Knowledge Extraction for Cryptographic Algorithm Validation Test Vectors by Means of Combinatorial Coverage Measurement

Dimitris E. Simos[1], Bernhard Garn[1], Ludwig Kampel[1],
D. Richard Kuhn[2], and Raghu N. Kacker[2]

[1] SBA Research, Vienna A-1040, Austria
{dsimos,bgarn,lkampel}@sba-research.org
[2] National Institute of Standards & Technology, Gaithersburg, MD, USA
{kuhn,raghu.kacker}@nist.gov

Abstract. We present a combinatorial coverage measurement analysis for test vectors provided by the NIST Cryptographic Algorithm Validation Program (CAVP), and in particular for test vectors targeting the AES block ciphers for different key lengths and cryptographic modes of operation. These test vectors are measured and analyzed using a combinatorial approach, which was made feasible via developing the necessary input models. The combinatorial model for the test data in combination with the coverage measurement allows to extract information about the structure of the test vectors. Our analysis shows that some test vectors do not achieve full combinatorial coverage. It is further discussed how this retrieved knowledge could be used as a means of test quality analysis, by incorporating residual risk estimation techniques based on combinatorial methods, in order to assist the overall validation testing procedure.

Keywords: Combinatorial Measurement, Cryptographic Applications, Data Analysis, Knowledge Extraction

1 Introduction

The implementation of cryptographic algorithms is a demanding task, involving various fields of computer science and software engineering. As such the testing of cryptographic applications is a complex task, while being of special importance, as the requirements/needs/standards for security and privacy grow in modern information society. Thorough testing of mission critical systems – such as medical, transportation or cryptographic systems – is of vital and crucial importance as recent studies have shown [14], [23].

The *Cryptographic Algorithm Validation Program* (CAVP) [19], [18] by the *National Institute of Standards and Technology* (NIST) provides validation testing of FIPS-approved and NIST-recommended cryptographic algorithms and their individual components. Cryptographic algorithm validation is a prerequisite of *cryptographic module validation*, which is the subject of the *Cryptographic Module Validation Program* (CMVP) [20] established at NIST in 1995.

The CMVP is a joint effort between NIST and the Canadian Centre for Cyber Security, a branch of the Communications Security Establishment. FIPS 140-2 [17] precludes the use of unvalidated cryptography for the cryptographic protection of sensitive or valuable data within Federal systems in USA.

As of this writing, the CAVP tests block ciphers including the *Advanced Encryption Standard* (AES) [15], among others. In *The Advanced Encryption Standard Algorithm Validation Suite* (AESAVS) [1] the testing requirements for different modes for the AES algorithm are specified.

Recently, the Secretary of Commerce approved Federal Information Processing Standards Publication (FIPS) 140-3, *Security Requirements for Cryptographic Modules* [22], which supersedes FIPS 140-2 and will come effective on September 22, 2019. FIPS 140-3 aligns with ISO/IEC 19790:2012(E) [5] and includes modifications of the Annexes that are allowed to the CMVP, as a validation authority. As of this writing, the corresponding documents have not been released yet[3]. In [1] it is noted that the testing performed within the AESAVS uses statistical sampling meaning that only a small number of the possible cases are tested. Nevertheless, AESAVS states to provide testing of an *Implementation Under Test* (IUT) to determine the correctness of the algorithm implementation. In a branch of software testing called combinatorial testing (CT) [11] combinatorial methods have been used to analyze *test sets* in term of *combinatorial coverage*.

In this work, we analyze the test data used in the AESAVS in terms of combinatorial coverage. To this end, we transformed the data into an appropriate combinatorial model which facilitated the combinatorial analysis of the test vectors. The combinatorial measurement quantifies the parameter interactions executed during testing and in doing so provides a structural analysis of the test data. Within a software testing context, extracting the knowledge about potentially left out combinations has been used to estimate the residual risk that remains after testing [10]. Available tools not only can compute these measurements, but also have the functionality to present the results in different ways which are easily intelligible for the human eye.

Contribution. In this paper, we perform a study of the combinatorial coverage of various test vectors used in the AES algorithm validation suite. A featured model extraction made feasible via a transformation of the test vectors into test sets which are then used as a basis for the analysis provided by combinatorial coverage measurement tools. We used visualization techniques that arise from the combinatorial coverage measurement results to interpret the extracted knowledge as recommendation for future testing endeavors.

The paper is structured as follows. In Section 2 we give some preliminaries. We present the derived combinatorial model for our analysis in Section 3 and present our results in Section 4. We discuss implications of these findings in Section 5 and conclude the paper in Section 6.

[3] According to [21], NIST plans to release drafts for public comment in mid- 2019 and final publication of those documents will occur by September 22, 2019 .

2 Background Information

In this section, we provide some necessary preliminaries that will be used throughout the paper. We summarize important properties about AES and how its testing is specified in AESAVS in Section 2.1 and introduce CT including employed combinatorial concepts in Section 2.2.

2.1 AES and AESAVS

AES. The AES algorithm is a symmetric block cipher that can encrypt and decrypt data. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. NIST has approved several modes of the approved block ciphers in a series of special publications [16].

AESAVS. The Advanced Encryption Standard algorithm validation suite is designed to test the following modes of operation [16]:

- ECB, which stands for electronic codebook mode
- CBC, which stands for cipher block chaining mode
- OFB, which stands for output feedback mode
- CFB, which stands for cipher feedback mode with the following variants:
  - CFB1 (CFB where the length of the data segment is 1 bit, s=1)
  - CFB8 (CFB where the length of the data segment is 8 bits, s=8)
  - CFB128 (CFB where the length of the data segment is 128 bits, s=128)
- Counter (Counter mode is tested by selecting the ECB mode)

Note that it is not necessary for validation for every mode implemented to support the same key sizes and ciphering directions [1]. To initiate a validation process of the AESAVS, a vendor submits an application to an accredited laboratory requesting the validation of their implementation. The AESAVS is designed for testing of an IUT at locations remote to the AESAVS using communications via `REQUEST` and `RESPONSE` files. The test data is provided to an IUT in `REQUEST` files. The IUT processes this data and creates a corresponding `RESPONSE` file, which in turn will be verified.

AESAVS specifies three categories of tests: the *Known Answer Test* (KAT), the *Multi-block Message Test* (MMT), and the *Monte Carlo Test* (MCT). The KAT category is further split into four types: GFSbox, KeySbox, Variable Key, Variable Text. The MMT is designed to test the ability of the implementation to process multi-block messages, which may require chaining of information from one block to the next. For each supported mode, ten messages are supplied with lengths of $i \cdot \text{blocklength}$, for $1 \leq i \leq 10$. Each MCT ciphers 100 pseudorandom texts, where these texts are generated using an algorithm depending on the mode of operation being tested.

2.2 CT and CCM

CT [11] is an efficient black-box software testing methodology for more effective software testing at lower cost. It is based on an *input parameter model* (IPM) of the *system under test* (SUT) that models the input space, by identifying finitely many variables that can take finitely many values [4]. The defining property of CT is the coverage of all *t-way interactions* of parameter-value assignments for any combination of t parameters, for a specific value of t. Informally, a t-way interaction can be described as a parameter value assignment for exactly t parameters. The key insight underlying the empirically observed effectiveness of CT results from a series of studies by NIST [6–9, 24, 2, 3, 26]. NIST research showed that most software bugs and failures are caused by one or two parameter interactions, with progressively fewer by three or more. These findings have important implications for software testing because, it means that testing these few parameter-value combinations can provide strong assurances. Based upon that, a hypothesis has been formulated – which is referred to as the *interaction rule* – stating that most failures are induced by single factor faults or by the joint combinatorial effect (interaction) of two factors, with progressively fewer failures induced by interactions between three or more factors [11].

CT methods can also be employed on top of a existing *legacy test sets*, where an existing test set is used as a basis and analyzed in terms of combinatorial coverage. Subsequently, should higher and/or complete t-way coverage be desired than exhibited in the legacy test set, it is possible to create additional tests specifically covering those missing interactions. The union of all test cases coming from the legacy test set and the newly created ones then achieves the desired coverage properties. This approach is an alternative to creating combinatorial test sets newly from scratch.

Measuring the achieved level of combinatorial coverage can help in estimating the degree of risk that remains after testing; meaning that if a high level of coverage has been achieved (e.g., more than 90%), then presumably the risk is small, but if the coverage is much lower, then the risk may be substantial [10].

To address the need for such measurements, NIST has developed suitable methods and tools to quantify the achieved combinatorial overage of test sets [10]. We briefly describe combinatorial coverage by means of an example and refer the reader to [11] for further information. Consider given a system under test (SUT) that is modelled by five binary parameters `A,B,C,D,E` that can take the values 0 or 1. A 3-way interaction for this SUT is specified by a combination of three of the five variables, together with a specification of a value for each variable, e.g. $(\texttt{A} = 0, \texttt{B} = 1, \texttt{E} = 1)$ is one 3-way interaction. In total, for such an SUT, there are $2^3 \cdot \binom{5}{3} = 80$ different 3-way interactions. Given the following four test vectors:

```
        A     B     C     D     E
test 1: 1     1     1     1     0
test 2: 1     0     1     0     1
test 3: 0     0     0     1     1
test 4: 0     1     1     0     1
```

we see that the 3-way interaction ($\texttt{A} = 0, \texttt{B} = 1, \texttt{E} = 1$) is *covered* by `test 4`, i.e. the parameters `A,B,E` take the values `0,1,1` in this test vector. From the overall 80 3-way interactions for this SUT, the four test vectors cover 39 different 3 way interaction, in other words, the the combinatorial coverage measurement of these vectors yields a 3-way coverage (also called *total* 3*-way coverage* in [25]) of 48.75%. To summarize, to perform combinatorial coverage measurement, one needs a test set together with an IPM against which we can measure the t-way coverage of the test set. In this paper, we consider the terms SUT and IUT interchangeably.

3 Modelling and Measuring Combinatorial Coverage of AESAVS Test Data

Our analysis concerns, for given test set file of the AESAVS, the achieved combinatorial coverage of the binary-transformed extracted hex-values of the given keys in the individual test vectors. We start with an example for the data extraction, before we detail how complete files containing test data are transformed and analyzed.

The test data for provided configuration (category of test, mode of operation and key size) are provided in `REQUEST` files. The `RESPONSE` files contain the same data as the `REQUEST` files with the addition of the ciphertext for encryption (or plaintext for decryption). The generic structure of a single test vector in a `RESPONSE` file is as follows:

- an AES key of length 128, 192 or 256 bits, denoted by `KEY`, which is to be used for encryption or decryption. The mode of operation is further encoded into the filename of the test data;
- an initialization vector (if applicable to the mode of operation), denoted by `IV`;
- a sample plaintext, denoted by `PLAINTEXT`;
- the corresponding ciphertext, denoted by `CIPHERTEXT`;
- where the order of plaintext/ciphertext or ciphertext/plaintext indicates the ciphering direction.

To make our approach more tangible, consider the test data provided in the file `CBCMCT192.rsp`, which specifies test vectors for CBC mode of operation with a key size of 192 bits for the category of MCT:

```
COUNT = 51
KEY = 3461389779e6debf3e58d02175a33cd46663812b73b66082
IV = 88687bf1375300b8412cf10e35f6a0b1
PLAINTEXT = 03c1f719854c00e5a16c302e25621807
CIPHERTEXT = cf5d505c14e1e272634b4ad58b6ef3d9
```

The `COUNT` variable simply indicates the ordinal number of the test vector in this file.

For our analysis of the test data provided by the CAVP, we focused on the combinatorial measurement of the keys used for testing. Hence, we extract the hexadecimal value that instantiates the key used in the AES implementation. This value is translated to a binary vector of length 128, 192 or 256, depending on the chosen key size.

A test set consists of test vectors for both encryption and decryption. In our analysis, we *aggregated* the binary vectors in two different files depending on their origin, e.g. one for encryption and one for decryption. For each of these two resulting sets of test vectors, we carry out a combinatorial analysis in two steps:

1. Extraction of an IPM,
2. Combinatorial coverage measurement based on this IPM.

In the first step we determine for each parameter, that models the key, the set of values it takes over the course of the whole test set being executed. Thus, we extract an IPM for the AES key, from the test vectors. These extracted models contain either 128, 192, or 256 parameters. Depending on the considered test set, these parameters are unary or binary. In the second step, we measure the combinatorial 2-, 3- and 4-way coverage, of the test vectors against the IPM obtained in the first step. In our study we used the Combinatorial Coverage Measurement Tool (CCMtool) [13], developed by NIST and the Centro Nacional de Metrologia of Mexico, for both of the just described steps. Other combinatorial coverage measurement tools include the CAmetrics tool [12] which provides for additional visualization and combinatorial metrics.

We make this process more explicit by means of the following example where we consider again the `CBCVarKey192` AES validation test set. This set contains 192 test vectors for testing encryption, from which we extract the values of the keys and transform them to binary vectors, which constitutes a test set of 192 binary vectors of length 192. From these vectors we extract an IPM consisting of 192 parameters. In this specific case, the first parameter is unary, only taking the value 1 and the remaining 191 parameters are binary, taking the values 0 or 1. Finally we measure the combinatorial coverage of the 192 test vectors with regard to this IPM. The test set covers 54817 out of 72962 2-way interactions and 4626975 out of 9217660 3-way interactions, i.e. it achieves 75.15% 2-way coverage and 50.2% 3-way coverage.

4 Measurement results

For the AES KAT Vectors, AES MCT Sample Vectors and AES MMT Sample Vectors, we measured the total 2-way through 4-way coverage, separately considering the keys for encryption of plaintext and decryption of ciphertext.

4.1 AES KAT

The vectors extracted from the AES KAT test sets are the same for the different modes (ECB, CBC, OFB, CFB1, CFB8, CFB128) when considering AES ver-

sions of the same size. Further, the keys for encryption and decryption are the same. Thus, we do not specify the mode when we refer to a set of test vectors, e.g. {Mode}GFSbox128 refers to CBCGFSbox128 as well as CFB1GFSbox128 and further do not distinguish between encryption and decryption. The size of each AES test set can be seen in Table 1, below.

	128	192	256
{Mode}GFSbox	7	6	5
{Mode}KeySbox	21	24	16
{Mode}VarKey	128	192	256
{Mode}VarTxt	128	128	128

Table 1: AES KAT test set sizes (for encryption or decryption).

Now, from the AES test sets we extracted the following IPMs:

- IPM({Mode}GFSbox): 1^{128} (all unary)
- IPM({Mode}KeySbox): $1^1, 2^{127}$ (first parameter unary, others binary)
- IPM({Mode}VarKey): 2^{128} (all binary)
- IPM({Mode}VarTxt): 1^{128} (all unary)

The results of our coverage measurement are depicted in Figure 1 to give an comprehensive overview. Moreover, in Table 2 we detail the results of the 3-way coverage measurement for the different test vectors of length 128.

	extracted IPM	# tuples	# tuples covered	coverage %
IPM({Mode}GFSbox128)	1^{128}	341376	341376	100 %
IPM({Mode}KeySbox128)	2^{128}	2731008	2575694	94.3 %
IPM({Mode}VarKey128)	$1^1, 2^{127}$	2699004	1357503	50.3 %
IPM({Mode}VarTxt128)	1^{128}	341376	341376	100 %

Table 2: 3-way coverage of vectors for 128 length against extracted IPM.

The results of the coverage measurement visualized in Figure 1 need to be interpreted carefully. For the case of {Mode}VarTxt and {Mode}GFSbox, the coverage is 100% because the IPM consists only of unary parameters. The vectors of the test sets where the extracted IPMs are not trivial, achieve lower t-way coverage.

4.2 AES MCT

The vectors extracted from the MCT test sets contained 200 vectors for each mode (ECB, CBC, OFB, CFB1, CFB8, CFB128) which are split into two test sets for encryption and decryption as previously. Again, we extract the values

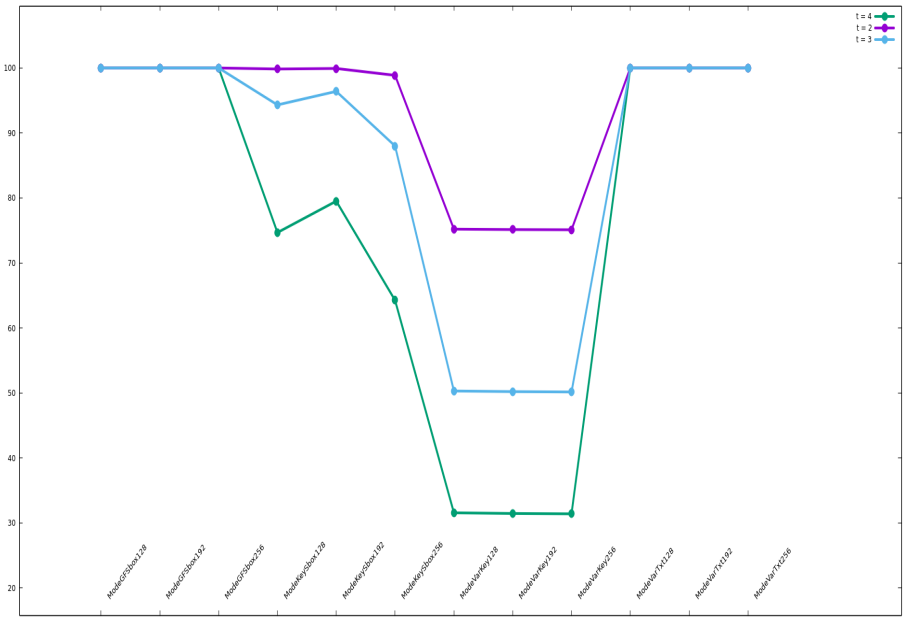

Fig. 1: 2-way, 3-way and 4-way coverage of AES KAT test sets.

for the keys from these vectors. For different modes the keys are instantiated differently and also the keys in the test vectors for encryption differ from the keys in the test vectors for decryption. For all modes and both test sets - for encryption and decryption - the IPMs extracted from the sets of keys consist of only binary parameters. Figures 2 and 3 show the results of our coverage measurements for 2-way, 3-way, and 4-way coverage. On the x-axes we denote the various AES modes with and the length and on the y-axes the percentage of t-way coverage. The figures show that for both, encryption and decryption, the AES keys achieve full 2-way coverage and almost full 3-way coverage (the lowest percentage across all lengths and modes being 99.9994% for encryption, and 99.9996% for decryption). The keys also have good 4-way coverage, staying above 99.80%, except for the case of `CFB128MCT128` achieving 99.77% 4-way coverage.

4.3 AES MMT

The vectors extracted from the MMT test sets contain 20 vectors, where again for each mode (ECB, CBC, OFB, CFB1, CFB8, CFB128) the test vectors are split in two sets for encryption and description. As before, for different modes the keys are instantiated differently and the keys for encryption differ from the keys for decryption. When extracting the IPMs from these test sets, we retrieve IPMs containing mostly binary parameters, but some IPMs extracted from sets of test

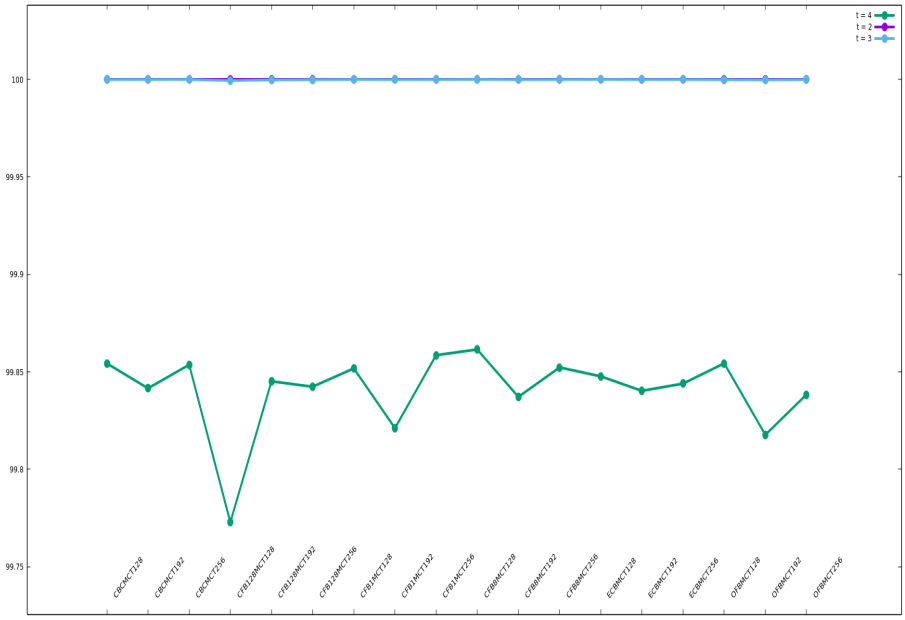

Fig. 2: 2-way, 3-way and 4-way coverage of AES MCT test sets for encryption.

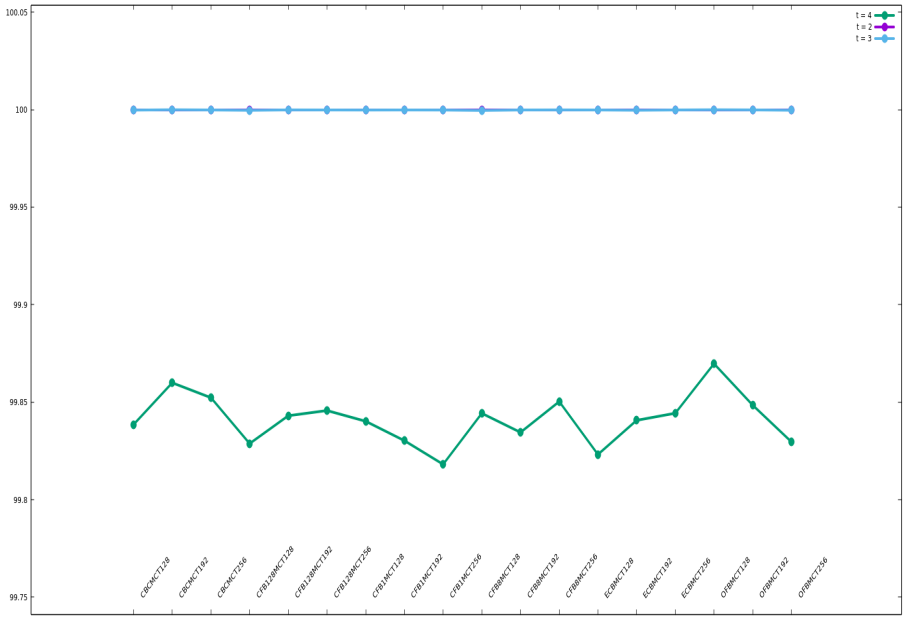

Fig. 3: 2-way, 3-way and 4-way coverage of AES MCT test sets for decryption.

vectors also contain unary parameters. To be more specific, from the encryption test sets, the IPMs extracted from the vectors for the modes `CFB1MMT192`, `ECBMMT256`, `CBCMMT128`, `CBCMMT256` and `CFB128MMT192` contain one unary parameter, while the remaining ones are binary; and for decryption the IPMs extracted from `CFB1MMT192`, `ECBMMT128`, `CBCMMT128`, `CFB128MMT192` contain one unary parameter and the one from `CFB8MMT128` contains two unary parameters, while the remaining parameters in all IPMs are binary.

Figures 4 and 5 depict the results of our t-way coverage measurements for $t \in \{2, 3, 4\}$, showing that the MMT test sets achieve high 2-way coverage above 90%, but only medium to small 3-way and 4-way coverage below 80% respectively 50%.

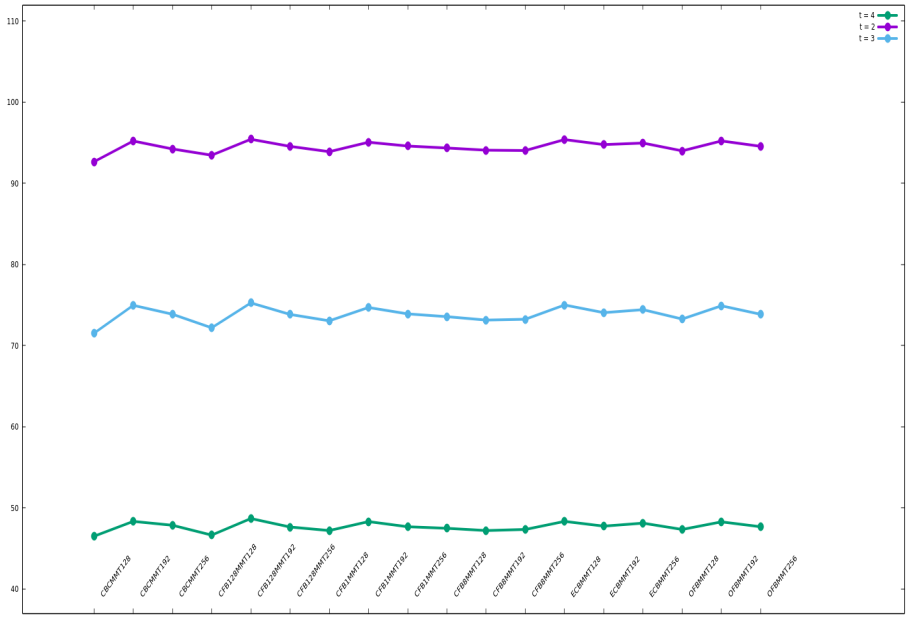

Fig. 4: 2-way, 3-way and 4-way coverage of AES MMT test sets for encryption.

5 Discussion related to Testing

The combinatorial coverage measurement analysis in the previous section shows that some of the extracted and transformed keys from the AESAVS test sets do not exhibit full t-way combinatorial coverage for some values of t. This finding has some implications for the currently specified testing requirements in AESAVS.

First, we already pointed out in the introduction that in the AESAVS document ([1]), it is noted that the testing performed within the AESAVS uses

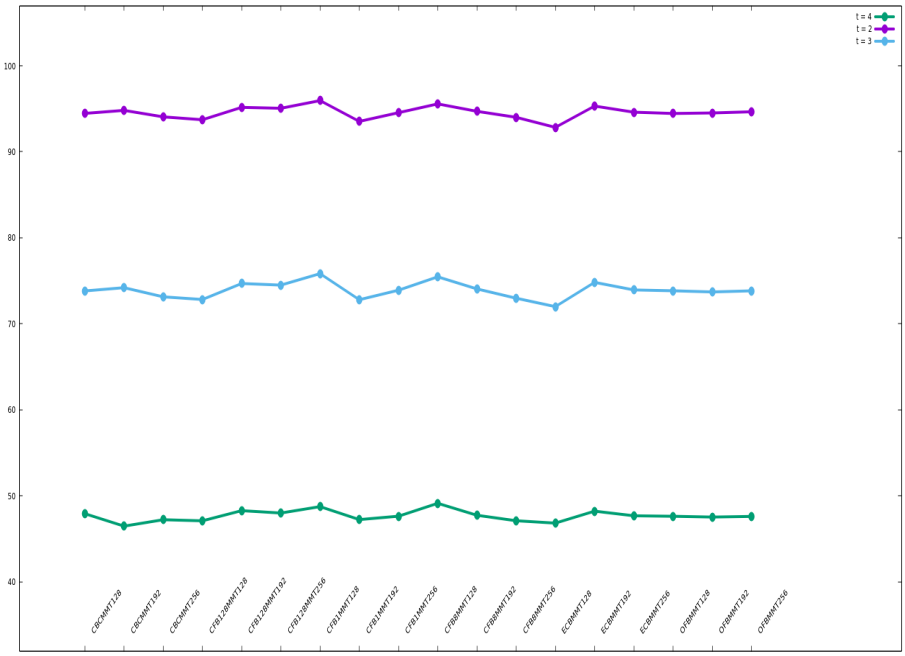

Fig. 5: 2-way, 3-way and 4-way coverage of AES MMT test sets for decryption.

statistical sampling to generate the test data. With our measurement approach we are now able to quantify the test sets sampling properties in terms of the achieved combinatorial t-way coverage.

Second, some works in the software testing literature have linked achieved combinatorial t-way coverage to the residual risk that remains after testing [10], [13]. An investigation whether similar conclusions could be drawn for validation testing purposes could be of interest.

Third, the presented case study here could be extended to also take into account not only the *key space*, but also simultaneously the *key space*, *IV space* and *ciphering direction space* (i.e., plain- or ciphertext space).

6 Conclusion

The Cryptographic Algorithm Validation Program validates implementations of various cryptographic algorithms, including AES and other popular cryptographic algorithms. This is accomplished by designing and developing validation test sets for every such recommended cryptographic algorithm, with the aim to check whether the algorithm has been implemented correctly. In this work we applied knowledge extraction and visualization techniques via combinatorial coverage metrics to perform an analysis of the various CAVP test vectors. Our coverage measurement results can be used as a complementary measure to assess the quality of the AES algorithm validation suite.

Acknowledgments. The research presented in this paper was carried out in the context of the Austrian COMET K1 program and partly publicly funded by the Austrian Research Promotion Agency (FFG) and the Vienna Business Agency (WAW).

Moreover, this work was performed partly under the following financial assistance award 70NANB18H207 from U.S. Department of Commerce, National Institute of Standards and Technology.

Disclaimer: *Products may be identified in this document, but identification does not imply recommendation or endorsement by NIST, nor that the products identified are necessarily the best available for the purpose.*

References

1. Bassham III, L.E.: The advanced encryption standard algorithm validation suite (aesavs). NIST Information Technology Laboratory (2002)
2. Bell, K.Z., Vouk, M.A.: On effectiveness of pairwise methodology for testing network-centric software. In: 2005 International Conference on Information and Communication Technology. pp. 221–235 (Dec 2005)
3. Ghandehari, L.S.G., Lei, Y., Xie, T., Kuhn, R., Kacker, R.: Identifying failure-inducing combinations in a combinatorial test set. In: 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation. pp. 370–379 (April 2012)

4. Grindal, M., Offutt, J.: Input parameter modeling for combination strategies. In: Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering. pp. 255–260. ACTA Press (2007)
5. ISO/IEC JTC 1/SC 27: Information technology – Security techniques – Security requirements for cryptographic modules. https://www.iso.org/standard/52906.html (2012), [Online; accessed 01-April-2019]
6. Kuhn, D.R., Kacker, R.N., Lei, Y.: Estimating t-way fault profile evolution during testing. In: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC). vol. 2, pp. 596–597 (June 2016)
7. Kuhn, D.R., Okum, V.: Pseudo-exhaustive testing for software. In: 2006 30th Annual IEEE/NASA Software Engineering Workshop. pp. 153–158 (April 2006)
8. Kuhn, D.R., Reilly, M.J.: An investigation of the applicability of design of experiments to software testing. In: 27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2002. Proceedings. pp. 91–95 (Dec 2002)
9. Kuhn, D.R., Wallace, D.R., Gallo, A.M.: Software fault interactions and implications for software testing. IEEE Transactions on Software Engineering **30**(6), 418–421 (June 2004)
10. Kuhn, D.R., Kacker, R.N., Lei, Y.: Combinatorial coverage as an aspect of test quality. CrossTalk **28**(2), 19–23 (2015)
11. Kuhn, D., Kacker, R., Lei, Y.: Introduction to Combinatorial Testing. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series, Taylor & Francis (2013)
12. Leithner, M., Kleine, K., Simos, D.E.: CAMETRICS: A tool for advanced combinatorial analysis and measurement of test sets. In: 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 318–327 (April 2018). https://doi.org/10.1109/ICSTW.2018.00067
13. Mendoza, I.D., Kuhn, D.R., Kacker, R.N., Lei, Y.: Ccm: A tool for measuring combinatorial coverage of system state space. In: 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 291–291. IEEE (2013)
14. Mouha, N., Raunak, M.S., Kuhn, D.R., Kacker, R.: Finding bugs in cryptographic hash function implementations. IEEE Transactions on Reliability **67**(3), 870–884 (2018)
15. National Institue of Standards and Technology: ADVANCED ENCRYPTION STANDARD (AES). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf (2001), [Online; accessed 01-April-2019]
16. National Institue of Standards and Technology: Recommendation for Block Cipher Modes of Operation: Methods and Techniques. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf (2001), [Online; accessed 01-April-2019]
17. National Institue of Standards and Technology: SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf (2001), [Online; accessed 01-April-2019]
18. National Institue of Standards and Technology: Nist special publication 800-165 (2012), http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-165.pdf
19. National Institue of Standards and Technology: Cryptographic algorithm validation program. https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program (2019), [Online; accessed 01-April-2019]

20. National Institue of Standards and Technology: Cryptographic module validation program. https://csrc.nist.gov/projects/cryptographic-module-validation-program (2019), [Online; accessed 01-April-2019]
21. National Institue of Standards and Technology: FIPS 140-3 Development. https://csrc.nist.gov/projects/fips-140-3-development#schedule (2019), [Online; accessed 01-April-2019]
22. National Institue of Standards and Technology: SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf (2019), [Online; accessed 01-April-2019]
23. Simos, D.E., Kuhn, R., Voyiatzis, A.G., Kacker, R.: Combinatorial methods in security testing. Computer **49**(10), 80–83 (2016)
24. WALLACE, D.R., KUHN, D.R.: Failure modes in medical device software: An analysis of 15 years of recall data. International Journal of Reliability, Quality and Safety Engineering **08**(04), 351–371 (2001)
25. Yu, L., Duan, F., Lei, Y., Kacker, R.N., Kuhn, D.R.: Constraint handling in combinatorial test generation using forbidden tuples. In: 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 1–9 (April 2015)
26. Zeller, A.: Isolating failure-inducing input. In: IEEE Transactions on Software Engineering. Citeseer (2001)