Survey on Electronic Voting Schemes *

Laure Fouard, Mathilde Duclos, and Pascal Lafourcade **

VERIMAG, 2 avenue de Vignate, 38610 Grières, France
firstname.lastname@imag.fr

Abstract:
We present a survey on electronic voting schemes. We first summarize properties than such protocols should guarantee. In a second time we describe cryptographic primitives used in the context of electronic elections. Using these definitions we present a list of protocols and for each protocol we mention claimed, supposed, achieved or proven properties. When it is possible we also give existing or new attacks.

Keywords: cryptographic protocols.

* This work has been partially supported by the ANR project AVOTÉ
** Corresponding author: Pascal Lafourcade, VERIMAG, 2 avenue de Vignate, 38610 Grières, France, *tel:* +33 4 56 52 04 14, *fax:* +33 4 56 52 03 44, pascal.lafourcade@imag.fr

Contents

List of Tables

Notations

- $\stackrel{\text{def}}{=}$: means "equals by definition",
- $\stackrel{?}{=}$: means that one checks the equality,
- $n \stackrel{\$}{\leftarrow} A$: means that an element n is taken randomly from the set A,
- $R \underset{\subset}{\stackrel{\$}{\leftarrow}} A, |R| = l$: means that a subset R of size l is taken randomly from A,
- $m' = [m]_k$: means that m is encrypted to m' using the key k.
- $m = \{m'\}_k$: means that m' is decrypted to m using the key k.

1 Introduction

Since the existence of democracy, populations have been often consulted about critical decisions. To achieve it, a lot of voting systems have been designed. By voting, we mean the act of freely expressing some choices between publicly known alternatives, e.g. candidates. One of the most spread system is the paper based elections. In this system the day of the election, voters come and after authentication they receive a ballot and decide their vote in a polling-booth. Finally they vote in a transparency ballot box in front of the authority and sign a register to confirm that they have voted. When the voting phase is finished, the ballot box is publicly opened by the authority, ballots are counted and results of the election are published. Even today, in such system frauds are still possible (unfortunately as we can see in some countries). But people are quite convinced that some properties as anonymity, or verifiability of the vote are satisfied by a paper based election.

This usual process requires some time, by consequence the tally process and accuracy of results need to be improved in order to accelerate the publication and application of decisions, while keeping some requirement as the privacy of the voter. Electronic voting is needed to get a faster tally: instead of hours (even days) of counts, the results can be got within an hour...

That's why since the 1980's, electronic voting schemes have required many researches. While more and more protocols have been developed, the set of security properties that a protocol has to achieve has evolved. Thus, researchers keep combining existing or created cryptographic primitives to construct efficient electronic voting schemes, with respect to these security requirements.

In the first part of this work, we try to survey the main properties that electronic voting protocols should satisfy. It is not surprising to see that many authors propose different definitions of the same requirement for such protocols. In verification of cryptographic protocols, most of the authors agree on the definition of secrecy, but for instance for authentication several subtle variations of the definition have been proposed over the years (see [Low97] for an overview). In voting protocols the main properties that we want to achieve are at least the ones that a paper based election can guarantee:

- correctness: attest the accuracy of the results,
- privacy: keep the voter's identity unknown,
- receipt-freeness: prevent vote-selling, and coercion,
- robustness: resist to coalition attack and faulty behaviors,
- verifiability: protocol can be trusted,
- democracy: only register voters vote once.
- fairness: votes are not influenced by any information about the election, even partial, during the voting phase.

As we will see in this paper there exist often different definitions of the same property according to the authors. These variations in the definition of the properties are subtle. We try to present the most common one for each notion and to compare the differences with the other existing definitions in the literature.

In order to achieve those properties, protocols often use cryptographic primitives like zero-knowledge proof, secret sharing, homomorphic encryption, blind signature... In our second part, we interest in describing in detail all these mechanisms used in electronic voting protocol for guarantying security properties.

After these two first parts, we have introduced all the material to clearly present an overview of existing electronic protocols. For each protocol we mention author's names, year of publication and which one of the primitives, explained in the previous section, are used. From published articles, we also give a synthetic description of the protocol, which hypothesis are done (for instance if an anonymous channel is used...) and which properties are claimed, supposed or proven in the literature. Moreover, when it is the case we give new or existing attacks. We could have classified protocols into three parts: those

based on blind signatures and anonymous channel, those based on homomorphic encryption, and those based on mix-net. But some of them use at the same time different primitives (for instance [ALBD04] uses homomorphic primitives and mix-net), so we prefer to list them with the idea to try to start by those using blind signature, then homomorphic encryption and finally mix-net. Finally, we summarize this classification with a comparison of the studied protocols with respect to their specificities and the satisfied properties.

Related works: other similar papers

Motivations and Contributions: Explain why we do not consider all protocols in the literraure and motivations of our choices.

That is why we will not take into account protocols that do not have the correctness property, all protocols discussed are correct.

Papers to read and add :

- HE [CFSY95,BFP+01,DJN03,**?**,FPS01] new : [HK04,KMO01,KY02,LK00,LK02,Sch]
- MN [LBD+03,ALBD04,OA00,Nef01,Gro03,GZJ+02,FS01,JJR02,Fur04,Ben06,PBD05,DLM82,PIK93,Cha04,SK9
- BS [Sak94]

Outline In first Section we describe properties of electronic voting protocols, then in Section 3 we present all different primitives used by such protocols. In Section 4 we list existing electronic voting protocols. Before concluding, we try in Section 5 to compare presented protocols according to achieved properties and to primitives used.

2 Security Properties

In this Section, we present the main requirements that have to be satisfied to obtain a practical voting protocol. We note that no one of the existing protocols satisfies all these properties at the same time. Indeed, it is difficult to construct a perfect voting protocol given that some of the security properties required seem to be discrepant. As mention in [CMFST06], there exist some incompatible properties in voting schemes.

For instance they show that it is not possible to achieve:

- universal verifiability of the tally (UV) and unconditional privacy of the votes (UP) simultaneously, unless all the registered voters actually vote.
- universal verifiability of the tally (UV) and receipt-freeness (RF), unless the voting process involves interactions between several voters (and possibly the voting authority).

Therefore, authors develop protocols that satisfy properties that they consider to be more important, depending of the type of elections performed.

TODO: Ref et comparison des differentes definitions existantes.

For each property we first give a definition, then we try to list other variations of such defintion found in different papers. We quote exact definitions givent in those papers.

2.1 Correctness

This property is evidently the most important and we can reasonably consider that all the published protocols satisfy it.

Definition 1 (Correctness) *If all the election's participants (voters, authorities, ...) are honest and behave as it is scheduled, then the final results are effectively the tally of casted votes.*

In [FOO92] this property is called *completeness* and means "all valid votes are counted correctly". In [CBY86], Benaloh and Yung proposed "correctness: A election scheme is said to be correct if the tally computed by the protocol matches the actual tally". In this paper [JL97,JLY98]the authors introduced the notion of "Tally Correctness: The result of a secret ballot protocol is said to be correct if the published tally is equal to the actual result of the election". In [Web06] correctness is definied by: "The protocol includes all (last-cast) votes of authorized voters correctly into the election result".

2.2 Privacy

Definition 2 (Privacy) *No participant other than a voter should be able to determine the value of the vote cast by that voter. (Definition from [CBT94]) We can distinguish two levels of privacy :*

Perfect Privacy *No coalition of participants (voters or authorities), not containing the voter himself, can gain any information about the voter's vote.*

n-Privacy *No "n-coalition of participants", not containing the voter himself, can gain any information about the voter's vote. ("n-coalition of participants" means coalition of at most n authorities and any number of voters.)*

FOO [FOO92]

Privacy All votes must be secret.

Juang and Lei [JL97,JLY98]

Privacy A secret ballot protocol is said to be private if the privacy of voters is preserved.

Lee, Boyd and al [LBD$^+$03]

Privacy Ensures the secrecy of the contents of ballots. Usually it is achieved by encrypting the ballot with the public key of a group of authorities and b.

Weber [Web06]

Privacy The votes must be kept secret and not be linked to a voter.

Benaloh, Tuinstra and Yung [CBT94,CBY86]

Privacy We say that an election system is private if no dishonest protocols can enable any participant to distinguish between a voter running V_0 and another voter running V_1 with probability greater than $\frac{1}{2} + \frac{1}{2^N}$. (Note that this definition allows for the partial information which may be given by the election tally.)

Radwin [Rad95]

Current proposals for electronic voting protocols describe several properties of privacy and security. First and foremost, a protocol must ensure that votes are private. Victor the voter must be sure that any third party cannot determine who he voted for. That is, when Victor submits his vote over a communication channel, he assumes that a malicious eavesdropper Eve is listening. In order to achieve privacy, the voting protocol must employ some form of encryption such as a public-key cryptosystem. This privacy An untraceable, universally verifiable voting scheme 2 depends on the assumption that it is computationally infeasible for Eve to decrypt Victor's encrypted vote.

Untraceability Another desired property is the untraceability of a vote. That is, if Victor submits a vote, a second party (the voting authority) or third party (Eve) should be unable to trace the vote back to him. Even after decryption, the voting authority should be unable to determine the origin of a given vote. It should be able to verify that a vote has come from a valid voter, but it should not be able to discover which one; Victor's anonymity would be preserved. Such untraceability is desired because it mimics the behavior of conventional voting protocols.

Cohen and Fischer [CF85,Coh86]

We say that privacy is maintained if any conspiracy of voters has at most a small advantage at distinguishing between any two vote assignments that have the same sub-tally on the set of proper voters.

Rjaskova [Rja02]

No coalition of participants (of reasonable composition) not containing voter himself can gain any information about the voter's vote. By reasonable composition we mean coalition of at most t authorities and any number of voters. We say that information-theoretic privacy is achieved when the ballots of the voters are indistinguishable independent of any cryptographic assumption; otherwise we say that computational privacy is achieved.

2.3 Receipt-Freeness

Réflexion à avoir sur Receipt-Freeness et Uncoercibility: + Uncoercibility implies Privacy and Receipt-Freeness, mais toutes les autres implications entre les trois propriétés sont fausses.

Receipt-freeness prevents vote selling/buying, ensuring that voters are not used as a proxy to cast votes.

Definition 3 (Receipt-Freeness) *Voters must neither be able to obtain nor construct a receipt which can prove the content of their vote.*
(Definition from [ALBD04])

In some works, receipt-freeness means that the protocol does not require receipts, but in this paper, we consider receipt-freeness as uncoercibility because some voting protocols can give "receipts" to voters without the voter being able to use these to prove his vote whereas others need not supply voters with receipts in order for voters to be able to construct proofs of how they voted. [DKR06,CBT94]

Benaloh and Tuinstra [CBT94] This paper brings to light the need of separation between the definitions of receipt-freeness and uncoercibility. While the first just mentions the fact of being unable to produce a receipt proving the vote value, the second requires the impossibility of coercing some voter, whatever the chosen way. It seems straightforward that uncoercibility implies receipt-freeness.

Uncoercibility No voter should be able to convince any other participant of the value of its vote.
We say that a protocol P' is publicly consistent with a protocol P if P' running on input(s) x produces publicly-observable output(s) which could have been produced by P running on x. We say that a protocol P is coercible if there is some protocol P' which is publicly consistent with P and which provides an additional (private) receipt which can be used to prove that P' is consistent with P. Finally, an election system is uncoercible if neither of the two voting protocols V_0 and V_1 is coercible. [DKR06,CBT94]

Lee, Boyd and al [LBD$^+$03]

Receipt-freeness Ensures that a voter neither obtains nor is able to construct a receipt which can prove the content of his vote. To achieve this the voter should not be able to choose his own randomness when he prepares his ballot. A user chosen randomness can work as a receipt.

Weber [Web06]

Receipt-freeness A voter may not be able to create a receipt, i.e. any information that can be used to convince an attacker or a coercer that he voted in a specific manner.

Radwin [Rad95]

Sako-Kilian and Gennaro credit Josh Cohen Benaloh and David Tuinstra with introducing the first receipt-free protocol for electronic voting [SK95],Gennaro, Rosario. A Receipt-Free Election Scheme Tolerating a Dynamic Coercer (with Applications to Key Escrow), Massachusetts Institute of Technology, November 1995. . Benaloh and Tuinstra showed that other protocols give Victor a receipt for his vote, allowing him to later prove to another party that he voted a certain way. Victor could use his receipt to sell his vote, or he could be coerced under some threat into revealing his vote to a third party [SK95],Gennaro. A voting protocol that does not give Victor such a receipt (and therefore makes selling votes and coercion impossible) is called receipt-free. Sako and Kilian achieve this receipt-free quality by using a secure, private communication channel through which the voting authority can send Victor a message [SK95]. Gennaro achieves the same goal by a different physical assumption: Victor has secure hardware that does "oblivious probabilistic encryption" – a smart card, which is an electronic encryption device that does not reveal the random numbers it generates Gennaro.

Rjaskova [Rja02]

Receipt-freeness and Uncoercibility We say that the scheme is incoercible if the voter cannot convince any observer how he has voted. This requirement prevents vote-buying and coercion. Before the election, someone can bribe or coerce the voter to vote in a particular way. The coercer can order the voter how he should behave during the voting process (e.g. generates for him random bits). During the election, the coercer can observe the public communication between the voter and the authorities. After the election, he will want to see a proof that the voter really voted this way. In the scheme achieving privacy, the coercer alone or with reasonable coalition of participants cannot open the voterâs vote. Thus the coercer will force the voter to show him his secret information. With it, he is capable of opening the ballot and seeing the vote. Incoercible scheme provides the voter with the ability to modify his secrets and to open his ballot in any desired way. Thus the voter can vote on his own will and he can feed the coercer with a false proof.

2.4 Robustness

Definition 4 (n-Robustness) *Faulty behavior of a n-coalition of authorities can be tolerated. No coalition of voters can disrupt the election and any cheating voter will be detected. (Definition from [Rja02])*

FOO [FOO92]

Soundness The dishonest voter cannot disrupt the voting.

Lee, Boyd and al [LBD+03]

Robustness Ensures that the voting system can tolerate a certain number of faulty participants.

Weber [Web06]

Coercion Resistance We allow the adversary to demand of coerced voters that they vote in a particular manner, abstain from voting, or even disclose their secret keys. We define a scheme to be coercion-resistant if it is infeasible for the adversary to determine whether a coerced voter complies with the demands.

Cohen and Fischer [CF85,Coh86]

Called Security The scheme is robust if no conspiracy of dishonest voters can prevent, with more than a very low probability, the successful completion of the election.

2.5 Verifiability

Definition 5 (Verifiability) *Correct voting processes must be verifiable to prevent incorrect voting results. (Definition from [ALBD04])*
There are two kinds of verifiability :

Universal (Public) Verifiability *Any participant or passive observer can convince himself of the validity of individual votes and of the final tally of election.*
Individual Verifiability *Every eligible voter can verify that his vote was counted. (Definition from [Rja02])*

FOO [FOO92]

Verifiability No one can falsify the result of the voting.

Weber [Web06]

Universal Verifiability Anyone can verify that the protocol correctly processed and tallied all valid votes.
Individual Verifiability Each voter can verify that his own (last-cast) vote is correctly included in the final result.

Benaloh and Tuinstra [CBT94]

Correctness (seems equivalent to universal verifiability) Every participant should be convinced that the election tally accurately represents the "sum" of the votes cast. An election system with m voters is correct if a specially designated output common to all participants who follow correct protocols gives a common correct tally with probability at least $1 - \frac{m}{2^N}$ for a given security parameter N.

Lee, Boyd and al [LBD+03]

Universal Verifiability Ensures that any party can be convinced that all valid votes have been included in the final tally. To achieve this all the relevant messages need to be published and the correctness of all processes (voting, mixing, tally) should be publicly verifiable.

Radwin [Rad95]

The Sako-Kilian and Rosario Gennaro proposals describe the property of individual verifiability, the ability for Victor to verify if his vote was received properly (Sako 395, Gennaro 7). Victor would desire this property because it proves to him that the voting authority has counted his vote, and gives him some evidence if he needs to levy a complaint because his vote was lost. Individual verifiability allows only Victor to check for the correct receipt of his ballot. Because each voter must check his or her own vote, an auditor would have to contact and receive the cooperation of every voter to audit the election. Universal verifiability allows any voter or interested third party to at a later time verify that the election was properly performed (Sako 394). Only with universal verifiability can an audit be performed easily, so this property is desired as long as it does not incur too substantial of a cost (Sako 395).

Cohen and Fischer [CF85,Coh86]

Let δ be a function of the security parameter N. The scheme is said to be *verifiable with confidence* δ if, for any election system ϵ, the *check* function satisfies the following properties for random runs of ϵ using N:

1. If the government is proper in ϵ, then, with probability at least $1 - \delta(N)$, *check* returns good and the government releases a correct tally.
2. Whether or not the government is proper, the joint probability is at most $\delta(N)$ that *check* returns good and the government releases an incorrect tally (or fails to release any tally).

The scheme is said to be *verifiable* if it is verifiable with confidence δ and $\delta = \frac{1}{p(N)}$ for some non-constant polynomial p with positive leading coefficient.

Rjaskova [Rja02]

Individual verifiability Each eligible voter can verify that his vote was really counted.

Universal verifiability Any participant or passive observer can check that the election is fair: the published final tally is really the sum of the votes.

2.6 Democracy

Definition 6 (Democracy) *There are two requirements to satisfy in democracy :*

Eligibility *Only authorized voters are allowed to vote.*

Prevention of Multiple Voting *All eligible voters are allowed to cast the scheduled vote's number (function of the election system and his part in it) and not more, such that each voter has his intended power in deciding the outcome of the voting.*

(Definition from [ALBD04])

[FOO92]

Unreusability No voter can vote twice. (similar to Prevention of Multiple Voting).
Eligibility No one who isn't allowed to vote can vote.

Lee, Boyd and al [LBD+03]

Prevention of double voting: Ensures that any eligible voters can vote only once. To achieve this the authority needs to check the authenticity and eligibility of the voter and record the participation in his secure database. If a public bulletin board is used as a public communication channel (most messages are posted there) and voters access it in an authenticated manner, double voting can be prevented easily.

Juang and Lei [JL97,JLY98]

Completeness A secret ballot protocol is said to be complete if the ballot of an eligible voter is always accepted by the administrator.
Eligibility A secret ballot election is said to be sound if no ineligible voter can vote.

Rjaskova [Rja02]

Eligibility Only eligible voters can cast the votes. Every voter can cast only one vote.

2.7 Fairness

Definition 7 (Fairness) *No participant can gain any knowledge, except his vote, about the (partial) tally before the counting stage (The knowledge of the partial tally could affect the intentions of the voters who has not yet voted).*
(Definition from [Rja02])

FOO [FOO92]

Fairness Nothing must affect the voting.

Juang and Lei [JL97,JLY98]

Fairness A secret ballot election is said to be fair if no one can get extra information of the tally result before the publication phase.

Lee, Boyd and al [LBD+03]

Fairness Ensures that no partial tally is revealed to anyone before the end of the election procedure, as it may affect the voting result in some way. Sometimes we need to trust that the authorities will not reveal partial tally.

2.8 Definition given in Radwin scheme

[Rad95]

Convenience Sako and Kilian preface their proposal by stating that the ultimate goal of secure electronic voting is to replace physical voting booths (Sako 393). Traditional voting places a burden on citizens because they must be at the appropriate polling place in order to vote in a physical voting booth. This inconvenience may affect voter turnout: according to a report from the Population Division of the Bureau of the Census, less than 45 percent of .S. citizens aged 18 years or older reported voting in the November 1994 election (Census). Electronic voting has the potential to greatly affect voter registration and turnout if the process of voting can be made more convenient. An untraceable, universally verifiable voting scheme 3 An electronic voting scheme which does not require Victor's presence in a physical voting booth would remove this inconvenience and is therefore desirable. A protocol for voting which allows Victor to vote from any one of several networked polling locations would be superior to the current system but still inconvenient. A greater degree of convenience is achieved when Victor is able to vote from any networked location such as a telephone, ATM machine, or interactive-television set. Ideally, Victor should not require any external device that interacts with the existing networked device. A lesser degree of convenience than that of traditional voting results if such a device is required.

2.9 Additional Properties Defined by Sako and Kilian

[SK94]

Communication Cost Represents the total cost of communications due to messages that have to be sent for various computations and proofs.

Round Complexity Shows the protocol efficiency during the election time, i.e. the vote casting and tally computation stages. One can consider the number of needed messages.

Preprocessing An election scheme is more efficient if a large amount of computations and communications is done in advance, before the election time.

Security Under reasonable heuristic assumptions, no coalition of voters or authorities (called centers) can unfairly influence an election or significantly delay its outcome.

Universal Verifiability Every action by a voter, whether preprocessing a vote or actually voting, is accompanied by a proof that the ballot is correctly constructed. The output of the counting center may also be easily checked for correctness. Any participant can verify that everyone's vote has been included in the tally.

Flexibility The scheme is adaptable in terms of number of authorities or security/efficiency tradeoff choice.

Anonymity property is studied in [GHPvR05] using unlikable notion.

3 Cryptographic Primitives

We present now the notions generally used in electronic voting protocols.

3.1 Interactive Zero Knowledge Proofs

Interactive zero knowledge proofs introduced in [GMR89] are simply explained in [QGAB89] and clearly described in the survey [Gol01,Gol02]. Here we just briefly explain main idea of this concept and notions that we will use in the description of protocols in next Section.

TODO: Clarify what we need.

In an interactive zero knowledge proof, a prover *P* interacts with a verifier *V* to demonstrate the validity of an assertion without revealing anything about the assertion to the verifier. This kind of proofs has three required properties:

Completeness If the prover is honest (i.e. the assertion is true), then the verifier should accept the proof. We say that the protocol is complete if the probability that the verifier accepts the proof given that the assertion is true, is close to 1.

Soundness If the prover is dishonest (i.e. the assertion is false), then the verifier should reject the proof with noticeable probability. We say that the protocol is sound if the probability that the verifier accepts the proof given that the assertion is false, is less than $\frac{1}{2}$.

Zero Knowledge The verifier should learn nothing more than the truth of the assertion during the interaction. Roughly speaking, the protocol is said "zero knowledge" if, no matter what the verifier outputs after the protocol, it could have produced the same output without interacting with the prover.

In voting protocols, interactive zero knowledge proofs are used in many stages: between authorities which want to prove their honesty in order to proceed the protocol, or between authorities and voters, for example when people want to be convinced of the validity of the tally (Universal Verifiability), or when a voter want to be convinced of the correctness of its vote's encryption (Individual Verifiability).

3.2 Secret Sharing

We present primitive called "Secret Sharing" introduced by A. Shamir in [Sha79]. The main idea of this scheme consists in sharing a secret s between N authorities, such that any set of at least $t+1$ authorities can recover s, while any set of at most t authorities can't get any information on s. Secret sharing primitives are generally based on polynomial interpolation.

As example, we detail in appendix **??** a scheme presented in [Sch99] where the secret is a point of a polynomial of degree t. The shares are points of the polynomial, and s is recovered using Lagrange interpolation.

Secret sharing is used in electronic voting in order to get the voting protocol to be robust against authorities coercions. E.g. in [HS00] detailed in next Section, the key used to decrypt ballots is shared between the authorities at the beginning of the protocol.

3.3 Homomorphic Encryption

Homomorphism is an algebraic property particularly useful in electronic voting schemes because it allows to apply operations on sets of encrypted ballots without need of decrypting them. In electronic voting schemes, this notion is used as follows:

Definition 8 (Homomorphic Encryption) *Let $\mathbb{M}$ be a plain-texts groups and $\mathbb{C}$ be a cipher-texts group. Then, we say that the encryption scheme is $(\otimes, \oplus)$-homomorphic if for any instance E of the encryption scheme, given $c_1 = E_{r_1}(m_1)$ and $c_2 = E_{r_2}(m_2)$, there exists an r such that : $c_1 \otimes c_2 = E_r(m_1 \oplus m_2)$.*

Such encryption schemes are often used in electronic voting protocols, for example, to compute the tally without decrypting each vote and therefore guarantee the privacy of voters. In Appendix, we recalled RSA [RSA77], ElGamal [ElG85] and Pailler [PP99] encryption scheme which are homomorphic and used in some electronic voting schemes described in the rest of the paper.

3.4 Re-Encryption

We give now a short description of re-encryption which has inherited properties of homomorphism. An immediate consequence of homomorphism is that a encryption scheme satisfying this property is able

to perform re-encryption : Given a cipher-text c, anyone can create a different cipher-text c' that encodes the same plain-text as c.

This follows from properties of homomorphic encryption (for further details, see [Sho05]): By the definition of groups, $\mathbb{M}$ has an identity plain-text m_0 such that $\forall m \in \mathbb{M}$, $m + m_0 = m$. Thus, given a homomorphic encryption scheme, we can define the re-encryption algorithm RE as follows: $RE_r(c) = c \otimes E_r(m_0)$. Let D be an instance of the decryption of E. If $D(c) = m$, then $D(RE_r(c)) = m$.

This property is used to achieve the receipt-freeness in some electronic vote protocols.

3.5 Blind Signature

A blind signature allows somebody for instance an authority to sign an encrypted message without decrypting it. Once the message signed and resent to the sender, he has a signed version of his vote by the authority and a guarantee that his vote has not been seen. In Appendix **??** we present how RSA can be used to produce genuine signature of a document.

Finally we introduce generic notations to describe blind signature scheme presented in [Rja02]. We use in the rest of the document these notations. Formally, the blind signature scheme with message space M is a 5-tuple $(\eta, \chi, \sigma, \delta, \Gamma)$, where

- η is a polynomial-time probabilistic algorithm, that constructs the signer's public key (pk) and its corresponding secret key (sk);
- χ is a polynomial-time blinding algorithm, that on input a message $m \in M$, a public key pk and a random string r, constructs a blind message m';
- σ is a polynomial-time signing algorithm, that on input a blind message m' and the secret key sk constructs a blind signature s' on m';
- δ is a polynomial-time retrieving algorithm, that on input a blind signature s' and the random string r extracts a signature s on m;
- Γ is a polynomial-time signature-verifying algorithm that on input a message signature pair (m, s) and the public key pk outputs either yes or no.

Blind signature is often used to get a token from the authority: in [FOO92] the voter gets a signature from the authority of his ballot and then he is able to cast his ballot. It is used to achieve eligibility.

3.6 Deniable Encryption

A *deniable* scheme offer the possibility to generate encrypted message that can be decrypted to different plain-texts, depending on the key used. Such encryption makes impossible to prove the existence of the real message without the proper encryption key. That is to say that the cipher-text can be decrypted into several clear-texts. This notion is explored in [CDNO97]. We can mention that the most secure encryption scheme, *i.e.*, One Time Pad, is a deniable encryption, because from any cipher if you do not know the encryption, you can apply a wrong key and get any messages.

Deniable encryption is a way to prevent coercibility by allowing the voter to produce a fake receipt so that a coercer cannot find out the real clear-text: it achieves receipt freeness. For instance in one of the encryption schemes proposed in [Rja02] the deniable property of encryption is used.

3.7 Mix-net

When using a ballot box voters vote in an order and when election is finished votes come out in a different order. This ensures the anonymity of the voter. One possibility to realize it electronically is to use so-called *mix-net* first introduced by Chaum [Cha81]. Such protocols mix messages by sending them through a network of authorities, where each authority shuffles the received list of messages before to send it to the next one, while keeping the permutation secret. There are many different types of

mix-nets, and many different definitions and constructions, an overview of this area is clearly done in B. Adida [Adi06]. Notice that some of these constructions use Zero Knowledge Proof like [FS01], Re-encryption [PIK93] or secret sharing [OKST97].

We present set-ups and stages that the most mix-nets protocols have in common. In voting protocols using mix-nets, mix-servers are authorities and each of them possesses a public key and a corresponding secret key (or more). The ballots have to be prepared before the elections using the mix-servers' public keys. During the election stage, each ballot is cast and passed through the mix-net to be decrypt by the successive mix-servers' secret keys before the final tally.

4 Electronic Voting Schemes

The goal of this section is to present an overview of electronic voting schemes using the cryptographic primitives described above. We start with procotols based on blind signatures, then those based on mix-net and finally those based on homomorphic encryption.

We can notice that there exist some hybrid protocols that mix many of these three categories. Such a mixing has to be constructed very carefully. Indeed, two protocols can verify a property while their combination does not.

Hence for each of them, we first make a short description of the protocol derived from published articles. We also present achieved, claimed supposed properties and when it is possible we mention existing or new attacks.

Here, we present you electronic voting schemes based on Blind Signature. Blind signature is usually used by the voter to get a token (a blinded signed message) from the authorities so that he can vote. It is often used with an anonymous channel.

4.1 Radwin's Scheme

Authors: Michael J. RADWIN (1995).
Primitives used: zero-knowledge proof, blind signature.
Requirements: This scheme uses anonymous channels supporting replays, and two two-argument functions that are collision-free.

Summary: This scheme proposed in [Rad95] is composed in 4 stages: an initialization stage, in which the authority sets up RSA; a registration stage (during which each voter constructs a token with the help of the authority), a voting stage (in which the voter constructs and casts his ballot, and the ballot is decrypted) and a counting stage (in which the results are published). It can be used for yes/no election.

It needs a single authority, even if it can be extended to include more authorities. The token (an object which allows someone to vote, like a voting card) is here a pseudonym (constructed by the voter, with the help of the authority) signed by the voter.

Protocol Description: Let f, g be two collision-free two-arguments functions, the output of f looks random and g is 1 to 1 with the first argument fixed. The symbol $\oplus$ denotes bitwise exclusive or, ID is the voter's identity, and l is the security parameter.

Initialization Stage:

– The authority (A) sets up RSA: (p, q, d), and publishes $(n, e), l$ (l is a security parameter used below.)

Registration Stage:

- The voter (V_i) produces the random values $a_k, c_k, d_k, r_k \overset{\$}{\leftarrow} \mathbb{Z}_n, \forall k \in [1, \cdots, 2l]$. Then V_i computes $x_k \overset{\text{def}}{=} g(a_k, c_k)$ and $y_k \overset{\text{def}}{=} g(a_k \oplus ID, d_k)$, and $B_k \overset{\text{def}}{=} {r_k}^3 f(x_k, y_k)$. Then V_i sends B_k to A .
- A picks up a random subset R of l elements of the B_k and challenges V_i to show if he had correctly constructed the B_k.
- V_i sends $a_k, c_k, d_k, r_k, \forall k \in R$ to A .
- A checks $\forall k \in R$ that $B_k \overset{?}{=} {r_k}^3 f(g(a_k, c_k), g(a_k \oplus ID, d_k))$.
- A sends $\prod_{k \notin R} {B_k}^{1/3} \mod n$ to V_i.
- V_i extracts his pseudonym: $P \overset{\text{def}}{=} \frac{\prod_{k \notin R} {B_k}^{1/3}}{\prod_{k \notin R} r_k} \mod n$.

Voting Stage:

- V_i prepares his ballot as $B \overset{\text{def}}{=} (v, P)$ (where v is the vote: 0 for "no", and 1 for "yes"), and sends its encryption using RSA $[B]_{pk} (= B^e \mod n)$ to A .
- A decrypts the ballot, and then chooses a random binary vector $Z \overset{\text{def}}{=} \{z_i\}$, $(i \notin R)$, and sends it to V_i, in order to challenge him to prove that P is valid.
- $\forall i \in \{1, \cdots, 2l \backslash R\}$,
  - if $z_i = 0$ then V_i sends a_i, c_i, y_i to A ,
  - if $z_i = 1$ then V_i sends $x_i, (a_i \oplus ID), d_i$
- A checks that P has the correct form, and checks that nobody has voted with the same P yet. If P is used twice, $\exists k$, $z_k \neq z'_k$ with high probability. Thus, one can deduce ID from $(a_k, a_k \oplus ID)$.
- A adds v to the tally. A stores Z, P and V_i answers to the Z challenge.

Counting Stage: A could simply publish a tally of "yes" and "no" ballots along with each corresponding pseudonym P.

The protocol is presented in table 1.

Properties Claimed:

Democracy - Prevent Multiple Voting Claimed by the author in the original paper. The voter needs the help of the authority to build his pseudonym: without this, he cannot cast his vote. If he tries to vote several times, he will be detected by the authority (in the voting stage) and will be traced back.

Privacy Claimed by the author in the original paper. During the registration phase, the voter discloses only one half of the B_k (to ensure the authority that he has correctly constructed his pseudonym), which are then not used in the construction of the pseudonym. So, the authority can't tell anything about the voter's pseudonym. As the output of f looks random, the pseudonym looks also random. Then, the blind signature prevents anyone from linking a voter to his pseudonym, and as an anonymous channel is used to sent the ballot, the voter's privacy is protected. However, if the voter tries to vote twice, he will reveal his identity.

Individual Verifiability Claimed by the author in the original paper. The voter can check in the list that the vote registered with his pseudonym is the vote he casted.

Attacks:

f, g are collision-free two-arguments functions, the output of f looks random and g is 1 to 1 with the first argument fixed. $\oplus$ is bitwise exclusive or. l is the security parameter.

private input	Voter (V_i) ID (V_i's identity)	Authority A
Initialization Stage		RSA: (p, q, d), $\boxed{(n, e), l}$
Registration Phase	$k = 1, 2, \cdots, 2l$ $a_k, c_k, d_k, r_k \overset{\$}{\leftarrow} \mathbb{Z}_n$ $x_k \overset{\text{def}}{=} g(a_k, c_k) \quad y_k \overset{\text{def}}{=} g(a_k \oplus ID, d_k)$ $B_k \overset{\text{def}}{=} {r_k}^3 f(x_k, y_k)$ $B_k, k = 1, \cdots, 2l$ `sent` $a_k, c_k, d_k, r_k; \quad k \in R$ `sent` $P \overset{\text{def}}{=} \frac{\prod_{k \notin R} {B_k}^{\frac{1}{3}} \mod n}{\prod_{k \notin R} r_k} \mod n$	$R \underset{\subset}{\overset{\$}{\leftarrow}} \{1, \cdots, 2l\}, \lvert R\rvert = l$ R `sent` $B_k \overset{?}{=} {r_k}^3 f(g(a_k, c_k), g(a_k \oplus ID, d_k))$ check fails $\Rightarrow$ registration rejected $\prod_{k \notin R} {B_k}^{\frac{1}{3}} \mod n$ `sent`
Voting Phase	ballot: $B(v, P)$ $B^e \mod n$ **`sent`** $z_k = 0 \Rightarrow t_k \overset{\text{def}}{=} (a_k, c_k, y_k)$ $z_k = 1 \Rightarrow t_k \overset{\text{def}}{=} (x_k, a_k \oplus ID, d_k)$ $T = \{t_k\} \quad T$ **`sent`**	$B^{de} \mod n$ $z_k \overset{\$}{\leftarrow} \{0, 1\}, k \notin R \quad Z = \{z_k\}$ Z **`sent`** $z_k = 0 \Rightarrow p_k \overset{\text{def}}{=} f(g(t_{k_1}, t_{k_2}), t_{k_3})$ $z_k = 1 \Rightarrow p_k \overset{\text{def}}{=} f(t_{k_1}, g(t_{k_2}, t_{k_3}))$ $P \overset{?}{=} \prod_{k \notin R} p_k$ adds v
Counting Stage		$L \overset{\text{def}}{=} \{(P, v)\} \boxed{L}$

Caption: element $\boxed{\text{published}}$, `sent`, **`sent through anonymous channel`**.

Table1. Radwin's voting scheme

Receipt-Freeness Attack proposed by the author in the original paper. (a_i, c_i, d_i, r_i) is a receipt for the voter.

Robustness The authority can cast votes instead of abstaining voters.

Universal Verifiability This latest attack also holds for universal verifiability: invalid votes are casted without being noticed.

4.2 FOO's Scheme

Authors: Atsushi FUJIOKA, Tatsuaki OKAMOTO, and Kazuo OHTA (1992).
Primitives used: zero-knowledge proof, blind signature.
Requirements: This scheme uses anonymous channels.

Summary: This scheme proposed in [FOO92] is composed in four stages: an initialization stage, a registration stage (during which every voter gets a token from the administrator using a blind signature scheme), a voting stage (the voters send their encrypted vote along with their token), and a counting stage (the votes are decrypted and counted).

It needs two authorities: an administrator, which is responsible for the tokens issuing, and a collector which collects the tokens and publishes the results of the election. A token is here an encrypted ballot, blindly signed by the administrator.

Protocol Description:

Initialization stage:

- The administrator (Ad) sets up a blind signature scheme $(\eta, \chi, \sigma, \delta, \Gamma)$ and a bit-commitment scheme ξ , and publishes ξ , its public key pk and its blinding algorithm χ ,
- V_i chooses his vote v_i, creates his ballot $B_i \stackrel{\text{def}}{=} \xi(v_i, k_i)$, using a random key k_i,
- V_i blinds his ballot by computing $e_i \stackrel{\text{def}}{=} \chi(B_i, pk, r_i)$, where r_i is a random number.

Registration stage: The voter (V_i) constructs his ballot with the help of Ad . He commits himself to his vote (v_i), then blinds this commitment and sends it together with his identity to Ad , which signs it after it has verified that the voter is able to vote. At the end of this phase, Ad announces the number of voters and publishes the list of the blinded commitments and identities.

- V_i signs $s_i \stackrel{\text{def}}{=} \sigma_i(e_i, sk_i)$ and sends (ID_i, e_i, s_i) to Ad ,
- Ad checks that V_i has the right to vote,
- Ad checks that V_i has not applied for a signature,
- Ad checks the signature s_i of e_i (i.e. $\Gamma_i(e_i, s_i, pk_i)$),
- If these checks have succeed, then Ad signs $d_i \stackrel{\text{def}}{=} \sigma(e_i, sk)$,
- Ad sends d_i as a token to V_i,
- At the end of the stage, Ad publishes the number of voters (n) who were given his signature, and a list containing (ID_i, e_i, s_i), for all V_i who has been registered.

Voting stage: V_i retrieves the signature of his ballot, and verifies it. Then he sends the token (that is to say, the ballot and its signature) anonymously to the collector (C). Finally, C checks the signature of the ballot.

- V_i retrieves the signature y_i of B_i by computing $y_i \stackrel{\text{def}}{=} \delta(d_i, r_i)$
- V_i checks the signature ($\Gamma(B_i, y_i, pk)$), if it fails, then V_i complains in showing that (B_i, y_i) is invalid,
- V_i sends (B_i, y_i) to C through the anonymous channel.
- C checks $\Gamma(B_i, y_i, pk)$. If it succeeds, C enters (l, B_i, y_i) onto a list (L) with number l.

Counting stage: C publishes the ballots and their signatures. The voters check that this publication is correct, and send their keys to C through an anonymous channel. Eventually, C retrieves the votes, checks them, counts them and publishes the election results.

- Once every one has voted, C publishes the list,
- V_i checks if $|L| \stackrel{?}{=} n$,
- V_i checks if $B_i \in L$. If it is not, he opens (B_i, y_i),
- V_i sends (k_i, l) to C by the anonymous channel.
- C opens $\xi(B_i)$, retrieves v_i, adds (k_i, l) on L and checks that v_i is valid.
- C counts the votes and publishes the results.

The protocol is presented in table 2.

	V_i	Ad	C
private input	ID (V_i's identity)		
Initialization Stage	$v_i, k_i \stackrel{\$}{\leftarrow} \mathbb{Z}$ $B_i \stackrel{\text{def}}{=} \xi(v_i, k_i)$ $r_i \stackrel{\$}{\leftarrow} \mathbb{Z}$ $e_i \stackrel{\text{def}}{=} \chi(B_i, pk, r_i)$	sets up $(\eta, \chi, \sigma, \delta, \Gamma)$ and ξ $\boxed{\xi\ ,\ pk\ ,\ \chi}$	
Registration stage	$s_i \stackrel{\text{def}}{=} \sigma_i(e_i, sk_i)$ (ID_i, e_i, s_i) `sent to` Ad	 $\Gamma_i(e_i, s_i, pk_i)$ $d_i \stackrel{\text{def}}{=} \sigma(e_i, pk)$ d_i `sent to` V_i $\boxed{n}$ $\forall i, \boxed{(ID_i, e_i, s_i)}$	
Voting stage	$y_i \stackrel{\text{def}}{=} \delta(d_i, r_i)$ $\Gamma(B_i, y_i, pk)$ (B_i, y_i) **`sent to`** C		 $\Gamma(B_i, y_i, pk)$ $(l, B_i, y_i) \rightarrow L$
Counting stage	 $\|L\| \stackrel{?}{=} n$ $B_i \stackrel{?}{\in} L$ (k_i, l) **`sent to`** C		$\boxed{L}$ open $\xi(B_i) \Rightarrow v_i$ $\boxed{(k_i, l) \rightarrow L}$ counts, $\boxed{\text{results}}$

Element $\boxed{\text{published}}$, `sent`, **`sent through anonymous channel`**.

Table2. FOO voting scheme

Properties Claimed:

Individual Verifiability: Claimed by Atsushi FUJIOKA, Tatsuaki OKAMOTO, and Kazuo OHTA in the original paper. The voter can check that his vote has been taken into account. Thus, individual verifiability is achieved.

Properties Checked:

Democracy: Proven by Steve KREMER and Mark RYAN in [KR05]. Only eligible voters can get a token. Invalid tokens or votes will be detected, and nobody can vote several times without being noticed.

Privacy: Proven in [KR05]. The link between the voter and his token is hidden by the blind signature. And as the voter communicates with the collector through an anonymous channel, the privacy is achieved, even is the administrator and the collector conspire.

Fairness: Proven in [KR05]. Ballots aren't counted until the end of the voting stage.

Attacks:

Universal Verifiability: Attack proposed by Zuzana RJASKOVA in [Rja02]. If some voters abstain vote, the administrator can cast his votes instead. Therefore, universal verifiability doesn't hold.

Robustness: This previous attack holds if the administrator is corrupted.

Receipt-Freeness: In [Rja02], an attack against Receipt-Freeness is also described. The token is a receipt (anyone can retrieve the vote with it). Thus this property isn't achieved. Note that [Oka97] is a modification of this scheme achieving receipt-freeness.

Privacy: An attack has been found in [KR05] when the stages have not been synchronized: if only one voter has voted when the collector begins to publish the list, then anyone can trace back the voter.

Democracy, robustness: Possible attack: note that a corrupted administrator can give token to anyone he wants to, even non-eligible voters.

4.3 Ohkubo et al.'s Scheme

Authors: Miyako OHKUBO, Fumiaki MIURA, Masayuki ABE, Atsushi FUJIOKA and Tatsuaki OKAMOTO (1999).
Primitives used: Blind signature.
Requirements: Sender-authenticated public channel.

Summary: This scheme proposed in [OMA+99] is inspired of the FOO scheme described in subsection 4.2 and is composed in 3 stages: in the registration stage, the voter prepares a ballot and gets an authorization from the administrator (by obtaining a blind signature from the administrator). Then in the voting stage, the voter sends his ballot anonymously, using a mix-net. In the counting stage, tailliers co-operatively open the ballots and count the votes.

This scheme needs an administrator (Ad), several mixers (M) and several tailliers (T).

Protocol Description: t is the threshold for the encryption scheme used by the tailliers.

Registration stage: The voter (V_i) prepares his ballot: he chooses his vote, encrypts it to get it blindly signed by the administrator (Ad). At the end of the stage, Ad publishes a list of who has been enabled to vote.

- The voter (V_i) selects vote (v_i) and encrypts v_i with talliers' public key (of the threshold encryption scheme) as $B_i \stackrel{\text{def}}{=} [v_i]_{pk_T}$. V_i blinds B_i as $e_i \stackrel{\text{def}}{=} \chi(B_i, r_i)$ where r_i is a randomly chosen blinding factor. V_i signs e_i: $s_i \stackrel{\text{def}}{=} \sigma_{V_i}(e_i)$ and sends (ID_i, e_i, s_i) to the administrator (Ad).

– Ad checks that V_i has the right to vote. If V_i doesn't have the right, Ad rejects the authorization. Ad checks that V_i has not already applied for a signature. If V_i has already applied, Ad rejects the authorization. Ad checks the signature s_i of message e_i. If they are valid, then Ad signs e_i: $d_i \stackrel{\text{def}}{=} \sigma_{Ad}(e_i)$ and sends d_i as Ad's certificate to V_i.
– V_i retrieves the desired signature y_i of ballot B_i by $y_i \stackrel{\text{def}}{=} \delta(di, ri)$. V_i checks that y_i is the Ad's signature of B_i. If the check fails, V_i claims it by showing that (B_i, y_i) is invalid.
– At the end of the Registration stage, Ad announces the number of voters who were given the Ad's signature, and publishes a list containing (ID_i, e_i, s_i).

Voting stage: V_i sends his vote encrypted with the mixers' (M) key, via a sender-authenticated public channel.

– V_i sends $c_i \stackrel{\text{def}}{=} [(B_i, y_i)]_{pk_M}$, the encryption of his ballot using the mixers' (M) key to the bulletin board.

Counting stage: Mixers mix the ballots, the tailliers (T) check the signatures of all ballots, then decrypt and count them.

– M decrypts the list of c_i and outputs the list of (B_i, y_i) in random order.
– Each taillier T_j checks the signature y_i of ballot B_i using the Ad's verification key. If the check fails and at least t tailliers agree on this, M has to reveal the corresponding c_i and to prove in zero-knowledge that $(B_i, y_i) = \{c_i\}_{sk_M}$. If the proof is correct, (B_i, y_i) is invalidated and excluded hereafter. If the proof is wrong, then M is malicious and is not to be used again.
– All tailliers cooperatively decrypt ballot B_i and retrieve vote v_i in communication via the bulletin board. The tailliers publish the voting results.
– Every one can check that the number of ballots (including excluded ones) on the list equals the number of voters and that all tailliers act correctly. If the checks fails, one can claim this.

The protocol is presented in table 3.

	V_i	Ad	M	T_j
Registration	$B_i \stackrel{\text{def}}{=} [v_i]_{pk_T}$, $r_i \stackrel{\$}{\leftarrow} \mathbb{Z}$, $e_i \stackrel{\text{def}}{=} \chi(B_i, r_i)$, $s_i \stackrel{\text{def}}{=} \sigma_{V_i}(e_i)$, (ID_i, e_i, s_i) `sent to` Ad $y_i \stackrel{\text{def}}{=} \delta(di, ri)$, $\Gamma_{Ad}(y_i)$	$\Gamma_{V_i}(s_i)$, $d_i \stackrel{\text{def}}{=} \sigma_{Ad}(e_i)$, d_i `sent to` V_i $\boxed{(ID_i, e_i, s_i)}$		
Voting	$c_i \stackrel{\text{def}}{=} [(B_i, y_i)]_{pk_M}$ $\boxed{c_i}$ **`sent through sender-authenticated public channel`**			
Counting			$(B_i, y_i) = \{c_i\}_{sk_M}$, $\pi \leftarrow$ random permutation, $\boxed{(x_{\pi(i)}, y_{\pi(i)})}$	$\Gamma_{Ad}(y_i)$, $\{B_i\}_{sk_T}$

Element $\boxed{\text{published}}$, `sent`, **`sent through sender-authenticated public channel`**.

Table3. Ohkubo et al.'s Scheme

Properties Checked: As this scheme is widely inspired of FOO scheme described in subsection 4.2, the authors claimed to fulfill the same properties, for the same reasons. Then we have:

Correctness Claimed by the author in the original paper.

Privacy Claimed by the author in the original paper.

Democracy Claimed by the author in the original paper.

Fairness Claimed by the author in the original paper.

Individual Verifiability Claimed by the author in the original paper. The authors add a property of convenience: the voter does not need to wait for the end of the registration stage to cast his vote, and then walk away.

Attacks: As for the claimed properties, most of the attacks used against FOO's scheme are also effective against this scheme:

Universal Verifiability If some voters abstain vote, the administrator can cast his votes instead. Therefore, universal verifiability doesn't hold.

Robustness Same attack holds.

Receipt-Freeness c_i is a token for V_i, as he sent it via a sender-authenticated public channel. It is the easy for the voter to show that c_i is an encryption of his vote.

4.4 Kim et al.'s Scheme

Authors: Kwangjo KIM, Jinho KIM, Byoungcheon LEE and Gookwhan AHN (2001).
Primitives used: blind signature, zero-knowledge proof.
Requirements: ElGamal cryptosystem is secure if discrete logarithm is intractable.

Summary: This scheme proposed in [KKLA01] is designed to be a remote voting scheme. It is composed in 3 stages: in the registration stage, the voter obtains a key pairs along with a certificate from the administrator; in the voting stage, the voter casts his ballot (blindly signed by the administrator) to the talliers. At last, in the counting stage, the ballots are decrypted and counted.

This scheme requires an administrator (Ad), a mixer (M), tailliers (T), a certification authority and a bulletin board ($\mathcal{BB}$).[1]

Protocol Description:

Registration stage: The voter (V_i) contacts the administrator (Ad) to be registered. At the end of the process, V_i obtains a key generation applet from Ad and a certificate from the certification authority.

- All authorities generate keys (using a public key infrastructure).
- V_i fills in an registration form, encrypts it using Ad's public key and sends it to Ad.
- Ad checks that V_i is allowed to vote (if it fails, he sends an error message). Ad enables V_i to download the key generation applet.
- V_i downloads the applet and generates key pairs. He then sends his public key to Ad.
- Ad asks the certification authority for a certificate (C_i) for V_i. V_i receives C_i.

[1] In the original paper, Ad, M, T and $\mathcal{BB}$ are servers.

Voting stage: V interacts with Ad to cast his vote: V asks Ad to get his vote (blindly) signed and then sends it along with his signature and certificate to the bulletin board ($\mathcal{BB}$).

- V_i downloads a login applet and identifies himself (with ID_i - password).
- Ad checks if V_i has already voted. If he has, he rejects the authorization. if he has not, he authorizes V_i to download the voting applet.
- V_i downloads the voting applet, selects his vote (v_i), encrypts it with the talliers' (T) public key of the threshold encryption scheme $B_i \stackrel{\text{def}}{=} [v_i]_{pk_T}$. Then V_i blinds it using r_i a random string, $e_i \stackrel{\text{def}}{=} \chi(B_i, r_i)$, signs it: $s_i \stackrel{\text{def}}{=} \sigma_{sk_{V_i}}(e_i)$ and sends (ID_i, e_i, s_i) to Ad .
- Ad checks the signatures, if it is valid, Ad signs e_i: $d_i \stackrel{\text{def}}{=} \sigma_{sk_{Ad}}(e_i)$ and sends it to V_i.
- V_i retrieves the signature $y_i \stackrel{\text{def}}{=} \delta(d_i, r_i)$ and checks the signature. If it fails, he complains by publishing (B_i, y_i). V_i encrypts (B_i, y_i): $c_i \stackrel{\text{def}}{=} [(B_i, y_i)]_{pk_M}$ and sends it to $\mathcal{BB}$ along with his signature and his certificate.
- $\mathcal{BB}$ checks the signature and certificate.
- At the end of the stage, Ad announces the number of signatures he has made and publishes the final list of all (ID_i, e_i, s_i).

Counting stage: The mixer (M) decrypts and mixes the ballots, then tailliers decrypt the ballots and count the votes.

- M decrypts the c_i and outputs the list of (B_i, y_i) in a random order.
- T_j checks the signature y_i. If it fails, T_j complains by publishing (B_i, y_i). If more than the threshold tailliers agree with T_j, M has to prove (in zero-knowledge) that (B_i, y_i) is the correct decryption of c_i and T_j checks the proof. If it fails, the M has produced a wrong proof. If it succeeds, the voter has casted an invalid vote. The vote is then excluded.
- Tailliers decrypt together the ballots B_i and retrieve $v_i = \{B_i\}_{sk_T}$.

The protocol is presented in table 4.

Properties Checked:

Fairness Claimed by the author in the original paper. As the ballots are decrypted only in the counting stage by the tailliers, no one can learn anything on the election's results.

Individual Verifiability Claimed by the author in the original paper. V_i can check if his ballot is in the list published during the counting stage, and as the tailliers are using a threshold cryptosystem, it is improbable that if a ballot is in the list, it is not count.

Democracy As any voter must interact with the administrator to get his vote signed, democracy is achieved.

Receipt-Freeness As the ballots are mixed before they're published, receipt-freeness is achieved.

Privacy The only way to bind a voter to his vote is that the mixer keeps track of his permutation, and then coerces a taillier to track back a voter. Thus, if the authorities are honest, privacy is achieved.

Attacks:

Universal Verifiability No one (except the tailliers) can verify the tally, since the decrypted ballots are not published.

	V_i	Ad	M	T
private input	ID-pass			
Registration		generate keys		
	$[\texttt{form}]_{pk_{Ad}}$ sent to Ad	checks form and V_i's eligibility key generator sent to V_i		
	generates keys pk_{V_i} sent to Ad	asks certification authority for C_i C_i sent to V_i		
Voting	$[ID_i\text{-pass}]_{pk_{Ad}}$ sent to Ad	checks if V_i has voted voting applet sent to V_i		
	$B_i \stackrel{\text{def}}{=} [v_i]_{pk_T}$, $r_i \stackrel{\$}{\leftarrow} \mathbb{Z}$, $e_i \stackrel{\text{def}}{=} \chi(B_i, r_i)$, $s_i \stackrel{\text{def}}{=} \sigma_{sk_{V_i}}(e_i)$, (ID_i, e_i, s_i) sent to Ad	$\Gamma(s_i)$, $d_i \stackrel{\text{def}}{=} \sigma_{sk_{Ad}}(e_i)$, d_i sent to V_i		
	$y_i \stackrel{\text{def}}{=} \delta(d_i, r_i)$, $c_i \stackrel{\text{def}}{=} [(B_i, y_i)]_{pk_M}$, $\boxed{(c_i, \sigma_{sk_{V_i}}(c_i), ID_i)}$	$\boxed{\text{number of signatures, } \forall i(ID_i, e_i, s_i)}$		
Counting			$\{c_i\}_{sk_M}$, $\pi \leftarrow$ random permutation, $\boxed{(B_{\pi(i)}, y_{\pi(i)})}$	$\Gamma(y_i)$, $v_i = \{B_i\}_{sk_T}$, count

Element $\boxed{\text{published in } \mathcal{BB}}$, sent, **sent through anonymous channel**.

Table4. Kim et al.'s Scheme

Robustness If the administrator is malicious, he can vote in place of any abstaining voter.

4.5 Juang and Lei's scheme

Authors: Wen-Shenq JUANG and Chin-Laung LEI (1997).
Primitives used: Blind signature.
Requirements: There exists an untraceable e-mail system (anonymous channel), and secure one-way permutation and function, RSA is secure if factorizations intractable, ElGamal cryptosystem is secure if discrete logarithm is intractable.

Summary: This scheme proposed in [JL97] is composed in 6 stages: during the initialization stage, the administrator sets up the system parameters, during the global key generating phase, all the scrutineers work together to generate a threshold verifiable public key and share it together without the help of any one else.

It uses a single administrator and some scrutineers, who monitor the election process.

The original paper considers that a voter cannot abstain after the registration stage, at least $(m-k+1)$ scrutineers don't disclose their shadow key until the publication case, and k scrutineers are honest and send their shadow key.

Protocol Description: Let $\|$ denotes the string concatenation operator. Let e_{Ad}, n_{Ad}, d_{Ad} be RSA keys for the administrator (Ad), let $e_{Scru,j}, n_{Scru,j}, d_{Scru,j}$ be RSA keys for the scrutineer j ($Scru_j$), let $e_{V,i}, n_{V,i}, d_{V,i}$ be RSA keys for the voter i (V_i).

Initialization Stage: Administrator (Ad), voters (V) and scrutineers ($Scru$) set up RSA. Ad sets up a one-way permutation (f), a one-way function (g) and a redundancy bits (RD). He also chooses a constant (k) which is used in a threshold cryptosystem.

- Ad chooses (p, q, g) such as p, q are large prime numbers, and $q|(p-1)$, $s/(\gcd(s,p) = 1, s \neq 1)$, f one way permutation, h one way function, RD redundancy bits, and $k \in \mathbb{N}$ constant
- Ad computes $g \equiv (s^{(p-1)/q} \mod p$
- Ad publishes $(n_{Ad}, n_{Ad}^{d_{Ad}})$, $(e_{Ad}, e_{Ad}^{d_{Ad}})$, $(p, p^{d_{Ad}})$, $(q, q^{d_{Ad}})$, $(g, g^{d_{Ad}})$, $(f, f^{d_{Ad}})$, $(h, h^{d_{Ad}})$, $(RB, RB^{d_{Ad}})$, $(k, k^{d_{Ad}})$.

Global Key Generation Stage: $Scru$ sets up a threshold ElGamal with the help of Ad .

- $Scru_j$ sets up ElGamal: $(g^{a_j} \mod p, -a_j)$
- $Scru_j$ constructs a random polynomial: $\forall i \in \{1, \cdots, k-1\}, f_{i,j} \overset{\$}{\leftarrow} \mathbb{R}$, $f_{0,j} \overset{\text{def}}{=} a_j$, $f_j(x) \equiv \sum_{i=0}^{k-1} f_{i,j} x^i \mod q$
- $Scru_j$ computes $\forall 0 \leq i \leq k-1,\ GF_{j,i} \equiv g^{f_{i,j}}$
- $Scru_j$ sends $(GF_{j,i}, GF_{j,i}^{d_{Scru}})$ to Ad
- Ad checks if the signature is correct, if it's not, Ad publishes the invalid signatures and stops.
- Ad computes $G_p \equiv \prod_{i=1}^{m} GF_{i,0} \mod p$ and publishes $G_p, (GF_{j,i}, GF_{j,i}^{d_{Scru}})\ \forall (j,i) \in \{1, \cdots, m\} \times \{0, \cdots, k-1\}$
- $Scru_j$ sends $(s_{j,l}, s_{j,l}^{d_{Scru}})$ to each scrutineers.
- Once $Scru_j$ has received all $s_{l,j} (\forall 1 \leq l \leq m, l \neq j)$, he computes $s_j \overset{\text{def}}{=} \sum_{i=1}^{m} s_{i,j}$, $CertGP_j \overset{\text{def}}{=} G_p^{d_{Scru}}$.
- $Scru_j$ sends $CertGP_j$ to Ad .
- Ad checks the signatures for all the scrutineers. If it is not valid, he publishes the invalid signatures and stops.
- Ad computes $CertGP \overset{\text{def}}{=} G_p^{d_{Ad}}$ and publishes $G_p,\ CertGP,\ CertGP_j\ (1 \leq j \leq m)$.

Registration Stage: V asks Ad for a token: V constructs a ballot and blinds it. Then, he sends it to Ad . Ad checks this ballot, registers V and sends him back a token.

- V_i selects two strings R_i and ϕ_i randomly, and a random number δ_i. He also chooses his vote x_i.
- V_i computes $V_i \stackrel{\text{def}}{=} x_i \parallel \delta_i \parallel h(x_i \parallel \delta_i \parallel RD)$.
- V_i selects randomly a value r_i such as $1 < r_i < n_{Ad}, \gcd(r_i, n_{Ad}) = 1$ and a number r_i' and then computes:
  - $EV_i \stackrel{\text{def}}{=} (P_i \parallel Q_i) \stackrel{\text{def}}{=} (g^{r_i'} \parallel ((G_p^{r_i'})V_i \mod p))$
  - $Reg_i \equiv (\phi_i \parallel h(\phi_i \parallel RD))^{d_{V,i}} \mod n_{V,i}$
  - $H_i \stackrel{\text{def}}{=} f(ID_i \parallel R_i)$
  - $M_i \stackrel{\text{def}}{=} H_i \parallel RD \parallel EV_i$
  - $Y_i \equiv (r_i^{e_{Ad}} M_i)$
  - $\mathcal{R}_i \stackrel{\text{def}}{=} ((Y_i \parallel Reg_i)^{d_{V,i}} \mod n_{V,i}) \parallel ID_i$
- V_i sends $\mathcal{R}_i$ to Ad .
- Ad checks if the ballot is valid. If it's not valid, Ad requests V_i to send him again the message. Ad checks the identity of V_i: $Reg_i^{e_{V,i}} \stackrel{?}{=} \phi_i \parallel h(\phi_i \parallel RD) \mod n_{V,i}$. If it doesn't hold, Ad rejects the registration of V_i. If it holds and V_i has already been registered, Ad rejects also the registration of V_i. Otherwise, Ad registers V_i by saving back $\mathcal{R}_i$ in the registration database.
- Ad computes $Z_i \equiv Y_i^{d_A d} \mod n_{Ad}$ and sends Z_i to V_i

Voting Stage: V sends his vote anonymously to Ad , and Ad verifies this ballot and records it.

- V_i computes $X_i \equiv Z_i r_i^{-1} \equiv M_i^{d_A d} \mod n_{Ad}$ and sends (X_i, M_i) anonymously to Ad (via untraceable e-mail).
- Ad checks $(X_i)_{Ad}^{e} \equiv H_i \parallel RD \parallel EV_i \equiv H_i \parallel RD \parallel (P_i \parallel Q_i) \equiv M_i \mod n_{Ad}$. If it's correct and RD is valid, Ad records (X_i, M_i).

Counting Stage: V verifies if his vote has been counted, and Ad decrypts the ballot (with the secret shared by the scrutineers), counts, and publishes the results.

- Ad publishes all accepted ballots (X_i, M_i).
- V_i checks if his ballot is contained in the published list. If it's not, he complains by revealing (X_i, M_i).
- if there is no objection, Ad requests k scrutineers $Scru_{p_j}$ $(p_j \in [1, m], 1 \leq j \leq k)$ for their s_{p_j}.
- Ad computes $G_s \stackrel{\text{def}}{=} -\sum_{j=1}^{k} s_{p_j} \prod_{i=1, i \neq j}^{k} \frac{-p_i}{p_j - p_i}$
- Ad computes $V_i \equiv (P_i)^{G_s} Q_i \mod q$.
- Ad publishes all (X_i, M_i, V_i), all $\mathcal{R}_j$ and G_s.

The protocol is presented in table 5.

Properties Claimed:

Privacy Claimed by the authors in the original paper. Sketch of proof: a ballot cannot be linked back to the voter and it is infeasible to extract the voter's ID from the ballot.

Individual Verifiability Claimed by the authors in the original paper. Sketch of proof: The voter must check that his ballot is indeed in the published list.

Properties Checked:

	V_i	Ad	$Scru_j$
Initialization Stage	RSA: $(d_{V_i}), (n_{V_i}, e_{V_i})$	RSA: $(d_{Ad}), (n_{Ad}, e_{Ad})$, (p, q, g) such as p, q are large prime numbers, and $q \mid (p-1)$, $g \equiv (s^{(p-1)/q} \mod p$, $(\gcd(s,p) = 1, s \neq 1)$, f one way permutation, h one way function, RD redundancy bits, $k \in \mathbb{N}$ constant $\boxed{(n_{Ad}, n_{Ad}^{d_{Ad}}), (e_{Ad}, e_{Ad}^{d_{Ad}}), (p, p^{d_{Ad}}), (q, q^{d_{Ad}}), (g, g^{d_{Ad}}), (f, f^{d_{Ad}}), (h, h^{d_{Ad}}), (RB, RB^{d_{Ad}}), (k, k^{d_{Ad}})}$	RSA: $(d_{Scru_j}), (n_{Scru_j}, e_{Scru_j})$
Key Generation Stage		$CertGF_{j,i}^{e_{Scru}} \stackrel{?}{=} GF_{j,i}$ checks false $\Rightarrow$ $\boxed{\text{invalid } (GF_{j,i}, GF_{j,i}^{d_{Scru}})}$, stops $G_p \equiv \prod_{i=1}^{m} GF_{i,0} \mod p$ $\boxed{G_p, (GF_{j,i}, GF_{j,i}^{d_{Scru}})}$ $\forall (j,i) \in \{1, \cdots, m\} \times \{0, \cdots, k-1\}$ $\forall 1 \leq j \leq m$, $CertGP_j^{e_{Scru}} \stackrel{?}{=} G_p$ $CertGP \stackrel{\text{def}}{=} G_p^{d_{Ad}}$ $\boxed{G_p, CertGP, CertGP_j (1 \leq j \leq m)}$	ElGamal: $(g^{a_j} \mod p, -a_j)$ $f_j(x) \equiv \sum_{i=0}^{k-1} f_{i,j} x^i \mod q$ $\forall i \in \{1, \cdots, k-1\}, f_{i,j} \stackrel{\$}{\leftarrow} \mathbb{R}$, $f_{0,j} \stackrel{\text{def}}{=} a_j$ $GF_{j,i} \equiv g^{f_{i,j}}$ $\forall 0 \leq i \leq k-1$ $CertGF_{j,i} = GF_{j,i}^{d_{Scru}}$ sends $(GF_{j,i}, CertGF_{j,i})$ to Ad $s_{j,l} \equiv f_j(l) \mod q$ sends $(s_{j,l}, s_{j,l}^{d_{Scru}})$ to each $Scru$ received: $s_{l,j}$ $(\forall 1 \leq l \leq m, l \neq j)$ checks: $g^{s_{l,j}} \equiv \prod_{i=0}^{k-1} (GF_{l,i}^{j^i})$ mod p. $s_j \stackrel{\text{def}}{=} \sum_{i=1}^{m} s_{i,j}$, $CertGP_j \stackrel{\text{def}}{=} \sigma(G_p, -a_j)$ sends $CertGP_j$ to Ad
Registration Stage	$\delta_i \stackrel{\$}{\leftarrow} \mathbb{Z}$,random string: R_i, ϕ_i. Vote: x_i $V_i \stackrel{\text{def}}{=} x_i \parallel \delta_i \parallel h(x_i \parallel \delta_i \parallel RD)$ $r_i \stackrel{\$}{\leftarrow} \mathbb{Z} / 1 < r_i < n_{Ad}, \gcd(r_i, n_{Ad}) = 1, r_i' \stackrel{\$}{\leftarrow} \mathbb{Z}$ $EV_i \stackrel{\text{def}}{=} (P_i \parallel Q_i) \stackrel{\text{def}}{=} (g^{r_i'} \parallel ((G_p^{r_i'}) V_i \mod p))$, $Reg_i \equiv (\phi_i \parallel h(\phi_i \parallel RD))^{d_{V,i}} \mod n_{V,i}$, $H_i \stackrel{\text{def}}{=} f(ID_i \parallel R_i)$, $M_i \stackrel{\text{def}}{=} H_i \parallel RD \parallel EV_i$, $Y_i \equiv (r_i^{e_{Ad}} M_i)$, $\mathcal{R}_i \stackrel{\text{def}}{=} ((Y_i \parallel Reg_i)^{d_{V,i}} \mod n_{V,i}) \parallel ID_i$, sends $\mathcal{R}_i$ to Ad	checks validity of $\mathcal{R}_i$, $Reg_i^{e_{V,i}} \stackrel{?}{=} \phi_i \parallel h(\phi_i \parallel RD) \mod n_{V,i}$ saves back $\mathcal{R}_i$, $Z_i \equiv Y_i^{d_{Ad}} \mod n_{Ad}$ sends Z_i to V_i	
Voting Stage	$X_i \equiv Z_i r_i^{-1} \equiv M_i^{d_{Ad}} \mod n_{Ad}$ **sends** (X_i, M_i) **to** Ad	checks $(X_i)_{Ad}^{e} \equiv H_i \parallel RD \parallel EV_i \equiv H_i \parallel RD \parallel (P_i \parallel Q_i) \equiv M_i \mod n_{Ad}$ saves back (X_i, M_i)	
Counting Stage	$(X_i, M_i) \stackrel{?}{\in}$ published list	$\forall i, \boxed{(X_i, M_i)}$ requests k $Scru_{p_j}$ $(p_j \in [1,m], 1 \leq j \leq k)$ for their s_{p_j} $G_s \stackrel{\text{def}}{=} -\sum_{j=1}^{k} s_{p_j} \prod_{i=1, i \neq j}^{k} \frac{-p_i}{p_j - p_i}$ $V_i \equiv (P_i)^{G_s} Q_i \mod q$ $\forall i, j$ $\boxed{(X_i, M_i, V_i), \mathcal{R}_j, G_s}$	

Element published, sent, **sent through anonymous channel**.

Table5. Juang and Lei's scheme

Democracy Proven by the authors in the original paper. Sketch of the proof: the voter can construct only one vote with the help of the authority (thanks to RSA). Invalid or double votes are excluded.

Robustness Proven by the authors in the original paper. Sketch of the proof: since every voter must check if their ballot is on the published list, the administrator cannot cast extra votes.

Fairness Proven by the authors in the original paper. Sketch of the proof: as the ballots are decrypted after all the ballots were collected, no information on the final tally can leak before the counting stage.

Attacks:

Receipt-freeness Attack proposed by Zuzana RJASKOVA in [Rja02]. (X_i, M_i) is a receipt for the voter.

Robustness Note that if a voter can abstain after registration stage and if the administrator is malicious, he can cast vote instead of voters who had abstained at the registration stage.

Universal Verifiability This last attack holds.

4.6 Juang, Lei and Yu's scheme

Authors: Wen-Shenq JUANG, Chin-Laung LEI and Pei-Ling YU (1998).
Primitives used: Blind signature, secret sharing.
Requirements: There exists an untraceable e-mail system (anonymous channel), and secure one-way permutation function, RSA signature scheme and ElGamal cryptosystem are secure.

Summary: This scheme proposed in [JLY98] is composed in 5 stages: in fact, it is the scheme described in 4.5, extended to several administrators, using secret sharing. It is also extended to the case where a voter abstains from voting after the registration stage. A collector is assigned to collect the ballots.

Protocol Description: Let μ be the threshold value of the blind threshold signature (to be used by administrators), and λ be the threshold value of the threshold cryptosystem (to be used by scrutineers). Let $\|$ denotes the string concatenation operator. We note d_{user} the secret key of $user$ and e_{user} the corresponding public key of $user$, in the RSA cryptosystem.

Initialization stage: The collector (C) sets up all public parameters and publishes them.

- the collector (C), every voter (V_i), administrator (Ad_i) and scrutineer ($Scru_i$) set up RSA, and publish the public keys.
- C publishes the public parameters, as the tag of the current election (tag), the number of administrators (n), the number of eligible voters (τ), the number of scrutineers (ς), μ, λ, and identification of all eligible voters, scrutineers and administrators. He also sets up and publishes ζ a one-way permutation and h a one-way hash function.
- C chooses (p, q, g) such as p, q are large prime numbers such that $q|\,(p-1)$ and ρ s.t. $(\gcd(\rho, p) = 1, \rho \neq 1)$. He then computes $g \stackrel{\text{def}}{=} \rho^{(p-1)/q} \mod p$. He publishes p, q, g.
- C computes all the signatures of what he published and publishes them, along with their according objects.

Key generation stage: Administrators (Ad) and scrutineers ($Scru$) set up their threshold public key and share their secret key.

Let x_i be a unique public number for Ad_i. Let y_i be a unique public number for $Scru_i$.

Administrators construct their group public key.

- Ad_i chooses a secret key $z_i \in \mathbb{Z}_q$ and secret $f_i(x) \overset{\text{def}}{=} \sum_{k=0}^{\mu-1} f_{i,k} x^k$ such that $f_{i,0} = z_i$. Then he computes $\psi_{i,k} \overset{\text{def}}{=} g^{f_{i,k}} \mod p$ for all $0 \leq k \leq \mu - 1$.
- Ad_i computes the signature $\sigma(h(\psi_{i,k}), d_{Ad_i})$, the commitment $C_i \overset{\text{def}}{=} \xi_{\gamma_i}(\psi_{i,0})$ (γ_i is a random string) and the signature $\sigma(h(C_i), d_{Ad_i})$.
- Ad_i sends $(C_i, \sigma(h(C_i), d_{Ad_i}))$, $(\psi_{i,k}, \sigma(h(\psi_{i,k}), d_{Ad_i}))$ to any other Ad .
- Once he received all the shares and the commitment from other Ad , Ad_i verifies all the signatures. If the check fails, he publishes the invalid signature and stops. Otherwise, he opens C_i and sends $\delta_{i,j} \overset{\text{def}}{=} f_i(x_j) \mod q$, and $\sigma(h(\delta_{i,j}), d_{Ad_i})$ to each other Ad .
- After checking each signature on the $\delta_{j,i}$, Ad_i verifies if $g^{\delta_{j,i}} \overset{?}{=} \prod_{l=0}^{\mu-1} (\psi_{j,l})^{x_i{}^l} \mod p$. If it fails, Ad_i publishes the invalid share with the signature and the name of the concerned administrator.
- Ad_i computes $y \overset{\text{def}}{=} \prod_{l=1}^{n} y_l = \prod_{l=1}^{n} \psi_{l,0} \mod p$ and $\sigma(h(y), d_{Ad_i})$, $\phi_{j,i} \overset{\text{def}}{=} g^{\delta_{j,i}} \mod p, 1 \leq j \leq n$ and $\sigma(h(\phi_{j,i}), d_{Ad_i})$. Then he sends $(\sigma(h(y), d_{Ad_i}))$, $(\phi_{j,i}, \sigma(h(\phi_{j,i}), d_{Ad_i}), \forall j \in [1, n])$ to each other Ad .
- Ad_i checks all the signatures he received. If it fails, Ad_i publishes the invalid signatures and stops.
- Otherwise, group public key y, and public key y_j for each Ad , and all $\phi_{l,j}$, $1 \leq l, j \leq n$ are published.

Scrutineers construct their group public key.

- $Scru_j$ chooses a secret key a_j and construct a polynomial $\theta_j(x) \overset{\text{def}}{=} \sum_{i=0}^{\lambda-1} \theta_{j,i} x^i \mod q$ where $\theta_{j,0} = a_j$.
- $Scru_j$ computes $F_{j,i} \overset{\text{def}}{=} g^{\theta_{j,i}} \mod p$ and $\sigma\left(h(F_{j,i}), d_{Scru_j}\right)$ for $0 \leq i \leq \lambda - 1$, and then sends $\left(F_{j,i}, \sigma\left(h(F_{j,i}), d_{Scru_j}\right)\right)$ to $Scru_i$, $1 \leq i \leq \varsigma$, $i \neq j$.
- $Scru_j$ verifies each signatures. If he finds an error, he publishes the incorrect signatures and then stops. Then he computes $\epsilon_{j,i} \overset{\text{def}}{=} \theta_j(y_i)$ and sends it along with his signature to $Scru_i$.
- $Scru_j$ verifies the signature and computes his share: $\epsilon_j \overset{\text{def}}{=} \sum_{i=1}^{\varsigma} \epsilon_{i,j}$.
- The scrutineers' group public key is then computed: $G_p \overset{\text{def}}{=} \prod_{i=1}^{\varsigma} F_{i,0}$, and every $Scru_j$ publishes $\sigma\left(h(F_{j,i}), d_{Scru_j}\right)$.

Registration stage: Voters (V) construct with the administrators a token (a ballot blindly signed by Ad).

We assume that V_i asks to get his token from μ arbitrary honest Ad ($Ad_j, 1 \leq j \leq \mu$).

- Ad_j chooses $k_{i,j} \overset{\$}{\leftarrow} \mathbb{Z}_q$ and then computes $\widehat{r_{i,j}} \overset{\text{def}}{=} g^{k_{i,j}} \mod p$, and sends it along with $\sigma\left(h(\widehat{r_{j,i}} \parallel tag), d_{Ad_j}\right)$ to V_i.
- V_i checks all the signatures. If it is valid, he chooses $\eta_i \overset{\$}{\leftarrow} \mathbb{Z}$ he computes $B_i \overset{\text{def}}{=} v_i \parallel \eta_i \parallel h(v_i \parallel \eta \parallel tag)$ (v_i is the intentional vote of V_i), and then chooses $R_i, \varphi_i, r_i' \overset{\$}{\leftarrow} \mathbb{Z}$.
- V_i computes $EB_i \overset{\text{def}}{=} (P_i \parallel Q_i) \overset{\text{def}}{=} \left(g^{r_i'} \parallel \left(\left(G_p^{r_i'}\right) B_i \mod p\right)\right)$, $H_i \overset{\text{def}}{=} \zeta(ID_{V_i} \parallel R_i)$ and $M_i \overset{\text{def}}{=} H_i \parallel tag \parallel EB_i$, and at last $Reg_i \overset{\text{def}}{=} \sigma\left(h(\varphi_i \parallel tag \parallel ID_{Ad_j}), d_{V_i}\right) \parallel \varphi_i \parallel ID_{Ad_j} \parallel ID_{V_i}$.
- V_i chooses $\alpha_i \overset{\$}{\leftarrow} \mathbb{Z}_q$, $\beta_i \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ and computes $r_{i,j} \overset{\text{def}}{=} g^{\alpha_i} \widehat{r_{i,j}}^{\beta_i} \mod p$, $r_i \overset{\text{def}}{=} M_i \prod_{k=1}^{\mu} r_{i,k} \mod p$, and $\widehat{m_i} \overset{\text{def}}{=} \beta_i^{-1} r_i \mod q$.

- V_i checks $\widehat{m_i} \stackrel{?}{=} 0$. If it is true, he re-does the last step. Otherwise, he sends $\mathfrak{R}_i \stackrel{\text{def}}{=} \sigma\left(h\left(\widehat{m_i} \parallel Reg_i\right), d_{V_i}\right) \parallel \widehat{m_i} \parallel Reg_i$ to all $Ad_j, 1 \leq j \leq \mu$.
- Ad_j checks if $\mathfrak{R}_i$ is correct. If it is not, he asks V_i to resend him $\mathfrak{R}_i$. Otherwise, he checks the identity of V_i (by verifying the signature), if he has not registered yet, and if Reg_i contains ID_{Ad_j}. If one of these statements doesn't hold, Ad_j rejects the registration. Otherwise, he registered V_i by keeping Reg_i in the registration database.
- Ad_j computes $\widehat{s_{i,j}} \stackrel{\text{def}}{=} \widehat{m_i}\left(z_j + \sum_{l=\mu+1}^{n}\left(f_l\left(x_j\right)\left(\prod_{\substack{k=1 \\ k\neq j}}^{\mu}\left(\frac{-x_k}{x_j - x_k}\right)\right)\right)\right) + k_{i,j} \mod q$, and sends it to V_i.
- Once he received all $\widehat{s_{i,j}} (1 \leq j \leq \mu)$, V_i computes $s_{i,j} \stackrel{\text{def}}{=} \widehat{s_{i,j}}\beta_i + \alpha_i \mod q$. He then checks $g^{-s_{i,j}} y_j^{r_i} r_{i,j} \stackrel{?}{=} \left(\prod_{l=\mu+1}^{n}\left(\phi_{l,j}\right)\right)^{\left(\prod_{k=1,k\neq j}^{\mu}\left(\frac{-x_k}{x_j - x_k}\right)\right)(-r_i)} \mod p, 1 \leq j \leq \mu$. If one of the $\widehat{s_{i,j}}$ is not valid, he asks the corresponding administrator to send it again. Otherwise, he computes $s_i \stackrel{\text{def}}{=} \prod_{j=1}^{\mu} s_{i,j} \mod q$.

Voting stage: Voters send their votes anonymously to the collector (C), and C checks the ballots.

- V_i sends anonymously to C (r_i, s_i, tag).
- C computes $M_i \stackrel{\text{def}}{=} g^{-s_i} y^{r_i} r_i \mod p = H_i \parallel tag \parallel EB_i \mod p$, and checks tag. Then he records (r_i, s_i, tag, M_i) (rejects it if the check is wrong). At the end of the phase, he sorts them by M_i and keeps only one M_i.

Counting stage: Administrators decrypt, count and publish the ballots.

- C publishes the list of the accepted (r_i, s_i, tag, M_i).
- Each voter checks if their ballot is in the list. If not, he objects by broadcasting his (r_i, s_i, tag, M_i).
- After a delay, if there is no objection, C asks λ arbitrary honest scrutineers ($Scru_j, 1 \leq j \leq \lambda$) to send their shadow key ε_j. Then he computes $G_s \stackrel{\text{def}}{=} -\sum_{j=1}^{\lambda}\left(\varepsilon_j \prod_{i=1, i\neq j}^{\lambda} \frac{-x_j}{(x_j - x_i)}\right) \mod q$, and computes $B_i = (P_i)^{G_s} Q_i \mod p$.
- C asks the administrators the Reg_i they have registered, sorts them by ID_{V_i} and keeps a unique copy per Reg_i.
- He publishes all the ballots $(r_i, s_i, tag, M_i, B_i)$, all registrations, and G_s.

The protocol is presented in tables 6 and 7.

Properties Claimed:

Privacy Claimed by the authors in the original paper. Sketch of proof: a ballot cannot be linked back to the voter and it is infeasible to extract the voter's ID from the ballot.

Individual Verifiability Claimed by the authors in the original paper. Sketch of proof: The voter must check that his ballot is indeed in the published list.

Universal Verifiability Anyone can count the ballots in the voting stage once they have been published.

Robustness As administrators and scrutineers are several, it is improbable to coerce many (at least more than the thresholds) of them. As the collector only decrypts and counts the votes, it is easy to verify his action in checking the lists he published (by the voter or any observer).

	V_i	Ad_j	$Scru_k$	C
Initialization Stage	$\boxed{e_{V_i}}$	$\boxed{e_{Ad_j}}$	$\boxed{e_{Scru_k}}$	$\boxed{(e_C, \sigma(e_C, d_C)), (tag, \sigma(tag, d_C)), (n, \sigma(n, d_C)), (\tau, \sigma(\tau, d_C)), (\varsigma, \sigma(\varsigma, d_C)), (\mu, \sigma(\mu, d_C)), (\lambda, \sigma(\lambda, d_C)), (ID_{V_i}, \sigma(ID_{V_i}, d_C)), (ID_{Ad_j}, \sigma(ID_{Ad_j}, d_C)), (ID_{Scru_k}, \sigma(ID_{Scru_k}, d_C)), (\zeta, \sigma(\zeta, d_C)), (h, \sigma(h, d_C))}$ $p, q \leftarrow \mathcal{P}, q \vert (p-1), \rho /$ $(\gcd(\rho, p) = 1, \rho \neq 1)$, $g \stackrel{\text{def}}{=} \rho^{(p-1)/q} \mod p$ $\boxed{(p, \sigma(p, d_C)), (q, \sigma(q, d_C)), (g, \sigma(g, d_C))}$
Key Generation Stage		$f_i(x) \stackrel{\text{def}}{=} \sum_{k=0}^{\mu-1} f_{i,k} x^k /$ $f_{i,0} = z_i$ $\psi_{i,k} \stackrel{\text{def}}{=} g^{f_{i,k}} \mod p$ $\sigma(h(\psi_{i,k}), d_{Ad_i})$ $C_i \stackrel{\text{def}}{=} \xi_{\gamma_i}(\psi_{i,0})$ $\sigma(h(C_i), d_{Ad_i})$ $(C_i, \sigma(h(C_i), d_{Ad_i}))$ $(\psi_{i,k}, \sigma(h(\psi_{i,k}), d_{Ad_i}))$ `sent to` Ad open C_i $\delta_{i,j} \stackrel{\texttt{def}}{=} f_i(x_j) \mod q,$ $\sigma(h(\delta_{i,j}), d_{Ad_i})$ `sent to` Ad $y \stackrel{\text{def}}{=} \prod_{l=1}^{n} y_l = \prod_{l=1}^{n} \psi_{l,0} \mod p$, $\sigma(h(y), d_{Ad_i})$, $\phi_{j,i} \stackrel{\text{def}}{=} g^{\delta_{j,i}} \mod p, 1 \leq j \leq n$, $\sigma(h(\phi_{j,i}), d_{Ad_i})$ $\left(\sigma\left(h(y), d_{Ad_i}\right)\right)$, $\left(\phi_{j,i}, \sigma\left(h(\phi_{j,i}), d_{Ad_i}\right), \forall j \in [1, n]\right)$ `sent to` Ad $\boxed{y, y_j, \phi_{l,j}, 1 \leq l, j \leq n}$	$\theta_j(x) \stackrel{\text{def}}{=} \sum_{i=0}^{\lambda-1} \theta_{j,i} x^i \mod q$ where $\theta_{j,0} = a_j$ $F_{j,i} \stackrel{\text{def}}{=} g^{\theta_{j,i}} \mod p$ $\sigma\left(h(F_{j,i}), d_{Scru_j}\right)$ for $0 \leq i \leq \lambda - 1$ $\left(F_{j,i}, \sigma\left(h(F_{j,i}), d_{Scru_j}\right)\right)$ `sent to` $Scru$ $\epsilon_{j,i} \stackrel{\texttt{def}}{=}$ $\theta_j(y_i), \sigma\left(h(\epsilon_{j,i}), d_{Scru_j}\right)$ `sent to` $Scru$ $\epsilon_j \stackrel{\text{def}}{=} \sum_{i=1}^{\varsigma} \epsilon_{i,j}$ $G_p \stackrel{\text{def}}{=} \prod_{i=1}^{\varsigma} F_{i,0}$ $\boxed{\sigma\left(h(F_{j,i}), d_{Scru_j}\right)}$	

Element published, `sent`, **`sent through anonymous channel`**.

Table6. Juang, Lei and Yu's scheme - part 1. Next: table 7

	V_i	Ad_j	$Scru_k$	C
Registration Stage	$\eta_i \overset{\$}{\leftarrow} \mathbb{Z}$ $B_i \overset{\text{def}}{=} v_i \parallel \eta_i \parallel h(v_i \parallel \eta \parallel tag)$ $R_i, \varphi_i, r_i' \overset{\$}{\leftarrow} \mathbb{Z}$ $EB_i \overset{\text{def}}{=} (P_i \parallel Q_i) \overset{\text{def}}{=}$ $\left(g^{r_i'} \parallel \left(\left(G_p^{r_i'}\right) B_i \mod p\right)\right)$ $H_i \overset{\text{def}}{=} \zeta\left(ID_{V_i} \parallel R_i\right)$ $M_i \overset{\text{def}}{=} H_i \parallel tag \parallel EB_i$ $Reg_i \overset{\text{def}}{=}$ $\sigma\left(h\left(\varphi_i \parallel tag \parallel ID_{Ad_j}\right), d_{V_i}\right) \parallel$ $\varphi_i \parallel ID_{Ad_j} \parallel ID_{V_i}$ $\alpha_i \overset{\$}{\leftarrow} \mathbb{Z}_q\ \beta_i \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ $r_{i,j} \overset{\text{def}}{=} g^{\alpha_i} \widehat{r_{i,j}}^{\beta_i} \mod p$ $r_i \overset{\text{def}}{=} M_i \prod_{k=1}^{\mu} r_{i,k} \mod p$ $\widehat{m_i} \overset{\text{def}}{=} \beta_i^{-1} r_i \mod q$ $\mathfrak{R}_i \overset{\text{def}}{=}$ $\sigma\left(h\left(\widehat{m_i} \parallel Reg_i\right), d_{V_i}\right) \parallel \widehat{m_i} \parallel$ Reg_i sent to Ad $s_{i,j} \overset{\text{def}}{=} \widehat{s_{i,j}} \beta_i + \alpha_i \mod q$ $g^{-s_{i,j}} y_j^{r_i} r_{i,j} \overset{?}{=}$ $\left(\prod_{l=\mu+1}^{n} (\phi_{l,j})\right)^{\left(\prod_{k=1,k\neq j}^{\mu}\left(\frac{-x_k}{x_j - x_k}\right)\right)(-r_i)}$ $\mod p, 1 \le j \le \mu$ $s_i \overset{\text{def}}{=} \prod_{j=1}^{\mu} s_{i,j} \mod q$	$k_{i,j} \overset{\$}{\leftarrow} \mathbb{Z}_q$ $\widehat{r_{i,j}} \overset{\text{def}}{=} g^{k_{i,j}} \mod p,$ $\sigma\left(h\left(\widehat{r_{j,i}} \parallel tag\right), d_{Ad_j}\right)$ sent to V_i saving back Reg_i in the registration database $\widehat{s_{i,j}} \overset{\text{def}}{=}$ $\widehat{m_i}\left(z_j + \sum_{l=\mu+1}^{n}\left(f_l(x_j)\left(\prod_{\substack{k=1 \\ k\neq j}}^{\mu}\left(\frac{-x_k}{x_j - x_k}\right)\right)\right)\right) +$ $k_{i,j} \mod q$ sent to V_i		
Counting Stage	(r_i, s_i, tag) **sent to** C			$M_i \overset{\text{def}}{=} g^{-s_i} y^{r_i} r_i$ $\mod p \equiv H_i \parallel tag \parallel EB_i \mod p$ records (r_i, s_i, tag, M_i)
Voting Stage	$(r_i, s_i, tag, M_i) \overset{?}{\in} L$			$\boxed{L \overset{\text{def}}{=} \{(r_1, s_1, tag, M_1) \cdots\}}$ asks λ $Scru$ for ε_j $G_s \overset{\text{def}}{=}$ $-\sum_{j=1}^{\lambda}\left(\varepsilon_j \prod_{i=1, i\neq j}^{\lambda} \frac{-x_j}{(x_j - x_i)}\right)$ $\mod q$ $B_i = (P_i)^{G_s} Q_i \mod p$ asks Ad Reg_i $(r_i, s_i, tag, M_i, B_i)$, registrations, G_s

Element published, sent, **sent through anonymous channel**.

Table7. Juang, Lei and Yu's scheme - part 2. Previous: table 6

Fairness Claimed by the authors in the original paper. Sketch of proof: the public information (before the publication stage) is of no use for the eligible voters.

Checked:

Democracy Proven by the authors in the original paper. Sketch of the proof: the voter can construct only one vote with the help of the administrators. Invalid or double votes are excluded.

Correctness Proven by the authors in the original paper. Sketch of the proof: The scrutineers (or only part of them) can cooperate in order to reconstruct the secret key.

Attacks:

Receipt-Freeness Attack proposed by the authors in the original paper. η_i is a receipt for the voter i.

Here, we describe some schemes which use mix-net. Usually, encrypted votes are shuffled by authorities before being decrypted and count. Sometimes, this method is used along with homomorphic encryption, in which case the authorities tally the votes before deciphering the result.

4.7 Prêt-à-Voter

Authors: Peter RYAN, Steve SCHNEIDER, and David CHAUM (2005).
Primitives used: Mix-nets, Zero-knowledge proof, Re-encryption.
Requirements: There exists a secure probabilistic encryption scheme.

Summary: This scheme proposed in [CRS05] is composed in 5 stages: Roughly speaking, "Prêt-à-Voter" is a protocol where the candidate list is permuted as random. The permutation function will be encoded successively by the public keys of mix-servers (named tellers) in order to construct an "onion" which is the final encoded value.

Then, onions are printed on ballots and each ballot is almost unique because of the probabilistic encryption used (that uses some randomly generated values for each layer of each onion).

Voters can tick the candidates chosen and cast the tick and the onion without the list.

Once the ballot is cast, tellers decrypt the layer of the wrapping for which they have the secret key.

Finally, decrypted ballots can be tallied to get the results.

Protocol Description All communications are operated via public bulletin board.

Initialization of Tellers: Assume that this protocol has n tellers as authorities ($2 \leqslant n$). For each teller, a key-pairs generator selects two different efficient key-pairs. We denote the tellers T_i, $i = 1..n$, and the T_i's key-pairs: $(pk_{T_{2i-1}}, sk_{T_{2i-1}})$ and $(pk_{T_{2i}}, sk_{T_{2i}})$.

Construction of Ballots:

- A random bit generator will produce $2n$ bit strings (sufficiently large) called seeds. We denote the seeds g_i, $i = 1..2n$.
- For $i = 1..2n$, we compute $d_i \stackrel{\text{def}}{=} H(g_i) \mod L$, with H a hash function and v= size of candidate list.
- Then, we take the candidate list in a predetermined order and we compute the "cyclic offset" θ (i.e. the order of the cyclic permutation of the list) as follows: $\theta = \sum_{i=1}^{2n} d_i \mod v$.

– The onion is constructed as follows: A random value D_1 is generated for the onion's first layer and each successive encryption is defined as: $D_{i+1} \stackrel{\text{def}}{=} [g_i, D_i]_{pk_{T_i}}$. Thus, the onion is defined as: $Onion \stackrel{\text{def}}{=} D_{2n+1}$. Practically, the encryption of the cyclic permutation of the candidate list can be represented as follows: $\left[g_{2n}, \left[g_{2n-1}, \cdots \left[g_2, [g_1, D_1]_{pk_{T_1}}\right]_{pk_{T_2}} \cdots\right]_{pk_{T_{2n-1}}}\right]_{pk_{T_{2n}}}$

Ballots' Casting: The ballots consist of a left hand column listing the candidates in the order determined by θ and a right hand column in which the voter will cross his choice and the onion is printed. Once the voter has crossed his choice, the two columns are separated and the right one is cast, and given back to the voter (after the ballot is scanned) as a receipt. We denote by r_{2n} the crossed case number.

Tellers' Manipulation: For each ballot cast, the first teller in the mix-net (i.e. T_n) receives a couple (r_{2n}, D_{2n}) where r_{2n} is the choice number and D_{2n} the onion. It applies its secret key and the hash function on the couple to remove the first layer of the onion as follows:

– $(r_{2n}, D_{2n}) \xrightarrow{sk_{T_{2n}}} (g_{2n-1}, D_{2n-1})$
– $d_{2n-1} = H(g_{2n-1}) \mod L$
– $r_{2n-1} = r_{2n} - d_{2n-1} \mod L$

T_n obtains a new couple (r_{2n-1}, D_{2n-1}).

Once these operations are made for all the received ballots, T_n makes a permutation of the ballot set and repeats the former process on the resulting permuted pairs using its second secret key $sk_{T_{2n-1}}$. After having done a new permutation, the resulting ballot set consists of pairs (r_{2n-2}, D_{2n-2}) and is sent to the following tellers (i.e. T_{n-1}).

The sent ballot set is proven to be correct by the teller T_n to T_{n-1} by an interactive zero knowledge proof appearing like those used for isomorphic graphs.

All these manipulations are performed by all the tellers and the last outputs a pairs (r_1, D_1) where r_1 is the choice number in the candidate list.

Tally Stage: Once the tellers' manipulations are done, the votes are decrypted and can be simply tallied for each candidate.

The protocol is presented in table 8.

Properties Checked:

Privacy Claimed in the original paper. Ballots are generated, printed and allocated randomly to voters, so, there exists no linkage between voters and ballots. Besides, rather than encrypting directly the vote value, the protocol encrypts the order of the candidate list.

Verifiability Both claimed in the original paper:

Universal The association between the bulletin board and the zero knowledge proofs of tellers gives confidence in the ballot sets integrity to observers.

Individual All the cast ballots are visible on a bulletin board, so each voter can check that his vote has been correctly cast by comparing the onion's value. In addition, universal verifiability guarantees to the voter that, if his ballot was well cast at the beginning of the protocol, then it is part of the tallied ballots.

Receipt-freeness Claimed in the original paper. The receipts kept by voters is only the left hand of the ballot and even if the voter shows its onion to a coercer, this cannot prove his vote.

communication via public bulletin board, n is the number of tellers, L is the number of candidates.

	V_i	T_j	A
Tellers' Initialization		Sets up 2 key-pairs $(pk_{T_{2j-1}}, sk_{T_{2j-1}})$, $(pk_{T_{2j}}, sk_{T_{2j}})$, $\boxed{pk_{T_{2j-1}}, pk_{T_{2j}}}$	$k = 1, \cdots, 2n$, $g_k \overset{\$}{\leftarrow} \mathbb{Z}$, $d_k \overset{\text{def}}{=} H(g_k) \mod L$, $\theta = \sum_{k=1}^{2n} d_k \mod L$, $D_{k+1} \overset{\text{def}}{=} [g_k, D_k]_{pk_{T_k}}$ $Onion \overset{\text{def}}{=} D_{2n+1}$ Ballot: 2 parts, left with θ permutation of candidates, right with boxes and D_{2n+1}
Ballots Casting	ticks his choice (r_{2n}), separates the two columns, casts the right-hand side column		
Tellers' Manipulation		${}^{T_{j+1}}\texttt{receives:}$ (r_{2j}, D_{2j}) $(g_{2j-1}, D_{2j-1}) = \{D_{2j}\}_{sk_{T_{2j}}}$ $d_{2j-1} = H(g_{2j-1}) \mod L$ $r_{2j-1} = r_{2j} - d_{2j-1} \mod L$, shuffles ballots, $(g_{2(j-1)}, D_{2(j-1)}) = \{D_{2j-1}\}_{sk_{T_{2j-1}}}$ $d_{2(j-1)} = H(g_{2(j-1)}) \mod L$ $r_{2(j-1)} = r_{2j-1} - d_{2(j-1)} \mod L$, (r_{2j-2}, D_{2j-2}) + $\texttt{proof sent}^{T_{j-1}}$	
Tally Stage		Votes are counted.	

Element $\texttt{sent}^{\texttt{Receiver}}$, ${}^{\texttt{sender}}\texttt{receives:}$.

Table8. Prêt-à-Voter

Robustness Claimed in the original paper. Tellers have different key pairs to construct and decrypt onions, so if there are one or more cheating tellers, the mix-net will not work. Moreover, there exists a threshold version of "Prêt-à-Voter" that allows to the protocol to go on even if there is a reasonable number of corrupted tellers.

Fairness Claimed in the original paper. Ballots are not counted before the end of the voting stage.

Attacks:

Coercion-Resistance Randomization attack can be operated on this scheme: an attacker can force a voter to tick, let's say the third candidate, and then show its receipt to the attacker.

4.8 Lee, Boyd and al. Protocol

Authors:
Byoungcheon LEE, Colin BOYD, Ed DAWSON, Kwangjo KIM, Jeongmo YANG and Seungjae YOO (2003)
Primitives used: Mixnet, re-encryption, designated-verifier re-encryption proof (DVRP) and threshold ElGamal encryption.
Requirements: One assumption in this protocol is that a Tamper Resistant Randomizer (TRR is a secured hardware device) is owned by each voter. This device is used to compute re-encryption and designated-verifier re-encryption proof (DVRP).

Summary: This scheme proposed in [LBD+03] provides a mixnet-based voting protocols which achieves the receipt-freeness property.

Protocol Description: This protocol combines typical mixnet voting protocol and randomization technique. It is composed of five stages:

System setup Stage The authority (A) prepares the vote by initializing threshold ElGamal encryption scheme, publishes the list of candidates and other required information. Then n taillers T_k generate key for (t, n)-threshold verifiable secret sharing scheme and publish the public key $h = g^s$ where s is the secret key shared by the n taillers, and g as usual an element of the group used.

Registration Stage The voter (V_i) registers to A . A checks V_i eligibility and TRR$_i$, and publishes the list of qualified voters with the certificates of voters and TRRs.

Voting Stage V_i and associated TRR$_i$ compute the final encryption ballot in three steps with the interactive protocol described in Table 9.

- V_i chooses $\alpha \in_R Z_q^*$, his ballot, computes $(x, y) = (g^\alpha, h^\alpha m_i)$ and sends (x, y) to TRR$_i$.
- TRR$_i$ randomizes (x, y) and gets $(x_f, y_f) = (xg^\beta, yh^\beta)$, where β is a secure random. He also generates a DVRP (c, r, t, u) for (x_f, y_f) where $k, r, t \in_R Z_q$, H is an hash function, $h_V = g^{s_V}$ (s_V is the secret key of the voter), $(a, b) = (g^k, h^k)$, $d = g^r h_V^t$, $c = H(a, b, d, x_f, y_f)$ and $u = k - \beta(c + r)$. Finally TRR$_i$ sends (c, r, t, u) and $Sig_{TTR_i}(x_f, y_f)$ to V_i.
- V_i verifies the validity of DVRP by

$$c \stackrel{?}{=} H(g^u(x_f/x)^{c+r}, h^u(y_f/y)^{c+r}, g^r h_v^t, x_f, y_f)$$

Finally V_i double signs the final ballot (x_f, y_f) and posts the following message on the bulletin board:

$$Sig_{V_i}(Sig_{TRR_i}(x_f, y_f))$$

Mixing Stage Before starting the mixing stage A double checks the signed message and publishes valid ballots on the bulletin board. Then m mixer M_j shuffle the ballots using proposed mixnet protocols such as [Abe99,Nef01,Gro03,FS01]

Tallying Stage After mixing A decrypts the message with the (t, n)-threshold ElGmal decryption protocol and gets the m_i. Finally A publishes the tally of the votes.

Stage	V_i	TTR$_i$	A
Setup			List of candidates, public key $h = g^s$ generates secret keys
Registration	Registration to A		checks TRR$_i$ List of V_i, certificates of V_i certificates of TRR$_i$
Voting	$m_i, \alpha \in_R Z_q^*$, $(x, y) = (g^\alpha, h^\alpha m_i)$ `sent to` TTR$_i$ $c \stackrel{?}{=} H(g^u(x_f/x)^{c+r}, h^u(y_f/y)^{c+r}, g^r h_v^t, x_f, y_f)$	β random, $k, r, t \in_R Z_q$ $(x_f, y_f) = (xg^\beta, yh^\beta)$, $h_V = g^{s_V}$, $(a, b) = (g^k, h^k)$, $d = g^r h_V^t$, $c = H(a, b, d, x_f, y_f)$ $u = k - \beta(c + r)$ and (c, r, t, u), $Sig_{TTR_i}(x_f, y_f)$ `sent to` V_i	
Mixing			checks $Sig_{V_i}(Sig_{TRR_i}(x_f, y_f))$ shuffles votes using a mix-net
Tallying			results

Element published, `sent`

Table9. Lee, Boyd and al. protocol [LBD$^+$03]

Property Checked:

Receipt-freeness Claimed by the authors in the original paper. The voter had lost his random seed using the TRR and he is convinced by the DVRP of the re-encryption performed by the TRR, by consequence he cannot give any receipt of his vote to a coercer. This property was proved on a simplified version of the protocol using formal method in [DKR06]. In the same paper the authors also show that there is an attack against without any assumption about the implementation of the protocol. Note that the coercion-resistance property is a stronger property than receipt-freeness (coercion-resistance implies receipt-freeness).

Properties Claimed:

Privacy Claimed by the authors in the original paper. The privacy of the voter depends on the security of the mixnet, and the secret of the vote is preserved assuming we have at least $n - t + 1$ honest taillers.

Prevention of double voting Claimed by the authors in the original paper. Double voting is prevented because the list of eligible voters is public and the voters participate in the protocol in an authenticated way (using their signatures and TTR's signatures).

Universal verifiability Claimed by the authors in the original paper. Since all messages in each stage are published on the bulletin board then any observer can verify the vote result.

Fairness Claimed by the authors in the original paper. With at least $n - t + 1$ honest taillers, no partial tally is revealed then fairness is guaranteed.

Robustness Claimed by the authors in the original paper. Partial failure of some voters can be detected and does not affect the election, and also that mixing and tallying phases are robust against failures of the servers

Extensions In [ALBD04] an efficient modified version of the Lee, Boyd and al. protocol is proposed. The efficiency is due to the fact that they use a modification of Golle et al.'s optimistic mixnet [GZJ$^+$02]. The scheme is also changed because now the administrator provides both the re-encryption service (using a TRR) and mixing service, and the communication between the voters and the Administrator is performed over an untappable channel.

4.9 Weber's scheme

Author: Stefan WEBER (2006).
Primitives used: Mix-nets, secret sharing, zero-knowledge proof.
Requirements: Untappable and anonymous channels, ElGamal cryptosystem is secure if discrete logarithm is intractable.

Summary: This scheme proposed in [Web06] is composed in 4 stages: during the initialization stage, authorities set up ElGamal threshold cryptosystem, and publish the public parameters. Then during registration, each voter identifies himself with the authorities and gets a credential. In the voting stage, voters cast their encrypted vote. At last, the ballots are verified, mixed, decrypted and counted.

Several authorities are used in this scheme: administrators (Ad) who register voters and issue credentials, tailliers (T), who decrypt and count the ballots, mixers (M), who mix the ballots, and managers (Ma), who operate the bulletin boards[2].

[2] In the original paper, the scheme is a remote electronic voting, and the bulletin boards are servers.

Protocol Description The list of the candidates must be published before the Voting Stage begins. Stages must be synchronised.

Initialization stage: All authorities ($Ad,\ T,\ M$ - say A for a general authority) set up El Gamal (t, n) threshold cryptosystem.

- Each A_i chooses $x_i \xleftarrow{\$} G_q$
- A_i chooses $f_i(X) \xleftarrow{\$} G_q[X]$ of degree $t-1$ such that $f_{i,0} \stackrel{\text{def}}{=} f_i(0) \stackrel{\text{def}}{=} x_i$. The coefficient of degree k is noted as $f_{i,k}$.
- A_i publishes $F_{ij} \stackrel{\text{def}}{=} g^{f_{i,j}} \mod p, \forall j \in \{0, \cdots, t-1\}$ on bulletin board $\mathcal{BB}_0$.
- A_i sends secretly $s_{i,j} \stackrel{\text{def}}{=} f_i(j) \mod q$ to A_j for all $j \in \{0, \cdots, n\}$
- A_i checks what he received by checking if $g^{s_{ji}} \stackrel{?}{=} \prod_{l=0}^{t-1} F_{jl}^{i^l} \mod p$. If it fails, A_i blames the corresponding A_j.
- If more than $t-1$ complain about A_j, this authority is discredited from the remaining steps of the election.
- A_i computes his share of the private key: $sk_i \stackrel{\text{def}}{=} \sum_{j=1}^{n} s_{ji} \mod q$
- A_i computes the public key $pk_A \stackrel{\text{def}}{=} \prod_{i=1}^{n} F_{i0} = \prod_{i=1}^{n} g^{x_i} \mod p$
- A_i is committed to sk_i by $\rho_i \stackrel{\text{def}}{=} g^{sk_i} = \prod_{j=1}^{n} g^{s_{ji}}$. ρ_i is published on $\mathcal{BB}_0$.
- (p, q, g, pk_A) are published on $\mathcal{BB}_0$.
- The first steps of this stage are rerun to get two more secret hash-keys h, e with shares h_i, e_i.

Registration stage: Each voter (V) gets a credential from the administrators (Ad).

- Administrators generate a unique credential $\sigma_i \in G_q$ for each eligible voter V_i and send it to him through an untappable channel.
- A list of all registered voters, along with their encrypted credentials $([\sigma_i]_{pk_A})$ is published on $\mathcal{BB}_1$.

Voting stage: V_i submits his encrypted vote with his encrypted credential, along with a zero-knowledge proof of the credential, and an encrypted time stamp. (H is a hash function)

- V_i chooses his vote v_j from a set of possible votes $\{v_1, \cdots, v_L\}$ and encrypts it $k \xleftarrow{\$} \mathbb{Z}, B_i \stackrel{\text{def}}{=} (x, y) \stackrel{\text{def}}{=} (g^k, pk_A^k v_j)$
- V_i constructs P_i a non-interactive zero-knowledge proof of validity of B_i. To prevent attackers from reusing his proof, he can use bits of the time stamp when he uses the hash function H.
  1. he chooses $w \xleftarrow{\$} G_q$, and computes $a_j \stackrel{\text{def}}{=} g^w \mod p, b_j \stackrel{\text{def}}{=} pk_A^w \mod p$
  2. for each $u \neq j$, he chooses $r_u, d_u \xleftarrow{\$} G_q$, and computes $a_u \stackrel{\text{def}}{=} g^{r_u} x^{d_u} \mod p, b_u \stackrel{\text{def}}{=} pk_A^{r_u} (y/v_u)^{d_u} \mod p$
  3. he computes $c \stackrel{\text{def}}{=} H(a_1, b_1, \cdots, a_L, b_L) \mod q$
  4. he computes $d_j \stackrel{\text{def}}{=} c - \sum_{u \neq j} d_u \mod q, r_j \stackrel{\text{def}}{=} w - k d_j \mod q$
  5. $P_i \stackrel{\text{def}}{=} (a_1, b_1, d_1, r_1, \cdots, a_L, b_L, d_L, r_L)$
- V_i encrypts σ_i: $k' \xleftarrow{\$} \mathbb{Z}, C_i \stackrel{\text{def}}{=} (x_i', y_i') \stackrel{\text{def}}{=} (g^{k'}, pk_A^{k'} \sigma_i)$
- V_i improves P_i with a non-interactive zero-knowledge proof of knowledge of σ_i:
  1. he chooses $w' \xleftarrow{\$} G_q$, computes $D \stackrel{\text{def}}{=} g^{w'} \mod p$
  2. he computes $c' \stackrel{\text{def}}{=} H(D, x_i', g) \mod q$
  3. he computes $r \stackrel{\text{def}}{=} w' + c'k' \mod q$
  4. he adds (D, c, r) to P_i
- V_i encrypts a time stamp $[\tau_i]_{pk_A}$.
- V_i publishes $(B_i, P_i, [\tau_i]_{pk_A}, C_i)$ in $\mathcal{BB}_2$. (As he uses time stamp, he may publish several tuples.)

Tallying stage: Tailliers (T) verify the votes (by checking the proofs), and publish the valid ballots. Mixers (M) mix and re-encrypt the ballots. Then multiple ballots are detected and eliminated according to the time stamp. At last, ballots are decrypted and counted.

1. T_j checks the proofs attached with the ballots. Anytime a proof is wrong, the ballot is discarded. Verification of the validity proof:
   (a) $c \stackrel{?}{=} \sum_i d_i \mod q$
   (b) $a_i \stackrel{?}{=} g^{r_i} x^{d_i} \mod p$
   (c) $b_i \stackrel{?}{=} pk_A^{r_i}(y/m_i) \mod p$
2. Verification of the proof of knowledge of the credential: T_j checks that $g^r \stackrel{?}{=} Dx'^{c'}$
3. Valid ballots are published in $\mathcal{BB}_3$, without P_i
4. M_k receives from M_{k-1} a batch of encrypted ballots, except for M_0 who works on $\mathcal{BB}_3$. M_k re-encrypts each ballot $E_{i,k} \stackrel{\text{def}}{=} [E_{i,k-1}]_{pk_A}$ ($E_{i,0} \stackrel{\text{def}}{=} \left[(B_i, [\tau_i]_{pk_A}, C_i)\right]_{pk_A}$), and then permutes them before handing them to M_{k+1} (publishing them on $\mathcal{BB}_4$ for the last M), with a zero-knowledge proof that the permutation is correct.
5. M_k computes $bx_k \stackrel{\text{def}}{=} x'^{h_k}_i$, $by_k \stackrel{\text{def}}{=} y'^{h_k}_i$ and publishes it on $\mathcal{BB}_5$ along with a zero-knowledge proof that $\log_g \rho_{h_k} = \log_x bx_k$, $\log_g \rho_{h_k} = \log_y by_k$
6. A subset Λ of t M computes $bx = x'^h_i = \prod_{k\in\Lambda} bx_k^{\lambda_{k,\Lambda}} \mod p$, where $\lambda_{k,\Lambda} \stackrel{\text{def}}{=} \prod_{l\in\Lambda\setminus\{k\}} \frac{l}{l-k} \mod q$. By the same way, M compute $by = y'^h_i$
7. M_k computes $x'_{i,k} = bx^{sk_k}$ and publishes it on $\mathcal{BB}_5$ along with a non-interactive zero-knowledge proof that $\log_g \rho_k = \log_{bx} x'_{i,k}$
8. A subset Λ of t M computes $\sigma_i^h = \frac{by}{\prod_{l\in\Lambda\setminus\{k\}} x'^{\lambda_{k,\Lambda}}_{i,l}} \mod p$
9. A subset Λ of t M decrypts $[\tau_i]_{sk_A}$. (σ_i^h, τ_i) are published on $\mathcal{BB}_6$, along with their ballots.
10. M detects multiple ballots according to the σ_i^h (using a hash-table). If collision is found (for the same σ_i), only the last cast ballot (according to $T - i$) is stored. At last, one ballot (B_i, C_i) per credential is saved and published in $\mathcal{BB}_7$.
11. Ballots are mixed once more like in the 4 step. The output is published on $\mathcal{BB}_8$
12. Credential from $\mathcal{BB}_2$ are also mixed like in the 4 step (with a different mix-net) and the output $(C_{\pi(i)})$ is published in $\mathcal{BB}_9$.
13. σ_i^e is computed, like in the steps 5 to 8 and published with their ballots in $\mathcal{BB}_{10}$.
14. $\sigma^e_{\pi(i)}$ is computed from $\mathcal{BB}_9$ and is published in $\mathcal{BB}_{11}$. Then it is put in a hash-table.
15. For each ballot on $\mathcal{BB}_{10}$, M check if σ_i^e is in the hash-table. If not, the ballot is discarded. Otherwise, the ballot is published in $\mathcal{BB}_{12}$
16. A subset Λ of t T decrypts the ballots in $\mathcal{BB}_{12}$ and post it on $\mathcal{BB}_{13}$. They then tally the votes and tally the results in $\mathcal{BB}_{14}$.

    The protocol is presented in table 10, 11 and 12.

Properties Checked:

Correctness Claimed by the author in the original paper. If the hash functions used are collision-free (as SHA-2), then the result computed by the protocol is correct.

Robustness Claimed by the author in the original paper. As the authorities must be at least t to compute anything damaging, the protocol is robust up to $t - 1$ malicious authorities.

Universal Verifiability Claimed by the author in the original paper. As each step is published on a public bulletin board with zero-knowledge proof, this scheme achieves universal verifiability.

	V_i	Ad_j	T_j	M_j
Initialization Stage		$x_j \xleftarrow{\$} G_q \quad f_j(X) \xleftarrow{\$} G_q^{t-1}[X]/f_{j,0} \stackrel{\text{def}}{=} f_j(0) \stackrel{\text{def}}{=} x_j;$ $\boxed{F_{jk} \stackrel{\text{def}}{=} g^{f_{j,k}} \mod p, \forall k \in \{0,\cdots,t-1\}}$ in $\mathcal{BB}_0$ $s_{j,k} \stackrel{\textbf{def}}{=} f_j(k) \mod q$ **sent to** A_k **for all** $k \in \{0,\cdots,n\}$ $g^{s_{kj}} \stackrel{?}{=} \prod_{l=0}^{t-1} F_{kl}^{j^l} \mod p \quad sk_j \stackrel{\text{def}}{=} \sum_{k=1}^{n} s_{kj} \mod q$ $pk_A \stackrel{\text{def}}{=} \prod_{i=1}^{n} F_{i0} = \prod_{i=1}^{n} g^{x_i} \mod p \quad \rho_i \stackrel{\text{def}}{=} g^{sk_i}$ $\boxed{(p,q,g,pk_A),\rho_i}$ in $\mathcal{BB}_0$ $a_j \xleftarrow{\$} G_q \quad \theta_j(X) \xleftarrow{\$} G_q^{t-1}[X]/\theta_{j,0} \stackrel{\text{def}}{=} \theta_j(0) \stackrel{\text{def}}{=} a_j$ $\boxed{\vartheta_{jk} \stackrel{\text{def}}{=} g^{\theta_{j,k}} \mod p, \forall k \in \{0,\cdots,t-1\}}$ in $\mathcal{BB}_0$ $p_{j,k} \stackrel{\textbf{def}}{=} \theta_j(k) \mod q$ **sent to** A_k **for all** $k \in \{0,\cdots,n\}$ $\theta^{p_{kj}} \stackrel{?}{=} \prod_{l=0}^{t-1} \vartheta_{kl}^{j^l} \mod p \quad h_j \stackrel{\text{def}}{=} \sum_{k=1}^{n} s_{kj} \mod q \quad b_j \xleftarrow{\$} G_q$ $\epsilon_j(X) \xleftarrow{\$} G_q^{t-1}[X]/\epsilon_{j,0} \stackrel{\text{def}}{=} \epsilon_j(0) \stackrel{\text{def}}{=} b_j$ $\boxed{\varepsilon_{jk} \stackrel{\text{def}}{=} g^{\epsilon_{j,k}} \mod p, \forall k \in \{0,\cdots,t-1\}}$ in $\mathcal{BB}_0$ $p_{j,k} \stackrel{\textbf{def}}{=} \epsilon_j(k) \mod q$ **sent to** A_k **for all** $k \in \{0,\cdots,n\}$ $\epsilon^{p_{kj}} \stackrel{?}{=} \prod_{l=0}^{t-1} \varepsilon_{kl}^{j^l} \mod p \quad e_j \stackrel{\text{def}}{=} \sum_{k=1}^{n} s_{kj} \mod q$		
Registration Stage		$\sigma_i \in G_q$ **sent to** V_i $\boxed{([\sigma_i]_{pk_A})}$ in $\mathcal{BB}_1$		

Element published in $\mathcal{BB}_n$, `sent`, **sent through untappable channel**.

Table10. Weber's Scheme - Part 1. Next: 11

$\|$ denotes concatenation operator.

	V_i	Ad_j	T_j	M_j
Voting Stage	$k \xleftarrow{\$} \mathbb{Z}, \quad B_i \stackrel{\text{def}}{=} (x,y) \stackrel{\text{def}}{=} (g^k, pk_A^k v_j)$ $w \xleftarrow{\$} G_q \quad a_j \stackrel{\text{def}}{=} g^w \mod p,$ $b_j \stackrel{\text{def}}{=} pk_A^w \mod p \quad \forall u \neq j, r_u, d_u \xleftarrow{\$} G_q$ $a_u \stackrel{\text{def}}{=} g^{r_u} x^{d_u} \mod p,$ $b_u \stackrel{\text{def}}{=} pk_A^{r_u} (y/v_u)^{d_u} \mod p$ $c \stackrel{\text{def}}{=} H(a_1,b_1,\cdots,a_L,b_L) \mod q$ $d_j \stackrel{\text{def}}{=} c - \sum_{u \neq j} d_u \mod q$ $r_j \stackrel{\text{def}}{=} w - kd_j \mod q$ $P_i \stackrel{\text{def}}{=} (a_1,b_1,d_1,r_1,\cdots,a_L,b_L,d_L,r_L)$ $k' \xleftarrow{\$} \mathbb{Z}, \quad C_i \stackrel{\text{def}}{=} (x_i',y_i') \stackrel{\text{def}}{=} (g^{k'}, pk_A^{k'} \sigma_i)$ $w' \xleftarrow{\$} G_q, D \stackrel{\text{def}}{=} g^{w'} \mod p$ $c' \stackrel{\text{def}}{=} H(D,x_i',g) \mod q$ $r \stackrel{\text{def}}{=} w' + c'k' \mod q \quad P_i \stackrel{\text{def}}{=} P_i \| (D,c,r)$ $\boxed{(B_i,P_i,[\tau_i]_{pk_A},C_i)}$ in $\mathcal{BB}_2$			
Tallying Stage			$c \stackrel{?}{=} \sum_i d_i \mod q$ $a_i \stackrel{?}{=} g^{r_i} x^{d_i} \mod p$ $b_i \stackrel{?}{=} pk_A^{r_i} (y/m_i) \mod p$ $g^r \stackrel{?}{=} Dx'^{c'}$ $\boxed{(B_i,[\tau_i]_{pk_A},C_i)}$ in $\mathcal{BB}_3$	

Element published in $\mathcal{BB}_n$, `sent`, **sent through untappable channel**.

Table11. Weber's Scheme - Part2. Previous: 10, next: 12

	V_i	Ad_j	T_j	M_j
Tallying Stage				$E_{i,k} \stackrel{\text{def}}{=} [E_{i,k-1}]_{pk_A}$ $\quad E_{i,k} \stackrel{\text{def}}{=} E_{\pi(i),k}$ $\lambda_{k,\Lambda} \stackrel{\text{def}}{=} \prod_{l \in \Lambda \setminus \{k\}} \frac{l}{l-k} \mod q$ $bx = x_i'^h = \prod_{k \in \Lambda} bx_k^{\lambda_{k,\Lambda}} \mod p$ $by = y_i'^h$ $\boxed{x'_{i,k} = bx^{sk_k}}$ in $\mathcal{BB}_5$ $\sigma_i^h = \frac{by}{\prod_{l \in \Lambda \setminus \{k\}} x'^{\lambda_{k,\Lambda}}_{i,l}} \mod p$ $\quad [\tau_i]_{sk_A}$ $\boxed{(\sigma_i^h, \tau_i)}$ in $\mathcal{BB}_6$ $\boxed{\text{ballot mixed}}$ in $\mathcal{BB}_8$ Credential from $\mathcal{BB}_2$ mixed $\boxed{C_{\pi'(i)}}$ in $\mathcal{BB}_9$ $\boxed{\sigma_i^e, B_i}$ in $\mathcal{BB}_{10}$ $\boxed{\sigma_{\pi'(i)}^e, B_i}$ in $\mathcal{BB}_{11}$
			$\boxed{\text{decryption}}$ in $\mathcal{BB}_{13}$	
			$\boxed{\text{tally}}$ in $\mathcal{BB}_{14}$	

Element $\boxed{\text{published}}$ in $\mathcal{BB}_n$, `sent`, **`sent through untappable channel`**.

Table12. Weber's Scheme - Part3. Previous: 11

Privacy Claimed by the author in the original paper. As the ballots are mixed and decrypted after the credentials have been removed, their is no way to link a voter to his vote.

Receipt-Freeness Claimed by the author in the original paper. A coercer has no way to know if the credential the voter used in a ballot is fake.

Democracy As the credentials are generated and stored by all the authorities, and only the last valid vote per credential is counted, a voter cannot fake a credential, nor use his credential more than once.

Fairness As the ballot are decrypted after the end of the voting stage, fairness is achieved.

Attacks:

Individual Verifiability Attack proposed by the author in the original paper. As the ballot are mixed and re-encrypted, the voter has no way to verify if his vote has been counted in the final tally.

Here, we show you some schemes based on homomorphic encryption. Usually, the encrypted votes are collected and counted before they are decrypted.

4.10 Hirt and Sako's Scheme

Authors: Martin HIRT and Kazue SAKO (2000).
Primitives used: Homomorphism encryption.
Requirements: Untappable one-way channels between authorities and voters, Properties of homomorphic re-encryption.

Summary: In [HS00] the authors propose a generic scheme for converting a voting protocol based on homomorphic encryption (with additionnal properties of the encryption function) into a receipt-free voting protocol. They improve Cramer et al. scheme described in subsection 4.14 from [CGS97].

It is based on several authorities, each of them generates a randomly ordered list with all encrypted votes, like in [SK95]. Using technique called designated-verifier proofs, proposed in [JSI96], a voter is allowed to trace his vote by pointing the cnryption of his choice. Tallying of votes is perfomed with homomorphic encryption.

Protocol Description: This is a 1 out of L scheme (i.e. the voter chooses a vote among L differents choices). Roughly, for each voter, the authorities encrypt a list of all possible votes, from which the voters select one. In fact, each authority (re)encrypts and mixes the votes, and proves (in a zero-knowledge way) the construction of the list. Then the voter selects his vote. At last, the votes are counted (in their encrypted form) and then decrypted jointly by the authorities.

The protocol needs several authorities (A_j), a cryptographic hash function (H) for the non-interactive zero knowledge proofs, and an untappable channel.

Vote Generation: The authorities set up robust threshold ElGamal encryption scheme, publish the public keys and the standard encryption of the ballots.

- A_j sets up ElGamal jointly with the other authorities: $h_j \stackrel{\text{def}}{=} g^{s_j}$, and $(p, g, h), h_j$ are published.
- $G_1, \cdots, G_L$ and $e_l{}^{(0)} \stackrel{\text{def}}{=} (1, G_l), l = 1, \cdots, L$ are published
- A_j receives: $e_l{}^{(j-1)}, l = 1, \cdots, L$, and then it chooses (randomly) $k_1, \cdots, k_L \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and $\pi_j \stackrel{\$}{\leftarrow} Perm(\{1, \cdots, L\})$
- A_j re-encrypts an mixes the ballots: $e^j_{\pi_j(l)} \stackrel{\text{def}}{=} \left[e_l^{(j-1)}\right]_{h,k_l}$
- A_j publishes a non interactive proof that it shuffled correctly (i.e. for each i, e_i^{j-1} there exists a k, e_k^j s.t. e_k^j is a re-encryption of e_i^{j-1}: assuming that e_t^j is the re-encryption of e_i^{j-1} with whiteness ξ, i.e. $e_i^{j-1} = (x, y), e_t^j = (x', y') = (g^\xi x, h^\xi y)$
  - A_j picks at random $d_1, \cdots, d_L$ and $r_1, \cdots, r_L$ and then computes: $a_i \stackrel{\text{def}}{=} \left(\frac{x_i}{x}\right)^{d_i} * g^{r_i}$ and $b_i \stackrel{\text{def}}{=} \left(\frac{y_i}{y}\right)^{d_i} * h^{r_i}$ for $i = 1, \cdots, L$)
  - A_j computes the challenge: $c \stackrel{\text{def}}{=} H(E\|a_1\| \cdots \|a_L\|b_1\| \cdots \|b_L)$ from $E \stackrel{\text{def}}{=} (x\|y\|x_1\|y_1\| \cdots \|x_L\|y_L)$ ($\|$ denotes the string concatenation operator)
  - A_j changes d_t s.t. $d_1 + \cdots + d_L = c \mod q$ and r_t s.t. $w = \xi d_t + r_t \mod q$
  - a verifier can check whether $d_1 + \cdots + d_L \stackrel{\text{def}}{=} H\left(E\|\left(\frac{x_1}{x}\right)^{d_1} g^{r_1}\| \cdots \|\left(\frac{x_L}{x}\right)^{d_L} g^{r_L}\|\left(\frac{y_1}{y}\right)^{d_1} h^{r_1}\| \cdots \|\left(\frac{y_L}{y}\right)^{d_L} h^{r_L}\right)$
- A_j sends to V_i the permutation π and a designated-verifier re-encryption proof (through an untappable channel) (per e_k^j): assuming that e_t^j is the re-encryption of e_i^{j-1} with whiteness ξ, i.e. $e_i^{j-1} = (x, y), e_t^j = (x', y') = (g^\xi x, h^\xi y)$:
  - A_j selects randomly d, w, r and computes $a \stackrel{\text{def}}{=} g^d, b \stackrel{\text{def}}{=} h^d, s \stackrel{\text{def}}{=} g^w h^r_{V_i}$.
  - A_j computes the challenge $c \stackrel{\text{def}}{=} H\left(E\|a\|b\|s\right)$ where $E \stackrel{\text{def}}{=} x\|y\|x'\|y'$.
  - A_j computes $u \stackrel{\text{def}}{=} d + \xi(c + w)$.
  - A_j sends the proof (c, w, r, u) to V_i
  - V_i tests $c \stackrel{?}{=} H\left(E\|\frac{g^u}{\left(\frac{x'}{x}\right)^{c+w}}\|\frac{h^u}{\left(\frac{y'}{y}\right)^{c+w}}\|g^w h^r_{V_i}\right)$
- V_i checks the proofs. It can publicly complain against at most $N - t$ authorities. If it does so, V_i sets $e_1^{(k)} = e_1^{(k-1)}, \cdots, e_L^{(k)} = e_L^{(k-1)}$. ($A_k$ is ignored.)

Vote Casting: V_i derives the position of his vote and publicly announces his choice.

Tallying Stage: The encrypted votes are multiplied: $(X,Y) \stackrel{\text{def}}{=} (\prod_i x_i, \prod_i y_i)$. The result is then decrypted: $W \stackrel{\text{def}}{=} \frac{Y}{X^s}$, and the result as well as a proof of correct decryption is published.

This is an 1-out-of-L scheme. The proofs used here are non interactive zero-knowledge proofs. There are N authorities, ElGamal's threshold is t.

	Voter (V_i)	Authority (A_j)
Vote Generation	not accept the proof $\Rightarrow$ $\boxed{\text{complain}}$ $\Rightarrow$ j^{th} authority ignored	ElGamal: $\boxed{(p,g,h),\quad h_j = g^{s_j}}$ $\boxed{G_1,\cdots,G_L}$, ${e_l}^{(0)} \stackrel{\text{def}}{=} (1, G_l)$ $\boxed{{e_1}^{(0)},\cdots,{e_L}^{(0)}}$ ${}^{A_{j-1}}\texttt{received: } {e_l}^{(j-1)}, l = 1,\cdots,L$ $k_1,\cdots,k_L \stackrel{\$}{\leftarrow} \mathbb{Z}_p \quad \pi_j \stackrel{\$}{\leftarrow} Perm(\{1,\cdots,L\})$ $e^j_{\pi_j(l)} \stackrel{\text{def}}{=} \left[e_l^{(j-1)}\right]_{h,k_l}$ $\boxed{\begin{array}{l}\text{proof that } \forall l \in \{1,\cdots,L\},\\ \left[e_l^{(j-1)}\right]_h \in \{e_1^{(j)},\cdots,e_L^{(j)}\}\end{array}}$ $\pi_j +$ **proof that** $e^{(j)}_{\pi_j(l)} = \left[e_l^{(j-1)}\right]_h$ **sent**^{V_i}
Vote Casting	selects $\boxed{e_v^{(N)}}$ (his vote)	
Counting Stage		$(X,Y) \stackrel{\text{def}}{=} (\prod_i x_i, \prod_i y_i)$ $W \stackrel{\text{def}}{=} \frac{Y}{X^s}$ $\boxed{\text{proof } A_j \text{ used its part of } s}$ $W = {G_1}^{T_1} \cdots {G_L}^{T_L}$

Caption: element $\boxed{\text{published}}$, ${}^{\text{sender}}\texttt{received:}$, $\texttt{sent}^{\text{Receiver}}$, **sent through untappable channel**$^{\text{Receiver}}$.

Table13. Hirt and Sako's voting scheme

This protocol is described in table 13.

Properties Claimed:

Privacy, robustness As the votes can only be decrypted by a set of at least $t+1$ authorities, if at least $t+1$ of the N authorities are honest, privacy is achieved.

Receipt-Freeness If the voter knows at least one authority not colluding with the coercer, then it is receipt-free, since the voter can forge a permutation proof to convince the coercer that he voted in the coercer's way.

Universal and Individual Verifiability Any verifier can check that the final list is correctly constructed, and that the tally has been correctly computed and decrypted. Thus, it achieves universal verifiability.

Fairness The tally is decrypted when all votes are cast, hence this protocol ensures fairness.

Attacks:

Democracy This protocol does not check, at any stage, the voter's identity. Then it doesn't handle democracy.

Coercion Resistance Attack suggested by the author in the original paper. It suffices that a coercer forces the voter to vote for a specific position (randomization attack), since his choice (for the position) is publicly published.

4.11 Rjaskova's scheme

Authors: Zuzana RJASKOVA (2002).
Primitives used: Deniable encryption, re-encryption, untappable channel, homomorphic encryption
Requirements: ElGamal is secure if logarithm is intractable.

Summary: This scheme proposed in [Rja02] is inspired by the scheme presented in subsection 4.10 from [HS00] and is composed in 3 stages: first the authorities set up a robust threshold ElGamal cryptosystem. Then the voter (V) shares his vote with the authorities (A), who re-encrypt the shares, before re-building it, and at last, tally it and decipher the result.

Protocol Description We assume that at least $t+1$ authorities are honest. λ_i denotes the Lagrange coefficient.

We stress that the author uses a modified ElGamal cryptosystem.

Initialization stage: Each authority A_j sets up a robust ElGamal cryptosystem.

Voting stage: Authorities generate shares of the voter's vote v_i using Shamir secret sharing and send it to the voter V_i. Then A_j encrypts it and proves the validity of its encryption to V_i. V_i selects the authorities he trusts (at least $t+1$). The remaining authorities combine their shares to find the encryption of V_i's vote and publish a proof that it is the encryption of a valid vote.

- A_j generates a random message s_j.
- A_j computes $B_j^0 \stackrel{\text{def}}{=} (1, G^{s_j})$, $B_j \stackrel{\text{def}}{=} \left[B_j^0\right]_{pk_j}$ and publishes B_j.
- A_j sends to V_i a designated-verifier proof that B_j is the re-encryption of B_j^0 through the untappable channel, along with s_j.
- V_i checks the proof. He can publicly complain about at most $N-t-1$ authorities, or point out $t+1$ authorities he trusts. We note A this set of authorities.
- V_i computes $B_v \stackrel{\text{def}}{=} \left[v_i - \sum_{j\in A} s_j\right]_{pk_A}$ and publishes (A, B_v)
- Trusted authorities combine their shares to calculate the encryption of v_i: $B \stackrel{\text{def}}{=} B_v \prod_{j\in A} B_j = \left[v_i - \sum_{j\in A} s_j\right]_{pk} \prod_{j\in A} [B_j]_{pk_j} = \left[v_i - \sum_{j\in A} s_j + \sum_{j\in A} s_j\right]_{pk_A}$
- A_j checks that B encrypts a valid vote, and publishes the proof.

Counting stage: The valid votes are multiplied between themselves, and then jointly decrypted.

The protocol is presented in table 14.

Properties Checked:

blabla.

	V_i	A_j
Initialization Stage		ElGamal set up.
Voting Stage	checks proof and points out $t+1$ authorities he trusts (A) $B_v \stackrel{\text{def}}{=} \left[v_i - \sum_{j \in A} s_j\right]_{pk_A}$ $\boxed{(A, B_v)}$	$s_j \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ $B_j^0 \stackrel{\text{def}}{=} (1, G^{s_j}), B_j \stackrel{\text{def}}{=} \left[B_j^0\right]_{pk_j}$ $\boxed{B_j}$ **designated-verifier proof that** $B_j = re-encryption(B_j^0), s_j$ **sent through untappable channel**^{V_i} $B \stackrel{\text{def}}{=} B_v \prod_{k \in A} B_j$ $\boxed{\text{checks that } B \text{ encrypts a valid vote}}$
Counting Stage		Count and then decrypt

Element $\boxed{\text{published}}$, `sent`, **`sent through untappable channel`**.

Table14. Rjaskova's Scheme

Privacy Claimed in the original thesis. The only way to get the voter's vote is to collect $t+1$ shares of the vote (s_j) which assumes that we can either tape the untappable channel, corrupt $t+1$ authorities or break the encryption scheme used by the authorities.

Universal Verifiability Claimed in the original thesis. Anyone can compute the encryption of a vote from the published shared encryption, multiply them and check the decryption.

Individual Verifiability Claimed in the original thesis. Except if the authority encrypts two shares in the same way, (which is negligeable), the voter can check that his vote has been published and then check the tally.

Receipt Freeness Claimed in the original thesis. The voter can adjust one share of his vote as he wants, when he is asked to reveal the shares. Then he can claim that he has voted the way the coercer wants.

Robustness $N-t-1$ authorities must be dishonest to fail the election scheme.

Fairness Only the final tally is decrypted, and getting information on votes before the decryption reduces to break ElGamal.

Attacks:

Eligibility Nowhere in the protocol the identity of the voter is checked by the authority. Although we assume that the voting stage takes place in a booth, there is no guaranty that a non-eligible party casts a valid vote, or that a voter runs the voting stage twice.

Remarks: The author proposes an implementation for untappable channels in her thesis, using deniable encryption.

4.12 Benaloh and Tuinstra's scheme (first version)

Authors: Josh BENALOH and Dwight TUINSTRA (1994).
Primitives used: Homomorphic encryption, zero knowledge proofs.

Requirements: Voting Booth (a place where voters are physically separated from outsiders. Its presence is necessary in this scheme to ensure privacy and prevent coercion), beacon (i.e. a publicly readable source of random bits).

Summary: This paper proposed in [CBT94] presents two protocols: a scaled down protocol where voters interact with a single authority, then a multiple tallying authorities protocol that improves privacy (see subsection 4.13). These protocols are based on the homomorphic properties of encryption, allowing the tally to be decrypted without doing so for each vote. Zero knowledge proofs are used in order to ensure verifiability at each step. The concept of receipt-freeness is introduced for the first time in this paper.

Protocol Description: We define as follows:

- N and r are given parameters to the probabilistic encryption method (see the appendix of [CBT94]).
- E, D and D' are polynomial-time computable, randomized encryption, decryption and certification functions.
- $\oslash$ and $\otimes$ are homomorphic functions such that, if z_1 is an encryption of x_1 and z_2 is an encryption of x_2, then $z_1 \otimes z_2$ is an encryption of $(x_1 + x_2) mod\, r$ and $z_1 \oslash z_2$ is an encryption of $(x_1 - x_2) mod\, r$

Ballots Generation:

- A generates $\{(x_i, y_i)$ / $x_i \in E(0), y_i \in E(1), 0 \leqslant i \leqslant N\}$.
- A publicly reveals $\{(\alpha_i, \beta_i)$ / $\alpha_i = min(x_i, y_i), \beta_i = max(x_i, y_i), 0 \leqslant i \leqslant N\}$.

Zero Knowledge Proofs: Using zero knowledge proofs, A publicly proves the fact that each encrypted pairs (α_i, β_i), and particularly the (α_0, β_0) pair, contains the both encrypted values. By a private channel, it also gives a proof of the ordering of (α_0, β_0) to V .

- V enters the voting booth.
- $\forall\, 0 \leqslant i \leqslant N$, A transmits $c_i = D(\alpha_i)$ over a private channel to V .
- $\forall\, 1 \leqslant i \leqslant N$, beacon generates N bits b_i, and transmits them over a public channel.
- V leaves the voting booth.
-
  - $\forall\, b_i$ such that $b_i = 0$, A opens (α_i, β_i) by publicly revealing $(D(\alpha_i), D(\beta_i))$ and the certificate pair $(D'(\alpha_i), D'(\beta_i))$.
  - $\forall\, b_i$ such that $b_i = 1$, A connects (α_i, β_i) by publicly revealing either $(D'(\alpha_i \oslash \alpha_0), D'(\beta_i \oslash \beta_0))$ if $D(\alpha_i) = D(\alpha_0)$, or $(D'(\alpha_i \oslash \beta_0), D'(\beta_i \oslash \alpha_0))$ if $D(\alpha_i) = D(\beta_0)$.

Vote Casting: V casts a 0-vote or a 1-vote by giving v the corresponding encrypted value over (α_0, β_0).

Tally Computation: Thanks to homomorphic properties of the encryption method, A can compute the final tally without decrypting each vote. The certification function provides a public proof of the tally correctness.

- With all the cast votes v_i, A computes: $T' = v_1 \otimes v_2 \otimes \ldots \otimes v_m$. While the v_i's are public, everyone can also compute T'.
- A reveals the final tally $T = D(T')$ together with the certificate $D'(T')$.

The protocol is presented in table 15.

Properties Checked:

	A	$V_j, \forall 1 \leqslant j \leqslant m$
Private input	$sk = (p, q)$	
Public input	$N, E = (n, y, .), r$	
Ballots Generation	$\forall 0 \leqslant i \leqslant N$: $\{(x_i, y_i) \,/\, x_i \in E(0), y_i \in E(1)\}$ $\boxed{\{(\alpha_i, \beta_i) \,/\, \alpha_i = min(x_i, y_i), \beta_i = max(x_i, y_i)\}}$	

Zero Knowledge Proofs	$\forall 0 \leqslant i \leqslant N$,, $c_i = D(\alpha_i)$ **sent** to V_j
	Beacon: $\forall 1 \leqslant i \leqslant N$, $\boxed{b_i} \xleftarrow{\$} \{0, 1\}$
	– $\forall i$ such that $b_i = 0$: "opens" of (α_i, β_i):
	$\boxed{((D(\alpha_i), D(\beta_i)), (D'(\alpha_i), D'(\beta_i)))}$
	– $\forall i$ such that $b_i = 1$: "connects" (α_i, β_i) to (α_0, β_0):
	If $D(\alpha_i) = D(\alpha_0)$ then
	$\boxed{(D'(\alpha_i \oslash \alpha_0), D'(\beta_i \oslash \beta_0))}$
	If $D(\alpha_i) = D(\beta_0)$ then
	$\boxed{(D'(\alpha_i \oslash \beta_0), D'(\beta_i \oslash \alpha_0))}$
		V enters the voting booth.
		V leaves the voting booth.

Vote Casting		To cast a 0-vote: If $c_0 = 0$ then $v_j = \alpha_0$ else $v_j = \beta_0$ To cast a 1-vote: If $c_0 = 0$ then $v_j = \beta_0$ else $v_j = \alpha_0$ $\boxed{v_j}$
Tally Computation	$T' = v_1 \otimes v_2 \otimes \ldots \otimes v_m$ $\boxed{T} = D(T')$	

Element published, **sent through private channel**.

Table15. Benaloh and Tuinstra: Scaled-down election protocol

Correctness Proven by the authors in the original paper. The protocol certifies that the cast votes are either 1 or 0. This is ensured by the unpredictability of the beacon bits.

Fairness Ballots are not counted until the end of the protocol.

Attacks:

Privacy If the single authority is corrupted or behaves in a faulty way, then it could reveal the individual votes with its decryption key and link it to their voter.

Uncoercibility Attack proposed in the paper's open problems. Although the protocol can stand up to static attacks, if the authority is corrupted by an dynamic adversary, uncoercibility does not hold.

Robustness A faulty behaviour of the single authority cannot be tolerated.

4.13 Benaloh and Tuinstra's scheme (second version)

Authors: Josh BENALOH and Dwight TUINSTRA (1994).
Primitives used: Homomorphic encryption, zero knowledge proofs, Shamir threshold secret sharing.
Requirements: Voting booth, beacon.

Summary: The second protocol of this paper ([CBT94]) is enhanced with a multiple tallying authorities protocol that improves privacy. This protocol is based on the homomorphic properties of encryption, allowing the tally to be decrypted without decrypting each vote. Zero knowledge proofs are used in order to ensure verifiability at each step. Shamir's secret sharing is used to allow a multiple authority decryption.

Protocol Description: We define as follows:

- the set of voters: $\{V_i, 1 \leqslant i \leqslant m\}$
- the set of authorities: $\{A_j, 1 \leqslant j \leqslant n\}$, each having different encryption, decryption and certification functions: (E_j, D_j, D'_j)
- r a prime larger than m.

Vote Construction and Zero Knowledge Proofs:

- $\forall\ 1 \leqslant j \leqslant n, \forall\ 1 \leqslant i \leqslant m$, A_j publicly provides a random encrypted value $z_{j,i}$ to V_i. Then, A_j proves by a private zero knowledge proof to V_i that $x_{j,i} = D(z_{j,i})$. This private proof is constructed like the Scaled Down Election Protocol's one.
- $\forall\ 1 \leqslant i \leqslant m$, V_i chooses his vote value $a_{0,i}$ between 0 and 1. Then, $\forall\ 1 \leqslant l \leqslant t-1$ (with t the threshold value of the secret sharing), he randomly selects $a_{l,i}$ within $\{0, \ldots, r-1\}$. $\{a_{.,i}\}$ will be the coefficients set of a polynomial $P = \sum_{i=0}^{t-1} a_i x^i$.

Vote Casting: $\forall\ 1 \leqslant j \leqslant n, \forall\ 1 \leqslant i \leqslant m$, V_i publicly transmits to A_j the value $y_{j,i} = (P_i(j) - x_{j,i}) mod\ r$. The vote cast by each V_i is the sequence: $< (y_{j,i}, z_{j,i}), 1 \leqslant j \leqslant n >$.

Tally Computation: Each A_j (and all observers and participants) can compute:

- the sum of the component j of all votes: $Y_j = \sum_{i=1}^{m} y_{j,i}$,
- the sum of blind factors of component j's encryption: $Z_j = \otimes_{i=1}^{m} z_{j,i}$;

and publicly reveals $X_j = D(Z_j)$ with the certificate value $D'(Z_j)$.

Participants and observers can reverse the blinding translation by computing $W_j = (X_j + Y_j) \mod r$. By this way, everybody can verify that the set of cast votes has not been altered, but one vote in particular remains hidden in the sum.

The polynomials properties allow now the authorities to interpolate the polynomial $Q(x) = \sum_{i=1}^{m} P_i(x)$ using the interpolation points (j, W_j). To finish, the sum of votes cast by the voters is $Q(0)$.

The protocol is presented in table 16.

	$A_j, \forall 1 \leqslant j \leqslant n$	$V_i, \forall 1 \leqslant i \leqslant m$
Private input	(D_j, D'_j)	
Public input	$N, r, E_j, \forall 1 \leqslant j \leqslant n$	
Ballots construction and Zero-knowledge proofs	$\forall 1 \leqslant j \leqslant n$, **random value** $z_{j,i}$ **sent** Zero-knowledge proof of $D(z_{j,i}) = x_{j,i}$ $\forall 1 \leqslant k \leqslant N, \xi_{j,k} \overset{\$}{\leftarrow} \{0, 1, ..., r-1\}$ $\rightarrow \zeta_{j,k} \in E(\xi_{j,k})$ $\boxed{(z_{j,i}, (\zeta_{j,1}, ..., \zeta_{j,N}))}$ $(x_{j,i}, (\xi_{j,1}, ..., \xi_{j,N}))$ **sent** Beacon: $\forall 1 \leqslant k \leqslant N, \boxed{b_k} \overset{\$}{\leftarrow} \{0, 1\}$ $\forall k$, such that $b_k = 0$: $\boxed{(D(\zeta_{j,k}), D'(\zeta_{j,k}))}$ $\forall k$, such that $b_k = 1$: $\boxed{(D(\zeta_{j,k} \oslash z_{j,i}), D'(\zeta_{j,k} \oslash z_{j,i}))}$	V_i enters the voting booth. $\overset{?}{=} \xi_{j,k}$ $\overset{?}{=} (\xi_{j,k} - x_{j,i}) \mod r$ V_i leaves the voting booth.
Vote casting		Shamir t-threshold secret-sharing: $a_{0,i} \in \{0, 1\}$ (the vote value) $\rightarrow P_i(x) = \sum_{l=0}^{t-1} a_{l,i} x^l$, with $a_{l,i} \in \{0, ..., r-1\}$ $\forall 1 \leqslant j \leqslant n, \boxed{y_{j,i}} = (P_i(j) - x_{j,i}) \mod r$
Tally computation	Computation: $\forall 1 \leqslant j \leqslant n$, $Y_j = y_{1,j} + y_{2,j} + ... + y_{m,j}$ $Z_j = z_{1,j} \otimes z_{2,j} \otimes ... \otimes z_{m,j}$ $\boxed{X_j = D(Z_j), d'(Z_j)}$ Polynomial interpolation (using (j, W_j)) $\rightarrow Q(x) = \sum_{i=1}^{m} P_i(x)$ $\rightarrow \boxed{Q_0}$ = sum of a_0-votes.	

Element published, **sent through private channel**.

Table16. Benaloh and Tuinstra's protocol: Elections with multiple tallying authorities

Properties Checked:

Correctness The proof is the same as in the protocol above.

Privacy Claimed by the authors in the original paper, under the assumption that uncoercibility is satisfied and that this implies privacy. Claimed by Ronald CRAMER, Rosario GENNARO and Berry SCHOENMAKERS in [CGS97], under the so-called r-th residuosity assumption.

Fairness Ballots are not counted until the end of the protocol.

Attacks:

Uncoercibility The scaled-down protocol attack for this property can also be applied here. Another attack is proposed by Martin HIRT and Kazue SAKO in [HS00]. In fact, they attack an extension of this paper where an additional protocol is added to restrict voters to selecting their votes from $\{0, 1\}$. This may allow the voter to construct a receipt proving his vote.

Both protocols do not consider the security properties called democracy (containing eligibility and prevention of multiple voting), and robustness.

4.14 Cramer et al. Scheme

Authors: Ronald CRAMER, Rosario GENNARO and Berry SCHOENMAKERS(1997).
Primitives used: Homomorphic encryption (El Gamal cryptosystem), Zero Knowledge Proofs, Threshold Secret Sharing: fault tolerant threshold decryption technique,
Requirements: Bulletin board (public broadcast channel with memory), beacon.

Summary: This scheme proposed in [CGS97], is composed of three stage: an initialization stage, a vote casting stage (the voter publicly posts an encrypted ballot both with the proof that it is well-formed on a bulletin board), and tally computation stage (using a threshold decryption technique, authorities obtain the final tally without decrypting each ballot thanks to homomorphic encryption properties). The basic protocol offers to voter a choice between two options. An extension is proposed in the same paper to permit elections with several choices.

It needs n authorities $A_1, ..., A_n$.

Protocol Description

Initialization Stage: Using the robust threshold ElGamal cryptosystem ([ElG85]), each authority A_j runs a key generation protocol and thus obtains, by the secret sharing technique, a share s_j of the private key s. Then, it computes $h_j = g^{s_j}$ and publicly reveals it as a public key.

Vote Casting Stage: The voter has two vote choices: $m_1 = G$ or $m_0 = \frac{1}{G}$ with G a generator of the ElGamal encryption subgroup G_q. Then, the vote casting consists of the publication of two parts:

- The encrypted ballot $(x, y) = (g^\alpha, h^\alpha G^b)$ with $b \overset{\$}{\leftarrow} \{-1, 1\}$ and $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_q$ (q is a parameter of ElGamal cryptosystem). In fact, this ballot is precomputed by the voter and what is really sent is: $(x^e, y^e) = (g^{\alpha e}, h^{\alpha e} G^{be}) = (g^{\alpha e}, h^{\alpha e} G^v)$ with $e \in \{-1, 1\}$ such that $be = v$, the vote value. This ballot is posted on the bulletin board.
- A non iteractive proof of validity using the zero knowledge proofs technique. By this way, the voter shows that his ballot is well-formed by proving the knowledge of the relation: $log_g(x) = log_h(\frac{y}{m})$ where $m \in \{m_0, m_1\}$, $h = g^s$ and g is an ElGamal encryption parameter.

Tally Computation Stage:

- Authorities check the proofs of validity and form the product: $(X, Y) = (\prod_{i=1}^{l} x_i, \prod_{i=1}^{l} y_i)$.
- Each authority A_j computes and broadcasts $W_j = X^{s_j}$ together with a non iteractive zero knowledge proof of the relation $log_g h_j = log_X W_j$.
- Authorities then use the threshold secret sharing decryption process to jointly compute T, the difference between the number of yes-votes and no-votes.

The protocol is presented in table 17.

p, q, g and G are ElGamal Cryptosystem parameters. Using this, authorities generate a secret key s and each obtains a part s_j of it by secret sharing.H is an hash function. ID_i is a unique public string identifying V_i. T is the difference between the number of yes-votes and no-votes.

<table>
<tr><th></th><th>Voters V_i, $1 \le i \le l$</th><th>Authorities A_j, $1 \le j \le n$, $l \gg n$</th></tr>
<tr><td>Private input</td><td></td><td>$s_j \in \mathbb{Z}_q$</td></tr>
<tr><td>Public input</td><td colspan="2">public key $h = g^s$</td></tr>
<tr><td>Vote Casting</td><td>Initialization: $m_1 = G, m_0 = \frac{1}{G}$
ElGamal Encryption: $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_q, b \overset{\$}{\leftarrow} \{-1,1\}$,
$(x,y) = (g^\alpha, h^\alpha G^b)$
(Non-Interactive) Proof of Validity:
<table>
<tr><td>$b = 1$</td><td>$b = -1$</td></tr>
<tr><td colspan="2">$x = g^\alpha$</td></tr>
<tr><td>$y = h^\alpha G$</td><td>$y = \frac{h^\alpha}{G}$</td></tr>
<tr><td colspan="2">$w \overset{\$}{\leftarrow} \mathbb{Z}_q$</td></tr>
<tr><td>$r_1, d_1 \overset{\$}{\leftarrow} \{-1,1\}$
$a_1 = g^{r_1} x^{d_1}$
$b_1 = h^{r_1}(yG)^{d_1}$
$a_2 = g^w$
$b_2 = h^w$</td><td>$r_2, d_2 \overset{\$}{\leftarrow} \{-1,1\}$
$a_1 = g^w$
$b_1 = h^w$
$a_2 = g^{r_2} x^{d_2}$
$b_2 = h^{r_2}(\frac{y}{G})^{d_2}$</td></tr>
<tr><td colspan="2">$c = H(ID_i, x, y, a_1, b_1, a_2, b_2)$</td></tr>
<tr><td>$d_2 = c - d_1$
$r_2 = w - \alpha d_2$</td><td>$d_1 = c - d_2$
$r_1 = w - \alpha d_1$</td></tr>
</table>
d_1, d_2, r_1, r_2
sent
$\boxed{e} \in \{-1,1\}$ such that $be = v$ (the vote value)
$\boxed{(x^e, y^e) = (g^{\alpha e}, h^{\alpha e} G^v)}$</td><td></td></tr>
<tr><td>Tally Computation</td><td></td><td>Check of Validity Proofs
$c \overset{?}{=} d_1 + d_2$
$a_1 \overset{?}{=} g^{r_1} x^{d_1}$
$b_1 \overset{?}{=} h^{r_1}(yG)^{d_1}$
$a_2 \overset{?}{=} g^{r_2} x^{d_2}$
$b_2 \overset{?}{=} h^{r_2}(\frac{y}{G})^{d_2}$
Computations:
$\bullet (X,Y) = (\prod_{i=1}^{l} x_i, \prod_{i=1}^{l} y_i)$, $\boxed{W_j} = X^{s_j}$
Zero Knowledge Proof * of the relation:
$log_g h_j = log_X W_j = s_j$
<table>
<tr><th>Prover</th><th>Verifier</th></tr>
<tr><td>$w \overset{\$}{\leftarrow} \mathbb{Z}_q$
$(a,b) = (g^w, X^w)$ sent</td><td></td></tr>
<tr><td></td><td>$c \overset{\$}{\leftarrow} \mathbb{Z}_q$ sent</td></tr>
<tr><td>$r = w + s_j c$ sent</td><td></td></tr>
<tr><td></td><td>$g^r \overset{?}{=} a h_j^c$
$X^r \overset{?}{=} b W_j^c$</td></tr>
</table>
• Joint Decryption: $\rightarrow W = G^T$, $\boxed{T} = log_G W$</td></tr>
</table>

* The proof is interactive but the actual protocol uses the corresponding non-interactive proof.

Element $\boxed{\text{published}}$ on bulletin board, or sent to the other participating part.

Table17. The "yes-no vote" protocol

Extension to Multi-Way Elections: To get an election for a 1-out-of-K choice, the process remains the same except the value of m: Instead of having the choice between $m_1 = G$ and $m_0 = \frac{1}{G}$, the protocol is initialized with $m_i = \frac{1}{G_i}, 1 \leq i \leq K$ with K independently generated generators G_i.

Properties Claimed:

Privacy Claimed by the authors in the original paper: Privacy is achieved by the way of encrypting and casting votes that hides the vote value to anyone.

Universal Verifiability Claimed by the authors in the original paper: Universal verifiability is obtained by the zero-knowledge proof performed by the authorities prior to decryption, and by the fact that the computation of $log_G W$ may be done by any party.

Robustness Claimed by the authors in the original paper: Robustness is achieved with respect to faulty authorities by employing fault tolerant threshold cryptosystems.

Prevention of double voting Claimed by the authors in the original paper: The challenge c is made voter-specific by adding the unique public string identifying V_i, ID_i, in the hash function parameters.

Attacks:

Receipt-Freeness/Incoercibility Attack proposed: The voter V_i can provide the value $c = H(ID_i, x, y, a_1, b_1, a_2, b_2)$ together with his digital signature as a receipt. These data permit the coercer to know $v = be$ the vote value, while being sure it is the actual vote of V_i thanks to ID_i.

4.15 Cohen and Fischer 's Scheme

Authors: Josh D. COHEN and Michael J. FISCHER (1985).
Primitives used: Interactive zero-knowledge proofs, homomorphic encryption.
Requirements: Publicly readable bulletin-boards (one for each sender), beacon.

Summary: This scheme proposed in [CF85] is composed in four stages: In a first step, the government provides the set of parameters that will be used in the following. Then, the voters generate some valid ballots together with an interactive proof of validity, before casting their vote. Finally, the government computes the tally, using the homomorphic properties of encryption.

It needs a unique authority A called government by the authors.

Here, we define some mathematical objects. Let r be a prime such that $r > J$. An integer n is said to be $N - admissible$ if $n = pq$ with p and q prime numbers such that $|p| = |q| = N$, and $r|(p-1), r \nmid (q-1)$. Let y an integer such that $gcd(n, y) = 1$. w is called an $i - vote$ w.r.t. r, n, y if $gcd(w, n) = 1$ and for $0 \leqslant i \leqslant r$, $\exists x$ such that $w = y^i x^r \mod n$.

Lemma 1. *If* $\left\{ \begin{array}{c} n \text{ is } N - admissible \\ gcd(w,n) = 1,\ gcd(y,n) = 1 \\ \nexists x \text{ such that } y \equiv x^r \mod n \end{array} \right\}$ *then* $\exists! i,\ 0 \leqslant i \leqslant r$ *such that* $w = y^i x^r \mod n$ *is an* $i - vote$.

Protocol Description

Initialization: Here, valid parameters are generated for encryption and zero-knowledge proofs processes.

- Given a security parameter N and a choosen prime r such that $r > J$, A randomly generates N pairs (n_i, y_i) such that n_i is $N - admissible$, $gcd(y_i, n_i) = 1$ and $\nexists x_i, x_i^r \equiv y_i \mod n_i$.
- Proof of parameters validity: The beacon generates m, $1 \leqslant m \leqslant N$. $\forall i \neq m$, A publicly reveals p_i and q_i to attest of the parameters form. (n_m, y_m) is now denoted (n, y).

Ballots generation and interactive proof of validity:

- Each voter V_j randomly constructs $(\lceil log_2(rN) \rceil + 1)$ ballots $B_{j,i} = (f_i^r \mod n, yg_i^r \mod n)$ with f_i and g_i relatively prime to n. The pairs values are in random order.
- Proof of ballots form: The beacon generates $\lceil log_2(rN) \rceil$ bits b_i, $1 \leqslant i \leqslant \lceil log_2(rN) \rceil$.
  $\forall i$ such that $b_i = 1$, the two values of $B_{j,i}$ are revealed.
  $\forall i$ such that $b_i = 0$, $B_{j,i}$ is connected with $B_{j,0}$ by revealing $(f_i f_o^{-1} \mod n, g_i g_o^{-1} \mod n)$.

Vote casting: Each voter selects, and publishes on his bulletin board, his actual vote value:

$$w_j = \begin{cases} yg_o^r & \text{to cast a yes-vote} \\ f_o^r & \text{to cast a no-vote} \end{cases}$$

Tally computation:

- A checks the voters' ballots form proofs to determine the set Γ consisting of all j such that $B_{j,0}$ is a valid ballot.
- A computes $W = \prod_{j \in \Gamma} w_j \mod n$.
  Lemma: If each of $w_1, w_2, ..., w_k$ is a valid vote and $k < r$ then W is the number of valid yes-votes.
- Proof of the tally validity: A randomly selects $\lceil log_2(N) \rceil$ numbers c_i such that $gcd(c_i, n) = 1$ and publicly reveals all $C_i = c_i^r$. The beacon generates $\lceil log_2(N) \rceil$ b_i, $1 \leqslant i \leqslant \lceil log_2(N) \rceil$.
  A computes x and t such that $W \equiv y^t x^r \mod n$ and publicly reveals the pair $(t, |\Gamma| - t)$ respectively corresponding to the number of yes-votes and no-votes.
  $\forall i$ such that $b_i = 1$, A reveals c_i.
  $\forall i$ such that $b_i = 0$, A reveals c_i' such that $y^t (c_i')^r = C_i W$.
  By this commitment scheme based zero-knowledge proof, each observer can verify the accuracy of the final tally.

  The protocol is presented in table 18.

Properties Checked:

Correctness Proven by the authors in the original paper. This property is satisfied thanks to zero-knowledge proofs performed during the protocol. Each voter has a function to check the election results. With very high probability, either the election passes the check and the announced tally of votes is correct, or the election fails the check and the government is not following its prescribed protocol.

Security Proven by the authors in the original paper. A protocol run can be simulate in a indistinguishable way so that a coercer cannot be convinced of a particular voter's vote value.

Attacks:

	A	$V_j, \forall 1 \leqslant j \leqslant J$
Public input	Security parameter N; r	
Initialization	$\boxed{\{(n_i, y_i), 1 \leqslant i \leqslant N\}}$ with n_i $N-admissible$, $gcd(y_i, n_i) = 1$ and $\nexists x_i, x_i^r \equiv y_i \mod n_i$. Beacon: $\boxed{m} \overset{\$}{\leftarrow} \{1, ..., N\}$ If $i \neq m$, $\boxed{(p_i, q_i)}$ s. t. $n_i = p_i q_i$, $r \mid (p_i - 1)$ and $r \nmid (q_i - 1)$. $(n, y) \overset{\text{def}}{=} (n_m, y_m)$	
Ballots generation		$B_{j,i} = (f_i^r \mod n, y g_i^r \mod n)$ randomly ordered with $gcd(f_i, n) = gcd(g_i, n) = 1$ $\boxed{\{B_{j,i}, 1 \leqslant i \leqslant \lceil log_2(rN) \rceil\}}$ Beacon: $\forall 1 \leqslant i \leqslant \lceil log_2(rN) \rceil$, $\boxed{b_i} \overset{\$}{\leftarrow} \{0, 1\}$ $\forall i$ s. t. $b_i = 1$, $\boxed{B_{j,i}}$ $\forall i$ s. t. $b_i = 0$, $\boxed{(f_i f_o^{-1} \mod n, g_i g_o^{-1} \mod n)}$
Vote casting		yes-vote: $\boxed{w_j = y g_o^r}$ no-vote: $\boxed{w_j = f_o^r}$
Tally computation and Proof of validity	Check of the ballots form to obtain: $\Gamma = \{j$ such that $B_{j,0}$ is valid$\}$ $W = \prod_{j \in \Gamma} w_j \mod n$ Random selection of c_i, $1 \leqslant i \leqslant \lceil log_2(N) \rceil$ $\boxed{C_i} = c_i^r$, $1 \leqslant i \leqslant \lceil log_2(N) \rceil$ Beacon: $\forall 1 \leqslant i \leqslant \lceil log_2(N) \rceil$, $\boxed{b_i} \overset{\$}{\leftarrow} \{0, 1\}$ Computation: of (x, t) s.t. $W \equiv y^t x^r \mod n$ $\boxed{(t, \lvert\Gamma\rvert - t)} = (\lvert yes - votes\rvert, \lvert no - votes\rvert)$ $\forall i$ s. t. $b_i = 1$, $\boxed{c_i}$ $\forall i$ s. t. $b_i = 0$, $\boxed{c_i'}$ s. t. $y^t (c_i')^r = C_i W$	

Element $\boxed{\text{published on bulletin boards}}$.

Table18. Cohen and Fischer's protocol

Privacy This property does not hold if the single authority is corrupted or has faulty behavior. However, privacy is satisfied under the assumption of a proper authority. The next presented protocol (4.16) improves privacy by dividing the power between several authorities.

4.16 Benaloh and Yung's Scheme

Authors: Josh COHEN BENALOH and Moti YUNG (1986).
Primitives used: Homomorphic encryption, interactive zero-knowledge proofs.
Requirements: Bulletin boards, beacon (but could be replaced by a sub-protocol presented in table 20).

Summary: This scheme proposed in [CBY86] is an extension of COHEN and FISHER's protocol where the government tasks are distributed between several tellers in order to enhance privacy. It is composed in six stages that follow the extended protocol's ideas, while adapting it to several authorities. The beacon can be replaced by a sub-protocol performed by the agents of the scheme.

The election participants are κ authorities and ω voters. The mathematical framework of election parameters are the same as in COHEN and FISHER's protocol.

Protocol Description

Initialization: Each teller A_i selects and posts a pair of election parameters (n_i, y_i).

Ballots generation: Using the previous parameters, each voter V_j generates a set of N unmarked "test ballots", consisting of an encrypted yes-vote and an encrypted no-vote $B_{j,n} = (Z^0_{j,n}, Z^1_{j,n}), 1 \leqslant n \leqslant N$ randomly ordered, with $Z^\alpha_{j,n} = \{z_{i,j,n} = y_i^{e^i} x_i^r, 1 \leqslant i \leqslant \kappa\}$ such that $(\sum_{i=1}^{\kappa} e_i) \mod r = \alpha \in \{0, 1\}$ constitutes the vote value. Each authority A_i receives its corresponding part of the ballot $z_{i,j,n}$.

Zero-knowledge proof of ballots validity: An interactive proof of ballots validity is engaged to demonstrate the ballots form. This proof is performed using a beacon (or the sub-protocol replacing). The principle remains the same as in COHEN and FISHER's protocol, but it is adapted to the multi-authority form:

- Each voter V_j selects randomly a ballot $B_{j,k}$ which will be the "primary" ballot of the proof. The remaining are called "auxiliary" ballots.
- The beacon generates N bits b_n, $1 \leqslant n \leqslant N$. $\forall n \neq k$ such that $b_n = 1$, the vote-values of $B_{j,n}$ are publicly revealed together with the corresponding values x_i so that the authorities can check that $B_{j,n}$ holds a yes-vote and a no-vote. $\forall n \neq k$ such that $b_n = 0$, $B_{j,n}$ is connected with $B_{j,k}$: Two ballots B_A and B_B are "type-equivalent" if the first value of B_A encrypts the same vote as one of the values of B_B and the second value of B_A encrypts the same vote as the other value of B_B, i.e. $\sum_{i=1}^{\kappa} e_{A,i} - e_{B,i} \equiv 0 \mod r$. To demonstrate this, the voter computes $\frac{z_{A,i}}{z_{B,i}} \mod r = y_i^{e_{A,i}-e_{B,i}} (\frac{x_{A,i}}{x_{B,i}})^r \mod r$ and proves to each concerned authority A_i that it is a $(e_{A,i} - e_{B,i}) - type$ value by using the proof of tally validity constructed by COHEN and FISHER, with $(\frac{x_{A,i}}{x_{B,i}})$ as commitment. Once done, the authorities jointly compute the sum of all the obtained values and check if the result equals $0 \mod r$.

  By this way, each observer can be convinced of the test ballots validity.

Proof of the tellers ability to distinguish between vote-types:

- Each voter "marks" each test ballot by randomly selecting one of the two votes.
- The tellers decrypt all voters' test votes.
- Each voter releases its own decryption of its test votes and checks if it is the same as the tellers' decryption.

  The election is continued only if the proofs are valid. If not, the protocol has to be restarted.

Vote casting:

- Each voter V_j selects an unmarked "master ballot" $B_{j,m}$, consisting of an encrypted yes-vote and an encrypted no-vote.
- The proof of the third stage is performed again to prove the selected ballot's validity.
- The voter marks its master ballot by designating one of the two votes as desired.

Tally computation and proof of correctness:

- Each teller A_i computes the votes part corresponding to its parameters:

$$\mathcal{W}_i \equiv \prod z_{j,m,i} \mod n_i$$

- Then, it releases a τ_i such that $\mathcal{W}_i \equiv y_i^{\tau_i}\mathcal{X}_i$ where $\mathcal{X}_i$ is a r^{th}-residue modulo n_i. Then, A_i engages once again the COHEN and FISHER proof of validity to demonstrate the accuracy of τ_i.
- Finally, the election tally is:

$$\Gamma = \sum_{i=1}^{\kappa} \tau_i$$

The protocol is presented in table 19.

Properties Checked:

Correctness Proven by the authors in the original paper. If all the participants behave as scheduled, then the election protocol execution time is bounded by a small polynomial in N. The probability that the final tally equals the actual tally is very high, and, if it is not the case, it will be detected with a very high probability.

Privacy Proven by the authors in the original paper. Privacy is obtained under the assumption that at least one of the tellers is honest. The paper's proof shows that if there exist a possibility of distinguishing between two vote values with a probability greater than $\frac{1}{2}$, then it is possible to correctly decide r^{th}-residuosity modulo n with a probability greater than $\frac{1}{2}$.

Verifiability Claimed by the authors in the original paper. The interactive proofs incorporated in various stages of the election process allow all participants to verify the accuracy of the final tally.

Attacks:

Robustness Attack proposed in the original paper. If a teller has a faulty or cheating behavior, the election must be restarted from the beginning (without the corrupted teller). The next protocol (**??**) presents a fault-tolerant version of this scheme.

5 Comparison of Electronic Voting Schemes

Because of the different cryptographic primitives used and the way they are combined, the following schemes do not satisfy the same properties. The following comparative statements allow users to choose a protocol more than another according to the properties they want to be satisfied. However, most of these schemes are not easily implemented since some of the theoretical assumptions are not practically constructed.

	$A_i, 1 \leqslant i \leqslant \kappa$	$V_j, 1 \leqslant j \leqslant \omega$
Public input	Security parameter N; $r > \omega$	
1 Initialization	$\boxed{P = \{(n_i, y_i), 1 \leqslant i \leqslant \kappa\}}^{\mathcal{BB}}$ with n_i $N - admissible$, $gcd(y_i, n_i) = 1$ and $\nexists x_i, x_i^r \equiv y_i \mod n_i$.	
2 Ballots generation		V_j performs $2N$ times the following computation: Given P, V_j randomly selects: $Z = \{z_i = y_i^{e_i} x_i^r, 1 \leqslant i \leqslant \kappa\}$, s.t. $\forall 1 \leqslant i \leqslant \kappa, \exists! e_i, 0 \leqslant e_i < r$ and s.t. $TYPE[Z, P] = (\sum_{i=1}^{\kappa} e_i) \mod r \in \{0, 1\}$ If $TYPE[Z, P] = 0$ then $Z = Z^0$ is a no-vote. If $TYPE[Z, P] = 1$ then $Z = Z^1$ is a yes-vote. to obtain N (yes-vote,no-vote) ballots: $\forall 1 \leqslant n \leqslant N, \boxed{B_{j,n}}^{\mathcal{BB}} = (Z^0, Z^1)$ randomly ordered
3 Proof of ballots validity	Beacon or sub-protocol: $\forall 1 \leqslant n \leqslant N, \boxed{b_n}^{\mathcal{BB}} \xleftarrow{\$} \{0, 1\}$	Random selection of a ballot: $k \xleftarrow{\$} [1, N]$ $\forall n \neq k$ s. t. $b_n = 1$, $\boxed{\{\sum_{i=1}^{\kappa} e_{j,n,i}, x_{j,n,i}, 1 \leqslant i \leqslant \kappa\}}^{\mathcal{BB}}$ $\forall n \neq k$ s. t. $b_n = 0$, V_j computes: $\forall 1 \leqslant i \leqslant \kappa$, $\frac{z_{j,n,i}}{z_{j,k,i}}$ and $\frac{x_{j,n,i}}{x_{j,k,i}}$
	Proof of ballots form (see "Proof of tally validity" in table 18)	
4 Proof of the tellers ability to distinguish vote-types	$\forall 1 \leqslant i \leqslant \kappa, \forall 1 \leqslant j \leqslant \omega$, $\boxed{\check{e}_{j,n,i}}^{\mathcal{BB}} = \log_{y_i}(\frac{z_{j,n,i}}{x_{j,n,i}^r}) \mod r$ with $z_{j,n,i} \in Z_{j,n}^{0/1}$	$\forall 1 \leqslant j \leqslant \omega, \forall 1 \leqslant n \leqslant N$, $\boxed{Z_{j,n}^0, \{x_{j,n,i}, 1 \leqslant i \leqslant \kappa\}}^{\mathcal{BB}}$ or $\boxed{Z_{j,n}^1, \{x_{j,n,i}, 1 \leqslant i \leqslant \kappa\}}^{\mathcal{BB}}$ $\check{e}_{j,n,i} \stackrel{?}{=} e_{j,n,i}$ $\check{e}_{j,n,i} \neq e_{j,n,i}$: Election restarts without A_i. $\check{e}_{j,n,i} = e_{j,n,i}$: Election goes on.
5 Vote casting and proof of validity		Selection of a master ballot: $m \xleftarrow{\$} [1, N]$ Proof of $B_{j,m}$'s validity by re-engaging the stage 3's proof using m instead of k. Vote casting: yes-vote: $\boxed{Z_{j,m}^0}^{\mathcal{BB}}$, no-vote: $\boxed{Z_{j,m}^1}^{\mathcal{BB}}$
6 Tally computation and proof of correctness	$\forall 1 \leqslant i \leqslant \kappa$, $\mathcal{W}_i \equiv \prod z_{j,m,i} \mod n_i$ $\boxed{\tau_i}^{\mathcal{BB}}$ such that $\mathcal{W}_i \equiv y_i^{\tau_i} \mathcal{X}_i$ and $\mathcal{X}_i$ is a r^{th}-residue modulo n_i	
	Proof of τ_i validity (see "Proof of tally validity" in table 18)	
	Tally: $\boxed{\Gamma}^{\mathcal{BB}} = \sum_{i=1}^{\kappa} \tau_i$	

Element $\boxed{\text{published on bulletin boards}}^{\mathcal{BB}}$

Table19. Benaloh and Yung's Protocol

$A_i, 0 \leqslant i \leqslant \kappa$	Other participants
$b_i \overset{\$}{\leftarrow} \{0,1\}$ Random selection of x_i s.t. $gcd(x_i, n_i) = 1$ If $b_i = 0$, $\boxed{z_i = x_i^r}$ If $b_i = 1$, $\boxed{z_i = y_i x_i^r}$	
Once all the z_i are posted: $\boxed{x_i}$	
	If $z_i = x_i^r$ then $b_i = 0$ If $z_i \neq x_i^r$ then $b_i = 1$
	$\boxed{b} = \otimes_{0 \leqslant i \leqslant \kappa} b_i$

Element $\boxed{\text{published on bulletin boards}}$. $\otimes$ corresponds to the XOR operation.

Table20. Sub-protocol replacing the beacon

5.1 According to Cryptographic Primitives Used

In table 21, we show you some schemes using some cryptographic primitives. Some of them are described in this report, the others have been studied this summer.

Protocols	Cryptographic primitives used							Special channels	
	ZKP	HE	RE	SS	BS	DE	MN	UC	AC
[FOO92](4.2)	X				X				X
[OMA+99] (4.3)	X			X	X		X		
[KKLA01](4.4)	X			X	X				
[Rad95](4.1)	X				X				X
[LBD+03](4.8)			X	X			X		
[JL97](4.5)				X	X				X
[JLY98](4.6)				X	X				X
[Web06](4.9)	X		X	X			X	X	X
[Rja02](4.11)	X	X	X	X		X		X	
[CB87]	X	X		X					
[Sch99]	X	X		X					
[CGS97](4.14)	X	X		X					
[HS00](4.10)	X	X	X				X	X	
[Jak98]		X					X		
[Abe99]	X						X		
[ALBD04]	X	X	X	X			X		
[CRS05](4.7)	X		X				X		
[Coh86](**??**)	X	X		X					
[CBY86](4.16)	X	X							
[CBT94](4.12)	X	X		X					
[CF85](4.15)									
[SK94](**??**)									

Abbreviations used: HE: Homomorphic Encryption, MN: Mix-Nets, BS: Blind Signatures, ZKP: Zero-Knowledge Proof, RE: Re-Encryption, SS: Secret Sharing, DE: Deniable Encryption, UC: Untappable Channel, AC: Anonymous Channel.
X: used.

Table21. Some Schemes and theirs Primitives

5.2 According to Security Properties Satisfied

In table 22, we have tried to list the different requirements achieved by the schemes given in table 21.

Protocols	Requirements							
	Privacy	Receipt-Freeness	Robustness	Verifiability		Democracy		Fairness
				U	I	E	PMV	
[FOO92](4.2)	P	A	A	A	C	P	P	P
[OMA+99](4.3)	C	A	A	A	C	C	C	C
[KKLA01](4.4)	A	C	A	A	C	C	C	C
[Rad95](4.1)	C	A	A	A	C		C	
[LBD+03](4.8)	P	C	C	C			C	C
[JL97](4.5)	C	A	A	A	C	P	P	P
[JLY98](4.6)	C	A	C	C	C	P	P	C
[Web06](4.9)	C	C	C	C	A	C	C	C
[Rja02](4.11)	C	C	C	C	C	A		C
[CB87]	C	C	C	C		S	S	
[Sch99]	C		C	C		S	S	
[CGS97](4.14)	C		C	C		C	C	
[HS00](4.10)	C	C	C	C	C	S	S	C
[Jak98]	P		P					
[Abe99]	P		P	P				
[ALBD04]	C	C	C	C		C	C	C
[CRS05](4.7)	C	C	C	C	C			C
[Coh86](**??**)	P		C	C				C
[CBY86](4.16)	P		C	C				C
[CBT94](4.12)	P			C				C
[CF85](4.15)								
[SK94](**??**)								

Abbreviations used:U: Universal, I: Individual, E: Eligibility, PMV: Prevent Multiple Voting.
C: claimed achieved requirement, P: proved achieved requirement, S: supposed achieved requirement, A: attacks found.

Table22. Comparison of some Schemes

6 Conclusion

We note that the subject of this report does not take into account remote voting in which network brings new security problems.

Homogenisation and Formalisation Problems

As shown in the comparative statements, none of the protocols satisfy all the required security properties.

However, it seems difficult to claim anything about these properties according to the fact that no formal definitions exist to blend protocols' verification proofs.

Moreover, the read articles do not refer to the same security properties and when they do, they are not defined in the same way.

For example, receipt-freeness is sometimes considered as the fact that no receipt is produced in the protocol, and other times, as the fact that whatever the produced receipt, the voter cannot prove anything with it. Concerning robustness, some authors claim their protocol to be robust but it is only robust against static adversaries. Yet, to get a secure protocol, it has to be robust against dynamic adversaries.

As a matter of fact, the comparative statement deals with properties claimed by authors (and are often not proven).
That is why our comparative statement about security properties could contain some misjudgements.

There is a real need of formalisation to make possible effective comparison.

Applicability Problems

It seems important not to lose sight of the fact that electronic voting schemes aim to be implemented and applied in "real life" with "real people" for elections.

In this purpose, there are many aspects to consider:

- We have to find practical applications to the theoretical assumptions made in protocols like:
  - Beacons that produce an unlimited supply of unpredictable bits. A physical construction of them could be a XOR-ing bits supplied by agents of the election system.
  - Anonymous Channels that can be implemented thanks to mix-nets.
  - Voting Booth that physically guarantee secret communication between the authorities and each voter. They have not been adapted into practical technics yet.
  - etc,...
- We have to check the said robust protocols to know if the election must be restarted from the beginning or if a tally is produced even in the presence of faulty behaviour.
- The number of voters' interventions is also crucial. For example, in [Oka97], there are three voters rounds:
  - the authorization stage,
  - the voting stage,
  - the claiming stage.

  It seems to be difficult to ask voters to come three times or more without lowering the voting participation.
- An other important aspect for voters is the good understanding of the protocol: if it is too intricate, they will not trust it.
- The efficiency of the electronic voting system will depend on other factors like the number and the cost of computational operations, the number of authorities,...

These open problems are much as ways of researches, as well in a theoretical point of view than in a practical one.

References

Abe99. Masayuki Abe. Mix-networks on permutation networks. In *Asiacrypt'99*, volume 1716 of *Lecture Notes in Computer Science*, pages 258 – 273, Berlin, 1999. Springer-Verlag.

Adi06. Ben Adida. *Advances in cryptographic voting systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2006. Adviser-Ronald L. Rivest.

ALBD04. Riza Aditya, Byoungcheon Lee, Colin Boyd, and Ed Dawson. An efficient mixnet-based voting scheme providing receipt-freeness. 3184:152–161, 2004.

Ben06. Josh Benaloh. Simple verifiable elections. In *EVT'06: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, pages 5–5, Berkeley, CA, USA, 2006. USENIX Association.

BFP+01. Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pages 274–283, New York, NY, USA, 2001. ACM.

BJR01. Shuki Bruck, David Jefferson, and Ronald L. Rivest. A modular voting architecture (frogs). In *Workshop on Trustworthy Elections*, 2001.

CB87. Josh Daniel Cohen Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, New Haven, CT, USA, 1987.

CBT94. Josh Daniel Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *STOC'94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 544–553, New York, USA, 1994. ACM.

CBY86. Josh Daniel Cohen Benaloh and Moti Yung. Distributing the power of a government to enhance the privacy of voters. In *PODC '86: Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, pages 52–62, New York, NY, USA, 1986. ACM.

CDNO97. Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. *Lecture Notes in Computer Science*, 1294:90–104, 1997.

CF85. Josh Daniel Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proceedings of 26th Symposium on Foundations of Computer Science*, pages 372–382, Portland, October 1985.

CFSY95. Ronald J.F. Cramer, Matthew Franklin, L. A.M. Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

CGS97. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *Lecture Notes in Computer Science*, 1233:103+, 1997.

Cha81. David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

Cha04. David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.

CMFST06. Benoit Chevallier-Mames, Pierre-Alain Fouque, David Pointchevaland Julien Stern, and Jacques Traore. On some incompatible properties of voting schemes. In *In Proceedings of the IAVoSS Workshop on Trustworthy Elections*, Robinson College, Cambridge, United Kingdom, june 2006.

Coh86. Josh Daniel Cohen. Improving privacy in cryptographic elections. Technical report, Yale University, 1986.

CRS05. David Chaum, Peter Y.A. Ryan, and Steve A. Schneider. A practical, voter-verifiable election scheme. volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer Berlin / Heidelberg, 2005.

DJN03. I. Damgard, M. Jurik, and J. Nielsen. A generalization of paillier's public-key system with applications to electronic voting. 2003.

DKR06. Stephanie Delaune, Steve Kremer, and Mark Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *CSFW '06: Proceedings of the 19th IEEE workshop on Computer Security Foundations*, pages 28–42, Washington, DC, USA, 2006. IEEE Computer Society.

DLM82. Richard A. DeMillo, Nancy A. Lynch, and Michael J. Merritt. Cryptographic protocols. In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 383–400, New York, NY, USA, 1982. ACM.

ElG85. Taher ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. pages 10–18, 1985.

FOO92. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *ASIACRYPT '92: Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques*, pages 244–251, London, UK, 1992. Springer-Verlag.

FPS01. Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *FC '00: Proceedings of the 4th International Conference on Financial Cryptography*, pages 90–104, London, UK, 2001. Springer-Verlag.

FS01. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 368–387, London, UK, 2001. Springer-Verlag.

Fur04. Jun Furukawa. Efficient, verifiable shuffle decryption and its requirement of unlinkability. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 319–332. Springer, 2004.

GHPvR05. Flavio D. Garcia, Ichiro Hasuo, Wolter Pieters, and Peter van Rossum. Provable anonymity. In *FMSE '05: Proceedings of the 2005 ACM workshop on Formal methods in security engineering*, pages 63–72, New York, NY, USA, 2005. ACM.

GMR89. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

Gol01. Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, Cambridge, UK, 2001.

Gol02. O. Goldreich. Zero-knowledge twenty years after its invention, 2002.

Gro03. Jens Groth. A verifiable secret shuffle of homomorphic encryptions. pages 145–160, 2003.

GZJ+02. Golle, G. Zhong, Boneh B. Jakobsson, , and Juels. Optimistic mixing for exit-polls. 2002.

HK04. Alejandro Hevia and Marcos A. Kiwi. Electronic jury voting protocols. *Theor. Comput. Sci.*, 321(1):73–94, 2004.

HS00. Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Eurocrypt*, volume 1807 of *Lecture Notes in Computer Science*, pages 539+, 2000.

Jak98. Markus Jakobssen. A practical mix. *Lecture Notes in Computer Science*, 1403:448, 1998.

JJR02. M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. 2002.

JL97. Wen-Sheng Juang and Chin-Laung Lei. A secure and practical electronic voting scheme for real world environment. *TIEICE: IEICE Transactions on Communications Electronics Information and Systems*, 1997.

JLY98. Wen-Sheng Juang, Chin-Laung Lei, and Pei-Ling Yu. A verifiable multi-authorities secret elections allowing abstaining from voting. *International Computer Symposium*, 1998.

JSI96. Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. *Lecture Notes in Computer Science*, 1070:143–??, 1996.

KKLA01. K. Kim, J. Kim, B. Lee, and G. Ahn. Experimental design of worldwide internet voting system using pki. 2001.

KMO01. Jonathan Katz, Steven Myers, and Rafail Ostrovsky. Cryptographic counters and applications to electronic voting. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 78–92, London, UK, 2001. Springer-Verlag.

KR05. Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. *Lecture Notes in Computer Science*, 3444:186–200, 2005.

KY02. Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In *PKC '02: Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems*, pages 141–158, London, UK, 2002. Springer-Verlag.

LBD+03. Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In Jong In Lim and Dong Hoon Lee, editors, *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 245–258. Springer, 2003.

LK00. B. Lee and K. Kim. Receipt-free electronic voting through collaboration of voter and honest verifier, 2000.

LK02. B. Lee and K. Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer, 2002.

Low97. Gavin Lowe. A hierarchy of authentication specifications. In *CSFW '97: Proceedings of the 10th Computer Security Foundations Workshop (CSFW '97)*, page 31, Washington, DC, USA, 1997. IEEE Computer Society.

Nef01. A. Neff. A verifiable secret shuffle and its application to e-voting. 2001.

OA00. Miyako Ohkubo and Masayuki Abe. A length-invariant hybrid mix. In *ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security*, pages 178–191, London, UK, 2000. Springer-Verlag.

Oka97. Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Security Protocols Workshop*, pages 25–35, 1997.

OKST97. Wakaha Ogata, Kaoru Kurosawa, Kazue Sako, and Kazunori Takatani. Fault tolerant anonymous channel. In *ICICS '97: Proceedings of the First International Conference on Information and Communication Security*, pages 440–444, London, UK, 1997. Springer-Verlag.

OMA+99. Miyako Ohkubo, Fumiaki Miura, Masayuki Abe, Atsushi Fujioka, and Tatsuaki Okamoto. An improvement on a practical secret voting scheme. In Masahiro Mambo and Yuliang Zheng, editors, *ISW*, volume 1729 of *Lecture Notes in Computer Science*, pages 225–234. Springer, 1999.

PBD05. Kun Peng, Colin Boyd, and Ed Dawson. Simple and efficient shuffling with provable correctness and zk privacy. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 188–204. Springer, 2005.

PIK93. Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *EUROCRYPT*, pages 248–259, 1993.

PP99. Pascal Paillier and David Pointcheval. Efficient public-key cryptosystems provably secure against active adversaries. In *ASIACRYPT '99: Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security*, pages 165–179, London, UK, 1999. Springer-Verlag.

QGAB89. Jean-Jacques Quisquater, Louis Guillou, Marie Annick, and Tom Berson. How to explain zero-knowledge protocols to your children. In *CRYPTO '89: Proceedings on Advances in cryptology*, pages 628–631, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

Rad95. Michael J. Radwin. *An Untraceable, Universally Verifiable Voting Scheme*. December 1995.

Rja02. Zuzana Rjaskova. *Electronic Voting Schemes*. PhD thesis, Comenius University, Bratislava, 2002.

RSA77. Ronald L. Rivest, Adi Shamir, and Leonard Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Number MIT/LCS/TM-82. 1977.

Sak94. Kazue Sako. Electronic voting scheme allowing open objection to the tally (special section on cryptography and information security). *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 77(1):24–30, 1994.

Sch. Berry Schoenmakers. Fully auditable electronic secret-ballot elections.

Sch99. Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. *Lecture Notes in Computer Science*, 1666:148–164, 1999.

Sha79. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

Sho05. Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005.

SK94. Kazue Sako and Joe Kilian. Secure voting using partially compatible homomorphisms. In *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, pages 411–424, London, UK, 1994. Springer-Verlag.

SK95. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In *EUROCRYPT*, pages 393–403, 1995.

Web06. Stefan Weber. A coercion-resistant cryptographic voting protocol - evaluation and prototype implementation. Master's thesis, Darmstadt University of Technology, 2006.