Electronic Voting in a Large-Scale Distributed System

Gianluca Dini
Dipartimento di Ingegneria della Informazione: Elettronica, Informatica, Telecomunicazioni, Via Diotisalvi 2, 56126 Pisa, Italy

In this paper, we propose a practical and secure electronic voting scheme suitable for large-scale distributed systems such as the Internet. As in most of the voting systems in the literature, the proposed voting scheme allows only eligible voters to vote once, protects the voters' privacy, and ensures the accuracy of the tally. In addition, the proposed scheme satisfies further important criteria, namely, it tolerates abstention and does not require voters to control whether their votes have been correctly processed. If a subset of voters does not vote, the elections cannot be disrupted. Furthermore, anyone, including an external observer, can easily be convinced that the election is fair, that is, that the published final tally is fairly computed from the ballots that were correctly cast. It follows that the proposed voting scheme strengthens the security properties of the electronic voting procedure, simplifies the interaction of voters with the electronic voting system, and contributes to increased voter confidence in the degree of security of the electronic voting procedure. © 2001 John Wiley & Sons, Inc.

Keywords: distributed systems; applied cryptography; security and privacy; voting protocols

1. INTRODUCTION

The development of cryptographic techniques and recent advances in computer and network technology allow us to carry out several traditional activities electronically over a large-scale distributed system such as the Internet. Electronic commerce, secure auctions, and secure elections are relevant examples of these activities. Unfortunately, many of the procedures and laws that have been developed to ensure the security of these types of activities becomes insufficient when these activities are performed electronically. Cryptographic protocols provide a response to this need for new ways of ensuring security [13]. In this paper, we consider the problem of building a secure election system which ensures the voters' privacy and accuracy of votes and is suitable for a large-scale distributed system.

The problem of deploying secure elections in a large-scale distributed system has been extensively investigated, and many solutions have been proposed [1, 2, 3, 5, 6, 9, 10, 15, 17, 18, 21]. Most of these, especially the earlier ones, deal with theoretical issues and are thus not practical for large-scale distributed elections. Recently, systems that are more suitable for a large-scale environment have been proposed. Relevant examples are the voting system proposed by Cranor and Cytron called Sensus [11], the voting scheme proposed by Fujioka et al. [14], upon which Sensus is built, and the system proposed by Chang and Wu [5]. These systems have basic requirements for effective use in a large-scale system: Only eligible voters can vote, any eligible voter can vote only once, and the voters' privacy is protected even from the voting center itself. In addition, the voting center cannot present a false tally without being discovered. Finally, no voter-to-voter interaction is required. However, the security of these systems is based upon two assumptions that strongly limit their effective deployment in a large-scale system: The first assumption is that all voters cast their votes. The second assumption is that the voters check that their votes are correctly processed. If these assumptions are not satisfied, the voting center can surreptitiously add votes of its own, thus disrupting the election results.

As to the former assumption, we believe it is too rigid and unrealistic [1]. In real life, there is always a possibility that a voter, either accidentally or intentionally, does not cast his/her vote even after applying for registration. Intentional abstention is nowadays a very common and ever-growing phenomenon. For instance, the abstention percentage in the last elections in Italy amounted to 42–43%. A similar large degree of abstention occurred in the most recent congressional elections in the United States.

In light of these considerations, it soon becomes evident that the latter assumption is also rigid and unreal-

Received October 1999; accepted April 2001
Correspondence to: G. Dini ; e-mail: g.dini@iet.unipi.it
Contract grant sponsor: Italian Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) within the framework of project MOSAICO (Design Methodologies and Tools of High Performance Systems for Distributed Applications) Project

istic. An (intentionally) abstaining voter is not likely to be interested in the election at all. Therefore, it is not reasonable to expect him/her to make any check. Consequently, security breaches may ensue if the security of the elections rests upon the voters' control. The assumption is also questionable for practical reasons. In considering the performance of an electronic-voting system, the effort required from a voter is clearly one of the main issues [11, 17]. While governments can mount a large organizational effort to hold elections, it is mandatory to make the voting protocol as simple and efficient as possible for voters. Requiring voters to check the correct allocation of their votes certainly complicates matters and does not reinforce the voters' confidence in the security level of the electronic procedure.

In this paper, we present two voting schemes that do not require the above assumptions. As in the Sensus system and Chang and Wu system, for instance, the proposed schemes ensure the voters' privacy, prevent double voting, and ensure the accuracy of the tally. In contrast to those systems, the proposed schemes tolerate abstention, that is, if a subset of voters does not vote, the voting center cannot surreptitiously add votes in their place. Thus, abstaining voters need not take any interest in elections if they so wish. In the first proposed scheme, voters who actually cast their votes are still required to check that their votes are correctly counted. The second proposed scheme removes this constraint, thus allowing any given external observer to check that the final tally does indeed correspond to the ballots submitted by the voters. If the observer detects a mistake, the observer can ensure its correction.

The paper is organized as follows: In Section 1.1, we present the set of requirements of the proposed voting schemes. In Section 2, we describe related research and provide some comparisons with other well-known voting schemes. The proposed voting schemes are presented as extensions to the Chang and Wu system, which was the inspiration for this work. In Section 3, we briefly review that system. In Section 4, we show how the aforementioned assumptions come into play. In particular, we show that in the Chang and Wu system the Voting Center can surreptitiously add votes of its own for all those voters who abstain and do not apply the required check. This section constitutes the starting point for introducing the extensions. The first extension, aimed at making the voting protocol tolerant to abstention, is discussed in Section 5. The second extension, aimed at allowing any observer to control the final tally, is discussed in Section 6.

1.1. Requirements

To design a voting protocol which is both secure and suitable for large-scale elections, it is necessary to identify a set of criteria which helps us to achieve those goals. Prior to the beginning of elections, organizers define the *electoral roll*, that is, the list of voters who are *eligible* to vote.

Eligibility. Only eligible voters are allowed to vote.

Double-Voting. Each eligible voter can vote only once.

Privacy. No one but the voter knows which voting strategy he/she has adopted.

In order to vote, any given voter casts a ballot specifying his/her voting strategy. At any moment, the voter may renounce his/her right to vote. In practice, a voter fails to vote by not casting a ballot.

Abstention. Any given voter's failure to cast his/her vote does not disrupt the elections.

Alternatively, a voter may cast additional ballots.

Recasting. Each voter can recast before the deadline for casting ballots expires.

It follows that the voting system may receive several ballots. However, the same voter may have specified more than one candidate. If this is indeed the case, the voting system has to choose one of them so that the Double-Voting requirement is not violated. The voting strategy specified by the chosen ballot will constitute the voter's vote. Typically, the voting system chooses the ballot that was received last [5, 16, 19].

When the deadline for casting ballots expires, the voting system will produce the final tally which must correspond to the ballots submitted by voters.

Accuracy. It is impossible for an invalid vote to be counted in the final tally or for a valid vote to be either altered or eliminated from the final tally.

If the tally contains any mistakes, any given observer can detect it and obtain its correction.

Verifiability. Anyone can independently verify that all votes have been correctly counted.

The requirement Recasting allows any given voter to change his/her mind. By recasting, the voter can cancel his/her vote for a given candidate and vote in favor of another one [5, 17, 19]. Recasting is generally not allowed in customary voting systems. In this kind of system, the procedures aimed at ensuring Privacy and Double-Voting constitute a large, if not unsurmountable, obstacle to Recasting. From this point of view, one can argue that cryptographic techniques offer a range of new possibilities and that seemingly impossible tasks now become possible [19]. On the other hand, the issue of whether it is appropriate to support the requirement Recasting and to what extent it may increase the motivation of an average voter to vote is open to de-

bate. As a consequence, there are different approaches in cryptography-based voting systems dealing with that requirement [1, 5, 11, 16, 17, 19]. However, we would like to emphasize that the central issue of this paper, namely, supporting Abstention and Verifiability requirements, is largely orthogonal to Recasting.

2. RELATED WORK

The problem of deploying secure elections in a large-scale distributed system has been extensively investigated. Several solutions have been proposed from both a theoretical and practical point of view [1, 2, 3, 5, 6, 9, 10, 11, 14, 15, 21]. However, some of these solutions are not practical for large-scale elections. The voting system proposed by Chaum can be disrupted by a single voter [6, 9]. The faulty voter can be traced and the election restarted excluding him/her, but, clearly, the voting system becomes impractical for large-scale elections. The voting scheme proposed by Cohen and Fischer ensures a limited form of the Privacy requirement because it does not protect the voters' privacy from the election authority [10]. The system proposed by Boyd does not meet the Accuracy requirement because the Voting Center can produce a false tally without being detected [2, 3]. The system proposed by Nurmi et al. guarantees the Privacy requirement, allows Recasting, and ensures that no one can change anyone else's vote without being discovered [19]. However, its use in a large-scale setting is limited by the fact that it is built on an ANDOS (all-or-nothing disclosure of secrets) protocol. The communication complexity of this protocol is $O(n^2)$, where n is the number of voters, because it requires voter-to-voter interactions. Consequently, the protocol requires an excessive amount of communication when the number of voters is large. In addition, the system does not completely meet the Accuracy requirement and fails to meet the Abstention requirement. In fact the Voting Center can surreptitiously cast votes for all the abstaining voters. These voters may discover such a violation and report it, but they cannot prove that they did not actually vote. Finally, as an explicit voter check of the final tally is necessary, the system only meets a limited form of the Verifiability requirement.

The more practical voting schemes, for instance Sensus [11], Fujioka et al.'s scheme [14], and Chang and Wu's system [5], do not require interactions between voters. However, as already discussed in Section 1, none of these systems support the Abstention and Verifiability requirements. They actually require a voter's direct intervention to verify if his/her own vote has been correctly counted. In both the Sensus system and Fujoka et al.'s system, the Voting Center can cast votes for all the abstaining voters. Both these voters and an external observer can detect the presence of those invalid votes. However, there is no way of identifying and removing them from the final tally. It follows that the Accuracy requirement is violated. One way to solve this problem is to require abstaining voters to submit blank ballots. In Chang and Wu's system, if the Voting Center casts votes for the abstaining voters, an external observer cannot even detect them. Only abstaining voters can identify the invalid votes and remove them from the final tally.

3. THE CHANG AND WU PROTOCOL

In this section, we review Chang and Wu's protocol (hereafter, the CW protocol) and the corresponding voting system. Details concerning both the voting protocol and its security analysis can be found in [5].

Let us consider an electoral roll composed of n voters that we denote by $U_1, \ldots, U_n$. The CW protocol has two phases: Registration and Voting. In the *Registration Phase*, the *Voting Center* (VC), generates a set of *n valid identifiers* and a set of n packages, each containing the ciphertext of a different identifier. Then, the *Shuffling Center* (SC) shuffles these packages and randomly distributes them, one to each eligible voter. Upon receiving a package, any given voter *unpacks* it and obtains the valid identifier contained therein. No one, including the VC, can know which identifier the voter has obtained. After that, the voter chooses his/her voting strategy and then performs the *Voting Phase*. In this phase, the voter tags his/her vote with the identifier and then *casts* the resulting ballot. If the voter wants to recast, he/she chooses another vote, tags it with the same identifier, and casts the resulting ballot. When the deadline for casting ballots is over, the VC works out the final tally. In doing so, the VC considers only *valid votes*, that is, votes tagged with valid identifiers. If several votes tagged with the same identifier exist, the VC considers only the last one received and discards the remaining ones. If a vote specifies a given candidate, the VC allocates the identifier which tags the vote to that candidate. Finally, the VC announces the result of the election by publishing the candidates and their corresponding identification tags.

For the sake of presentation, let us assume that the voting system supplies voters with two *voting operations*, allowing voters to carry out the voting phases. With reference to a given user U_j, the package o_i distributed to that voter, the identifier α_i contained in o_i, and the vote v_j chosen by voter U_j, the voting operations are

- Operation $\alpha_i = \mathsf{registration}(o_i)$, which takes the package o_i as its argument and returns the identifier α_i packaged in o_i.
- Operation $\mathsf{cast}(\alpha_i, v_j)$, which takes the identifier α_i and the vote v_j as its arguments, tags v_j with α_i, and casts the resulting ballot.

It follows that the voting activity of any given voter U_j can be described by the following pseudocode:

```
Upon (receiving o_i from SC)
begin
    α_i = registration(o_i);          {Registration}
    repeat
    begin
        choose vote v_j;
        cast(v_j, α_i);               {Voting}
    end
    until (voter is ready);
end
```

We shall describe the implementation of the voting operations cast() and registration() in Sections 3.1.1. and 3.1.2., respectively.

3.1. The Protocol

Consider Figure 1. During the Registration and Voting Phase, any given voter U_j interacts with the VC through a Semipublic Board (SPB) service. An SPB is a special form of blackboard service. Each client can write a message onto the blackboard, and the message can then be read by every user. Unlike a blackboard service, the SPB service guarantees both reader and writer anonymity, that is, the SPB guarantees the unlinkability between any given user and the messages the user reads and writes. It follows that the SPB allows a form of reliable anonymous communication between the VC and the voters. This type of communication is fundamental to ensure the voters' privacy. The implementation of this kind of communication is beyond the scope of this work. However, there are several papers dealing with this issue [6, 20, 21, 22].

3.1.1. The Registration Phase The Registration Phase begins with a prologue that involves the VC and the SC. Initially, the VC chooses a large prime number p and a primitive element e over a Galois Field $GF(p)$ and makes them public. Next, the VC randomly chooses n numbers $\alpha_1, \ldots, \alpha_n$ and writes the values $\{e^{\alpha_i} \bmod p\}_{i=1\ldots n}$ to the SPB. Then, the VC randomly chooses a secret key d over $GF(p)$, such that $\gcd(d, p-1) = 1$, computes *n packages*, $\{o_i = (e^{\alpha_i} \bmod p, \alpha_i^d \bmod p)\}_{i=1\ldots n}$, and sends them to the SC. Upon receiving these packages, the SC shuffles them and sends each of them to a different eligible voter.

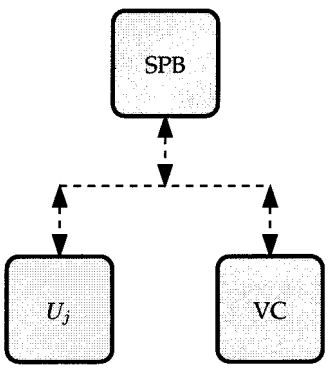

FIG. 1. The architecture of the CW system.

Upon receiving a package o_i from the SC, a given voter U_j carries out the operation α_i = registration(o_i) whose execution produces the following actions:

RP1: Voter U_j randomly chooses a secret key s_j, such that $\gcd(s_j, p-1) = 1$, computes $a_j = (\alpha_i^d)^{s_j} \bmod p$, and sends it to the SPB.
RP2: Upon reading a_j from the SPB, the VC computes $a_j' = (a_j)^{d^{-1}} = \alpha_i^{s_j} \bmod p$, with d^{-1} the inverse of d over $GF(p)$, and sends a_j' back to the SPB.
RP3: Finally, upon reading a_j' from the SPB, U_j computes $\alpha_i = (a_j')^{s_j^{-1}} \bmod p$, and verifies it using the first field in o_i.

3.1.2. The Voting Phase With reference to the ElGamal public key cryptosystem [12], let x_j and y_j be the secret and the public keys of voter U_j, respectively, such that $y_j = e^{x_j} \bmod p$. Furthermore, let x_0 and y_0 be the secret and the public keys of the VC, respectively, such that $y_0 = e^{x_0}$. Finally, let α_i be the identifier that voter U_j has received during the Registration Phase. Voter U_j casts a given vote v_j by calling the operation cast(α_i, v_j) whose execution produces the following actions:

VP1: U_j chooses a random number r_j over $GF(p)$ and computes the quantity $c_{j,1} = e^{r_j} \bmod p$ and the quantity $c_{j,2} = y_0^{r_j} \oplus v_j \bmod p$, where v_j is the voting strategy of voter U_j and $\oplus$ denotes the exclusive-or operator.
VP2: U_j encrypts $(\alpha_i, c_{j,2})$ by using a public key cryptosystem and sends the VC the ciphertext.
VP3: VC decrypts the ciphertext and acknowledges the reception by publishing the value $c_{j,2}$.
VP4: Upon reading $c_{j,2}$ from the SPB, U_j sends the ciphertext of $(\alpha_i, c_{j,1})$ to the VC.
VP5: The VC decrypts the ciphertext and publishes $(c_{j,1}, c_{j,2})$.

If voter U_j wants to recast, he/she has to choose another vote v_j' and then call operation cast (α_i, v_j').

When the deadline for casting ballots is over, the VC initially determines the votes cast by voters. For instance, for each pair $(c_{j,1}, c_{j,2})$, the VC computes $v_j = c_{j,1}^{x_0} \oplus c_{j,2}$. If the VC has received two or more pairs, the VC considers only the last pair received. After that, the VC announces the result of the election by publishing candidates and their corresponding identification tags.

3.2. Functional Properties of the CW Protocol

In this section, we report the functional properties of the CW protocol and the assumptions upon which those properties are based. For the proofs of the properties, readers can refer to the original paper by Chang and Wu [5] .

The assumptions made by Chang and Wu are as follows:

A1: The SC is trusted. In particular, the SC and VC are fully independent, that is, they do not work in cooperation.
A2: The SPB is trusted to ensure writer anonymity and reader anonymity. In particular, the SPB does not drop messages and is fully independent of the VC, that is, the SPB and VC do not work in cooperation.
A3: Solving the discrete logarithm is not computationally feasible.
A4: Each voter checks that his/her vote has been correctly counted in the final tally.

The Registration Phase has the following properties:

Property 1. *It is impossible for anyone, including the VC itself, to link a voter to his/her identification tag.*

Property 2. *It is impossible for an intruder to derive either a valid identifier $\alpha_i, i \in \{1, \ldots, n\}$ or the secret key d of the VC.*

The Voting Phase has the following properties.

Property 3. *The protocol supports the Recasting requirement.*

Property 4. *It is impossible for anyone, including the VC, to link a voter with his/her voting strategy.*

The CW protocol is resilient to any malicious behavior by the VC to produce a false tally. In particular, the VC can neither add surreptitiously invalid votes nor omit counting correctly valid votes. To surreptitiously add an invalid vote, the VC might choose an invalid identifier $\alpha \notin \{\alpha_1, \ldots, \alpha_n\}$ and then cast a vote of its own tagged with α. However, the following property states that this is impossible:

Property 5. *It is impossible that an invalid vote is added to the final tally without being detected.*

In practice, any external observer is able to detect that α is invalid and obtain its removal from the final tally.

As the VC cannot add invalid votes, it might attempt to incorrectly count valid ones. To correctly count a valid vote, the VC must allocate the valid identifier tagging that vote to the candidate specified by the vote. It follows that the VC might *alter* a vote by allocating the identifier tagging the vote to a different candidate. Alternatively, the VC might *eliminate* a vote by failing to allocate the identifier tagging the vote to any candidate. However, any attempt to either alter or eliminate a vote is bound to fail. According to assumption A4, any given voter checks that his/her valid identifier is correctly allocated to the chosen candidate. If a voter detects that this is indeed not the case, he/she can produce evidence of the mistake and ensure that the VC corrects it. Therefore, the following property holds:

Property 6. *The VC cannot incorrectly count a valid vote without being detected by the voter who cast that vote.*

4. AN ATTACK ON CHANG AND WU'S PROTOCOL BASED ON ABSTENTION

The CW protocol is based upon the assumption that all voters cast their votes and that they exert some control on the VC. In fact, as explained in the previous section, the voter is the only one able to detect whether his/her vote has been correctly counted. In this section, we show that if one or more voters abstain from both voting and exerting the required control the VC can surreptitiously add the same number of valid votes of its own.

Let α_i be a valid identifier obtained by a given voter during the Registration Phase. We call this voter the *legitimate holder* of identifier α_i and denote him/her by $U(\alpha_i)$. According to Property 1, no one, including the VC, can know who $U(\alpha_i)$ is. For the sake of simplicity, let us initially suppose that the VC knows that α_i will not be used in any execution of the Voting Phase. Thus, the VC can perform an execution of the Voting Phase by using that identifier and a vote of its own. In so doing, the VC surreptitiously adds a valid vote to the final tally. The VC can repeat this procedure for all those identifiers that will not be used, so surreptitiously adding the same number of valid votes to the final tally. Notice that, according to Property 4, an external observer cannot realize that it is the VC, and not the legitimate holder $U(\alpha_i)$, that has cast a vote tagged with α_i. According to Property 6, only $U(\alpha_i)$ can detect the abuse of identifier α_i, but this voter abstains and does not perform such a control.

Now, we remove the initial simplifying assumption and assume that the VC does not know which identifiers will not be used. Thus, the VC has to make a guess. However, it can make such guesses without being detected as follows: Let $\Delta = [t_b, t_c]$ be the interval of time during which the Voting Phase takes place. Furthermore, let identifier α_i be unused at time $t \in \Delta$, that is, no vote tagged with that identifier has been cast at time t. It follows that, at time t, the VC can assume that $U(\alpha_i)$ will abstain and, consequently, perform an execution of the Voting Phase using identifier α_i. If voter $U(\alpha_i)$ really abstains, the VC succeeds in surreptitiously adding a vote as described above. Suppose instead that $U(\alpha_i)$ performs the Voting Phase at a given time $t' \in \Delta, t' > t$. Once again, according to Property 4, an external observer cannot detect the VC's attempt to abuse the identifier α_i. In addition, even voter $U(\alpha_i)$ remains unaware of that attempt, provided that the VC correctly allocates his/her vote in the final tally.

Let α_i be unused at a given time $t \in \Delta$. The VC can use t to estimate the probability that voter $U(\alpha_i)$ will abstain and, thus, that its guess is good. For instance, let us assume, for the sake of simplicity, that the expected

overall abstention probability is p_0 and that the probability of a voter's abstention grows linearly with respect to time. It follows that the probability that $U(\alpha_i)$ abstains conditioned on α_i being unused at time t is

$$\Pr\{U(\alpha_i) \text{ abstains} \mid \alpha_i \text{ is unused at time } t\} = p_0 + (1 - p_0) \times \frac{t - t_b}{t_c - t_b}.$$

Therefore, the closer t is to t_c, the higher is the probability that $U(\alpha_i)$ abstains. Consequently, the VC could use identifiers that are still unused toward the end of the Voting Phase to increase the probability of succeeding in surreptitiously adding votes of its own to the final tally.

5. AN EXTENSION OF A SECURE VOTING SYSTEM ON A PUBLIC NETWORK

The CW protocol is exposed to the attack described in Section 4 because there is no way for an external observer to ascertain whether a vote really comes from the legitimate voter or not. In this section, we propose an extension of the CW protocol which solves this problem and still guarantees that the link between votes and voters is not revealed. The resulting extended CW protocol, hereafter the ECW protocol, makes it possible to tolerate abstentions while still maintaining voters' privacy. In addition, protocol ECW weakens the requirements on voters' behavior. Only voters who actually cast their votes, *actual voters*, are required to check whether their votes have been correctly processed, whereas the others, *abstaining voters*, need take no interest at all in the election, if they so wish.

Protocol ECW is obtained from protocol CW by adding the *Validation Phase*, carried out between the Registration Phase and the Voting Phase. In the Registration Phase, the voter obtains a valid identifier α_i, as described in Section 3.1.1. Then, in the Validation Phase, the voter *validates* that identifier and obtains a *validated* identifier α_i'. Finally, the voter chooses his/her voting strategy and performs the Voting Phase. In this phase, the voter tags his/her vote with the validated identifier and casts the resulting ballot. If the voter wants to recast, he/she chooses another vote, tags it with the same identifier, and casts the resulting ballot. The Voting Phase takes place as described in Section 3.1.2., provided that the identifier α_i is replaced with the corresponding validated identifier α_i'. When the deadline for casting ballots is over, the VC works out the final tally. In doing so, the VC only considers valid those votes tagged with validated identifiers. The VC announces the election results by publishing the candidates with their validated identifiers allocated.

Informally, the Validation Phase guarantees that only the legitimate holder of an identifier can validate it. It follows that the VC cannot exploit an unused identifier to surreptitiously add a valid vote because the VC cannot validate that identifier. Any given external observer is able to detect whether an invalid vote is present in the final tally. If this is indeed the case, the external observer can produce evidence of this and ensure that the VC removes the invalid vote from the final tally.

The addition of the Validation Phase causes the introduction of a new voting operation:

- Operation $\alpha_i' = \mathsf{validation}(\alpha_i)$, which takes the identifier α_i as its argument and returns the validated identifier α_i'.

Consequently, the voting activity of any given voter U_j can be described by means of the following pseudocode:

```
Upon (receiving o_i from SC);
begin
    α_i = registration(o_i);          {Registration}
    α_i' = validation(α_i);           {Validation}
    repeat
    begin
        choose vote v_j;
        cast(v_j, α_i');              {Voting}
    end
    until (voter is ready);
end
```

We describe the implementation of validation() in Section 5.2.

With reference to Figure 2, the Validation Phase is carried out through the *Validation Center* (VAC) which maintains a copy of the electoral roll. Any given voter interacts with the VAC through a bidirectional communication channel. The VAC is trusted. In particular, we assume that the VAC is fully independent from the VC, that is, they do not collude.

Let α_i be the identifier a given voter U_j has obtained in the Registration Phase. When voter U_j sends α_i to the VAC for validation, the VAC initially authenticates U_j and ascertains his/her authorization to vote. Then, the VAC validates the identifier α_i by digitally signing it.

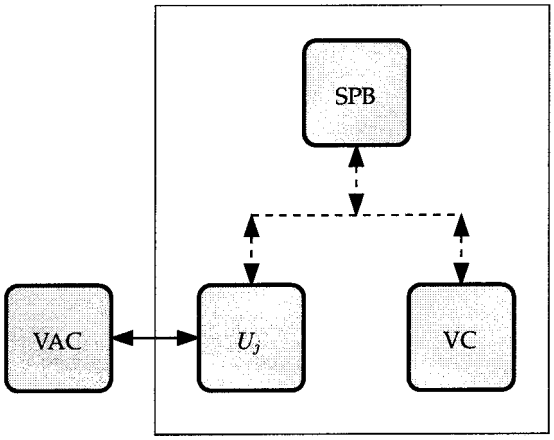

FIG. 2. The architecture of the ECW system. The rectangular box highlights the architecture of the original CW system.

It follows that a validated identifier α_i' consists of two fields, the identifier α_i and its signature $\sigma_{vac}(\alpha_i)$ by the VAC, that is, $\alpha_i' = \sigma_{vac}(\alpha_i)$. To preserve the voters' privacy, the link between a given validated identifier and its legitimate holder must not be disclosed to anyone, including the VAC itself. Therefore, VAC signs identifiers by means of a *blind signature* protocol.

5.1. Blind Signatures

A blind signature is a special form of digital signature. Just as in any digital signature scheme, only the signer can create blind signatures by means of his/her private signing function, while anyone can verify the signature against the public signature verification function of the signer. Unlike a normal digital signature scheme, the signer does not learn which messages he/she is actually signing. Moreover, the signer does not know which blind signatures he/she is actually creating, that is, the signer cannot correlate a given signed message with the act of signing it, even though the signer keeps a record of every blind signature he/she makes.

Several blind signature schemes exist: The original scheme proposed by Chaum [6, 7] is based upon RSA [23]. As a further example, Camenisch et al. [4] proposed a scheme based on the discrete logarithm problem [24]. For the sake of generality, in this paper, we consider blind signatures in an abstract form, without committing to a specific scheme.

By σ_a, we denote the *signature* function of a given principal A, and by σ_a^{-1}, the corresponding *verification* function. Principal A keeps σ_a secret, but makes σ_a^{-1} publicly known. We denote by $\sigma_a(x)$ the digital signature of a given quantity x by (means of the secret function σ_a of) principal A. Any given principal can verify such a signature by obtaining the public verification function σ_a^{-1} of principal A, calculating $y = \sigma_a^{-1}(\sigma_a(x))$, and, finally, ascertaining that $y \equiv x$. Any given principal can obtain the public verification function of any other given principal as needed.

We assume that functions σ_a and σ_a^{-1} are cryptographically strong, that is, although the pair of functions σ_a and σ_a^{-1} are mathematically related, it is not computationally feasible to derive one function from the knowledge of the other. Thus, although many people may know the verification function σ_a^{-1} of a given principal A and use it to verify the principal's signatures, they can neither discover that principal's signature function σ_a nor forge his/her digital signatures.

Let us suppose now that a given principal A wants to obtain the blind signature of a given information item x by a given signer B. Initially, principal A chooses a *commuting* function c_a and its inverse function c_a^{-1} which satisfy the following properties: (i) $c_a^{-1}(\sigma_b(c_a(x))) = \sigma_b(x)$; (ii) it is not possible to link $c_a(x)$ to x, even knowing both quantities; (iii) it is not possible to link $c_a(x)$ to $\sigma_b(x)$, even knowing both quantities; and (iv) both functions c_a and c_a^{-1} are known only to principal A. Then, principal A calculates $c_a(x)$ and sends this quantity to B. Upon receiving it, B signs it by means of σ_b and returns the signed quantity $\sigma_b(c_a(x))$ to A. Upon receiving this quantity, A calculates $c_a^{-1}(\sigma_b(c_a(x)))$ and obtains $\sigma_b(x)$.

Anyone can verify that quantity $\sigma_b(x)$ has been produced by B. However, on the basis of hypotheses ii–iv, the signer B cannot learn which information he/she is signing. Furthermore, he/she cannot establish any correspondence between $\sigma_b(x)$ and $\sigma_b(c_a(x))$ and therefore determine which blind signature he/she is creating [7].

Quantity x must satisfy certain redundancy constraints in order to make the search for valid signatures impractical [7, 8]. In the following, we shall assume that this requirement is satisfied.

5.2. The Validation Phase

Let α_i be the identifier a given voter U_j has obtained in the Registration Phase. Voter U_j validates α_i by calling the operation $\alpha_i' = \mathsf{validate}(\alpha_i)$, whose execution produces the following actions:

VAP1: Voter U_j selects a commuting function c_{u_j} and its corresponding inverse function $c_{u_j}^{-1}$. Then, U_j signs quantity $c_{u_j}(\alpha_i)$ so obtaining $\sigma_{u_j}(c_{u_j})$. Finally, voter U_j sends the VAC a message containing both $c_{u_j}(\alpha_i)$ and $\sigma_{u_j}(c_{u_j}(\alpha_i))$.

VAP2: Upon receiving this message, the VAC authenticates the voter by verifying the signature $\sigma_{u_j}(c_{u_j}(\alpha_i))$, and accesses the electoral roll to ascertain the voter's authorization (if any of these actions fails, the Voting Protocol is unsuccessfully terminated). Then, the VAC signs $c_{u_j}(\alpha_i)$ and sends U_j a message containing the resulting signed quantity $\sigma_{vac}(c_{u_j}(\alpha))$.

VAP3: Upon receiving this message from the VAC, voter U_j computes $\sigma_{vac}(\alpha_i) = c_{u_j}^{-1}(\sigma_{vac}(c_{u_j}(\alpha_i)))$ and verifies signature $\sigma_{vac}(\alpha_i)$ (if this action fails, the Voting Protocol is unsuccessfully terminated). Finally, U_j builds a valid identifier α_i' by pairing α_i with $\sigma_{vac}(\alpha_i)$.

5.3. Discussion

In this section, we argue that our protocol meets all the requirements mentioned in Section 1.1 except for the Verifiability requirement. In fact, protocol ECW only satisfies a weaker form of this requirement. As we shall argue, an external observer can check that no invalid vote is added to the final tally. Nevertheless, checking that valid votes are correctly counted remains a task for actual voters.

The security of the protocol ECW is based on assumptions A1–A3 and on the the following assumptions:

A4′: Each actual voter checks that his/her vote has been correctly counted.

A5: The VAC is trusted. In particular, the VAC and VC are fully independent, that is, they do not work in cooperation.

Notice that assumption A4′ is weaker than is assumption A4 since abstaining voters are not required to exert any check.

In the rest of this section, we shall proceed as follows: First, we shall prove some of the functional properties of the protocol. Then, we shall argue that these properties guarantee that the protocol satisfies the requirements.

5.3.1. Functional Properties of the Extended Protocol Since both the Registration Phase and the Voting Phase of the ECW protocol are the same as those of the CW protocol, Properties 1–5 continue to be satisfied under Assumptions A1–A3. As to Property 5, it should be noted that, in the ECW protocol, a valid vote is tagged with a validated identifier α'. Thus, any given observer can ascertain whether an identifier in the final tally is validated or not by checking whether the first field specifies a valid identifier and the second field specifies the signature of that identifier by the VAC.

The Validation Phase satisfies the following properties:

Property 7. *It is impossible for anyone, including the VAC, to link a voter to his/her validated identifier.*

Proof. The proof directly derives from Property 1 and the properties of the blind signature scheme used in the Validation Phase. ∎

Property 7 implies that the Validation Phase maintains the unlinkability between voters and identifiers.

Property 8. *The VC cannot obtain a validated identifier.*

Proof. To validate a given identifier α_i, it is necessary to obtain quantity $\sigma_{vac}(\alpha_i)$. Thus, the proof consists in showing that it is impossible for the VC to obtain that quantity. On the basis of assumption A5, the VC cannot obtain $\sigma_{vac}(\alpha_i)$ through an execution of the Validation Phase because the VAC discovers at step VAP2 that the VC is not an eligible voter. Furthermore, since the signature and the verification functions are cryptographically strong, no principal, including the VC, can either forge the signature $\sigma_{vac}(\alpha_i)$ or impersonate a voter in the Validation Phase. ∎

Property 9. *The VC cannot add valid votes surreptitiously.*

Proof. To surreptitiously add a valid vote, the VC needs a validated identifier. However, the VC cannot obtain such an identifier (Property 8). ∎

Property 9 prevents the VC from carrying out the attack described in Section 4. It follows that a voter can freely abstain and take no part in the election process if he/she so wishes. However, actual voters are still required to check the correct allocation of their votes.

Property 10. *The VC cannot omit correctly counting a valid vote without being detected by the actual voter who cast that vote.*

Proof. The proof closely follows that of Property 6 [5]. Let v_j be a valid vote tagged with a given valid identifier α'_i and let U_j be the voter who cast v_j. If the VC does not properly allocate v_j, voter U_j detects this by assumption A4′ and points it out by presenting $(\alpha'_i, C_{j,2}, C_{j,1})$ to a third party. By step VP3 of the Voting Phase, the third party realizes that U_j is right and, consequently, makes the VC correct the result. ∎

5.3.2. Correctness Proof of the Voting Protocol

Proposition 11 Eligibility. *The protocol satisfies the Eligibility Requirement.*

Proof. The proof follows directly from the following observations: First, identification tags are distributed only to eligible voters. Second, according to Property 2, it is impossible for an intruder to derive a valid identifier. ∎

Proposition 12 Recasting. *The protocol satisfies the Recasting Requirement.*

Proof. The proof follows directly from Property 3. ∎

Proposition 13 Double-Voting. *The protocol satisfies the Double-Voting Requirement.*

Proof. On the basis of Property 5 and Proposition 12, to vote twice or more, an eligible voter needs the same number of validated identifiers. Thus, the proof lies in showing that an eligible voter can obtain at most one validated identifier. As a validated identifier α'_i is basically a valid identifier α_i signed by the VAC, the proof is reduced to showing that an eligible voter can obtain at most one valid identifier. This is proved as follows: In the Registration Phase, the voter obtains one valid identifier and cannot derive another valid identifier (Property 2). ∎

Proposition 14 Privacy. *The protocol satisfies the Privacy Requirement.*

Proof. The proof follows directly from Properties 1, 7, and 4. ∎

Proposition 15 Abstention. *The protocol satisfies the Abstention Requirement.*

Proof. The proof follows from observing that the VC cannot exploit the identification tag of an abstaining voter to surreptitiously add a valid vote (Property 9). ▪

Proposition 16 Accuracy. *The protocol satisfies the Accuracy Requirement.*

Proof. The proof follows directly from Properties 9 and 10. ▪

6. A FURTHER EXTENSION

In the extended protocol ECW only actual voters are required to check whether their votes have been correctly accounted for, that is, every actual voter must ascertain that his/her vote has been neither eliminated nor altered. In this section, we present a further extension to the original CW protocol that makes it possible to remove this constraint. The resulting protocol, EECW, allows any given observer to check that the VC correctly counts valid votes. It follows that no voter intervention is required and thus the Requirement Verifiability is fully satisfied.

6.1. Preventing Vote Elimination

As discussed in Section 3.2, the VC may eliminate a vote by eliminating the identifier tagging it. It follows that one way to prevent the VC from eliminating a vote is to keep track of the validated identifier tagging that vote. This task is accomplished by a *Logging Center* (LC). With reference to Figure 3, the LC reads the SPB during the Voting Phase and keeps track of validated identifiers used to tag votes. The LC stores the validated identifiers in stable storage and makes them available to any given external observer. The LC is trusted. In particular, it is fully independent from the VC, that is, the LC and VC do not collude.

With reference to the Validation Phase (Section 3.1.2.), LC reads validated identifiers from messages at steps VP2 and VP4. However, the contents of those messages are encrypted by means of a VC's public key. It follows that the LC really reads validated identifiers in their encrypted form. However, when the deadline for casting ballots expires, the VC's private key is made available to the LC, which can thus obtain the cleartext of the validated identifiers.

According to the Recasting Requirement, the SPB may contain one or more validated identifiers whose first fields coincide but whose second fields differ. These identifiers are based on the same valid identifier but account for different votes. Therefore, the LC records only the last validated identifier which is read and discards the remaining ones.

6.2. Preventing Vote Alteration

As discussed in Section 3.2, the VC could alter a vote by allocating the identifier tagging the vote to a different candidate from the one specified by the vote. It follows that one way to prevent the VC from altering a vote is to indissolubly link the vote to the identifier tagging it. Therefore, if the identifier is allocated, it cannot be done so incorrectly. One way to obtain such a link is as follows:

Let α_i be the identifier voter U_j has obtained in the Registration Phase. The voter chooses a given vote v_j, computes $\alpha_i \| v_j$, where $\|$ denotes the concatenation operator, and then performs the Validation Phase using $\alpha_i \| v_j$ instead of α_i. This phase takes place as specified in Section 5.2 and thus the VAC blindly signs the quantity $\alpha_i \| v_j$. At the end of that phase, voter U_j obtains the quantity $\sigma_{vac}(\alpha_i \| v_j)$ that indissolubly links α_i to v_j. A validated identifier α_i'' is now given by the pair $(\alpha_i, \sigma_{vac}(\alpha_i \| v_j))$. Finally, the voter performs the Voting Phase, which takes place as described in Section 3.1.2. provided that α_i'' replaces α_i. The voting procedure of voter U_j can now be described by the following pseudocode:

```
Upon (receiving o_i from SC);
begin
    α_i = registration(o_i);            {Registration}
    repeat
    begin
        choose vote v_j;
        α_i'' = validation(α_i || v_j);  {Validation}
        cast(v_j, α_i'');                {Voting}
    end
    until (voter is ready);
end
```

It should be noted that, since a validated identifier indissolubly links an identifier with the vote tagged with it, the valid identifier can only be used for one execution of the Voting Phase. Consequently, if a given voter wants to cast a different ballot, before repeating the Voting Phase, the voter must repeat the Validation Phase and obtain another validated identifier according to the new voting strategy. Clearly, repeating the Validation Phase adversely affects the performance of the voting system. However, since these repetitions take place only when voters want to recast their votes, and since we expect that this seldom happens, the effect on the performance should be negligible.

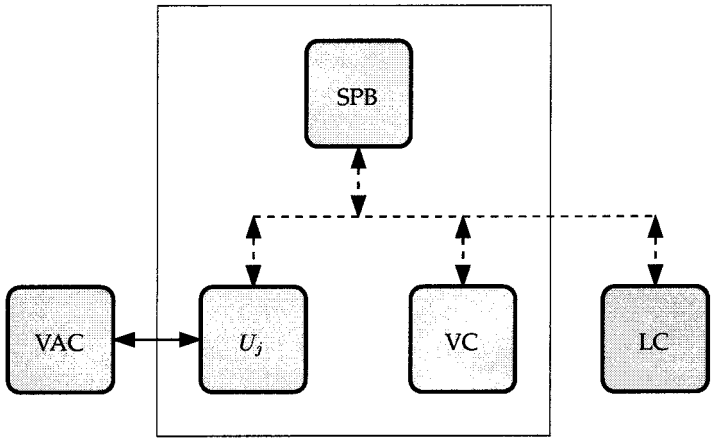

FIG. 3. The architecture of the EECW system.

6.3. *Enforcing Tally Validity*

An external observer can check the tally validity as follows:

C1: Initially, the external observer ascertains that no vote has been eliminated. To do this, the observer cross-checks the final tally with respect to the set of identifiers maintained by the LC. Every validated identifier that is present in the set must also be present in the final tally. If not, the VC has to correct the final result.

C2: Then, the external observer ascertains whether any vote has been altered. For each validated identifier that is present in the final tally, the external observer repeats the following actions: Let α_i'' be a validated identifier allocated to a given candidate C. The external observer initially obtains v_j from α_i'' by applying the verification function σ_{vac}^{-1} to $\sigma_{vac}(\alpha_i \| v_j)$, the second field of α_i''. Then, the observer ascertains whether v_j actually specifies C. If not, VC has to correct the result.

6.4. *Discussion*

In this section, we argue that the EECW protocol meets all the requirements mentioned in Section 1.1. The security of the EECW protocol is based on assumptions A1–A3 and A5 in addition to the following assumption:

A6: The LC is trusted. In particular, the LC and VC are fully independent, that is, they do not work in cooperation.

Given assumptions A1–A3 and A5, Properties 1–5, 7–9 continue to be valid in the EECW protocol. Consequently, Propositions 11–15 are valid. Proofs can easily be deduced by replacing α_i' with α_i''. Thus, it remains to be shown that protocol EECW meets the requirements of Verifiability and Accuracy.

Neither assumption A4 nor assumption A4′ is required. Therefore, in the EECW protocol, no voter, including actual voters, is required to check the correct processing of his/her vote. As we now show, this control can be successfully performed by any observer.

Property 17. *The VC cannot eliminate a vote without being detected by any given external observer.*

Proof. If the VC eliminates a vote, an external observer detects this mistake at step C1. Thus, the VC has to correct the final result. ∎

Property 18. *The VC cannot alter a vote without being detected by any given external observer.*

Proof. If the VC alters a vote, an external observer detects this mistake at step C2. Thus, the VC has to correct the final result. ∎

Property 19. *The VC cannot avoid correctly counting a vote without being detected by any given external observer.*

Proof. In order not to correctly count a vote, the VC can either alter or eliminate it. However, these actions are not possible according to Properties 17 and 18. ∎

Proposition 20 Verifiability. *The protocol satisfies the Verifiability Requirement.*

Proof. The proof follows directly from Property 19. ∎

Proposition 21 Accuracy. *The protocol satisfies the Accuracy Requirement.*

Proof. The proof follows from Properties 9 and 19. ∎

7. CONCLUSIONS

In this paper, we have presented a practical and se-

cure electronic voting scheme for large-scale distributed systems such as the Internet. In addition to Eligibility, Double-Voting, Privacy, and Accuracy requirements, which are generally satisfied by most of the systems in the literature, the proposed voting scheme tolerates the Abstention requirement and does not require voters to exert any control over the correct processing of their votes (Verifiability requirement). It follows that the voting scheme strengthens the security properties of the electronic-voting procedure, simplifies the interaction of voters with the electronic-voting system, and contributes to increased voter confidence in the security level of the electronic procedure.

Acknowledgements

The author is indebted to the anonymous referees for their comments and suggestions, which helped enormously to improve the paper.

REFERENCES

[1] A. Baraani-Dastjerdi, J. Pieprzyk, and R. Safavi-Naini, A secure voting protocol using threshold schemes, Proc 11th IEEE Ann Computer Security Application Conf, 1995, pp. 143–148.

[2] C. Boyd, Some applications of multiple keys ciphers, Proc Advances in Cryptology–EUROCRYPT 88, 1988, pp. 455–467.

[3] C. Boyd, A new multiple keys cipher and an improved voting scheme, Proc Advances in Cryptology–EUROCRYPT 89, 1989, pp. 617–625.

[4] J. Camenisch, J.-M. Piveteau, and M. Stadler, Blind signatures based on the discrete logarithm problem, Proc Advances in Cryptology–EUROCRYPT 94, 1994, pp. 428–432.

[5] C.-C. Chang and W.-B. Wu, A secure voting system on a public network, Networks 29 (1997), 81–87.

[6] D. Chaum, Untraceable electronic mail, return address, and digital pseudonyms, Comm ACM 24 (1981), 84–88.

[7] D. Chaum, Blind signatures for untraceable payments, Proc Advances in Cryptology–CRYPTO 82, 1983, pp. 199–203.

[8] D. Chaum, Security without identification: Transaction systems to make the big brother obsolete, Comm ACM 28 (1985), 1030–1044.

[9] D. Chaum, Elections with unconditionally secret ballots and disruption equivalent to breaking RSA, Proc Advances in Cryptology–EUROCRYPT 88, 1988, pp. 177-182.

[10] J.D. Cohen and M.J. Fischer, A robust and verifiable cryptographically secure election scheme, Proc 26th IEEE Ann Symp on Foundation Computer Science, 1985, pp. 372–382.

[11] L.F. Cranor and R.K. Cytron, Sensus: A security-conscious electronic polling system for the Internet, Proc 30th IEEE Hawaii Int Conf on System Sciences, 1997, pp. 561–570.

[12] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans Inf Theory IT-31 (1985), 469–472.

[13] M.K. Franklin and M.K. Reiter, The design and implementation of a secure auction service, IEEE Trans Soft Eng 22 (1996), 302–312.

[14] A. Fujioka, T. Okamoto, and K. Otha, A practical secret voting scheme for large scale elections, Proc of Advances in Cryptology–AUSCRYPT 92, 1992, pp. 244–251.

[15] K.R. Iversen, A cryptographic scheme for computerized general elections, Proc Advances in Cryptology–CRYPTO 91, 1991, pp. 405-419.

[16] J.K. Jan and C.C. Tai, A secure electronic voting protocol with IC cards, J Syst Soft 39 (1997), 93–101.

[17] J. Karro and J. Wang, Towards a practical, secure, and very large scale online election, Proc 15th IEEE Ann Computer Security Applications Conf, 1999, pp. 161–169.

[18] Y. Mu and V. Varadharajan, Anonymous secure e-voting over a network, Proc 14th IEEE Ann Computer Security Applications Conf, 1998, pp. 293–299.

[19] H. Nurmi, A. Salomaa, and L. Santean, Secret ballot elections in computer networks, Comput Sec 10 (1991), 553–560.

[20] W. Ogata, K. Kurosawa, K. Sako, and K. Takatani, Fault tolerant anonymous channel, Proc First Int Conf Information and Communications Security–ICICS ’97, 1987, pp. 440–444.

[21] C. Park, K. Itoh, and K. Kurosawa, Efficient anonymous channel and all/nothing election scheme, Proc Advances in Cryptology–EUROCRYPT ’93, 1993, pp. 248–259.

[22] A. Pfitzmann and M. Waidner, Networks without user observability, Comput Sec 6 (1987), 158–166.

[23] R.L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public key cryptosystems, Comm ACM 21 (1978), 120–126.

[24] B. Schneier, Applied cryptography—Protocols, algorithms, and source code in C, Wiley, New York, 1994.