Independent Computations for Safe Remote Electronic Voting

Alec Yasinsac
University of South Alabama

1. Introduction

In a world where a high percentage of citizens carry a smart phone or iPod and most households have one or more computers, it is difficult to understand why we have not been able to leverage powerful personal computing devices to allow citizens to cast their ballots electronically. The argument turns on security weakness in all kinds of network computing and on theoretical limits on the ability to protect critical applications.

Though there are several voting system vendors that provide Remote Electronic Voting (REV) services, the SERVE Report [1] and its subsequently published version [2] continue to define REV security discussions. One of the most serious security concerns with REV is the danger of malware on the voter's client machine. Malware can easily be designed to prevent the voter from successfully voting, or to violate the voter's privacy by sending a copy of the ballot to a third party, or even to surreptitiously modify the voter's choices before the ballot is encrypted for transmission so that the wrong votes transmitted and counted without anyone knowing. While we have no solution to the first two kinds of malware threats, in this paper we present an all-electronic protocol that greatly reduces the likelihood that malware can modify a ballot without detection. To date, we have not seen an Internet voting solution proposed for real elections whose design effectively addresses ballot integrity on a computer that is malware infected.

In this paper, we propose a verifiable, paperless Remote Electronic Voting protocol that leverages independent computations, one for voting and one for verification, to prevent acceptance of any ballot on which malware on the voting client has altered the voter's selections. Our solution reduces the likelihood that malicious software on the voting client or assistive device can alter REV voter selections. While our approach requires device properties that may not be widespread in the general population, we contend that these protocols are well suited for use by some constituencies, such as the U.S. military members and federal service employees serving overseas.

The rest of this paper is organized as follows. In the next section, we give an overview of Internet Voting architectures and follow with a description of our protocols, the prerequisites of the voting environment, and provide details of the system's properties, specifically including things that the protocol does not do. We close with summary and concluding remarks.

2. Internet Voting Architectures

Voting integrity is commonly considered in terms of the voter's ability to have justified confidence in three serial steps, i.e. that their selections are:

a. Cast as Intended
b. Recorded as Cast
c. Tallied as Recorded

These steps form the core of the five Voter Integrity Phases (VI Phases) shown in Table I.

CaI involves ensuring that the voter is able to find their preferred candidates/choices on the ballot and that they are able to unambiguously indicate their selections. This is usually managed through user interface activities such as ballot design, analysis and testing for paper ballots and electronic ballot engineering for electronic voting devices.

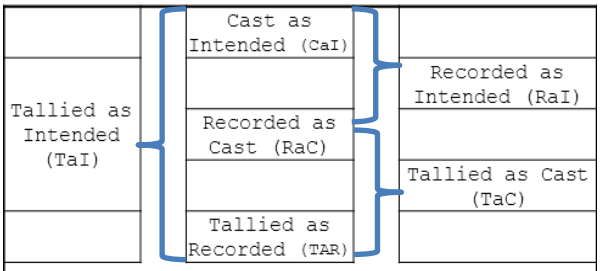

Table I. Voter Integrity Phases

The latter two self-descriptive VI Phases are beyond the capabilities of current election practice. That is, with existing voting technology (e.g. Precinct Count Optical Scan and touch screen Direct Recording Electronic devices) the voter has a limited ability to prove that their ballot is either recorded as cast or that it is properly included in the final tally.

Cryptographic voting systems leverage mathematical formula in order to attempt to provide voters the full spectrum "Tallied as Intended" (TaI) proof. Computer scientists have been developing cryptographic voting protocols that can have provable security and accuracy properties for years, e.g. [3, 4, 5]. The key to many cryptographic voting systems is that they leverage a voter a feedback channel. That is, at the time of voting, each voter receives, or generates, something that they can use after results are reported to ensure the accuracy of the voting process. Of course this information is

most useful if it also allows them to make corrections if they detect an error or malicious entry.

The following three phrases are often used synonymously: Cryptographic voting systems, End-to-End voting systems, and Universally Verifiable voting systems. The latter two are subsets of the former; that is, end-to-end voting systems and universally verifiable voting systems are distinct subsets of cryptographic voting systems. Differences in the two include the number of voters, the type of feedback channel, and the VI Phase that is involved.

Universal Verification (UV) provides strong voter integrity, giving the voter procedures that can provide strong confidence that ALL published votes were legally cast and accurately counted. Voting systems that provide UV [4, 6] rely on a public broadcast medium, e.g. a public bulletin board, to broadcast all the voting information necessary to confirm the election results. Usually, that includes the voter rolls and some form of each voted ballot that is cryptographically manipulated to both ensure election integrity and to preserve voter anonymity.

End to End (E2E) voting systems provide voters information that is sufficient to allow them to have a high level of confidence that their own votes are Recorded as Intended (RaI)[1]. This is a weaker standard because voters in E2E systems are not necessarily able to verify all ballots because their proof of inclusion is at the recording, rather than tallying, level.

The E2E feedback channel may be in the form of a text message, e-mail, or some other type of serial communication between the elections office and the voter. Like UV systems, E2E voter feedback may also take the form of a public broadcast medium, such as posting on a bulletin board or webpage, but the broadcast is not necessary to meet the E2E feedback requirement.

Voting client malware is the one of the greatest threats to Internet elections, so being able to ensure that the voting client is not infected can dramatically increase the security of an Internet voting system that leverages remote attestation. One approach to improving confidence in networked applications systems is for nodes to rigorously assess one another to determine whether or not either node is malware infected using is a technique known as *remote attestation*. There is substantial research in literature that details the approaches and technologies that can enable remote attestation with a high level of competence [7, 8, 9, 10]. Once these solutions are fully mature, voting applications may leverage remote attestation to mitigate voting client malware risks.

[1] Many equate E2E systems with those that offer universal verification. In this paper E2E has a slightly different meaning, reflecting transmission from the voter to the elections official but without universal verification.

3. Voting Protocols with Independent Computations

We propose to protect integrity for Remote Electronic Voting by requiring voters to create a signed, electronic version of their ballot that is independent of the voting client. The security of this approach turns on the voter's ability to safely enter the signed ballot into the voting client.

If the voter can generate the signed ballot without the help of a computer, then malware alterations can always be detected. Unfortunately generating a digital signature is a complicated operation that requires automated assistance.

Two of our protocols rely on the voter using two cooperating devices [11, 12]. In our case, the devices are a voting client and an assistive device to compute digital signatures. If the voting client and assistive device are strongly independent, then a voting client malware attack must independently infect both the voting client and the assistive device in order to undetectably alter a voter's ballot, making the attack much more difficult.

3.1. Preliminaries

Voters must first generate a public/private key pair and register the public key with a Certification Authority from which election officials can retrieve a certificate for the voter's public key. For the protocols that use two computing devices, the voter must have a smart assistive device with a camera/scanner, proper computational ability, and an appropriate voting application.

The assistive device holds the voter's private key which must not be accessible to the voting client. The primary computation on the assistive device during voting is to decode a barcode and to compute a hash and signature.

3.2. Device and Malware Independence

The phrase "device independence" can have many connotations. We are concerned about device relationships relative to malware infection, as given in the following definition.

Definition #1. Two electronic computing devices are *malware independent* if and only if, in the threat environment they are both embedded in, the probability that the two devices will be infected by a pair of cooperating malware modules at the time of voting is the product of

the probabilities that either of them will be infected by one of the malware modules separately.

The significance of this independence property is that the protocols that we describe can only be defeated when *both devices* are infected by a pair of malware modules *that were designed to cooperate to subvert the voting process*. If, for example, there is a (10^{-2}) probability that one device is infected with one of a pair of malware modules that can undermine our protocol and a 10^{-3} probability that the other device is infected with another malware module that can cooperate with the first one, then we want to be able to say that there is only a 10^{-5} probability that they are both infected with a cooperating malware pair, and if that is true, then the devices are malware independent.

Of course there are circumstances that can undermine malware independence. If the devices communicate directly with each other before the election and one of them is infected, the malware might be able to pass a cooperating malware infection to the other device, in which case the probability that both are infected can be almost as high as the probability that the first one is infected. If the two devices communicate indirectly with each other, or both communicate with the same third device or server, e.g. by visiting the same web page, they are also less malware independent.

Other factors affecting malware independence include the hardware and software architectures of the two devices. Sharing the same processor, motherboard, or disk drive model reduces independence, as does running the same operating system, device drivers, or other software because it makes it easier for the same malware module [with similar proliferation strategies] to infect both machines.

It is common for smart phone users to connect their phone directly to a laptop, e.g. to synchronize files or download mobile applications. Some smart phones tether to the laptop to provide remote connectivity. Devices that are directly connected dramatically reduce malware independence.

Any connection, wired or wireless, can allow a sophisticated intruder to install cooperating malware on the connected devices. So, in order to maintain the strongest malware independence, one of the two devices would never be network connected. In our protocols, only one message is sent to the assistive device and this as its last protocol action. This minimizes the amount of connectivity, and optimizes the devices' malware independence.

3.3. Digitizing the Ballot

Our protocols leverage properties of a ballot's binary representation and there are many ways to digitize a ballot. For our purposes, it is beneficial to have a representation that minimizes the ballot size and that the voters can compute themselves.

In the sample ballot shown in Figure 1, the fourth row represents a voter's selections reflecting the traditional 'x' in the box. The fifth row is the translation of the votes into their binary representation. Of course binary representation is not intuitive, or convenient, for voters. By partitioning the digital ballot into six-bit groups, we can translate the selections into an alphanumeric form a base-64 representation using digits 1-0, letters a-z and A-Z, and special symbols '@' and '*' to reflect the base-64 values. In Figure 1, the character string "kAo" represents selection of Hunt, Arthur, Snow, Went, Beck, and Good and no others.

Federal Contests									State Contests							
President			US Senator		US Congress				State Senator			State Representative		State Attorney General			
Doran	Hunt	Katz	Arthur	Ford	Mack	Snow	Clay	Jeff	Went	Rick	Tripp	Smith	Beck	Good	Farmer	Clark	Davis
	x		x			x			x				x	X			
0	1	0	1	0	0	1	0	0	1	0	0	0	1	1	0	0	0
k						A						o					

Figure 1. Sample Digital Ballot

We do not suggest this as a regular, general election voting approach. However, we argue that it is not unreasonable for certain constituencies, such as the U. S. military and federal service employees serving overseas, to be able to enter their votes using a ballot constructed as shown in Figure 1 and a conversion table as shown in Figure 2 to cast their ballot in base-64 format.

000000	0	001000	8	010000	g	011000	o	100000	w	101000	E	110000	M	111000	U
000001	1	001001	9	010001	h	011001	p	100001	x	101001	F	110001	N	111001	V
000010	2	001010	a	010010	i	011010	q	100010	y	101010	G	110010	O	111010	W
000011	3	001011	b	010011	j	011011	r	100011	z	101011	H	110011	P	111011	X
000100	4	001100	c	010100	k	011100	s	100100	A	101100	I	110100	Q	111100	Y
000101	5	001101	d	010101	l	011101	t	100101	B	101101	J	110101	R	111101	Z
000110	6	001110	e	010110	m	011110	u	100110	C	101110	K	110110	S	111110	*
000111	7	001111	f	010111	n	011111	v	100111	D	101111	L	110111	T	111111	@

Figure 2. Binary to Base 64 Conversion Table

3.4. Quick Response (QR) Code Technology

In simplified terms, a QR Code™ is a high capacity barcode definition[2]. QR codes store data in images that can be captured via a camera or scanner and translated with image-interpreting software.

Reading a QR Code from the screen of one device through the camera of another device offers several positive security properties. The communication is short range, with no repeaters, amplifiers, switches, routers, or other devices, and no software required between sender and receiver. Unlike electronic communications media, the "sender" is passive; it is the "receiver" that performs the active role. No network is necessary, no MAC address, IP address, phone number, or Bluetooth addresses are needed. The receiving device need not be discoverable or detectable other than to the intended sender.

4. Voting-Client-Malware Safe Voting Protocols

4.1. A Two-Pass Protocol

The *de facto* standard remote electronic voting configuration is for the voting client to reside on a classic networked workstation, such as a desktop or laptop computer, which provides a full suite of user interface tools. Most importantly, the workstation model provides a full screen display to allow the voter to effectively understand their options and accurately capture their intended selections. We refer to this workstation as the "voting client".

For our protocols, the voter selects a second, independent device as described above, to generate a computation that cannot be spoofed by the voting client. An obvious selection would be for the voter to use their smartphone or personal digital assistant for that purpose. We call this *the assistive device*.

Once the voting client and assistive device software and the other prerequisites are met, the voter attains a blank electronic ballot. The blank ballot may be delivered via electronic network or out of band as long as the ballot is delivered safely and does not compromise the cooperative voting devices' malware independence.

With the proper blank ballot loaded the voting protocol proceeds as follows:

1 The voter enters his or her selections on the voting client.

2 The voting client translates those selections into a digital ballot representation and presents it on its display screen as a QR code to the voter.

3 The voter scans the QR code, containing the voted ballot, with the assistive device.

4 The assistive device presents the voter's choices on its display for verification.

5 On voter approval, the assistive device generates a hash of the ballot representation, signed with the voter's private key.

6 The voter scans the signed hash into the voting client via QR code generated by the assistive device.

7 The voting client returns the signed hash along with the voter's ballot to the voting server.

8 The voting server calculates the hash of the ballot and compares it to the hash that it received.

9 If the hashes match, the voting server sends a success message to the voter on both devices, confirming his or her selections.

10 If the hashes do not match, the voting server refuses the ballot and sends a failure message notifying the voter of the problem.

The concept is straightforward, with the voter entering their selections into the voting client, transferring them to the assistive device via QR code signing the ballot on the assistive device, and then transferring the signature back to the voting client. The voting client then submits the digital envelope, containing ballot and signature, to the voting server where the signature is verified. When the votes are transferred between the assistive devices, they are presented to the voter for verification.

This protocol defends against the following three possible malware attacks:

a. Ballot manipulation on an infected voting client. Since the voting client does not have access to the voter's private key, the voting server will detect any ballot manipulation through malware on the voting client, reject the ballot, and notify the voter.

b. Ballot manipulation on an infected assistive device. The voting client submits the original ballot that the voter entered into the voting client. The assistive device has no access to, thus cannot manipulate, the ballot. The most that a malware-infected assistive device can do is to generate a false signature, which would be detected by the voting server and reported to the voter.

c. Denial of Service. The voter can detect a denial of service attack by either the voting client or the assistive device by noticing that the success notice

[2] www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=30789

does not arrive on both devices (specifically, it will not arrive on the uninfected device).

4.2. A One-Pass Protocol

In electronic communications protocols, every message transmission introduces vulnerability. If voters are willing and capable of making their selections on their assistive device, they may reduce the number of communications between the cooperative voting devices using the following steps.

1 The voter enters his or her selections on their assistive device.

2 The assistive device translates the voter's selections into a binary ballot representation, generates a hash of the ballot signed with the voter's private key, and presents it on its display screen as a QR code to the voter.

3 The voter scans the QR code containing the digital envelope containing the voted ballot and signature into the voting client, which decodes the ballot and presents the ballot choices to the voter.

4 The voter verifies his or her selections on the voting client and authorizes the voting client to return the signed hash with the voter's ballot to the voting server.

5 The voting server generates the same hash, decrypts the voter-provided hash using the voter's public key, and compares the two.

6 If the hashes match, the voting server sends a success message to the voter on both devices, confirming his or her selections.

7 If the hashes do not match, the voting server refuses the ballot and sends a failure message to the voter, notifying them of the problem.

The security properties of this protocol are similar to the previous protocol, except that in this protocol, there is only one electronic message between the cooperative voting devices.

4.3. Independent Computation With No Device-to-Device Communication

Two vulnerable components of the protocol given in Section 4.2 are: (1) The image processing software in the voting client (as described above in Section 3.4) and (2) The scanning device itself, which contains sensitive components. In addition, the PC-connected scanners that are needed to collect barcode messages at the voting client are in declining demand.

In the following protocol, the voter casts their ballot on the assistive device, but there is no "transmission" to the voting client. Rather, the voter enters an alphanumeric string that represents their encrypted ballot directly into the voting client.

An additional prerequisite to this protocol is for the voting client to hold a valid public key certificate for the voter.

1. The voter enters his or her selections on their assistive device.
2. The assistive device:
   a. Translates the voter selections into a binary ballot representation
   b. Generates a hash of the binary ballot signed with the voter's private key
   c. Appends the signed hash to the binary ballot to form the digital vote envelope.
   d. Translates the digital vote envelope into a base-64 format and
   e. Presents the vote envelope on its screen to the voter as a base-64, alphanumeric string.
3. The voter enters the base-64 digital vote envelope into the voting client via the voting client keypad.
4. The voting client verifies the hash using the voter's Public Key and displays the voter's selections on its display screen.
5. The voter verifies their selections and authorizes the voting client to deliver the digital vote envelope to the voting server.
6. The voting server generates the same hash, decrypts the voter-provided hash using the voter's public key, and compares the two.
7. If the hashes match, the voting server sends a success message to the voter on both devices, confirming their selections.
8. If the hashes do not match, the voting server refuses the ballot and sends a failure message to the voter, notifying them of the problem.

The security properties of this protocol are similar to the previous protocols, however, in this protocol, there are no electronic messages between the cooperative voting devices. The user enters their choices on the assistive device and manually transfers the ballot and signature to the voting client, where the voter verifies their votes.

5. Experimental Results

A team implemented the two-pass protocol in a system that included necessary elements of the voting server, voting client, and assistive device [13]. The project demonstrated the efficacy of the protocol that captured voter selections, exercised hashing and public cryptography to protect ballot integrity, passive communication capabilities to transfer data between the devices, and return messaging to allow voter verification.

QR Code software is openly available and the system effectively encoded, transferred, and delivered the ballots using QR Codes to communicate between the voting client and the assistive device. The implemented system meets the design functionality and demonstrated that the protocol is practical in prototype.

6. Security Review

We simplified these protocols to focus on the power of independent computations to protect against voting client malware. We do not claim that these protocols provide comprehensive security.

In this section, we discuss our protocols security strengths and weaknesses. We also describe the security properties that these protocols alone do not improve over existing remote voting methods.

6.1. Voting Client Malware

The goal of these protocols is to protect the integrity of cast ballots against voting client malware attacks. Independent computations in each protocol accomplish strengthened malware protection by isolating the voter's private key to protect it from compromise. In the second and third protocols the assistive device application does not receive any data other than through the keypad.

Consider the probability that any arbitrary Voting Client 'a' (vca) is infected with a specific Voting Malware version 'b' (vmb). Ifthat probability is non-absolute then:

$$0 <= P(vca, vmb) <= 1$$

This probability is difficult to assess, but is certainly dependent on the protective measures taken by the device and network administrators. If the probability is low, it complicates the attacker's job and reduces the possible impact that an attacker could have.

6.2. Computation Independence Attacks

As we noted above, in order for protocols 4.1 and 4.2 to conclusively prevent malware attacks on the voting client, the two computers used to conduct computations must be independent. Network-based software applications offer opportunity for sophisticated intruders to corrupt different devices with cooperating malware that could defeat our protocols. However, our protocol complicates the attacker's job in several ways.

First, the attacker must have cooperating malware versions that match the voting client and the assistive device that the targeted voter uses. Second, if the voting client and assistive devices are never connected, the attacker must infect the two devices independently, in which case, using the notation above, the Probability of a Successful Attack is:

$$PSA = P(vca, vmb) * P(ada', vmb')$$

Where ada' is the assistive device that matches voting client 'a' and vmb' is voting malware that can collaborate with vmb and is able to attack assistive device b.

On the other hand, like any other data transfer protocol, barcodes offer an avenue for intruders to introduce malware. That is, if there are software flaws in the barcode interpreter, an intruder might be able to construct a barcode that can inject malware into the interpreting device. Protocols 4.1 and 4.2 may be susceptible to barcode malware attacks.

6.3. Voting Server Malware

The protocols that we present are designed to prevent malware attacks on the voting client, but they are not intended to prevent attacks that install malware on the voting server. Our protection can ensure that the voter's ballot is cast to the voting client as intended and that an honest voting server can identify and refuse to accept a manipulated ballot.

Our protocols provide only "Recorded as Intended", not "Tallied as Intended", confidence. An infected voting server controls the interactions with the voter so could interact inappropriately with the voter (i.e. provide the results as the voter expects), but could tabulate maliciously without the voter being able to detect the changes.

REV systems that utilize our protocols must implement other protections against voting server malware attacks.

6.4. Cryptographic Key Protection Vulnerabilities

Like most schemes that depend on cryptography, key management is critical to our protocol's success. If the voter's secret key is divulged to a malicious intruder, that malicious intruder could masquerade as the voter.

6.5. Receipt-Freeness

Our protocols are receipt free-neutral. That is, none of the three protocols that we present address the issue of receipt-freeness or coercion resistance [14]. Because the voter is unsupervised, similarly to vote-by-mail, voters could demonstrate to a third party how they vote.

On the other hand, our protocols are simply designed for delivering verifiable results from the voter to the voting server and are in no way inconsistent with methods for preventing vote buying and voter coercion. So, coercion resistance could be handled via other mechanisms, many of which are in the literature, see e.g. [15, 16, 17, 18].

6.6. Voter Privacy

Similar to vote-by-mail, our protocols do not protect voter privacy. The voting application would need to incorporate standard network encryption to prevent transmission eavesdropping and elections officials would need to incorporate rigorous application operation procedures to ensure that voter privacy is not compromised.

Additionally, if the voting client is infected with malware, that malware can send a copy of the voted ballot, with voter identification, to a third party. Again, receipt-freeness and coercion resistance techniques could mitigate this effect.

6.7. Denial of Service.

As we noted above, the protocols can detect denial of service through a feedback loop. If notification is sent to both devices, neither can be used to independently accomplish undetectable denial of service.

7. Conclusions and Future Work

Malware on the voting client is one of the most challenging problems to overcome in remote electronic voting. Cryptographic voting protocols have attempted to provide systems that can overcome malware attacks by allowing voters to verify that their votes were Tallied as Intended independent of the voting platform.

Our approach provides *Recorded as Intended* confidence even in the face of malware infection. We offer three voting protocols that leverage *independent computations* to prevent acceptance of any ballot on which malware on the voting client has altered the voter's selections. These protocols are simple in design and rely on voters using two independent devices to cast their ballot.

By leveraging *malware independence* we ensure that the difficulty of malware infestation is factored across the two platforms. We also leverage the positive security properties of barcode transmission to reduce the likelihood of malware transfer between the voting devices and offer one protocol in which no electronic communication between the devices is necessary.

Because of the properties of our protocols, specifically the properties of the voting devices, this protocol may be best suited to military voters, where both of the voting devices may be government issued and professionally maintained.

In order to move these protocols to the general voting public, it may be necessary to incorporate a third voting device that is never network connected, but that only communicates via keyboard and QR codes. This research is ongoing.

In this paper, we introduced the concepts of *independent computations* and *malware independence* and leverage the positive security properties of QR Codes™ for safe device-to-device communication. We propose three protocols that reduce the prospective impact that a malware attack on either the voting client or the assistive device can have.

8. Acknowledgments

Many thanks to David Jefferson for many important suggestions and to Matt Bishop and Paul Syverson for their helpful comments on an early version of this paper. We also acknowledge the contributions of the University of South Alabama students that implemented this system (Erin Pettis, Naquita Hunter, Son Le, and Mengchu Lin) and their mentor, Terri Gilbert.

9. Bibliography

[1] David Jefferson, Aviel D. Rubin, Barbara Simons, and David Wagner, "A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE)," January 20, 2004, http://www.servesecurityreport.org/

[2] David Jefferson, Aviel D. Rubin, Barbara Simons, and David Wagner, "Analyzing Internet Voting Security," Communications of the ACM, October 2004, Vol. 47, No. 10, pp. 59-64

[3] Chaum, D., "Untraceable electronic mail, return addresses and digital pseudonyms," Comm. of the ACM, v.24, n.2, pp. 84-88, 1981

[4] J. D. Cohen(Benaloh) and M. J. Fischer, "A robust and verifiable cryptographically secure election scheme," In FOCS '85, pp. 372–382, 1985

[5] K. Sako and J. Kilian, "Secure voting using partially compatible homomorphisms," In Proc. of International Cryptology Conference (CRYPTO), pages 411–424, Aug. 1994

[6] Ben Adida, "Helios: Web-based Open-Audit Voting," Proceedings of the Seventeenth Usenix Security Symposium (USENIX Security 2008)

[7] George Coker, Joshua Guttman, Peter Loscocco, Amy Herzog, Jonathan Millen, Brian O'Hanlon, John Ramsdell, Ariel Segall, Justin Sheehy and Brian Sniffenm, "Principles of Remote Attestation", International Journal of Information Security, Volume 10, Number 2, 63-81, DOI: 10.1007/s10207-011-0124-7, from the issue entitled "Special Issue:10th International Conference on Information and Communications Security (ICICS)"

[8] Alec Yasinsac, "Identification: Remote Attestation", Chapter 11 in Wireless Sensor Network Security, Cryptology & Information Security

Series (CIS), IOS Press, November, 2007, ISBN: 978-1-58603-813-7

[9] Arvind Seshadri, Mark Luk, Adrian Perrig, Leendert van Doorn, Pradeep Khoslak, “SCUBA: Secure Code Update By Attestation in Sensor Networks,” ACM Workshop on Wireless Security, September 29, 2006, Los Angeles, California, USA, pp. 85-94

[10] E. Shi, A. Perrig, and L. van Doorn, “Bind: A fine-grained attestation service for secure distributed systems” In IEEE Symposium on Security and Privacy, 2005

[11] Alec Yasinsac and Matt Bishop, “The Dynamics of Counting and Recounting Votes”, IEEE Security and Privacy Magazine, May-June 2008, Volume: 6, Issue: 3, pp. 22-29

[12] Bruck, S., Jefferson, D., Rivest, R., "A Modular Voting Architecture (Frog Voting)" in Towards Trustworthy Elections, Chaum, D., Jakobsson, M., Rivest, R., Ryan, P., Benaloh, J., Kutylowski, M., Adida, B. (Eds)., Lecture Notes in Computer Science (LNCS 6000), Springer, 2010

[13] Erin Pettis, Naquita Hunter, Mengchu Lin, and Son Le, “Military Remote Electronic Voting Protocol”, Senior Project, School of Computing, University of South Alabama, Dec 6, 2012

[14] Stephanie Delaune, Steve Kremer, and Mark Ryan, “Coercion-resistance and receipt-freeness in electronic voting,” In Proc. of IEEE Computer Security Foundations Workshop, pp. 28–42, July 2006

[15] Martin Hirt and Kazue Sako, “Efficient receipt-free voting based on homomorphic encryption,” In Proc. of International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), pp. 539–556, May 2000

[16] A. Juels, D. Catalano, and M. Jakobsson, “Coercion-resistant electronic elections,” In WPES ’05, pages 61–70, 2005

[17] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers “Civitas: Toward a Secure Voting System”, In Proc. IEEE Symposium on Security and Privacy, pages 354-368, May 2008

[18] R. Küsters and T. Truderung, ”An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols,” In 2009 IEEE Symposium on Security and Privacy (S&P 2009), pp. 251–266, IEEE Computer Society, 2009