A framework for secure electronic voting

Stefan Popoveniuc and Poorvi L. Vora

The George Washington University
Department of Computer Science
poste,poorvi@gwu.edu

Abstract. We describes a framework in which to view the end-to-end-independently-verifiable (E2E) voting systems based on mixnets. We use the framework to invent new systems that combine front and back-ends from existing systems.

1 Introduction

Recent advances in cryptographic voting systems have led to the invention of several voting systems that are end-to-end-independently-verifiable (E2E) by the voter [4,8,13,6]. These systems provide receipts to the voter that can be used to ensure that her vote is counted, without revealing the vote. The mixnet-based E2E systems use different front-ends (for ballot preparation and the voting ceremony) and back-ends (for vote tallying and tally auditing), and most of the existing literature on these systems describes a single, complete mechanism that does not typically address the components separately. This paper describes the existing mixnet-based E2E voting systems in a single framework, and, by combining hitherto uncombined front and back-ends, presents new systems with hitherto unachieved properties.

As mentioned above, mixnet-based voting systems can be decomposed into two modules: the back-end and the front-end. The front-end is responsible for producing the inputs for the back-end: it defines what the ballots look like (either paper or electronic), what the voting ceremony is, how the receipt is produced, and how the voter interacts with the system after the casting process. It is responsible for the verifiable casting of votes. The back-end is the mixnet [9] itself, responsible for shuffling and eventually decrypting the ballots in an auditable fashion. It provides the verifiable tallying of the votes. With a number of options for front and back-ends, a jurisdiction can choose a back-end and a front-end that are most appropriate for a particular election. Some front-ends may be more usable, while others may protect privacy better. Some back-ends may be simpler to audit, while others yield higher performance. It is possible to use a single back-end and multiple front-ends for the same election. Having a range of choices gives freedom to the election authorities, who are directly responsible for choosing the technology most appropriate for their needs.

The contribution of this work is two-fold: on the one hand, it is a survey of front-ends and back-ends of the practical end-to-end mixnet based voting systems. On the other hand, it describes how they can be combined (see Table 1)

in ways that yield hitherto unachieved properties. We survey three back-ends: mixnets using public keys and onions [9], punchscanian mixnets using pre-committed paths and onions [13], and pointer-based mixnets [7], with pre-committed paths and no onions. The back-ends differ in the simplicity of the design, robustness and efficiency. We also survey four front-ends: visual cryptography [4], shuffling the order of the candidates (Prêt à Voter [8]), indirection-based encryption (PunchScan [13]), and a front-end that is an overlay to the current optical scan (Scantegrity II[6]). Proofs are outside the scope of this paper.

The rest of the paper presents related work, the front and back-ends with their advantages and disadvantages, and a framework that generalizes the front and back-ends. Within this framework, the client can choose the end-to-end voting solution that is best suited for her needs. Table 1 presents the existing systems and the contributions of this paper.

	Onion mixnets	**Punchscanian mixnet**	**Pointer mixnet**
Visual Crypto	✓	★	
Prêt à Voter	✓	★	★
PunchScan	★	✓	★
Scantegrity II	★	★	✓

Table 1. Recent mixnet-based voter-verifiable voting systems and the components they use. ✓ stands for work that has already been done while ★ represents new concepts proposed in the current work.

2 Previous Work

In Scratch&Vote [1], Adida and Rivest explain how their homomorphic scheme would work with two types of front-ends (Prêt à Voter and PunchScan). Van de Graaf [14] notices similarities between PunchScan and Prêt à Voter and describes a detailed mechanism that allows the use of a Prêt à Voter front-end with a PunchScan back-end. Lundin [11] investigates from a much larger perspective, all the aspects of voting, including registration, election method, election mechanics, election management, transfer methods, etc. Chaum [2] suggests writing a common XML format for representing the results of the scanning of PunchScan and Prêt à Voter ballots to allow the interchange of scanning technologies.

Our work is general, presenting the advantages of various ballot styles, and suggesting general ways of connecting them to mixnet based back-ends that have distinct properties. At the same time, our work provides details about each possible combination of front and back ends.

3 The Front-end

The front-end represents the manner in which the ballots are presented to the voters and the voter's interaction with the system. In the voting protocol, the front-end represents a voter-verifiable encryption of the vote; the receipt bears the encrypted vote. The ballots presented in this work fall in two categories. The first category is that of symmetric ballots: those that have two parts, each of which is sufficient to recover the vote (CVV– Sect. 3.4 and PunchScan– Sect. 3.5). On visual examination of the two parts appropriately laid out, the voter is able to visually confirm that each part bears the encryption of her vote. The second category is that of asymmetric ballots: those that have two parts, of which a specific one is needed to recover the vote (Prêt à Voter– Sect. 3.6 and Scantegrity II– Sect. 3.7). In this category too, the voter visually examines both parts of the ballot laid out in a particular manner (side by side) to confirm that the encrypted vote represents her ballot. Scantegrity II requires an indirection not present in other schemes, we discuss this in more detail later.

3.1 The General Receipt

We now describe more formally a receipt. In all the voting systems we study in this paper, the voter gets a receipt of serial number $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of all serial numbers. A vote is $v \in \mathcal{V}$, where $\mathcal{V}$ is the set of all possible votes. $\mathcal{E}$ is the encryption cipher used to generate the receipt, $\mathcal{E} = (\mathcal{V}, \mathcal{R}, \mathcal{K}, E, D)$, where $\mathcal{R}$ is the space of all ciphertexts for $\mathcal{E}$; $\mathcal{K}$ the keyspace, and $E : \mathcal{K} \times \mathcal{V} \to \mathcal{R}$ and $D : \mathcal{K} \times \mathcal{R} \to \mathcal{V}$ the encryption and decryption functions respectively. Finally, a function $f : \mathcal{S} \to \mathcal{K}$ provides the association between the key and the receipt. Note here that the encryption schemes used are deterministic; serial numbers are unique, keys are rarely, if ever, reused, and the encryption cipher is a private-key encryption, though f may involve the use of public key encryption. Using this notation, the receipt obtained by a voter in a correct instance of the voting protocol is the triplet $(s; x; E(f(s), v))$ where v is the vote, and x represents any other information, such as onions and commitments made by the voting system. This receipt is the only vote-related information to enter the voting system, and the tally is constructed from this information. In general, we will represent the receipt by the triplet (s, x, r), where s is the serial number, x any additional information on the receipt (for example, strings claimed to be onions by the system), and r the purported encrypted vote. Note that we are not always assured that $r = E(f(s), v)$ and that x is correctly formed.

3.2 The General Printing Audit

All front-ends discussed here are based on paper ballots; this enables the voter to retain a physical artefact of the voting process (a paper receipt), and provides some resemblance to voting with paper ballots, a process familiar to most voters. (We do not claim that all the E2E voter-verifiable voting systems are as easy to use as regular paper-ballot voting, we simply indicate that the choice of a

paper receipt, as opposed to any other type of receipt, has not been arbitrary). At some point, hence, the ballots and/or receipts need to be printed. Verifying the correctness of the printed ballots is referred to as a printing audit, and is carried out either before, during or after the election.

A correct printing provides the following. (a) *A Correct Encryption.* In systems such as Prêt à Voter, where the voter encrypts the vote, it enables the voter to do so correctly by simply filling up the ballot. In systems such as the visual encryption based system of Chaum, it simply prints the correct encryption. (b) *Correct Decryption.* It enables the system to correctly decrypt the vote. There are two possibilities for the mixnet-based decryption. First, the key may be referenced through s–either through a lookup table indexed by s, or by performing function f on s. In this case, a printing audit checks that the key used for encryption is indeed $f(s)$. Second, the receipt may bear additional information in the form of an onion that is used by successive mixes to effectively decrypt the receipt using the correct key. In this case, the printing audit checks that the key represented by the onion(s) is that used for encrypting the vote.

The most general printing audit allows each voter to choose two ballots, one to vote with, and the other one to spoil and verify. Because the choice of which ballot to spoil is made at random by the voters, the probability that an incorrect printing is undetected drops exponentially as the number of misprinted ballots increases. Variations on this method are possible, for example the voter does not actively spoil a ballot, as it becomes a natural result of the voting process: some voters simply make mistakes when filling them in, spoil them, and ask for a second ballot; the spoilt ones may be checked by the election authorities. Unfilled ballots may be checked at the end of an election. The printing of ballots made of two symmetric parts may also be audited by examining only the ballot half chosen for a receipt. Because the system cannot predict beforehand which half will be chosen, the probability that an incorrect printing is undetected decreases exponentially with the number of incorrect ballots. Leftover ballots can also be used after the election to perform the printing audit.

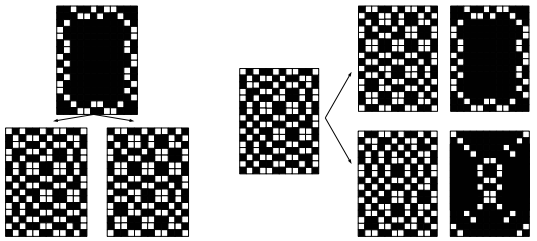
3.3 Types of Receipts

After all ballots are cast, the voting system makes available all receipts on a public bulletin board. The voter can check that the triplet (s, x, r) is among these. If she notices a discrepancy, she can file a complaint. There are two types of receipts the voter can get: proof receipts and indication receipts. A proof receipt is one that is produced and "signed" by the voting system itself (or by an election official). When a voter holds a proof receipt that is inconsistent with the information on the public bulletin board, and the digital signature scheme used is assumed secure, it is sufficient evidence that the bulletin board contains erroneous information. This, in turn, is an irrefutable indication that something went wrong with the election. On the other hand, indication receipts provide only a hint that something might have gone wrong, but additional evidence needs to be provided in order to prove that something went wrong. Indication receipts can be produced by voters themselves and are not "signed" by any election authority.

We now describe in more detail the "voting ceremony" for several front-ends, and indicate how the front-ends are specific instances of the general description above. The voting ceremony consists of the specific steps a voter needs to take to cast a ballot, and, eventually, to verify later that the ballot was printed correctly and recorded as cast. We will describe the receipt; in particular, we will describe the encryption process. We will intentionally not describe the value in x, as that depends on the back-end.

3.4 Ballots using visual cryptography

Chaum [3] describes a ballot made of multiple parts, such that the combination of all parts makes the text readable; but no information is revealed about the vote when only a subset is available. The first instantiation of this idea used visual cryptography [12].



(a) A ballot containing a vote for ZERO.

(b) One layer can represent with equal probability a vote for either ZERO or X, depending on what the other layer is

Fig. 1. A Sample Ballot using Visual Cryptography

A detailed explanation on how the layers are built can be found in [15]. On the top layer, the odd pixels are generated pseudo-randomly, while on the bottom layer, the even pixels are generated pseudo-randomly. The rest of the pixels are generated in a way that constructs the clear text image of the ballot only when the two layers are overlaid. The voter is able to read her vote when the two pages are overlaid (see Fig. 1(a)), but when looking at a single layer, no information is leaked about the voter's choice (see Fig. 1(b)). Because of the size of the font and the redundancy in the glyphs, the vote can be later recovered from a single layer, unambiguously.

A more formal description is as follows. A correct receipt is of the form $(s, x, v \oplus k_a)$ where $\oplus$ denotes bitwise XOR, k_a represents the key for the layer corresponding to the receipt. The other layer is encrypted using key $k_{\bar{a}}$. The keys are generated using different seeds for a pseudo-random number generator;

$k_a = f_a(s) = F(Sign(s, p_a))$, where $Sign(s, a)$ is the digital signature of the serial number using the private key of the polling machine that corresponds to layer a, and F the public pseudo-random number generator. If the two receipts are (s_a, x_a, r_a) and $(s_{\bar{a}}, x_{\bar{a}}, r_{\bar{a}})$, because of the manner in which the bits in each layer are presented, the voter can visually check that $r_a \oplus r_{\bar{a}} = v$, $s_a = s_{\bar{a}}$. The information in x depends on the back-end used.

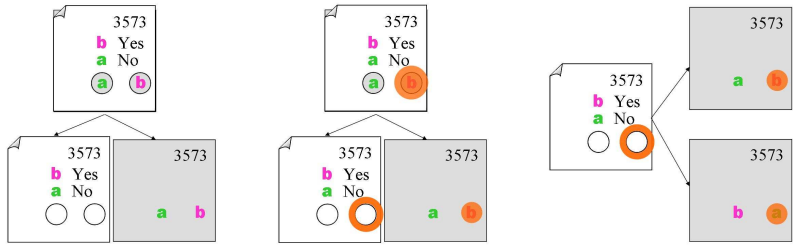
The voting ceremony On election day, the voters go to their assigned polling places, authenticate themselves as legitimate voters and uses a touch screen to make the desired selections. When finished, the computer prints the two layers, the voter checks that, when the two layers are overlaid, her vote is shown. The voter choose one of the layers as a receipt and watches the other one being destroyed. The computer prints additional information on the receipt, that allows to check that the pseudo-random pixels on the chosen layers have been constructed correctly. A digital signature is also printed. After election day, any voter can go to the election authority web site, enter the serial number for her ballot, check that the ballot is there and that it matches the page she possesses: the pixelized image and the strings on the receipt should be the same as the ones posted on the web site.

Advantage and disadvantages The advantages of this approach are the high degree of generality (it can accommodate any type of contest, including write-ins), the receipt is created automatically, it allows a fixed order of candidates, it offers excellent privacy (except the fact that the voting machines knows the clear text votes, and can thus compute an independent total), and there is no need for a strict chain of custody. The disadvantages are: the voters are not familiar with the receipt interface, the order of events must be precise, the alignment of pixels is difficult, the receipt is difficult to check by the voter, it is very difficult to implement in practice (one reason is the alignment of the two pages), it does not accommodate disabled voters, it does not allow manual recounts, the cost is very high (because one machine per booth is needed), and the administration of the system is difficult.

3.5 Ballot with indirection

To allow the same separation of information as in the previous case, the following technique can be used: on one page each candidate is associated with a random symbol; on another page the same set of symbols appears in some random order. For convenience the two pages can be overlayed, with the top page having holes and the symbols on the bottom page being visible through the holes. (see Fig. 2(a)). This technique was first proposed in PunchScan [5] and therefore this style of ballot is known as a PunchScan ballot.

In PunchScan, the voter uses a dauber to mark the selection of candidates. The diameter of the ink disc is greater than the diameter of the hole punched through the top page, therefore the dauber leaves a mark on both the top and



(a) A sample PunchScan ballot. When the two pages are overlayed, the symbols on the bottom page are visible through the holes.

(b) A voted ballot. Looking at each layer individually, one cannot say that the mark is for "Yes" or for "No".

(c) Given only one layer of the ballot, the marks on that layer are equally likely to represent a vote for any candidate.

Fig. 2. PunchScan's ballot

the bottom ballot pages. Fig. 2(b) contains a ballot voted for "Yes". Because the order of the symbols on the two pages of a ballot is different (and independently and uniformly distributed), one cannot determine which mark is for which candidate by viewing only one voted page. The association of candidates with symbols, and the order of the symbols on the bottom page, can be uniformly random, or pseudorandom.

Thus, in PunchScan, the receipt is of the form: $(s, x, E(f(s), v))$ where E is viewed as a permutation of the plaintext space (all encryptions are trivially permutations of plaintext space) composed of two distinct permutations: the first the association of candidate choice with dummy variables on the front (viewed as a map of candidates in canonical order, such as alphabetical order, to dummy variables in canonical order) and the second the association of dummy variables with positions on the back (again the map can be between canonical orderings). With abuse of notation, using the same notation for the key and the encryption function it represents, the key $f(s) = \sigma_a(s) \circ \sigma_{\bar{a}}(s)$ is the composition of two permutations, each a well-defined function of s.

The voting ceremony On election day, the voters go to their assigned polling places, authenticate themselves as legitimate voters, and before seeing the ballot, the voters commit to which page to keep as a receipt. In the privacy of a booth, the voter marks the hole that contains the symbol associated with her favorite candidate, and, when done, scans the page chosen in the first step, keeps it and shreds the other one. After election day, any voter can go to the election authority web site, enter the serial number for her ballot, check that the ballot is there and that it accurately resembles the page she possesses: her marks are

recorded correctly and the order of the symbols on her receipt is the same as the order posted.

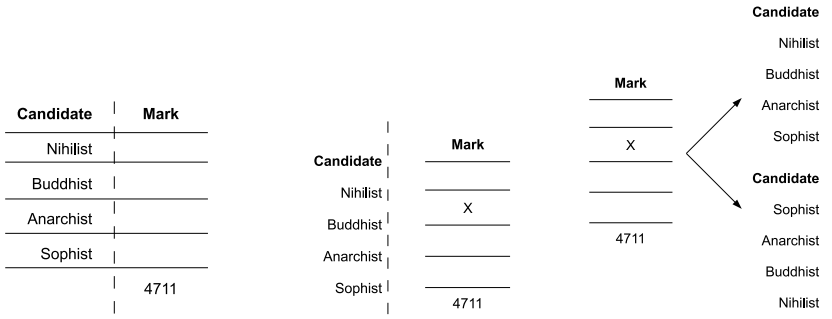
Advantages and disadvantages The advantages of this method are: the receipt is created automatically and is easily checkable by the voter, it allows a fixed order of the candidates on the ballot, it offers excellent privacy (the scanner does not know the clear text votes), the cost is low, the dispute resolution is easy and it accommodates disabled voters (see the PunchScan website for a brief description of such capability, which also follows for Prêt à Voter; this capability is, however, not described in detail yet, a paper is in preparation). The disadvantages are: it does not accommodate write-ins (but it accommodates most types of contests), the voters are not familiar with the voting interface (the indirection may cause usability problems), the voting machine cannot provide an independent tally, it needs a strict chain of custody before the ballots reach the voters and it does not allow a manual recount.

3.6 Permuting Candidate Order

Prêt à Voter [8] proposes a simplification of the two-part visual cryptography ballot presented in Sect. 3.4, see Fig. 3(a). The ballot is printed on a single page of normal paper, with the names of the candidates on the left and the places to mark on the right. A voter makes a mark next to her favorite candidate (see Fig. 3(b)). The names of the candidates are permuted on each ballot and when the left part is separated from the right part, the marks on the right are no longer associated with candidates (see Fig. 3(c)). This ballot style is an example of a ballot that has two parts (left and right) but the information is distributed asymmetrically in the two parts. Thus the Prêt ã Voter receipt is of the form $(s, x, E(f(s), v))$ where $f(s)$ is a permutation.

The voting ceremony On election day, the voters go to their assigned polling places, authenticate themselves as legitimate voters, gets two ballots from the election officials and chooses one to audit (printing correctness check) and one to use for voting. In the privacy of the voting booth, the voter makes an X on the right side of the ballot, next to her favorite candidate. The voter separates the list of candidates (on the left) from the marks (on the right), destroys the left side and scans the right side. The marks are recorded and made public. The scanned side is kept by the voter and anytime after election day, the voter can go to the election authority web site, enter the serial number for her ballot, check that the ballot is there and that it accurately resembles the page she possesses.

Advantages and disadvantages The advantages of this method are: the voters are somewhat familiar with the interface,the receipt is created automatically and is easily checkable by the voters, it accommodates disabled voters, and it offers excellent privacy (the scanner does not know the clear text votes) at a low



(a) A sample Prêt à Voter ballot. A permuted list of candidates is on the left

(b) A voted Prêt à Voter ballot. When the right side is separated from the left side the mark is not a clear vote anymore

(c) Given only the right side, the mark is equally likely to represent a vote for any candidate.

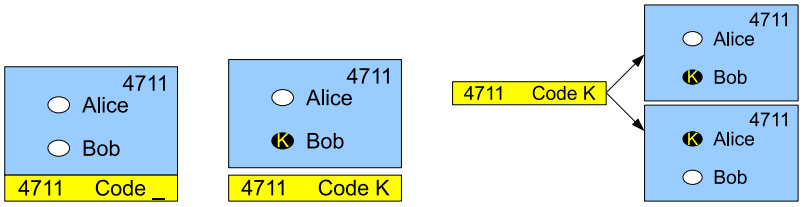
Fig. 3. Prêt à Voter ballot

cost. The disadvantages of the method are: it does not accommodate write-ins (but it accommodates most types of contests), it does not allow a fixed order of candidates, the voting machine cannot provide an independent tally, it needs a strict chain of custody before the ballots reach the voters, and it does not allow for a manual recount.

3.7 Standard optical scan ballot, encoded receipt

Scantegrity [7] and Scantegrity II [6] addresses the usability concerns of Punch-Scan while keeping the order of candidates fixed on all ballots. A Scantegrity ballot contains two asymmetrical parts, but because the two parts are never separated, it is printed on a normal piece of paper that will not be divided in any way. One part of the ballot is a normal optical scan ballot, which can be scanned and used by any certified optical scan voting system. The other part is a set of confirmation numbers associated with the candidates (e.g. printed next to the candidates). The association is different on each ballot. See Fig. 4 for a sample Scantegrity II ballot.

The difference between Scantegrity and Scantegrity II is that the voter only gets the confirmation numbers for the candidates she is choosing in Scantegrity II, while in Scantegrity the voter is able to see the confirmation numbers for all the candidates. The immediate benefit is that is in the dispute resolution process: voters that claim to have their ballots registered improperly must provide the confirmation numbers, which are random and hard to predict. The election authority can then discard the complains that contain confirmation numbers that do not appear on the indicated ballot and race.



(a) A sample Scantegrity ballot, just like a regular optical scan ballot.

(b) A voted Scantegrity ballot. When the oval is filled in, a confirmation number appears.

(c) The symbols on the receipt may correspond to any candidate.

Fig. 4. Scantegrity Ballot: Blue is the ballot form, and Yellow is the receipt. Typically, the indication receipt may contain the serial number and the confirmation number.

The voting ceremony On election day, the voters go to their assigned polling places, authenticate themselves as legitimate voters, gets two ballots from the election officials and chooses one to audit (printing correctness check) and one to use for voting. In the privacy of the voting booth, the voter marks the ballot as a normal optical scan ballot. On a separate piece of paper, she writes down the confirmation numbers associated with the voted candidates, tears off the bottom part of the ballot, with the serial number of the ballot on it and keeps it. The ballot is scanned by a regular optical scanner. After election day, any voter can go to the election authority web site, enter the serial number and check that the symbols she wrote down are on the web site.

Because the receipt the voter gets is an indication receipt (as opposed to a proof receipt), if the voter sees on the web site a different set of symbols then the ones on her own piece of paper, she has to have a way of challenging the records on the bulletin board. Depending on the length and unpredictability of the confirmation numbers, a set of dispute resolution techniques are possible; see [6] and [7] for details.

Advantages and disadvantages The advantages of Scantegrity II are: the voters are highly familiar with the interface, it is highly usable by the election officials, the voters can easily check their receipt, the dispute resolution process is easy, it allows for a fixed order of the candidates, it accommodates disabled voters, the voting machine can compute an independent tally, it allows a manual recount, the cost is very low, and it is very easy to administrate. The disadvantages of the method are: it does not accommodate write-ins, the voting machine knows the clear text votes and it needs a strict chain of custody after the voters mark their ballots to protect privacy.

Table 2 summarizes the advantages and disadvantages of the four types of front-ends.

	Visual Crypto	**PunchScan**	**Prêt à Voter**	**Scantegrity II**
Generality	Any type of contest	Most practical contests	Most practical contests	Most practical contests
Familiarity with the interface	Low	Low	Medium	High
Receipt Creation	Automatic	Automatic	Automatic	Requires voter effort
Voter verification	Difficult	Easy	Easy	Easy
Supports write-ins	Yes	No	No	No
Fixed order of the candidates	Yes	Yes	No	Yes
Ease of implementation in practice	Difficult	Easy	Easy	Easy
Accommodates disabled voters	No	Yes	Yes	Yes
The voting machine knows the clear text votes and can compute the tally	Yes	No	No	Yes
Chain of custody to protect privacy	No	Before the ballot reaches the voter	Before the ballot reaches the voter	After the ballot is marked
Manual recount	No	No	No	Yes
Cost	High	Low	Low	Low
Ease of administration	Difficult	Moderate	Moderate	Easy
Ease of dispute resolution	Easy	Easy	Easy	Easy

Table 2. Evaluation of various types of ballots

4 The Back-end

The back-end is responsible for producing clear text ballots from the encrypted receipts produced during the voting ceremony. The process has to be fully auditable by anyone, yielding universal verifiability, while preserving the secrecy of the votes. In two of the three cases presented here, the back-end is also responsible for creating the blank ballots initially.

Three main techniques are presented:

– classical mixnets using public keys and onions. The path followed by a vote is determined on-the-fly.

– punchscanian mixnet using pre-established and committed paths and onions.
– pointer-based mixnet with pre-established paths and no onions.

We briefly describe each back-end and suggest new simple ways to connect the back-end with a front-end described in Sect. 3 with which it was not associated before, as shown in Table 1

4.1 Traditional mixnets

Mixnets have been classically associated with onion routing because the payload can be viewed as an onion, with multiple layers of encryption; each mix strips off one of the layers. Besides the onion, the payload also contains a value (a ballot in the case of voting systems). After removing one layer of encryption from the onion, a mix finds a seed (sometimes called a germ) that is used to transform the value in the payload. This way, the output value is uncorrelated with the input value.

In general, the payload is a pair $(\texttt{Onion}, \texttt{Ballot})$. Thus, when the back-end is a traditional mixnet, the value of x for all the front-ends contains the onion. The serial number from the receipt is stripped after voters have checked the presence of the receipt on the bulletin board, and the triplet (s, x, r) is reduced to the pair (x, r), referred to as $(\texttt{Onion}, \texttt{Ballot})$. (`Ballot` hence represents the encrypted vote). For a particular mix j and a particular input-output pair, the input is $\texttt{Payload}_j = (\texttt{Onion}_j, \texttt{Ballot}_j)$ and the output is $\texttt{Payload}_{j+1} = (\texttt{Onion}_{j+1}, \texttt{Ballot}_{j+1})$, The relation between the two onions is

$$\texttt{Onion}_j = Enc_{(\texttt{PublicKeyOfMix}_j)}(seed_j, \texttt{Onion}_{j+1}) \tag{1}$$

where $Enc_{(\texttt{PublicKeyOfMix}_j)}$ represents encryption with the public key of mix j and the comma represents concatenation. The $(j+1)^{th}$ onion is obtained by decrypting the j onion and removing $seed_j$: $\texttt{Onion}_j = Dec_{(\texttt{PrivateKeyOfMix}_j)}(\texttt{Onion}_{j+1}) \setminus seed_j$, where $\setminus$ denotes removal from a string. The relation between the input and the output ballot is

$$\texttt{Ballot}_{j+1} = F_j(seed_j)(\texttt{Ballot}_j) \tag{2}$$

where $F_j(seed_j) \in G, \forall j$ for group G with operation $\odot$, and F_j is a public function. An important aspect is that the Onion and the Ballot have to travel together through the mix. Thus $\bigodot_j F_j(seed_j)(:) = D(f(s), :)$ decrypts the encrypted receipt.

A mixnet may be audited by either providing a zero-knowledge proof of correctness or using a randomized partial checking (RPC) technique [10]. In the latter case, the mix is required to reveal $seed_j$ for a significant fraction of the inputs (or outputs). Having the seed, the auditor (sometimes called the challenger or verifier) can check Eq. 1 and Eq. 2 for all the revealed input-output pairs.

The traditional mixnet is used as the back-end of the voting scheme using visual cryptography proposed by Chaum, and by Prêt à Voter. In the scheme of

Chaum, G is the set of all bitwise XORs acting on n-bit strings, and $F_j(seed_j)$ corresponds to a bitwise XOR using the pseudo-random string generated using $seed_j$. For $z_i \in G$ corresponding to XOR with string y_i, $z_1 \odot z_2 = z_3$ where $y_3 = y_1 \oplus y_2$, with $\oplus$ corresponding to bitwise XOR. In this case, $v = \bigodot_j F_j(seed_j)(r) = r \oplus k_a$. That is, the composition of the processing of individual mixnet entities corresponds to the bitwise XOR of the receipt bitstring with the bitstring used to encrypt it. In Prêt à Voter, G is the set of permutations on sets of size c, the number of candidates, $\odot$ is permutation-composition and $F = P \circ h$ where h is a one way function and P is a function that generates a permutation on a set of size c given a seed. $v = \bigodot_j F_j(seed_j)(r) = D(f(s), r)$ is the inverse of the permutation represented by $f(s)$, applied to receipt r. That is, the composition of the processing of individual mixnet entities corresponds to the inverse of the permutation used for encryption.

Advantages and disadvantages The advantages of onion mixnets are their truly distributed nature, support for dynamic paths and the possibility of setting up the system before the details of the elections are known. The disadvantage is low efficiency, both when processing the ballots and during the audit process.

We now describe the combination of the onion mixnet backend with the front-ends of PunchScan and Scantegrity.

PunchScan ballot with onion mixnet Recall that $\bigodot_j F_j(seed_j)(:) = D(f(s), :)$ for the onion mixnet, and $f(s) = \sigma_a(s) \circ \sigma_{\bar{a}}(s)$ for the PunchScan ballot. Hence, for a PunchScan front-end and an onion mixnet back-end, the ballot needs an onion, which will be contained in x. The onion contains seeds which will generate permutations whose composition will invert the encrypting permutation $\sigma_a(s) \circ \sigma_{\bar{a}}(s)$. Thus, G is the set of permutations on a set of size c, and $\odot$ is the composition of permuations. In order to generate the ballot, the ballot manufacturing facility produces a pseudorandom permutation, say σ, as the composition of several pseudorandom permutations σ_j (as many as there are mixes) each generated from a random seed. σ is decomposed into two permutations to be used for the two pages of the ballot, by choosing one of the permutations uniformly at random. The seeds for the σ_j are buried into the onion, which is part of x. Decryption involves generating the corresponding σ_j, and performing its inverse. That is, $F_j(seed_j) = \sigma_j^{-1}$.

Scantegrity ballot with onion mixnet The best dispute resolution properties for the indication receipts of Scantegrity are obtained when the set of confirmation numbers is large, when a confirmation number is used exactly once among all the ballots, and when the probability of guessing a valid confirmation number of any candidate in a receipt is low. When the Scantegrity ballot is used with onion mixnet, a permutation of a canonical ordering of all candidates over all ballots for each race is used to generate the confirmation numbers. That is, the permutation acts on cN values for a c-candidate race and N ballots,

ensuring distinct confirmation numbers on each ballot. The permutation is generated as for the PunchScan ballot, by first constructing as many pseudorandom permutations as there are mixes, and then composing them into a single permutation. The decryption is also similar thereafter; each mix applies the inverse of the pseudorandom permutation corresponding to the seed the mix obtains. The seeds may not be distinct for distinct ballots.

4.2 Punchscanian mixnet

A punchscanian mixnet [13] has been viewed as an integral part of PunchScan itself, however we describe how it may be used with other front-ends, after first providing a brief overview. The path through a punchscanian mixnet is fixed a-priori and commitments to it are published a priori; the paths and permutations are pseudorandomly generated. The advantage of having pre-set paths is that the onions do not have to be part of the payload anymore. The notion of an onion gets degraded to a chaining of secret seeds, which are fixed along the path. The payload becomes only the ballot itself, which carries the encrypted selection of the voter. In a punchscanian mixnet $\texttt{Payload}_j = (\texttt{Ballot}_j)$ and there is no relation between the degraded onions; that is, there is variable x in the receipt. Because the paths are pre-committed to, and only the mixnet knows the seeds, the mixnet itself produces the ballot (as opposed to the voting machine producing the onions). In this setting, the mixnet is a single entity, and not composed of several entities; however, this single entity may be split among several using standard secret sharing approaches. Assuming that RPC is used to audit the mixing, the single entity consists of two for the purposes of the audit and hence:

$$\texttt{Ballot}_j = F(seed_{j1}) \odot F(seed_{j2})(\texttt{Ballot}_{j-1}) \tag{3}$$

where F is a public function. Having access to only one of the three elements in the equation does not leak any information about the other two. The commitments to the seeds can be independent, or can be blended into the commitments to the paths. While traveling through the mixnet, the ballot is transformed according to Eq. 2.

After the ballots are produced, they are publicly committed to. To ensure that the produced ballots are consistent with the seeds used to generate them, a significant fraction of the ballots are randomly chosen to be opened, and Eq. 3 is checked for all of them. With high probability, the ballots that were not opened are also consistent with the paths and the seeds actually used for the decryption. The ballots that are opened need not be printed on paper. To ensure that the ballots that survived the audit are printed correctly, another audit (called a printing audit) has to be performed. If all the ballots are initially printed, then the printing audit can be combined with the mixnet correctness audit.

Advantages and disadvantages The major advantage is the high efficiency. Millions of ballots can be tallied in minutes. The disadvantages are the central

nature of the authority, and the need to know the details of the election before setting it up.

We now describe the use of the punchscanian mixnet with the other front-ends. The essential approach is to absorb the onions into the mixnet and to precommit to both onions and paths. Hence the onions used for the onion mixnet can also be used for the punchscanian mixnet, with two differences: the onions will not be carried with the ballots, and using more than two mixes serves no purpose.

Visual cryptography with punchscanian mixnets Two PunchScan mixnets are constructed, one for each layer. For each layer of each vote, the pseudorandom component contributed by each mix is committed to. along with the path the ballot layer will travel. The voter's choice of receipt determines the mixnet that will be used to decrypt her vote. From the chosen layer, the pixels that are generated pseudo-randomly are discarded and the other pixels are run through the corresponding punchscanian mixnet.

Prêt à Voter with punchscanian mixnets The onions of the Prêt à Voter ballot are committed to along with the path the ballot will take, at the mixnet. The onion is not part of the payload.

Scantegrity with punchscanian mixnets The same procedure is followed as for Scantegrity with the onion mixnet, except that the onion is not carried with the ballot, and is committed to in the appropriate mixes along with the pre-computed path the ballot will take.

4.3 Pointer-based mixnets, or mixnets with no explicit group operation

In its traditional form, the payload of the mix consisted of an onion and a ballot. A first simplification step, as in the punchscanian mixnet, was to separate the two, glue the onions to the mixnet and have only the ballot travel. A second step is to remove the onion altogether. The onion does not vanish from a conceptual perspective, but it is absorbed into the other operation that the mixnet is doing: mixing. This is because both the shuffling and the decryption can be viewed as permutations when the number of messages is small, and can be combined into one essential permutation. Another way of viewing this is to consider the vote for each candidate in a ballot (a mark or no mark) as a separate entity that travels independently through the mixnet (as opposed to being part of a ballot or a contest).

Let N be the number of ballots in an election and let c be the number of candidates on a ballot. Consider three tables: R (stands for receipt values) contains coded votes; T (stands for tallies and results) contains clear text votes that are countable by anyone; D (stands for decrypt) connects R with T. R is a

matrix with N rows and c columns, each row represents a ballot. R is a matrix with c rows and N columns, each row represents a candidate. An element (i, j) is either marked or not marked in R and T, a mark corresponds to a vote for the candidate. D is a blob with $N \times c$ elements (the number of rows and columns is irrelevant). Fig. 5 gives an example of the three tables for an election with six ballots and two candidates.

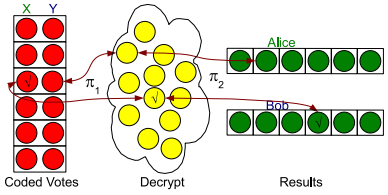


Fig. 5. Pointer-based mixnet

The tables are connected by two permutations, π_1 and π_2. π_1 connects R with D: $D_k = R_{\pi_1(k)}$, where k is some canonical representation of (i, j); for example, $k = (c-1)i + j$. π_2 similarly connects D with T: $T_k = D_{\pi_2(k)}$. These permutations are constrained to return a mark for a particular candidate in R to a mark for the same candidate in T.

For two candidates, the properties of the permutation may be formalized as follows: let $\pi_1 : [0, 1, ..., N \times c - 1] \rightarrow [0, 1, ..., N \times c - 1]$ be bijective and let $\pi_2 : [0, 1, ..., N \times c - 1] \rightarrow [0, 1, ..., N \times c - 1]$ be bijective such that no two elements belonging to the same ballot initially (in the same row in the initial set) are mapped to two elements belonging to the same candidate (the same row in the final set):

$$\forall i, j, i \neq j \; having \; [i/c] = [j/c] \Rightarrow [\pi_2(\pi_1(i))/b] \neq [\pi_2(\pi_1(j))/b] \tag{4}$$

where $[x]$ represents the greatest integer less than or equal to x. Note that no group operation is performed (such as modulo addition or permutation composition). For $c > 2$, the condition requires that the remainder on division by c (the candidate for a mark or nomark) be preserved; because the rows and columns are reversed, one may then wish to view table T canonically as listed column by column, and R row by row.

The audit checks that one of the two properties hold: $D_i = R_{\pi_1(i)}$ or $D_i = T_{\pi_1^{-1}(i)}$ and that the properties of the two permutations π_1 and π_2 hold; more precisely it is statistically checked that both π_1 and π_2 are injective functions and that Eq. 4 holds. Because the voting system cannot predict which property will be checks, a successful audit implies that both properties hold with high probability.

Advantages and disadvantages The advantages of pointer mixnets are their efficiency and the possibility of setting up the system before the details of the election are known. Their disadvantage is the central nature of the authority.

Visual Crypto with Pointer mixnet In order to use the pointer mixnet with the visual crypto front-end, one would need to treat each pixel as a candidate, and the resulting system would be quite inefficient.

Prêt à Voter with pointer mixnet Each candidate, and thus mark position, is treated independently and its path and ending point in the table with the clear votes are committed to, just as with Scantegrity.

PunchScan with pointer mixnet Each position that can be marked is treated independently and its path and ending point in the clear vote table is committed to, just as with Scantegrity.

Table 4.3 summarizes the advantages and disadvantages of the three types of mixnet-based decryption mechanisms.

	Onion mixnets	**Punchscanian mixnet**	**Pointer mixnet**
Distributed authority	Yes	No	No
Paths	Dynamic	Static	Static
Constructs the ballot	No	Yes	Yes
Efficiency	Low	High	Medium
Lazy ballot style	Yes	No	Yes
Cryptography used	Symmetric and asymmetric encryption, one way functions	Commitments	Commitments

Table 3. Properties of various mixnets

5 Conclusions

We have presented a unified view of four practical, end-to-end, voter-verifiable voting systems that have been proposed recently as monolithic blocks. We present a concrete separation between the way the ballot is presented and how the voters interact with the system (the front-end) and the way the ballots are decrypted and the tally is verified (the back-end). We present the properties of these front and back-ends, and describe simple ways to combine them. This gives great flexibility in the choice of a voting system for a particular jurisdiction that values some properties more then others (e.g. privacy more then usability). Our work opens a new way of looking at future voting systems, component-wise. Further

research can focus only on improving or changing a particular component of a voting system (e.g. back-end), as long as it can interact with the other component (e.g. front-end).

Acknowledgements

The authors would like to thank David Chaum, Jonathan Stanton, Aleks Esses, Rick Carback and Jeremy Clark for inspiring discussion;the anonymous reviewers, Geanina Popoveniuc, Gedare Bloom and Eugen Leontie for reviewing early versions of this work.

References

1. B. Adida and R. L. Rivest. Scratch & Vote: self-contained paper-based cryptographic voting. In *WPES '06: Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 29–40, New York, NY, USA, 2006. ACM Press.
2. D. Chaum. Private communications.
3. D. Chaum. U.S. patent 10348547 - Secret-ballot systems with voter-verifiable integrity, 2003.
4. D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, pages 38–47, January/February 2004.
5. D. Chaum. Recent results in electronic voting. In *Presentation at Frontiers in Electronic Elections (FEE 2005)*, Milan, Italy, September 2005. ECRYPT and ESORICS.
6. D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, and A. T. Sherman. Scantegrity ii: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *EVT'07: Proceedings of the USENIX/Accurate Electronic Voting Technology on USENIX/Accurate Electronic Voting Technology Workshop*. USENIX Association, 2008.
7. D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. T. Sherman, and P. Vora. Scantegrity: End-to-end voter verifiable optical-scan voting. *IEEE Security and Privacy*, May/June 2008.
8. D. Chaum, P. Y. A. Ryan, and S. Schneider. A practical voter-verifiable election scheme. In *In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, ESORICS, volume 3679 of Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.
9. D. L. Chaum. Untraceable electronic mail, return address, and digital pseudonym. *Communication of ACM*, February 1981.
10. M. Jakobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353, Berkeley, CA, USA, 2002. USENIX Association.
11. D. Lundin. Component based electronic voting systems. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2007)*, University of Ottawa, Canada, June 2007.
12. M. Naor and A. Shamir. Visual cryptography. *Lecture Notes in Computer Science LNCS*, 950:1–12, 1995.

13. S. Popoveniuc and B. Hosp. An introduction to PunchScan. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, Robinson College, Cambridge UK, June 2006.
14. J. van de Graaf. Merging Prêt à Voter and PunchScan. Cryptology ePrint Archive, Report 2007/269, 2007. `http://eprint.iacr.org/2007/269.pdf`.
15. P. L. Vora. David Chaum's voter verification using encrypted paper receipts, 2005.