Design and Prototype of a Coercion-Resistant, Voter Verifiable Electronic Voting System

Anna M. Shubina
Department of Computer Science
Dartmouth College
Hanover, NH 03755
E-mail: ashubina@cs.dartmouth.edu

Sean W. Smith
Department of Computer Science
Dartmouth College
Hanover, NH 03755
E-mail: sws@cs.dartmouth.edu

***Abstract*— In elections, it is important that voters be able to verify that the tally reflects the sum of the votes that were actually cast, as they were intended to be cast. It is also important that voters not be subject to coercion from adversaries. Currently most proposed voting systems fall short: they either do not provide both properties, or require the voter to be a computer.**

In this paper, we present a new voting system that uses voter knowledge to allow voter verification by using a receipt that is uninformative for a coercer without access to the voting machine or the contents of the cast ballots. Our system does not assume any trust in the voting machine, but requires a few other assumptions which we believe to be reasonable in the real-world situation. A basic prototype of this system is available on our website.

I. Introduction

The US presidential elections of 2000 made the general public aware of the problems of producing a voting system that could be trusted by the voters to submit their votes correctly. Despite the public review and control of the US electoral system, many US citizens felt that the system had failed them. Although the problems did not originate in the 2000 election, the situation where a very small number of votes was sufficient to flip the final tally raised the public's awareness of the inadequacy of the system.

The problem of producing a fair voting system has been well-known in countries and situations where adversaries have a very high degree of control. In totalitarian societies (or other situations with almost complete adversarial control), it may be futile to attempt to solve this problem. Such societies provide no guarantee that the adversary will comply with the solution, no guarantee that the observers and complainants will be able to speak up, and no guarantee that the situation will be corrected even if there is a valid complaint. However, in a free democratic society in the 21st century, the electoral system is subject to public review and control. Its failures do not have to be possible.

If an electronic voting system is to be applied in secret-ballot elections, it has to be *receipt-free*, i.e. not allow a voter to carry away any evidence of who he voted for, since such evidence would permit vote buying and coercion. Receipt-freeness is hard to combine with *voter verifiability*: if a voter is able to verify that his vote was counted as he cast it, what could prevent him from proving how he voted to a third party?

Cryptography can help address this seeming incompatibility between receipt-freeness and voter verifiability, if the voter has an encrypted copy of his vote and can verify that his encrypted vote made it to the final tally. However, that requires the voter to be able to verify that the encryption of his vote is correct. Chaum's layered receipts [5] (discussed below) solve this problem partially: they allow verification, but only with probability 50%.

In this paper, we examine these properties and survey the principal current approaches to electronic voting systems. We then present a new design that improves on the previous work, by being (arguably) the first one that achieves both voter verifiability and coercion resistance, while not assuming the voter is a computer, not relying on correct behavior by the voting machine, and detecting close to 100% of misbehaviour.

Section II briefly discusses the requirements for a secure election system. Section III presents a brief overview of methods used in receipt-free election schemes. Section IV discusses the most recently implemented election schemes. Section V presents our election scheme. Section VI discusses the practical applicability of our scheme. Section VII describes our prototype. Section VIII offers some concluding remarks.

II. Requirements for an Election System

The two basic properties commonly required of election systems are *correctness* and *privacy*. An election system should produce a *correct tally*—the result of the election. It should also ensure that the participant's vote will remain *private* (infeasible to find out without cooperation with the voter)—although sacrificing the privacy requirement permits making the system much simpler.

The more interesting properties extensively studied in theoretical literature are *receipt-freeness* and *coercion resistance* (as described in Section II-C) and *verifiability* (as described in Section II-D).

Would ensuring correctness, privacy, receipt-freeness, and verifiability be sufficient for a real-world election system? Arguably, almost all problems in existing real-world election systems stem from the lack of one of these properties. Less discussed, and not ensured by any of these properties, is another important property: *trust*.

A. Correctness

The very first requirements of every voting scheme are that every voter should be able to vote, but only vote once; only votes cast by registered voters should be included in the final tally; and all votes cast by registered voters should be correctly counted.

Defining a *correct tally* is somewhat harder, because it is unclear how to handle incorrectly cast votes. In the definitions of Benaloh and Tuinstra [3], a tally is *correct* if correctly cast votes representing a valid choice are counted, whereas incorrectly cast votes that do not represent a valid choice may be counted one way or another. It may be worth noting that in a real-world situation it may be unacceptable to include incorrectly cast votes into the tally, and thus the ability to distinguish incorrectly cast votes may be important. Hirt and Sako [6] get around this problem by requiring for correctness that no voter be able to cast an invalid vote.

B. Privacy

Privacy, in the context of elections, means ensuring that nobody except the voter can find what choices the voter made without interacting with the voter. More precisely, an adversary should not obtain more information about a voter's vote than provided by the election tally.

Privacy of a voting system is dependent on assumptions as to what the adversary can or cannot do and is often based on assertions about the adversary's computational power.

One of the possible physical assumptions made for privacy is a *voting booth* that allows the voter to secretly and interactively communicate with an authority. A weaker assumption is an *untappable channel* that allows a voter to send a message that cannot be observed by the outsiders. Whereas voting booths are used for voting in the real world, they are typically used only for communication with the voting machine and do not allow remote communication.

C. Receipt-freeness and Coercion Resistance

The initial papers on secret-ballot elections considered the property of *receipt-freeness*. As introduced by Benaloh and Tuinstra [3], receipt-freeness is the inability of a voter to prove to an adversary how he voted, even if the voter would like to provide this proof. Receipt-freeness is necessary in secret ballot elections. Indeed, if the voter had the ability to prove to an adversary the contents of his vote, the adversary would be able to demand from the voter that he vote in a particular manner and reward him for voting in this manner or penalize him for not complying with the demand.

More recently, Juels and Jakobsson [7] introduced a stronger notion of *coercion resistance*. A coercion-resistant system is a system where the voter can cheat an adversary who may interact with him and instruct him to vote in a given manner, but the adversary will not be able to determine whether the voter behaved as instructed—even if the adversary asks the voter to disclose his keys or to abstain from voting.

D. Verifiability

In the ideal voting scheme, the voter should be able to verify that his vote was committed as intended and made it into the final tally as cast (*individual verifiability*, or *voter verifiability*), and any observer should be able to verify the tally (*universal verifiability*).

These two properties provide *verifiability*—the possibility of verification that all votes are counted correctly.

E. Trust

The question of trust in a voting system is one of the most discussed and least agreed upon. Who should trust whom for what? Is it enough for an expert to trust another expert's assertion that the system functions correctly?

The range of opinions on what would constitute a trustworthy electronic voting system is very wide, ranging from Rebecca Mercuri's *Statement on Electronic Voting* [8] that demands the use of "an indisputable paper ballot" and the Mercuri Method [9], to Andrew Neff's [10] and David Chaum's [5] reliance on verification. There is, as yet, no agreement on whether it would be sufficient to have a system that the experts can prove sufficiently untamperable.

III. Basic Cryptographic Schemes

The literature provides three basic cryptographic approaches for secure electronic voting.

A. Voting Schemes Based on Homomorphic Encryption

Homomorphic encryption is encryption over an algebraic group such that the encryption of the sum of two elements of the group is the sum of the encryptions of these elements. The idea of using homomorphic encryption in electronic voting is to sum the encrypted votes, and then decrypt the sum—without decrypting individual votes.

The first homomorphic encryption voting scheme was proposed in the paper by Benaloh and Tuinstra [3]. This scheme was proven by Hirt and Sako in [6] not to be receipt-free. In the same paper Hirt and Sako proposed another, receipt-free, homomorphic encryption voting scheme.

Homomorphic encryption schemes do not support write-in votes.

B. Voting Schemes Based on Mix-nets

A number of electronic voting schemes are based on Chaum's *mix-nets* [4]. Mix-nets encrypt, permute, and re-encrypt the sequences of input elements, producing permutations of these elements intended to conceal their original order.

In mix-net voting schemes, a vote is encrypted with a sequence of the keys of the authorities and consequently decrypted by the authorities who prove the correctness of the decryption. An example of such a receipt-free voting scheme is the scheme proposed by Sako and Kilian in [13].

Mix-net based schemes can support write-in votes.

C. Voting Schemes Based on Blind Signatures

Blind signatures (also due to Chaum) allow an authority to sign an encrypted message without seeing its contents. In electronic voting schemes, blind signatures are used to allow the administrator to authenticate a voter by signing an encrypted ballot.

An example of a scheme based on blind signatures is the scheme proposed by Okamoto in [11]. Okamoto later showed this scheme not to be receipt-free and fixed it in [12].

Schemes based on blind signatures can support write-in votes.

IV. More Recent Election Schemes

The recent literature has also provided examples of more practical, implemented schemes.

A. David Chaum's Encrypted Receipts

David Chaum's new scheme [5] allows a voter to walk away from the polling place with an encrypted receipt that has previously been shown to him to correctly contain his vote.

The scheme functions in the following steps. The voter chooses his votes electronically. The voting machine prints out a two-layer image that displays the vote. The user selects which part of the image—the top layer or the bottom layer—he would like to keep, and walks away with it. The voter can later verify that the receipt was correctly posted on the election site by looking it up by the receipt's serial number in the *receipt batch*, the set of receipts the authority intends to count. The *tally batch*, the set of plaintext images of ballots as seen in the voting booth, is also posted in random order, allowing everyone to verify the tally.

On each ballot, the voting machine can cheat with probability 50% by either printing one incorrect layer, or reusing the serial number, or performing a tally process step incorrectly. Thus, if even only 10 modified ballots are verified, the chances that the tampering will go undetected would be less than 1 in 1000.

The scheme does not reply the question of what to do if the election did get tampered with. Since only 50% modified ballots will be detected, there is no opportunity to recast only the modified ballots.

B. VoteHere

In an attempt at providing voter verification and ensuring user trust, the VoteHere [1] [10] system hands a voter a paper receipt.

In the VoteHere scheme, a voter starts with a *voting token* (a smart card or a key). This voting token can be inserted into a machine that lists all candidates and provides a *verification code* for every candidate. The verification codes are different for every voting token. The voter picks the codes corresponding to his choices and gets a receipt listing the ballot number, these codes, and the signature of the ballot produced by the voting machine. Later, the voter can use the ballot number to verify that the codes were recorded correctly; however, he cannot verify that the voting machine did not swap candidates before presenting to him the codes. Instead, the voter should trust that the trustees responsible for the generation of codebooks and auditing of the machines made this impossible.

V. Proposal for a Verifiable Election Scheme

All of the above described voting schemes—both the theoretical ones and the implemented ones—appear to allow receipt-free implementations. However, verifiability (and also correctness, in schemes relying on verifiability for correctness) turns out to be harder to achieve in practice than in theory. VoteHere's scheme depends for its correctness and verifiability on the correct behavior of the voting machine; Chaum's scheme detects voting machine misbehavior in only 50% of votes.

To make a step toward ensuring users' trust, we would like to propose a scheme that would allow a user to verify how he voted, not only that a vote in his name made it to the destination. Our scheme does not make any assumptions about the correct behavior of a voting machine and makes voting machine misbehavior detectable in almost 100% cases, achieving correctness and voter verifiability at the cost of one extra assumption.

This goal is hard to achieve without sacrificing coercion resistance; indeed, if the voter can verify how he voted, then what prevents him from proving to someone else how he voted? We will attempt to solve this problem by taking into account voter knowledge—that is, information known to the voter but not accessible to the coercer.

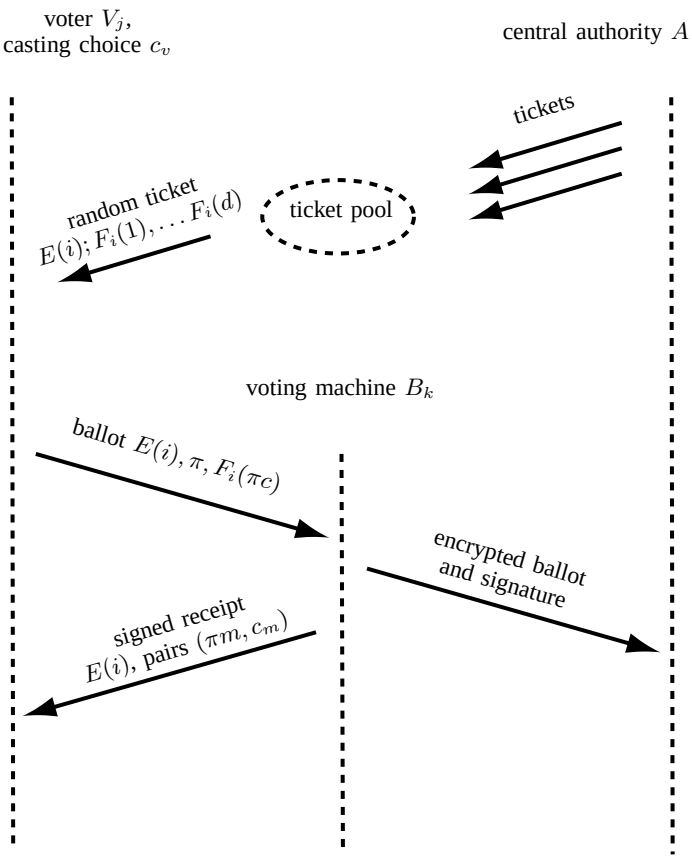
In our scheme, we make an attempt to achieve voter trust by permitting a voter to verify his vote by using a printed receipt to query the central authority for the record on a certain key. However, in order to avoid coercion, the voter should possess also the keys corresponding to his other possible votes and should be aware of this correspondence, perhaps by having it recorded on the same printed receipt. Also, in order for this idea to work, the voter's choice should be specified by the voter in such a manner that no malicious authority would be able to tamper with it without detection.

Our scheme provides the voter with pre-generated keys and allows him to permute them, matching the keys with the possible votes. This permutation ensures that a malicious authority cannot on its own generate a desired ballot. The permutation is also used to provide a visually satisfying receipt and verification routine. The voter's choice is specified by submitting the key corresponding to the desired candidate. Finally, a signature of the ballot is generated by the voting machine. This signature is printed on the voter's receipt and publicly posted. The signature can be used by the trusted observers to check that the recorded ballot matches it.

A. Assumptions

In our scenario, we are trying to imitate the real-world model of electronic voting. We assume that the world consists of:

Fig. 1. The voting process.



- the central authority A (in the real world, the central election committee);
- local authorities $B_1, \ldots, B_m$ (in the real world, the voting machines);
- n voters $V_1, \ldots, V_n$;
- only a finite number d of possible choices $c_1, \ldots, c_d$ for all votes.

(One of these possible choices should be "no selection", to allow the voter to abstain from voting. A few choices may be dummy votes.)

We assume no communication by the central authority with the voters, but we assume the existence of voting booths—that is, interactive communication of voters with the voting machines. This latter assumption does not appear to be unreasonable; voting booths are frequently used in the real world for ensuring voters' privacy.

Our scheme involves tickets generated by the central authority A. We require another assumption: that no information leaks from the initial process of generating voting tickets for voters leaks to local authorities. One method of ensuring this assumption could be allowing having different authorities participate in generating keys on cards, so that no authority would have the full information.

B. Process

The steps necessary to cast a vote are illustrated in Figure 1.

- The central authority A (or a set of authorities) uses its public-key encryption function E to encrypt some numbers. (This encryption function should not allow an adversary to guess the plaintext and then verify this guess by encrypting it. It would suffice, for example, to use RSA encryption with OAEP ([2]) padding.) The authority uses E to encrypt numbers $1, \ldots, n$ to serve as ticket identifiers. (We use n, so there will be a distinct identifier for each voter.) The authority also encrypts numbers $1, \ldots, dn$ to serve as keys printed on tickets. (We use dn, so there will be a distinct identifier for each choice, for each ticket.)
  For each ticket i, let us define $F_i(k) = E(di+k)$ (that is, the kth key on the ith ticket). The authority stores both the ticket identifiers and the keys.
- The central authority A then prints out n tickets (but does not publicize their contents). Ticket i consists of the ticket identifier $E(i)$ and keys $F_i(1), \ldots, F_i(d)$. In the real world the authority could either produce a number of sealed envelopes containing these tickets, or make them obtainable from the official election website only.
- Each voter selects a random ticket (for example, by pulling them out of a box or by getting them from the election website).
- Suppose voter V_j selected ticket i and wishes to vote for choice c_v (for some v with $1 \leq v \leq d$). Then V_j casts to his local authority B_k a ballot consisting of
  - ticket identifier $E(i)$;
  - a random permutation π of $1, \ldots, d$;
  - $F_i(\pi v)$. (Recall that v is the index associated with voter's choice c_v.)
- The local authority B_k prints out a receipt including ticket number i, all pairs $(\pi m, c_m)$ (ordered by πm) and the signature of the ballot, as shown in Table I. B_k hands the receipt to the voter.
- B_k encrypts the ballot and signature with the public key of the central authority and submits this to the central authority A.
- When the central authority A receives this ballot, it checks that it is properly formatted—and that the key (allegedly $F_i(\pi v)$) really does decrypt to the index of a valid choice c_v. The authority A also validates that the signature provided is valid for this ballot. If both these conditions are satisfied, the authority A records this ballot as a vote for c_v.

Within the ballot, the only information about the voter's selection is $F_i(\pi v)$, the key corresponding to the permuted index of the voter's real vote. The central authority A can decode it and find πv, thus finding the real vote (unless corrupted by the local authority). However, by the assumptions of cryptography, the local authority cannot extract πv from the key, and thus cannot find out who the voter voted for (unless the central authority A cheats).

TABLE I
AN EXAMPLE VOTER RECEIPT

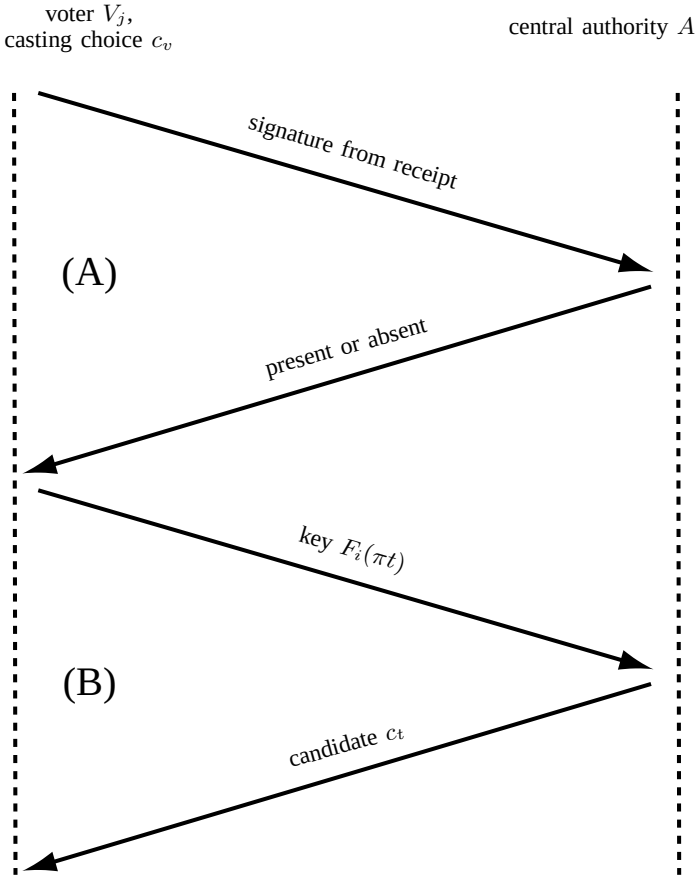
Ballot ID	3708DD567880145B
Index	Vote
1	
2	no selection
3	Bob
4	
5	
6	
7	
8	Alice
9	
10	
Signature	0DA7E8339A56730024C

Note also that the voter does not submit keys $F_i(t)$ for $t \neq v$. Thus if the central authority did not share information with the local authority, the local authority cannot know what these keys are.

If the local authority does not know the other keys on the ticket, it cannot manufacture a ticket that features the votes in the same order but with an incorrect vote matched to a real key. The local authority can attempt to record a different vote under a correct key or corrupt the correct key, but this attempt will later be caught by the verification process.

After the election is done, the central authority A publishes the receipt signatures, thus allowing any voter to verify that his ballot made it to the final count. For additional assurance, the authority A can provide a post-election query service: a voter can submit any key $F_i(\pi t)$ (and the ticket identifier $E(i)$), and check that he gets back c_t as shown. Figure 2 shows this process. If he sees that his vote did not make it as intended, he can request resubmission.

Fig. 2. The verification process, for voter V_j with ticket $E(i)$. In (A), a voter checks that his ballot made it; in (B), a voter checks that the ballot that made it in matches his ticket.



C. Adversarial Model

Our scheme will protect against all of the following:

- loss of votes,
- casting of an incorrect vote by a voting machine,
- coercion by an adversary that does not have access to the voting machine or the central authority,
- tampering with the tally by an adversary that did not have access to the voting machines during the election.

D. Properties

The ability to look a vote up by the corresponding key allows voter verification. Indeed, suppose looking up all keys on the ticket matches the votes submitted by the voter under the corresponding indices. This could only be achieved in three ways.

1) The real vote is submitted correctly.
2) The local authority submitted a fake vote as a real vote. However, this would require that the local authority know a key different than the one the voter submitted. Since the local authority does not know any of the other keys printed on the voter's ticket, it cannot swap the real and the fake vote provided by the voter without being detected.
3) The central authority considers a fake vote to be a real vote. However, the signatures are published and can be checked by voters, and trusted parties could verify that the decryptions of the encrypted votes produce signatures matching these records.

Therefore, if the local authority does not have any prior knowledge of the keys, neither the local nor the central authority can tamper with the voter's choice without detection. Thus the voter can verify whether his vote is submitted correctly.

As mentioned above, the central authority cannot lie about the contents of the ballots it got, because signatures will be posted and can be checked by voters, whereas the fact that the decryptions got correctly counted may be verified by trusted parties.

If an external adversary who does not have access to the voting machine or to the central authority's data instructs a voter to behave in a certain manner, the only evidence of the voter's behavior will be his receipt (unless, as mentioned above, the ballots are disclosed). However, even if the adversary's choices show on the receipt as requested, any of them can be the voter's real choice.

E. Vulnerabilities

Our scheme is vulnerable in the following respects.

- If the central authority shares the keys with the local authorities before the election, this would allow the local authorities to corrupt ballots without detection.
- If ballots are disclosed, the property of coercion resistance would be lost.
- A voter can claim that his vote was recorded incorrectly when in fact it was not.
- The central authority can check ballots and post signatures in accordance with the rules, but lie about the tally. Only trusted observers will be able to verify the tally.

VI. Practical Applicability

To cast one vote, the voters will have to submit two long numbers (the ticket identifier and the encrypted index of the real vote) and some short numbers. To avoid this in practice, a ticket could contain subtickets that can be scanned by the machine, containing these long numbers. Alternatively, the vote could be cast electronically (as in the prototype described in the next section).

To simplify the voter's interaction with the voting machine, the permutation of votes could be randomly generated by the machine for the voter.

The second problem is that if the user declares that an incorrect vote was submitted, there is no way to tell whether he is lying or whether the voting machine really submitted the incorrect vote.

Another usability problem is due to the fact that the voter should submit only one key. If the voter changes his mind after submitting this key, he will not be able to change his choice without either leaking information or drawing another ticket.

Finally, in a real election more than one vote usually has to be cast, and more than one ticket will have to be used.

VII. Prototype

We created a prototype for this scheme. The prototype imitates user interaction with the remote authority and with the local voting machine. It is currently accessible at `althing.dartmouth.edu/cgi-bin/electme2/master.pl`.

The interaction proceeds as follows.

- First, the user opens two browser processes: one simulating the remote authority, another simulating the local voting machine.
- The user requests initialization of the election from the remote authority. The remote authority generates keys and tickets.
- The user selects a random ticket. The remote authority marks the ticket as taken.
- The user submits the ticket ID to the local voting machine. (See Figure 3.)
- The local voting machine prompts the user for a permutation of the indices on the ticket corresponding to the placement of the votes ("customization of the ballot"). (See Figure 4.)
- The local voting machine then lets the user vote. (See Figure 5.)
- The local voting machine generates the receipt consisting of the ticket ID and the indices, and of the signature (the encryption of the hash of the message with the public key of the central authority).
- The local voting machine submits the encryption of the ballot and the signature to the central authority.
- The user can go to the website of the central authority and enter the keys listed under the indices submitted. If the keys have his receipt's vote choices listed under them, he either gets the vote listed back (e.g., Figure 6, or knows that his vote did not get submitted correctly (e.g., Figure 7).

VIII. Conclusions and Future Work

We believe that in real-world situations, our scheme may work to allow voter verification without making the voter susceptible to coercion more than he already is. (For example, in the real-world situation the voter may be coerced not to go to the polling place.) The scheme achieves these results by allowing the use of voter knowledge that cannot be used by a hypothetic coercer.

Our scheme takes the possible malfunctioning of or malicious interference with voting machines and loss of votes out of consideration, by replacing it with later verification. Whereas the correctness of VoteHere's scheme depends on the initial step of generation of codebooks and on verification that all voting machines function as intended (which may not be a trivial task), the correctness of our scheme depends only on the secrecy of data distributed before the election. Every voter's vote can be modified only with a small probability, ensuring that if the tampering is detected the votes can be cast again.

Our scheme hands out a readable receipt that allows the voter to see that his vote got cast as intended if the initial conditions of secrecy were met. We believe that this may help with achieving voter trust in the system.

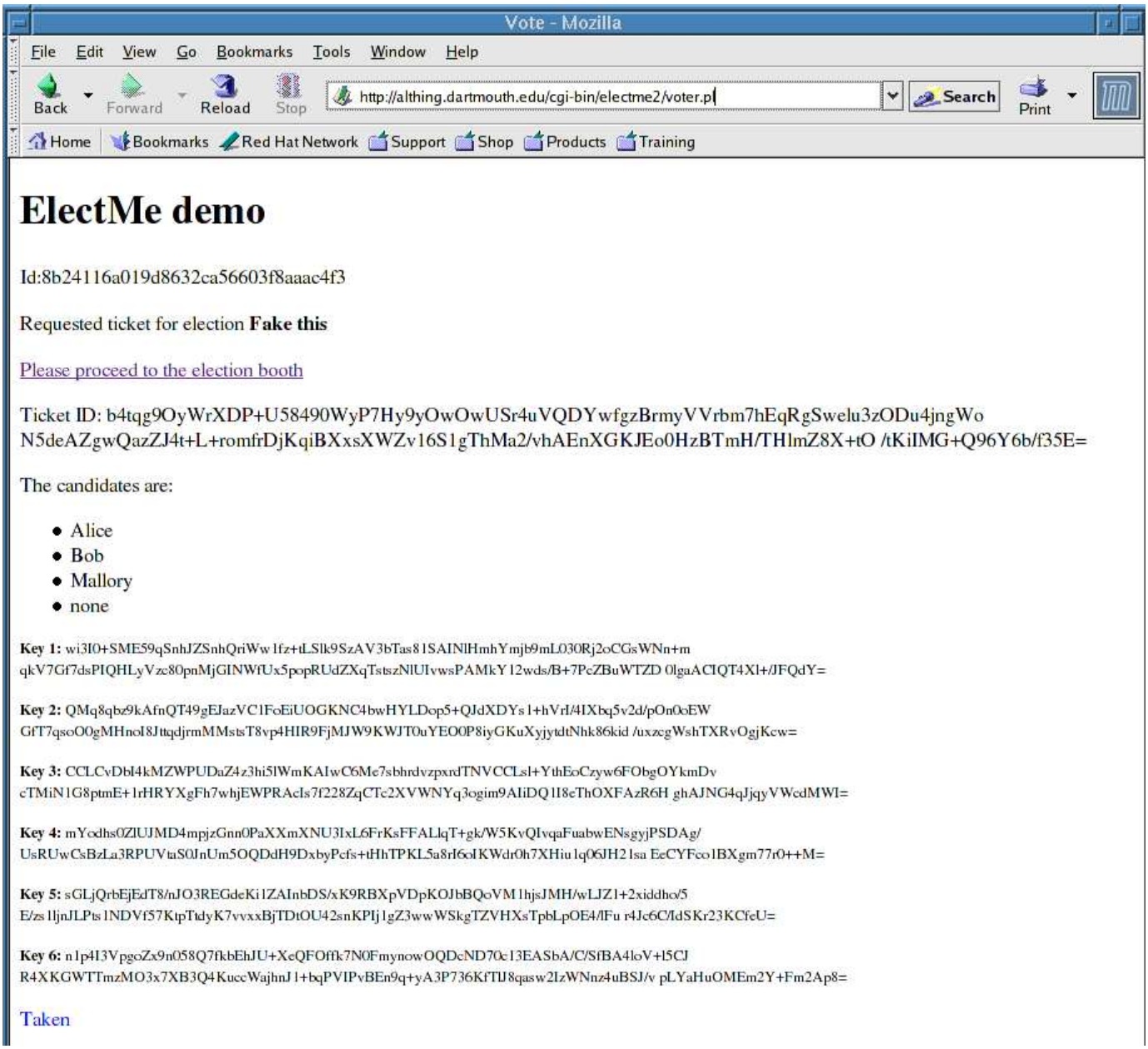
Our scheme is vulnerable if the central authority cheats before the election by cooperating with the local authorities, and if a sufficient audit process is not feasible to verify the tally after the election.

Our scheme does not reply to the question of whether the user is telling the truth if he claims his vote got recorded incorrectly. This is a problem we would like to try to solve in the future.

Our scheme is also vulnerable to hypothetic post-election disclosures of ballots. It may be possible to address this issue by providing an extra layer of encryption between the user and the voting machine, at the cost to usability. However, we do not believe that this is likely to be a problem in a situation without a strong adversary.

In future work, we plan to address these shortcomings, as well as carry out pilots with real users, and examine the usability of the various ways of a human user might communicate the keys to the voting machine.

Fig. 3. The user submits the ticket to the voting machine.



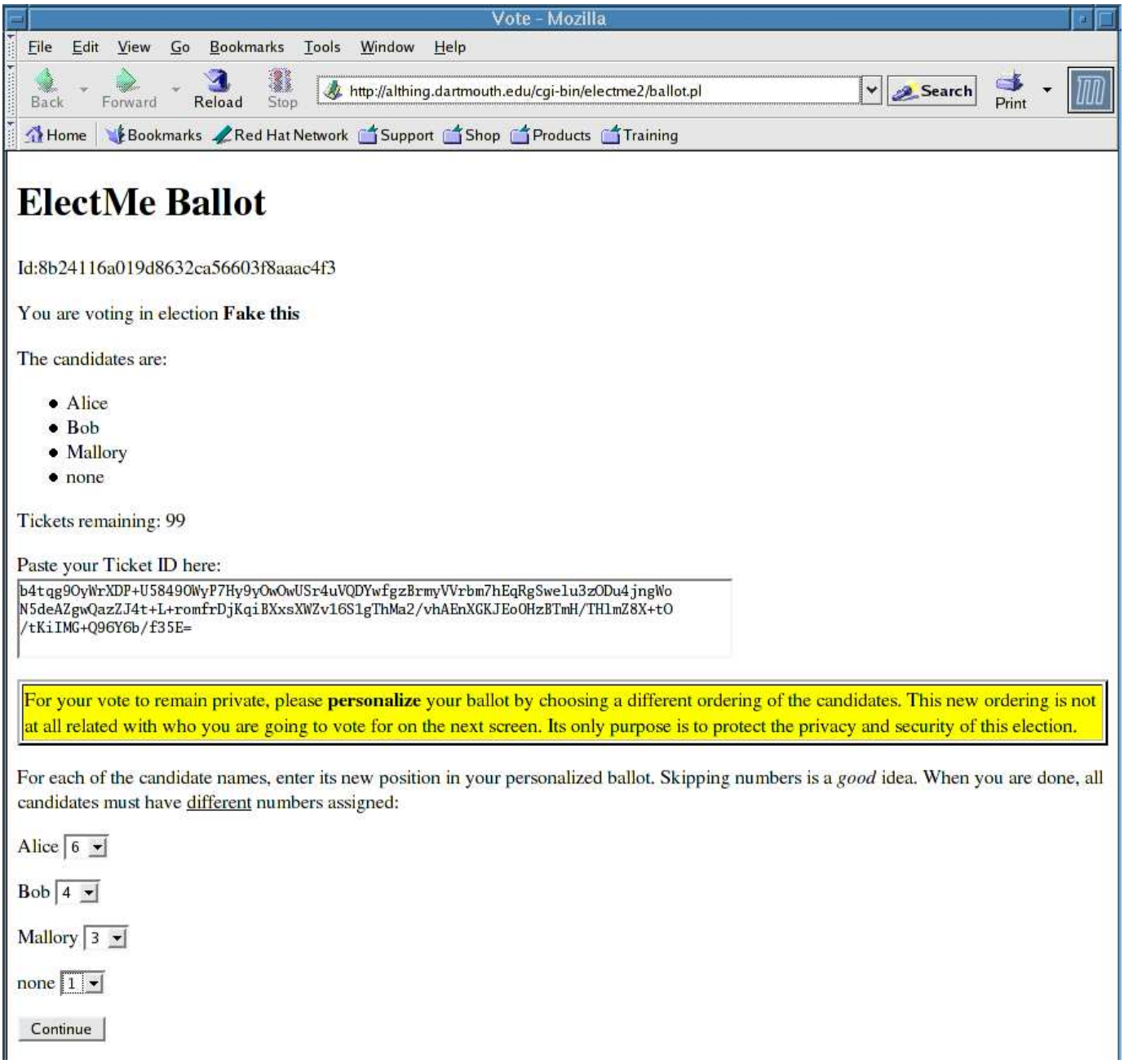
ACKNOWLEDGMENTS

This work was supported in part by the Mellon Foundation, by the NSF (CCR-0209144), by Internet2/AT&T, and by the Office for Domestic Preparedness, U.S. Dept of Homeland Security (2000-DT-CX-K001). The views and conclusions do not necessarily represent those of the sponsors.

REFERENCES

[1] "VoteHere," www.votehere.net.
[2] M. Bellare and P.Rogaway, "Optimal asymmetric encryption/how to encrypt with RSA," in *Advances in Cryptology—Eurocrypt '94*, vol. 950, 1994, pp. 92–111.
[3] J. Benaloh and D. Tuinstra, "Receipt-free secret-ballot elections," in *Proc. of 26th Symp. on Theory of Computing (STOC'94)*, New York, 1994, pp. 544–553.
[4] D. Chaum, "Untraceable electronic mail, return addresses and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–88, February 1981. [Online]. Available: http://www.eskimo.com/~weidai/mix-net.txt
[5] ——, "Secret-ballot receipts: True voter-verifi able elections," *IEEE Security & Privacy*, vol. 2, no. 1, pp. 38–47, January/February 2004.
[6] M. Hirt and K. Sako, "Effi cient receipt-free voting based on homomorphic encryption," *Lecture Notes in Computer Science*, vol. 1807, pp. 539–??, 2000. [Online]. Available: citeseer.ist.psu.edu/article/hirt00effi cient.html
[7] A. Juels and M. Jakobsson, "Coercion-resistant electronic elections." [Online]. Available: citeseer.ist.psu.edu/555869.html
[8] R. Mercuri, "Rebecca Mercuri's statement on electronic voting," http://www.notablesoftware.com/RMstatement.html, 2001.

Fig. 4. The voting machine responds with a prompt for the permutation.



[9] ——, "A better ballot box?" *IEEE Spectrum Online*, vol. 39, 10, October 2002.
[10] A. Neff and J. Adler, "Verifiable e-Voting," www.votehere.net/vhti/documentation/verifiable_e-voting.pdf, 2003.
[11] T. Okamoto, "An electronic voting scheme," in *IFIP'96, Advanced IT Tools*. Chapman & Hall, 1996, pp. 21–30.
[12] ——, "Receipt-free electronic voting schemes for large scale elections," *Security Protocols Workshop*, pp. 25–35, 1997.
[13] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme—a practical solution to the implementation of a voting booth." in *Advances in Cryptology—Eurocrypt '95*, vol. 921. Berlin: Springer-Verlag, 1995, pp. 393–403.

Fig. 5. The voter casts his vote.

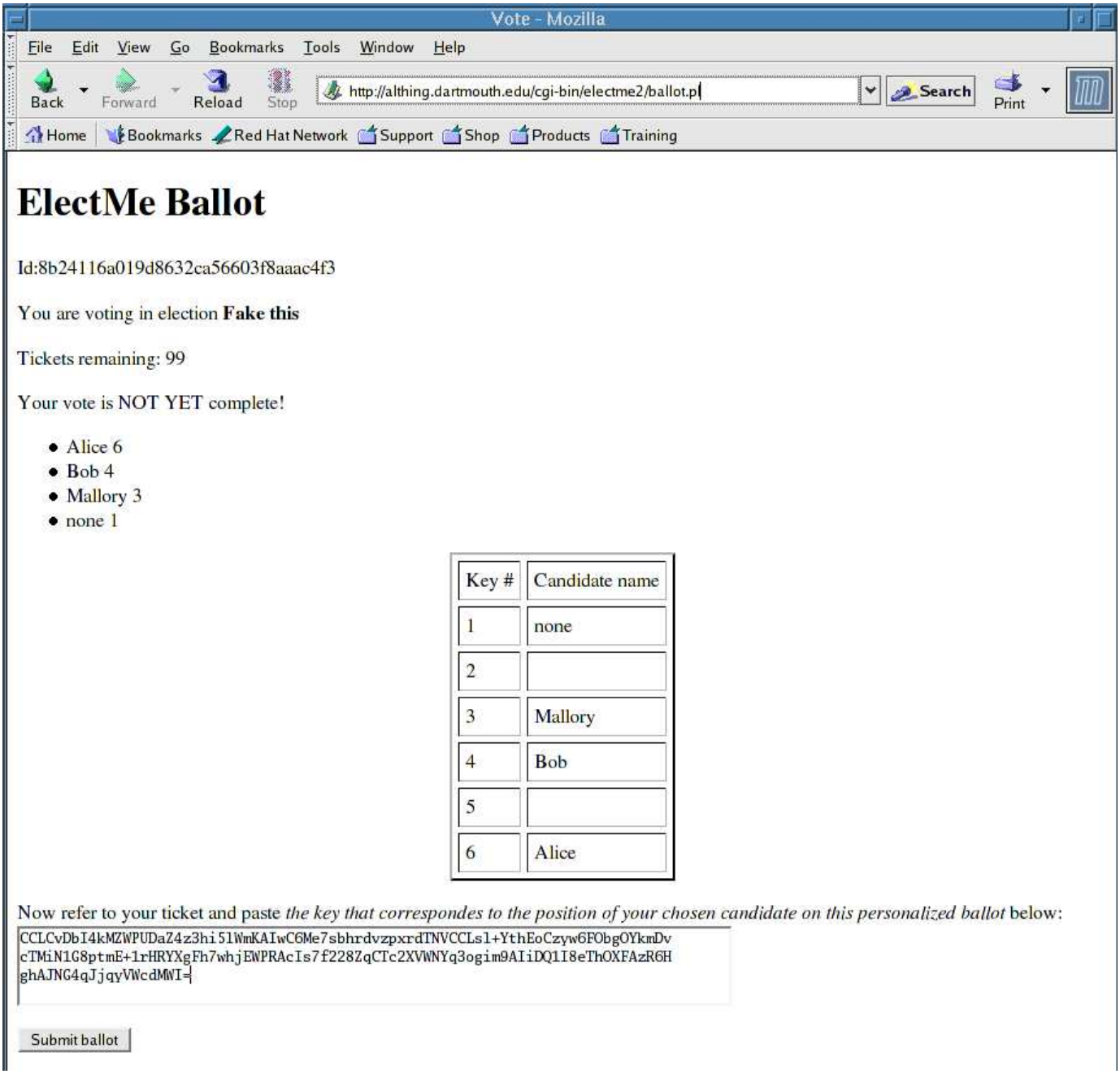


Fig. 6. The voter checks that a ballot matching his ticket has been cast.

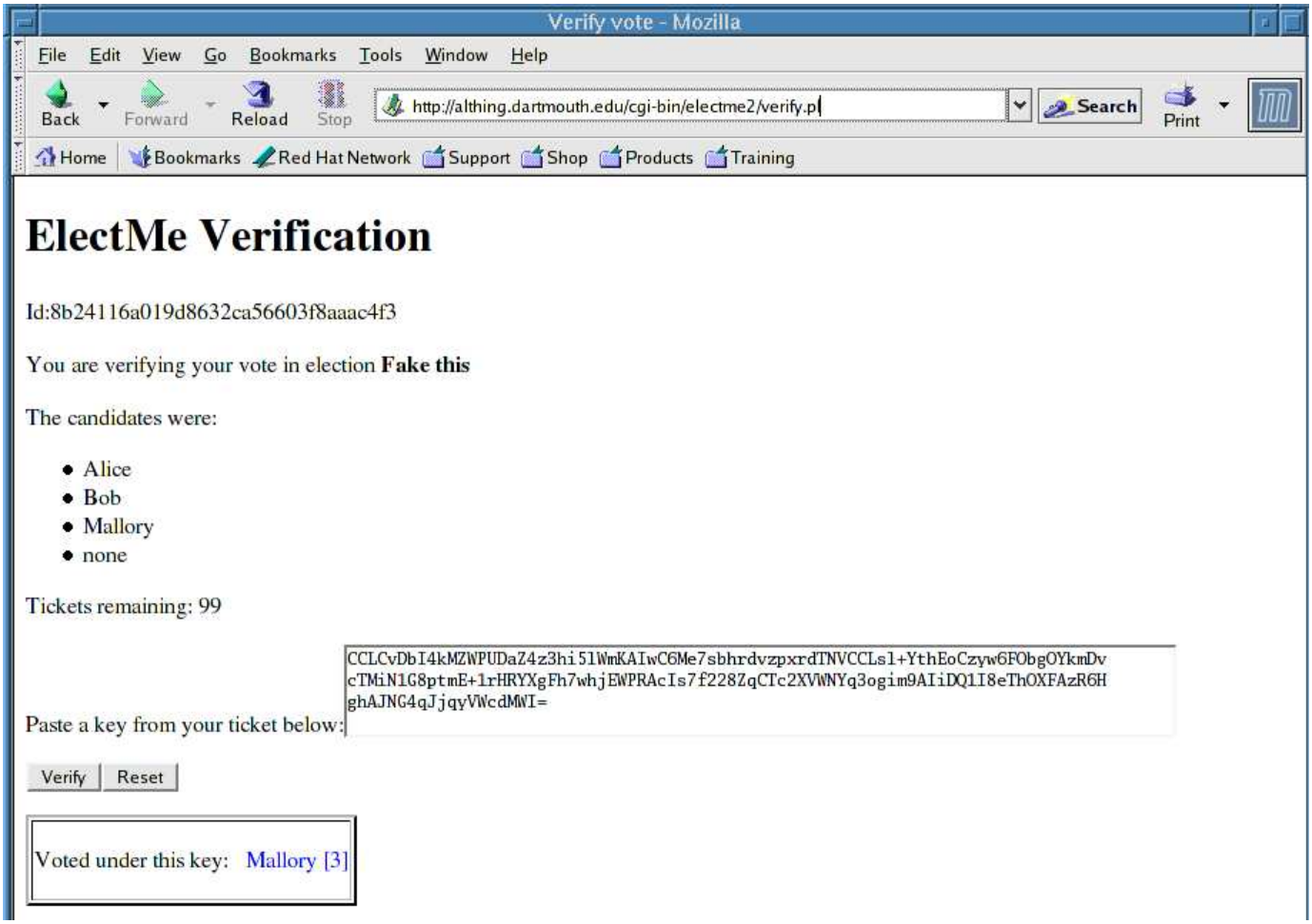


Fig. 7. The voter submits an invalid key.