NSWEC-7

DemTech Group

DEMTECHGROUP
DEMOCRACY, TECHNOLOGY & TRUST

Ribegade 19 st th
2100 Copenhagen
Denmark
CVR: 35527052

Review of the iVote 2.0 System

David Hook and Carsten Schürmann

January 2019

Final Report

Table of Contents

1 Executive Summary

We have assessed the software quality of the iVote system designed by Scytl by using off the shelf automatic code-scanning tools and by manually inspecting the source code.

The code-scanning analysis has shown that the overall quality of the code is in general high, and that the implementation is largely free from bad and insecure programming patterns with the exception of the code implementing the mixer, which is still under development and apparently has not undergone quality assurance. Scytl appears to use the Sonar code scanning tool throughout the software development process, which is good practice and explains our findings. For the manual analysis of the code, we have applied functional matching and a verifiability analysis, resulting in the following observations.

1. Some parts of the source code appear to be still under development when the snapshot was provided to us. This may mean that the findings of this report only apply to this particular version of the system, and not future versions of it. The source code contains several TODO comments and in particular, the code for error and exception handling in the mixer is still under active development. Automated analysis of the mixnet also flagged some possible quality issues regarding the mutability of objects.
2. The design documentation and the implementation are not always in sync. We have compared critical parts of the implementation line by line to the design documents provided to us. The results include that (1) the documentation differs from the implementation for certain algorithms for example mixing, (2) some functionality could not be identified in the source code, (3) the design documentation was found to lack explicit references in the implementation.
3. The code base contains a substantial amount of unused functionality, which is outside the scope of this review. The presence of such code is a security concern, since if it is executed, for example, by an insider attacker, it could be used to crash the application, or poison databases and log files.
4. The system relies on JavaScript clients to implement the voting app and the vote verifier. It appears that the functionality of both parts is imple-

mented in the *ivapi* library, of which different versions are included in the source code for vote casting and for vote verification.

5. There is a documentation gap concerning entropy for the JavaScript client. The quality of entropy is critical for the security of the iVote-system, reports concerning the quality of entropy generated in the client using recognized tests for representative browsers need to be provided.

6. The iVote-source code appears to be a combination of (1) standard Scytl modules (2) modules developed for iVote but intended to become part of the standard Scytl modules, such as mixing, and (3) custom developed modules for iVote to integrate the solution into existing infrastructure. The reuse of standard components, in general, is good practice however it also bloats the code base which renders the software review unnecessarily tedious.

7. The iVote system relies on several libraries, or dependencies, that were not included in the drop and that are necessary to compile the system. Such libraries include Spring, Hibernate frameworks and others. The iVote system and therefore the entire election dependent critically on the frameworks. Vulnerabilities in these libraries and frameworks translate directly into vulnerabilities of the iVote system. To check such vulnerabilities, use the CVE database as a resource.[1]

Recommendation 1 Scytl should provide better design documentation of the system. Without better documentation, it will be difficult to maintain the system, for example, by a future vendor.

Recommendation 2 Scytl should consider refactoring the source code in such a way that only the necessary parts are included in the distribution. This applies to the Java components as well as the JavaScript files.

Recommendation 3 NSWEC should require Scytl to make available a system that builds out of the box for future code reviews. At the very least, the code base should be accompanied by all version numbers and dependencies that the build relies on for the distribution to the reviewers.

[1] See https://www.cvedetails.com/vulnerability-list/vendor_id-9664/product_id-17274/Springsource-Spring-Framework.html

2 Scope and Review Methodology

This review is based on five documents that were provided by Scytl.

[redacted]

[4] NSW Electoral Commission. iVote Voting System, Voting Protocol Description . Internal release, April 24 2018.

[redacted]

The code base, provided to us directly by Scytl, covers around 14 different languages. The principle language is Java, with about 286000 lines of executable production code followed by JavaScript at about 83000 lines of executable code. The rest of the system consists of XML, SQL, HTML, CSS, JSON and shell scripts producing a total body of work of just over one million lines. Given the complexity and size, as well as consideration of time and budget, an exhaustive review of the entire code is beyond the scope of this review.

In this review we use automated code scanning tools to identify vulnerabilities, bugs, and possible bad programming style and conduct a functional matching and verifiability analysis by identifying and matching the functionality provided in the documentation with the source code. Scytl has also shared the result of different coverage analyses of some modules to allow for a review of the effectiveness of their current testing regime. As a result our review has concentrated on functional matching against the documentation we have been given, a review of implementation of the entire iVote architecture, including encryption of ballots, transmission, verification, cleansing, mixing, and decryption. Because of resource constraints, the review of Scytl's secure logging system and the phone voting interface are outside the scope

of this review. Automated analysis and developer tools were applied to the entire code base to look for duplication and unused or unnecessary code. Due to the lack of good code scanning tools for JavaScript, no automated code scanning analyses for JavaScript APIs was conducted.

Outside the scope of functional matching and the verifiability analysis are all parts of the implementation for which no design documents were provided. This includes some of the standard modules of the Scytl voting solution, but also configuration functionality related to iVote deployment and third-party libraries, such as Angular.js, RabbitMQ, Spring, Hibernate, and others. As the voting app and the verification app rely on the JavaScript code, we use off the shelf tools to deobfuscate an pretty-print JavaScript libraries, such as “ivapi-0.8.0-min.js”.

3 Major Findings

Client side The client side code is heavily dependent on the use of JavaScript. In some cases the files have been obfuscated, possibly in order to reduce their size. The use of obfuscation has made the review of the code harder but not impossible. There are tools for removing obfuscation which we have used and we are assuming as such tools are readily available it should clear the obfuscation would not affect the security of the system.

JavaScript, unused functionality The JavaScript appears to be full of unused functionality. We recommend getting some clarity on this and having any excess code removed. The issue with unused functionality is that as the JavaScript is readily downloading, an adversary is more likely to find exploits, with unused functionality as it is also often not tested or maintained sometimes providing an easier way in.

JavaScript, TODO markers The JavaScript code contains 39 “TODO” markers. While this is not a sign an issue in itself, we would recommend that any existing TODO are checked to make sure they have been forgotten. Again as with unused functionality, incomplete code can also provide a lever for exploit by an adversary.

Misuse of the Voting App An authenticated adversary can send invalid votes using the voting app by calling standard JavaScript functions with wrong keys, and then cry wolf and subsequently allege incompetence or a cyber attack. Both cases are possible, but any case will be a nightmare for NSWEC.

Commenting The Java/JavaScript code base does not exhibit a clear and consistent commenting policy. It would be so easy for the code to reference back to the design documentation and vice versa. This weakness makes manual code reviews unnecessarily cumbersome.

Entropy and randomness in the JavaScript client It was not immediately clear what kind of entropy the JavaScript code uses to create randomness. The JavaScript code uses cryptolib-js-securerandom, which indicates that robust sources of randomness are included. We could not identify code that would conduct a entropy quality assessment. After further discussion and

assistance from Scytl it appears the JavaScript RNG has been analyzed using both the Dieharder test suite and the NIST Statistical Test Suite. Both the test suites used are definitely appropriate for the task. We would strongly recommend the NSWEC acquire copies of these assessments as the quality of the entropy is a cornerstone of the security of the overall system, and it is highly likely that anyone querying the quality of the system will want such reports to be made available. In fact, it really would be a testimony to the quality system if such reports were available on short notice for later reviewers as well.

Coding style In some parts of the code, a concept of Job is defined to exploit concurrency and to administer computations elegantly and effectively. Such programming patterns contribute to efficient execution of programming tasks, but they make it more difficult for the reviewer to interpret and evaluate the quality of the code.

Potential overflow In the code [SecondCommittmentGenerator.java, line 55-56], a product of two potentially large numbers is computed, which might lead to a potential overflow. If triggered, this problem would mostly likely crash the application during mixing.

Secret key reuse The code is very clear that the secret election key is reconstructed for the purpose of mixing and decrypting the ballots. What kind of mechanisms are in place to prevent secret key reuse? Once the secret key is constructed it may be stored and stolen. Also, since the secret election key is being constructed, once lost, it can be used to decrypt the original ballots breaking vote secrecy. This renders mixing a nice but rather ineffective mechanism to protect the secrecy of the vote. (see [4], Section 2.6.3.1)

Missing arguments from calls to hash-functions While reviewing the generation of the parallel shuffle proof, we observed a deviation between documentation [4], page 15, and the implementation. The calls to hash functions in Prover.java, and Verifier.java appear to be be applied to fewer documents than required. (see [Verifier.java, line 141–143], [Prover.java, line 114–116], [prover.js, line 96], [verifier.js, line 84]).

4 Detailed Findings of Code Scanning Analysis

Owing to the number of artifacts involved, as well as their size and some dependencies that we are not able to resolve, we were not able to do scanning on all modules, but decided to do a sample. It took several days to build sections of the iVote system on our premises. In particular, this was because there are different versions of some of these dependencies used in different parts of the system, and as the build system is based on Maven, in some cases it appears that reference is made to in-house Maven plugins which we do not have, and also in a few cases, the dependencies appear to be local patched versions of standard libraries. Without the correct versions, or version details, for dependent libraries, building the system is impossible.

Full analysis was done on the following modules:

- cryptography
- cryptolib
- secure-logger
- mixnet-verifier
- online-voting-logging

In all cases the code being analyzed was Java code and the full analysis is available in the attached supplement (see Appendix A).

Overall the code analysis suggests the use of analysis tools is evidently taken seriously by the developers, in many cases there were either no defects, or the defects listed were reasonably trivial ones. While automated analysis can be useful it is true to say that sometimes, in the context of the execution environment of a particular piece of code, the automated analysis does overreact.

That said, there were some sections of the code that do warrant further investigation in the mixnet. Given the size of the reports for these sections of code (see pages 40 to 49 of the supplement for example), it does appear

that the code is probably recent and has not completely gone through internal analysis. One issue that is flagged is that some objects being used in the mixnet appear to be either mutable by intent, or expose internal fields and are mutable by accident. We would strongly recommend that where mutability has been reported it is either removed or if the object cannot be changed, steps are taken to protect the rest of the mixnet from the object's mutability. The mixnet does appear to be multi-threaded. As this is the case, mutable objects are quite dangerous. Even if in the current implementation the usage of mutability is unlikely to cause trouble, the reality is that any maintenance on the code base could easily result in the mutability causing errors if a developer either forgets or is unaware that an object may be shared between threads.

5 Detailed Findings of Code Coverage Analysis

Several of the documents provided made claims concerning code-coverage. Abbreviated coverage reports were provided for the following modules:

- cryptography
- cryptolib
- govlab-JavaScript-client-api
- mixnet-verifier
- nsw-consumers
- nsw-converter
- nsw-credential
- nswec-olv-voter-portal
- nsw-results
- online-voting-logging
- p7-cms
- pnyx-govlab
- pnyx-receipts-backend
- pnyx-receipts-frontend
- secure-logger

The abbreviated coverage reports were consistent with all claims made in the documentation.

Further analysis was done in this area also. One issue with abbreviated coverage reports, especially as applying to modules with large number of lines of code, is that while the abbreviated coverage may show coverage of more than 80 percent, the question always remains as to what was missed in the

other 20 percent. As with automated analysis, doing a full coverage check was beyond this review due to the size of code base, however we did request, and received in-depth coverage reports for four modules in order to gain an impression concerning the overall effectiveness of testing. The modules we received full analysis for were:

- cryptolib
- mixnet-verifier
- online-voting-logging
- secure-logger

These modules were chosen as they seemed to provide the greatest amount of "critical" code in the security sense for the overall system. Further sampling was done on the coverage reports due to the number of files involved and the need to read and follow the annotated HTML representing each class file that had coverage analysis done on it.

In the files that were examined it appeared that testing was indeed effective in the sense of providing meaningful coverage reports. Gaps generally appeared to be related to internal exception handling for things that had to be dealt with due to the use of checked exceptions in Java. This is a common shortcoming of the Java language in respect to coverage analysis as often checked exceptions need to be dealt with even if there is no chance of them being thrown. The result of this is that triggering an exception which would exercise the gap created by the code handling the exception in the first place would generally require a catastrophic failure of the system - something which is difficult, if not impossible, to emulate in a test. Other than this "gap issue", in general, branches within the code appeared to be getting tested correctly.

6 Detailed Findings of Functional Matching and Verifiability Analysis

6.1 Crypto Libraries

Basing the level of importance on the provided documentation the primary libraries involved with cryptography are:

- cryptolib
- secure-logger
- mixnet-verifier

Examination of the code also showed the use of some additional libraries which were provided to us as well.

- cryptography: a set of helper classes wrapping JCA/JCE functionality as well as providing a few primitives of its own.
- online-voting-logging: Logging package used for generating secure logs used by the mixnet.
- p7-cms: Utility APIs that capture a simple way of creating signed and enveloped messages from PKCS#7/RFC 5652 Cryptographic Message Syntax (CMS).

Broadly we found these packages meet the expectations we had from the supplied documentation, both in terms of the ones being used and also the security strengths of the algorithms involved.

6.2 System Architecture

In our review methodology for the system architecture section, we followed the documentation provided, and reviewed the part of the code implementing the functionality outlined in the documentation. This part of the review focuses on [4], Chapter 2. We structure this section correspondingly.

6.2.1 Underlying algorithms

We have reviewed the underlying algorithms. Scytl's implementation uses the ElGamal cryptosystem. The encryption of the vote happens in the voter app, and therefore it is the JavaScript crypto libraries that implement the encryption. While reviewing the ivapi-0.8.0-min.js we observed that encryption works with lists of messages (and not individual messages), which indicates that iVote uses the encryption scheme described in [4] Section 2.1.

6.2.2 Election Key Generation

The secret key is generated and then split into different shares (see also [3] Page 74). Eventually, the election private key is reconstructed. There are two problems with this approach. First, the system may be used with the same private election key for several elections. Second, the private election key is already known before the election and may be used by an adversary (who is assumed to be in possession of the key) to decrypt ballots as they arrive and it is reconstructed after the election which would an adversary allow to decrypt the unshuffled and uncleansed ballot box (assuming the adversary has access to the database). A preferred alternative would be to have each Board Member create their own public private key shares and then compute the public election key as the product of the public key shares. This way, by design no one will ever be in possession of the entire secret key. The code base implements the key generation process as described in [4].

As part of this section, we also reviewed the source code implementing the Credential Manager. We have found deviations between the code and the design documentation [3, 4]. Most notably, we observed that the implementation uses an encrypted "iVoteHasPin", that needs to be decrypted before it can be used to compute the credential ID. In the implementation, the "credentialID" also depends on an "apiKey" and the "instutionAlias" [sic] (see [CredentialManagerServiceAdditionalValues.java, line 94], note that institution is misspelled), which are not explained in the documentation.

The construction of a spare credential is also not mentioned in the documentation, but it is used in the implementation [CreateVPBServiceImp.java, line 87] in the computation of the credential ID.

In [CustomUserDetailService.java, line 25], there is a hard-coded username password combination, with a password "nopass". Without detailed design documents it is not clear if this is a real security weakness and how it could be exploited, but it is something that we should flag.

6.2.3 Voter Authentication

We reviewed the source code that implements voter authentication. Most notable, we could not verify the parameter to the PBKDF2 key derivation function. In order to be able to identify locations in the JavaScript code, we needed to deobfuscate and pretty print it. The line numbers refer to the line numbers in the deobfuscated file. The JavaScript code is also structured in blocks. For simplified access we refer here to to the block numbers as well.

[ivapi-0.8.0-min.js, line 42699] *Block 500* This block does the voter authentication (see [4], 2.3 Voter authentication. It is a bit confusing that it says “externalUsername” in the JavaScript, whereas it is called iVotenumber in [4]. Additional arguments to the pbkdf2 function in the JavaScript include “apiKey”, and “institutionAlias”. To run the pbkdf2 function to compute the “credID”, the JavaScript uses the following arguments, (1e4 = 10000 iterations), 16 and p.default.md.sha256.create(). There is concern that the the pbkd2 function generates only a 16 bit key and not a 128 bit key as required.

```
var r = p.default.util.bytesToHex(
  p.default.pkcs5.pbkdf2(
    t,
    'ID',
    1e4,
    16,
    p.default.md.sha256.create()
  )
);
```

The corresponding code in the iVote (see authentication-transform) uses 10000 and 128 as arguments to PBKDF2.

pnyx-govlab *Authentication Transform* Nowhere in the documentation are the different transforms (i.e. Edmonton, Maee) described. If unnecessary, remove from the distribution. It appears to be used in applets, but they are not used in the iVote system.

pnyx-govlab *Remaining modules* The review of this module is outside the scope of this review, because no documentation is provided. Also this project folder contains applets, but there are no applets (to the knowledge of the reviewers) for this version of the iVote system.

6.2.4 Vote Casting Details

Section 2.4 of [4] and section 5.4 of [1] appear to describe different mechanisms for constructing a ballot. After consultation with Scytl it became clear that Section 2.4 of [4] was the mechanism being used in the iVote system. It would be helpful to later reviewers if the the description in [1] was either removed or brought into line with the one in [4].

To identify the vote casting details, we reviewed the deobfuscated JavaScript code further. The iVote voting app will run in the browser. To this end, it uses Angular.js to render ballots. The review of this framework lies outside this review. Note, that vulnerabilities in Angular.js are also vulnerabilities for the iVote-system. While reviewing the deobfuscated JavaScript file, we observe that block 1–350 define functionality related to integrating the Voting app into different runtime environments, blocks 351- 494 implement a version of cryptolib, and blocks 495ff. the code of the actual voting client. We could verify that the different parts of [3], Figures 31 and 32 are present in the JavaScript file, but we could not determine the order in which the different actions are executed. This section summarizes our findings.

[ivapi-0.8.0-min.js, line 41586 ff.] *Matching* The functionality described in [3], page 59 ff. is implemented in this this file. There are discrepancies in naming. The documentation refers to it as OV Api while the file is called IV Api. The implementation of detect does more than its specification, i.e. determine if the app runs on a mobile phone or a computer and in which kind of browser. Institution Data in the document (page 60) is called Tenant Data in JavaScript. Authentication is implemented by the method “loginAdvanced” and “loginRecognized”. There was not enough information in the JavaScript file or the other files of the iVote system to determine where “credentialManagerIntegrationURL” is defined pointing too, and how it is used and secured. There was also not enough information provided in the documentation to determine if the validation methods are complete, see for example “validateParams”. We observed some code duplication on the JavaScript file, for example two different calls to T.getBallot.call(h) in advanced and recognized authentication methods. There are undocumented methods “getAudio” and “validateBallot”, which invokes another method called “ballotdelivery”.

[ivapi-0.8.0-min.js, line 41586 ff.] *Code organization* Regarding the functionality for “castVote”. The organization of the code is problematic. If vote validation is a precondition for “castVote”, then an adversary controlling the JavaScript may just invoke “castVote” directly circumventing validation. Therefore, judging from the code, any voter may cast an invalid vote. The figure on

page 66 also shows that the encrypted ballot is completely constructed in the voting app, and then send to the voting portal without additional verification.

[ivapi-0.8.0-min.js, line 41586 ff.] *Mismatch with documentation* There are methods “verifyVote” and “reportIncorrectVote” that are not described in the accompanying documentation, but they are available in in the JavaScript file. We observe that “verifyVote” provides an adversary with the possibility to learn or to fix the randomness, which may enable attacks through code injection á la Halderman. The same library that is also used for the mobile phone verification app. This is in conflict with a clear segregation of duties. The JavaScript also contains a generated shift/reduce parser, whose purpose is unclear.

[ivapi-0.8.0-min.js, line 44239] *Blocks 513* Why is there are hard-coded election relevant information in the JavaScript file? It looks like this block conducts a test of all of the functionality necessary to run the ivoteClient. There is some mention of such functionality in [4], but details are missing.

[ivapi-0.8.0-min.js, line 45345] *Blocks 518* As a best guess, this block implements the vote casting details from [4], section 2.4. The “electoralBoardPubic” key is located on line 46061. The blueprint for the envelope used is given on line 46441, in block 524. We wonder, why the public election key needs to be encrypted, one more time (see “encryptSecretKey”). This is not described in [4] and therefore a deviation. The get encrypt the vote (see “getEncryptedBallotData”), the encryption function of a “CryptoHelper” is being invoked, but it is unclear to which encrypt this refers to. However, it is the same encryption function that is used to encrypt the public election key, which refers to the encryption function defined on line 32798 (Block 424). It appears that the JavaScript implementation deviates from [4]. The signing of the ballot is implemented in “createVRR”, line 46114. Note, that the signature does not apply to the Schnorr proof that is also generated with the ballot.

[ivapi-0.8.0-min.js, line 47530] *Blocks 530* Here the receipt is extracted and displayed to the voter. This block contains many more functions than described in the design documentation [4]. Not enough documentation is provided to understand the intended meaning of the other blocks.

[ivapi-0.8.0-min.js, line 48093] *Blocks 532* The “sendVote” method (defined on line 47093) appears to send the vote to the voting service without waiting for a reply from the Voting Service with the signature of the encrypted vote that is to be checked. This means that we could not verify step 10 in Section 2.4 of [4].

[remote-login.js, line 97] *Hardcoded password* a default password is set to “password”. This seems unnecessary but not necessarily harmful.

[ball.candidate.js, line 200] *Software quality* This is a brittle way of encoding the pop. What if the type of “lastAnswer” is a capitalized “UNDEFINED” because someone made a change in a different module.

[ivapi-0.8.0-min.js] *Dead code* It is unclear how much of the remaining code is important for the iVote-system. We would like to point out, however, that salts and encryption algorithms are hard-coded in the JavaScript file. Functionality such as “getEncrytpedOptions” is available, but not used anywhere.

[ivapi-0.8.0-min.js, line 42728] *Block 500* The initialization vector n and tag o occur only in the implementation, but not in [4]. The format of the the credential is not described in [4], which seems to be a string with three components separated by ’-’: called “transformedPin”, “transformedUsername”, “password”.

[ivapi-0.8.0-min.js, line 42841] *Block 501* In the JavaScript code, it seems that the label “makeSecretKey” appears to be a factored out version of the call to the “pbkdf2” method, called one time with the “SALT_VOTER”, and the other with the salt for the Password (see password salt in ([4], page 6), to be consistent with [4]. Also in this call, it appears that pbkdf2 is called with argument 16 instead of 256.

[ivapi-0.8.0-min.js, line 42841] *Blocks 502-507* It is not clear what the code does in relation to [4].

[ivapi-0.8.0-min.js, line 43516] *Blocks 508* is the “getPrivateKey” method here the recovery of the authentication token from the Voting Service? It is not clear. The naming in the JavaScript could be improved.

[encrypter.js, line 139] *Documentation deviation* The encrypter provides two modes of operation when encrypting ballots, depending on if “saveSecret” is true or false. The “secret” is what is called r in Section 2.4 of [4] and will need to be stored for all ballots for the purpose of verification. Therefore, “saveSecret” must always been set to true. Consider removing the other option. Regarding secrets: it is very important that they are not stored together with the encrypted ballot (which could then be decrypted because the it is enough to know the “secret” to decrypt, knowledge of the election private key

is unnecessary). Therefore, the “secret” should be separated from the encrypted ballot as early as possible, preferable already right after encryption and Schnorr proof generation. The documentation should be updated to reflect how, when and where the “secret” is stored, and when it is deleted.

[schnorr-proof-handler.js, line 101] *Imprecise Contract* The method generate should require that “options.preComputation” is not equal to ’Undefined’. Being precise about contracts will simplify the code, avoid duplication, and simplify the argument why the code is correct.

[prover.js, line 63] *Imprecise Contract* In the case that the argument “elements” is undefined, the method prove will still generate a Schnorr proof. It is not clear what this proof represents.

[prover.js, line 69] *Hash computation* When the hash function is called to compute the challenge in the zero knowledge proof, it seems that the only public value used to generate the hash is “gamma”. There are other values that are public that could be passed to the hash generator.

[SecureMessageHelper.js, line 96] *Deviation from description* To prepare the envelope with the encrypted vote, the implementation encrypts a secret key. This secret key is not mentioned in [4]. If the secret key refers to the randomness used to generate the encryption of the vote then this indicates a bug in the implementation.

6.2.5 Cast-as-Intended Verification

Internet Voting Verification

Just as the voting app, the verification app consists of an embedded JavaScript. Interestingly enough, both JavaScript programs ivapi-0.8.0-min.js (voting app) are ivapi-0.7.0.js (verification app) are equal in substantial parts of the code, it almost feels like the voting app runs a new newer version of the JavaScript library than the verification app. This is very concerning as maintainers could easily get confused, and developers may already have done so. The JavaScript code is started from the Android App or the iPhone app, with “nsw-vote-verification” [MainActivity.java, line 12], which is also our entry point for the source code review.

[index.js, line 91] *LoginParams* It is strange that password is set to "password" and that there is a field "isforVerification". We would have expected that the ballot is retrieved the standard way, based on the "credID". Where does the voting app learn this information from? Judging from the source code alone, it is not clear where QR-scanning and authentication are invoked. We suspect that this part of the code is executed after the QR code is scanned and login details are collected. This would also explain why randomness is passed to the verifier as an argument at launch time.

[ivapi-0.7.0.js, line 38067] *Block 336* We could not identify the lines of code that would indicate that randomness is computed from the seed s that was obtained from the QR code, and must therefore assume that this computation happens elsewhere.

[ivapi-0.7.0.js, line 38092] *Blocks 336* It is not clear from the JavaScript code, why the public election key was used (as required by [4]). Furthermore, it is not entirely clear, that the decryption function uses the inverse of the quadratic residue method.

Phone Voting Verification

The software provided to us did not contain the essential modules for the IVR system that would allowed us to review the phone voting verification.

packages2/default/config.json *Typos* Different default fields with a name ending in "length" are misspelled. This could be problematic if the names are spelled correctly when referred elsewhere.[2].

6.2.6 Cleansing

The source code contains different modules that refer to cleansing. Some functionality can be found in the pnxy-govlab/mixing module, the other in the pnxy-govlab/cleansing. This is confusing. The code should perhaps be refactored.

Our main observation and concern is that the way cleansing is implemented, it is not software independent. The cleanser is defined in such a way that the reasons for removing a ballot are documented and can be most likely

[2] See, for example, main.750c626e71564bfca248.js

be checked by an auditor. However, if the cleanser skips ballots (leaves too many in the ballot box), for example, by a programming mistake or if the cleanser is under the adversary's control, an auditor will not have enough information to determine if such ballots should have been cleansed. This would mean, that if voters can revote as many times as they want (which is the case for NSWEC), the iVote-system may be neither universal nor end-to-end verifiable.

Regarding the specification outlined in [4], 2.6.6.1, we could not identify any code that would do neither the ballot box verification nor the ballot box filtering.[3] In relation to the specification outlined in [4], 2.6.1.3, we found that the certificate verification does not check the certificate chain. We also found that the signature of the encrypted vote is not verified.

[VerifiableMixingParser.java, line 149] *Class conversion* In this line, the object "pk" is typecast to "(BCDHPublicKey) pk". This may fail as the key type is provider specific.

[CleansingOutput.java, line 128] *Commenting* The lack of useful comments in the source file is not good practice.

pnyx-govlab *Cleansing* This module contains functionality for cleansing ballots, whose Schnorr proof does not verify. The real cleanser appears to be embedded in the pnyx-govlab mixing.

[CleansingService.java, line 221] *Unresolved TODO comment*

```
// TODO validations depend on SecurityModel too
// proofs for elgamal and homomorphic
// vote length for elgamal
```

[SelectSecureMessagesTask.java, line 74ff] *Code repetition* The way how messages are distributed into batches, is programmed in a strange way. First, there is code duplication. Second, it looks as if the last messages of a previous batch may appear also as the first message in the next batch. There is no problem, per se, as the SQL query in [SecureMessageJdbcDAO.java, line 184] uses a "$<$" symbol, but nevertheless, this could easily lead to a programming mistake.

[3] In the case that this functionality is implemented we would appreciate a pointer to it. Thank you.

[SecureMessageJdbcDAO.java, line 185] *Observation* Why is there a leading “sql.”. For consistency should be removed, including the ; on the line before. We were not given enough information about the SQL database schemas to evaluate the quality of the query.

[CleansingValidationProcessor.java, line 72] *Completeness* It looks like the Cleanser only checks that all mandatory field are provided [SecureMessageOpener.java, line 159], that voter information is correct [SecureMessageOpener.java, line 163], and that the fields are correctly formatted [SecureMessageOpener.java, line 168], such as proof timestamp, login, and electionID. However, we could not identify any code in the cleanser that would remove duplicate votes from the same voter.

[SecureMessageOpener.java, line 187] *Method calculateReceipt* There is no case for homomorphic, so we assume that Scytl’s implementation is configured as “VERIFIABLE_MIXING”. Why is “homomorphic” a part of the code base?

[4] *Section 2.6.1.3* The naming of the variables in the implementation and the names in the documentation do not mix, which renders the review more complicated than necessary.

[SecureMessageOpener.java, line 330] *validateCertificateChannelAndMode* The validity of the certificate chain or the X509 certificate is not checked here. This is a deviation from [4] Section 2.6.1.3.

[4] *Section 2.6.1.3* The signature of the Encrypted is Vote is not checked, at least not here. Neither is the receipt number of the vote. This is a deviation from the specification.

[ReceiptVerifier, line 20] The receipt verifier inherits its “verify” method from “BaseVerifier”. This is a bit funny, because one would think that making a new class for “ReceiptVerifier” would also need to determine how receipts are verified.

[5] *Section 6* “The third step is to check that only the allowed number of votes were cast”. The source code does not refer to “TOO_MANY_VOTES”.

6.2.7 Mixing

The mixing module implemented in the iVote-system appears to be new and under construction. Many error cases are not implemented completely, and

judging from some of the comments in the code, it is not clear what the correct functionality is. In the form the code is now, it is possible for it to crash with an an “Unmatched exception” during mixing (as opposed to dying gracefully). The mathematical foundations of the mixer code are extremely challenging. The paper by Bayer and Groth present different versions of their mixing algorithm, and judging from our review, the one that is implemented is not the one that is described in [4].

Mixing/mixing-core

[MixingController.java, line 39] *Spurious comments* The mixing code has many disconcerting comments, such as

```
//TODO [Marc] Warning: the caller to this method must
                       wait until it has finished
```

All such comments should be resolved by the time the code goes into production.

Mixing/commons

[GjosteenElGamal.java,GjosteenElGamalRandomness.java, etc.] *Missing exception handlers* The mixing code lacks proper exception management and therefore also maturity. This code is central to the Bayer and Groth mixer implementation.

Mixing/mixnet-shuffle

[ElGamalShuffler.java, line 156 ff.] *Lack of specification* To a large extent, the code this file appears ok, however, due to the lack of a specification, it is impossible to judge the soundness of and completeness of the code.

[4] Section 2.6.2.2 *Unwarranted claim* The security proof can never show 100% security. Everything is up to negligible probabilities.

Mixing/proof generation

The root of all functionality in the implementation is [ShuffleProofGeneration.java, line 68 ff.]

[ParallelZeroProofGenerator.java, line 62–66] *Undocumented resetting of* m This operation is not described in either of the documents. Three classes participate in the generation of the by Bayer-Groth proof, including "FirstCommitmentGenerator", "SecondCommitmentGenerator" and "ParallelSecondAnswerGenerator".

[MultiExponentianImpl.java, line 69] *Omitted sources* In [BaseCommitmentGenerator.java, line 67 and 71] the permutation is converted into a vector referred to as exponents (which is called **a** in [4], Protocol 1) and a random vector called **r** is generated. The commitment is then computed in [PrivateCommitment.java, line 63ff.]. The class "Multiexponentiator" was not found.

[SecondCommittmentGenerator.java, line 55-56] *Immature code* As noted in the comment, if $n \times m > maxInt$ then bad things will happen. This depends on the size of the ballotbox, but can have serious consequences. In [4], there is no need to compute $n \times m$, as the length of vector B is simply N. Consequently, the length of $vecx$ should be N and not $n \times m$. (x is called challengex in the code). Related, b in the documentation is a vector, whereas $bExponents$ is a matrix in the implementation. Very likely, the documentation describes a simple instance of the Bayer Groth, whereas the implementation implements what is needed to get iVote going.

ParallelSecondAnswerGenerator.java *Documentation mismatch* The implementation does not really match the protocol as the implementation uses multi-dimensional matrices and the documentation uses one-dimensional vectors. We could not complete functional matching but suspect that the exposition in the protocol is a simplified version of what is implemented.

6.2.8 Decryption

We started the review with AbstractBallotDecryptionService.java which contains the implementation of the decryption algorithm is implemented. The main part of "ProveDec" (see [4], page 15) is implemented in Prover.java. The implementation matches the decryption process as described in documentation.

[AbstractBallotDecryptionService.java, line 98] *Threshold cryptography* We would have expected a threshold key share to decrypt ballots. It appears that the election secret key must have been entirely constructed before calling attempting to decrypt the ballot.

[AbstractBallotDecryptionService.java, line 98] *Incompatible naming between documentation and implementation* It would make the review much easier, if the naming would be applied consistently.

[ZpSubgroupProofProver.java, line 494] *validateDecryptionProofGeneratorInput* It would be nice to more details in the design documents to assess if these are precisely the correct validity checks, or if something is missing. (see also validatePublicValues [Verifier.java, line 185], validateProof [Verifier.java, line 207], etc.)

[Verifier.java, line 151] *Variable naming* The exponent should be a hash function h, but in the code it is renamed to c. This is a bit confusing.

6.2.9 Proof of Correct Decryption

We are not totally clear on the decryption proof overall as the library code appears to be written to support several kinds, however it does appear that VerifiableMixer.js calls schnorr-proof-handler.js and does so with options.voterId and options.electionId set.

If our understanding is correct, our concern is that the plaintext of the ballot is not included in the hash for the proof (or at least it does not appear to be). In the previous version of the NSWEC system this was also the case but the ballot was encrypted using GCM so there was some argument as to how much of what was in the encrypted ballot needed to be included in the hash for the proof to be effective since authenticated encryption was involved due to the use of GCM. In this case the ElGamal encryption is solely relied on to carry the encrypted options from the ballot so it would seem to be appropriate that the plaintext going into that encryption was also included in the hash created for the zero knowledge proof associated with the encryption. We would like this matter to be clarified and should it be the case that further information should be included in the hash related to the zero-knowledge proof we would strongly encourage this to be done as the proof would be weaker than it should be otherwise.

Appendix A: Spotbugs report

The following pages include the spotbugs reports for all modules that were checked.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptography/build/classes/java/main

Metrics

1050 lines of code analyzed, in 51 classes, in 8 packages.

Metric	Total	Density*
High Priority Warnings	3	2.86
Medium Priority Warnings		0.00
Total Warnings	**3**	**2.86**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Internationalization Warnings	3
Total	**3**

Warnings

Click on a warning row to see full context information.

Internationalization Warnings

Code	Warning
Dm	Found reliance on default encoding in com.scytl.crypto.CertificateHelper.certificatesPEM(Certificate[]): java.io.ByteArrayOutputStream.toString()
Dm	Found reliance on default encoding in com.scytl.crypto.CertificateHelper.certificatesPEM(Certificate[]): String.getBytes()
Dm	Found reliance on default encoding in com.scytl xml.DomHelper.serializeAndClose(OutputStream, Document): String.getBytes()

Details

DM_DEFAULT_ENCODING: Reliance on default encoding

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-proofs/build/classes/java/main

Metrics

152 lines of code analyzed, in 7 classes, in 4 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-elgamal/build/classes/java/main

Metrics

617 lines of code analyzed, in 20 classes, in 4 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-commons/build/classes/java/main

Metrics

25 lines of code analyzed, in 3 classes, in 2 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-symmetric/build/classes/java/main

Metrics

12 lines of code analyzed, in 1 classes, in 1 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-symmetric/build/classes/java/main

Metrics

12 lines of code analyzed, in 1 classes, in 1 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-digital-envelope/build/classes/java/main

Metrics

361 lines of code analyzed, in 11 classes, in 3 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-stores/build/classes/java/main

Metrics

19 lines of code analyzed, in 2 classes, in 2 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-stores/build/classes/java/main

Metrics

19 lines of code analyzed, in 2 classes, in 2 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-proofs/build/classes/java/main

Metrics

1789 lines of code analyzed, in 49 classes, in 5 packages.

Metric	Total	Density*
High Priority Warnings	1	0.56
Medium Priority Warnings		0.00
Total Warnings	**1**	**0.56**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Dodgy code Warnings	1
Total	**1**

Warnings

Click on a warning row to see full context information.

Dodgy code Warnings

Code	Warning
RV	Return value of new java.util.ArrayList() ignored, but method has no side effect

Details

RV_RETURN_VALUE_IGNORED_NO_SIDE_EFFECT: Return value of method without side effect is ignored

This code calls a method and ignores the return value. However our analysis shows that the method (including its implementations in subclasses if any) does not produce any effect other than return value. Thus this call can be removed.

We are trying to reduce the false positives as much as possible, but in some cases this warning might be wrong. Common false-positive cases include:

- The method is designed to be overridden and produce a side effect in other projects which are out of the scope of the analysis.

- The method is called to trigger the class loading which may have a side effect.

- The method is called just to get some exception.

If you feel that our assumption is incorrect, you can use a @CheckReturnValue annotation to instruct SpotBugs that ignoring the return value of this method is acceptable.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-certificates/build/classes/java/main

Metrics

394 lines of code analyzed, in 17 classes, in 5 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-certificates/build/classes/java/main

Metrics

394 lines of code analyzed, in 17 classes, in 5 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-asymmetric/build/classes/java/main

Metrics

21 lines of code analyzed, in 2 classes, in 2 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-mathematical/build/classes/java/main

Metrics

439 lines of code analyzed, in 9 classes, in 4 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-scytl-keystore/build/classes/java/main

Metrics

757 lines of code analyzed, in 21 classes, in 4 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-commons/build/classes/java/main

Metrics

550 lines of code analyzed, in 20 classes, in 7 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-commons/build/classes/java/main

Metrics

550 lines of code analyzed, in 20 classes, in 7 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-asymmetric/build/classes/java/main

Metrics

1341 lines of code analyzed, in 44 classes, in 13 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-asymmetric/build/classes/java/main

Metrics

1341 lines of code analyzed, in 44 classes, in 13 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-digital-envelope/build/classes/java/main

Metrics

44 lines of code analyzed, in 3 classes, in 3 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-symmetric/build/classes/java/main

Metrics

594 lines of code analyzed, in 30 classes, in 9 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-primitives/build/classes/java/main

Metrics

174 lines of code analyzed, in 11 classes, in 4 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-primitives/build/classes/java/main

Metrics

174 lines of code analyzed, in 11 classes, in 4 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-asymmetric/build/classes/java/main

Metrics

21 lines of code analyzed, in 2 classes, in 2 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-scytl-keystore/build/classes/java/main

Metrics

21 lines of code analyzed, in 2 classes, in 2 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-certificates/build/classes/java/main

Metrics

737 lines of code analyzed, in 23 classes, in 5 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-test-tools/build/classes/java/main

Metrics

301 lines of code analyzed, in 8 classes, in 3 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-interoperation/build/classes/java/main

Metrics

3108 lines of code analyzed, in 38 classes, in 12 packages.

Metric	Total	Density*
High Priority Warnings	1	0.32
Medium Priority Warnings		0.00
Total Warnings	**1**	**0.32**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Bad practice Warnings	1
Total	**1**

Warnings

Click on a warning row to see full context information.

Bad practice Warnings

Code	Warning
UI	Usage of GetResource in com.scytl.cryptolib.interoperation.js.commands.Command.evaluateScript(String) may be unsafe if class is extended

Details

UI_INHERITANCE_UNSAFE_GETRESOURCE: Usage of GetResource may be unsafe if class is extended

Calling `this.getClass().getResource(...)` could give results other than expected if this class is extended by a class in another package.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-primitives/build/classes/java/main

Metrics

1185 lines of code analyzed, in 48 classes, in 12 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings	1	0.84
Total Warnings	**1**	**0.84**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Bad practice Warnings	1
Total	**1**

Warnings

Click on a warning row to see full context information.

Bad practice Warnings

Code	Warning
OS	com.scytl.cryptolib.primitives.primes.utils.PrimesUtils.getPrimes() may fail to close stream

Details

OS_OPEN_STREAM: Method may fail to close stream

The method creates an IO stream object, does not assign it to any fields, pass it to other methods that might close it, or return it, and does not appear to close the stream on all paths out of the method. This may result in a file descriptor leak. It is generally a good idea to use a `finally` block to ensure that streams are closed.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-api-elgamal/build/classes/java/main

Metrics

698 lines of code analyzed, in 21 classes, in 3 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-stores/build/classes/java/main

Metrics

106 lines of code analyzed, in 7 classes, in 2 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/cryptolib/cryptolib-stores/build/classes/java/main

Metrics

106 lines of code analyzed, in 7 classes, in 2 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixnet-engine/build/classes/java/main

Metrics

548 lines of code analyzed, in 14 classes, in 4 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings	2	3.65
Total Warnings	**2**	**3.65**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Bad practice Warnings	1
Malicious code vulnerability Warnings	1
Total	**2**

Warnings

Click on a warning row to see full context information.

Bad practice Warnings

Code	Warning
NP	com.scytl.products.ov.mixnet.mvc.VerifyingController.verify(ApplicationConfig, LocationsProvider) has Boolean return type and returns explicit null

Malicious code vulnerability Warnings

Code	Warning
EI2	new com.scytl.products.ov.mixnet.batches.MixingBatch(int, BGMixer, ElGamalEncryptedBallots, Integer, int, BGParams, ZpSubgroup, GjosteenElGamal, ApplicationConfig, int, ElGamalPublicKey, ElGamalPrivateKey, LocationsProvider, MultiExponentiation, ParallelComputeAllE, Randomness[], List, PrimitivesServiceAPI) may expose internal representation by storing an externally mutable object into MixingBatch._batchRandomness

Details

EI_EXPOSE_REP2: May expose internal representation by incorporating reference to mutable object

This code stores a reference to an externally mutable object into the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Storing a copy of the object is better approach in many situations.

NP_BOOLEAN_RETURN_NULL: Method with Boolean return type returns explicit null

A method that returns either Boolean.TRUE, Boolean.FALSE or null is an accident waiting to happen. This method can be invoked as though it returned a value of type boolean, and the compiler will insert automatic unboxing of the Boolean value. If a null value is returned, this will result in a NullPointerException.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixnet-tools/build/classes/java/main

Metrics

1508 lines of code analyzed, in 44 classes, in 9 packages.

Metric	Total	Density*
High Priority Warnings	5	3.32
Medium Priority Warnings	37	24.54
Total Warnings	**42**	**27.85**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Bad practice Warnings	2
Correctness Warnings	1
Experimental Warnings	5
Internationalization Warnings	2
Dodgy code Warnings	32
Total	**42**

Warnings

Click on a warning row to see full context information.

Bad practice Warnings

Code	Warning
RV	Exceptional return value of java.io.File.mkdirs() ignored in com.scytl.products.ov.mixnet.tools.actions.generation.FullGenerationAction.exportPublicKey(ElGamalKeyPair, Path)
RV	Exceptional return value of java.io.File.mkdirs() ignored in com.scytl.products.ov.mixnet.tools.actions.generation.KeyPairGenerationAction.exportPublicKey(ElGamalKeyPair, Path)

Correctness Warnings

Code	Warning
UwF	Unwritten field: com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.pubKey

Experimental Warnings

Code	Warning
OBL	com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init() may fail to clean up java.io.InputStream
OBL	com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.testBGMixnetShuffleAndProofsWithIOParallel() may fail to clean up java.io.InputStream
OBL	com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.testBGMixnetShuffleAndProofsWithIOParallel() may fail to clean up java.io.OutputStream
OBL	com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.initProofGeneration() may fail to clean up java.io.InputStream
OBL	com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.testParametersSuitabilityParallel() may fail to clean up java.io.InputStream

Internationalization Warnings

Code	Warning
Dm	Found reliance on default encoding in com.scytl.products.ov.mixnet.tools.actions.generation.io.BallotsWriter.write(List, Path, String): new java.io.FileWriter(File)
Dm	Found reliance on default encoding in com.scytl.products.ov.mixnet.tools.actions.generation.io.EncryptionParamsWriter.write(ZpSubgroup, Path, String): new java.io.FileWriter(File)

Dodgy code Warnings

Code	Warning
NP	Possible null pointer dereference in com.scytl.products.ov.mixnet.tools.actions.generation.FullGenerationAction.exportPublicKey(ElGamalKeyPair, Path) due to return value of called method
NP	Possible null pointer dereference in com.scytl.products.ov.mixnet.tools.actions.generation.KeyPairGenerationAction.exportPublicKey(ElGamalKeyPair, Path) due to return value of called method
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.computeAllE from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.config from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()

ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.elGamalEncryptedBallotsLoader from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.encryptedBallots from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.m from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.multiExponentiation from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.n from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.pars from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.proofsWriter from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.pubKey from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.shuffler from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.zp from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.encryptedBallots from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetBenchmark.testBGMixnetShuffleAndProofsWithIOParallel()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.concurrencyLevel from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.computeAllE from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.config from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.elGamalEncryptedBallotsLoader from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.multiExponentiation from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.n from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.numBallots from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.pars from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.shuffler from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.sufixEncParams from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.zp from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.encryptedBallots from instance method com.scytl.products.ov.mixnet.tools.benchmark.ParallelBGMixnetParametersBenchmark.testParametersSuitabilityParallel()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.PermutationBenchmark.listCiphertext1 from instance method com.scytl.products.ov.mixnet.tools.benchmark.PermutationBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.PermutationBenchmark.listCiphertext2 from instance method com.scytl.products.ov.mixnet.tools.benchmark.PermutationBenchmark.init()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.TutorialBenchmark.counter from instance method

	com.scytl.products.ov.mixnet.tools.benchmark.TutorialBenchmark.finish()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.TutorialBenchmark.counter from instance method com.scytl.products.ov.mixnet.tools.benchmark.TutorialBenchmark.addOne()
ST	Write to static field com.scytl.products.ov.mixnet.tools.benchmark.TutorialBenchmark.counter from instance method com.scytl.products.ov.mixnet.tools.benchmark.TutorialBenchmark.addTwo()

Details

DM_DEFAULT_ENCODING: Reliance on default encoding

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

NP_NULL_ON_SOME_PATH_FROM_RETURN_VALUE: Possible null pointer dereference due to return value of called method

The return value from a method is dereferenced without a null check, and the return value of that method is one that should generally be checked for null. This may lead to a `NullPointerException` when the code is executed.

OBL_UNSATISFIED_OBLIGATION: Method may fail to clean up stream or resource

This method may fail to clean up (close, dispose of) a stream, database object, or other resource requiring an explicit cleanup operation.

In general, if a method opens a stream or other resource, the method should use a try/finally block to ensure that the stream or resource is cleaned up before the method returns.

This bug pattern is essentially the same as the OS_OPEN_STREAM and ODR_OPEN_DATABASE_RESOURCE bug patterns, but is based on a different (and hopefully better) static analysis technique. We are interested is getting feedback about the usefulness of this bug pattern. For sending feedback, check:

- contributing guideline
- malinglist

In particular, the false-positive suppression heuristics for this bug pattern have not been extensively tuned, so reports about false positives are helpful to us.

See Weimer and Necula, *Finding and Preventing Run-Time Error Handling Mistakes*, for a description of the analysis technique.

RV_RETURN_VALUE_IGNORED_BAD_PRACTICE: Method ignores exceptional return value

This method returns a value that is not checked. The return value should be checked since it can indicate an unusual or unexpected function execution. For example, the `File.delete()` method returns false if the file could not be successfully deleted (rather than throwing an Exception). If you don't check the result, you won't notice if the method invocation signals unexpected behavior by returning an atypical return value.

ST_WRITE_TO_STATIC_FROM_INSTANCE_METHOD: Write to static field from instance method

This instance method writes to a static field. This is tricky to get correct if multiple instances are being manipulated, and generally bad practice.

UWF_UNWRITTEN_FIELD: Unwritten field

This field is never written. All reads of it will return the default value. Check for errors (should it have been initialized?), or remove it if it is useless.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixnet-prover/build/classes/java/main

Metrics

1491 lines of code analyzed, in 49 classes, in 9 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings	10	6.71
Total Warnings	**10**	**6.71**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Malicious code vulnerability Warnings	10
Total	**10**

Warnings

Click on a warning row to see full context information.

Malicious code vulnerability Warnings

Code	Warning
EI	com.scytl.products.ov.mixnet.proofs.bg.shuffle.ShuffleProofInputData.getEncryptedCiphertexts() may expose internal

	representation by returning ShuffleProofInputData._encryptedCiphertexts
EI	com.scytl.products.ov.mixnet.proofs.bg.shuffle.ShuffleProofInputData.getReencryptedCiphertexts() may expose internal representation by returning ShuffleProofInputData._reencryptedCiphertexts
EI	com.scytl.products.ov.mixnet.proofs.bg.shuffle.ShuffleProofInputData.getRho() may expose internal representation by returning ShuffleProofInputData._rho
EI2	new com.scytl.products.ov.mixnet.proofs.bg.multiexp.MultiExponentiationProofGenerator(ProofsGeneratorConfigurationParams, Ciphertext[][], Randomness[], BasicProofGenerator, ReductionProofGenerator) may expose internal representation by storing an externally mutable object into MultiExponentiationProofGenerator._ciphertexts
EI2	new com.scytl.products.ov.mixnet.proofs.bg.multiexp.MultiExponentiationProofGenerator(ProofsGeneratorConfigurationParams, Ciphertext[][], Randomness[], BasicProofGenerator, ReductionProofGenerator) may expose internal representation by storing an externally mutable object into MultiExponentiationProofGenerator._initialRandomExponents
EI2	new com.scytl.products.ov.mixnet.proofs.bg.shuffle.ShuffleProofGenerator(ProofsGeneratorConfigurationParams, CommitmentParams, Ciphertext[][], Permutation, PrivateAndPublicCommitmentsGenerator, MultiExponentiation, int, Ciphertext[][], Randomness[], ComputeAllE) may expose internal representation by storing an externally mutable object into ShuffleProofGenerator._originalCiphertexts
EI2	new com.scytl.products.ov.mixnet.proofs.bg.shuffle.ShuffleProofGenerator(ProofsGeneratorConfigurationParams, CommitmentParams, Ciphertext[][], Permutation, PrivateAndPublicCommitmentsGenerator, MultiExponentiation, int, Ciphertext[][], Randomness[], ComputeAllE) may expose internal representation by storing an externally mutable object into ShuffleProofGenerator._reEncryptedCiphertexts
EI2	new com.scytl.products.ov.mixnet.proofs.bg.shuffle.ShuffleProofInputData(Ciphertext[][], Permutation, PrivateAndPublicCommitmentsGenerator, Ciphertext[][], Randomness[]) may expose internal representation by storing an externally mutable object into ShuffleProofInputData._encryptedCiphertexts
EI2	new com.scytl.products.ov.mixnet.proofs.bg.shuffle.ShuffleProofInputData(Ciphertext[][], Permutation, PrivateAndPublicCommitmentsGenerator, Ciphertext[][], Randomness[]) may expose internal representation by storing an externally mutable object into ShuffleProofInputData._reencryptedCiphertexts
EI2	new com.scytl.products.ov.mixnet.proofs.bg.shuffle.ShuffleProofInputData(Ciphertext[][], Permutation, PrivateAndPublicCommitmentsGenerator, Ciphertext[][], Randomness[]) may expose internal representation by storing an externally mutable object into ShuffleProofInputData._rho

Details

EI_EXPOSE_REP: May expose internal representation by returning reference to mutable object

Returning a reference to a mutable object value stored in one of the object's fields exposes the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Returning a new copy of the object is better approach in many situations.

EI_EXPOSE_REP2: May expose internal representation by incorporating reference to mutable object

This code stores a reference to an externally mutable object into the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Storing a copy of the object is better approach in many situations.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixnet-spring/build/classes/java/main

Metrics

96 lines of code analyzed, in 2 classes, in 2 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixing-prover/build/classes/java/main

Metrics

2213 lines of code analyzed, in 56 classes, in 9 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings	18	8.13
Total Warnings	**18**	**8.13**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Malicious code vulnerability Warnings	18
Total	**18**

Warnings

Click on a warning row to see full context information.

Malicious code vulnerability Warnings

Code	Warning
EI	com.scytl.ov.mixing.proofs.bg.shuffle.ShuffleProofInputData.getEncryptedCiphertexts() may expose internal representation by returning ShuffleProofInputData._encryptedCiphertexts
EI	com.scytl.ov.mixing.proofs.bg.shuffle.ShuffleProofInputData.getReencryptedCiphertexts() may expose internal representation by returning ShuffleProofInputData._reencryptedCiphertexts
EI	com.scytl.ov.mixing.proofs.bg.shuffle.ShuffleProofInputData.getRho() may expose internal representation by returning ShuffleProofInputData._rho
EI2	new com.scytl.ov.mixing.proofs.bg hadamard.HadamardProductProofVerifier(int, int, CommitmentParams, PublicCommitment[], PublicCommitment, BigInteger, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into HadamardProductProofVerifier._cA
EI2	new com.scytl.ov.mixing.proofs.bg.multiexp.basic.MultiExponentiationBasicProofVerifier(int, int, Cryptosystem, CommitmentParams, Ciphertext[][], Ciphertext, PublicCommitment[], BigInteger, CiphertextTools, MultiExponentiation) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofVerifier._cA
EI2	new com.scytl.ov.mixing.proofs.bg.multiexp.basic.MultiExponentiationBasicProofVerifier(int, int, Cryptosystem, CommitmentParams, Ciphertext[][], Ciphertext, PublicCommitment[], BigInteger, CiphertextTools, MultiExponentiation) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofVerifier._vecC
EI2	new com.scytl.ov.mixing.proofs.bg.multiexp.MultiExponentiationProofGenerator(ProofsGeneratorConfigurationParams, Ciphertext[][], Randomness[], BasicProofGenerator, ReductionProofGenerator) may expose internal representation by storing an externally mutable object into MultiExponentiationProofGenerator._ciphertexts
EI2	new com.scytl.ov.mixing.proofs.bg.multiexp.MultiExponentiationProofGenerator(ProofsGeneratorConfigurationParams, Ciphertext[][], Randomness[], BasicProofGenerator, ReductionProofGenerator) may expose internal representation by storing an externally mutable object into MultiExponentiationProofGenerator._initialRandomExponents
EI2	new com.scytl.ov.mixing.proofs.bg.multiexp.reduction.MultiExponentiationReductionProofVerifier(int, int, Cryptosystem, ZpSubgroup, CommitmentParams, Ciphertext[][], Ciphertext, PublicCommitment[], int, int, CiphertextTools, MultiExponentiation) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionProofVerifier._cA
EI2	new com.scytl.ov.mixing.proofs.bg.multiexp.reduction.MultiExponentiationReductionProofVerifier(int, int, Cryptosystem, ZpSubgroup, CommitmentParams, Ciphertext[][], Ciphertext, PublicCommitment[], int, int, CiphertextTools, MultiExponentiation) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionProofVerifier._vecC
EI2	new com.scytl.ov.mixing.proofs.bg.product.ProductProofVerifier(int, int, CommitmentParams, PublicCommitment[], Exponent, BigInteger, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into ProductProofVerifier._cA
EI2	new com.scytl.ov.mixing.proofs.bg.shuffle.ShuffleProofGenerator(ProofsGeneratorConfigurationParams, Ciphertext[][], Permutation, PrivateAndPublicCommitmentsGenerator, MultiExponentiation, Ciphertext[][], Randomness[], ComputeAllE) may expose internal representation by storing an externally mutable object into ShuffleProofGenerator._originalCiphertexts
EI2	new com.scytl.ov.mixing.proofs.bg.shuffle.ShuffleProofGenerator(ProofsGeneratorConfigurationParams, Ciphertext[][], Permutation, PrivateAndPublicCommitmentsGenerator, MultiExponentiation, Ciphertext[][], Randomness[], ComputeAllE) may expose internal representation by storing an externally mutable object into ShuffleProofGenerator._reEncryptedCiphertexts
EI2	new com.scytl.ov.mixing.proofs.bg.shuffle.ShuffleProofInputData(Ciphertext[][], Permutation, PrivateAndPublicCommitmentsGenerator, Ciphertext[][], Randomness[]) may expose internal representation by storing an externally mutable object into ShuffleProofInputData._encryptedCiphertexts
EI2	new com.scytl.ov.mixing.proofs.bg.shuffle.ShuffleProofInputData(Ciphertext[][], Permutation, PrivateAndPublicCommitmentsGenerator, Ciphertext[][], Randomness[]) may expose internal representation by storing an externally mutable object into ShuffleProofInputData._reencryptedCiphertexts
EI2	new com.scytl.ov.mixing.proofs.bg.shuffle.ShuffleProofInputData(Ciphertext[][], Permutation, PrivateAndPublicCommitmentsGenerator, Ciphertext[][], Randomness[]) may expose internal representation by storing an externally mutable object into ShuffleProofInputData._rho

EI2	new com.scytl.ov.mixing.proofs.bg.shuffle.ShuffleProofVerifier(ZpSubgroup, Cryptosystem, CommitmentParams, Ciphertext[][], Ciphertext[][], BGParams, CiphertextTools, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into ShuffleProofVerifier._C
EI2	new com.scytl.ov.mixing.proofs.bg.shuffle.ShuffleProofVerifier(ZpSubgroup, Cryptosystem, CommitmentParams, Ciphertext[][], Ciphertext[][], BGParams, CiphertextTools, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into ShuffleProofVerifier._Cprime

Details

EI_EXPOSE_REP: May expose internal representation by returning reference to mutable object

Returning a reference to a mutable object value stored in one of the object's fields exposes the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Returning a new copy of the object is better approach in many situations.

EI_EXPOSE_REP2: May expose internal representation by incorporating reference to mutable object

This code stores a reference to an externally mutable object into the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Storing a copy of the object is better approach in many situations.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixnet-mixer/build/classes/java/main

Metrics

376 lines of code analyzed, in 12 classes, in 2 packages.

Metric	Total	Density*
High Priority Warnings	1	2.66
Medium Priority Warnings		0.00
Total Warnings	**1**	**2.66**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Internationalization Warnings	1
Total	**1**

Warnings

Click on a warning row to see full context information.

Internationalization Warnings

Code	Warning
Dm	Found reliance on default encoding in com.scytl.products.ov.mixnet.io.VotesWithProofWriter.write(OutputStream, OutputStream, Stream): new java.io.OutputStreamWriter(OutputStream)

Details

DM_DEFAULT_ENCODING: Reliance on default encoding

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixing-core/build/classes/java/main

Metrics

865 lines of code analyzed, in 38 classes, in 3 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixnet-verifier/build/classes/java/main

Metrics

841 lines of code analyzed, in 8 classes, in 8 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings	9	10.70
Total Warnings	**9**	**10.70**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Malicious code vulnerability Warnings	8
Dodgy code Warnings	1
Total	**9**

Warnings

Click on a warning row to see full context information.

Malicious code vulnerability Warnings

Code	Warning
EI2	new com.scytl.products.ov.mixnet.proofs.bg hadamard.HadamardProductProofVerifier(int, int, CommitmentParams, PublicCommitment[], PublicCommitment, BigInteger, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into HadamardProductProofVerifier._cA
EI2	new com.scytl.products.ov.mixnet.proofs.bg.multiexp.basic.MultiExponentiationBasicProofVerifier(int, int, Cryptosystem, CommitmentParams, Ciphertext[][], Ciphertext, PublicCommitment[], BigInteger, CiphertextTools, MultiExponentiation) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofVerifier._cA
EI2	new com.scytl.products.ov.mixnet.proofs.bg.multiexp.basic.MultiExponentiationBasicProofVerifier(int, int, Cryptosystem, CommitmentParams, Ciphertext[][], Ciphertext, PublicCommitment[], BigInteger, CiphertextTools, MultiExponentiation) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofVerifier._vecC
EI2	new com.scytl.products.ov.mixnet.proofs.bg.multiexp.reduction.MultiExponentiationReductionProofVerifier(int, int, Cryptosystem, ZpSubgroup, CommitmentParams, Ciphertext[][], Ciphertext, PublicCommitment[], int, int, CiphertextTools, MultiExponentiation) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionProofVerifier._cA
EI2	new com.scytl.products.ov.mixnet.proofs.bg.multiexp.reduction.MultiExponentiationReductionProofVerifier(int, int, Cryptosystem, ZpSubgroup, CommitmentParams, Ciphertext[][], Ciphertext, PublicCommitment[], int, int, CiphertextTools, MultiExponentiation) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionProofVerifier._vecC
EI2	new com.scytl.products.ov.mixnet.proofs.bg.product.ProductProofVerifier(int, int, CommitmentParams, PublicCommitment[], Exponent, BigInteger, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into ProductProofVerifier._cA
EI2	new com.scytl.products.ov.mixnet.proofs.bg.shuffle.ShuffleProofVerifier(ZpSubgroup, Cryptosystem, CommitmentParams, Ciphertext[][], Ciphertext[][], int, int, int, int, CiphertextTools, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into ShuffleProofVerifier._C
EI2	new com.scytl.products.ov.mixnet.proofs.bg.shuffle.ShuffleProofVerifier(ZpSubgroup, Cryptosystem, CommitmentParams, Ciphertext[][], Ciphertext[][], int, int, int, int, CiphertextTools, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into ShuffleProofVerifier._Cprime

Dodgy code Warnings

Code	Warning
REC	Exception is caught when Exception is not thrown in com.scytl.products.ov.mixnet.BGVerifier.verify(ApplicationConfig, LocationsProvider)

Details

EI_EXPOSE_REP2: May expose internal representation by incorporating reference to mutable object

This code stores a reference to an externally mutable object into the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Storing a copy of the object is better approach in many situations.

REC_CATCH_EXCEPTION: Exception is caught when Exception is not thrown

This method uses a try-catch block that catches Exception objects, but Exception is not thrown within the try block, and RuntimeException is not explicitly caught. It is a common bug pattern to say try { ... } catch (Exception e) { something } as a shorthand for catching a number of types of exception each of whose catch blocks is identical, but this construct also accidentally catches RuntimeException as well, masking potential bugs.

A better approach is to either explicitly catch the specific exceptions that are thrown, or to explicitly catch RuntimeException exception, rethrow it, and then catch all non-Runtime Exceptions, as shown below:

```
try {
    ...
} catch (RuntimeException e) {
    throw e;
} catch (Exception e) {
    ... deal with all non-runtime exceptions ...
}
```

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixnet-shuffler/build/classes/java/main

Metrics

182 lines of code analyzed, in 6 classes, in 3 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixing-commons/build/classes/java/main

Metrics

3416 lines of code analyzed, in 130 classes, in 25 packages.

Metric	Total	Density*
High Priority Warnings	6	1.76
Medium Priority Warnings	86	25.18
Total Warnings	**92**	**26.93**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Bad practice Warnings	7
Internationalization Warnings	6
Malicious code vulnerability Warnings	72
Performance Warnings	3
Dodgy code Warnings	4
Total	**92**

Warnings

Click on a warning row to see full context information.

Bad practice Warnings

Code	Warning
RV	Exceptional return value of java.io.File.mkdirs() ignored in com.scytl.ov.mixing.commons.configuration.file.FileBasedApplicationConfig.convertMixer(FileBasedMixerConfig, Integer, Integer)
RV	Exceptional return value of java.io.File.mkdirs() ignored in com.scytl.ov.mixing.commons.configuration.file.FileBasedApplicationConfig.convertVerifier(FileBasedVerifierConfig, Integer, Integer)
Se	Class com.scytl.ov.mixing.commons.concurrent.LoopParallelizerAction defines non-transient non-serializable instance field _ranges
Se	Class com.scytl.ov.mixing.commons.concurrent.LoopParallelizerAction defines non-transient non-serializable instance field process
Se	Class com.scytl.ov.mixing.commons.concurrent.LoopParallelizerTask defines non-transient non-serializable instance field _operation
Se	Class com.scytl.ov.mixing.commons.concurrent.LoopParallelizerTask defines non-transient non-serializable instance field _ranges
Se	Class com.scytl.ov.mixing.commons.concurrent.LoopParallelizerTask defines non-transient non-serializable instance field process

Internationalization Warnings

Code	Warning
Dm	Found reliance on default encoding in com.scytl.ov.mixing.commons.configuration.file.FileBasedApplicationConfig.convertToStreamConfig(FileBasedApplicationConfig): new java.io.FileReader(File)
Dm	Found reliance on default encoding in com.scytl.ov.mixing.commons.io.CommitmentParamsReader.readCommitmentParamsFromStream(ZpSubgroup, InputStream): new java.io.InputStreamReader(InputStream)
Dm	Found reliance on default encoding in com.scytl.ov.mixing.commons.io.ElGamalEncryptedBallotsLoader.loadCSV(ZpSubgroup, InputStream): new java.io.InputStreamReader(InputStream)
Dm	Found reliance on default encoding in com.scytl.ov.mixing.commons.io.ElgamalPublicKeyReader.readPublicKeyFromStream(InputStream): new java.io.InputStreamReader(InputStream)
Dm	Found reliance on default encoding in new com.scytl.ov.mixing.commons.io.HashCalculatingWriter(OutputStream, MessageDigest): new com.scytl.ov.mixing.commons.io.HashCalculatingWriter(OutputStream)
Dm	Found reliance on default encoding in com.scytl.ov.mixing.commons.tools.RandomOracleHash.getHash(): String.getBytes()

Malicious code vulnerability Warnings

Code	Warning
EI	com.scytl.ov.mixing.commons.beans.proofs.BayerGrothProof.getFirstAnswer() may expose internal representation by returning BayerGrothProof._firstAnswer
EI	com.scytl.ov.mixing.commons.beans.proofs.BayerGrothProof.getInitialMessage() may expose internal representation by returning BayerGrothProof._initialMessage
EI	com.scytl.ov.mixing.commons.beans.proofs.HadamardProductProofInitialMessage.getCommitmentPublicB() may expose internal representation by returning HadamardProductProofInitialMessage._cB
EI	com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationBasicProofAnswer.getExponentsA() may expose internal representation by returning MultiExponentiationBasicProofAnswer._a
EI	com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationBasicProofInitialMessage.getCiphertextsE() may expose internal representation by returning MultiExponentiationBasicProofInitialMessage._E
EI	com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationBasicProofInitialMessage.getCommitmentPublicB() may expose internal representation by returning MultiExponentiationBasicProofInitialMessage._cB
EI	com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationReductionAnswer.getExponentsB() may expose internal representation by returning MultiExponentiationReductionAnswer._b
EI	com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationReductionInitialMessage.getCiphertextsE() may expose internal representation by returning MultiExponentiationReductionInitialMessage._E
EI	com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationReductionInitialMessage.getCommitmentPublicB() may expose internal representation by returning MultiExponentiationReductionInitialMessage._cb
EI	com.scytl.ov.mixing.commons.beans.proofs.SingleValueProductProofAnswer.getExponentsTildeA() may expose internal representation by returning SingleValueProductProofAnswer._tildeA

EI	com.scytl.ov.mixing.commons.beans.proofs.SingleValueProductProofAnswer.getExponentsTildeB() may expose internal representation by returning SingleValueProductProofAnswer._tildeB
EI	com.scytl.ov.mixing.commons.beans.proofs.ZeroProofAnswer.getExponentsA() may expose internal representation by returning ZeroProofAnswer._a
EI	com.scytl.ov.mixing.commons.beans.proofs.ZeroProofAnswer.getExponentsB() may expose internal representation by returning ZeroProofAnswer._b
EI	com.scytl.ov.mixing.commons.beans.proofs.ZeroProofInitialMessage.getCommitmentPublicD() may expose internal representation by returning ZeroProofInitialMessage._cD
EI	com.scytl.ov.mixing.commons.beans.ShuffleOutput.getExponents() may expose internal representation by returning ShuffleOutput._exponents
EI	com.scytl.ov.mixing.commons.beans.WarmUpOutput.getRandomExponents() may expose internal representation by returning WarmUpOutput._randomExponents
EI	com.scytl.ov.mixing.commons.proofs.bg.commitments.CommitmentParams.getG() may expose internal representation by returning CommitmentParams._g
EI	com.scytl.ov.mixing.commons.proofs.bg.commitments.PrivateCommitment.getM() may expose internal representation by returning PrivateCommitment._exponents
EI2	new com.scytl.ov.mixing.commons.beans.proofs.BayerGrothProof(PublicCommitment[], PublicCommitment[], ShuffleProofSecondAnswer, CommitmentParams) may expose internal representation by storing an externally mutable object into BayerGrothProof._firstAnswer
EI2	new com.scytl.ov.mixing.commons.beans.proofs.BayerGrothProof(PublicCommitment[], PublicCommitment[], ShuffleProofSecondAnswer, CommitmentParams) may expose internal representation by storing an externally mutable object into BayerGrothProof._initialMessage
EI2	com.scytl.ov.mixing.commons.beans.proofs.BayerGrothProof$Builder.withFirstAnswer(PublicCommitment[]) may expose internal representation by storing an externally mutable object into BayerGrothProof$Builder._firstAnswer
EI2	com.scytl.ov.mixing.commons.beans.proofs.BayerGrothProof$Builder.withInitialMessage(PublicCommitment[]) may expose internal representation by storing an externally mutable object into BayerGrothProof$Builder._initialMessage
EI2	new com.scytl.ov.mixing.commons.beans.proofs.HadamardProductProofInitialMessage(PublicCommitment[], int) may expose internal representation by storing an externally mutable object into HadamardProductProofInitialMessage._cB
EI2	new com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationBasicProofAnswer(Exponent[], Exponent, Exponent, Exponent, Randomness) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofAnswer._a
EI2	new com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationBasicProofInitialMessage(PrivateCommitment, PrivateCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofInitialMessage._E
EI2	new com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationBasicProofInitialMessage(PublicCommitment, PublicCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofInitialMessage._E
EI2	new com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationBasicProofInitialMessage(PublicCommitment, PublicCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofInitialMessage._cB
EI2	new com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationReductionAnswer(Exponent[], Exponent, MultiExponentiationBasicProofInitialMessage, MultiExponentiationBasicProofAnswer) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionAnswer._b
EI2	new com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationReductionAnswer(Exponent[], Exponent, MultiExponentiationBasicProofInitialMessage, MultiExponentiationBasicProofAnswer, MultiExponentiationReductionInitialMessage, MultiExponentiationReductionAnswer) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionAnswer._b
EI2	new com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationReductionAnswer(Exponent[], Exponent, MultiExponentiationReductionInitialMessage, MultiExponentiationReductionAnswer) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionAnswer._b
EI2	new com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationReductionInitialMessage(PrivateCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionInitialMessage._E
EI2	new com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationReductionInitialMessage(PublicCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionInitialMessage._E
EI2	new com.scytl.ov.mixing.commons.beans.proofs.MultiExponentiationReductionInitialMessage(PublicCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionInitialMessage._cb
EI2	new com.scytl.ov.mixing.commons.beans.proofs.SingleValueProductProofAnswer(Exponent[], Exponent[], Exponent, Exponent)

	may expose internal representation by storing an externally mutable object into SingleValueProductProofAnswer._tildeA
EI2	new com.scytl.ov.mixing.commons.beans.proofs.SingleValueProductProofAnswer(Exponent[], Exponent[], Exponent, Exponent) may expose internal representation by storing an externally mutable object into SingleValueProductProofAnswer._tildeB
EI2	new com.scytl.ov.mixing.commons.beans.proofs.ZeroProofAnswer(Exponent[], Exponent[], Exponent, Exponent, Exponent) may expose internal representation by storing an externally mutable object into ZeroProofAnswer._a
EI2	new com.scytl.ov.mixing.commons.beans.proofs.ZeroProofAnswer(Exponent[], Exponent[], Exponent, Exponent, Exponent) may expose internal representation by storing an externally mutable object into ZeroProofAnswer._b
EI2	new com.scytl.ov.mixing.commons.beans.proofs.ZeroProofInitialMessage(PublicCommitment, PublicCommitment, PublicCommitment[], int) may expose internal representation by storing an externally mutable object into ZeroProofInitialMessage._cD
EI2	new com.scytl.ov.mixing.commons.beans.ShuffleOutput(Permutation, Randomness[], Object) may expose internal representation by storing an externally mutable object into ShuffleOutput._exponents
EI2	new com.scytl.ov.mixing.commons.beans.WarmUpOutput(Randomness[], List) may expose internal representation by storing an externally mutable object into WarmUpOutput._randomExponents
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.A0Processor(Ciphertext[], Ciphertext[][], Exponent[], CiphertextTools, int) may expose internal representation by storing an externally mutable object into A0Processor._B
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.A0Processor(Ciphertext[], Ciphertext[][], Exponent[], CiphertextTools, int) may expose internal representation by storing an externally mutable object into A0Processor._E
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.A0Processor(Ciphertext[], Ciphertext[][], Exponent[], CiphertextTools, int) may expose internal representation by storing an externally mutable object into A0Processor._Ex
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.EncryptConcurrentCalculatorProcessor(Randomness[], List, Cryptosystem, ZpSubgroup) may expose internal representation by storing an externally mutable object into EncryptConcurrentCalculatorProcessor._randomExponents
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.EncryptRaisingToRandomProcessor(Ciphertext[], Exponent[], Randomness[], Cryptosystem) may expose internal representation by storing an externally mutable object into EncryptRaisingToRandomProcessor._E
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.EncryptRaisingToRandomProcessor(Ciphertext[], Exponent[], Randomness[], Cryptosystem) may expose internal representation by storing an externally mutable object into EncryptRaisingToRandomProcessor._b
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.EncryptRaisingToRandomProcessor(Ciphertext[], Exponent[], Randomness[], Cryptosystem) may expose internal representation by storing an externally mutable object into EncryptRaisingToRandomProcessor._tau
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.EvaluatorProcessor(Ciphertext[], int, int, Exponent[][], Ciphertext[][], BigInteger, CiphertextTools, Exponent[][]) may expose internal representation by storing an externally mutable object into EvaluatorProcessor._a
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.EvaluatorProcessor(Ciphertext[], int, int, Exponent[][], Ciphertext[][], BigInteger, CiphertextTools, Exponent[][]) may expose internal representation by storing an externally mutable object into EvaluatorProcessor._evalC
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.EvaluatorProcessor(Ciphertext[], int, int, Exponent[][], Ciphertext[][], BigInteger, CiphertextTools, Exponent[][]) may expose internal representation by storing an externally mutable object into EvaluatorProcessor._vandermondeOmega
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.EvaluatorProcessor(Ciphertext[], int, int, Exponent[][], Ciphertext[][], BigInteger, CiphertextTools, Exponent[][]) may expose internal representation by storing an externally mutable object into EvaluatorProcessor._vecC
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.InterpolatorProcessor(Ciphertext[], Ciphertext[], Exponent[][], CiphertextTools) may expose internal representation by storing an externally mutable object into InterpolatorProcessor._C
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.InterpolatorProcessor(Ciphertext[], Ciphertext[], Exponent[][], CiphertextTools) may expose internal representation by storing an externally mutable object into InterpolatorProcessor._E
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.InterpolatorProcessor(Ciphertext[], Ciphertext[], Exponent[][], CiphertextTools) may expose internal representation by storing an externally mutable object into InterpolatorProcessor._e
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.LimMultiExpoConcurrentCalculatorProcess(ZpGroupElement[], Exponent[], CommitmentParams, MultiExponentiation) may expose internal representation by storing an externally mutable object into LimMultiExpoConcurrentCalculatorProcess._base
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.LimMultiExpoConcurrentCalculatorProcess(ZpGroupElement[], Exponent[], CommitmentParams, MultiExponentiation) may expose internal representation by storing an externally mutable object into LimMultiExpoConcurrentCalculatorProcess._exponents
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.NextCiphertextsAndCAPrimeProcessor(Ciphertext[][], PrivateCommitment[], Ciphertext[][], int, PrivateCommitment[], Exponent, BigInteger, int) may expose internal representation by storing an externally mutable object into NextCiphertextsAndCAPrimeProcessor._cA
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.NextCiphertextsAndCAPrimeProcessor(Ciphertext[][],

	PrivateCommitment[], Ciphertext[][], int, PrivateCommitment[], Exponent, BigInteger, int) may expose internal representation by storing an externally mutable object into NextCiphertextsAndCAPrimeProcessor._cAprime
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.NextCiphertextsAndCAPrimeProcessor(Ciphertext[][], PrivateCommitment[], Ciphertext[][], int, PrivateCommitment[], Exponent, BigInteger, int) may expose internal representation by storing an externally mutable object into NextCiphertextsAndCAPrimeProcessor._ciphertexts
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.NextCiphertextsAndCAPrimeProcessor(Ciphertext[][], PrivateCommitment[], Ciphertext[][], int, PrivateCommitment[], Exponent, BigInteger, int) may expose internal representation by storing an externally mutable object into NextCiphertextsAndCAPrimeProcessor._nextCiphertexts
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.PreComputationCalculatorProcessor(ZpGroupElement[], Randomness[], Cryptosystem) may expose internal representation by storing an externally mutable object into PreComputationCalculatorProcessor._arrayOfIdentityElement
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.PreComputationCalculatorProcessor(ZpGroupElement[], Randomness[], Cryptosystem) may expose internal representation by storing an externally mutable object into PreComputationCalculatorProcessor._requiredRandomExponents
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.PrivateAndPublicCommitmentProcessor(PrivateCommitment[], PublicCommitment[], Exponent[][], Exponent[], CommitmentParams, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into PrivateAndPublicCommitmentProcessor._exponents
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.PrivateAndPublicCommitmentProcessor(PrivateCommitment[], PublicCommitment[], Exponent[][], Exponent[], CommitmentParams, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into PrivateAndPublicCommitmentProcessor._privateCommitments
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.PrivateAndPublicCommitmentProcessor(PrivateCommitment[], PublicCommitment[], Exponent[][], Exponent[], CommitmentParams, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into PrivateAndPublicCommitmentProcessor._publicCommitments
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.PrivateAndPublicCommitmentProcessor(PrivateCommitment[], PublicCommitment[], Exponent[][], Exponent[], CommitmentParams, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into PrivateAndPublicCommitmentProcessor._r
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.PublicCommitmentCalculatorProcess(PublicCommitment[], PrivateCommitment[]) may expose internal representation by storing an externally mutable object into PublicCommitmentCalculatorProcess._privateCommitments
EI2	new com.scytl.ov.mixing.commons.concurrent.processor.PublicCommitmentCalculatorProcess(PublicCommitment[], PrivateCommitment[]) may expose internal representation by storing an externally mutable object into PublicCommitmentCalculatorProcess._publicCommitments
EI2	new com.scytl.ov.mixing.commons.homomorphic.impl.GjosteenElGamal(ZpSubgroup, Exponent[]) may expose internal representation by storing an externally mutable object into GjosteenElGamal._privateKey
EI2	new com.scytl.ov.mixing.commons.homomorphic.impl.GjosteenElGamalPlaintext(ZpGroupElement[]) may expose internal representation by storing an externally mutable object into GjosteenElGamalPlaintext._m
EI2	new com.scytl.ov.mixing.commons.mathematical.tools.LUDecomposition(Exponent[][], BigInteger) may expose internal representation by storing an externally mutable object into LUDecomposition._lu
EI2	new com.scytl.ov.mixing.commons.proofs.bg.commitments.CommitmentParams(ZpSubgroup, ZpGroupElement, ZpGroupElement[]) may expose internal representation by storing an externally mutable object into CommitmentParams._g

Performance Warnings

Code	Warning
SBSC	com.scytl.ov.mixing.commons.io.ciphertext.CiphertextSerializer.serialize(Ciphertext, JsonGenerator, SerializerProvider) concatenates strings using + in a loop
SS	Unread field: com.scytl.ov.mixing.commons.io.ElGamalEncryptedBallotEntryParser.PATTERN; should this field be static?
SS	Unread field: com.scytl.ov.mixing.commons.tools.UniversalElGamalEncryptionParamsGenerator.CERTAINTY_LEVEL; should this field be static?

Dodgy code Warnings

Code	Warning
BC	instanceof will always return true for all non-null values in com.scytl.ov.mixing.commons.concurrent.processor.LimMultiExpoConcurrentCalculatorProcess.getZpGroup(ZpSubgroup), since all com.scytl.cryptolib.mathematical.groups.impl.ZpSubgroup are instances of com.scytl.cryptolib.mathematical.groups.impl.ZpSubgroup
BC	instanceof will always return true for all non-null values in com.scytl.ov.mixing.commons.proofs.bg.commitments.CommitmentParams.extractZpGroup(ZpSubgroup), since all

	com.scytl.cryptolib.mathematical.groups.impl.ZpSubgroup are instances of com.scytl.cryptolib.mathematical.groups.impl.ZpSubgroup
DLS	Dead store to reconstructedZpGroup in com.scytl.ov.mixing.commons.io.CommitmentParamsReader.readCommitmentParamsFromStream(ZpSubgroup, InputStream)
RV	com.scytl.ov.mixing.commons.configuration.file.FileBasedApplicationConfig.convertToStreamConfig(FileBasedApplicationConfig) discards result of readLine after checking if it is non-null

Details

BC_VACUOUS_INSTANCEOF: instanceof will always return true

This instanceof test will always return true (unless the value being tested is null). Although this is safe, make sure it isn't an indication of some misunderstanding or some other logic error. If you really want to test the value for being null, perhaps it would be clearer to do better to do a null test rather than an instanceof test.

DLS_DEAD_LOCAL_STORE: Dead store to local variable

This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error, because the value computed is never used.

Note that Sun's javac compiler often generates dead stores for final local variables. Because SpotBugs is a bytecode-based tool, there is no easy way to eliminate these false positives.

DM_DEFAULT_ENCODING: Reliance on default encoding

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

EI_EXPOSE_REP: May expose internal representation by returning reference to mutable object

Returning a reference to a mutable object value stored in one of the object's fields exposes the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Returning a new copy of the object is better approach in many situations.

EI_EXPOSE_REP2: May expose internal representation by incorporating reference to mutable object

This code stores a reference to an externally mutable object into the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Storing a copy of the object is better approach in many situations.

RV_DONT_JUST_NULL_CHECK_READLINE: Method discards result of readLine after checking if it is non-null

The value returned by readLine is discarded after checking to see if the return value is non-null. In almost all situations, if the result is non-null, you will want to use that non-null value. Calling readLine again will give you a different line.

RV_RETURN_VALUE_IGNORED_BAD_PRACTICE: Method ignores exceptional return value

This method returns a value that is not checked. The return value should be checked since it can indicate an unusual or unexpected function execution. For example, the `File.delete()` method returns false if the file could not be successfully deleted (rather than throwing an Exception). If you don't check the result, you won't notice if the method invocation signals unexpected behavior by returning an atypical return value.

SBSC_USE_STRINGBUFFER_CONCATENATION: Method concatenates strings using + in a loop

The method seems to be building a String using concatenation in a loop. In each iteration, the String is converted to a StringBuffer/StringBuilder, appended to, and converted back to a String. This can lead to a cost quadratic in the number of iterations, as the growing string is recopied in each iteration.

Better performance can be obtained by using a StringBuffer (or StringBuilder in Java 1.5) explicitly.

For example:

```
// This is bad
String s = "";
for (int i = 0; i < field.length; ++i) {
    s = s + field[i];
}

// This is better
StringBuffer buf = new StringBuffer();
for (int i = 0; i < field.length; ++i) {
    buf.append(field[i]);
}
String s = buf.toString();
```

SE_BAD_FIELD: Non-transient non-serializable instance field in serializable class

This Serializable class defines a non-primitive instance field which is neither transient, Serializable, or `java.lang.Object`, and does not appear to implement the `Externalizable` interface or the `readObject()` and `writeObject()` methods. Objects of this class will not be deserialized correctly if a non-Serializable object is stored in this field.

SS_SHOULD_BE_STATIC: Unread field: should this field be static?

This class contains an instance final field that is initialized to a compile-time static value. Consider making the field static.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/mixnet-verifier/mixnet-commons/build/classes/java/main

Metrics

3355 lines of code analyzed, in 128 classes, in 24 packages.

Metric	Total	Density*
High Priority Warnings	6	1.79
Medium Priority Warnings	86	25.63
Total Warnings	**92**	**27.42**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Bad practice Warnings	7
Internationalization Warnings	6
Malicious code vulnerability Warnings	72
Performance Warnings	3
Dodgy code Warnings	4
Total	**92**

Warnings

Click on a warning row to see full context information.

Bad practice Warnings

Code	Warning
RV	Exceptional return value of java.io.File.mkdirs() ignored in com.scytl.products.ov.mixnet.commons.configuration.file.FileBasedApplicationConfig.convertMixer(FileBasedMixerConfig, Integer, Integer)
RV	Exceptional return value of java.io.File.mkdirs() ignored in com.scytl.products.ov.mixnet.commons.configuration.file.FileBasedApplicationConfig.convertVerifier(FileBasedVerifierConfig, Integer, Integer)
Se	Class com.scytl.products.ov.mixnet.commons.concurrent.LoopParallelizerAction defines non-transient non-serializable instance field _ranges
Se	Class com.scytl.products.ov.mixnet.commons.concurrent.LoopParallelizerAction defines non-transient non-serializable instance field process
Se	Class com.scytl.products.ov.mixnet.commons.concurrent.LoopParallelizerTask defines non-transient non-serializable instance field _operation
Se	Class com.scytl.products.ov.mixnet.commons.concurrent.LoopParallelizerTask defines non-transient non-serializable instance field _ranges
Se	Class com.scytl.products.ov.mixnet.commons.concurrent.LoopParallelizerTask defines non-transient non-serializable instance field process

Internationalization Warnings

Code	Warning
Dm	Found reliance on default encoding in com.scytl.products.ov.mixnet.commons.configuration.file.FileBasedApplicationConfig.convertToStreamConfig(FileBasedApplicationConfig): new java.io.FileReader(File)
Dm	Found reliance on default encoding in com.scytl.products.ov.mixnet.commons.io.CommitmentParamsReader.readCommitmentParamsFromStream(ZpSubgroup, InputStream): new java.io.InputStreamReader(InputStream)
Dm	Found reliance on default encoding in com.scytl.products.ov.mixnet.commons.io.ElGamalEncryptedBallotsLoader.loadCSV(ZpSubgroup, InputStream): new java.io.InputStreamReader(InputStream)
Dm	Found reliance on default encoding in com.scytl.products.ov.mixnet.commons.io.ElgamalPublicKeyReader.readPublicKeyFromStream(InputStream): new java.io.InputStreamReader(InputStream)
Dm	Found reliance on default encoding in new com.scytl.products.ov.mixnet.commons.io.HashCalculatingWriter(OutputStream, MessageDigest): new com.scytl.products.ov.mixnet.commons.io.HashCalculatingWriter(OutputStream)
Dm	Found reliance on default encoding in com.scytl.products.ov.mixnet.commons.tools.RandomOracleHash.getHash(): String.getBytes()

Malicious code vulnerability Warnings

Code	Warning
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.HadamardProductProofInitialMessage.getCommitmentPublicB() may expose internal representation by returning HadamardProductProofInitialMessage._cB
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationBasicProofAnswer.getExponentsA() may expose internal representation by returning MultiExponentiationBasicProofAnswer._a
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationBasicProofInitialMessage.getCiphertextsE() may expose internal representation by returning MultiExponentiationBasicProofInitialMessage._E
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationBasicProofInitialMessage.getCommitmentPublicB() may expose internal representation by returning MultiExponentiationBasicProofInitialMessage._cB
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationReductionAnswer.getExponentsB() may expose internal representation by returning MultiExponentiationReductionAnswer._b
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationReductionInitialMessage.getCiphertextsE() may expose internal representation by returning MultiExponentiationReductionInitialMessage._E
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationReductionInitialMessage.getCommitmentPublicB() may expose internal representation by returning MultiExponentiationReductionInitialMessage._cb
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.ShuffleProof.getFirstAnswer() may expose internal representation by returning ShuffleProof._firstAnswer
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.ShuffleProof.getInitialMessage() may expose internal representation by returning ShuffleProof._initialMessage
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.SingleValueProductProofAnswer.getExponentsTildeA() may expose internal representation by returning SingleValueProductProofAnswer._tildeA
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.SingleValueProductProofAnswer.getExponentsTildeB() may expose internal representation by returning SingleValueProductProofAnswer._tildeB
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.ZeroProofAnswer.getExponentsA() may expose internal representation by returning ZeroProofAnswer._a
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.ZeroProofAnswer.getExponentsB() may expose internal representation by returning

	ZeroProofAnswer._b
EI	com.scytl.products.ov.mixnet.commons.beans.proofs.ZeroProofInitialMessage.getCommitmentPublicD() may expose internal representation by returning ZeroProofInitialMessage._cD
EI	com.scytl.products.ov.mixnet.commons.beans.ShuffleOutput.getExponents() may expose internal representation by returning ShuffleOutput._exponents
EI	com.scytl.products.ov.mixnet.commons.beans.WarmUpOutput.getRandomExponents() may expose internal representation by returning WarmUpOutput._randomExponents
EI	com.scytl.products.ov.mixnet.commons.proofs.bg.commitments.CommitmentParams.getG() may expose internal representation by returning CommitmentParams._g
EI	com.scytl.products.ov.mixnet.commons.proofs.bg.commitments.PrivateCommitment.getM() may expose internal representation by returning PrivateCommitment._exponents
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.HadamardProductProofInitialMessage(PublicCommitment[], int) may expose internal representation by storing an externally mutable object into HadamardProductProofInitialMessage._cB
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationBasicProofAnswer(Exponent[], Exponent, Exponent, Exponent, Randomness) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofAnswer._a
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationBasicProofInitialMessage(PrivateCommitment, PrivateCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofInitialMessage._E
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationBasicProofInitialMessage(PublicCommitment, PublicCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofInitialMessage._E
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationBasicProofInitialMessage(PublicCommitment, PublicCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationBasicProofInitialMessage._cB
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationReductionAnswer(Exponent[], Exponent, MultiExponentiationBasicProofInitialMessage, MultiExponentiationBasicProofAnswer) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionAnswer._b
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationReductionAnswer(Exponent[], Exponent, MultiExponentiationBasicProofInitialMessage, MultiExponentiationBasicProofAnswer, MultiExponentiationReductionInitialMessage, MultiExponentiationReductionAnswer) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionAnswer._b
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationReductionAnswer(Exponent[], Exponent, MultiExponentiationReductionInitialMessage, MultiExponentiationReductionAnswer) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionAnswer._b
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationReductionInitialMessage(PrivateCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionInitialMessage._E
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationReductionInitialMessage(PublicCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionInitialMessage._E
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.MultiExponentiationReductionInitialMessage(PublicCommitment[], Ciphertext[], int) may expose internal representation by storing an externally mutable object into MultiExponentiationReductionInitialMessage._cb
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.ShuffleProof(PublicCommitment[], PublicCommitment[], ShuffleProofSecondAnswer) may expose internal representation by storing an externally mutable object into ShuffleProof._firstAnswer
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.ShuffleProof(PublicCommitment[], PublicCommitment[], ShuffleProofSecondAnswer) may expose internal representation by storing an externally mutable object into ShuffleProof._initialMessage
EI2	com.scytl.products.ov.mixnet.commons.beans.proofs.ShuffleProof$Builder.withFirstAnswer(PublicCommitment[]) may expose internal representation by storing an externally mutable object into ShuffleProof$Builder._firstAnswer
EI2	com.scytl.products.ov.mixnet.commons.beans.proofs.ShuffleProof$Builder.withInitialMessage(PublicCommitment[]) may expose internal representation by storing an externally mutable object into ShuffleProof$Builder._initialMessage
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.SingleValueProductProofAnswer(Exponent[], Exponent[], Exponent, Exponent) may expose internal representation by storing an externally mutable object into SingleValueProductProofAnswer._tildeA
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.SingleValueProductProofAnswer(Exponent[], Exponent[], Exponent, Exponent) may expose internal representation by storing an externally mutable object into SingleValueProductProofAnswer._tildeB
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.ZeroProofAnswer(Exponent[], Exponent[], Exponent, Exponent, Exponent) may expose internal representation by storing an externally mutable object into ZeroProofAnswer._a
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.ZeroProofAnswer(Exponent[], Exponent[], Exponent, Exponent, Exponent) may expose internal representation by storing an externally mutable object into ZeroProofAnswer._b
EI2	new com.scytl.products.ov.mixnet.commons.beans.proofs.ZeroProofInitialMessage(PublicCommitment, PublicCommitment, PublicCommitment[], int) may expose internal representation by storing an externally mutable object into ZeroProofInitialMessage._cD
EI2	new com.scytl.products.ov.mixnet.commons.beans.ShuffleOutput(Permutation, Randomness[], Object) may expose internal representation by storing an externally mutable object into ShuffleOutput._exponents
EI2	new com.scytl.products.ov.mixnet.commons.beans.WarmUpOutput(Randomness[], List) may expose internal representation by storing an externally mutable object into WarmUpOutput._randomExponents

EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.A0Processor(Ciphertext[], Ciphertext[][], Exponent[], CiphertextTools, int) may expose internal representation by storing an externally mutable object into A0Processor._B
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.A0Processor(Ciphertext[], Ciphertext[][], Exponent[], CiphertextTools, int) may expose internal representation by storing an externally mutable object into A0Processor._E
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.A0Processor(Ciphertext[], Ciphertext[][], Exponent[], CiphertextTools, int) may expose internal representation by storing an externally mutable object into A0Processor._Ex
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.EncryptConcurrentCalculatorProcessor(Randomness[], List, Cryptosystem, ZpSubgroup) may expose internal representation by storing an externally mutable object into EncryptConcurrentCalculatorProcessor._randomExponents
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.EncryptRaisingToRandomProcessor(Ciphertext[], Exponent[], Randomness[], Cryptosystem) may expose internal representation by storing an externally mutable object into EncryptRaisingToRandomProcessor._E
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.EncryptRaisingToRandomProcessor(Ciphertext[], Exponent[], Randomness[], Cryptosystem) may expose internal representation by storing an externally mutable object into EncryptRaisingToRandomProcessor._b
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.EncryptRaisingToRandomProcessor(Ciphertext[], Exponent[], Randomness[], Cryptosystem) may expose internal representation by storing an externally mutable object into EncryptRaisingToRandomProcessor._tau
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.EvaluatorProcessor(Ciphertext[], int, int, Exponent[][], Ciphertext[][], BigInteger, CiphertextTools, Exponent[][]) may expose internal representation by storing an externally mutable object into EvaluatorProcessor._a
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.EvaluatorProcessor(Ciphertext[], int, int, Exponent[][], Ciphertext[][], BigInteger, CiphertextTools, Exponent[][]) may expose internal representation by storing an externally mutable object into EvaluatorProcessor._evalC
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.EvaluatorProcessor(Ciphertext[], int, int, Exponent[][], Ciphertext[][], BigInteger, CiphertextTools, Exponent[][]) may expose internal representation by storing an externally mutable object into EvaluatorProcessor._vandermondeOmega
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.EvaluatorProcessor(Ciphertext[], int, int, Exponent[][], Ciphertext[][], BigInteger, CiphertextTools, Exponent[][]) may expose internal representation by storing an externally mutable object into EvaluatorProcessor._vecC
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.InterpolatorProcessor(Ciphertext[], Ciphertext[], Exponent[][], CiphertextTools) may expose internal representation by storing an externally mutable object into InterpolatorProcessor._C
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.InterpolatorProcessor(Ciphertext[], Ciphertext[], Exponent[][], CiphertextTools) may expose internal representation by storing an externally mutable object into InterpolatorProcessor._E
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.InterpolatorProcessor(Ciphertext[], Ciphertext[], Exponent[][], CiphertextTools) may expose internal representation by storing an externally mutable object into InterpolatorProcessor._e
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.LimMultiExpoConcurrentCalculatorProcess(ZpGroupElement[], Exponent[], CommitmentParams, MultiExponentiation) may expose internal representation by storing an externally mutable object into LimMultiExpoConcurrentCalculatorProcess._base
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.LimMultiExpoConcurrentCalculatorProcess(ZpGroupElement[], Exponent[], CommitmentParams, MultiExponentiation) may expose internal representation by storing an externally mutable object into LimMultiExpoConcurrentCalculatorProcess._exponents
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.NextCiphertextsAndCAPrimeProcessor(Ciphertext[][], PrivateCommitment[], Ciphertext[][], int, PrivateCommitment[], Exponent, BigInteger, int) may expose internal representation by storing an externally mutable object into NextCiphertextsAndCAPrimeProcessor._cA
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.NextCiphertextsAndCAPrimeProcessor(Ciphertext[][], PrivateCommitment[], Ciphertext[][], int, PrivateCommitment[], Exponent, BigInteger, int) may expose internal representation by storing an externally mutable object into NextCiphertextsAndCAPrimeProcessor._cAprime
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.NextCiphertextsAndCAPrimeProcessor(Ciphertext[][], PrivateCommitment[], Ciphertext[][], int, PrivateCommitment[], Exponent, BigInteger, int) may expose internal representation by storing an externally mutable object into NextCiphertextsAndCAPrimeProcessor._ciphertexts
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.NextCiphertextsAndCAPrimeProcessor(Ciphertext[][], PrivateCommitment[], Ciphertext[][], int, PrivateCommitment[], Exponent, BigInteger, int) may expose internal representation by storing an externally mutable object into NextCiphertextsAndCAPrimeProcessor._nextCiphertexts
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.PreComputationCalculatorProcessor(ZpGroupElement[], Randomness[], Cryptosystem) may expose internal representation by storing an externally mutable object into PreComputationCalculatorProcessor._arrayOfIdentityElement
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.PreComputationCalculatorProcessor(ZpGroupElement[], Randomness[], Cryptosystem) may expose internal representation by storing an externally mutable object into PreComputationCalculatorProcessor._requiredRandomExponents
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.PrivateAndPublicCommitmentProcessor(PrivateCommitment[], PublicCommitment[], Exponent[][], Exponent[], CommitmentParams, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into PrivateAndPublicCommitmentProcessor._exponents
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.PrivateAndPublicCommitmentProcessor(PrivateCommitment[],

	PublicCommitment[], Exponent[][], Exponent[], CommitmentParams, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into PrivateAndPublicCommitmentProcessor._privateCommitments
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.PrivateAndPublicCommitmentProcessor(PrivateCommitment[], PublicCommitment[], Exponent[][], Exponent[], CommitmentParams, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into PrivateAndPublicCommitmentProcessor._publicCommitments
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.PrivateAndPublicCommitmentProcessor(PrivateCommitment[], PublicCommitment[], Exponent[][], Exponent[], CommitmentParams, MultiExponentiation, int) may expose internal representation by storing an externally mutable object into PrivateAndPublicCommitmentProcessor._r
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.PublicCommitmentCalculatorProcess(PublicCommitment[], PrivateCommitment[]) may expose internal representation by storing an externally mutable object into PublicCommitmentCalculatorProcess._privateCommitments
EI2	new com.scytl.products.ov.mixnet.commons.concurrent.processor.PublicCommitmentCalculatorProcess(PublicCommitment[], PrivateCommitment[]) may expose internal representation by storing an externally mutable object into PublicCommitmentCalculatorProcess._publicCommitments
EI2	new com.scytl.products.ov.mixnet.commons.homomorphic.impl.GjosteenElGamal(ZpSubgroup, Exponent[]) may expose internal representation by storing an externally mutable object into GjosteenElGamal._privateKey
EI2	new com.scytl.products.ov.mixnet.commons.homomorphic.impl.GjosteenElGamalPlaintext(ZpGroupElement[]) may expose internal representation by storing an externally mutable object into GjosteenElGamalPlaintext._m
EI2	new com.scytl.products.ov.mixnet.commons.mathematical.tools.LUDecomposition(Exponent[][], BigInteger) may expose internal representation by storing an externally mutable object into LUDecomposition._lu
EI2	new com.scytl.products.ov.mixnet.commons.proofs.bg.commitments.CommitmentParams(ZpSubgroup, ZpGroupElement, ZpGroupElement[]) may expose internal representation by storing an externally mutable object into CommitmentParams._g

Performance Warnings

Code	Warning
SBSC	com.scytl.products.ov.mixnet.commons.io.ciphertext.CiphertextSerializer.serialize(Ciphertext, JsonGenerator, SerializerProvider) concatenates strings using + in a loop
SS	Unread field: com.scytl.products.ov.mixnet.commons.io.ElGamalEncryptedBallotEntryParser.PATTERN; should this field be static?
SS	Unread field: com.scytl.products.ov.mixnet.commons.tools.UniversalElGamalEncryptionParamsGenerator.CERTAINTY_LEVEL; should this field be static?

Dodgy code Warnings

Code	Warning
BC	instanceof will always return true for all non-null values in com.scytl.products.ov.mixnet.commons.concurrent.processor.LimMultiExpoConcurrentCalculatorProcess.getZpGroup(ZpSubgroup), since all com.scytl.cryptolib.mathematical.groups.impl.ZpSubgroup are instances of com.scytl.cryptolib.mathematical.groups.impl.ZpSubgroup
BC	instanceof will always return true for all non-null values in com.scytl.products.ov.mixnet.commons.proofs.bg.commitments.CommitmentParams.extractZpGroup(ZpSubgroup), since all com.scytl.cryptolib.mathematical.groups.impl.ZpSubgroup are instances of com.scytl.cryptolib.mathematical.groups.impl.ZpSubgroup
DLS	Dead store to reconstructedZpGroup in com.scytl.products.ov.mixnet.commons.io.CommitmentParamsReader.readCommitmentParamsFromStream(ZpSubgroup, InputStream)
RV	com.scytl.products.ov.mixnet.commons.configuration.file.FileBasedApplicationConfig.convertToStreamConfig(FileBasedApplicationConfig) discards result of readLine after checking if it is non-null

Details

BC_VACUOUS_INSTANCEOF: instanceof will always return true

This instanceof test will always return true (unless the value being tested is null). Although this is safe, make sure it isn't an indication of some misunderstanding or some other logic error. If you really want to test the value for being null, perhaps it would be clearer to do better to do a null test rather than an instanceof test.

DLS_DEAD_LOCAL_STORE: Dead store to local variable

This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error, because the value computed is never used.

Note that Sun's javac compiler often generates dead stores for final local variables. Because SpotBugs is a bytecode-based tool, there is no easy way to eliminate these false positives.

DM_DEFAULT_ENCODING: Reliance on default encoding

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

EI_EXPOSE_REP: May expose internal representation by returning reference to mutable object

Returning a reference to a mutable object value stored in one of the object's fields exposes the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Returning a new copy of the object is better approach in many situations.

EI_EXPOSE_REP2: May expose internal representation by incorporating reference to mutable object

This code stores a reference to an externally mutable object into the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Storing a copy of the object is better approach in many situations.

RV_DONT_JUST_NULL_CHECK_READLINE: Method discards result of readLine after checking if it is non-null

The value returned by readLine is discarded after checking to see if the return value is non-null. In almost all situations, if the result is non-null, you will want to use that non-null value. Calling readLine again will give you a different line.

RV_RETURN_VALUE_IGNORED_BAD_PRACTICE: Method ignores exceptional return value

This method returns a value that is not checked. The return value should be checked since it can indicate an unusual or unexpected function execution. For example, the `File.delete()` method returns false if the file could not be successfully deleted (rather than throwing an Exception). If you don't check the result, you won't notice if the method invocation signals unexpected behavior by returning an atypical return value.

SBSC_USE_STRINGBUFFER_CONCATENATION: Method concatenates strings using + in a loop

The method seems to be building a String using concatenation in a loop. In each iteration, the String is converted to a StringBuffer/StringBuilder, appended to, and converted back to a String. This can lead to a cost quadratic in the number of iterations, as the growing string is recopied in each iteration.

Better performance can be obtained by using a StringBuffer (or StringBuilder in Java 1.5) explicitly.

For example:

```
// This is bad
String s = "";
for (int i = 0; i < field.length; ++i) {
    s = s + field[i];
}

// This is better
StringBuffer buf = new StringBuffer();
for (int i = 0; i < field.length; ++i) {
    buf.append(field[i]);
}
String s = buf.toString();
```

SE_BAD_FIELD: Non-transient non-serializable instance field in serializable class

This Serializable class defines a non-primitive instance field which is neither transient, Serializable, or `java.lang.Object`, and does not appear to implement the `Externalizable` interface or the `readObject()` and `writeObject()` methods. Objects of this class will not be deserialized correctly if a non-Serializable object is stored in this field.

SS_SHOULD_BE_STATIC: Unread field: should this field be static?

This class contains an instance final field that is initialized to a compile-time static value. Consider making the field static.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/online-voting-logging/logging-core/build/classes/java/main

Metrics

202 lines of code analyzed, in 8 classes, in 5 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings	4	19.80
Total Warnings	**4**	**19.80**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Performance Warnings	4
Total	**4**

Warnings

Click on a warning row to see full context information.

Performance Warnings

Code	Warning
SS	Unread field: com.scytl.products.oscore.logging.core.layout.EscapingPatternLayout.bReplacement; should this field be static?
SS	Unread field: com.scytl.products.oscore.logging.core.layout.EscapingPatternLayout fReplacement; should this field be static?
SS	Unread field: com.scytl.products.oscore.logging.core.layout.EscapingPatternLayout nReplacement; should this field be static?
SS	Unread field: com.scytl.products.oscore.logging.core.layout.EscapingPatternLayout rReplacement; should this field be static?

Details

SS_SHOULD_BE_STATIC: Unread field: should this field be static?

This class contains an instance final field that is initialized to a compile-time static value. Consider making the field static.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/online-voting-logging/logging-api/build/classes/java/main

Metrics

151 lines of code analyzed, in 13 classes, in 5 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings		0.00
Total Warnings	**0**	**0.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Total	**0**

Warnings

Click on a warning row to see full context information.

Details

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/secure-logger/validator/build/classes/java/main

Metrics

1001 lines of code analyzed, in 23 classes, in 7 packages.

Metric	Total	Density*
High Priority Warnings	2	2.00
Medium Priority Warnings	1	1.00
Total Warnings	**3**	**3.00**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Bad practice Warnings	1
Internationalization Warnings	1
Performance Warnings	1
Total	**3**

Warnings

Click on a warning row to see full context information.

Bad practice Warnings

Code	Warning
Nm	The class name com.scytl.slogger.validator.Level shadows the simple name of the superclass org.apache.log4j.Level

Internationalization Warnings

Code	Warning
Dm	Found reliance on default encoding in com.scytl.slogger.util.LogValidator.main(String[]): new java.io.FileWriter(String)

Performance Warnings

Code	Warning
Bx	com.scytl.slogger.validator.LogFilePatternReceiver.initialize() invokes inefficient new Integer(int) constructor; use Integer.valueOf(int) instead

Details

DM_NUMBER_CTOR: Method invokes inefficient Number constructor; use static valueOf instead

Using `new Integer(int)` is guaranteed to always result in a new object whereas `Integer.valueOf(int)` allows caching of values to be done by the compiler, class library, or JVM. Using of cached values avoids object allocation and the code will be faster.

Values between -128 and 127 are guaranteed to have corresponding cached instances and using `valueOf` is approximately 3.5 times faster than using constructor. For values outside the constant range the performance of both styles is the same.

Unless the class must be compatible with JVMs predating Java 1.5, use either autoboxing or the `valueOf()` method when creating instances of `Long`, `Integer`, `Short`, `Character`, and `Byte`.

DM_DEFAULT_ENCODING: Reliance on default encoding

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

NM_SAME_SIMPLE_NAME_AS_SUPERCLASS: Class names shouldn't shadow simple name of superclass

This class has a simple name that is identical to that of its superclass, except that its superclass is in a different package (e.g., `alpha.Foo` extends `beta.Foo`). This can be exceptionally confusing, create lots of situations in which you have to look at import statements to resolve references and creates many opportunities to accidentally define methods that do not override methods in their superclasses.

SpotBugs Report

Project Information

Project:

SpotBugs version: 3.1.6

Code analyzed:

- NSWEC-iVote-source-code/secure-logger/logger/build/classes/java/main

Metrics

1658 lines of code analyzed, in 44 classes, in 11 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings	2	1.21
Total Warnings	**2**	**1.21**

(Defects per Thousand lines of non-commenting source statements)*

Contents

Summary

Warning Type	Number
Bad practice Warnings	1
Correctness Warnings	1
Total	**2**

Warnings

Click on a warning row to see full context information.

Bad practice Warnings

Code	Warning
Se	Class com.scytl.slogger.event.SecureLoggingEvent defines non-transient non-serializable instance field secureMessage

Correctness Warnings

Code	Warning
NP	Null passed for non-null parameter of new java.io.File(String) in com.scytl.slogger.SecureFileAppender.activateOptions()

Details

NP_NULL_PARAM_DEREF: Method call passes null for non-null parameter

This method call passes a null value for a non-null method parameter. Either the parameter is annotated as a parameter that should always be non-null, or analysis has shown that it will always be dereferenced.

SE_BAD_FIELD: Non-transient non-serializable instance field in serializable class

This Serializable class defines a non-primitive instance field which is neither transient, Serializable, or `java.lang.Object`, and does not appear to implement the `Externalizable` interface or the `readObject()` and `writeObject()` methods. Objects of this class will not be deserialized correctly if a non-Serializable object is stored in this field.