Scytl

Innovating Democracy

Scytl iVote Voting System

Response to “NSWEC-10 Review of the Revised iVote 2021 System”

by David Hook and Carsten Schürmann, July 2021

September 2021

Final

Scytl – Secure Election Technologies

Table of contents

1 Introduction

Reviewers from the DEMTECH GROUP (the Reviewers) performed a source code review in June 2021 on the new release of the iVote®[1] voting system, as part of the planned security assurance processes. This new release (1.8.5) is an evolution of the iVote voting system used in the 2019 state election. As the Reviewers mentioned in their report, considerable effort has been made to reduce complexity of the software and implement some of the improvements identified by the Reviewers on the source code analysis made in 2019[2], as well as the findings detected in 2019 during the Swiss voting system source code review that could affect the iVote voting system (reference [15] of the Reviewers report); and the feedback received under the Scytl Online Voting Source Code Review Program following the 2019 State election[3]. The Reviewers still see ways for improving the iVote voting system, as explained in the document released after the last source code review: "Review of the Revised iVote 2021 System" dated July 2021 by David Hook and Carsten Schürmann (the "Final Report").

In this document, Scytl provides additional information and future considerations to the recommendations given by the Reviewers, as we did with the previous report released in 2019. The aim of this document is to complement the Reviewers analysis and recommendations with technical responses, to help the reader to better understand the context of the recommendations. As in the 2019 response, Scytl intends to assist with the readability and assessment by dealing with the topics relevant to Scytl in the same order as presented in the Final Report.

The feedback gathered from the Reviewers is valuable and useful for the improvement of the iVote voting system and Scytl's software, and can enrich areas such as the source code and technical documentation, whilst noting that these must be taken into account within the context of the contract between Scytl and its customer; the NSW Electoral Commission (NSWEC). The findings of the Reviewers and outcomes of this response will continue to be considered by Scytl during the ongoing development process as part of a continuous improvement program for Scytl's software.

[1] iVote is a registered trademark of the NSW Electoral Commission; the registration symbol will not be used throughout the rest of this document.
[2] "NSWEC-7 Final Report" by David Hook and Carsten Schürmann, January 2019
[3] Review of the attack described in the report "Faking an iVote decryption proof" by Vanessa Teague, Associate Professor, dated October 2, 2019.

Report summary:

In Scytl's opinion, whilst the Final Report did find areas of interest and discussion for both the Reviewers and Scytl, nothing significant was found relating to the security and integrity of the system.

As the Reviewers mentioned in the Final Report, a significant effort has been made to reduce the complexity of the 2019 voting system, address vulnerabilities and continue to manage existing risks.

As an extension of the prior report, Scytl acknowledges that areas of the code remain somewhat difficult to review and were challenging for the Reviewers, as detailed specifications are not implemented in a traditional document format, but are in fact tracked within the capabilities of Scytl's internal Jira system. As the iVote voting system is based on Scytl's generic voting system (Invote), this does complicate the review process, given that Invote's code contains functionalities that are not used by iVote voting system but by other electoral systems also supported by Invote. These can be seen in the codebase, whilst not actually being called at runtime due to the product-based approach taken by Scytl. For this reason, it is important to identify which functionality is actually used by the iVote voting system at runtime.

Scytl's management of code is an ongoing activity, and a significant task given the size and complexity of the Invote/iVote source code required to deliver the extensive capabilities of a secure online voting system. Other matters raised by the Reviewers are described in the response herein.

2 Scope and Review Methodology (Part 2 of the Final Report)

Reader note: The text contained in text boxes within this section is quoted from the Final Report.

2.1 Coverage info and lack of scanning tools

Due to the lack of good code scanning tools for JavaScript, no automated code scanning analysis for JavaScript APIs was conducted.

Some information which affects the ability to properly assess the quality of the software has been unavailable. The Vendor has not provided meaningful coverage information, despite request. The absence of information contributes to a lack of transparency that makes it impossible to asses the overall quality of the software.

Scytl's response:

Scytl provided the coverage for the plugins used in the iVote voting system and the coverage that is run in the base voting system product Invote, from where the iVote voting system takes the voting protocol. However, we agree with the Reviewers that there is still margin for improvement in the JavaScript plugins.

3 Functional Matching (Part 3 of the Final Report)

3.1 The Quality of the Documents Provided

We expected that the files included in the source code drop would give us sufficient information (1) to serve as design documents and (2) to provide us with enough information on how the different software modules interact. Unfortunately, this was not the case. When we reviewed the Interface specification [9], we found it to be under-specified and incorrect. For example, Figure 1 in [9] gives a good overview over eleven modules that define iVote. The document explains some of the interfaces, but not all. In the figure, interfaces for the voter portal and the verification app are depicted, but they are not explained in the body of the document. Some of the arrows in the figure carry different identifications than those used in the body of the document and some were not explained at all. In addition, there is no direct matching between Figure 1 and the implementation.

For some modules, no documentation was provided, for example the inVote system, an integrated solution developed and provided by the Vendor for organization public and private elections. When asking the Vendor for additional information, we were informed that it was unavailable. From the logs supplied with the source code, we learned that inVote's development originated around 2006, but modules for iVote were only added in November 2019.

Scytl's response:
Documentation provided to the Reviewers contained the overall iVote voting system specifications, but it did not contain implementation details. Detailed specifications are not available in a standard document format, as it is introduced directly in our software development management tool (Jira). Scytl understands that this does not help the review process, as it requires the Reviewers to have access to the Scytl Jira system in addition to the specification documents.

Regarding the history of the Invote voting system, it began in 2006, though there was also an earlier version in 2001. The earlier version (Pnyx.core) was based on C/C++ and Invote is the Java evolution of this voting system. The presence of references to older versions is normal considering that we are maintaining some backward compatibility with previous versions.

Scytl's product has advanced over many projects, evolving from the original Pnyx.core, and into sVote and then Invote, upon which the current release of the NSW-customised iVote voting system is based. In 2014-15, iVote voting system was developed as a standalone product, building on Scytl's core building blocks; the version the Reviewers have seen in their prior review. Following that time, NSWEC expressed an interest in moving towards a standard product, albeit with customisations to suit the NSW electoral model, which pushed Scytl into developing the iVote voting system on the Invote product line.

Recommendation: The Vendor should supply additional documentation for inVote.

Scytl's response:
Scytl takes note of the recommendation. Within our development process, specifications are written as documents, however we record the implementation details within our Jira system.

3.2 Verifiability Analysis

3.2.1 Complexity

Scytl's general comment regarding complexity:
As the Reviewers mentioned, Scytl has reduced the code's complexity compared with the previous iVote voting system version, though we agree that there is still work to do. Our goal in general is to reduce code duplication and unify dependencies where possible, which is balanced against the time and testing requirements that exist as a normal part of project delivery.

Recommendation: The build of the nsw-ivapi.min.js should be simplified to make it possible for reviewers to generate a file without duplication.

Scytl's response:

Scytl takes note of this recommendation.

The nswec_govlab module:

This module contains the source code of the inVote framework that is a central part of the iVote system.

Nswec_govlab_1_3_0-RC2

nswec_govlab_1_3_1

We observe that two Java files, and multiple POM files are different. A POM file captures the dependencies of modules on other modules and/or libraries. Two files are updated,

NSWImportVotersAndVotes.java

CsvManagerJdbcDao.java

and the differences in both cases appear to be meaningful. Which govlab module is in use?

Scytl's response:

There are independent components implemented at different moments, so there are cases in which these modules are built on different versions of the software. In this case, nswec_govlab_1_3_0-RC2 is only used in the invote-receipt-admin module, and the updated classes are used in other modules. The RC2 dependency is present for this reason, as a future release of the receipt admin module is expected to include an upgrade to the latest version of nswec_govlab.

The cryptolib module:

The library that implements the basic cryptographic operations, cryptolib,

cryptolib_2_4_1

cryptolib_2_7_2

show substantial differences. The only module depending on 2.7.2 is the secure logger. Assuming that 2.7.2 is more recent than 2.4.1, we wonder why doesn't the entire iVote system use the latest release? Is the 2.4.1 release still maintained?

Scytl's response:

This case is similar to the one before: different modules were implemented at different times and have dependencies to different versions of the same library. This does not mean that there is a maintenance problem as the cryptolib is an internal library, and therefore maintenance is under our control rather than dependent on third parties. In this case, an update was not done as there were no critical issues with the cryptolib functionality used by iVote voting system. Only the Secure Logger is using it because it was implemented later in the development life cycle. Future Invote releases will continue to update the cryptolib version as required in a supported manner.

> The jbasis_cryptolib module:
>
> Also the jbasis_crypto library appears at different versions and levels of maturity.
>
> jbasis_crypto_4_1_0
>
> jbasis_crypto_4_2_1
>
> jbasis_crypto_4_2_1_3
>
> jbasis_crypto_4.3.1
>
> In this case jbasis_crypto_4_1_0 and jbasis_crypto_4_2_1 appear largely the same, while a diff shows substantial differences they appear to be due to a change of formatting. jbasis_crypto_4_2_1_3 has a couple of differences from jbasis_crypto_4_2_1 and also includes git conflict messages in the source code indicating a failed merge. The differences between 4_2_1 and 4_2_1_3 appear to be in the XML parsing. jbasis_crypto_4.3.1 appears to add some use of generics with further formatting changes and a change to the method for outputting certificate PEM files but appears to be missing the XML changes from 4_2_1_3. Each module appears to be used at least once. In post election maintenance, the modules need to properly reviewed and the divergence that seems to have happened in 4_2_1_3 resolved.

Scytl's response:

Scytl agrees with the Reviewers that there should be a unified version of cryptolib, and has begun working on this, though there are still modules to migrate. Migration priority has been given to modules where migration has been prioritised to resolve potential issues. Scytl continues to work on the remaining modules.

The p7_cms module:

These modules are duplicates that appear at different version numbers and levels of maturity.

p7_cms_1_2_0

p7_cms_1_5_1

p7_cms_1_5_1_1

1.2.0 is substantially different from 1.5.1, 1.5.1.1 uses the latest version of the Bouncy Castle crypto library. It is also concerning the 1.5.1 is still in use with invote plugin_counting_tally_1_4_3_1, invote_plugin_counting_mixing_1_4_2_1 and invote_plugin_counting_cleansing_1_4_2_1, as it overrides the choice of Bouncy Castle to use bcmail 1.55, which is well out of date. This last issue is also concerning as the resolution of the transitive dependency on bcmail 1.55 during building may override the parent dependency on BC 1.68 resulting in the use of bcprov 1.55 which is also subject to a number of CVEs. 1.2.0 is well out of date and should not be present at all.

The scytl_math module:

scytl_math_1_0_1

scytl_math_1_1_0

The 1.1.0 adds two new methods to Bigintegers "class". We could not identify any other meaningful changes, which seems to suggest that 1.0.1 is unnecessary. Why 1.0.1 is present?

The nsw_commons module:

nsw_commons_lib_1_7_2

nsw_commons_lib_1_7_4

Differences appear to be related to the POM files. The Java source files are different only in the context of the copyright notice which has been updated from 2020 to 2021.

Scytl's response:

The duplication of these modules poses no risk to the iVote voting system, though future releases could address further complexity reduction.

Recommendation: The source code should be further simplified

Scytl's response:

As mentioned by the Reviewers, there has been considerable work done to reduce complexity by reducing the amount of modules, and it is accepted that there is further opportunity for simplification. Scytl sees value in reducing the complexity of the code in new releases, so that dependencies are unified to a single version of the same library for the modules that are using it. Scytl also notes that the product continues to be developed over time with the competing needs of including updated functionalities whilst maintaining compatibility.

The examples given by the Reviewers in this section are related to the fact that different modules are implemented at different moments and are therefore linked to the specific version of the library available at the time that it was implemented or updated. For this reason, they found that different versions of the same library were present on different modules having dependencies to the same library. Priority is given to migrating modules that address potential issues. This work is ongoing.

3.2.2 Explicit Erasure of Votes

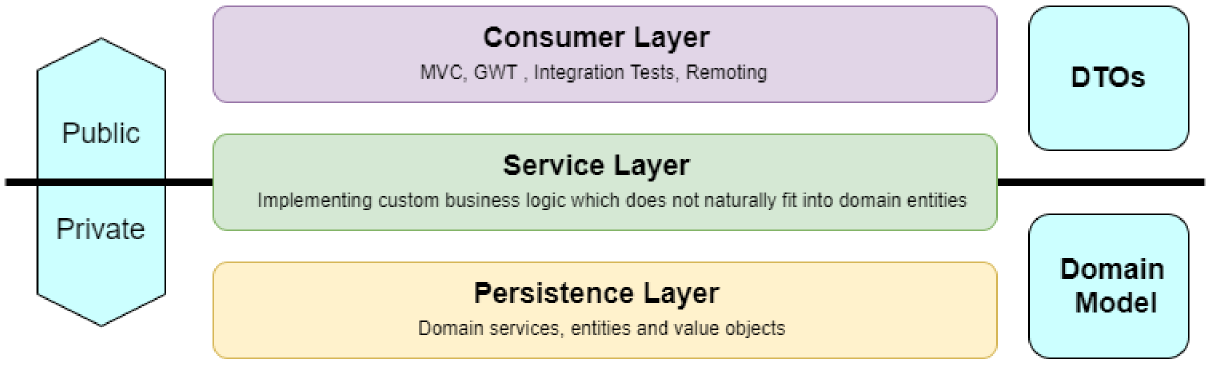
Recommendation: The Vendor has responded [16] that all naked SQL calls can be regarded as safe or as dead code. If the code is dead code it should be elimintad by the Vendor in future.

Scytl's response:

As the Reviewers mention in the recommendation, in our response related to this matter [15] and [16] Scytl has explained why there are queries that allow the deletion of information: these are only present in the voting system configuration back office as part of the functionality used by election managers to maintain and decommission election data (i.e., these queries are not present in the voting portal). Scytl also provided responses to the four questions raised in this section as follows:

Transaction management:

Transaction management is implemented at the service layer, as the service layer encapsulates and implements the business logic and is therefore the layer that performs the functional requirements of the system. When the business logic requires interaction with different entities and data sources (e.g., different tables in a database), the service layer ensures that there is an atomic transaction to avoid data inconsistencies among the differing data repositories, in the event that one transaction fails at the persistence layer (ie: upon SQL query execution).


Consumer Layer
MVC, GWT , Integration Tests, Remoting
Service Layer
Implementing custom business logic which does not naturally fit into domain entities
Persistence Layer
Domain services, entities and value objects
Public
Private
DTOs
Domain Model

This implementation removes the risk of inconsistencies due to transactions being improperly managed, such as in the case of a naked SQL failure.

Verifiability:

To allow detecting the deletion of any vote if any queries are executed, a verifiability strategy has been implemented in the voting system, which takes into account the following three main objectives: log relevant data transactions to allow a complete review of the election (completeness), protect the log information from manipulation (immutable logs) and provide individual verifiability means to voters (voting receipts). These measures do not prevent deletion but ensure that if a deletion occurs, it can be detected. Prevention is based on access control and security controls implemented at the infrastructure level (e.g., firewalls, HIDS, hardening, physical isolation, network segmentation, separation of duties, backups…).

Misuse of functionality:

The functionality behind these queries exists so that it can be used only by election managers when configuring and decommissioning election events in the back-office component of the voting platform. If an attacker is one of these election managers, they could misuse their privilege for example by executing a decommission of the voting system, however as has been described earlier, this action cannot be hidden and can be detected.

Thread session:

As mentioned in our response, getSession() performs a null check to guarantee that the sessionFactory is not null, and only get the session when it is not. Therefore, there is no risk that the SQL command fails because it is executed when there is a session failure, which would open the door to inconsistencies.

Whilst the Reviewers consider the presence of naked SQL a concern, the existence of this code is carefully considered to ensure transaction safety and ensure that any misuse by election managers can be detected.

3.2.3 Key Generation and Randomness

Recommendation: In future, it must be possible to build the minified JavaScript as part of the review process. The Vendor should investigate the causes of duplication and eliminate them where possible.

Scytl's response:

The main PRNG code has not been changed as it is considered stable and no major changes were required. Scytl provided the old entropy test results to the Reviewers since they were still valid. In addition we are attaching the recent results we obtained in an annex to this document, to confirm there are no changes.

Additionally, we would like to clarify that our Javascript PRNG is not used to create keys but only to generate randomness for the ElGamal encryption component.

Regarding instructions for building a minimised version of JavaScript, we will consider them as a future improvement for external review processes. Scytl's review of coding practices is an ongoing activity.

3.2.4 Unused Code

Recommendation: The source code must be refactored and all unused modules and functionality removed. Any production build should only be based on the cleaned code.

Scytl's response:

Scytl takes note of this recommendation, however, due to the product nature of the iVote voting system, this recommendation does not align entirely with other implementation priorities of the system or Scytl. Scytl will review this recommendation with our customer to decide on the preferred approach.

The basis of the iVote voting system is our generic voting system, Invote, which implements several capabilities to allow a more complete configuration of different election types and cryptographic protocol properties. This provides both benefits and constraints, as using more generic components allows customers to take advantage of new capabilities of the voting system, but will also add unused code if these capabilities are not used. This does not automatically justify classification as a security risk in Scytl's view as the functionalities are not used, however it does make source code review of the system more complex.

3.2.5 Missing Contracts and Invariants

Recommendation: The Vendor should add contracts to all methods that define the iVote system.

Scytl's response:

Coding guide-lines are currently under review and updates will address this concern. Historically Scytl has not required contracts for all implemented methods and assessing this will be part of the coding review. The coding review is expected to be completed for end Q1-2022, and the outcome will be shared with NSWEC.

3.2.6 Passwords

Recommendation: All unnecessary functionality must be removed from the source code.

Scytl's response:

Scytl agrees with the reviewers regarding the benefit in reduction of presence of code that is not used, and as noted by the reviewers some effort has been already done in this direction. Scytl notes that the potential attack mentioned by the auditors due to the presence of an optional password component is not feasible. This optional password component (password) is not present as an alternative authentication mechanism to the other authentication components (ivote number and pin), but as an additional optional complement to them. For instance, it can be used to request a third secret from the voter, such as a personal challenge or a one-time password as used in some systems. Therefore, it is not possible to disable any of the other two mandatory credentials (ivote number or pin) to make an attack (despite it has a hardcoded value when this third authentication component is requested), because the other two are always needed (ie: only iVote number and pin are used in the transform function used to retrieve and decrypt the key container that has the digital certificate that allows the casting of a valid vote).

3.2.7 Hardcoded Passwords

Recommendation: The hardcoded passwords should be removed from the source code.

Scytl's response:

Scytl agrees that production passwords should not be included in source code, and Scytl has found that they are not. The passwords found are passwords used in development environments and they are completely non-functional in a production implementation and cannot be used in any way – in a production environment passwords are stored in secure password vaults.

4 Static Analysis (Part 4 of the Final Report)

4.1 Trusted Build

> Recommendation: NSWEC should have access to the original source, for example, a git repository, and should be able to build the production iVote system from scratch.

Scytl's response:

The original source code is accessible for review purposes. Extending the access for other purposes is a contractual matter between Scytl and the NSWEC.

4.2 Analysis of SLOCcount Report

Scytl response:

Scytl takes note of the Reviewers' comments to reduce code duplication. Please refer to the response to section 3.2.1 - Complexity.

4.3 SpotBugs Static Analysis

> Recommendation: We would recommend the NSWEC review the SpotBugs report with the Vendor, paying particular attention to concurrency issues and invalidated servlet parameters, and patch where possible. Testing should also be done to with faulty, invalid, and out of range servlet parameters to ensure the system deals with them gracefully.

Scytl response:

Scytl is open to working with our customer on automated reporting-based approaches to software bug management. As the Reviewers mentioned in the Final Report, the concurrency issues reported in Spotbugs are likely to be false positives, so Scytl's focus is typically to carry out performance tests. If these tests are successful, in general, we do not expect possible concurrency issues.

Regarding the use of faulty, invalid, and out of range parameters, Scytl will evaluate the impact of using fuzz testing tools in the system development environment and communicate this with the NSWEC.

Annex 1: PRNG Scytl

A1.1: Multiple browsers and multiple sessions

This test validates the sum of outputs from several cryptolib js PRNGs created from different browsers. The tests' rationale is to demonstrate that the data produced by all the PRNGs is completely different and uncorrelated even when different instances and browsers are used.

In total there were 14 sessions for each browser with around 350 MB of random data produced per session. The browsers used were:

- Firefox 72.0.2
- Chrome 79.0.3945.130
- Edge 79.0.309.71
- Opera 66.0.3515.44

A1.2: The results of Dieharder

These are the results of last random quality test done on July 2021 using the Dieharder tool on the Javascript PRNG and data data entropy collector implemented in Invote and used by iVote voting system.

```
#=============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================#
   rng_name    |           filename             |rands/second|
 file_input_raw|           joined.dat           |  1.29e+07  |
#=============================================================================#
```

test_name	ntup	tsamples	psamples	p-value	Assessment
diehard_birthdays	0	100	100	0.42278	PASSED
diehard_operm5	0	1000000	100	0.992412	PASSED
diehard_rank_32x32	0	40000	100	0.443462	PASSED
diehard_rank_6x8	0	100000	100	0.993152	PASSED
diehard_bitstream	0	2097152	100	0.274582	PASSED
diehard_opso	0	2097152	100	0.686947	PASSED
diehard_oqso	0	2097152	100	0.728511	PASSED
diehard_dna	0	2097152	100	0.74323	PASSED
diehard_count_1s_str	0	256000	100	0.169253	PASSED
diehard_count_1s_byt	0	256000	100	0.95291	PASSED
diehard_parking_lot	0	12000	100	0.979382	PASSED
diehard_2dsphere	2	8000	100	0.03963	PASSED
diehard_3dsphere	3	4000	100	0.265687	PASSED
diehard_squeeze	0	100000	100	0.84385	PASSED

diehard_sums	0	100	100	0.419213	PASSED
diehard_runs	0	100000	100	0.482744	PASSED
diehard_runs	0	100000	100	0.818143	PASSED
diehard_craps	0	200000	100	0.889763	PASSED
diehard_craps	0	200000	100	0.376657	PASSED
marsaglia_tsang_gcd	0	10000000	100	0.389574	PASSED
marsaglia_tsang_gcd	0	10000000	100	0.469853	PASSED
sts_monobit	1	100000	100	0.042662	PASSED
sts_runs	2	100000	100	0.803625	PASSED
sts_serial	1	100000	100	0.354823	PASSED
sts_serial	2	100000	100	0.341664	PASSED
sts_serial	3	100000	100	0.280517	PASSED
sts_serial	3	100000	100	0.955434	PASSED
sts_serial	4	100000	100	0.848816	PASSED
sts_serial	4	100000	100	0.192911	PASSED
sts_serial	5	100000	100	0.221315	PASSED
sts_serial	5	100000	100	0.988523	PASSED
sts_serial	6	100000	100	0.180215	PASSED
sts_serial	6	100000	100	0.12133	PASSED
sts_serial	7	100000	100	0.977024	PASSED
sts_serial	7	100000	100	0.654789	PASSED
sts_serial	8	100000	100	0.98076	PASSED
sts_serial	8	100000	100	0.935927	PASSED
sts_serial	9	100000	100	0.911053	PASSED
sts_serial	9	100000	100	0.630347	PASSED
sts_serial	10	100000	100	0.564345	PASSED
sts_serial	10	100000	100	0.829171	PASSED
sts_serial	11	100000	100	0.139839	PASSED
sts_serial	11	100000	100	0.106635	PASSED
sts_serial	12	100000	100	0.846064	PASSED
sts_serial	12	100000	100	0.406906	PASSED
sts_serial	13	100000	100	0.049023	PASSED
sts_serial	13	100000	100	0.606318	PASSED
sts_serial	14	100000	100	0.733359	PASSED
sts_serial	14	100000	100	0.971512	PASSED
sts_serial	15	100000	100	0.56105	PASSED
sts_serial	15	100000	100	0.803618	PASSED
sts_serial	16	100000	100	0.798243	PASSED
sts_serial	16	100000	100	0.72114	PASSED
rgb_bitdist	1	100000	100	0.825537	PASSED
rgb_bitdist	2	100000	100	0.106178	PASSED
rgb_bitdist	3	100000	100	0.976592	PASSED
rgb_bitdist	4	100000	100	0.986826	PASSED
rgb_bitdist	5	100000	100	0.885152	PASSED
rgb_bitdist	6	100000	100	0.557871	PASSED
rgb_bitdist	7	100000	100	0.54923	PASSED
rgb_bitdist	8	100000	100	0.232158	PASSED

```
rgb_bitdist            9    100000   100 0.722286  PASSED
rgb_bitdist           10    100000   100 0.034189  PASSED
rgb_bitdist           11    100000   100 0.682904  PASSED
rgb_bitdist           12    100000   100 0.279668  PASSED
rgb_minimum_distance   2     10000  1000 0.437482  PASSED
rgb_minimum_distance   3     10000  1000 0.160192  PASSED
rgb_minimum_distance   4     10000  1000 0.900253  PASSED
rgb_minimum_distance   5     10000  1000 0.383807  PASSED
rgb_permutations       2    100000   100 0.337012  PASSED
rgb_permutations       3    100000   100 0.060314  PASSED
rgb_permutations       4    100000   100 0.497925  PASSED
rgb_permutations       5    100000   100 0.809984  PASSED
rgb_lagged_sum         0   1000000   100 0.110331  PASSED
rgb_lagged_sum         1   1000000   100 0.582097  PASSED
rgb_lagged_sum         2   1000000   100 0.757885  PASSED
rgb_lagged_sum         3   1000000   100 0.632589  PASSED
rgb_lagged_sum         4   1000000   100  0.73915  PASSED
rgb_lagged_sum         5   1000000   100 0.493763  PASSED
rgb_lagged_sum         6   1000000   100 0.893604  PASSED
rgb_lagged_sum         7   1000000   100 0.897801  PASSED
rgb_lagged_sum         8   1000000   100 0.265144  PASSED
rgb_lagged_sum         9   1000000   100 0.959314  PASSED
rgb_lagged_sum        10   1000000   100  0.33238  PASSED
rgb_lagged_sum        11   1000000   100 0.087148  PASSED
rgb_lagged_sum        12   1000000   100 0.281615  PASSED
rgb_lagged_sum        13   1000000   100 0.887862  PASSED
rgb_lagged_sum        14   1000000   100 0.819194  PASSED
rgb_lagged_sum        15   1000000   100 0.545132  PASSED
rgb_lagged_sum        16   1000000   100 0.883137  PASSED
rgb_lagged_sum        17   1000000   100 0.549478  PASSED
rgb_lagged_sum        18   1000000   100 0.436385  PASSED
rgb_lagged_sum        19   1000000   100  0.93481  PASSED
rgb_lagged_sum        20   1000000   100 0.979775  PASSED
rgb_lagged_sum        21   1000000   100 0.282879  PASSED
rgb_lagged_sum        22   1000000   100 0.047357  PASSED
rgb_lagged_sum        23   1000000   100 0.443598  PASSED
rgb_lagged_sum        24   1000000   100 0.044703  PASSED
rgb_lagged_sum        25   1000000   100 0.979771  PASSED
rgb_lagged_sum        26   1000000   100 0.817458  PASSED
rgb_lagged_sum        27   1000000   100 0.877601  PASSED
rgb_lagged_sum        28   1000000   100  0.46385  PASSED
rgb_lagged_sum        29   1000000   100 0.977621  PASSED
rgb_lagged_sum        30   1000000   100 0.495754  PASSED
rgb_lagged_sum        31   1000000   100 0.291066  PASSED
rgb_lagged_sum        32   1000000   100 0.720531  PASSED
rgb_kstest_test        0     10000  1000 0.733923  PASSED
dab_bytedistrib        0  51200000     1 0.898609  PASSED
```

```
dab_dct          256     50000   1  0.630205  PASSED
dab_filltree      32  15000000   1  0.365178  PASSED
dab_filltree      32  15000000   1  0.421989  PASSED
dab_filltree2      0   5000000   1  0.286912  PASSED
dab_filltree2      1   5000000   1  0.407294  PASSED
dab_monobit2      12  65000000   1  0.532862  PASSED
```