Mental Voting Booths

Jérôme Dossogne[1] and Frédéric Lafitte[2]

[1] Université Libre de Bruxelles, Department of Computer Science,
Boulevard du Triomphe - CP212, 1050 Brussels, Belgium
jdossogn@ulb.ac.be
[2] Royal Military Academy, Department of Mathematics,
Renaissancelaan 30, 1000 Brussels, Belgium
frederic.lafitte@rma.ac.be

Abstract. In this paper, we introduce the notion of mental voting booths, i.e., a building block for voting schemes that provides remote voters with similar protection as that offered by physical voting booths, essentially protecting them from over-the-shoulder coercion attacks (shoulder-surfing). We introduce a framework to model voting booths and formulate a property of the modelled booths that is sufficient to ensure over-the-shoulder coercion resistance. Next, we propose an example of mental booth that is simple enough to be used by any voter without prior training and show that an execution of the remote booth in the presence of the adversary is equivalent to that execution in his absence (e.g., inside a physical booth). The only cost lies in the use of an untappable channel in order to transmit a piece of information before the voting phase. Mental booths also allow for the voter to safely delegate his own voice to an untrusted person while still being able to verify that the untrusted person followed his instructions while voting.

Keywords: remote voting, i-voting, e-voting, home-voting, shoulder surfing, over-the-shoulder, coercion resistance, voting booth.

1 Introduction

Electronic voting (e-voting) is a growing trend [34] and a growing concern. A key issue slowing the adoption of such technologies, in particular remote e-voting [32,15](e.g. Internet voting [20,24]) is trust. The security of the different proposed schemes often relies on cryptographic primitives and protocols which are not easily understood by the majority of the designated users. This lack of understanding and errors occurring in the implementation of proven protocols can lead to a growing mistrust. However, these are not the only factors slowing the deployment of e-voting. Indeed, factors like the lack of physical voting booths protecting the user from coercion[1] while he votes is also a recurring argument against remote e-voting [20]. The fact that electoral authorities do not have control over all the equipment used by voters is perhaps the main challenge faced

[1] Nowadays, a physical voting booth does not prevent a voter from using his cellphone to take a picture of his vote in order to sell it afterwards.

P. Laud (Ed.): NordSec 2011, LNCS 7161, pp. 82–97, 2012.

by remote e-voting schemes. In this paper, we present a technique allowing for a voter to be protected from coercion [22,14] by creating what we call a mental voting booth which compensates for the lack of physical voting booths in the context of remote e-voting. The idea is to create a voting interface for the user such that no attacker can distinguish between a vote for candidate 1 from a vote for candidate 2 by observing the voter's interactions with the interface (over-the-shoulder attacks [11], also known as shoulder surfing [33]) or by operating the computer system on behalf of the voter.

Separation of duty. In any scheme where a voter is associated to an anonymous identifier, and the votes are encrypted, it is obvious that an authority who owns both the decryption key and the identity of the voter has the possibility of coercing the voter. Therefore, it is imperative to use the separation-of-duty concept in order to distribute the different responsibilities between different authorities and to limit the required communications between them as much as possible. This would force the authorities to collude in order to coerce the voter. A rapid separation-of-duty could be the following:

1. Key generation office: generates the secret keys and anonymous identifiers for the voters and the counting office. Transmits the keys and the associated identifier to each party.
2. Polling office: gathers all the votes and corresponding identifiers, transmits them after the election to the counting office.
3. Counting office: establishes the result of the tally using the data sent by the polling office and from the key generation office.
4. The voter(s): uses his identifier and his key given by the key generation office to vote.

With this separation of duties, neither the key generation office nor the polling office can learn for which candidate a particular user voted for (unless authorities collude). As mentioned earlier, the purpose of this paper is to present techniques allowing a better protection against shoulder surfing that could be reused in other voting scheme lacking this property.

Mental booths. In the following, we restrict security analysis to the security of the voting booth. We define a voting booth simply as an interface that offers limited actions to the voter, each action generating a feedback. Our goal is to show that remote voting can be made as secure as voting from a physical booth. This assertion is formally established using behavioural equivalence between two executions: a honest execution of the interface inside a physical booth, and an adversarial execution of the same interface from a remote location. A voting booth that satisfies this requirement offers over-the-shoulder coercion resistance against an adversary that monitors executions of the interface during the entire voting procedure: honest executions are indistinguishable from executions that pretend to be honest. The technique and security analysis are rather simple and easy to understand.

Related work. None of the following schemes JCJ/Civitas [12,22], Helios [1,2], protects voters against over-the-shoulder attacks by a visible attacker (a relative or a coercer watching or influencing the voter during the voting phase) or an invisible one (malware such as keyloggers [20]). The recent Selections [11] does provide over-the-shoulder coercion-resistance against a visible attacker by establishing panic passwords between the voter and the authority once with an untappable channel. In our case, we aim to protect the user also against an invisible attacker (e.g. malicious code) and currently require the same use of an untappable channel. Grünauer[2] provides a scheme stated as keylogger resistant and based on TAN (transaction numbers) which requires that the users memorizes a number associated to each choice. Their solution, as the paper indicates, is acceptable only for small organizations where the number of voters and the number of candidates is small. Compared to the approaches mentioned above, our solution is scalable, protects against a stronger adversary, and requires less effort from the voter. Our proposal could reminisce of independently developed systems such as CodeVoting (see SureVote [9,10,21]), maybe of Bingo Voting [7] or more exactly as an evolution of a combination of both systems. The following paragraphs describe those techniques.

SureVote. SureVote is based on the idea of supplying the voter with a list of "sure codes" and "vote codes" per candidate in a polling place, then the voter uses the vote code associated to the candidate of his choice to vote and receives back the associated sure code. Therefore, the voter is ensured that his ballot has been correctly lodged, regardless of any actions performed by any intermediary between voter and authority. In other words, this system ensures only the voter's ability to detect modifications made to the ballot he sent and does not protect him from coercion against an over-the-shoulder adversary: the attacker could very well request the printed list and observe him while he votes. If the list is never printed, then the voter has to remember two random values per candidate, which prevents the scheme from being scalable.

CodeVoting. CodeVoting is introduced in [21] as an extension of SureVote that offers to distribute the codes via a physical "code card" (which is common for certain netbanking services [17]) and to use a smartcard reader in order to translate the codes on the code card into vote codes. However, their system suffers several drawbacks. Since a code card is involved, their system does not provide any protection against shoulder surfing (the attacker might read the card). Moreover, their system requires an infrastructure for the management of CodeCards, VoteCard (the smartcard), and also a smartcard reader per voter and a certified ad hoc smartcard reader-printer (that could be shared in a public place). The authors state that the trust in the machine is moved to the smartcard / smartcard-reader. A smartcard could very well be easier to check than a computer, but would still require a very high level of expertise for the average voter and would force him to trust experts. Another drawback is the possibility of a successful "mistrust attack" regardless of the countermeasure proposed in

[2] `http://easyvote-app.sourceforge.net`

[21]. Indeed, the authors argue that in order to create mistrust and confusion, malware could make the user believe that the procedure failed (while it did not) which would lead the voter to retry. The server would then refuse the new vote since a vote was already received, thus damaging the trust placed in the voting system. The authors then state that if the system allows the voter to cast several votes this attack would not be a problem. However, firstly nothing prevents the malware to continue lying to the voter and to state that the procedure failed, and secondly the malware could very well simulate the behavior of a voting system where the voter can only vote once. Since such an attack aims for trust, a voter would have either to believe that the voting system works fine and that he is under attack or that the system fails to behave correctly. Since both situations are possible, this ambiguity is already a successful attack on trust. Basically, our approach can also be seen as a code sheet, but unlike [21], has the following properties: it is scalable, does not require dedicated hardware nor user training, and most importantly allows for creating an over-the-shoulder coercion free voting environment.

Bingo Voting. Our proposal also appears to have common grounds with Bingo Voting, a voting scheme where the user receives one dummy random number per candidates and later, at the moment he votes in a physical voting booth, an additional effective random number. The voter then associates all the dummy numbers to the candidates, except for the candidate of his choice. For that candidate, the effective number is used, which is distinct from all the dummy numbers. The voter then leaves the voting booth with a receipt for his vote, free from coercion from an attacker since the latter cannot know which of the numbers is the effective one. Then, the list of all unused dummy values is published along with zero knowledge correctness proofs used during the protocol. To summarize the common grounds, both approaches rely on the use of cryptographically secure random number generation, both have an available implementation in Java, and both associate a number to each candidate in each ballot. On the other hand, Bingo Voting was created for local e-voting protocols while our proposal is designed for remote e-voting (and thus also works in the local case), therefore, Bingo Voting did not have the need and thus does not protect against over-the-shoulder attacks. The correctness of the proposal in [7] is ensured only if each voter verifies a cryptographic proof in order to dismiss fake ballots. The biggest difference is that in order to provide coercion protection, the scheme relies on the voting machine. It must not be tampered with and must guarantee the secrecy of votes. Likewise the voting booth has to be secured, e.g. no hidden cameras may be able to monitor the voting while our technique is designed on purpose to prevent such requirement, considered as an unrealistic hypothesis. In our case, the voting machines sanity is not important to protect the voter from coercion. Furthermore, due to the lack of such a requirement our scheme is immune to all the attacks allowing coercion on the voter due to a tampered voting machine or booth described as effective against their scheme in [7]. Also, Bingo Voting requires additional devices such as a trusted and certified printer.

Contribution. The purpose of this paper is to present a building block used to create a coercion-free voting environment that can be combined to existing electronic voting schemes. The environment is coercion-free even if the coercer is allowed to monitor the entire voting procedure (over-the-shoulder). Protection against this strong opponent is based on the assumption that the voter and the authority distributing the secret keys are allowed to communicate once via an untappable channel, before the voting phase (e.g., at registration). In order to allow the voter to dispute the published results of the voting procedure, a signed receipt of his ballot should be transmitted by the polling office to the voter at the end of the voting phase. Obviously, in order to create an acceptable remote voting platform, other techniques should also be used to provide other required properties [31] such as the possibility to vote anonymously [16] or verifiability [5]. As with the other mentioned schemes, the technique we propose does not protect the user from an attacker denying him access to a computer or rendering his ballots void by entering random values as input to the voting system. Also, obvious as it is, it does not protect from an attacker deducing that a coerced voter did not follow his instructions if, for example, not a single voter did vote for the attacker's choice. In such case, if the results are published, the attacker will obviously know that the voter cheated him or that the system did malfunction. To summarize, our mental booth has the following properties:

- The voter obtains the guarantee (i.e. receipt) that his vote has been correctly received by the polling office.
- The voter cannot convince the adversary of whom he voted for by using his receipt.
- The coercer cannot force the user to cancel a vote, nor to vote for a particular candidate, even if the user reveals his secrets and lets the adversary vote on his behalf. That is, the adversary cannot tell apart fake and valid secrets.
- If for some reason (e.g., disability) the voter is unable to vote, he can safely delegate his voice.
- Mental booths can be plugged into existing e-voting schemes in order to achieve remote voting with equivalent security.
- The only effort required from the voter is to remember a number in $\mathbb{Z}_n$ where n is linked to the security parameter.

As suggested in section 4, one can come up with many enhancements of this proposal in order to increase usability by using, instead of numbers in $\mathbb{Z}_n$, representations such as pictures, sounds, etc. and requiring from the user only his ability to remember the chosen secret after seeing (or hearing) it among others. An implementation of an i-voting scheme using a variant of the presented technique (and more to provide other desirable properties such as anonymity and verifiability) is available at `http://qualsec.ulb.ac.be`.

Outline. Section 2 starts by over-viewing definitions of coercion-resistance. Next, the notion of mental booth is introduced and a property necessary for over-the-shoulder coercion resistance is formulated. Section 3 proposes a simple example that is shown to be over-the-shoulder coercion-resistant, according to the definition given in section 2. A variant of this scheme is proposed in section 4 with

the aim to increase usability. Section 4 also discusses the possibility of vote delegation. Finally, section 5 concludes and discusses to what extent our solution also applies to non-remote voting.

2 Definitions

Examples of security requirements for e-voting protocols are privacy, accuracy, fairness, robustness, universal verifiability, incoercibility and receipt-freeness [30,28]. In this work, we focus on coercion-resistance, a property that is linked to receipt-freeness [6] and for which different definitions can be found in the literature. We start by over-viewing some current definitions and notice that they do not capture over-the-shoulder coercion resistance. Then, we formulate a property of remote booths that is necessary for protection against over-the-shoulder coercers. A *mental booth* is simply a remote booth that satisfies this property, thus offering coercion resistance against an adversary that monitors and influences the honest voter at any point of the protocol (possibly during the whole execution). This allows to protect the voter from malware that might be present in his machine, but also against an adversary who uses the machine on behalf of the voter.

Coercion-resistance. Several definitions for coercion resistance have been proposed in the literature. Juels *et al.* define coercion resistance as the following four requirements [5,22].

1. Receipt-freeness. A coercer cannot force a voter to cast a certain vote and to provide a receipt that would certify his vote.
2. Immunity to simulation attacks. A coercer cannot exploit secrets revealed by the voter since he cannot tell apart real and fake secrets.
3. Immunity to forced abstention attacks. A coercer should not be able to tell whether a particular voter has voted or not, so that he cannot force the voter to abstain.
4. Immunity to randomization attacks. A voter cannot be forced to divulge or nullify his vote by using random messages chosen by the coercer.

However, they assume a remote voting setting where the machines used to cast a vote are not compromised [22]. On the other hand, they take into account forced abstention attacks. As noted in [29], anonymous channels are necessary to achieve immunity to forced abstention since monitoring the (lack of) activity of a non-anonymous channel allows the adversary to make sure that the voter did abstain as instructed. Moran and Naor [29] define receipt-freeness based on an ideal voting functionality, building upon the definition of coercion proposed by Canetti and Gennaro in the context of multiparty computation [8]. However, their solution is tailored for settings where a physical booth is available: they assume the existence of an untappable channel between voter and authority during the voting phase. This assumption is also made in [14] where the authors formally define coercion-resistance and receipt-freeness in terms of process algebra (applied π-calculus).

The difference between their two definitions lies in the ability of the adversary to interact with the voter during the voting phase. That is, in [14], both notions capture the property that a voter cannot cooperate with a coercer in order to prove which candidate he voted for. But in the weaker notion of receipt-freeness, the adversary can only interact with the user before and after he voted but not during the voting phase. The intuition that receipt-freeness is necessary to achieve coercion-resistance has been formally confirmed in [14].

Over-the-shoulder coercion-resistance. In the case of remote voting, in particular Internet voting, an additional security requirement arises: resistance against shoulder-surfing [20]. To the best of our knowledge, very few schemes [11] deal with this class of attack. In [11] the voter chooses a password and also a set of *panic passwords* allowing the voter to fake a session when coerced by a visible adversary. As stated in [11], this solution requires some user training and also requires the voter to remember a *set* of passwords. Finally, one major drawback of this solution is that if the voter is unaware of the presence of the adversary (e.g. keylogger), he will use his actual password and allow the adversary to replay it.

Assumptions. The only secret involved in the use of the voting interface is a symmetric key $k \in \{0,1\}^\eta$ where $\eta \in \mathbb{N}$ is the security parameter. This key could be chosen by the key generation authority and communicated to the voter via an untappable channel. The authority then associates an anonymous identifier to the key and communicates it to the user. In practice, the exchange can be done physically upon registration of the voter. In order to provide over-the-shoulder coercion-resistance against the adversary described above, we base our scheme on the requirement that if a user reveals his key k, it is impossible for him to convince the adversary that it is the right one. That is, no matter how the adversary interacts with the voting booth, he must not be able to determine if a revealed key is correct, thus preventing vote selling. The adversary can still guess the key with negligible probability $2^{-\eta}$.

Modeling the booth. We model a voting booth as an interface that offers limited actions to the voter (e.g. vote, re-vote, verify, etc.) each generating a feedback. This definition can be instantiated rigorously using different formalisms. In the next section, we model the booth as a finite state machine whose state transitions are triggered by the available actions. It is assumed that the user successfully identified himself to the interface using his anonymous identifier. This opens a new session between the voter and the interface that can be secured according to the underlying voting scheme (we focus on the voting booth that can be built on top of this scheme). For any set of actions, the corresponding feedbacks must be chosen so that no adversary can tell if the voter followed his instructions or if he just pretended to do so. If so, the resulting interface is coercion resistant against over-the-shoulder adversaries. This leads to the following security definition.

Security definition. The security definition is based on the following intuition. Any *dishonest* execution starting from any *honest* state of the system, should be indistinguishable from an *honest* execution starting from the initial state (i.e.

first use of the interface). The set of dishonest executions is modeled by requiring that the actions do not make use of the secret k. On the contrary, the set of honest executions are defined such that all actions use k. That is, we assume a honest user always uses the correct key when voting. This approximation of honest vs. dishonest executions is sound: an honest execution cannot be considered dishonest, and a dishonest execution is considered honest with probability negligible in the security parameter. The rigorous meanings of "indistinguishable", "execution", and "state" can be adapted to the formalism underlying the proof. In the next section, the proposed system is modeled as a labeled transition system and "indistinguishability" is established by means of trace equivalence in a rather exhaustive manner.

3 Example of Mental Booth

We propose a simple mental booth for the case of "choose 1 out of l" elections. Before defining the interface, we start by describing the different phases of the voting procedure (see figure 1). Keeping in mind the objective and scope of this paper, we will not provide the same amount of detail for each phase.

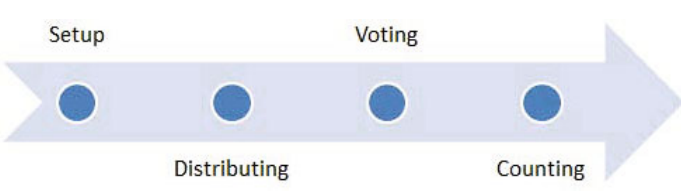


Fig. 1. Phases of a voting procedure

- Set-up: The key generation office creates random pairs of the form (id, k) where id is an anonymous identifier and k a secret integer in $\mathbb{Z}_n$.
- Distributing: the random pairs are (encrypted then) transmitted to the poll office. Upon registration of a voter, the key generation office picks at random a pair (id, k) and transmits it to the voter using an untappable channel. This phase is the only moment when the voter should not be observable from the attacker. Using designated verifier signatures [16,13,26], it is possible to provide an additional signed receipt. This signature should be verifiable only by the voter and a judge (or by extension a (group of) witness(es) considered as legitimate support to the eyes of a judge) in order to provide the voter with a proof in case of dispute later on. Of course, by doing so, the witness gains the same power of coercion against the voter as the authority of distribution.
- Voting: during this phase, the voter goes and identifies himself to the interface (e.g. a website). The voter is asked to associate a distinct number from $\mathbb{Z}_n$ to each candidate. If the voter wishes to select candidate 2, he associates k to that candidate and random numbers to other candidates. By doing so the polling office is able to determine which candidate the anonymous voter id voted for. On reception of the vote, the polling office provides a signed receipt of the casted ballot.

– Counting: the tally is created based on the casted votes, the associated anonymous identifiers, the secret keys and associated identifiers.

Let us now describe the actions in more detail. We assume without loss of generality that the list of candidates is ordered, so that a vector of l numbers is sufficient to make the ballot unambiguous.

$\mathsf{newvote}(k_1, \ldots, k_l)$ In order to cast a ballot, the voter associates one integer $k_i \in \mathbb{Z}_n$ to each candidate. If $k_i = k$ for some i in $\{1, \ldots, l\}$, then the vote is validated for candidate c_i. Otherwise, the vote is discarded. In the case of a malformed ballot (e.g. vectors do not have the same length, or some components are equal), the user receives a feedback $\otimes$, otherwise, he receives a feedback $\top$ meaning that the message was successfully sent and the corresponding vote received and saved (possibly overwriting a previous vote).

$\mathsf{receipt}(k_1, \ldots, k_l)$ This function can be used to terminate a session. The input must again be a vector of l integers in $\mathbb{Z}_n$. If the vector contains k, the message is considered honest, otherwise dishonest. In the case of an honest action, the interface checks if an honest ballot has already been received and replies either with a signature (receipt) r if the ballot was found and with a feedback $\perp$ otherwise. Similarly, if the action is not honest, the interface returns either a signed (dishonest) ballot in case one was already received or a message $\perp$ otherwise.

Practical considerations. Clearly, it is unlikely that users left on their own will enter values appropriately. For example, a user unaware of the attacker's presence might start assigning his secret number to the selected candidate, and only then assign the number zero (for example) to all the other candidates. This issue can be prevented using an appropriate implementation of the interface. For example, the implementation must forbid duplicated values and must allow the user to associate a number to the next candidate only when numbers have already been assigned to all previous candidates.

Concrete values for the security parameter can only be given according to a specific application. However, in our case, using brute force to guess-determine the secret is not a threat since the attack cannot be carried off-line. On-line guessing are easy to prevent using an exponential backoff/delay. Furthermore, the interface is built so that it is impossible to determine whether a guess is correct or not. Therefore, depending on the election, the secret to memorize could be shorter than a PIN code.

Remark. In order to illustrate what could possibly go wrong, let us assume an attack where the voter told the adversary his number is k' when in fact it is k. Assume the adversary sends the ballot $(k', 22, 38)$. Then a message $\mathsf{receipt}(k', 1, 2)$ should return r' whereas a message $\mathsf{receipt}(22, 1, 2)$ should return $\perp$. However, the interface is unable to know which of the numbers $k', 22$ or 28 the adversary is using as k. This would allow the adversary to identify a user that pretended to reveal the correct secret k. In order to exclude this possibility, the interface

must reply with r' if any of the numbers in a message receipt were associated to the ballot. Therefore, it is necessary that the interface adopts the same behavior when receiving an *honest* ballot. That is, after sending the ballot (k, k_1, k_2), a call to the function receipt with argument k should yield the same result as with argument k_1 or k_2, even if the interface does know which integer is the correct one. This requirement implies that the probability of an adversary using the secret number moves from $2^{-\eta}$ to $l2^{-\eta}$.

Defining the interface. The feedbacks returned by the interface are determined by the following pseudo-code where it is assumed that the interface uses variable hb (db) to store the last received honest (dishonest) ballot. We abuse somewhat the notation by writing $v \cup w$ ($v \cap w$) for the set containing all components of vectors v, w (common components of v, w). Also, for the sake of clarity, we omit the pseudo code for indicating a malformed ballot (feedback $\otimes$). This has no impact on the proof since the interface does not use k to realize that a ballot is malformed.

- Procedure newvote on well formed input $K \in \mathbb{Z}_n^l$:
  1. if $k \in K$ then
     $\mathsf{hb} = K$; return $\top$
  2. if $k \notin K$ then
     $\mathsf{db} = K$; return $\top$
- Procedure receipt on well formed input $K \in \mathbb{Z}_n^l$:
  1. if $k \in K$ then
     if $\mathsf{hb} \neq \emptyset$ and $K \cap \mathsf{hb} \neq \emptyset$ then return r
     else return $\perp$
  2. if $k \notin K$ then
     if $K \cap (\mathsf{hb} \cup \mathsf{db}) \neq \emptyset$ then return r'
     else return $\perp$

Security Proof

According to the pseudocode above, the interface can only be in one the following states.

A	initial state, no ballots received
B	received honest ballot but no dishonest one
C	received dishonest ballot but no honest one
D	received both honest and dishonest ballots

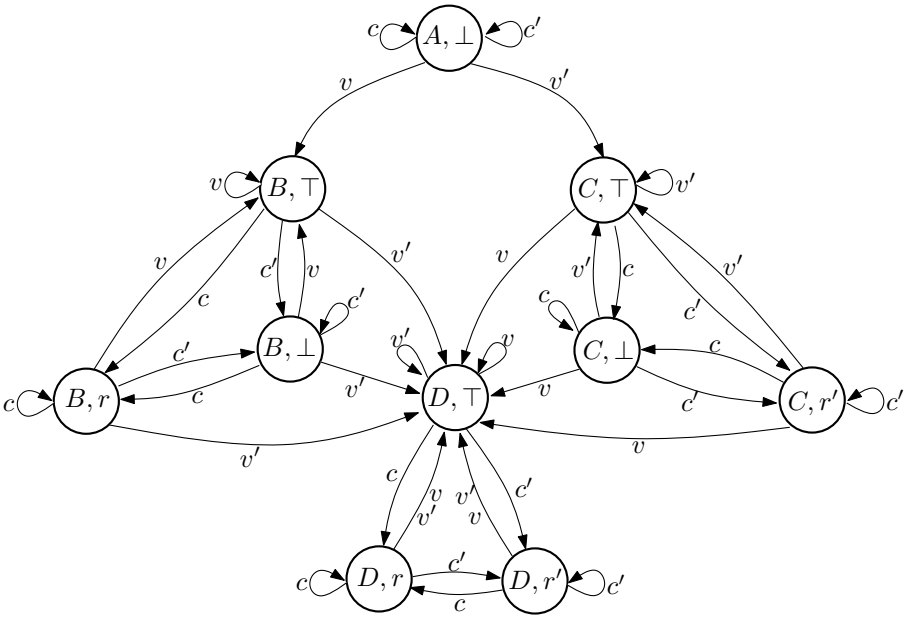
The interface can be defined by a labeled transition system, i.e., a directed graph whose nodes correspond to states and edges are labeled by actions that trigger state transitions. Our interface uses the following actions and feedbacks.

Actions	Feedbacks
v honest vote (with k)	$\top$ vote received
v' dishonest vote (without k)	$\perp$ no vote received
c honest check (with k)	r receipt for honest vote
c' dishonest check (without k)	r' receipt for dishonest vote

In order to comply with the formal definition of trace equivalence (e.g. see [27]), the transitions should be given in the following form.

$$(A, v) \xrightarrow{\top} \{(B, v), (B, v'), (B, c), (B, c')\}$$
$$(A, v') \xrightarrow{\top} \{(C, v), (C, v'), (C, c), (C, c')\}$$
$$\text{etc.}$$

Based on those transitions, two graphs can be defined, one representing all honest executions, the other all dishonest ones. The two graphs are then trace equivalent if all paths (transitions) have the same labels. However, for our purpose it is convenient to assume that the feedback is part of the state (see figure). For example, performing action v from state $(A, \bot)$ yields the observable feedback $\top$ and executing c from the same state returns feedback $\bot$. Thus, equivalence holds when honest and dishonest paths generate the same feedbacks.



In the graph, adversarial (honest) executions follow arcs labeled with actions v' or c' (v or c). The initial state of a dishonest execution can be any honest state and that execution must be equivalent with an honest execution that starts at state $(A, \bot)$. That is, for any honest initial state s and for any sequence of actions a' in $\{v', c'\}^*$, the execution of a' from state s produces the same feedbacks as the corresponding honest actions a in $\{v, c\}^*$ executed from state $(A, \bot)$. This equivalence can be tested exhaustively in $O(H \cdot L \cdot 2^L)$ where H is the number of honest states and L the length of the longest cycle that visits each node once. The probability of the adversary using k is $xl/2^{-\eta}$ where x is the number of actions executed, l the number of candidates, and η the security parameter

($\eta = \log_2(n)$). In this rather small example, one can check manually that the executions are trace equivalent.

4 Increasing Usability

Electronic voting must be accessible to the widest possible range of users. Keeping this in mind, this section proposes an alternative version of the mental booth introduced in the previous section. In this version, the interface displays m values in $\mathbb{Z}_n$ that the voter has to bind to candidates, including the secret value k. Actually, the integers in $\mathbb{Z}_n$ can be mapped to representations that are easier to remember (e.g. pictures). The voter would only have to associate the given representations to the given candidates in order to vote. This has several implications:

1. The probability of an attacker submitting a ballot supporting his choice is now $1/m$, where m is the number of values displayed by the interface. Also, the probability of casting a valid ballot for the wrong candidate is l/m.
2. It is likely that, in the previous version, a user would introduce "fake values" that are not distributed uniformly over $\mathbb{Z}_n$. Forcing him to choose values among truly random ones might actually increase security.
3. Since the user only has to associate values with choices, there should not be any type (word, number, picture, music) of value unusable.
4. The user must not memorize his secret value, he is only required to identify it among other values.

An attacker could ask the voter to reveal his secret value before interacting with the booth. If afterwards the revealed value is not displayed by the interface, the adversary would know that the voter lied. Therefore, it is imperative to always display a fixed set of values and to ask the voter to memorize one of those values in addition to his secret k. By doing so, the voter has the possibility to reveal one of the $m - 1$ values that will appear on the webpage. If the adversary asks the voter to reveal both remembered values, the voter can safely argue that he only memorized the correct one.

Description. We will now describe this variant by describing each phase of figure 1 for an election with l choices.

- Set-up: For each voter, the authority chooses m random values and selects one of them as the voter's secret value. This value will form the shared secret key. This authority has to transmit the list of values for each voter to the authority in charge of the website who would then not learn any more information than the attacker would.
- Distributing: This phase is the same as the corresponding phase in section 3.
- Voting: During this phase, the voter goes and identifies himself with the website of the election and is shown the list of candidates along with m values. He is then asked to associate one different value to each candidate. He will associate his secret value to his choice and if and attacker is trying to

coerce him, he will associate one of the other values to the attackers choice. Then, the voter submits his vote. By doing so the polling office associates the casted vote and the anonymous identifier of the voter. On reception of the vote, the polling office provides a signed receipt of the casted vote.
- Counting: This phase is the same as the corresponding phase in section 3.

Absentee ballots. A property of mental booths is that it offers the possibility to delegate votes. This property offers an appropriate alternative when one does not wish to trade security against usability. An absentee could reveal his secret value to an honest person and ask that person to vote on his behalf. The receipt would convince an honest absentee that his vote is in the ballot box. Furthermore, using the scheme from section 3 or 4, a voter could ask someone to cast a vote on his behalf without revealing the selected candidate. Of course, a vote buyer cannot exploit this vote delegation, since the buyer has no guarantee that the vote is valid (i.e. it is not worth buying). This feature could be an important improvement over existing electronic voting systems (remote or local) for disabled persons currently forced to rely on the honesty of a helper.

Remarks. In order to prevent a user from voting (forced abstention), the attacker has to keep him under his surveillance during the whole voting period in order to ensure that the voter does not choose any of the values to associate to candidates. Even then, the attacker would still not be able to make him vote for a particular candidate but only to deny him the right to vote. As mentioned earlier, we do not consider forced abstention attacks also because it would require the use of an anonymous channel. The mentioned technique requires an untappable channel once during a brief period of time. If such a channel could be materialized by a permanently stationed distribution booth available to any citizen in the case of a regional election, such a channel would require a real identity and presence (by opposition to a virtual one) and thus could not suit the needs of virtual communities and their elections. Finally, if a voter forgets his secret, he should restart the distribution procedure. This should not have any impact in legislation where every voter is legally obliged to vote (e.g. Belgium) since only voting without using his secret would be considered as a legitimate blank vote.

5 Conclusions

In this paper, we present a technique that allows a voter to cast a ballot in front of an attacker without allowing the latter to learn information about the selected candidate nor to force the voter to vote for the attacker's choice. It turns out that the technique also allows a user to delegate his vote: by instructing someone how to complete the ballot and asking him to return a receipt, the voter is ensured that his instructions were followed without revealing the selected candidate.

Perhaps the proposed techniques might improve the security offered by physical booths: an adversary might not enter a physical booth with the voter, but he can force the voter to enter the booth with an inconspicuous camera

(for example) and ask him to record the procedure. In fact, mental booths also offer protection against electromagnetic eavesdropping (van Eck phreaking), an attack that applies to *non-remote* electronic voting [25] or against the new man-in-the-middle attacking the Diebold voting machines revealed by the VAT team of the Argonne National Lab recently in [19,18,3] (and earlier to the Sequoia AVC Voting Machine [4]).

Usability of mental booths can be largely improved either by a careful choice of actions/feedbacks or by using representations of the secret integers that are easier to remember. The framework for proving over-the-shoulder coercion resistance of voting booths is also subject to improvement. In particular, approaches that are more efficient than exhaustive state space exploration would allow to guarantee the security offered by very elaborated interfaces. In any case, the general approach can be used as a sound guarantee that adding functionalities (i.e. actions/feedbacks) to the interface will not jeopardize over-the-shoulder coercion resistance.

We did not consider to what extent security holds over multiple sessions. The proposed scheme requires from the voter to register once per election, or to remember a sequence of numbers, one number for each session.

References

1. Adida, B.: Helios: web-based open-audit voting. In: Proceedings of the 17th Conference on Security Symposium, pp. 335–348. USENIX Association, Berkeley (2008)
2. Adida, B., De Marneffe, O., Pereira, O., Quisquater, J.J.: Electing a university president using open-audit voting: analysis of real-world use of helios. In: Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections, EVT/WOTE 2009, p. 10. USENIX Association, Berkeley (2009)
3. Argonne National Laboratory, The Brad Blog: "Man-in-the-middle" remote attack on Diebold touch-screen voting machine by Argonne national lab (video) (2011), `http://www.youtube.com/watch?feature=player_embedded\&v=DMw2dn6K1oI`
4. Argonne National Laboratory, The Brad Blog: Remote vote tampering attack on a sequoia avc voting machine by argonne national lab (2011)
5. Backes, M., Hritcu, C., Maffei, M.: Automated verification of remote electronic voting protocols in the applied pi-calculus. In: Proceedings of the 21st IEEE Computer Security Foundations Symposium, pp. 195–209. IEEE Computer Society (2008)
6. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, STOC 1994, pp. 544–553. ACM, New York (1994)
7. Bohli, J.M., Mueller-Quade, J., Roehrich, S.: Bingo Voting: Secure and coercion-free voting using a trusted random number generator (2007), `http://eprint.iacr.org/2007/162`
8. Canetti, R., Gennaro, R.: Incoercible multiparty computation. In: Annual IEEE Symposium on Foundations of Computer Science, p. 504 (1996)
9. Chaum, D.: SureVote: Technical Overview. In: Preproceedings of the Workshop on Trustworthy Elections. In: WOTE 2001 (2001)
10. Chaum, D.: SureVote: How it works (2011), `http://www.surevote.com/`

11. Clark, J., Hengartner, U.: Selections: An internet voting system with over-the-shoulder coercion-resistance. In: Financial Cryptography and Data Security (2011)
12. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: IEEE Symposium on Security and Privacy, pp. 354–368. IEEE Computer Society (2008)
13. Dall'Olio, E., Markowitch, O.: Voting with designated verifier signature-like protocol. In: International Conference WWW/Internet, pp. 295–301. IADIS (2004)
14. Delaune, S., Kremer, S., Ryan, M.: Coercion-resistance and receipt-freeness in electronic voting. In: IEEE Computer Security Foundations Workshop, pp. 28–42. IEEE Computer Society, Los Alamitos (2006)
15. Dill, D.L., Castro, D.: Point/counterpoint: The u.s. should ban paperless electronic voting machines. Commun. ACM 51, 29–33 (2008)
16. Dossogne, J., Markowitch, O.: A tripartite strong designated verifier scheme based on threshold rsa signatures. In: International Conference on Security & Management, pp. 314–317. CSREA Press (2009)
17. Dossogne, J., Markowitch, O.: Online banking and man in the browser attacks, survey of the belgian situation. In: Goseling, J., Weber, J.H. (eds.) Proceedings of the 31th Symposium on Information Theory in the Benelux (WICSITB 2010), Rotterdam, The Netherlands, pp. 19–26 (2010)
18. Friedman, B.: Diebold voting machines can be hacked by remote control (September 27, 2011), `http://politics.salon.com/2011/09/27/votinghack/`
19. Friedman, B.: National Security Lab Hacks Diebold Touch-Screen Voting Machine by Remote Control With $26 in Computer (September 27, 2011), `http://www.bradblog.com/?p=8785`
20. Jefferson, D., Rubin, A.D., Simons, B., Wagner, D.: Analyzing internet voting security. Commun. ACM 47, 59–64 (2004)
21. Joaquim, R., Ribeiro, C.: CodeVoting protection against automatic vote manipulation in an uncontrolled environment. In: Proceedings of the 1st International Conference on Evoting and Identity, pp. 178–188. Springer, Heidelberg (2007)
22. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, pp. 61–70. ACM, New York (2005)
23. Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (eds.) Towards Trustworthy Elections. LNCS, vol. 6000, pp. 37–63. Springer, Heidelberg (2010)
24. Kenski, K.: To I-Vote or Not to I-Vote?: Opinions About Internet Voting from Arizona Voters. Social Science Computer Review 23, 293–303 (2005)
25. Kuhn, M.G.: Electromagnetic Eavesdropping Risks of Flat-Panel Displays. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 88–107. Springer, Heidelberg (2005)
26. Laguillaumie, F., Vergnaud, D.: Multi-designated verifiers signatures: anonymity without encryption. Information Processing Letters 102(2-3), 127–132 (2007)
27. Laroussinie, F., Schnoebelen, P.: The State Explosion Problem from Trace to Bisimulation Equivalence. In: Tiuryn, J. (ed.) FOSSACS 2000. LNCS, vol. 1784, pp. 192–207. Springer, Heidelberg (2000)
28. Magkos, E., Burmester, M., Chrissikopoulos, V.: Receipt-freeness in large-scale elections without untappable channels. In: Schmid, B., Stanoevska-Slabeva, K., Tschammer, V. (eds.) Towards the E-Society. IFIP, vol. 74, pp. 683–693. Springer, Boston (2002)

29. Moran, T., Naor, M.: Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)
30. Qadah, G.Z., Taha, R.: Electronic voting systems: Requirements, design, and implementation. Computer Standards & Interfaces 29(3), 376–386 (2007)
31. Sampigethaya, K., Poovendran, R.: A framework and taxonomy for comparison of electronic voting schemes. Computers & Security 25(2), 137–153 (2006)
32. Sanford, C., Rose, J.: Characterizing eparticipation. International Journal of Information Management 27(6), 406–421 (2007)
33. Tari, F., Ozok, A.A., Holden, S.H.: A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords. In: Proceedings of the Second Symposium on Usable Privacy and Security, SOUPS 2006, pp. 56–66. ACM, New York (2006)
34. Weldemariam, K., Villafiorita, A.: A survey: Electronic voting development and trends. In: Electronic Voting, pp. 119–131 (2010)