Combinatorial Coverage Measurement of Test Vectors used in Cryptographic Algorithm Validation

Dimitris Simos¹, Stavros Mekesis¹, D. Richard Kuhn², Raghu N. Kacker²
SBA Research, Vienna, Austria;
dsimos, smekesis@sba-research.org¹
National Institute of Standards & Technology, Gaithersburg, MD, USA;
kuhn, raghu.kacker@nist.gov²
Overview

• Measuring the combinatorial coverage of test vectors used in the Cryptographic Algorithm Validation Program (CAVP).

• Using differential testing and golden model testing in tandem to identify incorrect behavior in the context of multiple AES implementations tested together.
The Cryptographic Algorithm Validation Program (CAVP) encompasses validation testing for FIPS-approved and NIST recommended cryptographic algorithms.

It was established by NIST and the Communications Security Establishment Canada (CSEC) in July 1995.

It currently validates implementations of the following cryptographic algorithms: AES, TDES, Skipjack, DSA, ECDSA, RSA, SHA, RNG, DRBG, KAS, CMAC, CCM, HMAC, GCM, and GMAC.
AES Validation Testing

• NIST developed a set of tests, referred to as the AES Validation test suite, to ensure the quality and correctness of AES implementations.

• The AES Algorithm Validation System (AESAVS) specifies validation testing requirements for the ECB, CBC, OFB, CFB, and CTR modes for the AES algorithm.

• If a vendor’s AES implementation successfully passes all the tests, NIST issues an AES Validation certificate to the vendor.
Modeling of Test Vectors

• We map existing test vectors into discrete parameters and values that can be analyzed and evaluated using coverage measurement tools.

• Assume that we perform a cryptographic operation (e.g. encryption, decryption) using cryptographic keys of either 128, 192, or 256 bits.

• The Input Parameter Model (IPM) for our combinatorial coverage measurement problem consists of 128, 192, or 256 binary parameters corresponding to cryptographic keys.
Combinatorial Coverage of CAVP Tests

• We use the Combinatorial Coverage Measurement (CCM) tool to measure the combinatorial coverage of the AES KAT Vectors, AES MCT Sample Vectors, AES MCT Intermediate Values, and AES MMT Sample Vectors.

• Our model contains either 128, 192, or 256 binary parameters.

• The combinatorial coverage refers to 2-way, 3-way, and 4-way coverage.
• A test set consists of either test vectors which are properly formatted in response (.rsp) files or files with intermediate results (.txt) which are supplied to help with debugging.

• Each file contains cryptographic keys grouped with other cryptographic information (e.g. plaintexts, initialization vectors, ciphertexsts).

• The filename contains information about the test set and the key length (e.g. 128, 192, 256 bits).
Measurement Results

• We measured the total 2-way through 4-way coverage considering

  • all keys for both encryption of plaintext and decryption of ciphertext,

  • only encryption keys, and

  • only decryption keys.
Implication for Testing

• In some cases, the AES test vectors do not achieve a full 2-way to 4-way combinatorial coverage.

• If 90% - 100% of the relevant state space has been covered, then presumably the risk is small, but if coverage is much smaller, then the risk may be substantial.

• Using the CCM tool we generated all missing combinations to achieve full coverage for t=2.

• It entails making certain that the necessary (missing) combinations to achieve a full 2-way coverage are meaningful to vendors.
Case Study

• We use differential testing and golden model testing in tandem to identify incorrect behavior in the context of multiple AES implementations tested together.

• We test the following AES implementations on macOS 10.12.5 to determine whether they comply with the specifications and requirements in the standard:

  • OpenSSL 1.0.2k (26 Jan 2017),
  • LibreSSL 2.5.3,
  • Crypto++ 5.6.5,
  • PyCrypto 2.6.1.
Differential Testing

• A Python script uses CCM to generate the missing combinations (AES keys) for a test suite and constructs a list of tuples of the following form from the test suite:

  [(missing key from test suite, IV in test suite, plaintext or ciphertext in test suite)].

• All clients are executed against each tuple to generate a .csv file in which each line is of the following form:

  message, key, iv, openssl, libressl, pycrypto, cryptopp, same_output.
• **message** stands for either a plaintext or a ciphertext in the corresponding test suite.

• **key** stands for a missing key.

• **iv** stands for an Initialization Vector in the test suite.

• Each of openssl, libressl, pycrypto, cryptopp stands for the output (ciphertext or plaintext) of the corresponding library.

• **same_output** stands for a boolean variable which is true if and only if all libraries generate the same output.
Golden Model Testing

• We use the Cryptographic Algorithm Validation System (CAVS) as a test oracle.

• A script constructs a list of tuples of the following form from a CAVP test suite:

  [(key in test suite, IV in test suite, plaintext or ciphertext in test suite)].

• The script executes all clients against each tuple and generates a .csv file in which each line is of the following form:

  message, key, iv, openssl, libressl, pycrypto, cryptopp.
- **message** stands for either a plaintext or a ciphertext in the corresponding test suite.

- **key** stands for a key in the test suite.

- **iv** stands for an Initialization Vector in the test suite.

- Each of openssl, libressl, pycrypto, cryptopp stands for a boolean variable which is true if and only if the library generates the output (ciphertext or plaintext) in the test suite.
Testing Results

• We conducted differential testing using more than 2,000,000 tuples, finding no discrepancies between the AES implementations.

• Golden model testing using more than 25,000 tuples did not reveal any problems.
Conclusions

• We measured the combinatorial coverage of test vectors provided by the NIST Cryptographic Algorithm Validation Program (CAVP).

• Input models were defined and test vectors measured and analyzed for 2-way, 3-way, and 4-way combinatorial coverage.

• The results of our measurement show that some test vectors do not achieve a full 2-way to 4-way combinatorial coverage, so we generated the missing combinations for these vectors and extended the test suites to achieve a full 2-way coverage.
• We also conducted differential testing on popular AES implementations, such as OpenSSL (v. 1.0.2k 26 Jan 2017), LibreSSL (v. 2.5.3), Crypto++ (v. 5.6.5), PyCrypto (v. 2.6.1), using the extended test suites. Our differential testing of AES implementations on these test suites showed no discrepancies between the implementations.

• Finally, we use the NIST Cryptographic Algorithm Validation System (CAVS) as a golden system against which the AES implementations are tested. Our testing did not reveal any problems.
Questions - Comments

Thank you for your attention!