Verified Voting New Mexico
Is a non partisan, not for profit,
public interest group.

A Critical Review of the Triple Ballot Voting System. Part 2:

Cracking the Triple Ballot Encryption.

(Draft V1.5 October 8 2006)

Charlie E. M. Strauss
Verified Voting New Mexico
Los Alamos, NM
cems@vvnm.org

1. Introduction

This review concerns the voting method introduced and dubbed "triple Ballot" by Prof. Ron Rivest. It's debut paper circulated in the final week of September 2006. It has garnered a great deal of attention, even being cited before a congressional hearing as an example of how to secure an election process.

The Three Ballot voting system[1] is a method of symmetric public-key encryption that divides the message, namely a voter's intended vote, into 3 encrypted pieces, no piece of which can reveal the voter's real vote. In this regard if is reminiscent of Chaum's binary decomposition[2]. The brilliant features of this approach are that the encrypted decomposition is done directly by the voters as a side effect of the way they make the marks on the ballot, and since the sum of the marks on the separated pieces is the equivalent to the original vote, there is no need to ever "decrypt" (i.e. rejoin the three pieces), and thus any record of their connection can be deliberately destroyed. Consequently, it requires no computer to generate (hand paper ballots a can be used), and since there are no secret keys to communicate or escrow, direct attacks on keys or the loss of data secrecy do not present a latent threat to the secret ballot or present opportunities for vote manipulation. Because it is intended to overcome these show stopping problems for other proposed encryption approaches it deserves a close analysis to assure it does not have it's own intrinsic security issues.

After the voter completes the triple ballot, the 3 pieces are physically separated and at the end of the voting day, all 3 pieces from all the voters are published in a scrambled-order aggregate. Voters can thus see that their pieces are present without revealing their vote. To allow vote manipulation or errors to be provably detected, the voter is given a receipt which shows the vote pattern contained on one of the 3 ballot strips the voter cast. Since a single one of the pieces does not reveal the vote, the receipt (supposedly) cannot be used for vote selling.

In Part 1 of this analysis we examined the practical implementation issues and revealed the Three Ballot system's enhanced potential for vote selling and cheating, it's difficult user interface, and it's practical risks to the secret ballot. In Part 2, we move from the implementation issue to fundamental defects in the encryption algorithm.

While it is true that no one can determine the voters actual vote solely from this receipt, we show here that in conjunction with the entire published ballot library the vote corresponding to a receipt can be computed in practical cases. We reveal that the encryption is easy to crack completely for any normal election circumstances, and we provide a fast algorithm for doing so in a matter of seconds. The crack uses no special insider access or knowledge, nor does it make statistical assumptions, but rather it simply reverse engineers all of the voted ballots from the published vote data, and consequently ties any voter's receipt to his actual vote. The latter means that this receipt pierces the veil of the secret ballot and directly permits vote selling and coercion. This is tested under various limits to demonstrate the robustness of the crack under expected conditions.

2.0 Review of the Three Ballot process

An elaborate introduction with variations and discussion of the features provided by the triple ballot can be found in Prof. Rivest's working paper[1] on the Three Ballot voting system. Let us briefly review the irreducible elements of the three-ballot scheme. For simplified terminology, we will refer to a ballot as being composed of races (or contest) and races

[1] http://theory.lcs.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf

[2] David Chaum, Peter Y. A. Ryan, and Steve A. Schneider. A practical, voter-verifiable election scheme. Technical Report CS-TR-880, University of Newcastle upon Tyne, 2004. http://www.cs.ncl.ac.uk/research/pubs/ trs/papers/880.pdf.

composed of choices of candidates. And we will further limit discussion to the usual case where the voter can vote for one and only one candidate in any given race, though everything generalizes to choose-N type contests.

1. Beside each candidate's name are 3 bubbles in a row.
2. The voter marks any two of the three bubbles to vote "for" a chosen candidate.
3. Additionally, the voter must mark one (any) of the three bubbles for each of the other candidates in that race. Rivest dubs this marking action, voting “against”, to distinguish it from the conventional ballot process of leaving the oval blank for unwanted candidates.
4. Note that certain patterns of bubble blocks are not allowed: A ballot is validated if every candidate has one or two marked bubbles and no race has more than one candidate with two marked bubbles. A “checker” machine validates the ballot obeys the rules, adds a red stripe to the bottom, and slices the original ballot into three strips.
5. Once red-striped, the all 3 strips must be cast. Each of these is now called a ballot or ballot strip.
6. Each separate ballot strip has a different, untraceable ID number.
7. The “checker” machine gives the voter a copy of one of the strips (her choice which) to take home.
8. To cast the ballot, the voter inserts the 3 strips into a “dumb” optical scan machine which increments the tally of a candidate for each filled oval for that candidate. It has no idea which ballot strips went with each other.
9. All of the cast ballots and their ID numbers are published, accessible to everyone. Which ballots were part of the same triplet is not revealed. And the ID numbers are not correlated in any way that would allow this.

For clarity we note that Rivest's term "against" is a bit confusing when first encountered. It literally means filling in just one column oval to indicate the vote is not “for” a candidate. Since this requires action rather than inaction some verb was needed for the action of deselecting the unchosen candidates. As far as the tally machine is concerned every filled oval is a positive vote, so there’ no concept of a negative vote arising from a mark. Notice that because an oval is filled in even for candidates whom the voter is not voting "for", the final totals for each candidate that the tally machine reports is the number of "for" votes plus an overall offset of the number of voters. The latter offset can be subtracted-off if desired though that is not obviously necessary since it lifts all the candidates equally.

The primary concept of the system is that from any single ballot strip alone it would not be possible to determine what was on the other strips. Without those triplets its would seem that the voters choices for race cannot be determined. Thus it would seem safe to allow the voter to keep a receipt showing the pattern of one of the strips. The existence of this receipt is fundamental since it provides all of the provable verification features of the triple ballot.

3.0 Posing the problem

We wish to test the hardness of the inverse problem of reconstructing the “for” votes corresponding to any given receipt and the public library of all ballots cast. Moreover, if we can do this reliably for any given receipt, which after all is just one of the published ballot strips, we can iterate this over the entire set of published strips to determine the “For” votes corresponding to every ballot in the library.

Originally we had planned to measure the success of the algorithm against real-world ballot conditions, but this proved too easy to crack. For example, in Bernalillo County New Mexico ballots tend to be in the range 60 to 90 questions/races, there are multiple candidates in many of the races and a given legislative district will see vote levels as low as 2500 (and as high as 15000). It turns out that with that many races and that turn-out, the inverse problem is trivial to solve in nearly 100% of cases. Inversion only begins to get difficult at less than a fifth of that size ballot size and four times that minimum district turnout. For example, the race is well cracked even as low as 14 races, and more than 10,000 voters, with limited multi-party contests (e.g. 2 five-candidate races, 3 four-candidate races, 4 three-candidate races, and 5 two candidate races).

Therefore, we will instead examine how small the ballot must be and how many ballots must be aggregated before it becomes resistant to a crack. We also explore what assumptions the answer depends upon. Then we will reflect on the further implications of how external conditions (e.g. vote selling or data leaks) weaken the remaining intrinsic security.

4.0 The Crack

The crack relies on the recognition that not every possible binary pattern of three strips is a valid triple ballot, since it would violate the marking instructions. Thus three ballot strips pulled at random from the public library will not necessarily form a valid triplet. Indeed it turns out that the probability is vanishingly small for ballots of even mild complexity.

If we consider just one race at a time, the potential "mates" of a given ballot that satisfy the triplet rules is a subset of the entire ballot population. Moreover, each race on a given ballot mates with a *different* subset of ballot strips. Only the ballots at the intersection of these subsets are a valid mates for the given ballot. The crack is not a statistical inference but relies on strict logical elimination and so it is not guaranteed to deliver a conclusive answer in every case since accidental collisions could occur. However, it turns out that under a wide range of practical conditions, this intersection yields a single unique triplet, and even when the set is not unique, it can occur that all the possible triplets in the set correspond to the same voting pattern. So even if you don't know the exact mates, you may still know the candidates the voter voted "for". Lastly, even when a "for" vote cannot be uniquely determined, the intersection set of possible mates is often a very small fraction of the total number of ballots, with many of it's possible "for" votes ruled out and some of the races uniquely determined.

How one ballot restricts choices on another ballot.

Let us first look in detail at a single race with two choices. As a lexical notation, I will designate '1' to indicates a marked oval and '0' an unfilled oval. We might write a valid triplet for a two-way contest like this (1,0),(0,1),(1,0), where each group would be a vertical column on the ballot (which in turn will become one of the ballot strips). There are three ballots so we have three pairs. Summing up the three pairs element-wise gives, in this case, (1,0) + (0,1) + (1,0) = (2,1). So that is read as two-filled ovals for the first candidate (a "for" vote) and one filled oval for the second candidate (an "against" vote), and so that constitutes a valid ballot.

Switching to the inverse problem, suppose, we knew that one member of a valid triplet was (1,1), does this limit our selection of the other members? Clearly, yes, since we cannot have any another member with (1,1) as that would be at least two votes for both candidate, which is not allowed . Another invalid triplet is (1,1),(1,0),(1,0). Indeed, for this case of a set containing (1,1), the only valid triplets that vote "for" one of the candidates and votes "against" the other,) are the six permutations of (1,1), (1,0),(0,0) and six permutations of (1,1),(0,1),(0,0). There's one more legal ballot pattern, the three permutations of (1,1),(0,0), (0,0), which corresponds to a vote for "against" both candidates. (that is, simply voting for neither). This means that only 15 permutations out of a possible XXX available in the public library would form a valid triplet for this race.

4.1 The inverse process

Beginning with a "query" ballot we want to reverse engineer the intended vote. Conceptually, the first layer of the process is simple. Consider every pair of the published ballots, and see if they form a valid mate in every race for the query ballot. So for N ballots, this is an N-squared comparison. This produces as small subset of possible ballots, and indeed often solves for a unique answer when it exists.

Sometimes this process falls short of revealing the unique answer or maximally resolving the ambiguity when there is no unique answer. However one can take the analysis to a second layer in a hierarchy. Namely, repeat this process for every ballot, reducing it to either a unique answer or set of possible triplets. One can then apply a “soduku-like” logic to find a smaller, self-consistent set of triplets (i.e.. integer programing).

The most common secondary analysis situation occurs when one of the ballots of a triplet leads to an ambiguous set of triplet but the other members lead to unique triplets. That is, if (**A,B,C**) were a latent true triplet, we might find by the initial analysis that our “query” ballot **A** can form legal triplets containing ballots (**B,C,X,Y** and **Z**) and thus it’s true triplet is ambiguous. But it may turn out that **B** can only form triplets with just **A** and **C** which we will find from the intersection analysis of **B**. Thus when we see **B** forms a unique triplet and is mated with **A**, we now have resolved our “query” ballot **A**’s latent triplet. Furthermore, by now eliminating ballots **A,B** and **C** from the system, other ballots with ambiguous sets that contained those (e.g. **X,Y,** or **Z**) might now become unique, and so the process may cascade. Notice that this analysis is still an N-squared level of complexity

There are hierarchically many other logic operation to apply. For example, if I merge the intersection sets of any two ballots and the union set has less than 6 ballots, then I know that the two query ballots must be the true mates (proof left as an exercise left for the reader’s pleasure).

5.0 Experimental results

I partially implemented this method in the language Python. The elimination algorithm finds the intersection set for a query ballot. If that set is ambiguous, it subsequently treats each of these as queries, and removes any unique sets found. The implementation does not recurse further, nor does it apply any hierarchically higher logic. For the circumstance I considered below, it turned out that normally either the first level recursion was sufficient or there were so many members in the ambiguity set that I believed that further analysis was not likely to noticeably reduce the ambiguity. (Of course, that lazyman’s hunch might be wrong). Consequently, the experimental results I give for the minimum ballot sizes required for the crack to succeed are upper bounds since more sophisticated analysis could only lower these bounds. Additionally, these must be considered upper bounds because other environmental factors I discuss below will also lower these bounds.

It will make it a bit easier to present the simulation results if we reveal the key finding now: the solution process behaves cooperatively so that for a given set of conditions, it tends to either crack nearly all of the queries or none of them. The dividing line between those regimes is dominated by the number of contests on the ballot and the number of candidates in the contest. The geometric dependence of the dependence on these is like an abrupt switch. Up to a certain number the contests the vote cannot be reverse engineered, then, give or take a contest or two, nearly all of the ballots can be reverse engineered in nearly all elections.

As a result of this abruptness, the simulation I will describe is extremely insensitive to most of the environmental parameters I considered. For example, the difference between a highly polarized party preference and random voting did not change the location of the switching point by more than one contest. Likewise the affect of increasing the number of voters is logarithmically insensitive: raising the number of voters by factors of ten only moves the transition point higher by a few additional contests. The one parameter that does matter greatly is the number of candidates per contest. Going from 3 man races to a 5 man races, can roughly halve the number of contests needed to reach the transition point where the crack nearly always succeeds.

5.1 Simulation conditions.

I ran the simulation varying the contests-per-ballot, contestants-per-contest, and number of voters. For these I would simulate 30 elections, and then randomly select a dozen query ballots to solve for the voter’s candidate choices. The

		100 Voters (machine/precinct aggregation)																		
		Number of contests on triple ballot																		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Candidates per contest	**2**	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	3	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	4	🟩	🟩	🟩	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	5	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

		1000 Voters (legislative district aggregation)																		
		Number of contests on triple ballot																		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Candidates per contest	**2**	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟥	🟥
	3	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	4	🟩	🟩	🟩	🟩	🟩	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	5	🟩	🟩	🟩	🟩	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

		10000 Voters (state-wide aggregation)																			
		Number of contests on triple ballot																			
		6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Candidates per contest	**2**	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟥	🟥
	3	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	4	🟩	🟩	🟩	🟩	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	5	🟩	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

Key color	Chance of cracking
🟩	Less than 10% chance
🟨	Transition
🟥	Greater than 90% chance

Figure 1: Yellow breakpoints shown are optimistic. Actual margins may be much lower for reasons discussed in the text. Based on preliminary data Oct 8 2006--Subject to refinement in final draft.

results are shown below in three tables. The green zones are where the crack mainly failed and the red zones are where the attack mainly succeeded. The simulations were run for different numbers 300, 3000, and 30000 ballot strips. These levels were chose since they crudely correspond to the number of voters at various levels of aggregation that are in need of protection. So to protect the votes in a single voting machine one needs to protect down to about 100 votes. To

Affect of Party Bias on Election with 7 races, each with 5 contestants, 30,000 ballots

percentages are probability a ballot can be connected to its corresponding vote

	Fully cracked	Partially cracked	Not cracked
No Party Preference	18%	21%	61%
Two major parties, weak loyalty: deviate with 50% frequency	17%	31%	52%
Two major parties, strong loyalty: deviate with 10% frequency	45%	27%	28%
One major party, strong loyalty: deviate with 10% frequency	90%	8%	2%

protect the votes on a given legislative district, or ballot style, one needs to be able to protect down to roughly 1000 votes. And to protect state offices the ballot needs to secure at the 10,000 vote level.

Random and Biased voters:

I considered the case where the voters are fully independent (random) and the case where they each voter's responses in different contests are correlated. In the random case, the voter chooses chooses among each candidate (or abstaining) with equal probability. All of the above results were for such "Independent" voters.

For the biased case I created a simplified model of the normal environmental situation where their a voter has a party preference and this introduces a correlation between races: if he votes for the candidate from party 1 in one race then he will more likely vote for candidate 1 in all the races. The bias was implemented as follows. The voter is assigned a party preference and probability he will pick deviate from his preferred party. When he deviates he chooses with equal probability amongst the other contestants or abstaining in that race. I simulated both a single major party and two equally strong major parties (half the voters strictly assigned to one or the other party). That is, each voter belongs to a major party and deviates from the party line on a certain fraction of the races; in this simplified model, there are no members of minor parties, instead minor parties get their votes from deviating major party voters. This model is intended to demonstrate the affect of correlations, not to realistically model the environment, thus I avoided introducing an ad hoc model for ranked party preference.

It turns out even a heavy level of voter preference correlation had a minor effect in the dividing point between green or red zones on in the maps above. I therefore present the results for simulating party bias in one of the yellow transition zones. In the table below "partially cracked" means the voter's true vote for at least one of the 7 races was left ambiguous on a query ballot in which some of the voter's votes were uniquely identified. (Note: In the green or red zones, the fraction of partially cracked ballots plunges to near zero: either all the races get cracked or none do)

In the case of the biased election, counter-intuitively, it becomes easier to crack. My expectation had been that increasing correlation would make for more ambiguous collisions. It appears that bias also forces the set's vote-space support to be more restricted and thus has fewer degrees of freedom to attack. Therefore the results for the random case should be considered upper bounds on the number of required races to crack, since environmental bias does exist. While the effect shown in the table is dramatic, even the strongest bias did not budge the transition point by more than one race. That is, a strong bias required perhaps one less race on the ballot than for uncorrelated random case (6 instead of 7) to be highly crackable.

Other environmental correlations:
I did not simulate the notion that perhaps a voter would not choose random patterns for the three ovals to mark his "For" or "Against" votes but instead might always use the same pattern. For example, always voting "Against" using, say, the horizontal pattern 010 and never 001 or 100 to fill in the three columns. I will return to this point later.

6.0 Discussion

6.1 The Trouble with Quibbles

First lets raise and bat down a series of straw-man quibbles with this analysis.

- a. *The bias you chose is not a good model since voters tend to vote off-parties closer to their preferred one rather than randomly. You need a ranked preference model.*

Answer: If the security of the triple ballot depends upon an uncharacterized hypothesis about voter preferences it's not a good security model. The legitimate concern here is that the votes for off-major parties offer degrees of freedom in which to embed a distinguishing signature that might improve the odds of a crack. The convincing counter argument is that the results don't change substantially as the bias approaches 100% for a single party (so there are almost no off-major selections and thus off-major characteristic signature). Moreover, the point of the model was to show the effect of correlation is counter intuitive, and lowers the safety of the triple ballot.

- *b. The 5-man races are not fair. How many realistic ballots are 100% 5-man races?*

Answer: Again, if the security of the triple ballot depends on that, we should not adopt it as standard. Moreover, while it would be unusual for a 60 race ballot to be all 5-way races, it would not be unusual for a 60 race ballot to contain say half a dozen 5-way races. That would make it insecure at the precinct, legislative district and state-wide aggregation levels.

- *c. Your aggregation levels are too low. You need to boost this.*

Answer: First, remember the gains with aggregation size are logarithmic: going two orders of magnitude up from ten thousand to to a million would only add a small number of additional contests it could support before the onset of cracking.

Second, How? In the real world, voters in different districts see different ballot styles, so unavoidably the published ballots will be segregated by district and the number of voters using a given ballot style may be quite small. Thus a reasonable choice for a feasible aggregation requirement is perhaps 1000 and indeed that might be generous. The proper number can grow or shrink depending on what sort of ballot aggregation is allowed. In a given election, the effect of intersecting legislative districts, school districts, and so forth can cause a proliferation of ballot styles such that the number of voters voting on any one ballot style may be near 1000. Additionally, it's the common practice never to mix, physically, ballots that came from different precincts or even voting machines. Thus even if the the entire pool of a given ballot style were aggregated on the public website, the ballots themselves perpetually remain segregated physically according to which machine they came from. If there is a risk of exposure of the ballots at the machine level (and there certainly is a risk, particularly during recounts), then the proper number to consider for the security model would drop to a couple hundred. Conversely, one could argue that one intends only to aggregate and publish races that are federal. In this case the published aggregation level can be higher. However, while federal races are more headline grabbing, it's nearly certain that election cheating is more effective at the legislative district level where smaller conspiracies are more likely to evolve and the number of ballots to disguise far fewer. Thus it is reasonable to ask if we can protect races in modest sized aggregates as well.

- *d. Real voters won't mark the triples in random way, they will have a "system".*

Answer: First, as before if the security model depends upon the truth of that assumption then it's a dubious security model. At the very least, if that is a requirement, then we need to change the instructions to the voters so they won't make random marked triplets. As pointed out in part one of this article, the marking requirements are already nearing prohibitive complexity to pass the "checker" without an error, and new rules are already needed to handle oddball exceptions such as for write-ins. Adding more pattern rules will not aid matters.

Next, it's not clear if this will undermine my analysis. After all focusing voters by party bias made the problem worse not better. It might turn out that some "systems" are worse not better.

Moreover, it's actually hard to think of marking pattern suggestions for the voters that would not cause other problems or be too complex. For example, if it were suggested that everyone make their "against" votes in column 1, then life just got easier for vote sellers who can now select unusual patterns to make their ballots even more distinctive. If we make the advice obligate, then that knowledge may help the inversion, or asking for receipts on certain columns may reveal the vote. While I can devise marking rules that if all voters followed them it would defeat the general case algorithm I used, on the other hand if they are obligate then I could use that fact to aid the inversion with a special case algorithm. Also the now-predictable relationship between the columns might lower the risk getting caught when using probabilistic vote manipulation attacks such as those suggested by Andrew Appel. It's a serious point worth dwelling on but specific proposals are need for specific answers.

6.2 Enhanced threats to the secret ballot

It should be carefully noticed that among the various reasons for the secret ballot are the prevention of vote selling and the protection of people with unpopular beliefs. In both categories, one will also find people who make unusual vote patterns. I believe it will be the case that these will be easier to reverse engineer than for people who vote with the majority. If so then then thresholds for protection needed are lower than random results suggest. I did not model this.

I did not take any *a priori* statistical knowledge into account, but it's easy to see how this could be an adjunct to the elimination process that would further reduce requirements for a successful crack inferrence. For example, when an intersection set is not unique, the number of members in the set may be quite low even in the "green" zones. If one is knows that a particular class of voter is predisposed to vote a certain way in some of the races, then it is plausible one could infer relationship in this the small set. Again this is likely to most impact voters making unpopular choices, whom are in most need of the secret ballot. As a second example, I note that in the simulation of party bias, I did not use any knowledge of that bias when I inverted the ballots. If I had a wanted a statistical answer rather than a unique answer, I could have incorporated this knowledge. The likely outcome would be that I could identify a voter's vote with high certainty with much fewer ballots than the exact solution requires. Again I would expect this to be most pronounced for the voters making unpopular choices.

6.3 Rescuing the Three Ballot System

It seems apparent that the triple ballot cannot successfully encrypt even a relatively short ballot and even in the presence of a large number of voters. Yet we know it should offer some protection at the single race level. As the number of races grows the likelihood it can be cracked grows geometrically. Thus for a given number of voters, there will be a sharp threshold in the number of races it can protect with high likelihood. The problem is not easily remedied by going to larger aggregation sizes since the threshold only varies logarithmically with this number.

To apply the triple ballot in a practical election several things must be considered. First, the smallest number of votes that will be maintained in an unaggregated state must be determined. This is likely the voting machine itself. If the ballot size in use exceeds this then severe administrative controls must be enforced to protect ballot secrecy. One might wish to only use the triple ballot for offices that only occur on state-wide ballots to increase the possible aggregations size in the public ballot library. Alternatively, one might wish to chunk the ballot in to multiple sub-ballots. So for example in a 80 race election one would have perhaps 16 triple ballots (5 contests in each chunk), for a total of 48 ballot strips. This would present logistic complication since we would have to assure that every strip was cast.

An obvious solution is rather than cutting ballots into ever finer partitions simply have the tally machine do a virtual decomposition into chunks and strips, and the generation of receipts. The problem with this is that one of the virtues of the triple ballot system is that it avoided a lot of the need to trust the tally machine software. By splitting the jobs between the voter, the checker and the tally machine the original triple ballot system avoided software conspiracy and enhanced transparency. In real elections, it is more valuable that the voters can see for themselves that intrisically cannot be a software conspiracy, than elaborate policies out of the voters view are supposed to assure that there is no conspiracy. Thus giving up this separation of function is no small matter.

Rivest has also suggested a system where voters exchange receipts. The idea to erase the connection between the voter and his votes by giving him someone else's receipts. This is an important improvement since it mitigates attacks on the secret ballot that would use the receipt itself. It does no however solve the problem of coercion and vote selling, since one does not need a receipt for that. (Nor does it reduce the problem of vote manipulation discussed in part 1 or by Appel.) The vote seller need only report either the IDs of their votes or the specific three ballot combinations for a handful of races and as this analysis shows that can identify the triplet exists in the public record. The probability that a faux vote seller could pick three patterns and have that triplet exist in the library is vanishingly small.

It is however worth noting that the triple ballot solves one of the issues that defeats other forms of ballot encryption. Coercion and vote selling are highly similar but difference in few important characteristics. With coercion, retribution can occur long after the fact. Thus as long as their is a chance that the encryption keys for a past election could be leaked at any time in the future, and thus expose past votes by the voter, there is a plausible threat to the freedom of the would-be coerced voter. For example, a prominent political figure might be blackmailed long after the fact when a data leak shows he, heaven forbid, voted for the baby-seal-killer party. Rivest's model avoids this secure key escrow problem, and as long as the number of races sufficiently small the security can be permanent.

7.0 Method:

Function: Given a selected ballot that contains N races, find the possible mates, and recover the vote. Our strategy is to first perform some order-N screens that eliminate many ballot as consideration as mates. Then on this reduced ballot count we perform the more time consuming order-N-squared operation. This reduces the potential ballot mates to a small intersection set, and often solves the problem if the set is a unique triplet. At this point we perform the meta-level search on the members of this set. Namely, all of the members of this set are treated as query ballots in their own right and we look to see if any of them form unique triplets.

0. Select a ballot to query for it's mates.
1. Phase 1: (worst-case: order N)

1. Start with the first race:
2. find all the ballot pairs that would be legal mates for that race.
3. keep these, discard the other ballots
4. iterate over rest of the races and remaining ballots.

2. **Halt if unique triplet remains**
3. **Phase 2: (worst case: order N)**
    1. For each race, compute all the pairs of patterns that could form valid triplets
    2. iterate these, checking to see if ballots exist that could satisfy the pair for that race.
    3. discard any ballots that did not meet the criteria for at least one pattern.
    4. iterate over the rest of the races and remaining ballots.
4. **Halt if unique triplet remains**
5. **Phase 3: version 1 (Worst case: order N-squared)**

```
1. loop over all ballots
2.         loop over all ballots (upper diagonal only)
3.                 loop over all races:
4.                 if ballot pair forms valid triplet in all races with query ballot,
5.                 add this to the intersection set
```

6. **Halt if unique triplet remains**
7. **Phase 4: (quasi-N-squared):**
    1. repeat phases 1-3 for every member of the intersection set
    2. capture any valid triplets and remove them from all the other intersection sets
8. **Halt if query's intersection set if now unique.**

7.1 Optimizations

In practice phase 4 is skipped if the size of the intersection set from phase 3 is too large to make it worthwhile. By introducing that cut off and nor reusing further phase 4 is quasi-N-squared.

In practice there's some significant optimizations of phase 3 that can accelerate the search. I switched the order of the two inner loops. This requires more bookkeeping but the payoff is that once a valid triplet is found we can truncate the remainder of inner loop over ballots, since the ballot is already added to the intersection set.

Finally, if one is planning to invert the entire election rather than a small selection of query ballots then it appears that there may be a different approach can keep the process down to N-squared rather than N-cubed. However I have not validated that approach yet. The problem is that the bookkeeping required to implement is memory intensive and Python is a poor choice for large memory applications.