From: "Miles Smid" <masmid@erols.com>
To: <aesround2@nist.gov>
Cc: <jfoti@nist.gov>, <edward.roback@nist.gov>, <william.burr@nist.gov>,
"Miles Smid" <smid@cygnacom.com>
Subject: AES Round 2 Comments
Date: Tue, 23 May 2000 17:44:41 -0400
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2615.200

Dear Sir:

Please see the three attachments containing my AES Round 2 Comments.

Thank you very much,

Miles Smid

May 22, 2000

A Strategy for Analyzing Public Comments and Preparing the Round 2 Status Report

Miles E. Smid
CygnaCom Solutions
An Entrust Technology Company

Introduction

This report was developed for the National Institute of Standards and Technology (NIST) as the first deliverable under Purchase Order 43SBNB067018. It is intended to suggest one or more strategies that the Computer Security Division at NIST can use to 1) systematically organize and analyze public comments on the AES finalists and the AES process, and 2) prepare a Round 2 status report that will also announce the AES winner(s). The opinions expressed in this report are those of the author, and are not intended to reflect any positions held by the NIST AES team or official positions of CygnaCom Solutions.

Strategy or Strategies

This report advocates one basic strategy with provisions for variations. It is felt that NIST must analyze public comments and select the AES winner(s) based on the criteria that it has announced as the basis of the AES selection process. That criteria consists of three major factors; security, cost, and algorithm implementation characteristics. However, certain issues have arisen as the result of public comments that need to be addressed. In particular:

a. Should NIST select multiple winners (future resiliency)?
b. How much credit should be given to candidates with a large security margin (speed versus security margin tradeoff)?
c. How important are low-end smart card implementations?
d. What is the relative importance of hardware versus software performance?
e. What modes of operation should be supported by AES?

These issues can and should be addressed in the context of the criteria as already specified by NIST. The intent is not to answer these questions here but rather to provide a framework whereby NIST can determine the answers. This report will develop a proposed strategy by examining the NIST criteria in detail, indicating where these and other issues apply, and making recommendations as to how NIST should proceed. The recommendations in this report are written in *italics* when contained within the body of the report. They are also fully listed in Appendix A. **Bold face** is used for major section headings and AES evaluation requirements as

stated by NIST. After examining the report NIST may accept recommendations, reject recommendations, substitute alternative approaches, or add new approaches as it sees fit.

The report recommends that NIST use the same basic criteria that it originally published. However, the report suggests that NIST use a much more rigorous process than the one employed to select the finalists. This process encourages NIST to develop scores for security, efficiency, and flexibility. Then an overall score for each of the thirty-one combinations of possible AES winner(s) may be determined. We assume that picking no winner is not an option. Any scoring system is no better than the accuracy of the data upon which it is based. However, such a process provides a methodology upon which to make conclusions. It would be better for NIST to derive its AES decision using a logical process than for NIST to use an undefined or illogical one. In developing these scores, NIST may be led to consider combinations and possibilities that it might not otherwise have considered. The disadvantage of this extra rigor is that it gives critics more information about the process to question. NIST is strongly encouraged to consult with its Statistical Engineering Division before finalizing the exact measures that it intends to use. Finally, a simple spreadsheet illustrating the procedures outlined in this report is provided so that NIST might consider the merits of the approach.

Multiple Winners

During the round 1 comment period, the comment was made that NIST should select multiple winners. In response to the comment, NIST said that it had not restricted the process to a single winner and began referring to "AES Winner(s)". Public comments have been divided on the issue. Generally those who favor multiple winners would like to have a backup in case one of the winners is found to be weak, patented, or inefficient in a particular application (future resiliency). Those who favor a single winner, believe that a single algorithm will support interoperability and efficient implementation. These issues will be discussed in more detail later.

By allowing for multiple winners, NIST has increased the number of cases that it must consider from five to thirty-one. In this report we shall use (n, k) to indicate n items taken k at a time. Therefore, there are (5,1) = 5 possible single winners, (5,2) = 10 possible double winners, (5,3) = 10 possible triple winners (5,4) = 5 possible quadruple winners, and (5,5) = 1 possible quintuple winner combinations. This does not imply that it was a mistake to allow multiple winners but rather it is just a consequence of the action. To some it may appear easy to eliminate certain combinations, but it is recommended that NIST not jump to conclusions before considering the reasons why a combination is to be rejected. *Since NIST is likely to be asked to justify the elimination of any possibility, all possible combinations of AES algorithms should be considered.*

We will now consider each of the AES selection criteria.

Security

- **Actual security of the algorithm compared to other submitted algorithms (at the same key and block size)**

Following the strategy outlined above implies that NIST must not only analyze the security of each of the five candidates but also the security of each combination of candidates. Let p_i be the probability that algorithm i will NOT be compromised during the life of the algorithm and let f^j_i be the fraction of all data protected by combination j that is protected by algorithm i. In this case $f^j_1+f^j_2+f^j_3+f^j_4+f^j_5 = 1$. Then the security provided by combination j would be given by

(1) $SECU_j = p_1f^j_1+p_2f^j_2+p_3f^j_3+p_4f^j_4+p_5f^j_5$ where $f^j_i = 0$ if algorithm i is not in combination j.

If NIST is able to assign relative values to p_i for each i, then the $SECU_j$ may be useful in comparing different combinations. For example, suppose that NIST determines that candidate 1 and candidate 2 offer significantly better security than the other candidates and that their probability of not being compromised is about equal ($p_1 = p_2 = p$). Suppose also that if two algorithms are used, each algorithm would protect half of the data. NIST would like to decide whether selecting one winner or two winners is better from a security viewpoint. Then

$SECU_1 = p(1) = p$ and

$SECU_2 = p(.5)+p(.5) = p.$

Therefore, neither combination offers any greater security than the other in terms of security. If the algorithms are equally strong, selecting multiple algorithms does not lower the security as has been claimed by some.

However, consider the above situation where $p_1 = .9$, $p_2 = .6$, $f^1_1=1$ (algorithm 1 used for all data), and $f^2_1=f^2_2=.5$ (protected data split equally between algorithm 1 and algorithm 2). Suppose NIST wants to decide between selecting candidate 1 only, or both candidate 1 and candidate 2. In this case

$SECU_1 = .9(1) = .9$ and

$SECU_2 = .9(.5)+.6(.5) = .45+ .30 = .75.$

Therefore, the more secure choice would be to select only candidate 1.

The above argument indicates the importance of selecting strong algorithms. *NIST should examine each of the five candidates and the analysis received in order to determine their relative security strengths. NIST should also examine*

each algorithm to determine whether it meets the minimum NIST security requirements.

Some additional points about $SECU_j$ need be made. It is not necessarily true that selecting the largest $SECU_j$ is always the best course from a security viewpoint.

a. The above analysis assumed that all data was protected by an AES winner(s) ($f^j_1+f^j_2+f^j_3+f^j_4+f^j_5 = 1$). It may be the case that the strongest candidate may not be appropriate for certain applications. For these applications other non-AES winners or no cryptography might be selected. The non-winners and non-cryptographic solutions may offer significantly less security than the runner-up AES candidates. Strong security not used is no security. (This is the flexibility argument applied to security.) Thus the lack of flexibility, which will be discussed later, can affect security. Data may be compromised that might otherwise have been protected had a more flexible approach been approved.

b. The above analysis only considers the security up to the point of compromise and does not consider how data is protected after compromise. If no other high quality algorithms are available, then systems will either have to run in a reduced security mode or discontinue operation. If NIST adopts multiple algorithms, some products will support more than one algorithm. In the case where a detected attack only applies to one of the algorithms, it might be very easy to convert to the other technique thereby protecting subsequent data. (This is the future resiliency argument applied to security.)

c. The use of multiple algorithms reduces the fraction of data compromised through an undetected compromise. An undetected compromise or a compromise not detected for many years on a single algorithm could be disastrous. (This is the "Don't put all your eggs in one basket theory".)

Argument a. might be countered by the argument that a single algorithm would increase interoperability and therefore provide for securing more data than would be secured by multiple algorithms. (However, this counter argument seems weak in light of the many algorithms that exist today for securing a variety of data.) In addition, too many algorithms make rigorous analysis less likely. Argument b. and argument c. carry significant weight.

NIST should balance the security provided by selecting the strongest algorithm against the security provided by selecting multiple algorithms that can support a greater number of applications and back up one another in case of compromise.

A single winner should be selected if only one algorithm meets the NIST security requirements.

A single winner should be considered if one algorithm offers a significant security advantage over the others.

Multiple algorithms should be considered if the best candidates appear to be equally secure and in light of the security arguments made in a., b., and c. in this (Security) section.

Multiple algorithms may also be considered when there are several candidates that offer strong security at differing levels. If multiple algorithms offer security that exceeds NIST's security requirements, but some exceed the requirements by more than others, it may still be prudent to allow multiple algorithms for cost and flexibility reasons as specified below. It would be unwise to select an expensive or inflexible algorithm when there was another efficient and flexible algorithm that, though less secure, still exceeded NIST's security needs.

- **The extent to which the algorithm output is indistinguishable from a random permutation on the input block**

  Modern block ciphers should be indistinguishable from a random permutation. *NIST should examine the submitted comments and its own statistical tests. If a candidate can consistently be distinguished from a random permutation, then it should be eliminated as an AES winner.*

- **Soundness of the mathematical basis for the algorithm's security**
  Other security factors raised by the public during the evaluation process, including any attacks which demonstrate that the actual security of the algorithm is less than the strength claimed by the submitter

  It is difficult to determine what is a sound mathematical basis and what is not. None of the algorithms offer a formal mathematical basis for their security. Some of the candidates seem to be mathematically simple while others appear to be complex. Those who favor simplicity argue that simplicity makes analysis easier and therefore the security of the algorithm may be more quickly determined. Others argue that complexity can itself add to the security of the algorithms. *If multiple algorithms are selected, NIST might pair a candidate known for its simplicity with a candidate known for its complexity.*

  Reduced Round Attacks and Security Margin:
  Reduced round attacks are useful in developing initial ideas for possible attacks on candidates and for determining the security margin. Some have argued that algorithms with a larger security margin are likely to offer greater security.

  NIST should attempt to determine the fair security margin, SM_i ranging from 0 to 1, of each algorithm. NIST should calculate a relative p_i for each algorithm i, considering SM_i as well as other security factors, and insert the results into

equation (1). By setting all f^j_i used in combination j to be equal or to other selected values, *NIST should determine $SECU_j$ for each combination j.*

For example, if NIST decides that its only measure of security is the security margin, then NIST might define p_i as follows:

$p_i = SM_i = 1-((\#\text{ rounds in best attack})/ (\#\text{ rounds in algorithm}))$.

However, NIST must keep in mind that a large security margin does not guarantee a strong algorithm and a low security margin does not necessarily mean a week algorithm. The DES and Skipjack algorithms are example of algorithms with low security margins that still seem to live up to their stated work factors. Nevertheless, it seems prudent in this early stage of the life of the AES candidates that a reasonable security margin should exist for any accepted candidate. *NIST should establish a minimum security margin for AES winners.*

Speed versus Security Margin:
One of NIST's questions was how NIST should weigh the security versus efficiency trade-offs in making the AES selection. While the security margin may be important when considering security, the speed versus security margin tradeoff seems irrelevant since it is unlikely that NIST will make such tradeoffs in the initial standard. The number of rounds of each candidate is fixed and therefore both the security and the speed are fixed for a given implementation. If an algorithm has a large security margin, then that should be considered when considering the security of the algorithms and if the algorithms is slow because of the extra security margin, then that fact should be considered when considering cost. *Unless NIST plans to allow the number of rounds of the algorithms to be varied in the initial AES standard, NIST should consider security margin and speed as issues under security and cost, respectively, just as it does other security and cost issues.*

Cost

- **Licensing requirements: NIST intends that when the AES is issued, the algorithm(s) specified in the AES shall be available on a worldwide, non-exclusive, royalty-free basis.**

Once NIST announces its selection of AES winner(s) it will be much more difficult to avoid licensing issues that may arise. *NIST should complete its patent search on the five finalists. Algorithms that are encumbered by reasonable patent claims should be avoided or ranked lower than other algorithms of equivalent status.*

NIST should not announce an algorithm as a winner until the owner agrees that the NIST write-up of the algorithm is free from any owner claims. The write-up of the algorithms for the standard should be flexible enough to provide for royalty-free upgrades in the future should they be necessary. For example, varying the

block size, key size, number or rounds, and key schedule to produce larger keys should not infringe on patents held by the owner.

Infringement attacks:
A standard is always a juicy candidate for infringement attacks whereby someone who was silent during the development process comes forth and claims that the standard infringes an existing patent. While these attacks cannot be prevented with certainty, NIST has been making provisions to minimize their likelihood. For quite some time, NIST has requested that those who have patents that are infringed by AES candidates should come forward with their claims. *NIST should again make a patent disclosure request when proposing the AES winner(s) as a Federal Information Processing Standard (FIPS).* In addition, NIST has performed a patent search on the five finalists.

If multiple winners of the AES are selected then NIST might have more flexibility in dealing with infringement claims. NIST could threaten to remove the selected algorithm from the winners list if the infringement claim could not be resolved. If only a single winner is selected, then it may be problematic to find a replacement. (This is the future resiliency argument as applied to infringement attacks.) It has also been pointed out that multiple algorithms may be more likely to attract at least one infringement claim. Nevertheless, NIST seems to have a stronger bargaining position with multiple algorithms. *Multiple candidates should be considered to mitigate infringement attacks.*

- **Computational efficiency: The evaluation of computational efficiency will be applicable to both hardware and software implementations. Round 1 analysis by NIST will focus primarily on software implementations and specifically on one key-block size combination (128-128); more attention will be paid to hardware implementations and other supported key-block size combinations (particularly those required in the "Minimum Acceptability Requirements" section) during Round 2 analysis.**

  Computational efficiency essentially refers to the speed of the algorithm. NIST's analysis of computational efficiency will be made using each submission's mathematically optimized implementations on the platform specified under "Round 1 Technical Evaluation" below. Public comments on each algorithm's efficiency (particularly for various platforms and applications) will also be taken into consideration by NIST.

  NIST should examine the results of its computational efficiency testing as well as the results of others. NIST should attempt to resolve inconsistencies in the results so that the best conclusions may be drawn. Algorithms that are efficient across all platforms should be given higher ranking in this area than those that are not. NIST should also examine combination of algorithms across different platforms.

  Software versus Hardware Efficiency:

Symmetric-key algorithms are used today because they are significantly more efficient than asymmetric-key algorithms. Therefore, efficiency is an important issue in the AES selection. Some have feared that too much emphasis has been placed on efficiency over security. However, when two algorithms offer comparable security, efficiency becomes a major deciding factor. Today cryptographic algorithms are probably most often implemented in software. Nevertheless, hardware tends to be used in cases where extra speed is required usually at an extra cost. Therefore, it seems reasonable to give credit to an algorithm that is efficient in both hardware and software.

- **Memory requirements: The memory required to implement a candidate algorithm--for both hardware and software implementations of the algorithm--will also be considered during the evaluation process. Round 1 analysis by NIST will focus primarily on software implementations; more attention will be paid to hardware implementations during Round 2.**

  Memory requirements will include such factors as gate counts for hardware implementations, and code size and RAM requirements for software implementations.

  Testing will be performed by NIST using the mathematically optimized implementations provided in the submission package. Memory requirement estimates (for different platforms and environments) that are included in the submission package will also be taken into consideration by NIST. Input from public evaluations of each algorithm's memory requirements (particularly for various platforms and applications) will also be taken into consideration by NIST.

  Memory requirements are most significant in low-end smart cards. This leads to the NIST question as to how important are low-end smart cards and related environments when selecting the AES algorithm(s). It appears that for the foreseeable future there will continue to be applications where memory is a restricted commodity. In addition, the memory requirements of the application will come ahead of the memory requirements for security. In these cases product developers will select encryption algorithms based upon these requirements. If no satisfactory algorithms exist, the product developers will likely develop their own algorithms. Such a decision may affect interoperability with AES systems as well as the security of the data protected.

  It has been argued that low-end smart cards cannot protect against some of the attacks that are known today. Therefore, the ability of an AES algorithm to support these cards should not be a factor for evaluation. While this argument has some merit, NIST should not discount low-end smart card applications until we a major trend to fewer of them.

  NIST should examine the memory requirements of each algorithm in its own tests and the tests of others. NIST should attempt to resolve inconsistencies in the results so that the best conclusions may be drawn. Algorithms that minimize

memory requirements should be given higher ranking in this area than those that do not. However, this factor alone should not eliminate an excellent algorithm that requires more RAM memory than is currently available today in low-end smart cards.

Relating speed, memory requirements, and key agility:
NIST lists both speed (efficiency) and memory requirements as cost considerations. In addition, the key agility of an algorithm may significantly affect its cost of operation. Therefore, NIST should consider all three factors when evaluating algorithm cost.

Multiple Algorithms, Interoperability, and Cost:
Selecting multiples algorithms can both reduce and increase costs. One can always select the fastest of the allowed algorithms for a given application thereby maximizing the efficiency and reducing the encryption costs. However, multiple algorithms are more expensive to implement and maintain. Interoperability was not listed as a selection factor but, at some level, it is an important feature of most standards. Those who favor a single algorithm often use interoperability as a major reason. They argue that the same level of interoperability with multiple algorithms comes at too great a cost.

NIST will have to consider the cost advantages of a single algorithm.

There is a question as to whether to compute an efficiency score or a cost score. This report recommends the efficiency score since efficiency is directly proportional to security and flexibility.

We will now suggest a possible method for combining speed, memory, and key agility for software and hardware implementations of the algorithms. First NIST must determine the software speed, the software memory, the software key agility, the hardware speed, hardware memory, and hardware key agility for each algorithm. NIST may then define s_i and h_i for each algorithm as follows:

$$s_i = ((\text{SoftwareSpeed}_i * \text{SoftwareKeyAgility}_i) / \text{SoftwareMemory}_i)$$

$$h_i = ((\text{HardwareSpeed}_i * \text{HardwareKeyAgility}_i) / \text{HardwareMemory}_i).$$

NIST can now determine the software and hardware efficiency ratings for each combination.

ERS_j = MAX(s_i where algorithm i is in combination j)/n, where n algorithms are in combination j

ERH_j = MAX(h_i where algorithm i is in combination j)/n, where n algorithms are in combination j.

We divide by the number of algorithms in each combination because implementing multiple algorithms increases the cost and therefore decreases the efficiency.

By determining the relative importance of hardware and software, NIST can calculate a combined efficiency rating for each combination. For example, suppose NIST determines that software efficiency should have a weight of wts and hardware efficiency of wth where wts + wth = 1. For each of the thirty-one AES algorithm combinations, NIST could calculate an efficiency rating, $EFFI_j$ where $0<j<32$.

$$EFFI_j = (wts*ERS_j) + (wth*ERH_j).$$

Since cost (efficiency) is a major selection factor, NIST should develop an efficiency rating that combines the software and hardware speeds, the software and hardware memory, the software and hardware key agility, and the cost of implementing multiple algorithms for each combination

This value will be used later when computing the overall score for a combination.

Algorithm and Implementation Characteristics

- **Flexibility: Candidate algorithms with greater flexibility will meet the needs of more users than less flexible ones, and therefore, inter alia, are preferable. However, some extremes of functionality are of little practical application (e.g., extremely short key lengths)--for those cases, preference will not be given.**

  Some examples of "flexibility" may include (but are not limited to) the following:

  a. The algorithm can accommodate additional key- and block-sizes (e.g., 64-bit block sizes, key sizes other than those specified in the Minimum Acceptability Requirements section, [e.g., keys between 128 and 256 that are multiples of 32 bits, etc.]).

  NIST should give credit to algorithms that can easily accommodate varying block size and varying number of rounds.

  b. The algorithm can be implemented securely and efficiently in a wide variety of platforms and applications (e.g., 8-bit processors, ATM networks, voice & satellite communications, HDTV, B-ISDN, etc.).

  Much of the discussion in the Memory Requirements Section would also apply here. *The ability of an algorithm to function on a wide variety of processors should be given significant weight.*

c. The algorithm can be implemented as a stream cipher, Message Authentication Code (MAC) generator, pseudo-random number generator, hashing algorithm, etc.

This does not appear to be a distinguishing factor since all the algorithms appear to meet these qualifications.

- **Hardware and software suitability: A candidate algorithm shall not be restrictive in the sense that it can only be implemented in hardware. If one can also implement the algorithm efficiently in firmware, then this will be an advantage in the area of flexibility.**

  Clearly, the AES winner(s) have to provide to both hardware and software implementations. It is highly likely that all the algorithms can be implemented in either software or hardware. Therefore, the issue seems to boil down to cost of implementation. For this reason, *NIST should combine hardware and software suitability into efficiency considerations made in the Cost Section.*

- **Simplicity: A candidate algorithm shall be judged according to relative simplicity of design.**

  The value of simplicity was difficult to judge when considering simplicity for security reasons. Some of the AES candidates appear to be simpler to understand or implement. Yet, they all are understood well enough to implement. Why should a simpler algorithm be preferred? In public-key cryptography, elliptic curve cryptography is probably the most difficult form of cryptography to understand. If the complexity of an AES candidate resulted in a more expensive implementation, then that factor should be considered in the cost section. If simplicity would make NIST more confident in the analysis of the security of the algorithm, then that factor should be considered in the security section. *NIST should consider what advantages simplicity provides that are not already considered in other sections. If this requirement only reflects the human tendency to prefer things simple, then it NIST should give it little weight.*

  This report argues that the major consideration under Algorithm Implementation Characteristics is flexibility. *NIST should develop a flexibility score for each combination of algorithms.* Unfortunately, this task requires subjective judgment not only for the individual algorithms but also for how the flexibility of the individual algorithms is combined to form the flexibility of the combination. NIST might first assign a flexibility score, v_i, to each of the five algorithms. Then for each multi-algorithm combination, NIST must decide how the individual algorithms combine to provide the overall flexibility of the combination. Each combination score, v_j, should be at least as large as the score for the most flexible algorithm in the combination, and no combination score should be greater than the sum of the scores of the algorithms in the combination. However, the algorithms that are flexible in the same areas will gain little by combination while the algorithms that compliment one another will gain significantly by the

combination. Finally, the scores should be normalized by dividing each score by the maximum score.

$$FLEX_j = v_j/MAX(v_i,i=1,31).$$

Modes of Operation

NIST should support the original DES modes (ECB, CBC, CFB, and OFB) for AES. NIST should consider the pipeline and interleave modes of ANSI X9.52 in order to provide for multiplexed encryption. NIST should consider supporting a 256-bit block mode that could be used as a hash function. In addition NIST should consider the PCBC modes specified in [1] to provide data authentication modes of encryption.

If multiple algorithms are selected, NIST should support a mode consisting of two or more algorithms whereby the compromise of any single algorithm leaves the mode as secure as the remaining un-compromised algorithm(s).

NIST should request papers on suggested modes of operation and then host a workshop discussing analysis of the suggested modes.

Combining the Scores

This paper suggests methods by which NIST can calculate three major measures.

1. $SECU_j$ measures the security of combination j by calculating the expected fraction of un-compromised data when using combination j.

2. $EFFI_j$ measures the efficiency of using combination j.

3. $FLEX_j$ measures the flexibility of combination j.

NIST should calculate an overall score for each combination by combining the security, efficiency, and flexibility measures. Suppose w_s, w_e, and w_f are the weights that NIST wants to place on security, efficiency, and flexibility respectively. Then NIST could define the overall score as

$$SCORE_j = (w_s\ SECU_j\ + w_e\ EFFIj + w_f\ FLEX_j).$$

In particular, NIST may decide to give a .5 weight to security, a .3 weight to efficiency and a .2 weight to flexibility. In that case

$$SCORE_j = (.5\ SECU_j\ + .3\ EFFI_j + .2\ FLEX_j).$$

The overall scores ($SCORE_j$) and the individual scores ($SECU_j$, $EFFI_j$, and $FLEX_j$ should be used by the NIST evaluation team as a guide for discussion and analysis. A spreadsheet could be devised which would aid each team member in performing the necessary computations.

Conclusion

It is hoped that the NIST evaluation team will find this report useful. Estimating the probability that a cryptographic algorithm will not be compromised is a difficult task. However, if NIST cannot distinguish between the security of the algorithms, then it has little choice but to rank them equal. In this case, efficiency and flexibility considerations will likely decide the winning combination. *NIST should compute several scores under differing input assumptions to obtain confidence that each score is realistic. Each member of the evaluation team should compute his or her own scores as inputs to the discussion and analysis process.* Small differences in scores should not be considered significant. But large differences may indicate a fundamental underlying cause. There may also be other factors, not easily accommodated in a score, that need to be considered. Nevertheless, this factor could be included in the flexibility score. It is hoped that after carefully taking into account all of the above considerations, NIST will be able to select the best combination of algorithms for the AES winner(s). Determining the best combination is more important than rigidly following any particular strategy.

There is more to the process of making a standard than picking the winner(s). NIST has to successfully negotiate several hurdles before the draft is presented to the Secretary of Commerce for signature. Some of the steps that might be followed in the process are outlined in Appendix B.

References

1. Integrity-Aware PCBC Encryption Schemes, Virgil Gligor and Pompiliu Donescu, 7th International Workshop on Security Protocols, Cambridge, U.K., April 1999, August 6, 1999.

Appendix A: Recommendations

1. This report advocates one basic strategy with provisions for variations.

2. Since NIST is likely to be asked to justify the elimination of any possibility, all possible combinations of AES algorithms should be considered.

3. NIST should examine each of the five candidates and the analysis received in order to determine their relative security strengths. NIST should also examine each algorithm to determine whether it meets the minimum NIST security requirements.

4. NIST should balance the security provided by selecting the strongest algorithm against the security provided by selecting multiple algorithms that can support a greater number of applications and back up one another in case of compromise.

5. A single winner should be selected if only one algorithm meets the NIST security requirements.

6. A single winner should be considered if one algorithm offers a significant security advantage over the others.

7. Multiple algorithms should be considered if the best candidates appear to be equally secure and in light of the security arguments made in a., b., and c. in this (Security) section.

8. Multiple algorithms may also be considered when there are several candidates that offer strong security at differing levels.

9. NIST should examine the submitted comments and its own statistical tests. If a candidate can consistently be distinguished from a random permutation, then it should be eliminated as an AES winner.

10. If multiple algorithms are selected, NIST might pair a candidate known for its simplicity with a candidate known for its complexity.

11. NIST should attempt to determine the fair security margin, SM_i ranging from 0 to 1, of each algorithm. NIST should calculate a relative p_i for each algorithm i, considering SM_i as well as other security factors, and insert the results into equation (1). By setting all f^j_i used in combination j to be equal or to other selected values, NIST should determine $SECU_j$ for each combination j.

12. However, NIST must keep in mind that a large security margin does not guarantee a strong algorithm and a low security margin does not necessarily mean a week algorithm.

13. NIST should establish a minimum security margin for AES winners.

14. Unless NIST plans to allow the number of rounds of the algorithms to be varied in the initial AES standard, NIST should consider security margin and speed as issues under security and cost, respectively, just as it does other security and cost issues.

15. NIST should complete its patent search on the five finalists. Algorithms that are encumbered by reasonable patent claims should be avoided or ranked lower than other algorithms of equivalent status.

16. NIST should not announce a algorithm as a winner until the owner agrees that the NIST write-up of the algorithm is free from any owner claims. The write-up of the algorithms for the standard should be flexible enough to provide for royalty-free upgrades in the future should they be necessary. For example, varying the block size, key size, number or rounds, and key schedule to produce larger keys should not infringe on patents held by the owner.

17. NIST should again make a patent disclosure request when proposing the AES winner(s) as a Federal Information Processing Standard (FIPS).

18. Multiple candidates should be considered to mitigate infringement attacks.

19. NIST should examine the results of its computational efficiency testing as well as the results of others. NIST should attempt to resolve inconsistencies in the results so that the best conclusions may be drawn. Algorithms that are efficient across all platforms should be given higher ranking in this area than those that are not. NIST should also examine combination of algorithms across different platforms.

20. NIST should examine the memory requirements of each algorithm in its own tests and the tests of others. NIST should attempt to resolve inconsistencies in the results so that the best conclusions may be drawn. Algorithms that minimize memory requirements should be given higher ranking in this area than those that do not. However, this factor alone should not eliminate an excellent algorithm that requires more RAM memory than is currently available today in low-end smart cards.

21. NIST will have to consider the cost advantages of a single algorithm.

22. Since cost (efficiency) is a major selection factor, NIST should develop an efficiency rating that combines the software and hardware speeds, the software and hardware memory, the software and hardware key agility, and the cost of implementing multiple algorithms for each combination

23. NIST should give credit to algorithms that can easily accommodate varying block size and varying number of rounds.

24. The ability of an algorithm to function on a wide variety of processors should be given significant weight.

25. NIST should combine hardware and software suitability into efficiency considerations made in the Cost Section.

26. NIST should consider what advantages simplicity provides that are not already considered in other sections. If this requirement only reflects the human tendency to prefer things simple, then it NIST should give it little weight.

27. NIST should develop a flexibility score for each combination of algorithms.

28. NIST should support the original DES modes (ECB, CBC, CFB, and OFB) for AES. NIST should consider the pipeline and interleave modes of ANSI X9.52 in order to provide for multiplexed encryption. NIST should consider supporting a 256-bit block mode that could be used as a hash function. In addition NIST should consider the PCBC modes specified in [1] to provide data authentication modes of encryption.

29. If multiple algorithms are selected, NIST should support a mode consisting of two or more algorithms whereby the compromise of any single algorithm leaves the mode as secure as the remaining un-compromised algorithm(s).

30. NIST should request papers on suggested modes of operation and then host a workshop discussing analysis of the suggested modes.

31. NIST should calculate an overall score for each combination by combining the security, efficiency, and flexibility measures.

32. NIST should compute several scores under differing input assumptions to obtain confidence that each score is realistic.

33. Each member of the evaluation team should compute his or her own scores as inputs to the discussion and analysis process.

Appendix B: Process for Selecting and Announcing the Winner(s)

The following process is suggested for selecting the AES winner(s):

1. Form an AES selection team

   The AES selection team should consist of a variety of individuals who are knowledgeable in the three major criteria areas upon which the selection will be based. The team should also include some members who are familiar with the standards development process at NIST.

2. Consider all official comments and AES3 Report

   The team will consider all official comments as well as the AES3 Report.

3. Develop and agree to scoring method

   The team members should discuss and agree to the scoring method that will be used to rate the algorithm combinations. It is important the team members are willing to use the technique and see how it works.

4. Compute scores

   Team members will then be asked to develop their scores for Security, Cost, Flexibility and overall Score for each of the combinations. These scores will be used as the basis for team analysis and discussions. The initial computation of scores should be done independently by each team member.

5. Discussion and analysis

   The team should meet and analyze their results. It may be necessary for individual team members to recomputed scores based on the discussions and analysis, but an individual should not recomputed a score just because the majority felt differently.

6. Establish consensus as to the winning combination

   Consensus does not require unanimity but it does require near unanimity.

7. Develop a report justifying the selection

   This report should describe the process used to select the winning combination and put forth a convincing argument for the decision.

8. Brief NIST Director on selection

9. Write up Federal Register Announcement and send it for publication

   Since the announcement need not reveal the actual combination that is selected, it can be developed and put forward prior to that selection of the winner(s).

10. Write draft of AES standard

The standard should parameterize the algorithm(s) so that the standard could be adjusted with respect to rounds, block size, key schedule, and key size in the future.

11. Perform final legal review of winning combination and make final agreements with submitters on draft standard and possible variances

    It is important that the winner(s) agree that implementation of the standard as written and with the adjusted parameters does not require that they be paid infringement royalties. If agreement cannot be reached, it may be necessary to consider another combination for the standard.

12. Obtain support for the new Draft Standard

    NIST should actively seek support from other government agencies (Treasury, GSA, and DOD) and standards groups (ANSI, IEEE, IETF, and ISO).

13. Make public announcement with statements of support

    The AES marks a significant accomplishment with is a major contribution to the security community. NIST should advertise this milestone.

14. Address public comments to draft AES FIPS.

    Expect some gripes and last minute tweaks.

15. Send FIPS to Secretary of Commence for approval.

16. Announce final AES FIPS approval.

    Congratulations!