermore, the actual physical storage and transmission of votes is costly and potentially inaccurate with respect to digital alternatives. An improved voting system shall reduce these expenses, improve accuracy, and maintain a comparable level of security.

In an electronic voting system, the voter indirectly records his vote by means of some computer, hereafter referred to as the voting terminal. Electronic voting is attractive due to its convenience and efficiency, but these advantages may be used *against* the system. If the system is not secure, then an adversary may execute successful, automated attacks against it conveniently and efficiently. An adversary who tampers with a voting terminal may modify or replace its software, thereby impacting a great quantity of votes while leaving negligible physical evidence of the attack. The voting system therefore extends trust to these voting terminals as well as all parties with the capacity to access and modify the voting terminals. In such a system, administrators must apply expensive, cumbersome policies to secure the system. In practice, these policies are often poorly enforced. The vast number of voting terminals and the cost of proper policy implementation guarantees that a significant number of machines are left unattended and unsecured prior to an election. The policies additionally limit the voting system's availability, preventing, for example, Internet voting through public computers. We propose a solution that addresses these concerns, securing the voting system such that it promotes both confidence and increased accessibility.

Many schemes have been proposed which allow users to vote both securely and anonymously, limiting the trust that must be placed in other entities [Okamoto, 1997, Hirt and Sako, 2000]. While these theoretical approaches do satisfy some important requirements of the voting problem, other practical concerns, such as usability, are virtually ignored. Similarly, Chaum's voting protocol [Chaum, 2004] and Prêt à Voter [Ryan, 2005] require a voter to perform additional verification and auditing tasks to ensure his vote is counted, and even these countermeasures do not absolutely guarantee the integrity of the vote. In many other schemes, the voter is responsible for performing complex mathematical operations that require the use of a computer. The voting terminal can clearly perform the operations, but then the user is again forced to trust the terminal. One solution which has been proposed is the use of a personal trusted and tamper-resistant device. Through a secure registration process, each voter obtains a trusted voting device that is capable of performing the necessary computations on his behalf. Such devices can be more easily audited than large-scale voting systems.

If the voter is able to directly interact with the trusted device, then the system works. However, in most cases, such devices are not capable of performing such interactions due to issues of form factor and cost. The voter instead connects the trusted device to an untrusted terminal, which may or may not faithfully relay messages between the two parties. Such a terminal could modify votes in a completely automated fashion, changing votes for one candidate to votes for another. In brief, this system lacks a trusted path from the human to the voting device.

If such a device is to be useful, there must be a system that allows a human voter to transmit votes to a trusted device across an untrusted medium in such a way that it is difficult for the votes to be automatically modified, discarded, or forged. Furthermore, the system must be easy for voters to use. This paper introduces a secure, easy-to-use system based on a one-time random substitution and the use of hard artificial intelligence problems. Section 2 describes background concepts needed to understand the system. Section 3 outlines the protocol itself, giving specific examples. Section 4 analyzes the security aspects of the protocol.

2. Definitions and Related Work

Let Alice be a human voter who wants to take part in an election. She will choose some candidate c from the set of candidates $C = [c_1, c_2, \ldots, c_n]$. Alice must convey c to the tallier, Trent, without compromising her anonymity, while also proving to Trent that a vote for c was cast by a registered voter. We assume that Alice obtains a voting token through a secure registration process. Furthermore, this token is a trusted computing device that performs any necessary computations on Alice's behalf, depending on the exact method of vote counting. For example, in a blind signature voting scheme [Okamoto, 1997], this device must blind the votes, send them to the registrar along with Alice's identity, then send the signed votes to Trent.

We assume that Alice and Trent cannot directly interact. This is a reasonable assumption in any large-scale voting system: Direct interaction with Trent would require a very large number of verifiably high assurance voting machines with trusted paths – secured at all layers, from physical to application. This approach is clearly cost prohibitive. Consequently, the voters must centralize trust in a remote tallying system. Because Alice and Trent cannot directly interact, they must use an untrusted terminal to relay messages to each other. We will refer to the untrusted terminal as Mallory; she is capable of intercepting, modifying, or fabricating any message transmitted between Alice and Trent.

The objective here is to securely transmit c from Alice to Trent, henceforth assuming that all other election procedures are secure, including the distribution of voting tokens and the actual tallying process, as these problems are adequately addressed elsewhere. To accomplish this stated goal, we will construct a protocol that uniquely combines the tools listed below.

2.1. CAPTCHA

The CAPTCHA is a relatively recently defined tool, introduced in [von Ahn et al., 2003], and its applications as a security primitive have yet to be fully explored. Because it is a very useful tool in preventing automated attacks, it is sensible to use this tool to combat automated attacks accompanying the transition from traditional to electronic voting systems. Indeed, CAPTCHAs are often used to deter ballot stuffing in web-based polls. However, since the security requirements of political elections are much more stringent, we shall see that a traditional application of CAPTCHAs cannot solve their problems.

CAPTCHAs are used in challenge-response protocols in order to distinguish humans and computers. Typically, a human is presented with a problem that modern artificial intelligence cannot solve, such as a computer vision problem, and the human responds with a solution to the problem. For instance, the human may identify an object within a randomly generated and randomly distorted image. Commercial web site operators have found this tool invaluable in averting automated attacks. More specifically, CAPTCHAs have successfully defended email service providers by preventing the automated registration of email accounts.

We have already established that the primary difference between electronic and traditional voting lies in an adversary's ability to automate attacks efficiently and without fear of detection. While it may appear that CAPTCHAs offer a sound defense, a trivial application of CAPTCHAs is inadequate. Suppose that a voter must solve a traditional CAPTCHA to vote. Trent randomly generates s, a string of text, and $T(s)$, an image wherein this text appears mangled and unreadable by a machine, but readable by a human.

Trent transmits $T(s)$ to Alice through Mallory, and Alice responds with s and c, Alice's chosen candidate. While this does prove to the tallier that each vote is associated with a human, it does not prove that the voter actually cast the specified vote; Mallory may have accepted the Alice's response to the CAPTCHA, s, while also tampering with the vote, c. Since the steps of voting and human authentication are separate, in this case, the simple challenge-response protocol is vulnerable. It fails because c is not protected from Mallory's modification. We shall address this vulnerability by combining the CAPTCHA with one-time random substitution.

2.2. One-time random substitution

One method Alice and Trent can use to communicate securely is a one-time random substitution. In this case, Alice and Trent pre-arrange a set of messages M, which may be public, and a secret bijective mapping $K : C \rightarrow M$. If Alice wishes to vote for a candidate c_i, she sends the message $m_i = K(c_i)$. Mallory can modify m_i in transit, but without knowing the meaning of the message (given by the mapping K), she cannot make a meaningful modification. This scheme is a simplified version of the protocol given in [Stabell-Kuløet al., 1999].

This system has several drawbacks. One is that Alice and Trent must agree upon the secret mapping K beforehand. If the candidate c_i is ever revealed to Mallory, then the mapping K is trivial for her to compute. Thus a given mapping should be used only once, making pre-agreement even more burdensome. The ease with which such a system can be used is also poor. Alice must remember or carry with her the mapping K and vote indirectly using that mapping. Further below, we shall see that CAPTCHAs enable Trent to securely transmit K to Alice.

2.3. KHAP

The Keyed Hard AI Problem (KHAP) protocol [King et al., 2004, King and dos Santos, 2005] is designed to transmit messages securely from a human to a computer across an untrusted medium. Like CAPTCHAs, the protocol relies on the difficulty computers have in solving certain hard artificial intelligence problems. Alice and Trent pre-arrange a key, the exact nature of which depends on the AI problem being used. Alice transmits her vote to Trent in an insecure fashion. Trent then returns an instance of a hard AI problem which embeds both her vote and the key. For example, if Alice's vote is ”Alice” and her key is ”bananas”, Trent may respond with a two-dimensional rendering of a 3-D scene showing a picture of Alice and a picture of a banana. The intent is that Mallory can neither produce a forged image (because she doesn't know the key, and cannot extract it from the scene) nor modify the image to carry a different message (because of the complexity of the scene). Thus Alice believes that Trent correctly received her vote. She then signals success to Trent by sending a confirmation message; the message can be pre-arranged or embedded in the returned image.

While this protocol satisfies the requirements of a voting system, it also suffers from the pre-arrangement of the key. Though a single key may be used multiple times, repeated use increases the chance that the details of the key may be extracted by Mallory. The protocol is also much more complex for Alice to use. Not only must she remember and recognize her key, but there is now an additional step for her to perform. Previously, she needed only transmit her vote; now she must verify and confirm Trent's message.

3. Proposed Solution

We have explained several methods whereby a human, Alice, may transmit a vote across an untrusted medium, Mallory, to a trusted tallier, Trent. Unfortunately, each of the solutions referenced creates obstacles that might impede a practical implementation in an actual election. Here, we shall take inspiration from two techniques explained above. The goal is to empower the human voter with the ability to encrypt a message such that it cannot be deciphered or forged by a computer adversary. This shall be achieved through a process that is entirely transparent and natural to the user. In this scheme, Trent will transmit a CAPTCHA-encrypted, human-readable key to Alice, which Alice will use to implicitly encrypt her vote without additional effort. We assume that all messages between Mallory and Trent are transmitted through a secure channel, and all messages from Alice to Trent are authenticated using Alice's voting token according to the blind signature scheme referenced in Section 2.

3.1. Protocol

The actors in the protocol are, as described above, Alice, Trent, and Mallory. Now, we may formally state the data and functions required by each actor to participate in the protocol:

- Each actor in the protocol knows the public list of candidates $C = [c_1, c_2, \ldots, c_n]$.
- Each actor in the protocol knows the public set of random elements $R = [r_1, r_2, \ldots, r_m]$ such that $m \geq n$.
- Trent can create a random mapping of candidates to random elements $K : C \mapsto R$. Since this is a random mapping, for all i and j, the probability $P(K(c) = r_i)$ is equal to $P(K(C) = r_j)$. K has the inverse mapping $K^{-1} : R \mapsto C$. As a one-time pad, this function performs simple substitution on a candidate c to yield a random element r.
- Trent can encrypt an arbitrary message m using the CAPTCHA test encryption function $T(m)$ such that Mallory cannot derive m from $T(m)$, although Alice may obtain m from $T(m)$.

Let us assume that it is possible for the CAPTCHA function T to encrypt a representation of the mapping K in a human-readable format. This is denoted by $T(K)$. In practice, to aid Alice in evaluating $K(c)$, the elements of R may be related to the CAPTCHA transformation, although this is not a requirement of the protocol. Examples below will elaborate on how this is done.

The protocol is defined as follows:

1. Trent generates and sends an encrypted ballot.
   - 1.1. Trent creates the random, bijective mapping $K : C \mapsto R$.
   - 1.2. Trent evaluates $T(K)$, encrypting the mapping using the CAPTCHA transformation.
   - 1.3. Trent transmits $T(K)$ to Alice through Mallory.
2. Alice responds with the encrypted candidate.
   - 2.1. Alice decrypts $T(K)$ using her cognitive and perceptual abilities.
   - 2.2. Alice evaluates $K(c)$ to obtain r.
   - 2.3. Alice transmits r to Trent through Mallory.
3. Trent decrypts Alice's preferred candidate.
   - 3.1. Trent evaluates $K^{-1}(r)$ to obtain c.

Trent first generates the mapping K; this is a one-time substitution cipher. This mapping is used as the input of a CAPTCHA function. The output of the CAPTCHA function is a representation of the mapping that Alice can easily understand, and Alice therefore easily obtains r, which she sends to Trent. Since Trent has defined the mapping K, he has the inverse mapping K^{-1}, and he uses this to determine Alice's chosen candidate c. This outlines a procedure that enables communication between Alice and Trent through Mallory, but we must additionally prove that Mallory cannot compromise this communication. In Section 4, we shall evaluate the security of this protocol, demonstrating how it prevents attacks that would otherwise threaten an electronic voting system.

3.2. Examples

Due to the vast variety of CAPTCHAs, this protocol gives us a great amount of flexibility in developing secure voting systems. It is clear that a voting system using this protocol must afford a reasonable interface to ensure availability among all eligible voters. The use of a CAPTCHA implies that Alice has particular cognitive and perceptual abilities. Furthermore, the ballot embedded in this CAPTCHA must present a clear mapping between C and R. The implementation of a CAPTCHA voting system must consider these implications carefully. To further specify what this may entail, we shall introduce the following examples.

3.2.1. Text CAPTCHA

The traditional, text-based CAPTCHA is the most familiar CAPTCHA domain, and it would therefore serve as a sound basis for this first example of a CAPTCHA-based voting system. In this example, suppose that the CAPTCHA function T renders the names of the candidates $c_1, c_2, \ldots, c_n$ in a two-dimensional image. The positions and orientations of these names in the image are randomly generated, and the region in the image occupied by each name is an element of R. Each element of R is therefore a random, non-overlapping range of image coordinates. This clearly expresses the mapping $K : C \mapsto R$. Additionally, the CAPTCHA function T adds to the image some amount of noise sufficient to render the resulting images unreadable to Mallory. This noise consists of arbitrary lines, shapes, and distortion filters. The resulting image is $T(K)$.

Trent sends this image to Mallory, who displays it to Alice. Alice may be oblivious to the mapping between candidates and image regions; she simply acknowledges that she may securely cast her vote by identifying and selecting her chosen candidate in the image. For instance, Alice might touch the name of the candidate, which is displayed on a touchscreen interface. Once Alice selects the region r of the image containing candidate c. Alice then sends r to Trent through Mallory. Since Trent produced K, he may easily obtain the inverse mapping. He evaluates $K^{-1}(r)$, determining which candidate name was rendered in the specified image region. Trent thus acquires c, and he records Alice's vote.

3.2.2. 3D Animation CAPTCHA

Now, we consider the domain of three-dimensional animation. A CAPTCHA based in this domain presents Mallory with a challenging computer vision problem. In this scenario, Trent's CAPTCHA function T renders a three-dimensional environment. In this 3D scene, T randomly generates a background setting and some objects that contribute to the noise of the scene; the amount of noise is adequate to prevent Mallory from extracting any useful information from the scene. Additionally, every candidate in C has a 3D representation; these 3D representations of the candidates are randomly distributed throughout the scene, maintaining some reasonable distance from each other. These candidate representations might be 3D objects resembling the candidate, the symbol of the candidate's party, the name of the candidate, or some combination of these. Finally, the “camera” of the 3D scene is animated such that its path traverses the scene in a random manner, yet this path is defined such that the camera approaches and focuses on each candidate c for some distinct set of animation frames r. In this example, K is therefore the mapping of candidates to sets of frames. The rendering of this animation is $T(K)$.

Trent sends $T(K)$ to Mallory, who plays the animation for Alice. The animation individually displays each candidate's representation in a 3D environment. Alice again unconsciously deciphers the mapping; a particular candidate is shown in a particular set of frames of the animation, but Alice is not required to acknowledge the underlying mapping and security protocol. When Alice sees her preferred candidate c represented in the scene, she notifies Mallory. Mallory then transmits the current frame identifier f to Trent. Trent determines that $f \in r$, and he calculates $K^{-1}(r) = c$. Trent may easily perform this calculation; since he rendered the animation, he may determine which candidate was displayed in the specified frame. Finally, Trent records the vote.

3.2.3. Audio CAPTCHA

Speech recognition is another difficult problem for Mallory, while it is a trivial problem for Alice, and therefore we may employ speech synthesis in the CAPTCHA transformation T [Kockhanski et al., 2002]. This approach is quite similar to the previous example in that it relies upon a temporal mapping of candidates. T represents each candidate c through speech synthesis. In the resulting audio, each candidate is represented during a distinct set of audio time codes r. K thereby maps each candidate c to a distinct r. In addition to synthesizing the speech representations of candidates, T must also generate audio noise as an obstacle to Mallory, who may attempt to acquire K. The composite audio output is $T(K)$.

Trent transmits $T(K)$ to Mallory, who plays the audio for Alice. Once again, it is not necessary for Alice to understand the mapping between candidates and time codes; Alice listens to the audio, notifying Mallory at time t when she hears the name of her preferred candidate. Mallory transmits t to Trent, who recognizes that $t \in R$. Thereafter, it is simple for Trent to determine which candidate's name was spoken at this particular moment in the audio. Trent calculates Alice's vote, $K^{-1}(r) = c$, which he proceeds to record.

4. Analysis

Here we shall characterize the threats leveled by Mallory against the protocol defined above in the context of an electronic voting system. We will evaluate how the proposed system performs when confronted with various types of attacks, ultimately judging the protocol's security according to a reasonable evaluation criterion: Is the security of the proposed system comparable to that of a traditional voting system?

Fabricated Vote through Guessed K Mallory may attempt to challenge the integrity of Alice's vote, replacing her vote c with a vote for Mallory's preferred candidate c'. With no information as to the mapping K, Mallory must transmit to Trent some r' such that $K^{-1}(r') = c'$. Let us assume that Mallory has access to both $C = [c_1, c_2, \ldots, c_n]$ and $R = [r_1, r_2, \ldots, r_n]$. Lacking the mapping K, Mallory cannot directly calculate $K^{-1}(r')$, and she is left with no recourse but to select an arbitrary $r'' \in R$. Trent evaluates $K^{-1}(r'') = c''$. If $c'' = c'$, Mallory has voted for her preferred candidate. We may trivially evaluate the probability of this event in the case that $m = n$. Because K is a random mapping, it maps $c_k \in C$ to $r_k \in R$ with uniform proability, and so $P(c'' = c') = \frac{1}{n}$.

Consequently, in this scenario Mallory has a $\frac{1}{n}$ probability of voting for her preferred candidate and a $\frac{n-1}{n}$ probability of voting for another candidate. Mallory clearly cannot influence the final tally in her favor, and therefore she has little incentive to adopt this strategy. It is impossible to attack a traditional voting system in this fashion, yet we should note that the attack cannot reliably sway the tally toward any particular candidate, but instead it “randomizes” Alice’s vote.

In the case where the number of candidates, n, is less than the number of random elements, m, the probability that Mallory will guess the correct element of R is less than $\frac{1}{n}$. Instead, it is entirely possible that $K^{-1}(r'')$ is undefined, since r' does not necessarily map to a candidate. The probability of this event is $\frac{m-n}{m}$, and the probability that Mallory will guess the correct element of R is $\frac{1}{m}$. If Mallory selects an invalid r'' (some r'' such that $K^{-1}(r'')$ is undefined) Trent detects the attack, and it is thereby thwarted.

If Mallory modifies a single vote in this manner, the impact on the election is insignificant; it is much more likely that Mallory will attempt to modify many votes, which will result in a disproportionately high number of invalid votes. These invalid votes may be detected during the election or during an auditing process to verify the accuracy of the election, investigate attacks, and, if necessary, discard the questionable votes.

Fabricated Vote through Cracked T In the above scenario, we assumed that Mallory had no knowledge of K due to the encryption of the CAPTCHA function T. Recall that T is essentially a cipher whose key is composed of some set of human cognitive and perceptual abilities. Suppose that Mallory can simulate these abilities; then she has cracked T and may thus obtain K. Having obtained K, Mallory may evaluate $K(c') = r$, and send this result to Trent. Alice’s vote was thus stolen in an automated fashion, and she cannot detect the attack. The security provided by the voting system in this case is no greater than the security of current electronic voting systems. Moreover, if Mallory is capable of intelligently analyzing the output of T, even if she cannot obtain K, she may increase the probability $P(K(r') = c')$ and thereby influence the election in her favor.

However, this attack is based on the assumption that T is insecure. It is asserted in [von Ahn et al., 2003] that it is impossible to prove the security of a CAPTCHA due to frequent advances in the field of artificial intelligence. However, it is also noted that cracked CAPTCHAs may be easily replaced with CAPTCHAs based upon harder problems, and CAPTCHAs have a successful history of securing commercial applications.

Unfortunately, there is a fundamental difference between standard CAPTCHAs and those described in the examples above. Standard CAPTCHAs issue a random challenge to the human, generally in the form of a text string; in the examples detailed above, the CAPTCHA transmitted by Trent expresses the nonrandom set C. The security of these examples depend on Mallory’s inability to identify elements of C the CAPTCHA, yet the nonrandom components of these CAPTCHAs are presumably easier to identify than the random components of standard CAPTCHAs. This disadvantage may be counterbalanced by increasing the difficulty of the CAPTCHA, as suggested by [von Ahn et al., 2003].

Cracking T through a Human Adversary We have assumed heretofore that Mallory, a computer, is responsible for all threats to the voting systems. Depending on the context, collaboration between a human adversary and Mallory could lead to successful exploitation of the voting process; Mallory could forward $T(K)$ to the human collaborator, who may easily acquire K and cast a vote with the stolen ballot. Alice’s actual vote is then discarded by the voting terminal. The security of this system is similar to that of a traditional voting system; the modification of ballots is possible, but it requires a manual effort on the part of the attacker. One important difference is that the human attacker may modify ballots without leaving any physical trace; the adversary has no fear of negative repercussions. While this threat is disheartening, it may be mitigated through a common countermeasure.

Following the example of current voting systems, policies may be established to prevent the human adversary from interfering with the communication between Alice, Mallory, and Trent. These policies could physically restrict access to the voting communication network during the election and prevent communication between the voting network and any outside parties. This would unfortunately disable Internet voting, yet it does have advantages over current implementations of electronic voting systems: It is relatively secure and reduces the cost of administering the election.

To enable Internet voting, the system could simply accept the potential for remote human collaborators. We must note that the voting protocol is time-dependent: Mallory and human collaborators may only steal Alice's vote while Alice is engaged in the protocol, since the vote must be signed by Alice's voting token. Consequently, not only must human collaborators manually solve cast each vote, these votes must be synchronous with a voter's use of a compromised terminal. To significantly influence the election, the adversary requires assistance from many human collaborators, all of whom may remotely communicate with many compromised voting terminals. In this case, both the organized group of attackers and large number of malicious terminals greatly increase the likelihood that the attack is detected by voting administrators. Since traditional voting systems face similar threats from large and highly coordinated groups of attackers, an Internet voting system may be viable. However, it must still address the problem of coercion.

Fabricated Ballot CAPTCHAs are by definition publicly available, and while Mallory cannot decrypt the CAPTCHA encryption function, she may easily obtain T. Furthermore, it is trivial for Mallory to generate a fabricated mapping $K' : C \mapsto R$. Then, Mallory displays $T(K')$ to Alice, who cannot discern between this fabricated ballot and the legitimate $T(K)$. Alice therefore responds as she normally would, transmitting r' to Mallory. Mallory may then calculate $K'^{-1}(r') = c$; this gives Mallory knowledge of Alice's preference. Since the voting process is anonymous, this tactic alone is inconsequential, as it does not compromise Alice's vote. However, this approach may be effectively used in conjunction with other attacks.

Selective Denial of Service Now, we assume that Mallory cannot obtain K through weak CAPTCHA encryption or a human collaborator. Mallory may successfully execute an undetectable denial of service attack against Alice, but the impact of this attack indiscriminately discards votes without regard for the candidates voted for. Because Mallory doesn't know Alice's preference, she may be throwing away votes for her preferred candidate, c'.

In order to perform a useful denial of service, Mallory must learn about Alice's voting preference. She may do this with the aid of some heuristic – using Alice's and Mallory's geographical location, for instance. If the location typically has the preference for one candidate, then discarding votes from that location will favor other candidates overall.

Mallory may also use a fabricated ballot to learn Alice's preference. She can then deny Alice's vote if it does not coincide with Mallory's preference. However, there is a complication. If Mallory does not want to deny Alice's vote, she must still submit the vote. Because of the transformation T, Mallory cannot submit the vote herself. However, Alice's vote is only valid for the fabricated ballot, not for the real ballot sent by Trent. Thus, Mallory must present Alice with the real ballot and ask her to re-submit her vote.

This may indicate to Alice that Mallory is cheating. Unfortunately, since Mallory and Alice share a common goal – to vote for a particular candidate – Alice may or may not respond to the detected attack. This situation is highly unlikely, however, since it assumes that Alice is aware and complicit with this abuse, whereas voters supporting other candidates are ignorant of the attack.

If multiple votes are being cast, each with its own CAPTCHA ballot, Mallory doesn't have to worry about this scenario. She can use a fabricated ballot to learn Alice's preference for the first vote. This can give Mallory a reasonable chance at guessing Alice's preferences for future votes (e.g., if Alice votes along party or ideological lines). The simplest solution to this is for all votes to be cast as part of a single transformed ballot.

While Mallory may attack the system in this automated fashion, it is possible for Alice and Trent to detect a dropped vote. To drop Alice's vote, Mallory has two options: sending a random, fake vote or sending nothing. If Mallory arbitrarily guesses r'', there is a probability of $\frac{m-n}{m}$ that this r'' is invalid, as shown above, and Trent therefore detects the attack. If Mallory simply ceases communication with Trent, refusing to cast Alice's vote, then Alice may detect this denial of service by contacting the registrar and learning whether a vote was cast on her behalf.

Educated Guessing Mallory may also use her knowledge of Alice's preference to increase her own chances of successfully casting a vote for her preferred candidate. Assume that Mallory has guessed Alice's preference c (through a fabricated vote or other means), and that it is not equal to Mallory's preference, c'. Mallory intercepts and discards Alice's transmission of $K(c) = r$. She now knows that $K(c') \neq r$, increasing the likelihood of successfully guessing $K(c')$ to $\frac{1}{m-1}$. This threat can be mitigated by keeping m large.

5. Conclusions and Future Work

The current state of electronic voting is problematic; despite the changing nature of threats against the system – specifically, undetected, automated attacks – the current voting systems attempt to employ strategies associated with traditional voting systems, security policies based in the physical domain. We have sought to improve upon this system through the introduction of an enhanced voting protocol tailored to address these new threats. The protocol presented combines a unique application of a burgeoning countermeasure, the CAPTCHA, with a provably secure one-time pad. The combination of these tools yields a protocol that enables humans to easily peform encryption with secure key transmission. This has direct applications in the domain of electronic voting, where a human must securely transmit a message to a trusted party across an untrusted channel. Due to the proposed voting system's use of CAPTCHAs, the system requires human interaction, protecting it from automated attacks that reliably sway the vote tally. Automated, selective denial of service attacks may succeed in stealing individual votes, yet large-scale attacks have a high chance of detection. It is interesting to note that traditional attacks against traditional voting systems share this security property, and therefore we may conclude that the security of these systems is comparable.

Many related issues demand further exploration. Research in this area is largely driven by practical concerns, namely a human's lack of computational and storage resources needed to easily perform encryption. Due to this emphasis on practicality, it is important to obtain quantitative usability data regarding CAPTCHA-encrypted ballots to ensure the system's fairness; it must be equally usable by all eligible voters. To this end, feasibility testing of implementations of the described example systems will help us identify an ideal balance of usability and security. Simultaneously, it is important to consider threats from breakthroughs in the field of artificial intelligence. Finally, we may generalize our research, applying the protocol defined herein to new scenarios.

References

Chaum, D. (2004). E-voting: Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47.

Hirt, M. and Sako, K. (2000). Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology – EUROCRYPT '2000*, pages 539–556. Springer-Verlag.

King, J. and dos Santos, A. (2005). A user-friendly approach to human authentication of messages. In *Proceedings of FC05, Financial Cryptography and Data Security*.

King, J., dos Santos, A., and Xuan, C. (2004). KHAP: Using keyed hard AI problems to secure human interfaces. In *Proceedings of IV Workshop em Seguranca de Sistemas Computacionais*, Gramado, RS, Brasil.

Kockhanski, G., Lopresti, D., and Shih, C. (2002). A reverse turing test using speech. In *Proceedings of the International Conference on Language Processing*, Denver, Colorado.

Okamoto, T. (1997). Receipt-free electronic voting schemes for large scale elections. *Proc. of Workshop on Security Protocols '97*, pages 25–35.

Ryan, P. Y. A. (2005). A variant of the chaum voter-verifiable scheme. In *WITS '05: Proceedings of the 2005 workshop on Issues in the theory of security*, pages 81–88, New York, NY, USA. ACM Press.

Stabell-Kulø, T., Arild, R., and Myrvang, P. H. (1999). Providing authentication to messages signed with a smart card in hostile environments. In *USENIX Workshop on Smartcard Technology*.

von Ahn, L., Blum, M., Hopper, N., and Langford, J. (2003). CAPTCHA: Using hard AI problems for security. In *Proceedings of Eurocrypt 2003*.