Analysis and Solutions for CAVS Testing Errors

Yuan Xu
atsec information security corporation
yuan@atsec.com

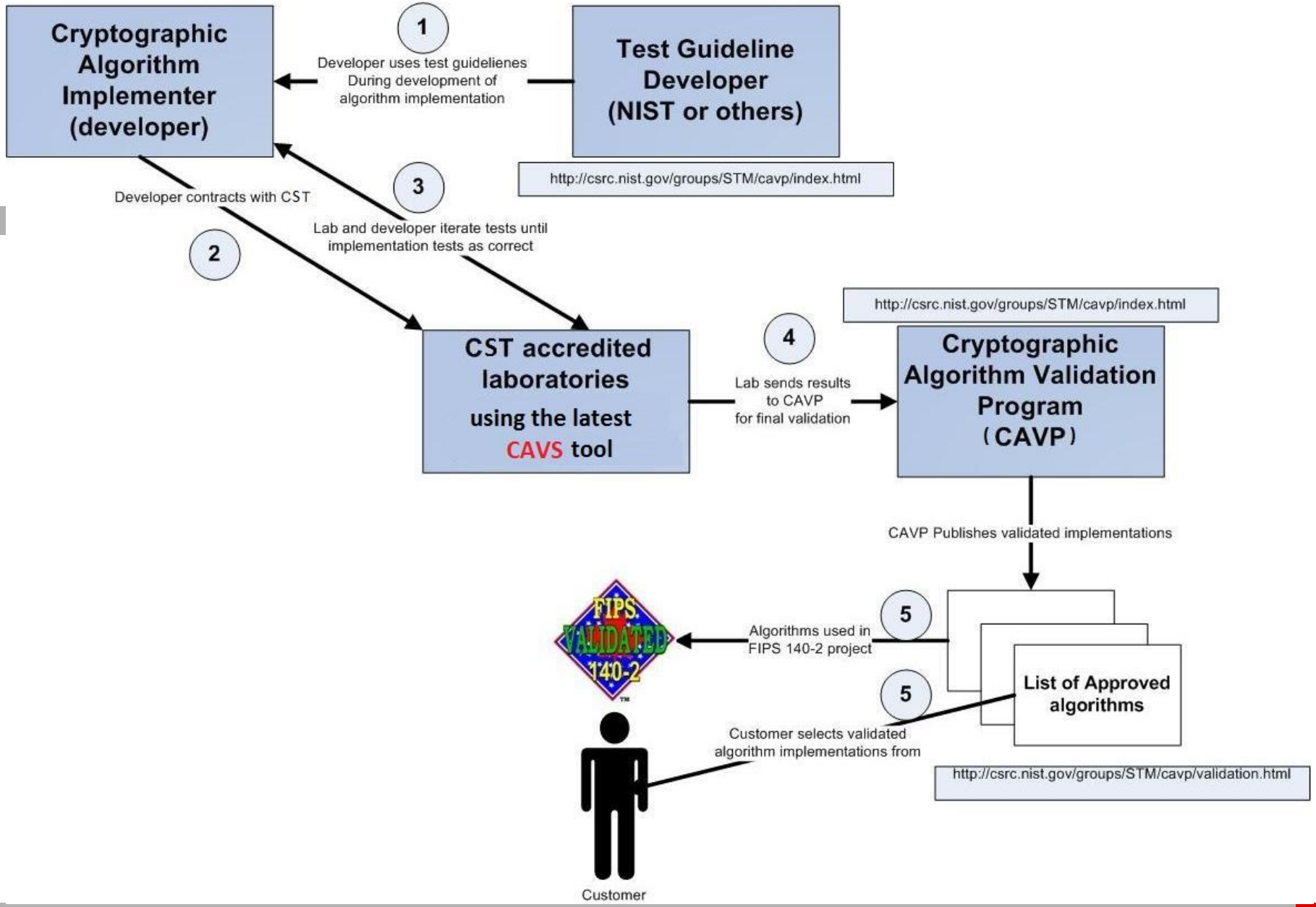
ICMC 2016, May 18-20, 2016, Ottawa, Canada

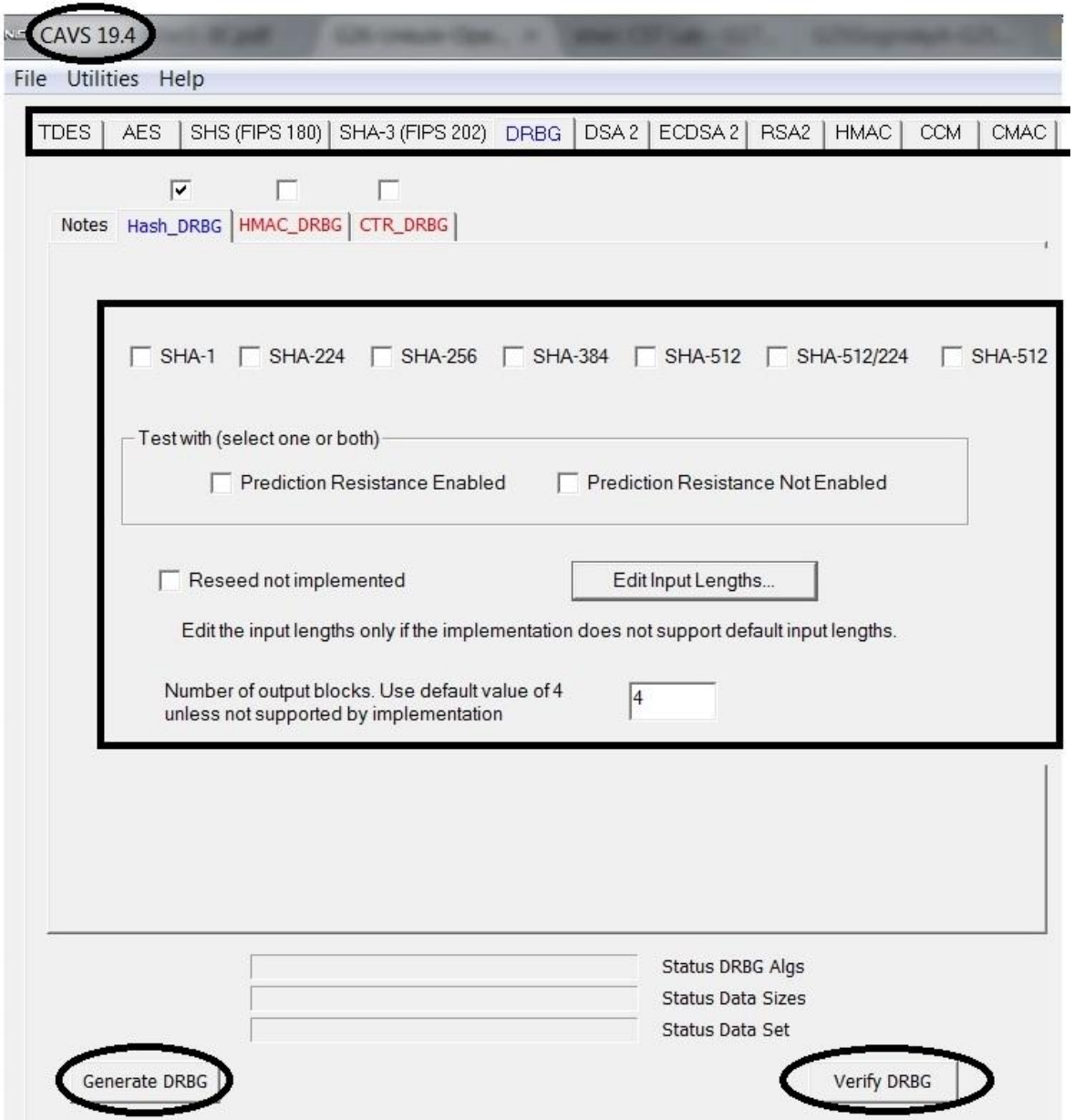
Overview

- Introduction to CAVS testing
- Types of CAVS testing errors
- Examples of analyzing minor errors
- Examples of analyzing major errors

Introduction to CAVS Testing

- What is CAVS (Cryptographic Algorithm Validation System)?
  - The testing tool for the CAVP to assure that the implementation of algorithms is correct
  - A prerequisite for both FIPS validation and CC evaluation under the NIAP scheme


Cryptographic Algorithm Implementer (developer)
1
Developer uses test guidelienes During development of algorithm implementation
Test Guideline Developer (NIST or others)
http://csrc.nist.gov/groups/STM/cavp/index.html
Developer contracts with CST
2
3
Lab and developer iterate tests until implementation tests as correct
CST accredited laboratories using the latest CAVS tool
4
Lab sends results to CAVP for final validation
http://csrc.nist.gov/groups/STM/cavp/index.html
Cryptographic Algorithm Validation Program (CAVP)
CAVP Publishes validated implementations
FIPS VALIDATED 140-2
5
Algorithms used in FIPS 140-2 project
List of Approved algorithms
5
Customer selects validated algorithm implementations from
http://csrc.nist.gov/groups/STM/cavp/validation.html
Customer


CAVS 19.4
File Utilities Help
TDES AES SHS (FIPS 180) SHA-3 (FIPS 202) DRBG DSA 2 ECDSA 2 RSA2 HMAC CCM CMAC
Notes Hash_DRBG HMAC_DRBG CTR_DRBG
SHA-1 SHA-224 SHA-256 SHA-384 SHA-512 SHA-512/224 SHA-512
Test with (select one or both)
Prediction Resistance Enabled
Prediction Resistance Not Enabled
Reseed not implemented
Edit Input Lengths...
Edit the input lengths only if the implementation does not support default input lengths.
Number of output blocks. Use default value of 4 unless not supported by implementation
4
Status DRBG Algs
Status Data Sizes
Status Data Set
Generate DRBG
Verify DRBG


PASS
FAIL

Introduction to CAVS Testing

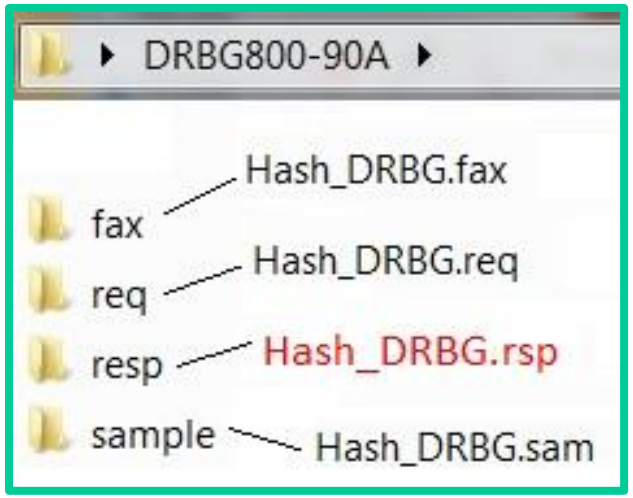
- How is an algorithm tested by CAVS?

The CST lab uses the latest CAVS tool and the information supplied by the vendor to generate input vector(s). The CAVS tool generates three types of file.

fax: answer

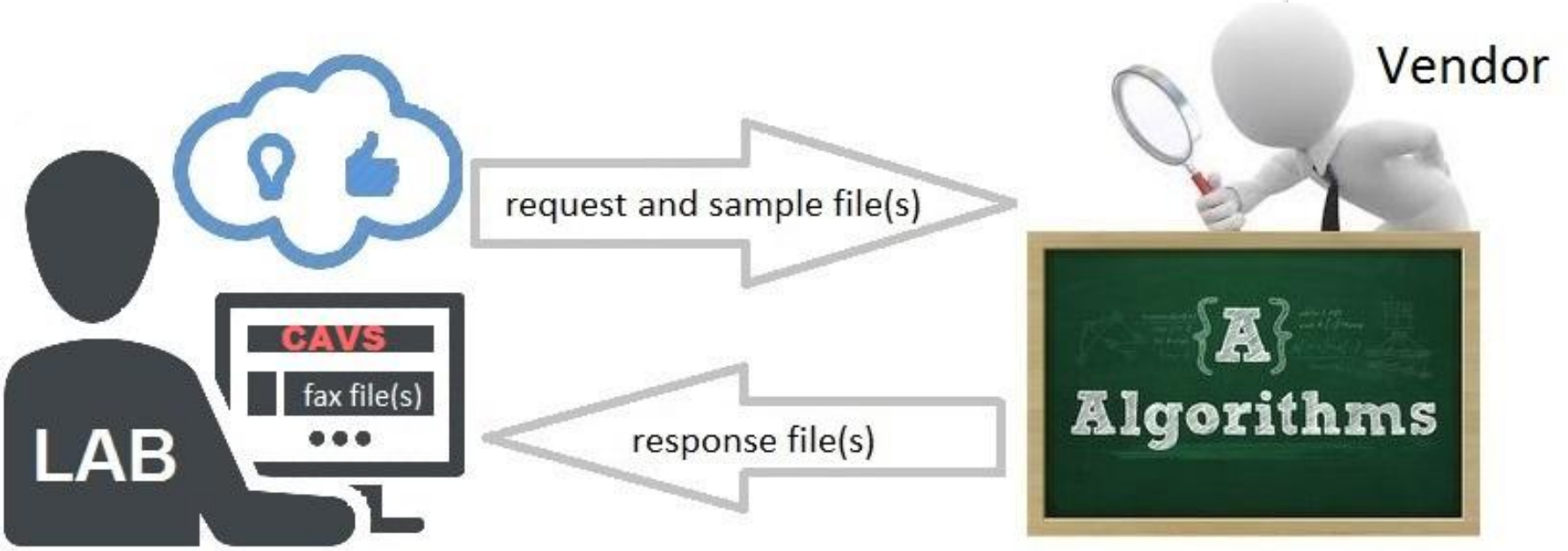
request: input test vector

sample: format required by the response file



The response file is generated by the vendor's algorithm implementation.

Introduction to CAVS Testing



- If any error occurs, the log file contains information pertaining to the error.
- The lab may help the vendor to analyze the errors.

Types of CAVS testing Errors

- Type 1 – Major Error (implementation flaws)
- Actions:
  - Review the algorithm implementation code against the specific algorithm standard, for example, the FIPS186-4 - Digital Signature Standard.
  - Review the test harness against the specific algorithm validation system, for example, the AESAVS - AES Algorithm Validation Suite.
  - Update the algorithm implementation code or test harness.

Types of CAVS Testing Errors

- Type 2 – Minor Error (formatting problem)
- Actions:
  - Make sure the request file are parsed correctly.
  - Compare the response file with the sample file and check the formatting of outputs.
  - Update the test harness.

Error Analysis and Solutions

- How to classify if it's a Major or Minor error?

  Review the log file!

  - **Minor error**: If the log file shows the test response cannot be read correctly or it implies that there is a formatting problem.
  - **Major error**: If the log file shows the test response cannot be verified and there is no formatting problem.

Examples of Minor Errors

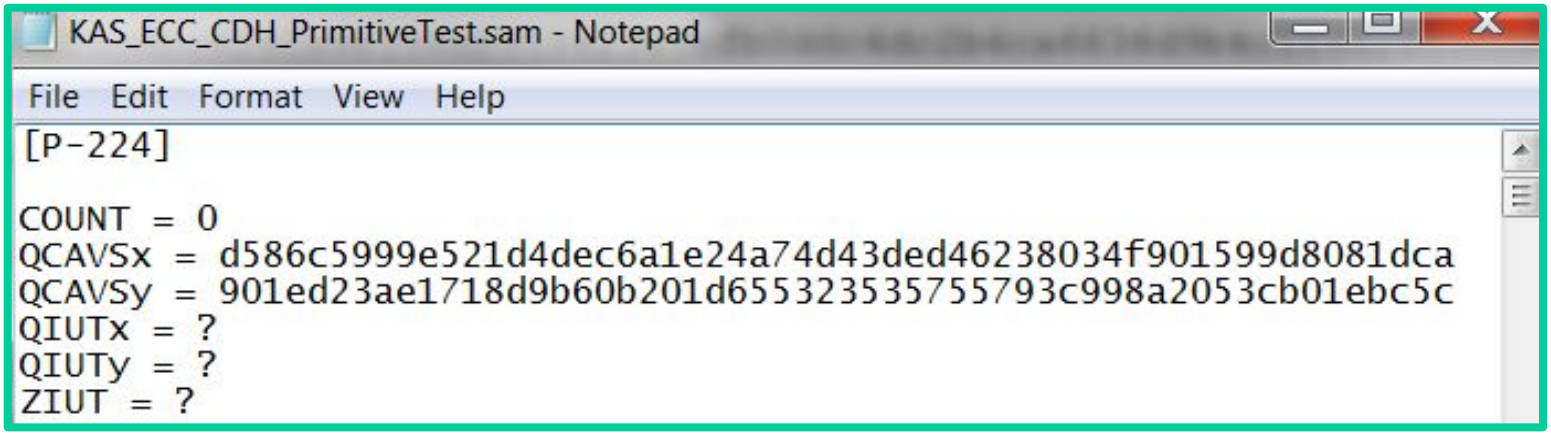
EXAMPLE

- E1: The log file shows XXX.rsp could not be opened.

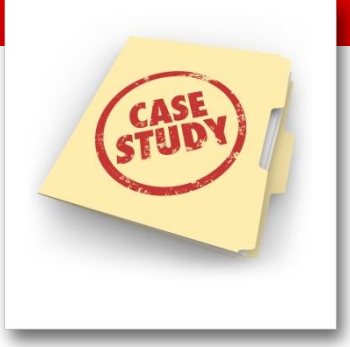
  Make sure the file extension is **.rsp**, NOT .resp!

- E2: The log file shows Error when reading a value.

  Make sure the response file contains all the values needed as specified in the sample file.



Examples of Minor Errors



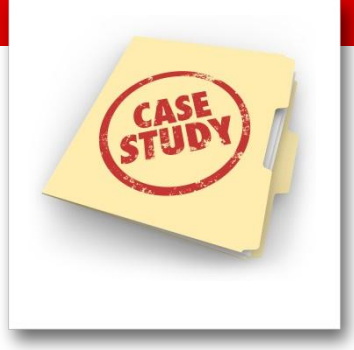
- E3: The log file shows the length of a value is wrong. It should reach the required length. If not, it could be fixed by padding the output value with leading zeros in test harness.

RSA2_LogFile.txt - Notepad

File Edit Format View Help

```
PrimeMethod = ProbRP (There are 2 tests for this method)

Test 1:  KeyGenRSA2_B3_3_Known Answer Test

        Mod 2048        Table for M-R Test C.3
        Verified 47 out of 47

        Mod 3072        Table for M-R Test C.3
        Verified 37 out of 37

Test 2:
KeyGenRSA2_B3_3_Test2
        Mod 2048        Table for M-R Test C.3
"d" is wrong length in rsp file
        is 2044, should be 2048

Can't read d from rsp file
```

Examples of Minor Errors



- E4: The log file shows the response cannot be verified, for example, when testing ECDSA KeyGen

ECDSA2_LogFile.rsp

```
#  Key Pair Validation:
    P-224:D is 000004658065538311966475285138569097922220474041058894839014230272584993
Qx is c51f97e6423a972e7751c1053bb30d96388c84ca6fa9f19821b57300
Qy is dc3e320b3a8e5b12423856a8ff905f10231f76422108e11ab49c18b9
Qx_me is fd87e75c5327ea387efa8da338b23ec1d38685736b2c880634a56ddc
Qy_me is 7291961b6f3f108d198e94c007db92382936dc2da84831a6fb64229b
   Q != dG
```

The *d* value should be output as a hex string instead of a decimal value!

Examples for Major Errors

- E1: The Monte Carlo Test (MCT) of AES ECB mode failed. The cipher texts in response file don't match those in fax file.

Monte Carlo Algorithm - ECB:

```
Key[0] = Key
PT[0] = PT
For i = 0 to 99
        Output Key[i]
        Output PT[0]
        For j = 0 to 999
                CT[j] = AES(Key[i], PT[j])
                PT[j+1] = CT[j]
        Output CT[j]
        ……
```

  - Analysis: The tester reviewed the test harness against section 6.4.1 of AESAVS and found that the iteration number was not set correctly.
  - Solution: Update the test harness to make sure that 1,000 iterations are looped before getting the ciphertext for each of the 100 rounds of tests.

Examples for Major Errors

- E2: The Monte Carlo Test (MCT) of TDES CBC mode failed.

According to [NIST SP800-20], TDES MCT consists of 400 groups of 10,000 iterations. Table 18 shows for $i \in [0, 399]$, $KEY1[i+1] = KEY1[i] \oplus C[9999]$. The tester downloaded the TDES MCT test vectors with intermediate values and found the unmatched result.

KEY1[0] ⊕ C[9999]= fee5791a94b9869b ⊕ f5b62d1079b2c1e9 = 0b53540aed0b4772, but the expected value is 0b52540bec0b4673.

- Analysis: The two values only differ on the last bit of each byte, and that is the odd parity bit!
- Solution: The test harness of TDES MCT needs to be updated. For each byte of a TDES key, the odd parity bit needs to be set, i.e. 53 = 0101 0011 -> 52 = 0101 0010.

Examples for Major Errors

- E3: The log file shows that ECDSA Public Key Validation (PKV) test failed.

PKV.rsp

```
1  [P-521]
2  Qx = 04dcf53fb04480e71974674db8ca84
3  Qy = 370201209871ef807ad5ea21c5cb07651
4  Result = P
```

PKV.fax

```
1  [P-521]
2  Qx = 04dcf53fb04480e71974674db8ca848d3e8
3  Qy = 370201209871ef807ad5ea21c5cb0765129
4  Result = F (1 - Q_x or Q_y out of range)
```

- Analysis: Section 6.3 of ECDSAVS states that the implementation determines whether or not the public key (supplied in PKV.req) passes all the conditions in section 5.2.2 of ANSI X9.62. There are 5 steps for PKV. By reviewing the module's source code of PKV, it is determined that the 4th step to verify *nQ=infinity* is skipped.
- Solution: The implementation code of ECDSA PKV needs to be updated to add the 4th step.

Examples for Major Errors

- E4: The log file shows that the RSA KeyGen test failed.

RSA2_LogFile.txt

```
1  #FIPS186_3 Key Generation RSA:
2  PrimeMethod = ProbRP (There is 1 test for
3  Test 1:
4  KeyGenRSA2_B3_3_Test2
5      Mod 2048    Table for M-R Test C.2
6  d values do not match
7  d from resp file = a46cf43b5d729fdfd8c9c97
8  d calculated = 03ffef445acfc1cef03a3a3dacb
```

KeyGen_RandomProbablyPrime3_3.rsp

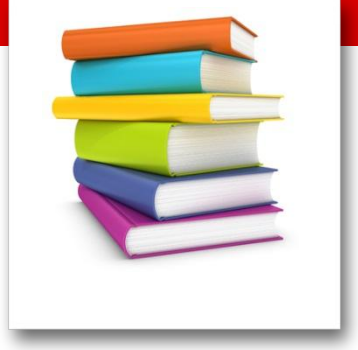
```
1  [mod = 2048]
2  [Table for M-R Test = C.2]
3
4  e = 00000000000000000000000000000000000000000000
5  p = d7e9cdff68afc0f6d0dd585819ab0cae389ab24e
6  q = ce83b4706d834e08d790ec6b7dd38b554560df67
7  n = ae2d3897276effba96183517578c5b8bd78487ec
8  d = a46cf43b5d729fdfd8c9c974dba518d70f0181ff
```

- Analysis: As mentioned in RSA2VS, the key generation must be compliant to FIPS186-4. The tester reviewed the source code, and found that the generation of d doesn't satisfy $d = e^{-1} \bmod (\mathrm{LCM}(p-1, q-1))$.
- Solution: The implementation code of RSA KeyGen must be updated accordingly.

Conclusions

- Check the log file.
- Make sure the test vectors are parsed correctly.
- Compare the response file with the sample file and check if the formatting is correct.
- Review the algorithm implementation to see if it is compliant to the algorithm standard.
- Check the testing harness to make sure it follows the algorithm validation system.

References



- http://csrc.nist.gov/groups/STM/cavp/
- http://csrc.nist.gov/groups/STM/cavp/documents/CAVPMM.pdf
- http://csrc.nist.gov/groups/STM/cavp/documents/CAVPFAQ.pdf

Thank You