Simulation-based analysis of E2E voting systems

Olivier de Marneffe, Olivier Pereira, Jean-Jacques Quisquater

Crypto Group – Université catholique de Louvain
Place du Levant, 3
B-1348 Louvain-La-Neuve, Belgium
{olivier.demarneffe, olivier.pereira,
jean-jacques.quisquater}@uclouvain.be

Abstract. End-to-end auditable voting systems are expected to guarantee very interesting, and often sophisticated security properties, including correctness, privacy, fairness, receipt-freeness, ... However, for many well-known protocols, these properties have never been analyzed in a systematic way. In this paper, we investigate the use of techniques from the simulation-based security tradition for the analysis of these protocols, through a case-study on the ThreeBallot protocol.
Our analysis shows that the ThreeBallot protocol fails to emulate some natural voting functionality, reflecting the lack of election fairness guarantee from this protocol. Guided by the reasons that make our security proof fail, we propose a simple variant of the ThreeBallot protocol and show that this variant emulates our functionality.

1 Introduction

End-to-end (E2E) universally auditable voting systems, which include Punchscan [1, 2], Prêt à Voter [3], ThreeBallot, VAV, and Twin [4] for instance, attract more and more attention from the scientific community due to the highly desirable security properties they can offer while preserving a high level of usability. By including the delivery of receipts that can be used for verification on a public bulletin board, those systems allow voters to get confidence into the facts that their ballot selection has been cast as intended, properly recorded, and included in the final tally. A central challenge in the design of these systems is the construction of vote receipts that can be efficiently used for auditing elections, while not introducing the possibility of coercing voters.

While these voting systems are expected to guarantee sophisticated security properties, like correctness of the tally, ballot secrecy, election fairness, and receipt-freeness, only very few works address the precise definition (and proof) of the properties guaranteed by these protocols. This fuzziness on the properties guaranteed by protocols is actually also reflected in the definition of these properties: as an example, the notions of coercion resistance and/or receipt-freeness [5], which informally guarantee that a voter should not be able to convince anyone else of her vote, is defined in three different ways by Juels et al. [6], Delaune et al. [7], and Moran and Naor [8].

Dagstuhl Seminar Proceedings 07311
Frontiers of Electronic Voting
http://drops.dagstuhl.de/opus/volltexte/2008/1297

In this paper, we investigate the modeling and analysis of E2E voting systems through the case-study of the ThreeBallot system [4, 9], which presents the interesting characteristic of not relying on the use of any cryptographic scheme. We perform our analysis by comparing the considered E2E protocol to an *ideal functionality*, following the simulation-based security approach [10, 11] and the notion of secure protocol emulation, as used in the UC [12] and RSim [13] frameworks for instance.

To this purpose, we define a generic ideal functionality for voting, capturing the functionalities used by Groth [14] and Moran and Naor [8] for instance. We then describe the ThreeBallot protocol and compare it to a natural instance of our functionality. This analysis shows that the ThreeBallot protocol actually does not realize the considered functionality, reflecting the fact the election fairness in not guaranteed by this protocol (this has also been observed by Araujo et al. [15] and Clark et al. [16]). As a result, we propose a relaxed version of our functionality, which quantifies the information leaked by the receipts issued in ThreeBallot elections.

Observing the reasons that made the emulation proof fail for our original functionality, we then propose a simple variant of the ThreeBallot protocol and show that it securely emulates our original functionality. Relying on the composability of the secure protocol emulation notion, we split our proof into two stages: in a first stage, we show that the use of the ThreeBallot receipts guarantees the authenticity of the public bulletin board. Then, relying on this first result, we show that the ThreeBallot protocol with authentic bulletin boards emulates our functionality. We eventually observe that, up to the forced abstention case that we did not consider here, a protocol securely emulating the functionality we consider here also satisfies the receipt-freeness property proposed by Moran and Naor [8].

2 Modeling voting systems

2.1 Voting in an ideal world

Voting systems involve sophisticated protocols, designed to be used in complex environments. Specifying their properties in such context might therefore be a challenging task. A more accessible challenge would probably be to specify the expected behavior of a voting system in an ideal world, where parties can be trusted, and communication channels are assumed to be private and authentic. Intuitively, in such a context, each voter V_i from a set $\{V_1, \dots V_n\}$ will simply identify herself by showing her identifier U_i to a trusted party $\mathcal{F}_{vote}$, and give this party her vote x_i. Then, when $\mathcal{F}_{vote}$ has received all votes, it will compute the election result as $f(x_1, \dots, x_n)$, and broadcast it.

This specification of a voting system probably corresponds to the first intuition we have from a voting process: all votes are as private as possible (nobody will ever be able to infer more about them than what can be inferred from the election outcome), and the election outcome is always correct, as $\mathcal{F}_{vote}$ is assumed to be trusted.

However, this specification is clearly too strong: we usually cannot expect a voting systems to hide the mere fact that someone voted to external observers. Besides, it should probably be tolerated that an adversary can make an election fail, by sabotaging some part of the election process or corrupting some parties. The way it is tolerated that an adversary interferes with a voting system will typically change from one system to another. Therefore, we will leave this part of the voting functionality unspecified for the moment, and simply consider that $\mathcal{F}_{vote}$ accepts to play some protocol π with the adversary, in the ideal world. (We will see concrete instances of π later). The behavior of the functionality $\mathcal{F}_{vote}$ is illustrated in Fig. 1.

2.2 Voting in the real world

We now would like to show that a real voting system securely emulates this ideal functionality. To this purpose, we use the notion of secure emulation, as proposed in the universally composable (UC) security framework [12], or in the reactive simulatability framework [13] for instance.

Intuitively, one says that a protocol emulates an ideal functionality if everything an adversary can do by interacting with the protocol can be matched by another adversary interacting with the ideal functionality. Now, if the behavior of the ideal functionality can be regarded as safe, this implies that the adversaries interacting with the protocol do not harm. The simulation-based approach of security has two main benefits: first, the ideal functionality being typically much simpler than the protocol, it is much easier to understand what a sophisticated protocol really does by looking at the functionality it emulates. Secondly, secure emulation definitions usually come with secure composition theorems, which allow using a functionality as a component of larger protocols while guaranteeing that the security properties will be preserved when the functionality is replaced by any protocol emulating it.

In the following sections we investigate the use of this secure emulation notion on a example: the ThreeBallot voting system.

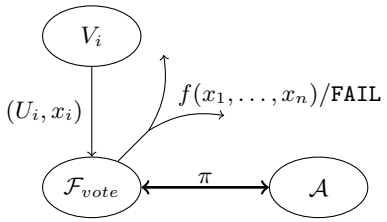

Fig. 1. Template of the ideal functionality for an E2E voting system

3 The ThreeBallot E2E voting system

The ThreeBallot system, originally proposed by Rivest [9] and later improved in collaboration with Smith [4], is a paper-based E2E system with the appealing specificity that it does not make use of any cryptographic algorithm or any other sophisticated computational procedure in any stage of its use. We briefly summarize the ThreeBallot system definition and the related analysis works.

3.1 Protocol description

A voter taking part to an election using the ThreeBallot system receives a paper *multi-ballot* made of three ballots, each containing the list of candidates with bullets next to them, but all differing by a ballot-ID present at the bottom of each ballot. Those ballot-IDs are unique, random, all generated independently of each other, and assumed to be too complex to be memorized. An example of multi-ballot that can be used for a race of five candidates is proposed in Fig 2.

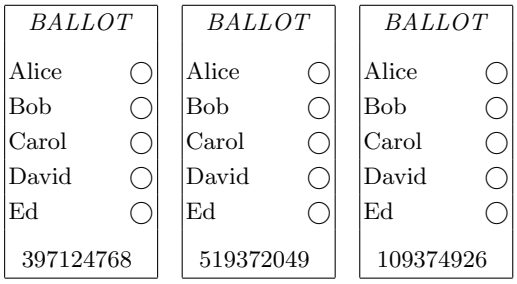

Fig. 2. An example of empty multi-ballot for a single race election with five candidates.

In order to express her opinion, the voter places the three ballots side-by-side in the voting booth, and fills them as follows: (i) if she approves a candidate, she randomly fills *two* of the three bullets corresponding to that candidate, while (ii) she randomly fills *one* of the three bullets corresponding to the candidates she rejects. As a result, each row of a filled multi-ballot contains one or two filled bullets (not zero, nor three).

When this is done, the multi-ballot is inserted into a "checker machine" that verifies whether it was correctly filled. If it is the case, the voter asks the checker machine to produce a copy of one of the three ballots, which will be used as a take-home receipt, and the machine then drops all three original ballots in the ballot box. One may observe that the receipt cannot be used by the voter to convince a third party of the candidate(s) for which she voted: given one ballot, it is possible to build a pair of ballots so that the reconstructed multi-ballot is a valid vote for any candidate(s) of the election.

Once the election is over, all the casted ballots are mixed and published on a Public Bulletin Board (PBB). The outcome of the election can then be computed by anyone: the number of votes for one candidate is the number of filled bullets for this candidate minus the number of voters. Each voter can also be assured that her vote was taken into account properly by checking that her receipt is on the PBB (using the ballot-ID). This verification process guarantees that, if everyone checks the integrity of its receipt on the PBB, the probability that someone can alter t ballots without being noticed is actually equal to $(2/3)^t$.

3.2 Related works – Analysis and critics of ThreeBallot

It has been observed and discussed by several authors that, when the number of election candidates is large, the receipt-freeness property of ThreeBallot cannot be guaranteed. Notably, Appel [17] and Strauss [18, 19] showed that a receipt associated with the PBB may result in the multi-ballot reconstruction and thus to voter privacy loss, which can lead to attacks by coercion or vote-selling. This attack is essentially possible because the number of ways of filling a ballot grows exponentially with the number of candidates, which makes it highly probable, when the number of voters is small, that only two ballots can be gathered with one given receipt to make a valid vote. Some Rivest's students at MIT showed that this kind of attack is practical [20]: they corrupted a mock election organized for a course. As a result, Rivest and Smith [4] introduced the *Short Ballot Assumption*, stating that the length of the ballots must be kept small (possibly by splitting them into several parts), which guarantees that a reconstruction is not likely to be possible. Concrete bounds on the length of the ballots have been estimated by several authors, including [19, 21].

Another issue, which we address in more detail in this paper, is the fairness of the elections: Araujo et al. [15] and Clark et al. [16] show that it is possible to approximate the outcome of an election by using a few receipts only. As far as we know, no practical solution to this problem has been proposed.

Although these researches pointed issues in ThreeBallot, we are still convinces that this protocol remains quite appealing considering the simplicity of the proposed solution.

4 Modeling the ThreeBallot protocol

We now describe our modeling of the ThreeBallot protocol, and a simple instance $\mathcal{F}_{vote}(\pi_{TB})$ of the $\mathcal{F}_{vote}$ ideal functionality, that we would expect to be emulated by this protocol. However, we will observe that the ThreeBallot protocol does not emulate our candidate $\mathcal{F}_{vote}(\pi_{TB})$ functionality. Examining the reasons that make the secure emulation proof fail, we will propose a relaxed version of the $\mathcal{F}_{vote}(\pi_{TB})$ functionality, which will give a better idea of the quantity of information leaked by the ThreeBallot protocol.

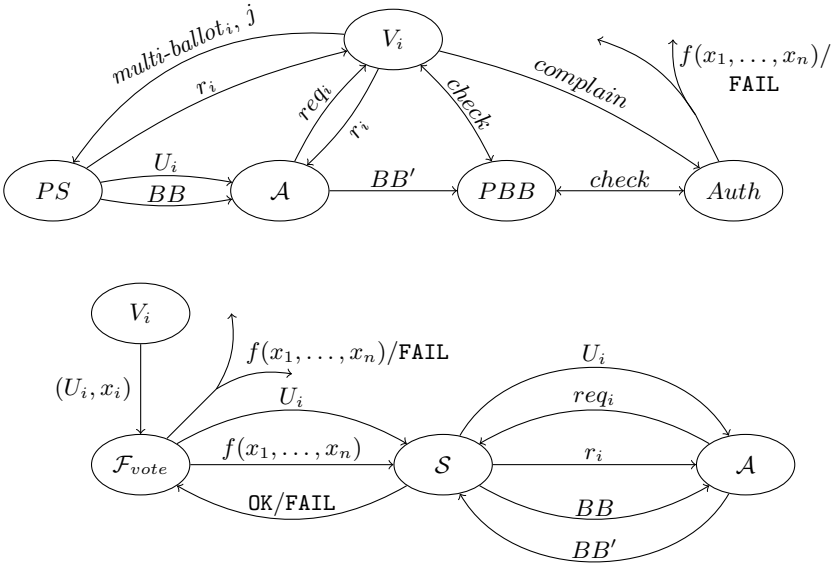

Fig. 3. Real world model (top) — Ideal world model (bottom)

4.1 Ideal world and real world definitions

Real world model Our modeling of the real world interactions taking place in the ThreeBallot system is depicted on top of Fig. 3. These interactions take place as follows. First, the voter V_i casts her vote using a (valid) *multi-ballot*$_i$ sent to the poling station (PS), together with the index j of the ballot she would like to have as a receipt. The polling station then sends this receipt, r_i, back to V_i. It also informs the adversary $\mathcal{A}$ that this voter voted by sending him the identifier of the voter (U_i). The adversary may now ask V_i to show him her receipt through the instruction req_i. We consider that the voters answer this request every time but we will bound the number of times the adversary can issue it.

Once the casting period is over, the polling station sends the ballot box (BB) to the adversary who can modify it as he wishes in order to produce a new ballot box (BB') which is sent to the public bulletin board (PBB). So, we assume that the adversary is not able to make any change in the ballot box until the end of the vote casting process.

The audit process consists in several interactions between the voters, the PBB and an authority centralizing the complains. Voters check whether their receipts are present on the bulletin board and complain to the authority if it is not the case. After verification, the authority records the objections and, when the time

for complaining is over, broadcasts the outcome of the election $f(x_1, \ldots, x_n)$ (that is, the number of votes casted for the different candidates), or declares that the election failed (due to tampering detected during the audit phase).

Ideal world model We now describe a candidate ideal functionality $\mathcal{F}_{vote}(\pi_{TB})$ for the ThreeBallot protocol, based on the $\mathcal{F}_{vote}$ functionality described in Section 2.1, and in which the π_{TB} protocol describes the interactions between the ideal voting functionality and the adversary.
$\mathcal{F}_{vote}(\pi_{TB})$ executes as follows:

1. On input x_i from voter V_i, $\mathcal{F}_{vote}(\pi_{TB})$ stores x_i and transmits V_i's identifier U_i to the adversary;
2. When the vote casting process is complete, $\mathcal{F}_{vote}(\pi_{TB})$ sends the election outcome $f(x_1, \ldots, x_n)$ to the adversary;
3. On input `OK` or `FAIL` from the adversary, $\mathcal{F}_{vote}(\pi_{TB})$ broadcasts $f(x_1, \ldots, x_n)$ or `FAIL` (respectively) as election outcome.

We observe that, in this definition, the protocol π_{TB} counts three types of messages. Messages of the first type notify the adversary that the voter V_i casted her vote. The same information is transmitted to the adversary in our model of the ThreeBallot protocol. The second message type corresponds to the sending of the election outcome to the adversary, waiting for an approval which is sent through the last message of the π_{TB} protocol. This corresponds to the fact that, in the ThreeBallot protocol, the adversary is assumed to be able to access and change the ballot box before it appears on the bulletin board. However, the election integrity is perfectly guaranteed here: the ideal adversary is able to make the election fail, but not to make it result in a wrong tally. (Actually, small discrepancies between the ideal and real worlds will however be tolerated, reflecting the fact that the ThreeBallot adversary has a noticeable probability to make small changes in the ballot box content without being noticed.)

It is worth noting that, as identity of voters is considered to be public information, this ideal functionality guarantees all the classical security requirements for a voting protocol (correctness, privacy, fairness...). The only available information about the cast votes is what can be inferred from the election outcome, once it has been revealed.

4.2 Simulatability of the ThreeBallot protocol

Following the secure emulation notion discussed in Section 2.2, we now would like to show that everything that can be done by an adversary interacting with the ThreeBallot protocol can be matched by another adversary interacting with the $\mathcal{F}_{vote}(\pi_{TB})$ functionality. To this purpose, we build a *simulator* $\mathcal{S}$ which will interact with $\mathcal{F}_{vote}(\pi_{TB})$ and use these interactions to feed $\mathcal{A}$ with a consistent view of the real world protocol execution. So, the ideal world adversary is actually made of a simulator $\mathcal{S}$ that runs a copy of the real world adversary $\mathcal{A}$. A high-level view of these interactions is represented in the lower part of Fig. 3.

$\mathcal{F}_{vote}(\pi_{TB})$-based simulation impossibility In order to simulate a real world protocol execution, the simulator $\mathcal{S}$ has to (i) provide $\mathcal{A}$ with the user-id of the voters, (ii) answers $\mathcal{A}$'s requests for vote receipts, (iii) send $\mathcal{A}$ a ballot-box that is going to be consistent with the election result, (iv) decide whether $\mathcal{A}$'s changes in the fake ballot box are supposed to lead to an election fail.

The first part of this simulation is trivial, as $\mathcal{F}_{vote}(\pi_{TB})$ gives $\mathcal{S}$ the user-id of the voters: $\mathcal{S}$ can just forward this message to $\mathcal{A}$. However, the second and third parts are much more problematic.

Consider, for the sake of simplicity, a single race election with only two traditional candidates, Alice and Bob. We can observe that, for each of these candidates, a voter has actually 9 ways to express her vote: these votes are depicted in Tab. 1.

Multi-ballots for Alice	**Multi-ballots for Bob**
[○][●][●] [○][●][○] [●][○][●]	[○][○][●] [○][●][○] [○][○][●]
[●][●][○] [●][○][●] [●][●][○]	[○][●][○] [●][○][○] [●][●][○]
[●][○][●] [●][○][●] [○][●][○]	[○][●][○] [●][○][●] [○][●][●]

Table 1. Possible multi-ballots in a two candidates election

We can see every possible ballot does not occur with the same probability. Tab. 2 shows the different distribution for specific election outcomes. The last line of the table gives the distribution of the receipts for any election outcome when there is a proportion p of votes for Alice. Obviously, this is also the distribution of the ballots in the ballot box.

Receipts	$r = [\circ\,\circ]$	$r = [\bullet\,\circ]$	$r = [\circ\,\bullet]$	$r = [\bullet\,\bullet]$
100% "Alice"	2/9	4/9	1/9	2/9
Tie	2/9	5/18	5/18	2/9
100% "Bob"	2/9	1/9	4/9	2/9
Prop. p for Alice	$\frac{2}{9}$	$\frac{1}{9} + \frac{p}{3}$	$\frac{1}{9} + \frac{1-p}{3}$	$\frac{2}{9}$

Table 2. Distribution $\{r_i\}$ for different election outcomes – in the real world

However, the simulator may have to produce receipts during the election process: the adversary is able to ask parties for their receipts before all votes have been cast. Looking at the distributions in Tab. 2, he will however not be able to produce receipts distributed as in the ideal world if he does not know p, that is, if he does not know the election outcome in advance.

This impossibility to simulate a view of the real world election by interacting with $\mathcal{F}_{vote}(\pi_{TB})$ actually reflects the lack of election fairness guarantee of the ThreeBallot protocol: by looking at receipts, a real world adversary is actually able to obtain an estimation of the election outcome.

A vote leaking version of $\mathcal{F}_{vote}(\pi_{TB})$ In order to reflect this lack of fairness in the ThreeBallot protocol, we propose a variant of our ideal functionality, which we call $\mathcal{F}_{vote}(\pi_{TBL})$, which is an instance of the $\mathcal{F}_{vote}$ functionality with a protocol π_{TBL} leaking some information on the votes casted to the adversary (while the π_{TB} protocol was keeping those votes perfectly private).

The $\mathcal{F}_{vote}(\pi_{TBL})$ functionality is essentially the same as the $\mathcal{F}_{vote}(\pi_{TB})$ functionality, except that the first part of its definition is modified as follows:

On input x_i from voter V_i, $\mathcal{F}_{vote}(\pi_{TBL})$ stores x_i and tosses a biased coin c giving "heads" with probability 4/9. Then if:

- if c is "heads", $\mathcal{F}_{vote}(\pi_{TBL})$ transmits V_i's identifier U_i to the adversary (as $\mathcal{F}_{vote}(\pi_{TB})$ does)
- if c is "tails", $\mathcal{F}_{vote}(\pi_{TBL})$ transmits U_i together with the vote x_i to $\mathcal{A}$ with probability 4/5, or with a fake vote $\overline{x_i}$ with probability 1/5

Essentially, the $\mathcal{F}_{vote}(\pi_{TBL})$ functionality leaks vote information following the distributions described in Tab. 2: with probability 4/9, the receipt chosen by the voter will have two identical bullets, and this is independent of the cast vote (so, no information is leaked); while with probability 5/9, the receipt contains two different bullets, and these bullets give a 4/5 probability of correctly guessing the candidate supported by the voter (so, the correct vote is transmitted with probability 4/5).

So, by interacting with $\mathcal{F}_{vote}(\pi_{TBL})$, a simulator can produce a convincing receipt r_i for V_i as follows, assuming d_i a random bit chosen by the simulator and b_i is the (possibly wrong) vote information transmitted by $\mathcal{F}_{vote}(\pi_{TBL})$:

$$\begin{array}{rl} \text{when receiving } (V_i) & \text{if } d_i = 0, \text{ choose } r_i = \left[\begin{smallmatrix}\circ\\ \circ\end{smallmatrix}\right] \text{ else } r_i = \left[\begin{smallmatrix}\bullet\\ \bullet\end{smallmatrix}\right] \\ \text{when receiving } (V_i, b_i) & \text{if } b_i = \text{Alice}, \text{ choose } r_i = \left[\begin{smallmatrix}\bullet\\ \circ\end{smallmatrix}\right] \text{ else } r_i = \left[\begin{smallmatrix}\circ\\ \bullet\end{smallmatrix}\right]. \end{array}$$

5 A tweak on the ThreeBallot protocol

In the previous section, we showed that the ThreeBallot protocol does not securely emulate the $\mathcal{F}_{vote}(\pi_{TB})$ functionality. Essentially, the reason of this non-emulation comes from the fact that the receipt of each voter contains some probabilistic information about the way she voted, which in turn prevents a simulator to produce a convincing receipt if he has no information about the content of the vote.

So, in order to securely emulate the $\mathcal{F}_{vote}(\pi_{TB})$ functionality, a protocol should not leak any *noticeable* information about the votes. This suggests a simple modification in the ThreeBallot protocol, which we present and analyze in this section.

5.1 Definition of the modified protocol

A simple way to modify the original protocol in order to achieve simulatability is to choose the receipt before expressing any voting preference: therefore, no information about the vote can be leaked by any receipt. The protocol variant we suggest is as follows:

(i) While in the voting booth, the voter first randomly fills one bullet per row of the multi-ballot (or receives such a pre-filled multi-ballot);
(ii) She decides which of the three ballots she wants to be copied and taken away as a receipt;
(iii) She casts her vote as in the original protocol, except that she is no more allowed to modify the ballot she pointed out as the receipt.

It is worth noting that she is still able to express her vote by filling one bullet on the row(s) she wants to vote for on the two remaining ballots.

How to insure that the voter follows the procedure, not modifying the receipt-ballot, is a practical issue we leave out of this study. For instance, this could be achieved by committing on the multi-ballot to the checker machine before stage (iii), and by requiring the checker machine to verify that the receipt is not modified during that stage.

5.2 Simulatability of the modified ThreeBallot protocol

We claim that our variant of the ThreeBallot protocol securely emulates the $\mathcal{F}_{vote}(\pi_{TB})$ functionality, under the following assumptions:

- there is a large proportion of the receipts which are unknown to the adversary, and
- the Short Ballot Assumption is satisfied.

We split our analysis into two steps, introducing an intermediate system IS between the real world system RWS and the ideal world system IWS depicted in Fig. 3.

The intermediate system Our intermediate system IS, which is depicted in Fig. 4, differs from RWS by the fact that the ballot box used in the tally is guaranteed to be authentic. In order to reflect this, we assume the existence of an authentic transmission functionality $\mathcal{F}_{auth}$, inspired from Canneti's message authentication functionality [12, Section 6.2], which behaves as follows: when it receives a ballot box, it forwards it to the adversary, then waits for an `OK` or `FAIL` message and, according to the value of this message, forwards the ballot box or the `FAIL` signal (respectively) to the authority.

IS is then built from RWS as follows: the polling station PS remains identical; each voter V_i is replaced by a voter V_i' that behaves as V_i except that it does not take part into any audit phase, the PBB is removed (it is not needed anymore as election integrity is now guaranteed), and the authority $Auth$ is replaced by an authority $Auth'$ that simply receives the ballot box or the `FAIL` signal and broadcasts the election outcome accordingly. The IS adversary is now built as a copy of the RWS adversary $\mathcal{A}$ interacting with a simulator $\mathcal{S}'$.

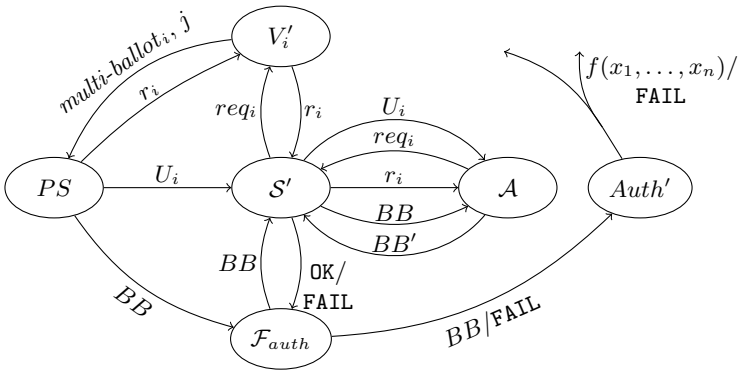

Fig. 4. Intermediate System (IS) for ThreeBallot

From *RWS* to *IS* The distinction from the the view of $\mathcal{A}$ in RWS and in IS can come from one single place: the election outcome. Indeed, $\mathcal{S}'$ can see all other messages that $\mathcal{A}$ sends or receives in RWS, and can just forward them transparently.

So, the challenge for $\mathcal{S}'$ is essentially to decide when it must send the signal OK or FAIL to $\mathcal{F}_{auth}$. We claim that $\mathcal{S}'$ can adopt the following strategy:

- If the ballot box BB' differs from BB by a ballot that $\mathcal{S}'$ forwarded between a voter and $\mathcal{A}$, then send the FAIL instruction to $\mathcal{F}_{auth}$;
- If any other ballots have been modified by $\mathcal{A}$, then estimate the probability that this modification would be detected in RWS, and send the OK/FAIL message to $\mathcal{F}_{auth}$ accordingly.

The probability that any change in the ballot box will be noticed can be estimated from the distribution of the receipts in RWS, which is given in Tab. 3.

Receipts	$r = \left[\begin{smallmatrix}\circ\\\circ\end{smallmatrix}\right]$	$r = \left[\begin{smallmatrix}\bullet\\\circ\end{smallmatrix}\right]$	$r = \left[\begin{smallmatrix}\circ\\\bullet\end{smallmatrix}\right]$	$r = \left[\begin{smallmatrix}\bullet\\\bullet\end{smallmatrix}\right]$
Probability	$\frac{4}{9}$	$\frac{2}{9}$	$\frac{2}{9}$	$\frac{1}{9}$

Table 3. Distribution of receipts in the real world (modified protocol)

The behavior we just described guarantees that the election will fail with the same probability in RWS and IS. On the other hand, when the election succeeds, discrepancies may appear between the election outcomes in these two worlds, as the outcome in IS does not reflect the changes in the ballot box performed by $\mathcal{A}$. However, we observe that the difference between the two distributions of the election outcome is small if the SBA assumption is satisfied, that is, when ballots are short, the number of voters is large, and the number of receipts obtained by the adversary is small.

For instance, in the case of our two candidates election, we can observe that the adversary's best strategy is to change ballots of the form $\left[\begin{smallmatrix}\bullet\\ \bullet\end{smallmatrix}\right]$. If we assume that the adversary does not see any receipt, a change in such a ballot will only be noticed with probability 1/9. Therefore, if the adversary tampers t of those receipts, he will get caught during the audit process with probability $1-(8/9)^t$ which is above 1/2 for $t > 6$ and around $1 - 10^{-5}$ for $t = 100$ (we expect that the strong impact of a single modification notice will convince him to keep the ballot box integrity).

So, we can consider that the audit process of the ThreeBallot guarantees that the tally of a ThreeBallot election will be very close from the correct one.

From IS to IWS We just showed that everything that an adversary can do by interacting with the real world system can essentially be done by an adversary interacting with our intermediate system. We now show that this behavior can also be matched by an adversary interacting with the ideal world system.

The main difference between IS and IWS lies in the fact that the IWS adversary does not see the real world receipts and ballot box anymore. Instead, he only sees the election tally before it is made public. So, we need to adapt the strategy of our simulator accordingly, transforming $\mathcal{S}'$ into the IWS simulator $\mathcal{S}$ depicted in Fig. 3.

The strategy of $\mathcal{S}$ in order to produce the receipts and the ballot box can be as follows:

- When $\mathcal{A}$ asks for a receipt from the voter V_i, $\mathcal{S}$ produces a receipt by selecting a ballot randomly according to the receipt distribution indicated in Tab. 3;
- When $\mathcal{A}$ receives the tally from $\mathcal{F}_{vote}(\pi_{TB})$, it produces a ballot box by internally playing an election leading to the same tally, while being consistent with the already produced receipts.

The rest of $\mathcal{S}$'s behavior can be taken from the behavior of $\mathcal{S}'$.

We observe that the distribution of the receipts sent by $\mathcal{S}$ to $\mathcal{A}$ perfectly matches the one of the receipts transmitted in RWS and IS. Besides, the Short Ballot Assumption guarantees that the adversary is not able to reconstruct the multi-ballot corresponding to a receipt (linked to a voter) by finding the two missing parts in the ballot box. Therefore, $\mathcal{A}$ will not be able to distinguish the distribution of the ballot box he receives in IWS from the one he receives in the other systems. This shows that our variant of the ThreeBallot protocol emulates the $\mathcal{F}_{vote}(\pi_{TB})$ functionality and concludes our analysis.

6 Conclusion

Throughout this paper, we reported a case-study on the use of standard cryptographic protocol analysis techniques, namely simulation-based security in the line of the UC framework, on the ThreeBallot end-to-end voting protocol.

These techniques provided a systematic way to detect potential issues in the ThreeBallot scheme, and suggested ways to avoid those issues: we propose a

variant of ThreeBallot that guarantees election fairness at the price of a linear loss in the precision of the election tally.

References

1. Fisher, K., Carback, R., Sherman, A.T.: Punchscan: Introduction and system definition of a high-integrity election system. In Ryan, P., ed.: IAVoSS Workshop On Trustworthy Elections (WOTE 2006). (June 2006)
2. Popoveniuc, S., Hosp, B.: An introduction to punchscan. In Ryan, P., ed.: IAVoSS Workshop On Trustworthy Elections (WOTE 2006). (June 2006)
3. Chaum, D., Ryan, P.Y.A., Schneider, S.A.: A practical, voter-verifiable election scheme. Technical Report CS-TR: 880, School of Computing Science, Newcastle University (December 2004)
4. Rivest, R.L., Smith, W.D.: ThreeVotingProtocols: ThreeBallot, VAV, and Twin. In: Electronic Voting Technology Workshop (EVT'07). (August 2007)
5. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections. In: Proceedings of the 26th ACM Symposium on Theory of Computing, Montreal, PQ, ACM (May 1994) 544–553
6. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: ACM Workshop on Privacy in the Electronic Society, ACM (2005) 61–70
7. Delaune, S., Kremer, S., Ryan, M.: Coercion-resistance and receipt-freeness in electronic voting. In: 19th IEEE Computer Security Foundations Workshop, (CSFW-19), IEEE Computer Society (2006) 28–42
8. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In Dwork, C., ed.: CRYPTO 2006. Volume 4117 of Lecture Notes in Computer Science., Springer (September 2006) 373–392
9. Rivest, R.L.: The ThreeBallot voting system. Avaiable from `http://people.csail.mit.edu/rivest/Rivest-TheThreeBallotVotingSystem.pdf`. Accessed on Aug 12, 2007. (October 2006)
10. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC'85). (1985) 291–304
11. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game a completeness theorem for protocols with honest majority. In: Proceedings of the 19th Annual ACM Symposium on the Theory of Computing (STOC), ACM Press (1987) 218–229
12. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In Naor, M., ed.: Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, IEEE Computer Society (2001) 136–145 Full version available on `http://eprint.iacr.org/2000/067`.
13. Pfitzmann, B., Waidner, M.: A model for asynchronous reactive systems and its application to secure message transmission. In: IEEE Symposium on Security and Privacy, Oakland, CA, IEEE Computer Society (May 2001) 184–200
14. Groth, J.: Evaluating security of voting schemes in the universal composability framework. In Jakobsson, M., Yung, M., Zhou, J., eds.: Applied Cryptography and Network Security, Second International Conference, ACNS 2004. Volume 3089 of LNCS., Springer (2004) 46–60
15. Araujo, R., Custodio, R.F., van de Graaf, J.: A verifiable voting protocol based on Farnel. In Benaloh, J., ed.: IAVoSS Workshop On Trustworthy Elections (WOTE 2007). (July 2007)

16. Clark, J., Essex, A., Adams, C.: On the security of ballot receipts in e2e voting systems. In Benaloh, J., ed.: IAVoSS Workshop On Trustworthy Elections (WOTE 2007). (July 2007)
17. Appel, A.A.: How to defeat rivest's threeballot voting system. Available from `http://www.cs.princeton.edu/~appel/papers/DefeatingThreeBallot.pdf`. Accessed on Aug. 12, 2007 (October 2006)
18. Strauss, C.: The trouble with triples. a critical review of the triple ballot (3ballot) scheme. Available from `http://www.cs.princeton.edu/~appel/voting/Strauss-TroubleWithTriples.pdf`. Accessed on Aug. 12, 2007. (October 2006)
19. Strauss, C.: A critical review of the triple ballot voting system, part2: Cracking the triple ballot encryption. Available from `http://www.cs.princeton.edu/~appel/voting/Strauss-ThreeBallotCritique2v1.5.pdf`. Accessed on Aug. 12, 2007. (October 2006)
20. Jones, H., Juang, J., Belote, G.: Threeballot in the field. Term paper for MIT course 6.857. Available from `http://theory.csail.mit.edu/classes/6.857/projects/threeBallotPaper.pdf`.. Accessed on Aug. 12, 2007. (December 2006)
21. Henry, K., Stinson, D.R., Sui, J.: The effectiveness of receipt-based attacks on threeballot. Cryptology ePrint Archive, Report 2007/287 (2007) `http://eprint.iacr.org/`.