THE PIMA COUNTY AZ DEMOCRATIC PARTY'S PROPOSED
TEST PROTOCOLS FOR DIEBOLD DATABASES
by Jim March, 5/27/08

This document outlines what we want to do with Diebold central tabulator databases obtained under public record laws. This document is in two parts: the first part outlines in plain English what we want to do, the second represents a semi-formal set of program specifications for an automated test tool to assist with the process ad goals outlined in the first section.

(The various notes on each test are in some ways a draft edition of the final manual for the proposed application. While I cannot be of much help in coding it, I am very well qualified (perhaps uniquely qualified) to assist in writing the final user manual for the tool.)

Part One: The Post-Election Database Test Process And Goals

1) AUTOMATED INTEGRITY CHECKS: We would take all iterations of a given election from the initial cycle L&A to the end, and run an automated comparison tool to examine whether the "frozen tables" have changed and if so, how. "Frozen tables" refers to the fact that the election setup information (ballot layouts, candidate IDs and the like) are legally "frozen" beginning with the initial L&A (Logic And Accuracy) testing. The tool would report changes from one database iteration to the next and human eyeballs would then determine whether they indicate problems. See below for the program design specifications of this tool. We anticipate spending $3k or more on programming for the tool, but once built it can be run against any election set. We told the AG's office this sort of thing was necessary before they hired iBeta[1] to test the May 2006 Pima County bond election databases (the "RTA election") to no avail.

2) MAIL-IN SANITY CHECKING: We would take all data files from a given election that record mail-in vote totals and chart the vote totals in every race. To understand this test, note that in Pima County the mail-in vote is processed in a "homogenized" fashion: ballots stream in from all areas of the county blended and randomized. That in turn means that candidates seldom "jockey for position" during mail-in processing - their vote totals tend to increment very smoothly as compared to election-night returns on a statewide race. In a statewide race, as a liberal county like Pima reports numbers the trends shift left, and when Maricopa numbers come in they swing right. The Pima mail-in ballot numbers aren't like that – being "blended", sudden swings indicate trouble of some sort, except for rare cases where a candidate's scandal breaks out pre-election or major ad money floods in all of a sudden. Such events can be accounted for with a bit of digging.

3) CHECKING BOTH SETS OF BOOKS AGAINST EACH OTHER: The Diebold databases write all vote totals to three tables at once, which violates some very basic database management principles. We don't know what the third table does, but we understand the first two: one reports results on a precinct-

[1] Plaintiff's consultant Jim March notes a story his father told regarding his job as a machinist working on jet airliner parts: on several occasions a job rated at six hours would come through, and his father would spend a couple of hours building a better tool for that job and three or four hours actually completing the task. Thereafter, he could do that job in about half of the "rated" time and would submit the tool and revised process for official review. We see this is being the same situation: build the right (software) tool for analysis and then do faster and more complete reviews as data is available. The Arizona AG's office and iBeta didn't take this attitude.

by-precinct basis, another reports the county-wide vote totals. We know that the following scenario is possible: the database can be rigged such that county-wide totals are always wrong, but a query for a precinct's totals will always be right. It appears to have been designed to facilitate fraud – an honest elections official could compare individual hand-counted precincts with official tallies all day long with no error found, yet the election is in fact hacked "county-wide". We know the name of the GEMS product manager at Global Elections in 2000 when this "feature" first appears: Jeffrey Dean. We are quite concerned about Mr. Dean's $425,000 felony embezzlement conviction for computer aided fraud in the late 1980s in a sophisticated scheme involving computerized accounting. In any case, we CAN perform this check on the final set of data the court has agreed to provide for the '06 Primary and General. Checking across multiple databases for the same election with this process would be more complete, especially as manual entry corrections of any sort have the ability to "auto-reconcile" the three data sets for the precinct being manually edited if they're not already in agreement. This feature appears to be set up to eliminate evidence of problems if somebody checks their work after manual entries, as is human nature.

4) INTERNAL TIMESTAMP ANALYSIS: Each table inside a Microsoft Access database has it's own set of timestamps. These are unaffected by operating system-level copy commands. Unfortunately they're affected by Diebold's .GBF compression process. Since they are, we won't be able to see if the "legally frozen tables" have been altered after the date at which they legally freeze (the initial L&A) unless we can convince counties to save time slices via Windows-level copies of the MDB files which may or may not be practical[2]. Where possible at all, this check can be performed on the final end-of-period table. However, timestamps can be falsified by altering (back-dating) the computer's date and time and performing an operation in order to change the timestamp on a table. If this operation is done in MS-Access it won't leave an audit log entry (the GEMS audit log only increments on GEMS operations, not MS-Access or other illicit tampering external to GEMS). It would be more difficult to fake the timestamps across all the iterations of the data for a given election, therefore this test too would benefit from analysis of all the data files for a given election.

Program Design Specifications for the "Diebold Automated Manipulation Notary Tool" (working title).

PURPOSE AND SCOPE

The purpose is to provide an automated integrity checker program ("tool") for election databases produced with the Diebold GEMS central tabulator software. An initial version of the tool can be command-line driven with the user inputs as command line parameters, or the whole thing can be menu driven in a graphical user interface. It should operate in any common language that can read from an Access database. It must be completely open source, and it's own data and reports must be readable in free tools such as the OpenOffice database and/or spreadsheet or open-source equivalent. It should not be necessary to own a paid-up copy of MS-Access to use the tool. Microsoft provides a free Access database viewer for Windows and equivalents that runs under Linux exists.

Running under Windows is (sigh) preferred but cross-platform compatibility would be very welcome. If we have to use Linux, maybe building a LiveCD for this purpose would work?

---

[2] It may not be possible to use the operating system to copy an MDB file that's still in use by GEMS – and if not, asking them to pull out of GEMS each time a copy is made will at a minimum meet resistance in terms of harming the "flow".

The tool will take all the versions of the data for a given election from the initial L&A test to the last and perform a series of automated integrity checks against them. It will create it's own database, write out it's findings to that database and create a series of reports based on them.

The tool's purpose isn't to "detect fraud", but rather to detect places in the data warranting further "human eyeball study". We do not think there is an alternative to intelligent human analysis; rather, the tool is about sorting the "wheat from the chaff" and assist an informed human in looking for potential manipulation or corruption in Diebold GEMS data.

The core purpose of the tool is to make sure that elements of the databases that are legally not supposed to change through the election cycle (after the initial Logic & Accuracy "L&A" test) do not in fact change. It does so through internal timestamp analysis and comparisons of table data. It starts out "knowing" (through user input) which tables are known to increment in a properly run election – the audit logs and vote totals for starters, and possibly more we'll learn as we go – and then looks for changes in what's not supposed to change according to the law.

Finally, it will take the vote totals tables (both main ones at a minimum, the third if possible) and compare them – alerting on any data iteration that doesn't match.

In some cases we anticipate the tool's reports will be created by local election integrity activists and then Emailed to experts elsewhere to get a "first impression" as to the data's integrity.

RUNTIME CONDITIONS

1) Any .GBF files from a given election cycle will be converted to .MDB prior to the execution of the tool by the users - the tool need only cope with .MDBs. (Taking apart GBF files on the fly would be a "version 2" feature – since .GBF is Diebold's variant of the old Pkware .ZIP compression system it should be possible with some additional R&D.)

2) All of the files for a given election cycle will be loaded into a given directory ("folder") prior to execution, with nothing else in that directory.

THE TOOL'S LOGIC

a) First, the tool will be designed so as NOT to alter existing databases. It will create it's own database to hold the data it collects and perform reports.

b) The tool will accept the target directory of the data as it's initial input (command line or menu driven).

c) The tool will accept the names of tables that are "known to grow" (audit logs, vote totals, others?) as it's second input, command line or menu driven. If we learn through running the tool that other tables also increment through the Diebold process as normal procedure, we can start over and add the names of those tables to the tool's input so as to reduce the number of "false positive hits" (for changes) later. (In other words, as we use the tool and analyze the databases with it, we should learn enough to

improve the automated process if need be.)

d) The tool will also accept as user input the known date and time of the initial L&A test.

e) The first piece of data recorded from the GEMS databases will be their filenames and timestamps for file creation, modification and "last accessed".

f) The second piece of data the tool will extract is the name of every table. These will be recorded to the tool's database, and then each iteration of the GEMS data will be probed to make sure it has all of those tables and no others – in other words, that all table names match.

g) The third piece of data extracted from the GEMS databases will be the timestamps for each table in each database.

h) At this point, a report can be run listing the data files in their apparent order per their time and date stamps. A human can look at these and make sure they correspond to the filenames ("Early day one", "early day two" and the like). If things are out of order, that may be a clue that poor data management practices (at a minimum) are going on and (possibly) vote-hacking.

i) Next, the user can ask for a "table timestamp sanity check" – it can take the names of tables that it suspects shouldn't change (as the "known to change" tables are already in it's data and can be excluded) and report the time and date stamps of any table in any database that appears to be after the initial L&A time and date. Any positive reports (listed by database name and table) should be checked with a human eyeball to see if this is a table that should be frozen post-L&A and if it is, do the time and date stamps indicate improper tampering? This is a reporting function separate from any other report.[3]

j) The user now tells the tool to compare all tables suspected of being the frozen type (in other words, all tables except those the tool has been told normally change) and compare them across time, reporting where changes occur by filename, table and line number. This will ID possible tampering in the "frozen tables". (If a given table has a large number of changes, that should be analyzed by a human to determine if that's proper GEMS behavior and if so, add it to the list of "don't alert" tables. Once we learn more through the use of the tool about how GEMS works, the tables excluded from this test can be hard-coded into the tool.)[4]

k) Finally, the tool should have been pre-programmed with the tables known to contain each of the multiple sets of vote totals (already understood by the community of people studying these things) and for every database iteration, report whether or not vote totals match. Where they don't match, list the table names and line numbers of each discrepancy for human review. (If they don't match at one point in time but do match later, a human needs to look at it and ask if the database somehow "broke" and was manually corrected later. It would also be useful to make sure any human corrections were accurate, and recorded properly in the audit log. If the audit log doesn't show corrections, they were made illicitly outside of the normal GEMS process (MS-Access or equivalent).

---

[3] If the MDB files have been extracted from MDB files, this step won't work. But we should still code for it; in all elections the FINAL database should be an .MDB file that was never crunched into GBF, which means the table timestamps will be accurate in that file only. Hence the tool should be able to do this particular test against a given single MDB file as an option.

[4] This should be considered the single most valuable element of the tool.

Note that storing the votes three times violates some very basic data management practices on Diebold's part and is suspected of being a deliberately "hack friendly feature". We know what two of the three tables do: one reports vote totals if you ask the Diebold software for a given precinct's vote summary while another is polled for county-wide totals. Let's say an honest elections official/staffer "smells a rat" - he hand-counts the votes for a precinct as a spot-check and compares it to that precinct's report. It can come up good (over and over again) while the county-wide totals are in fact hacked. The product manager for this stuff at Global Election Systems (pre-Diebold) had a previous felony embezzlement conviction of over $425,000 in which he used a rigged accounting computer...kinda makes us wonder...?

LICENCING:

The test tool will be “public source”, probably under the GPLv2 (Gnu Public License, second edition). GPLv3 may not work as it bars interaction with certain types of closed-source applications and/or copy protection methods. In short, the added complexity isn't worth it for this application.