

Scratch & Vote

Self-Contained Paper-Based Cryptographic Voting

Ben Adida
CSAIL, MIT
32 Vassar Street, Cambridge MA 02139
ben@mit.edu

Ronald L. Rivest
CSAIL, MIT
32 Vassar Street, Cambridge MA 02139
rivest@mit.edu

ABSTRACT

We present *Scratch & Vote* (S&V), a cryptographic voting system designed to minimize cost and complexity: (1) ballots are paper-based and can be printed using today's technology, (2) ballots are universally verifiable without election-official intervention, and (3) tallying requires only one trustee decryption per race, thanks to homomorphic aggregation.

Scratch & Vote combines the multi-candidate election techniques of Baudron et al. with the ballot-casting simplicity of Chaum and Ryan's paper-based techniques. In addition, S&V allows each voter to participate directly in the audit process on election day, *prior* to casting their own ballot.

Categories and Subject Descriptors

H.4.m [Information Systems Applications]: Miscellaneous; J.1 [Administrative Data Processing]: Government; K.4.1 [Computers and Society]: Public Policy Issues

General Terms

Security, Human Factors, Verification

Keywords

Cryptographic Voting, Scratch Surface, Barcode, Paper Ballot, Homomorphic Tallying, Paillier Cryptosystem

1. INTRODUCTION

Cryptography can reconcile public auditability and ballot secrecy in voting. Votes are encrypted and posted on a public bulletin board, along with the voter's name (or voter identification number) in plaintext. Everyone can see that Alice has voted, though, of course, not what she voted for. The encrypted votes are then anonymized and tallied using publicly verifiable techniques.

Most cryptographic voting schemes require complex equipment and auditing. A certain degree of complexity is un-

avoidable, as the functional goal of cryptographic voting is to run an election correctly *while trusting third parties as little as possible*. Unfortunately, this complexity often stands in the way of adoption. If it takes significant expertise to understand how a voting system functions, and if the operation of the system is particularly complex, election officials and the public may be reluctant to adopt it. The question, then, is how much can we simplify the voting process while retaining cryptographic verifiability?

Voting systems & scratch surfaces. In recent months, Arizona has proposed running a cash-prize lottery for all citizens who vote [27]. In response, a well-known online satirical publication jokingly proposed a "Scratch & Win" voting system [30]. Though our proposal, Scratch & Vote, uses scratch surfaces, it should not be confused with a game of chance. That said, we hope that, given the public's familiarity with scratch surfaces, our own use of them will help spark more widespread interest in the topic of cryptographic voting.

1.1 Our Proposal

We propose *Scratch & Vote* (S&V), a cryptographic voting method that provides public election auditability using simple, immediately deployable technology. S&V offers:

1. **Paper ballots:** ballot casting is entirely paper- and pen-based.
2. **Self-contained ballot auditing:** ballots contain all the necessary information for auditing; there is no need to interact with the election officials.
3. **Simple tallying:** ballots are tallied using homomorphic encrypted counters rather than mixnets. Anyone can easily verify the final tally, and election officials need only cooperate to decrypt a single tally ciphertext per race (or even a single tally ciphertext for the entire election, if it isn't too large.)

The voter experience is simple and mostly familiar:

- **Sign in:** Alice signs in and obtains a ballot with randomized candidate ordering. Election officials should not see this candidate ordering. The ballot is perforated along its vertical midline, with candidate names on the left half and corresponding scannable bubbles on the right. A 2D-barcode is positioned just below the checkboxes on the right. A scratch surface, labeled "void if scratched," is positioned just below the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CCS'06, October 30–November 3, 2006, Alexandria, Virginia, USA.
Copyright 2006 ACM 1-59593-556-8/06/0010 ...\$5.00.

barcode, and an additional perforation separates the scratch surface from the rest of the right half. (See Figure 1.)

- **Audit [optional]:** Alice may select a second ballot for auditing. She scratches off the scratch surface, hands the now void ballot to a helper organization on the premises – i.e. a political party or activist organization she trusts – and receives confirmation that the ballot was well-formed. This gives Alice confidence that her first ballot is also well-formed: if enough voters perform the audit, even a handful of bad ballots will be quickly detected. (See Figure 2.)
- **Make selection:** Alice steps into the isolation booth to make and review her selection.
- **Detach ballot halves:** Alice separates the two halves of the ballot. A receptacle is available for her to discard her left ballot half. Note that this discarded half carries no identifying information, only a randomized candidate ordering. (See Figure 3.)
- **Casting:** Alice presents the right half of her ballot to an election official, who inspects the scratch surface to ensure it is intact. The official then detaches the scratch surface and discards it in sight of all observers, including Alice herself. Alice then feeds what remains of her ballot – the checkmark and barcode – into a scanner. This is effectively her encrypted ballot. Alice takes it home with her as a receipt. (See Figure 4.)
- **Verification:** Alice can log on to the election web site to verify that her ballot – including checkbox and barcode – has been correctly uploaded to the bulletin board. If it hasn't, she can complain with receipt in hand. Alice can also verify the entire tally process, including the aggregation of all ballots into a single encrypted tally, and the verifiable decryption performed by election officials. (See Figure 5.)

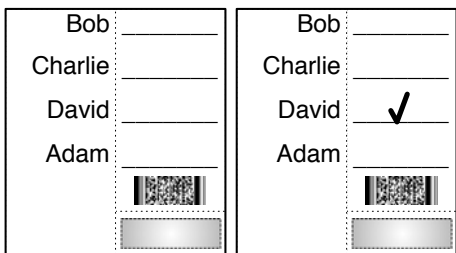


Figure 1: A Scratch & Vote ballot, before and after Alice makes her selection. The ballot is perforated along two axes: down the vertical midline, and between the barcode and scratch surface on the right half.

So far, this description uses the Scratch & Vote adaptation of the Ryan Prêt-a-Voter ballot. In section 5, we also show how to achieve the same features based on a new ballot inspired by Chaum’s Punchscan [20], whose physical layout accommodates numerous races more easily. We consider the threat model and certain extensions to our proposal that further increase practicality.

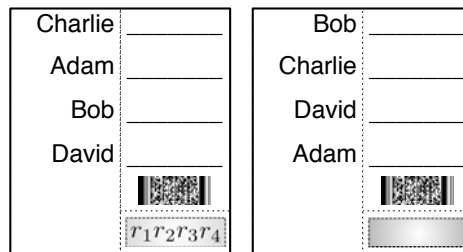


Figure 2: Auditing the S&V ballot. Alice receives two ballots and chooses to audit one at random, removing its scratch surface. In this diagram, Alice selects the ballot on the left. Alice’s chosen helper organization then scans the barcode, reads the randomization data r_1, r_2, r_3, r_4 (one value per candidate) previously hidden under the scratch surface, and confirms that the ballot is correctly formed. Alice then votes with the second, pristine ballot.

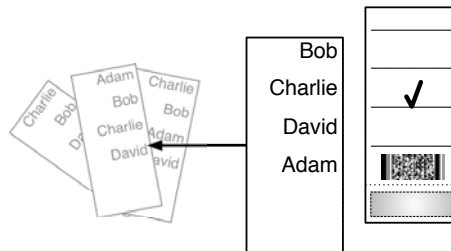


Figure 3: Separating the S&V ballot. Alice separates the left half of her ballot and places it into the appropriate receptacle which contains other discarded left halves (Alice could easily take one to claim she voted differently.)

1.2 Overview of the Ideas

Scratch & Vote combines a number of existing cryptographic voting ideas in a novel way, with some interesting new variations.

Homomorphic tallying. Cryptographic paper ballots do not naturally support write-in votes. Generally, when Alice wants to write in a name, she selects the “write-in” pre-determined pseudo-candidate option, and follows a separate process to specify her candidate. Thus, our first proposal for S&V is to use homomorphic aggregation to simplify the tally for pre-determined candidates, as originally proposed by Benaloh [9, 3] and, more specifically, as extended by Baudron et al. [2] for multi-candidate election systems. This design choice opens the door to further simplifications.

2D-barcode and scratch surface. With homomorphic tallying and pre-determined candidate names, all of the ciphertexts required for one race on one ballot can be fully represented using a single 2D-barcode. The randomization values used to generate these ciphertexts are also printed on the ballot, under a scratch surface. Thus, a ballot is entirely self-contained: by scratching and checking the encryption of the candidate ordering against the ciphertexts in

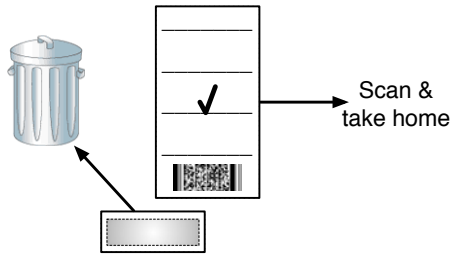


Figure 4: Casting the S&V ballot. The election official verifies that the scratch surface is intact, then discards it. The remainder of the ballot is cast using a typical modern scanner. Alice then takes it home as her receipt.

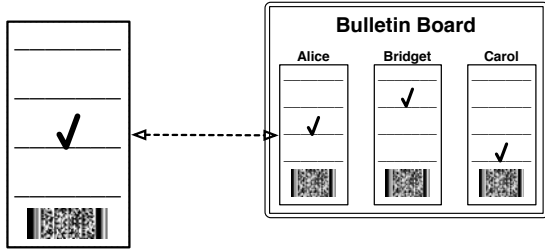


Figure 5: Verifying proper S&V casting. Alice can look up her ballot on the web, using her name and confirming that the barcode matches (assuming she or her helper organization has a barcode scanner.)

the 2D-barcode, one can immediately verify ballot correctness. This auditing requires only a barcode scanner, a basic computer, and the public election parameters. In particular, this auditing process *does not* require network access.

Cut-and-choose at the precinct. Once a ballot is audited, it cannot be used for voting: with its randomization values revealed, the ballot is no longer privacy-protecting. Thus, auditing is used in a cut-and-choose process: each voter may select two ballots, auditing one and voting with the other. The specific advantage of S&V is that this cut-and-choose auditing requires no election official intervention: the ballot and the public election parameters are sufficient. Thus, auditing in S&V is practical enough to be performed live, in front of the voter, just before she casts her ballot. In addition, local election officials may audit a number of ballots on their own before voting begins: once again, these local election officials only need the public election parameters to successfully audit.

Proofs of correctness and certified ballot list. In a homomorphic tallying system, auditors want assurance that the encrypted ballots contribute no more than one vote per race; otherwise, a malicious official and voter could collude to artificially inflate a candidate’s tally. For this purpose, election officials prepare zero-knowledge proofs of correctness for each official ballot. These proofs are published on the bulletin board for all to see prior to election day, and only ballots whose proofs verify are included in the tally.

As a result of this tallying condition, voters now need

assurance that their ballot won’t be disqualified at some point after ballot casting. Unfortunately, the sheer size of the proof precludes printing it on the ballot alongside the ciphertexts.

To address this concern, election officials produce a certified ballot list containing ballots that officials are prepared to guarantee as correct. This certified list can be easily downloaded to each physical precinct before the polls open. The voter can then check that his ballot is present on the certified list before voting. In addition, this certification prevents spurious complaints from malicious voters who might inject fraudulent ballots in to the system solely for the purpose of complaining and holding up the proper execution of the election.

1.3 Related Work

Chaum [7] introduced the first paper-based cryptographic scheme in 2004. Ryan et al. [8] proposed a variant, the Prêt-a-Voter scheme, recently extended to support reencryption mixes and just-in-time ballot printing [29]. Another variant by Randell and Ryan [28] suggests the use of scratch surfaces (though for different goals than ours). Chaum’s latest variant, called Punchscan [20, 6], proposes a number of interesting variations to further simplify the paper-based voting approach.

All of these methods make use of mixnets to anonymize the ballots, though it should be noted that the mixnet proposed by Punchscan is simplified and uses only hash-based commitments for the shuffle permutation. In this latter case as well as in most decryption mixnets, proofs are performed by randomized partial checking [17]. In the case of reencryption mixnets, Neff’s proof [23] can be used. In all cases, however, the officials are involved in the anonymization of the ballots.

1.4 This Paper

In section 2, we cover some preliminaries. We cover the basic S&V method in section 3, some potential extensions in section 4, and the adaptation of S&V to the Chaum ballot in section 5. We consider the system’s threat model in section 6, before concluding in section 7.

2. PRELIMINARIES

2.1 Paillier Cryptosystem

The Paillier public-key cryptosystem [24] provides semantically secure encryption, efficient decryption, and an additive homomorphism by ciphertext multiplication. The details of the Paillier cryptosystem can be found in the original paper. We provide a brief summary:

- $\text{Gen}(1^k)$: generate two safe primes p_1 and p_2 . The secret key sk is $\lambda = \text{lcm}(p_1 - 1, p_2 - 1)$. The public key pk includes $n = p_1 p_2$ and $g \in \mathbb{Z}_n^*$ such that $g \equiv 1 \pmod n$. Oftentimes, $g = n + 1$.
- $\text{Enc}_{pk}(m; r)$: encrypt a message $m \in \mathbb{Z}_n$ with randomness $r \in \mathbb{Z}_n^*$ and public key pk as $c = g^{m+r \cdot n} \pmod{n^2}$. We write $c = \text{Enc}_{pk}(m)$ when the randomness r is not crucial to the explanation.
- $\text{Dec}_{sk}(c)$ decrypt a ciphertext $c \in \mathbb{Z}_{n^2}^*$. Consider function $L(x) = (x - 1)/n$. Decryption is then:

$$\frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$$

The additive homomorphism enables the following:

$$\text{Dec}_{sk}(\text{Enc}_{pk}(m_1) \cdot \text{Enc}_{pk}(m_2)) = m_1 + m_2$$

Generalized Paillier. Damgård and Jurik [11] provide a generalization of the Paillier cryptosystem that yields a larger plaintext domain with relatively less ciphertext expansion. Specifically, a plaintext in \mathbb{Z}_n^s can be encrypted into a ciphertext in $\mathbb{Z}_n^{(s+1)}$, where the security of the scheme still depends on the bit-length of n . The security of this generalized scheme is proven under the same assumption as the original Paillier scheme.

Threshold Decryption. The Paillier cryptosystem supports fairly efficient threshold decryption [14], even in its generalized form [11]. It also supports fairly efficient distributed key generation [12]. In other words, it is possible for k officials to jointly generate a Paillier keypair (pk, sk) such that each has a share $sk^{(i)}$ of the secret key. It is then possible for an h -sized subset of these k officials to jointly decrypt a ciphertext in a truly distributed protocol.

Practical Considerations. Paillier relies on the hardness of the factoring problem. Thus, we must assume at least a 1024-bit modulus, and potentially a 2048-bit modulus. Given a κ -bit modulus, plaintext size is κ bits while ciphertext size is 2κ bits. For a larger plaintext domain, we can use Generalized Paillier, as described above.

2.2 Homomorphic Counters

The homomorphic voting approach was made practical by Baudron et al. [2], using techniques introduced by Benaloh [9, 3]. The homomorphic multi-counter was specifically formalized by Katz et al. [19].

Baudron et al. describe a multi-counter encrypted under an additive cryptographic system, like Paillier. The bit-space of the plaintext is partitioned into separate counters, ensuring that enough bits are dedicated to each counter so that no overflow occurs from one counter to another (as this would violate the correctness of the multi-counter).

Assuming a message domain of \mathbb{Z}_n where $b = |n|$ is the bit-size of n , we encode a value t_j for counter $j \in [1, l]$ as $t_j \cdot 2^{(j-1)M}$. Thus, each counter can run only up to $2^M - 1$, and we must ensure that $b > lM$. To add 1 to counter j contained within the multi-counter T , we use the additive homomorphic property:

$$T' = T \cdot \text{Enc}_{pk}(2^{(j-1)M})$$

Note that, given the semantic security of the Paillier cryptosystem, an observer cannot tell, from looking at this homomorphic operation, which internal counter was incremented. In other words, given encrypted messages, the homomorphic aggregation into an encrypted counter can be performed and verified by any observer.

2.3 Proofs of Correctness and NIZKs

If Alice encrypts a message m into a ciphertext c using the Paillier cryptosystem, she can prove, in honest-verifier zero-knowledge, that c is indeed the encryption of m , using a typical, three-round interactive protocol similar to Guillou and Quisquater’s proof of RSA pre-image [15].

Using the techniques of Cramer et al. [10], this protocol can be extended to prove that ciphertext c encrypts *one* of possible values (m_1, m_2, \dots, m_k) , without revealing which one. Combining this with the homomorphic proof technique of Juels and Jakobsson [16], one can prove, fairly efficient and in zero-knowledge, that a set of ciphertexts (c_1, c_2, \dots, c_k) encrypts a permutation of m_1, m_2, \dots, m_k , assuming that no two subsets of $\{m_i\}$ have the same sum:

- for each c_i , prove that c_i encrypts one of m_1, \dots, m_k ,
- the homomorphic ciphertext sum $\bigoplus_i c_i$ is the correct encryption of the plaintext sum $\sum_i m_i$.

For more than a handful of plaintexts, more efficient proof techniques are available, including Neff’s shuffle proof of known plaintexts [23].

In any case, all of these proofs can be made non-interactive using the Fiat-Shamir heuristic [13], where the interactive verifier challenge is non-interactively generated as the cryptographic hash of the prover’s first message in the three-round protocol. These types of proof, first introduced by Blum et al. [4], are abbreviated NIZKs.

2.4 Paper Ballots

Existing paper-based cryptographic methods use two types of ballot layout: the Prêt-a-Voter split ballot, and the Punchscan layered ballot. We review these approaches here, as they can be both adapted to use S&V.

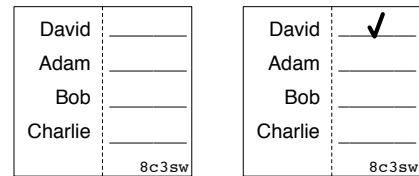


Figure 6: The Prêt-a-Voter Ballot: A ballot is a single sheet of paper with a midline perforation. The Voter fills in her choice, then tears the left half off and destroys it, casting the right half.

Prêt-a-Voter. In Ryan’s Prêt-a-Voter, the ballot is a single sheet of paper with a perforated vertical midline. Candidate names appear on the left in a per-ballot-randomized order, with a corresponding space on the right half. After Alice, the voter, has marked her choice in the appropriate space, the two halves are separated: the left half (the one with the candidate names) is discarded, and the right half is cast. The right half contains a mixnet onion that allows administrators to recreate the left half of the ballot and determine the voter’s choice. (See Figure 6.)

Punchscan. In Chaum’s Punchscan [20], the ballot is composed of two super-imposed sheets. The top sheet contains

the question, an assignment of candidates to codes (randomized by ballot), and physical, circular holes half-an-inch wide which reveal codes on the bottom sheet. The codes on the bottom sheet match the codes on the top sheet, though their order on the bottom sheet is randomized.

Alice, the voter, selects a candidate, determines which code corresponds to this candidate, and uses a “bingo dauber” to mark the appropriate code through the physical hole. The use of this thick marker causes both sheets to be marked. Then, Alice separates the two sheets, destroys one, and casts the other. Individually, each sheet displays the voter’s choice as either a code or a position, but the correspondence of code to position is only visible when both sheets are together. A hash-committed permutation on both sheets allows the administrators to reconstitute the discarded half and recover the vote. Because Alice chooses which half to destroy and which half to cast, she can eventually get assurance, with 50% soundness, that her ballot was correctly formed: the election officials eventually reveal what the kept halves should look like.

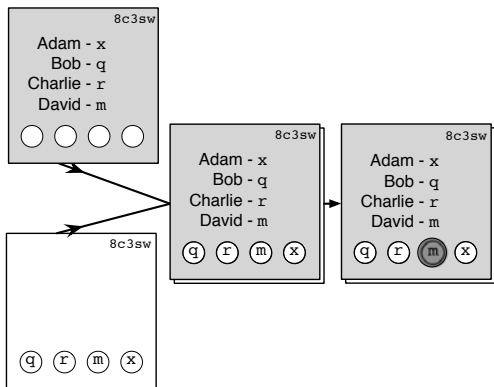


Figure 7: The Chaum Ballot: A ballot is composed of two super-imposed sheets. Alice, the voter, marks both sheets simultaneously using a dauber. The two sheets are separated, one is discarded, and the other is scanned and posted on the bulletin board. This same half is also the voter’s receipt.

Auditing Prêt-a-Voter and Punchscan. In both Prêt-a-Voter and Punchscan, there are two ballot auditing components: verification of the *correct ballot form*, and verification of *correct tallying*. In both schemes, a system-wide cut-and-choose is performed before the election: for a randomly selected half of the ballots, election officials reveal the randomization values used in creating the ballots. These audited ballots are thus spoiled, as they no longer protect ballot secrecy. The remaining ballots, now proven to be almost all correct with very high probability, are used in the actual election.

Once ballots are cast, they are shuffled, and the post-election audit consists of Randomized Partial Checking [17] on the shuffle and the prior permutation commitment. **Punchscan** adds an additional verification of ballot form after the election, thanks to the voter decision of which half to keep and which half to discard. This guarantees that any cheated ballot that made it through the initial audit will be detected with 50% probability.

Limitations. In the case of Prêt-a-Voter, significant synchronous involvement of election officials is required during all audits. It is particularly challenging to interactively reveal the randomization values for half the ballots while keeping the other half truly secret. Consequently, this audit is performed by election officials in advance. Individual voters must trust that this audit was performed correctly, in particular that election officials didn’t collude to produce faulty ballots. This is a slightly weaker verification property than we would like: ideally, voters should get direct assurance that their vote was recorded as intended, without having to trust election officials.

In the case of **Punchscan**, there is also some degree of dependence on synchronous involvement of the election officials. While the additional check performed on the ballot form certainly alleviates this situation—Alice now gets direct assurance that her ballot was correctly formed—this assurance comes *after* the close of elections. This delay may reduce Alice’s trust in the system, since the error correction protocol will likely be onerous.

3. THE SCRATCH & VOTE METHOD

We assume a single race for now. Section 4 naturally extends these techniques to multiple races.

3.1 Election Preparation

At some point prior to election day, the list of candidates is certified by election officials and publicized for all to see. The l candidates are ordered in some fashion for purposes of assigning index numbers (alphabetical order is fine). This ordering need not be known by individual voters. Thus, election laws on candidate ordering should not apply to here.

Election officials then jointly generate a keypair for the election, where official O_i holds share $sk^{(i)}$ of the decryption key sk , and the combined public key is denoted pk . A parameter M is chosen, such that 2^M is greater than the total number of eligible voters. Election officials ensure that $b = |n|$ is large enough to encode a multi-counter for l candidates, each with M bits, i.e. $b > Ml$. A vote for candidate j is thus encoded as $\text{Enc}_{pk}(2^{(j-1)M})$. When all is said and done, the election parameters are publicized:

$$params = (pk, M, \{cand_1, cand_2, \dots, cand_l\})$$

Example. With $l = 4$ candidates, a candidate ordering is assigned: Adam is #1, Bob #2, Charlie #3, and David #4. With 2×10^8 eligible voters (big enough for the entire US), we set $M = 28 > \log_2(2 \times 10^8)$. A Paillier public key with $|n| = 1024$ bits is largely sufficient for this single-race election. In fact, we could have up to 35 candidates for this single race, or 7 races with 5 candidates each, without having to alter cryptographic parameters.

3.2 Ballot Preparation

The Ballot. Our first S&V ballot is based on the Ryan Prêt-a-Voter ballot[8]. Each ballot is perforated along a vertical midline. On the left half of the ballot is the list of candidate names, in a randomized order. The right half of the ballot lines up scannable checkboxes (represented as lines in our diagrams) with each candidate on the left half. The voter will mark one of those checkboxes.

Also on the right half, the S&V ballot contains the representation of ciphertexts that encode votes corresponding to each ordered option. The representation of these ciphertexts can be machine-readable only and should be scannable, e.g. a 2D-barcode [32]. Just below this barcode, the ballot also includes a scratch surface, under which are hidden the randomization values used to produce the ciphertexts in the barcode. (See Figure 8.)

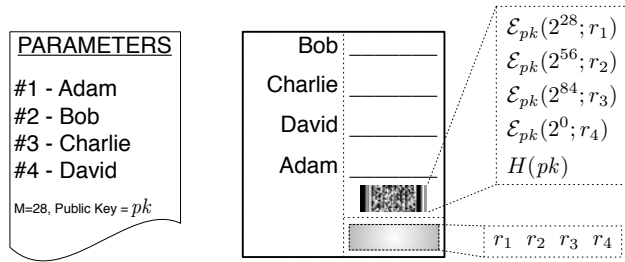


Figure 8: Public Election Parameters and the S&V Ballot. Election parameters are on the left. The ballot is midline-perforated. The order of the candidate names is randomized for each ballot. The 2D-barcode on the bottom right corner of the ballot contains ciphertexts encoding the candidates in the corresponding order according to the public election parameters. The scratch surface on the left hides the randomization values r_1, r_2, r_3, r_4 used to create the ciphertexts on the right.

The Proofs. Election officials must also generate NIZK proofs of ballot correctness. These proofs will not be printed on the ballot, as they are too long. Instead, they are kept on the public bulletin board, indexed by the sequence of ciphertexts on the corresponding ballot (or a hash thereof), and used at tallying time to ensure that all ballots contribute at most one vote per race.

In addition to these proofs, election officials compile an *official ballot list*, which includes all properly created ballots designated again by the sequence of ciphertexts on each ballot. Officials digitally sign this ballot list, posting the list and signature on the bulletin board. This official ballot list is particularly useful to help prevent various denial-of-service attacks against both voters and election officials.

Example. Assume the randomized candidate ordering of a given ballot is “Bob, Charlie, David, Alice”, or, by index, “2,3,4,1”. Recall that $M = 28$. The machine-encoding on the right ballot half should then be: $c_1 = \text{Enc}_{pk}(2^{28}; r_1)$, $c_2 = \text{Enc}_{pk}(2^{56}; r_2)$, $c_3 = \text{Enc}_{pk}(2^{84}; r_3)$, $c_4 = \text{Enc}_{pk}(2^0; r_4)$.

This encoding requires 4 ciphertexts, or 8192 bits. Under the scratch surface lie, in plaintext, r_1, r_2, r_3, r_4 . Election officials also generate $\Pi_{H(c_1, c_2, c_3, c_4)}$, a NIZK of proper ballot form indexed by the ciphertexts, then post it to the bulletin board. The same hash $H(c_1, c_2, c_3, c_4)$ is also included in the compiled list of official ballots, which the election officials eventually sign.

3.3 Ballot Auditing

Ballot auditing in S&V uses a cut-and-choose approach to verify candidate ordering (while further ballot correctness is

enforced by the bulletin-board NIZK). A random half of the ballots are audited, and almost all remaining ballots are then guaranteed to be correctly constructed: the probability that more than x bad ballots go undetected is 2^{-x} . Once audited, a ballot is spoiled and cannot be used to cast a vote.

Auditing a single ballot. This auditing process is similar to that of Prêt-a-Voter and Punchscan, with one major difference: auditing can be performed using only the *public* election parameters, without election-official interaction.

1. **Scratch:** the scratch surface is removed to reveal the randomization values.
2. **Encrypt:** the candidate ordering is encrypted with the revealed randomization values.
3. **Match:** the resulting ciphertexts are matched against the ciphertexts in the 2D-barcode.

Note that, to automate this process without having to scan or type in the displayed candidate ordering, one might perform the matching the other way around: read the ciphertexts, try all possible plaintext candidates with each revealed randomization value (effectively a decryption), and display the expected candidate ordering. Matching the actual ordering against this expectation can be performed by the voter herself.

Spoiling a ballot. A ballot no longer protects privacy if its randomization values are revealed. Thus, if its scratch surface is removed, a ballot should be considered spoiled, and it cannot be cast. This is consistent with existing uses of scratch surfaces, e.g. those used on lottery tickets, and the usual “void if scratched off” message can be printed on top of the scratch surface. This condition for ballot casting must be enforced at ballot casting time, as will be described Section 3.4.

Who should audit?. Though we find that individual voter auditing is preferable, some might prefer to audit ballots in a centralized fashion. Scratch & Vote supports such an audit method, of course. One can also imagine officials auditing a few ballots on their own before election day, in addition to per-voter auditing. S&V enables all of these auditing combinations.

Checking for Variability in Ordering. Malicious election officials might attempt to breach Alice’s privacy by presenting all voters only ballots with the *same* candidate ordering. To protect against this de-randomization attack, Alice should select her two ballots herself, ensuring that there is enough variability between the ballots offered to her.

Chosen ballot check. Alice must also check the ballot she actually uses: she needs assurance that her ballot will count, specifically that it won’t be disqualified for some unforeseen reason, e.g. an invalid NIZK proof at tallying time. For this purpose, Alice checks the presence of her ballot on the certified official ballot list, which she can obtain from the

bulletin board ahead of time. If, at a later date, Alice’s ballot is disqualified for any reason, she can present the signed official ballot list as a complaint.

3.4 Ballot Casting

On election day, after having audited and spoiled a first ballot, Alice enters the isolation booth with a second ballot. She fills it out by applying a checkmark (or filling in a bubble) next to the candidate name of her choice. Then, she proceeds to ballot casting:

1. **Detach:** Alice detaches the left half of the ballot and discards it in the appropriate receptacle (inside the booth). She then leaves the voting booth.
2. **Confirm:** An election official verifies that the scratch surface on Alice’s ballot is intact. This is crucial to ensuring the secret ballot: if a voter sees the randomization values for the ballot she actually casts, then she can prove how she voted to a potential coercer.
3. **Second Detach:** The official detaches and discards the scratch surface, in view of all observers, including Alice.
4. **Scan:** Alice feeds the remainder of her ballot through a typical modern scanner, which records the barcode and checkmark position and posts them on a bulletin board *along with Alice’s name (or other identifier)*.
5. **Receipt:** Alice retains this same remainder as her encrypted receipt. She can later check that her ballot is indeed on the bulletin board.

3.5 Tallying

For each ballot on the bulletin board, election officials and observers check its NIZK. If it verifies, the ciphertext corresponding to the checkmark position is extracted from the 2D-barcode and aggregated into the homomorphic counter, just like any other homomorphic voting system. Anyone can verify that only valid ballots have been aggregated, as any observer can verify the NIZK and re-perform the appropriate homomorphic aggregation.

Similarly, all election trustees can independently verify that the homomorphic aggregation has been performed correctly. Then, the single resulting ciphertext counter is decrypted by a quorum of these trustees, along with proofs of correct decryption. The resulting plaintext reveals the vote count for each candidate. The tally and trustee proofs are posted to the bulletin board for all to see.

3.6 Practical Considerations

We consider computation requirements and physical constraints of the ballot:

- generating the NIZK proofs for each ballot, given that ZK proofs are typically expensive,
- auditing a ballot at voting time, given the voter’s limited time and patience, and
- fitting all of the ciphertext data in reasonably sized 2D barcodes.

Consider an election with 5 races, each with 5 candidates.

Proofs. Each race contains 5 ciphertexts, one per candidate. Using the CDS [10] proof-of-partial-knowledge technique, each ciphertext must be proven to encrypt one of the 5 candidates. The CDS technique simulates 4 of these proofs and computes a fifth one for real. This requires the equivalent work of 5 proofs, both in terms of computation time and number of bits needed to represent the proof. In addition, the homomorphic sum of the ciphertexts must be proven to encrypt the sum of the candidate representations, which is one more proof. Thus, each race requires 26 proofs, and 5 races thus require 130 proofs.

Each of these proofs, whether real or simulated, requires two modular exponentiations. The entire proof thus requires a total of 260 modular exponentiations. Conservatively, modern processors can perform a 1024-bit modular exponentiation is approximately 12ms on a 2.8Ghz machine running GMP [1]. Thus, a single ballot proof can be performed in just over 3 seconds.

Each of these proofs is composed of 2 Paillier ciphertext space elements, and one Paillier plaintext space element (the challenge). Assuming a 1024-bit modulus, the ciphertext elements are 2048 bits and the plaintext is 1024 bits. Thus, each proof require 5120 bits, and the entire ballot thus requires 83 kilobytes of proof data, which is posted on the bulletin board (e.g. a web site.)

Ballot Verification. At the polls, the only verification cost is the single ballot audit per voter. Given the 5 randomization values, all 5 values of r^n can be computed through modular exponentiation, after which only modular multiplications are needed to re-discover the randomized ordering. These multiplications are negligible in comparison to the modular exponentiations. Thus, ballot verification can be performed in 60ms per race, or 300ms for our considered ballot. The scratch-off and the time allotted for each person to vote (1-2 minutes) will likely make the cryptographic cost negligible.

Barcode Size. The PDF417 2D-barcode standard [31] can store 686 bytes of binary data per square inch, using a symbol density that is easily printed and scanned. In our example with 25 candidates, 25 Paillier ciphertexts are required, which means 6400 bytes, assuming a 1024-bit modulus for Paillier. Thus, 10 square inches are sufficient to represent all of the ciphertexts we need for this sample election. Even if we factor in significant error correction, this represents less than 1/8th of the surface area of a a US Letter page.

4. EXTENSIONS

We now explore a few extensions to make Scratch & Vote more practical.

4.1 Helper Organizations

We do not expect individual voters to show up to the polls with the equipment required to audit ballots and check the official ballot list. Instead, *helper organizations*, e.g. political parties and activist organizations, can provide this service at the polls. Voters can consult one or more of these, at their discretion. Most importantly, these helpers are not trusted with any private election data.

4.2 Multiple Races & Many Candidates

When the election contains more than one race, it may outgrow the space of the multi-counter, which can only hold $|n|/m$ counters. One solution is to use higher-degree encryption using the Damgård-Jurik generalization [11], so that the counter space can be $s|n|$ rather than $|n|$, with a corresponding ciphertext length of $(s + 1)|n|$. Unfortunately, this ciphertext size may outgrow the 2D-barcode encoding, which is expected to hold no more than a few kilobytes.

Another option is to designate, in the public election parameters, separate multi-counters, potentially one per race. In that case, the parameters must also indicate the race/multi-counter assignments. With a separate 2D-barcode per race, most practical cases are accounted for, barring elections such as the California Recall election of 2004, which had more than 100 candidates.

In such borderline cases, there is no choice but to use the Damgård-Jurik generalization, as the individual ciphertexts for a given race should not be distinguishable and thus cannot be assigned to different multi-counters. If a single 2D-barcode cannot hold all the required ciphertexts for that one race, we can, as a last resort, designate a separate 2D-barcode for each candidate. The resulting auditing complexity is an inevitable consequence of these extreme conditions.

4.3 Reducing the Scratch Surface

As the printed material behind the scratch surface may become damaged from the scratching, we cannot expect to reliably store significant amounts of data behind this scratch surface. In fact, it is easy to reduce this data by having election parameters designate a pseudo-random function, call it PRF, which generates all the randomization values from a single short seed, which need only be 128 bits. Such a data length can be easily encoded as alphanumeric characters or as a single-dimension barcode, both of which offer enough redundancy to withstand quite a few scratches.

4.4 Chain Voting and Scratch Surfaces

All paper-based voting systems have long been known to be susceptible to chain voting attacks. In these attacks, a coercer gives Alice a pre-marked ballot before she enters the polling location, expecting her to cast this pre-marked ballot and return a blank ballot to him on her way out.

The well-known countermeasure to chain voting attacks [18] suggests having unique ballot identifiers on a tear-off slip attached to the ballot. An official writes down the ballot identifier for Alice before she enters the isolation booth. At ballot casting time, the official checks that this ballot identifier still matches the pre-recorded value. Then, for anonymity, the identifier is torn off and discarded.

This process is, in fact, almost identical to the scratch surface tear-off we suggest. Thus, our election-official verification process can be piggy-backed onto existing practices. In addition to checking the ballot identifier, the election official must simply check the scratch surface integrity. The overhead of our proposal at casting time is thus minimal.

4.5 Write-Ins

Like the Prêt-a-Voter and Punchscan ballots, Scratch & Vote does not support write-in votes out of the box. A separate system should be used, where a special option named “write-in” is selected by the voter, and the voter can separately cast the content of the write-in. The details of this

process can be worked out for all paper-based schemes, possibly using the vector-ballot method of Kiayias and Yung [21].

5. ADAPTING PUNCHSCAN

Recall that Chaum’s Punchscan facilitates more races per ballot than Prêt-a-Voter, because the full ballot face can be used without a midline separation. However, Punchscan is also more complicated, because the voter may cast either sheet. This makes the mandatory destruction of the remaining half more delicate, since Alice could easily sell her vote if she successfully preserves the second half.

Thus, we propose a new ballot that combines the qualities of Punchscan and Prêt-a-Voter and adds the S&V method. As this ballot originated from Punchscan, we call it the Punchscan Scratch & Vote ballot. However, we note that it inherits some properties from Prêt-a-Voter, too.

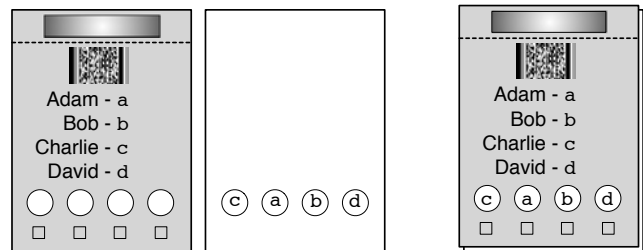


Figure 9: The Punchscan Scratch & Vote variant. The left and middle sheets are superimposed to create the ballot on the right. The bottom sheet contains no identifying information. The top layer has circular holes big enough to let the candidate ordering from the bottom sheet show through. The checkmark locations, represented by small squares, are only on the top layer.

Punchscan Scratch & Vote Ballot. The Punchscan Scratch & Vote ballot, like Punchscan, is composed of two superimposed sheets. Unlike the original Punchscan, the two sheets serve different purposes. Alice, the voter, will be expected to separate both halves and cast the top half in all cases. The bottom half, like Prêt-a-Voter’s left half, is generic, and its destruction need not be strongly verified.

The top sheet lists the races and candidates in standard order, with a standard code assigned to each candidate (e.g. ‘D’ for democrats, ‘R’ for republicans.) Again, this differs from the Punchscan ballot, where random codes are assigned to candidates. This top sheet offers checkboxes for each race, and one hole *above* each checkbox which reveals the code letter displayed on the bottom half at that position. Also included on the top sheet are the S&V components: the ciphertexts in a 2D-barcode, and the randomization values hidden under a scratch surface.

The bottom sheet contains only the standard candidate codes in random order. The ciphertexts on the front sheet should match this random order. Note, again, that this bottom sheet is entirely generic: it contains no identifier of any kind, no ciphertexts, and no randomization values. It functions exactly like the Prêt-a-Voter left half.

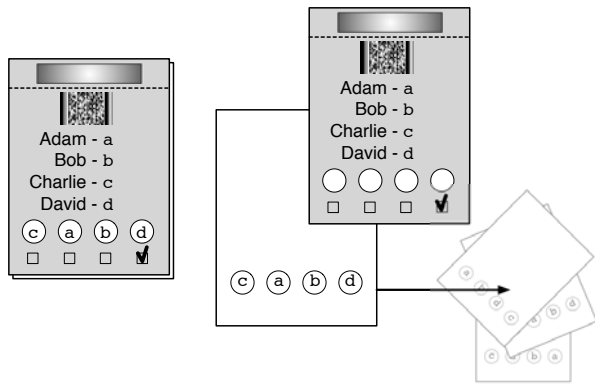


Figure 10: The Punchscan Scratch & Vote ballot separation. Alice separates the top and bottom sheets, depositing the bottom sheet in the appropriate receptacle. The top sheet is effectively an encrypted ballot. Note how the “codes” for each candidate are intuitively chosen, not random.

Pre-voting Verification. Just like in the original Prêt-a-Voter-based S&V ballot, Alice chooses two ballots. She audits one by scratching off the scratch surface and having a helper organization verify the randomization values for the ballot’s candidate ordering. Alice then votes with the second ballot, as the audited ballot with the scratch surface removed is now void. One notable advantage of S&V is that Alice can perform the audit *before* she casts her ballot.

Casting the Ballot. Alice marks her ballot in isolation. Unlike the original Punchscan method, the markings in the top-sheet bubbles do not bleed through to the bottom half. When she is ready, Alice detaches the bottom half of the ballot and discards it in the proper receptacle (where, again, she can easily grab another bottom half to claim that she voted for someone else.)

Alice then presents the top sheet of her ballot to the election official, who treats it exactly like in the Prêt-a-Voter Scratch & Vote ballot: verify the scratch surface, detach and discard it, and scan the remainder. Again, this remainder serves as Alice’s receipt.

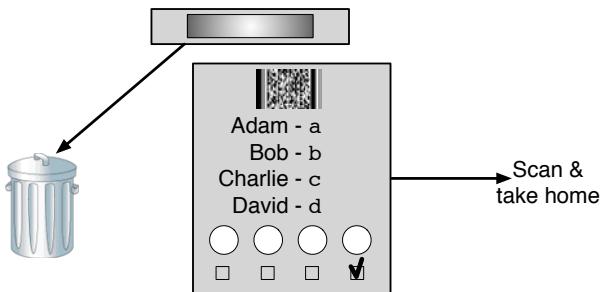


Figure 11: Casting a Punchscan Scratch & Vote ballot. An election official verifies that the scratch surface is intact, then tears it off and discards it in view of all observers. The remainder is scanned and cast. Alice then takes it home as her receipt.

6. THREAT MODEL

We consider various threats and how Scratch & Vote handles them. We cover the threats thematically rather than chronologically, as some threats pertain to multiple steps of the election lifecycle.

6.1 Attacking the Ballots

Election officials. An election official might create bad ballots in two ways: a completely invalid ballot or, more perniciously, a valid ballot that does not match the human-readable candidate ordering. In either case, the first line of defense is the voter cut-and-choose: only a small number of such ballots can go undetected, since half of them will get audited randomly. In the case of completely invalid ballots, the verification is much more stringent: election officials would have to answer for certified ballots that do not have a proper NIZK proof, and only valid ballots can have proper NIZK proofs.

External parties. External parties may wish to introduce fraudulent ballots, most likely as a denial-of-service attack against voters at certain precincts, or, by vastly increasing the number of complaints, as a denial-of-service attack against election officials. These problems are thwarted by the certified ballot list. The moment an election official discovers an uncertified ballot, he can begin an investigation. If officials fail to catch the problem, the voters’ helper organizations will. Consequently, fraudulent ballots should be caught before the voter enters the isolation booth.

Collusion between officials and voters. A single maliciously crafted ballot could alter the count significantly in the homomorphic aggregation. This is particularly problematic if the officials collude with a voter who won’t perform the proper cut-and-choose audit because he is very much interested in using the fraudulent ballot. Once again, the NIZK proof on the bulletin board prevents this from *ever* happening, providing a universally verifiable guarantee that each cast ballot only contributes a single vote to a single candidate for each race.

6.2 Attacking Ballot Secrecy

Leaky Ballots. Election administrators could leak information about the ballot candidate ordering using the ciphertext randomness. This threat is somewhat lessened with the use of seed-based randomness, where a portion of the seed is public and selected *after* the individual’s ballot seeds are picked. However, this topic requires further exploration.

Tampering with the scratch surface. Eve, a malicious voter, may attempt to remove the scratch surface from her ballot, read the randomization values, and replace the scratch surface undetected. This would allow Eve to sell her vote, given that her encrypted vote will be posted on the bulletin board, along with Eve’s full name, for all to see and audit. We must assume, as a defense against this threat, that the scratch surface is sufficiently tamper-proof to prevent such an easy replacement that would fool an election administrator. Real-world experiments will be necessary to determine if this is feasible.

Ballot face recording. One significant weakness of all pre-printed paper-based cryptographic voting, including Scratch & Vote, is that election officials who produce the ballots may record the ballot ciphertexts and candidate orderings, thus violating ballot secrecy.

Even in Prêt-a-Voter and Punchscan, which use multiple authorities in a mixnet setting, the last mix server knows the final candidate ordering. It may be worthwhile to explore clever ways of distributing the ballot generation mechanism, though the best solution may be process-based, where machines produce ballots and immediately forget the randomness used. The recent proposal of Ryan and Schneider [29] addresses this threat with on-demand ballot printing, though this requires significantly more deployed technology at the precinct level.

The ballot face recording problem exists for more than just election officials: those who transport ballots may have a chance to record the correspondence of candidate ordering to barcode. We note, again, that Prêt-a-Voter and Punchscan suffer from the same problem. One promising defense in all cases is simply to hide some portion of the ballot such that it can no longer be uniquely identified during transport. For example, individual ballots can be sealed individually in opaque wrappers. Alternatively, the 2D-barcode can be hidden under opaque plastic that can be peeled off prior to voting. If the printed barcode is particularly resilient, one can even use an additional scratch surface.

Casting. At ballot casting time, election administrators must ensure that the cast ballot has not revealed its randomization values, for this would enable vote selling. Of course, this verification must be performed without violating ballot secrecy in the process. Our proposal specifically addresses this threat: an election official only sees the encrypted half of the ballot. He can take all the necessary care to verify that the scratch surface is intact, while ballot secrecy remains protected.

Another threat remains: official-voter collusion. If an election official and voter collude to preserve, rather than discard, the scratch surface, the voter may be able to reveal his selection to the official (and later to other parties.) Sufficient observation of the voting process by competing political parties should address this issue. S&V further mitigates this risk with ballots created such that the “missing half” is generic. Thus, voters can easily pick up alternative “missing halves” to claim they voted differently, and a coercer may not be certain whether the claimed half is, indeed the proper decryptor half for Alice’s ballot.

Coerced Randomization. Recently, a new threat to paper-based voting systems has been pointed out: coerced randomization [5]. In this attack, a coercer wishes to “reduce to random” Alice’s vote. Consider, for example, the situation where Alice votes in a precinct that historically favors one political party by a wide margin. Such precincts are quite common in the United States. A coercer can ask Alice to vote for the first candidate in the list, no matter what that candidate is. The coercer can check this by viewing the bulletin board under Alice’s name or voter identification number. Of course, the coercer won’t know for sure who Alice voted for—in fact she may, by sheer luck, have

obtained a ballot where this position coincides with her preferred choice—but he will have effectively reduced her vote to random. With enough voters, the coercer can statistically reduce the number of votes for the party typically favored by this precinct.

By way of countermeasure, one can offer the voter enough ballot ordering selections that she can pick a ballot where the prescribed behavior fits her personal choice. Unfortunately, the attack can become much more complex: for example, the prescribed behavior may involve voting for a position on the ballot that depends on the ballot identifier. This issue merits *significant* research. However, we note that Scratch & Vote does not make this problem any worse: Punchscan and Prêt-a-Voter are equally vulnerable.

6.3 Attacking the Bulletin Board and the Tally

Much of the security of the tallying process depends on the bulletin board. An attacker may want to insert fraudulent data, for example to change the election parameters or replace an honest citizen’s cast vote. Digital signatures on all posted data can prevent such attacks, assuming that a minimal PKI is deployed to certify the public keys of election officials. Public observers of the bulletin board content, including helper organizations, can then detect bad data.

Alternatively, an attacker may want to suppress information from the bulletin board. In particular, the bulletin board server itself may suppress information. To protect against this attack, the bulletin board should be implemented by multiple servers. These servers may run a distributed Byzantine agreement protocol to ensure consistency of content [22], or observers may simply rely on cryptographic signatures of the content and the redundancy of the servers to catch any server that suppresses content, using, for example hash trees [26].

Given a secure bulletin board implemented as above, attacks on the tallying process can be prevented, because every step is verified with proofs. Ballots are proven well-formed by the NIZK proof on the bulletin board, any observer can re-perform the homomorphic aggregation, and the final tally decryption is also proven correct.

7. CONCLUSION

We have proposed Scratch & Vote, a simple cryptographic voting system that can be implemented with today’s technology, at very low cost and minimized complexity. Most importantly, ballots are self-contained: any helper organization, or the voter herself, can audit the ballot without interacting with election officials, *before* the voter casts her ballot. Given its intuitive use of scratch surfaces, Scratch & Vote may prove particularly useful in providing an accessible explanation of the power of cryptographic verification for voting systems.

8. ACKNOWLEDGMENTS

The authors wish to thank Peter Ryan and David Chaum for useful discussions, Stephen A. Weis for useful suggestions regarding scratch surfaces, and the anonymous reviewers for particularly insightful comments. The authors wish to acknowledge support from a Knight Foundation grant to the Caltech/MIT Voting Technology Project.

9. REFERENCES

- [1] Swox AB. The GNU Multi-Precision Arithmetic Library. <http://www.swox.com/gmp/>.
- [2] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *PODC*, pages 274–283. ACM, 2001.
- [3] Josh Benaloh and Moti Yung. Distributing the power of government to enhance the power of voters. In *PODC*, pages 52–62. ACM, 1986.
- [4] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112. ACM, 1988.
- [5] C. Andrew Neff. Coerced Randomization, April 2006. private conversation with and unpublished manuscript by Andy Neff.
- [6] David Chaum. Punchscan. <http://punchscan.org> viewed on August 13th, 2006.
- [7] David Chaum. Secret-Ballot Receipts: True Voter-Verifiable Elections. *IEEE Security and Privacy*, 02(1):38–47, 2004.
- [8] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A practical voter-verifiable election scheme. 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] J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme. In *FOCS*, pages 372–382. IEEE Computer Society, 1985.
- [10] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.
- [11] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of paillier’s probabilistic public-key system. In Kwangjo Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001.
- [12] Ivan Damgård and Maciej Koprowski. Practical threshold rsa signatures without a trusted dealer. In Pfitzmann [25], pages 152–165.
- [13] A. Fiat and A. Shamir. How to prove yourself. practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–189. Springer, 1987.
- [14] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In Yair Frankel, editor, *Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 90–104. Springer, 2001.
- [15] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In C. G. Günther, editor, *EUROCRYPT*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer, 1988.
- [16] Markus Jakobsson and Ari Juels. Millimix: Mixing in small batches. Technical Report 99-33, Center for Discrete Mathematics & Theoretical Computer Science (DIMACS), 1999.
- [17] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In Dan Boneh, editor, *USENIX Security Symposium*, pages 339–353. USENIX, 2002.
- [18] Joseph P. Harris. *Election Administration in the United States*. Brookings Institute Press, 1934.
- [19] Jonathan Katz, Steven Myers, and Rafail Ostrovsky. Cryptographic counters and applications to electronic voting. In Pfitzmann [25], pages 78–92.
- [20] Kevin Fisher and Richard Carback and Alan Sherman. Punchscan: Introduction and System Definition of a High-Integrity Election System. In Peter A. Ryan, editor, *Proceedings of the IAVoSS Workshop On Trustworthy Elections (WOTE’06)*, Cambridge, UK, June 2006.
- [21] A. Kiayias and M. Yung. The vector-ballot e-voting approach. In Ari Juels, editor, *Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 72–89. Springer, 2004.
- [22] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. On the composition of authenticated byzantine agreement. In *STOC*, pages 514–523. ACM, 2002.
- [23] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *ACM Conference on Computer and Communications Security*, pages 116–125. ACM, 2001.
- [24] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
- [25] Birgit Pfitzmann, editor. *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*. Springer, 2001.
- [26] Ralph C. Merkle. *Secrecy, authentication, and public key systems*. PhD thesis, Stanford University, 1979.
- [27] Randal C. Archibold. Arizona Ballot Could Become Lottery Ticket, July 2006. <http://www.nytimes.com/2006/07/17/us/17voter.html?ex=1310788800&en=9626060428eeb1ed&ei=5088&partner=rssnyt&emc=rss>.
- [28] B. Randell and P.Y.A. Ryan. Voting technologies and trust. *IEEE Security & Privacy*, 2005. To appear.
- [29] P.Y.A. Ryan and S A Schneider. Prêt à voter with re-encryption mixes. In *ESORICS*, Lecture Notes in Computer Science. Springer, 2006.
- [30] Scratch ‘N Win Ballots To Debut In November, July 2006. <http://www.theonion.com/content/node/50640>.
- [31] The Barcode Software Center. PDF-417, 2003. <http://www.mecsw.com/specs/pdf417.html>.
- [32] Wikipedia. http://en.wikipedia.org/wiki/2D_barcode.