Making Code Voting Secure against Insider Threats using Unconditionally Secure MIX Schemes and Human PSMT Protocols

Yvo Desmedt$^{1,2}$ * and Stelios Erotokritou$^{1,3}$

$^1$ Department of Computer Science, The University of Texas at Dallas, US
$^2$ Department of Computer Science, University College London, UK
$^3$ CaSToRC, The Cyprus Institute, Nicosia, Cyprus

{y.desmedt,s.erotokritou}@cs.ucl.ac.uk

Abstract. Code voting was introduced by Chaum as a solution for using a possibly infected-by-malware device to cast a vote in an electronic voting application. Chaum’s work on code voting assumed voting codes are physically delivered to voters using the mail system, implicitly requiring to trust the mail system. This is not necessarily a valid assumption to make - especially if the mail system cannot be trusted. When conspiring with the recipient of the cast ballots, privacy is broken.

It is clear to the public that when it comes to privacy, computers and “secure” communication over the Internet cannot fully be trusted. This emphasizes the importance of using: (1) Unconditional security for secure network communication. (2) Reduce reliance on untrusted computers.

In this paper we explore how to remove the mail system trust assumption in code voting. We use PSMT protocols (SCN 2012) where with the help of visual aids, humans can carry out mod 10 addition correctly with a 99% degree of accuracy. We introduce an unconditionally secure MIX based on the combinatorics of set systems.

Given that end users of our proposed voting scheme construction are humans we cannot use classical Secure Multi Party Computation protocols. Our solutions are for both single and multi-seat elections achieving:

i) An anonymous and perfectly secure communication network secure against a t-bounded passive adversary used to deliver voting,

ii) The end step of the protocol can be handled by a human to evade the threat of malware.

We do not focus on active adversaries.

Keywords: Voting Systems, Internet Voting, Information Theoretic Anonymity, Private and Secure Message Transmission, Computer System Diversity.

1 Introduction

Electronic voting over the Internet enables to cast votes from an Internet-connected device from any physical Internet accessible location compared

*A part of this work was done while being, part time, at RCIS/AIST, Japan.
to booth based electronic voting systems developed by the cryptographic community [10][22]. Internet voting does not require voters to be physically present at a polling station.

Even though secure Internet voting is in its infancy, many countries and organizations are considering adoption or have already done so such as Estonia [33] and Switzerland [2]. In Estonia, participation increased by 17% [34]. Similarly, after IACR used the Helios Internet voting system [29] which allowed its member’s who are based in different geographical locations to cast their secure vote online, voting increased from 20% to around 30%-40%.

Experts agree that achieving secure Internet voting will be even more difficult than booth-based electronic voting [26]. For example, the 2003 CRA Grand Research Challenges Workshop on Information Security [1] ranked secure Internet voting as one of the most challenging open problems in information security. These issues were put in the spotlight at the 2013 RSA Conference panel [40] and by Rivest in [37]. The difficulties lie in the fact that computational devices are vulnerable to security attacks and are easy to hack. Although SSL uses cryptography, modern browsers are vulnerable to attacks such as click-jacking, cross-site scripting, and man-in-the-browser attacks - as demonstrated against Helios 2.0 in [21].

Given that the computer of a voter can easily be hacked, in 2001 Chaum proposed a breakthrough solution called “code voting” [7] where one can use a possibly hacked computer to perform a secure operation. In code voting, a voter receives through the postal mail a long enough unique code for every candidate. To vote, voters would just enter the code corresponding to the candidate of their choice.

Chaum’s approach to code voting assumes the postal mail to be secure from a reliability and privacy viewpoint. This is not a valid assumption to make. Indeed, a collaboration of the postal service with the returning officer may allow for the anonymity of all votes to be broken by divulging the identity of voters to whom specific voting codes were delivered. Another problem is that if one knows who is likely not to vote, Chaum’s scheme is not very secure against ballot stuffing. Furthermore, if malicious postmen do not deliver voting codes, this prevents voters from casting their votes. If the election is tight and the number of undelivered ballots is high, this could undermine the reliability and trustworthiness of code voting through the postal service. So, one question we address is how we can make Chaum’s code voting secure against passive insiders.

---

4 A returning officer oversees elections in one or more constituencies [44].
5 Since we focus on a passive adversary, our paper does not address this attack.
Obviously we need to maintain the anonymity of voters. One way to achieve anonymity is through the use of MIX-networks. These were first introduced by Chaum in [8] and are used in electronic voting. MIX-networks allow senders to input a number of (usually encrypted) messages to a MIX-network which then outputs and delivers each message to all recipients without the receiver being able to identify the sender. Various ways with which MIX-networks are constructed are described in Section 2.1. The main issue with such constructions is that they are based on tools based on computational assumptions which when used within the context of an electronic voting scheme only allows for conditional security thus conditional privacy and conditional anonymity to be achieved.

Note that no conditional secure cryptosystem designed so far has withstood cryptanalysis for more than 300 years. Quantum computers will undermine computational voting schemes cryptographers have proposed, in particular these based on ElGamal. For many goals, unconditionally secure solutions have already been proposed, e.g., since 1988 [49] we have unconditionally secure multiparty computation. This is a further motivation for proposing an unconditionally secure voting scheme in which $t$ insiders can be corrupted. Due to the revelations by Snowden [14], some have questioned the security of the NIST standards [42]. So, one can wonder whether we want to promote voting systems which might be broken, if not now, then in the future. The importance of requiring unconditional vote security is further highlighted with the following example:

In 2020 Alice turns 18 and votes using a popular ElGamal based electronic voting scheme. 50 years later, Alice is a candidate for president of the USA. Imagine that in 2070 USA politics is going through a new McCarthy [41] witch hunt. Unfortunately for Alice, ElGamal security has since been broken. The newspapers find that Alice voted for the what is then considered the “wrong” party!

In this paper we focus on unconditional security proposing alternative MIX constructions (using set systems and shares of messages), to generate the correctness of the vote unconditionally. To counter technological threats and the possible influence of elections by foreign governments (where hardware are manufactured), our proposed Internet code voting solution uses the concept of diversity, first described in [23]. So, we employ a diversity of computing systems to achieve security in our proposed solution. Using diversity of network paths we also ensure that any $t$-bounded adversarial presence is unable to break the privacy of any votes. We consider the $t$-bounded computationally unlimited adversary to be capable
of taking control of any node between the vote authority and the voters which includes nodes in the MIX-network, nodes in the communication network or voters computational devices (through malware). It should be noted that we do not consider the human voters to be corruptible.

The main part of our work assumes a passive adversary which can only observe but cannot cause deviation of protocol execution in any way. We also assumes that the adversary cannot look at the information on the whole network but only inside \( t \) nodes. Our solution considering an active adversary will be presented in a future full version of this paper.

Considering a \( t \)-bounded adversary we emphasise the following:

**Important Statement 1** As shown in [24], when the number of corrupted nodes is at most \( t \), the minimum number of disjoint paths to allow for private communication between a sender and a receiver is \( t + 1 \).

**Corollary 1** Because of the above, voters will have to use a number of computing devices to securely receive (or dually send) their voting codes.

The impact of Corollary 1 is not as bad as it might initially seem. Nowadays, many people in developed countries can have effortless access to more than one device such as PC’s, laptops, smartphones and tablets. Such devices can include those they own or can access through friends and relatives or through public access (such as a library). Furthermore, each of these devices can be connected to a communication network in a different manner (Internet or cellular) which could be serviced by different providers. These devices may run different operating systems (e.g. Windows, iOS, Android) thus a threat to one device may not necessarily constitute a threat to another - even with the same user.

Similar to the work of [5,30] which considers security protocols as used by humans who can execute them without relying on a fully trusted computer we do the same in this paper in the context of Internet voting.

Motivated by all the above, we propose an unconditional Internet code voting protocol which is secure against the possible presence of an adversary and malware in the network and on voter’s devices respectively. We present solutions for single seat and multi-seat elections both of which are designed to be user friendly - so that human voters can use it correctly with high accuracy.\(^6\) In EVOTE2014 [36] the authors addressed a very

\(^6\) It should be noted that the main goal of our work is Internet code-voting secure against \( t \) insiders. The work of [6] is independent and their MIX servers are different using a homomorphic, unconditionally hiding commitment scheme to encrypt audit information and achieve unconditional security. Furthermore, their solution assumes
similar problem as our current work. However, their solution achieves conditional security which could be broken in the future against a computationally unlimited adversary. Furthermore, the authors consider the adversary to be present in the MIX network only and do not take into account the possible presence of malware upon the tablets with which voters will use to cast their votes. Passive malware could possibly identify to an adversary how someone voted, whereas active malware could alter the way someone votes - thus rigging the result of an election.

When combined with [20], one can view our proposed method for delivering codes to voters as a distributed implementation of a one-time-pad-secured communication channel for votes. Because of this, our solution can also be used for other established code voting schemes as it is a way of removing the use of a possibly untrusted mail system and transmitting the voting codes securely, reliably and anonymously to voters.

The text is organized as follows. Background and relevant previous work are presented in Section 2. In Section 3 a high level description of the protocol is given and we identify the required cryptographic tools. In Section 4 we provide a simplified version of the MIX private and anonymous communication protocol. This is used in Section 5 in a more efficient manner where we present private and anonymous communication protocols for the transmission of voting codes to voters for single seat and multi seat elections. In Section 6 the electronic code voting protocol is presented and the security proof of the protocol is also given.

2 Background and Previous Work

2.1 Previous Work

This section describes previous work related to various aspects to be presented in this paper.

MIX-networks can be constructed using a shuffle (permutation). One way of achieving this [32,39] is by using approaches which are based on zero-knowledge arguments [25,45]. In [17] the use of zero-knowledge was avoided. MIX-networks based on zero-knowledge arguments can be used the use of two mix-networks one of which is private and thus cannot be corrupted by the adversary. Our solution does not make this assumption and instead counters the threat of the adversary presence for protocol correctness accordingly. However, due to the possible presence of malware the only way we know how to achieve this, is using unconditionally secure techniques achieved through the use of cover designs. Additionally we use results from previous work [20] which allows for humans to privately and reliably receive and decode messages, something achieved with unconditional security.
in electronic voting protocols - as has been proposed in recent publications [27,28]. Earlier work [35] similarly used shuffles in electronic voting based on zero-knowledge proofs. Other work on MIX-networks includes the work of Abe in [3].

Such constructions are based on computational assumptions which only allow for conditional security. The work we present is based on the stronger model of unconditional security.

Anonymity in practice is difficult to achieve. One proposed implementation was that of [31] but it was shown to be insecure in [43].

A voting scheme similar to the one we propose which achieves information theoretic security and requires the voter to carry out modular addition is that presented in [35]. Contrary to the voting scheme proposed in this paper, the work of [35] is not an Internet voting scheme as it requires voters to cast their votes at a polling station.

The work of [13] describes an election scheme which requires computational modular exponentiation operations to be carried out by voters. These operations require software or hardware. Furthermore, public key-cryptography is used, meaning that the security properties achieved are computational and not information theoretic - as achieved in our scheme.

### 2.2 Message Transmission Security Properties

Below we define message transmission security properties which will be required throughout the text. For formal definitions, see [19]. In our setting we have a single receiver $S$ connected to $m$ number of senders $(r_1, \cdots, r_m)$ over a possibly corrupt underlying network.

**(Perfectly) Correct** - When the receiver accepts a message, it was sent by a sender $S$.

**(Perfectly) Reliable** - When a sender $S$ transmits a message, this message will be received by the receiver with probability 1.

**(Perfectly) Private** - Only the designated receiver(s) can read a message transmitted by $S$. i.e., for any coalition of $t$ parties, their probability of correctly determining a message is the same whether the coalition is given their transmission view or not.

**(Perfect) Security** - Means perfect correctness, perfect reliability and perfect privacy.

**(Perfectly) Anonymous** - Considering the single receiver wants to receive $m$ different messages over the network so that each of $m$ number of senders transmitted one of these messages (and each message is transmitted and received only once), perfect anonymity is achieved when
for any coalition of $t$ parties, their probability of correctly determining
the sender of any message is the same whether the coalition observes
the transmission view or not. In the context of Internet voting, perfect
anonymity is achieved when the voting protocol used does not facilitate
any party involved in the voting process to correlate any cast vote to a
specific voter with greater probability than any other.

2.3 Existential Honesty

Some of our ideas use concepts of existential honesty, defined in [17] as:

“It is possible to divide the MIX servers into blocks, which guar-
antee that one block is free of dishonest MIX servers, assuming
the number of dishonest MIX servers is bounded by $t$.”

To achieve this, [17] defined and used the following (see also [20, Sec-
tion 2.3] for an extensive description of set systems and how these relate
to covering designs.):

**Definition 1 ([12]).** A set system is a pair $(X, \mathcal{B})$, where $X \triangleq \{1, 2, \ldots, m\}$ and $\mathcal{B}$ is a collection of blocks $B_i \subset X$ with $i = 1, 2, \ldots, b$.

**Definition 2 ([17]).** $(X, \mathcal{B})$ is an $(m, b, t)$-verifiers set system if:

1. $|X| = m$,
2. $|B_i| = t + 1$ for $i = 1, 2, \ldots, b$, and
3. for any subset $F \subset X$ with $|F| \leq t$, there exists a $B_i \in \mathcal{B}$ such that $F \cap B_i = \emptyset$.

We assume that private channels connect MIX servers of correspond-
ing blocks (i.e. when for block $B_k$, MIX server $MIX_{k,i}$ needs to commu-
nicate with MIX server $MIX_{k+1,j}$, where $1 \leq i, j \leq t+1$ and $k < b$, then
there is a private channel). We also assume such channels between the
receiver and $MIX_{1,i}$ and similarly, between $MIX_{b,i}$ and the sender.

2.4 Human Perfectly Secure Message Transmission Protocols

Perfectly secure message transmission (PSMT) protocols where the sender
or receiver is a human were introduced in [20]. In such protocols it is as-
sumed that the human receiver does not have access to a trusted device
since these may be faulty and/or infected with malware. Because the re-
ceiver is a human, such protocols aim to achieve perfectly secure message
transmission (PSMT) in a computationally efficient and computationally
simple manner. It is also important that the amount of information and operations the human receiver should process be kept to a minimum.

Addition mod 10 was used by humans in these protocols [20] to reconstruct the secret message of the communication protocol from received shares through addition mod10. The idea of using addition mod10 for human computable functions was also used in [5] but within a different security context. By regarding in [20] $\mathbb{Z}_{10}$ as a subgroup of $S_{10}$ the operation became very reliable for humans to perform. Experiments have shown that given clear, correct and precise instructions, coupled with visual aids, allowed for the correct usage of these protocols by a very high percentage of human participants.

2.5 Secure Multiparty Computation in Black-box Groups

Black box multiparty computation protocols against a passive adversary for non-Abelian group have been presented in [11] and in [16] through the use of a $t$-reliable $n$-coloring admissible planar graph. These papers studied in particular the existence of secure $n$-party protocols to compute the $n$-product function $f_G(x_1, \ldots, x_n) := x_1 \cdot \ldots \cdot x_n$ where each participant is given the private input $x_i$ from some non-Abelian group $G$ where $n \geq 2t + 1$. It was assumed that the parties are only allowed to perform black-box operations in the finite group $G$, i.e., the group operation $((x, y) \mapsto x \cdot y)$, the group inversion $(x \mapsto x^{-1})$ and the uniformly random group sampling $(x \in_R G)$.

3 Secure Code Voting with Distributed Security

In this section we provide a high level description of the secure code voting protocol we will present in this paper. We assume the reader is familiar with Chaum’s code voting scheme [7].

3.1 High Level Description

We call Code Generation Entity (CGE) the entity in the code voting protocol which is responsible for creating the codes with which voters will cast their votes. These codes are unique and are sent to the voters so that each of these codes is used only once for the whole election. For single seat elections each voter receives as many codes as there are candidates. For multi-seat elections each voter receives a single permutation - which is a permutation of the alphabetical ordering of the candidates. After these
codes pass through a MIX network, they will be sent to voters using perfectly secure message transmission, i.e. using secret sharing. Voters will receive each share using a different device, identify the shares which correspond to the candidate of their choice and reconstruct using human computation this voting code. To cast their vote, voters will send this code back to the CGE via the MIX servers, which perform inverse operations. For each of the received cast codes, the CGE will identify the candidate the code corresponds and will tally up the cast votes for each candidate.

Our protocol does not use the mail system for the delivery of voting codes to voters, but instead these are sent by the CGE to voters over a MIX network and using PSMT. Similarly, cast votes will be sent by voters to the CGE over a network as explained in Section 6.3.

3.2 Required Cryptographic Tools

The process should not facilitate the CGE (and indeed any $t$ other parties) should not be able to identify that a specific voter (from the set of $v$ voters) cast a particular vote. Furthermore, a number of the underlying network nodes may be corrupt. Even though secret sharing is used, any protocol should ensure that voting codes are not learned by any $t$ parties apart from voters themselves, otherwise anonymity of votes could be broken.

Human perfectly secure message transmission protocols as presented in [20] are employed. We rely on the feasibility tests performed which confirm that humans can perform these basic operations. We use the secret sharing scheme friendly to humans as presented in [20, Section 2.2] which guarantees perfect privacy unconditionally. Except for the voters computing the codes from the shares they receive, all other computations are carried out by computers, of which no more than $t$ of these are curious.

4 Transmit and Reply Protocol

In this section we present the first of the required primitives - a perfectly private and perfectly anonymous network communication protocol. For didactic purposes, the simplest form of our proposed protocol will be presented - with more efficient constructions described later.

Suppose that we have a single receiver and $v$ senders each of whom needs to receive a secret one time pad so as to sender a secret back to the receiver in an interactive anonymous way.

\footnote{The dual problem is that instead of having $v$ senders, we have $v$ receivers and one sender. Obviously a solution for the first provides a similar solution for the second.}
We assume the passive adversary controls at most \( t \) MIX servers. As in Chaum’s work \([8]\) and most conditional MIX servers, each MIX server is involved in one mixing in our protocol. \( t + 1 \) blocks of MIX servers will be required - denoted as \( B_1, \ldots, B_{t+1} \), with each block consisting of \( t + 1 \) MIX servers and we use \( B_k = \{ MIX_{k,1}, MIX_{k,2}, \ldots, MIX_{k,t+1} \} \) to identify MIX servers of the \( k^{th} \) block and call \( MIX_{k,1} \) “\( B_k \)’s leader”.

### 4.1 Protocol Main Idea

The receiver will share each of the \( v \) one-time pads to transmit into \( t + 1 \) shares using XOR. Each (of the \( t + 1 \)) share will be given to a corresponding MIX server (i.e. one of the \( t + 1 \) servers) in the first block \( B_1 \).

The shares of the \( i^{th} \) one-time pad and those of the \( j^{th} \) one-time pad might be transposed and will also be altered. To guarantee shares of the same pad stay together, the transpositions and alterations are chosen by the block leader. After the last MIX operation, the final block of MIX servers delivers the shares of the one time pad to the senders, with each sender reconstructing the received and altered one-time pad sent by the receiver. Each sender will then XOR the secret message to be sent to the receiver with the received altered one-time pad and send the result to the receiver over the MIX network. During this reverse transmission, the inverse alterations will be applied by each block leader.

By XOR’ing the one time pad initially sent out by the receiver, the secret message sent by each sender can be obtained by the receiver.

### 4.2 The MIX Communication Protocol - 1A: Receiver to Sender Transmission

We now present the steps in the MIX communication protocol for the transmission of the one-time pads from the receiver to the set of senders.

**Protocol 1 Private and Anonymous Communication Protocol**

**Step 1** Let \( \pi^1_i \) be the \( i^{th} \) one-time pad (where \( 1 \leq i \leq v \)). The receiver shares each \( \pi^1_i \) into \( t + 1 \) shares \( \pi^1_{i,j} \in F_2^t \) using XOR (where \( 1 \leq j \leq t + 1 \)) and privately sends \( \pi^1_{i,j} \) to the corresponding MIX \( MIX_{1,j} \) in block \( B_1 \).

**Step 2** The leader of \( B_1 \) (we call \( MIX_{1,1} \)) informs all others MIX servers in \( B_1 \) how they have to permute the \( i \)-index of all above \( \pi^1_{i,j} \). This permutation is defined by \( \rho_1 \in_R S_v \).
Step 3 On the $i$ indices all MIX servers in $B_1$ apply the permutation $\rho_1$. So, $\pi_{i,j} := \pi_{\rho_1(i),j}^1$.

Step 4 The leader of $B_1$ chooses $t+1$ random bit modifiers $\omega_{i,j}^1 \in_R F_2$ and privately sends $\omega_{i,j}^1$ to parties in $B_1$.

Step 5 For each $(i,j)$ the $t+1$ values $\pi_{i,j}^1$ are regarded as shares of $\pi_i^1$. Similarly, the $t+1$ values $\omega_{i,j}^1$ are regarded as shares of $\omega_i^1$. The MIX server in $B_1$ computes $\pi_{i,j}^2 = \omega_{i,j}^1 + \pi_{i,j}^1$. $\pi_{i,j}^2$ are regarded as shares of $\pi^2$, the $\rho_1(i)$ permuted and modified one time pad.

Step 6 Steps 2-5 are repeated, incrementing by one the indices of $B_1$ and $B_2$ until the last block $B_b$ is reached.

Step 7 Shares held by MIX-servers of block $B_{t+1}$ are denoted as $\phi_{i,j}$. MIX$_{t+1,j} \in B_{t+1}$ then sends $\phi_{i,j}$ to the $i^{th}$ sender.

4.3 The MIX Communication Protocol - 1B: Sender to Receiver Transmission

Upon the end of the first phase, each sender reconstructs their respective altered one-time pad using XOR over all shares received. Using this altered one-time pad, a sender encrypts their secret using XOR. Senders then proceed to send their encrypted secret to the leader of block $B_{t+1}$. These are then sent back to the receiver in much the same way as transmitted from receiver to sender, only this time, data are sent between leaders of MIX blocks, the inverse permutations will be applied and all modifiers used will now have be invalidated. Thus the leaders of each block of MIX servers will use the inverse permutations $\rho_b^{-1}$ and invalidation of modifiers $-\omega_b^k$ (invalidating using XOR).

The data that are sent back to the receiver correspond to the encrypted message transmitted by senders, and by applying XOR to this using the respective one-time pad, the secret message transmitted by senders can be obtained.

It should be noted, that this anonymous and private communication protocol can be used for various practical applications. One such example is anonymous therapy sessions with extensions of the protocol also allowing for anonymous feedback.

4.4 Security Proof

In this section we present the security proof for Protocol 1.
Theorem 1. Protocol 1 is a reliable, private and anonymous message transmission protocol.

Proof. The protocol achieves perfect reliability due to the passive nature of the adversary. Perfect privacy is achieved as each one-time pad or encrypted message is “shared” over \( t+1 \) shares. As each MIX server is used only once and as the adversary can control at most \( t \) MIX servers, secrecy of these transmitted data is retained. We now prove the perfect anonymity of the protocol - for simplicity of the proof we assume that there are only two messages (two one time pads).

As \( t+1 \) blocks of MIX servers are used and each MIX server is used only once, there exists a block \( B_i \) - \( 1 \leq i \leq b \), free from adversary controlled MIX servers. Because of this, the adversary is unable to learn the modifiers and permutation which are added and implemented respectively to the shares of the messages.

Assuming the adversary is present in \( B_{i+1} \) and absent from \( B_i \), the view of the adversary of a share for both messages can be one of the following two possibilities: \((\omega_1^{i+1}+\pi_1^{i+1-1}, \omega_2^{i+1}+\pi_2^{i+1-1})\), \((\omega_2^{i+1}+\pi_2^{i+1-1}, \omega_1^{i+1}+\pi_1^{i+1-1})\)

Obviously, the adversary cannot distinguish between the first and the second possibility as the modifiers and permutation used in block \( B_i \) are random and not learned by the adversary. Indeed, there exists an \((\omega'_1, \omega'_2)\) such that \((\omega_2^{i+1}+\pi_2^{i+1-1}, \omega_1^{i+1}+\pi_1^{i+1-1})=(\omega'_1+\pi_1^{i+1-1}, \omega'_2+\pi_2^{i+1-1})\). So, the adversary cannot distinguish whether the messages have been interchanged or not.

Without loss of generality, the proof can be extended to any number \( v \) of messages. \( \square \)

5 Reducing the Number of MIX Servers

In this section we improve on the “Transmit and Reply Protocol” presented in Section 4 presenting a solution for the single seat election case where an Abelian group is used.

Our solution uses Chaum’s code voting and considers a single receiver (e.g., CGE) and \( v \) human voters who each need to receive voting codes (one code per candidate) in a non-interactive anonymous way. We consider the CGE as the receiver and the human voters as the senders of the communication because at the end of the combined protocol, the human voters will send back to the CGE the voting code which corresponds to the candidate of their choice. We regard codes intended for the same receiver as a long string and the MIX servers MIX the strings (i.e. those intended for different receivers).
A more efficient network of MIX servers is used as our solution is not confined to using each MIX server only once, thus the total number of MIX operations done is $b$. We denote the set of MIX servers by $X$ and assume we have an $(X, B)$ set system, which is an $(m, b, t)$-verifiers set system as defined in [17]. We let $B_k = \{MIX_{k,1}, MIX_{k,2}, \ldots, MIX_{k,t+1}\}$ and call $MIX_{k,1}$ “$B_k$’s leader”.

The main idea of the protocol is very similar to the communication protocol of the previous section. This time, the receiver (e.g., CGE) will share each of the $v$ messages to transmit using an appropriate secret sharing scheme (and not using XOR). In a similar fashion, messages are permuted and altered as they are transmitted within the MIX network. After the last MIX operation, the final block of MIX servers delivers the shares of messages to the senders, with each sender reconstructing the secrets (voting codes) sent by the receiver. We will assume the transmission of the shares of these secrets uses the human friendly method presented in [20]. Similarly, since a code is only used once, it can be modified using addition over a finite Abelian group. To be compatible with [20] one such example is addition mod 10 over the group used. Senders will then transmit back to the receiver the voting code corresponding to their choice.

5.1 Virtual Directed Acyclic Graphs

When an Abelian group is used and when blocks of the $(m, b, t)$-verifiers set system can share common MIX servers between them, we define the construction of a virtual vertex-labeled Directed Acyclic Graph (DAG). The set of vertices of the DAG is composed of parties participating in the protocol (which is similar to Protocol 3), with the source of the graph being the receiver of the protocol and the sink being a sender.

The directed edges of the DAG identify the transmission of messages from one party to another amongst different levels in the DAG. We define levels of the DAG as the receiver, a sender and the different blocks of MIX servers used. Considering block $B_i$ as a tuple (ordered set), when $B_i$ is a block where $|B_i| = l$ and $b \in B_i$, at location $k$ in this tuple, we say that $b$ is at position $k$. With the above definition, directed edges of the DAG will occur (i) from the receiver to all $b_j$ in $B_1$ ($1 \leq j \leq l$), (ii) from each $b_j$ in block $B_i$ to the sender, (iii) moreover, we have edges between nodes in $B_i$ and nodes in $B_{i+1}$. The following is required:

1. If a unique color was to be assigned to each party of the protocol, based on the results of [18], the sender and receiver can privately
communicate, if when choosing any \( t \) colours and removing the vertices of the DAG with those \( t \) colours the sender and receiver remain connected - meaning that there still exists a directed path from the sender to the receiver on the reduced DAG.

2. We require that if at level \( k \) the parties in \( B_k \) receive shares of \( \pi^k_i \), the parties in \( B_{k+1} \) (i.e., at level \( k + 1 \)) receive shares of \( \pi^{k+1}_i = \omega_i^k + \pi^k_{\rho(i)} \).

Two methods can be used to achieve the above requirements. One uses re-sharing - such as the redistribution scheme described in [15]. The other uses a large set of MIX servers \( X \) to guarantee the following property.

Definition 3. We say that set \( X \) of MIX servers is under \( t \)-confinement if all members of set \( T \) where \( |T| = t \) appear in at most \( t \) positions over all blocks of MIX servers used and this for all \( T \subseteq X \) where \( |T| = t \).

It is easy to see that the above satisfies the DAG requirements.

5.2 The MIX Protocol

In the case of Internet voting this is used as a pre-voting protocol for the transmission of voting codes to voters and it is used to achieve anonymity of voting codes. We assume \( S \) to be a finite Abelian group and denote with \( v \) the number of senders, and thus the number of messages (sets of voting codes) that need to be transmitted. In the following, we only describe the required difference when compared to Protocol 1.

Protocol 2 Private and Anonymous Random Communication Protocol

Step 1 Let \( s_i \) be the \( i \)-th message (where \( 1 \leq i \leq v \)). The sender shares each \( s_i \) by choosing \( l \) shares \( \pi^1_{i,j} \in_R S \) (using an appropriate secret sharing scheme over an Abelian group where \( 1 \leq j \leq l \)) and privately sends \( \pi^1_{i,j} \) to the corresponding party \( B_{1,j} \) in \( B_1 \).

- As an \((m, b, t)\)-verifiers set system is used, \( l = t + 1 \) denotes the number of shares.

Step 2 Same as in Protocol 1

Step 3 Same as in Protocol 1

Step 4 The leader of \( B_1 \) chooses modifiers \( \omega^1_{i,j} \in_R S \) and privately sends \( \omega^1_{i,j} \) to parties in \( B_1 \).

Step 5 Similar as in Protocol 1 Only:

The MIX servers in \( B_1 \) compute shares of \( \pi^2_i = \omega^1_i + \pi^1_i \), i.e. party \( P_j \in B_i \) adds the modifiers it receives from the leader of \( B_i \) to the share(s) it holds. The shares of the \( \pi^2_i \) are denoted as \( \pi^2_{i,j} \).
Step 6 If the concept of $t$-confinement is not used, re-sharing of shares $\pi_{i,j}^2$ is carried out by parties in $B_1$ using the redistribution scheme described in [15]. That means that each party in $B_2$ receives $l = t + 1$ values, which they then compress.

Step 7 Steps 2-5 are repeated incrementing by one the indices of $B_1$ and $B_2$ until the last block $B_b$ is reached. For all iterations - except when the last block $B_b$ is reached, Step 6 is also repeated (except if $t$-confinement is used).

Step 8 If $t$-confinement is not used, shares held by the MIX-servers of block $B_b$ are re-shared.

Step 9 Shares held by MIX-servers of block $B_b$ are denoted as $\phi_{i,j}$. $MIX_{b,j} \in B_b$ then sends $\phi_{i,j}$ to the $i^{th}$ voter using [20].

It should be noted, that as in [20], MIX servers will send shares to voters using network disjoint paths, as the communication network cannot be trusted with the adversary capable of listening to at most $t$ of these paths. The way voters cast their vote will be described in Section 6.

5.3 Security Proof

In this section we present the security proof for Protocol 2.

**Corollary 2** Protocol 2 is a reliable, private and anonymous message transmission protocol.

**Proof.** Formally, we have:

**Perfect Reliability** - This is the same as in Theorem 1.

**Perfect Privacy** - The protocol achieves perfect privacy as each message is “shared” over $l = t + 1$ shares. In the case of $t$-confinement, the view of the adversary will consist of at most $t$ shares. This number is one less that the number required to reconstruct a secret and thus perfect privacy is achieved. In the case of re-sharing, the re-sharing guarantees that shares at level $i$ are independent of those at level $i + 1$ (note that the adversarial parties are passive). The rest follows from [18] and through the use of re-sharing or $t$-confinement. When using re-sharing we ensure that there is no cut of $t$ vertices (colors) that can disconnect the sender and the receiver. This is because the resharing of shares makes certain that the parties in block $b_i$ receive shares from $t + 1$ parties in block $b_{i-1}$. So, any adversarial $t$ parties in block $b_{i-1}$ will not allow to cut the graph. It is easy to see that the condition of [18] (i.e. no $t$ parties are able to cut a graph) is satisfied when using $t$-confinement thus allowing for secure solutions.
**Perfect Anonymity** - This is very similar to the anonymity proof of Theorem 1. The only difference is that now where a lower number of MIX servers are used, due to Property 3 from the definition of verifier set systems, there exists a block $b_i - 1 \leq i \leq b$, free from adversary controlled MIX servers. Because of this, the adversary is unable to learn the modifiers and permutation which are added and implemented respectively to the shares of the messages.

5.4 Use of non-Abelian Group - Multi-seat Election Case

When a non-Abelian group is used, the protocol is similar to that presented in Section 5.2. Due to the non-Abelian nature of the group, alternative additional techniques will have to be employed to manage the fact that dealing with shares cannot be done locally (due to the multiplication) thus this needs to be shared and securely computed among many parties using techniques presented in [16].

Suppose we have an election in which we have $s$ seats in which every voter can vote for up to $s$ of the $c$ candidates - where $s \leq c$. To enable *blinding* of the code, we give to each voter a secret permutation $\pi \in S_c$, where $S_c$ is the symmetric group. For each favourite candidate $i$ the voter wants to vote for, $\pi(i)$ is transmitted to the returning officer.

Note that $\pi$ is *not* necessarily unique to the election, as opposed to Chaum’s code voting. The protocol is organised to avoid that this creates a problem. In the case of Internet voting, the following protocol is used as a pre-voting protocol, for the transmission of $v$ number of voting “codes” (i.e. permutations) to $v$ number of voters and it is used to achieve anonymity of voting codes. We assume $S = S_c$ to be a finite non-Abelian group.

It should be noted that the protocol to be presented is only useful for the private and anonymous transmission of permutations with which receivers can cast their vote.

**Protocol 3 Private and Anonymous Random Communication Protocol**

**Step 1** Same as in Protocol 2 only now a non-Abelian group is used and permutations are transmitted.

**Step 2** The leader of $B_2$ chooses modifiers $\omega_{i,j}^2 \in R S_c^l$ and privately sends $\omega_{i,j}^2$ to parties in $B_2$ such that the $l$ values $\omega_{i,j}^2$ are regarded as shares of $\omega_{i,j}^2$.  

---

8 As shown in [16], to securely compute $\pi$ and $\omega$ where $\pi$ is chosen by one party and $\omega$ by another, we need $2t + 1$ parties where $t$ parties are curious. To mimic as closely
Step 3  For each \((i, j)\) the \(l\) values \(\pi_{i,j}^1\) are regarded as shares of \(\pi_i^1\).
The MIX servers in \(X'_{1,2} \subseteq X\) where \(|X'_{1,2}| \geq 2t + 1\) and \(B_1 \cup B_2 \subseteq X'_{1,2}\) compute shares of \(\pi_i^2 = \omega_i^2 \circ \pi_i^1\) using a black box non-Abelian multiparty computation protocol \(9\) (see Section 2.5).
This is done so that \(\omega_i^2\) blinds \(\pi_i^1\). The shares of the product are denoted as \(\pi_{i,j}^2\) and are obtained by the parties \(10\) in \(B_2\).

Step 4  The leader of \(B_2\) informs all other MIX servers in \(B_2\) how they have to permute the \(i\)-index of all shares they hold from the above operations. This permutation is defined by \(\rho_2 \in R S_v\). On the \(i\) indices the MIX servers in \(B_2\) apply the permutation \(\rho_2\).
So, \(\pi_{i,j}^2 := \pi_{\rho_2(i),j}^2\).

Step 5  The above three steps are repeated by incrementing by one the indices of \(B_1\) and \(B_2\) (thus \(B_k \neq B_{k+1}\)). After parties in \(B_k\) permute the \(i\) indices of \(\pi_{i,j}^k\) using \(\rho_k\)  - where \(2 \leq k \leq b - 1\), the leader of \(B_{k+1}\) chooses modifiers \(\omega_{i,j}^3 \in R S_v^k\) which are given to parties in \(B_k\), the black box non-Abelian multiparty computation sub-protocol is executed by parties in \(X'_{k,k+1} \subseteq X\) where \(B_k \cup B_{k+1} \subseteq X'_{k,k+1}\) \(|X'_{k,k+1}| \geq 2t + 1\) and the process continues till the final block of servers \(B_b\) is reached.

Step 6  After parties in \(B_b\) permute the \(i\) indices of \(\pi_{i,j}^b\) using \(\rho_b\), the leader of \(B_1\) chooses modifiers \(\omega_{i,j}^1 \in R S_v^l\) which are given to parties in \(B_1\), the black box non-Abelian multiparty computation sub-protocol is executed between parties in block \(B_b\) and \(B_1\) and the output of which is held by parties in \(B_1\). \(MIX_{1,j} \in B_1\) sends the output it holds to the \(i^{th}\) voter using \(20\).

It should be noted, that as in \(20\), MIX servers will send shares to voters using network disjoint paths, as the communication network cannot be trusted with the adversary capable of listening to at most \(t\) of these paths. The way voters will use what they receive to cast their vote will be described in Section 6.

We now present the security proof for Protocol 3.

as possible the working of \(10\), \(\omega_{i,j}^2\) is chosen by the leader of \(B_2\) and not by the leader of \(B_1\).

Note that the MIX servers in \(B_1 \cup B_2\) can also be a in \(X'_{1,2}\) where \(|X'_{1,2}| \geq 2t + 1\). Additionally, the efficiency of black box non-Abelian multiparty computation protocols is better when \(|X'_{1,2}| >> 2t + 1\).

Note that \(16\) allows to organise the computation such that the output, i.e. shares of \(\pi_i^1\), are received by parties in \(B_2\).
Theorem 2. Provided Protocol 3 together with the appropriate black box non-Abelian multiparty computation sub-protocol is used, then Protocol 3 is a reliable, private and anonymous random transmission protocol.

The proof of the above theorem is similar to the proof of Theorem 1, but relying on either \[11,16\].

6 Electronic Code Voting Protocol

In this section we outline how components of previous sections are combined.

6.1 Preparation, Mixing and Transmission of Voting Codes

As described in Section 3.1 the CGE is responsible for creating the codes with which voters will cast their votes. We first explain this for the single-seat election.

Considering an election has \(c\) number of candidates and that there are \(v\) number of voters, the CGE will create \(v\) random initial codes for each of the \(c\) candidates. In total, \(c \times v\) unique number of codes will be generated. The CGE will then group these codes to form \(v\) number of \(c - t\)uples, with each tuple containing a single code for each of the \(c\) candidates.

Each of these codes will then be transmitted as one-time pads to the voters in the same way as described by Protocol 2. It should be noted that Protocol 2 describes the transmission of only \(v\) codes as opposed to \(c \times v\) required by the voting protocol. To transmit all the voting codes, \(c\) executions of Protocol 2 will be executed at the same time. These executions should not be independent between them but instead should use the same permutations \((\rho \in R S_v\) in Step 2) and modifiers \((\omega_{i,j}\) in Step 4) used throughout all executions of the protocol, i.e. the same modifier is used for all codes the same voters will receive and they remain bundled together (i.e. by reusing \(\rho\)). These \(c\) executions can be carried out either in parallel or sequentially, as long as each voter receives \(c\) voting codes.

In the case of multi-seat elections, each voter will receive a single permutation over \(S_v\) - which is a permutation of the alphabetical ordering of the candidates. Moreover, Protocol 3 will be used.

6.2 Receiving and Reconstructing Voting Codes

We first explain the single-seat case. Each voter will receive \(l = t+1\) shares for each voting code, receiving each one using a different computational
device. It should be noted that the $i^{th}$ share of each of the $c$ voting codes will be received upon the same computational device.

The voter can then identify the code which corresponds to the candidate of their choice. Once all pieces of each code are received, the code corresponding to their choice can be reconstructed in a similar manner as described in Section 2.4.

In the multi-seat election, instead of receiving a $c$-tuple, a single permutation is received - which is a permutation of the alphabetical ordering of the candidates. Similar to the single seat case, $t+1$ shares of this permutation will be received by the voter who will reconstruct the permutation as described in [21] Section 4.2, Section 4.3. This will allow the voter to identify the candidates of their choice. Supposing the voter wants to vote for candidate $c$ and candidate $c'$, the reconstruction of the permutation will help the voter identify $\pi(c)$ and $\pi(c')$ which correspond to the candidates of their choice. To cast their vote, voters will have to send back to the CGE these $\pi(c)$ and $\pi(c')$ values.

**6.3 Transmission, Mixing and Counting of Cast Votes**

We first explain this for the single-seat case. A voter identifies the code corresponding to the candidate of their choice and sends this code back to the CGE by transmitting this code to the leader of the last block of MIX.

To transmit voter codes in the reverse direction (towards the CGE), the leaders of each block of MIX servers will have to carry out the reserve operations on the codes. Thus the inverse permutations ($\rho_i$) and modifiers ($-\omega_i$) are used. Once a code arrives to the CGE, it will identify the candidate it corresponds to and the vote will be counted.

The multi-seat case is similar. Once a voter identifies one of the $\pi(c)$ which corresponds to one of their chosen candidates, they will have to send this $\pi(c)$ to the leader of the last block of MIX servers. Similar to the single-seat case, the reserve operations on the codes will have to be carried out. Once a voter’s $\pi(c)$ arrives to the CGE, the CGE will apply $\pi^{-1}$ and identify the candidate the voting corresponds to and the vote will be counted.

**Acknowledgements:** The authors would like to thank the anonymous referees for their valuable comments on improving the presentation and clarity of this paper. The authors would also like to thank Rebecca Wright, Juan Garay and Amos Beimel for expressing their interests in Private and Secure Mes-
sage Transmission in which one cannot trust the equipment used by the receiver.
References

1. Four Grand Challenges in Trustworthy Computing. In CRA Conference on Grand Research Challenges in Information Security and Assurance. November 16–19, 2003, Warrenton, Virginia.

2. Official State of Geneva e-voting site. [http://www.geneve.ch/evoting/english/welcome.asp](http://www.geneve.ch/evoting/english/welcome.asp).

3. M. Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In Advances in Cryptology - EUROCRYPT ’98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding, pages 437–447.

4. M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In Proceedings of the twentieth annual ACM Symp. Theory of Computing, STOC, pages 113–131, May 2–4, 1988.

5. J. Blocki, M. Blum, and A. Datta. Human computable passwords. CoRR, 2014.

6. J. Buchmann, D. Demirel, and J. van de Graaf. Towards a publicly-verifiable mix-net providing everlasting privacy. In Financial Cryptography and Data Security 2013. Short Paper, Okinawa, Japan, April 15, 2013.

7. D. Chaum. SureVote: Technical Overview. Proceedings of the Workshop on Trustworthy Elections. [http://www.vote.caltech.edu/wote01/pdfs/securevote.pdf](http://www.vote.caltech.edu/wote01/pdfs/securevote.pdf). August 26-29 2001. Tomales Bay, CA, USA.

8. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM, 24(2):84–88, February 1981.

9. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In Proceedings of the twentieth annual ACM Symp. Theory of Computing, STOC, pages 11–19, May 2–4, 1988.

10. D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. T. Sherman, and P. L. Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. IEEE Security & Privacy, 6(3):40–46, 2008.

11. G. Cohen, I. B. Damgård, Y. Ishai, J. Kölker, P. B. Miltersen, R. Raz, and R. D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae. In CRYPTO (2), volume 8043 of LNCS, pages 185–202. Springer, 2013.

12. C. J. Colbourn and J. H. Dinitz. Handbook of Combinatorial Designs, 2nd Edition (Discrete Mathematics and Its Applications). Chapman & Hall/CRC, 2006.

13. R. Cramer, M. K. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In EUROCRYPT, volume 1070 of LNCS, pages 72–83. Springer. Zaragoza, Spain, May 1996.

14. Daily Mail. US senators demand ‘traitor’ NSA whistleblower be extradited from Hong Kong to face trial in America. [http://www.dailymail.co.uk/news/article-2338534](http://www.dailymail.co.uk/news/article-2338534).

15. Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Tech. Report ISSE-TR-97-01, George Mason University, July 1997. ftp://isse.gmu.edu/pub/techrep/97Tr1jajodia.ps.gz.

16. Y. Desmedt, J. Pieprzyk, R. Steinfeld, X. Sun, C. Tartary, H. Wang, and A. C.-C. Yao. Graph coloring applied to secure computation in non-abelian groups. Journal of Cryptology, 25(4):557–600, 2012.

17. Y. Desmedt and K. Kurosawa. How to break a practical MIX and design a new one. In Eurocrypt 2000, Proceedings LNCS 1807, pages 557–572. Springer-Verlag, 2000. Bruges, Belgium, May 14-18.

18. Y. Desmedt, Y. Wang, and M. Burmester. A complete characterization of tolerable adversary structures for secure point-to-point transmissions without feedback. In
19. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, January 1993.
20. S. Erotokritou and Y. Desmedt. Human perfectly secure message transmission protocols and their applications. In *SCN*, volume 7485 of *LNCS*, pages 540–558. Springer, 2012.
21. S. Estehghari and Y. Desmedt. Exploiting the client vulnerabilities in internet e-voting systems: Hacking Helios 2.0 as an example. In *2010 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE ’10)*, August 9–10, 2010.
22. C. for American Politics and Citizenship. Characteristics of contemporary voting machines.
23. S. Forrest, A. Somayaji, and D. H. Ackley. Building diverse computer systems. In *Workshop on Hot Topics in Operating Systems*, pages 67–72, 1997.
24. M. K. Franklin and M. Yung. Secure hypergraphs: Privacy from partial broadcast. *SIAM J. Discrete Math.*, 18(3):437–450, 2004.
25. J. Furukawa. Efficient and verifiable shuffling and shuffle-decryption. *IEICE Transactions*, 88-A(1):172–188, 2005.
26. E. Gerck, C. A. Neff, R. L. Rivest, A. D. Rubin, and M. Yung. The business of electronic voting. In *Financial Cryptography*, volume 2339 of *LNCS*, pages 234–259. Springer, 2001.
27. J. Groth. Linear algebra with sub-linear zero-knowledge arguments. In *CRYPTO*, volume 5677 of *LNCS*, pages 192–208. Springer, 2009.
28. J. Groth and Y. Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *EUROCRYPT*, volume 4965 of *LNCS*, pages 379–396. Springer, 2008.
29. Helios. Helios Voting. [http://heliosvoting.org/](http://heliosvoting.org/)
30. N. J. Hopper and M. Blum. Secure human identification protocols. In *ASIACRYPT*, volume 2248 of *LNCS*, pages 52–66. Springer, 2001.
31. S. Katti, J. Cohen, and D. Katabi. Information slicing: Anonymity using unreliable overlays. In *Proceedings of the 4th USENIX Symposium on Network Systems Design and Implementation (NSDI)*, pages 43–56. Cambridge, Massachusetts, U.S.A., April 11–13, 2007.
32. S. Khazaei, T. Moran, and D. Wikström. A mix-net from any CCA2 secure cryptosystem. In *Advances in Cryptology - ASIACRYPT 2012 Beijing, China, December 2-6*, pages 607–625.
33. E. Maaten. Towards remote e-voting: Estonian case. In *Electronic Voting in Europe - Technology, Law, Politics and Society*, volume 47 of *LNI*, pages 83–100. GI, 2004. July 7th–9th 2004, Bregenz, Austria.
34. A. Malkopoulou. Lost voters: Participation in eu elections and the case for compulsory voting., 2009.
35. T. Moran and M. Naor. Split-ballot voting: Everlasting privacy with distributed trust. *ACM Trans. Inf. Syst. Secur.*, 13(2), 2010.
36. M. O. Rabin and R. L. Rivest. Efficient end to end verifiable electronic voting employing split value representations. (To appear in Proc. EVOTE 2014 (Bregenz, Austria).
37. R. L. Rivest. Thoughts on appropriate technologies for voting. Invited keynote given at online special event, “E-Voting: Risk and Opportunity,” organized by CITP at Princeton University. November 1, 2012.
38. K. Sako and J. Kilian. Secure voting using partially compatible homomorphisms. In CRYPTO, volume 839 of LNCS, pages 411–424. Springer, 1994. August 21-25, 1994, Santa Barbara, California, USA.

39. K. Sampigethaya and R. Poovendran. A survey on mix networks and their secure applications. In Proceedings of the IEEE, volume 94, pages 2142–2181.

40. Security Musings. 2013 RSA Conference Opening Keynotes. http://securitymusings.com/article/3912/2013-rsa-conference-opening-keynotes.

41. The Cold War Museum. Senator Joseph McCarthy, McCarthyism and the Witch Hunt. http://www.coldwar.org/articles/50s/senatorjosephmccarthy.asp

42. The Register. NIST denies it weakened its encryption standard to please the NSA. http://www.theregister.co.uk/2013/09/11/nist_denies_that_the_nsa_weakened_its_encryption_standard/2013/09/11.

43. A. Tran, N. Hopper, and Y. Kim. Hashing it out in public: common failure modes of DHT-based anonymity schemes. In Proceedings of WPES 2009, Chicago, Illinois, USA, November 9, pages 71–80.

44. Wikipedia. Returning officer. http://en.wikipedia.org/wiki/Returning_officer

45. D. Wikström. The security of a mix-center based on a semantically secure cryptosystem. In INDOCRYPT, volume 2551 of LNCS, pages 368–381. Springer, 2002. Hyderabad, India, December 16-18, 2002.