The Effects of Multiple Algorithms in the Advanced Encryption Standard

Ian Harvey, nCipher Corporation Ltd, 4th January 2000

Abstract

This paper presents a discussion of the issues relating to the selection of encryption algorithms in practical situations. An AES standard which recommends multiple algorithms in a variety of ways is discussed, and it is shown that this can present an overall advantage.

Introduction

The Advanced Encryption Standard aims to become the first choice for most situations requiring a block cipher. To do this it has to satisfy a wide variety of requirements for - amongst other things - security, performance, and resource constraints.

Each of the current five candidate algorithms for AES satisfies a different balance of these constraints; the 'best' algorithm depends on circumstances, which are impossible to know beforehand. Furthermore, one of the principal requirements - that of security of the algorithm - cannot easily be measured; subjective judgements therefore must be made (based, for instance, on notions of 'safety margin' or 'conservative design') which may prove to be inaccurate or irrelevant in years to come.

In the absence of a precise definition of what constitutes the 'best' algorithm, or accurate means even to measure this, any choice of algorithm is somewhat arbitrary. In security terms this seems needlessly risky, and it has been suggested that avoiding the need for a single final algorithm would have advantages.

In the sections below, we discuss the factors which affect algorithm choice, and then examine the effect on these of an AES which offers multiple algorithms.

Factors in algorithm choice

Many factors may be involved in the selection of an algorithm. In most cases, one single factor is overwhelmingly the most important, and often some are of little or no importance.

Aside from performance issues, which are discussed later, criteria for algorithm selection include:

- **Security against theoretical attacks**

The reputation of an algorithm is frequently a major factor in its selection, both in terms of the design of the algorithm, and the extent to which it has been studied for cryptanalysis. A theoretical attack does not have to become practical before the cipher is rendered commercially unusable (for instance, liability insurance on a system using it may become void).

It is to be expected that no candidate with a known theoretical weakness will be given final recommendation within AES. Furthermore, the effect of the AES 'brand name' will be to concentrate research into the selected algorithms; this will (all being well) improve their reputation as time passes.

- **Security of implementations**

Some of the most practicable attacks of recent years have been directed against particular implementations of algorithms, rather than their theoretical definitions. These include timing attacks, power-analysis attacks, and fault-induction attacks.

In general, defending against these attacks is done when the implementation is designed, using a range of proprietary techniques. Some algorithms may have features which make them particularly difficult to defend, but it is generally not possible simply to define a 'good' feature set. Selection of an algorithm to resist implementation attacks can often only be done when the threat model has been decided, and little generalised guidance can be given.

- **Cost of implementation**

The effort required to correctly implement a given algorithm is frequently a major issue. For software implementations, the factors which affect this include:
- availability of reference implementations in a given language
- ease of understanding and adapting the reference implementation
- complexity of any optimisations required for optimum performance, and
- the ease and completeness of correctness testing.

For hardware implementation, important factors are:
- the complexity of the cipher
- the clarity of the cipher's given description
- availability of test vectors sufficient to give complete coverage.

Many developers will want to choose AES on the grounds of easy access to good reference material.

- **Architectural implications**

The precise functional 'shape' of an algorithm will have an impact on the way systems and protocols are designed to accommodate the algorithm. The block size and key size are the principal parameters, and fortunately all AES candidates are required to be compatible here.

However, additional parameters or features offered by particular algorithms - variable number of rounds, keys of other than standard length - may create additional complexities. Where it is not possible to adapt existing protocols to deal with these parameters, behaviour is often implied. This can be a major cause of interoperability problems - something which must not be allowed to bedevil the AES.

- **Legal issues**

Patent, copyright, and export-control issues affect no area of computing more than cryptography. Commercial developers will accept some licensing costs, but only up to a point - many of the potentially superior alternatives to DES (e.g. IDEA or RC5) have not been deployed widely, mainly for cost reasons. Free software developers are often unable to accept any restrictions on algorithm use.

One of the goals of the AES process is to produce a cipher which can be deployed universally, and a leading reason for choosing it will be freedom from legal impediments.

Performance issues

Every situation has its own unique performance criteria, which are invariably a trade-off between system requirements and speed, given 'enough' security. There are three main categories:

- **Best ideal-case speed**

The highest bit rate is required, irrespective of implementation complexity. The platform for deployment can be chosen (or at least unsuitable ones eliminated). Typically this means an algorithm which does well when hand-optimised in assembler for a modern processor, or can use parallelism in a large ASIC.

This type of performance is required by high-end hardware manufacturers, software developers who choose to target few platforms, and users who can choose platforms for best performance.

- **Best worst-case speed**

An acceptable bit rate is required, on a wide variety of platforms, or on a relatively non-standard platform. There should be no platforms on which speed is significantly lower than an alternative algorithm.

This type of performance is required by software developers who target a broad range of platforms, and is often associated with good speed available from a portable C implementation. It is also of significance to manufacturers (of e.g. embedded systems) whose choice of platform determined by other factors and cryptography is a secondary consideration.

- **Minimum implementation size**

Bit rate is not important, but constraints are placed on the resources required: gate count, code size or table size.

This type of performance is required by manufacturers of embedded systems for mass deployment, where unit hardware cost is critical. However, this group typically has much less need of interoperability outside the embedded application. The main reason to choose AES in this case will be the brand-name security.

It should be noted that performance and available resources will increase dramatically over time, but resistance to attacks will decrease. A standard intended for the long term should favour security over performance or resource requirements.

Approaches towards multiple algorithms

An Advanced Encryption Standard may be proposed which recommends more than one algorithm. There are a number of ways in which this might be done.

Multiple algorithms may be made optional; they must however be specified in such a way that any conforming AES implementation can interoperate with any other.

In some situations, both encryption and decryption can be controlled by one party, but in others they are controlled by separate parties and the choice of algorithm must be a mutually acceptable one. Any two such implementations must therefore share an algorithm, and the AES recommendations must guarantee this. The following approaches will be suggested:

A. All AES implementations must include all algorithms.

B. All AES implementations must include one primary algorithm, and a choice of secondary algorithms (possibly also ranked in order). Implementations will include secondary algorithms if it is to their advantage.

C. Given a set of N algorithms, an AES implementation must include at least N/2+1 algorithms from that set - this ensures any two implementations have at least one algorithm in common. Implementations will choose the subset of algorithms that best fulfils their requirements.

Theoretical Security / Implementation Security

Properly managed, multiple algorithm choice should enhance security. Should one algorithm fall to cryptanalysis, a second choice will already be available to provide backup. Also, given a choice, an developer will be able to pick the algorithm which best resists implementation attacks in the available technology.

Improperly managed, multiple algorithm choice will detract from security. If the choice of algorithm can be subverted in a given protocol, an attacker will be able to pick the easiest target.

Approach A is the most robust; two communicating parties can negotiate the 'strongest' algorithm and it will be used. Should an algorithm be broken, it is simply removed and the next best is selected.

Approach B allows implementers to choose to implement the secondary algorithms if they require a fallback. If secondary algorithms are broken first, all systems can revert to the primary algorithm, but if this is broken, some systems may not have an alternative. This scheme is therefore as resilient as approach A, except where systems omit the secondary algorithms due to cost considerations. In this approach, it would be prudent to choose a primary algorithm with a good security 'safety margin'.

Approach C, would in theory allow implementers to implement their choice of the 'strongest' algorithms, and two communicating parties would agree on at least one they considered secure. However, should *any* algorithm subsequently be broken, some combinations of communicating parties will be left unable to communicate. Also, this relies on the implementers holding opinions about algorithm security, which is contrary to the spirit of a security standard.

In practice, this makes approach C less secure than a single-algorithm selection.

The beneficial effect of an Advanced Encryption Standard concentrating cryptanalytic efforts will be diluted if too many algorithms are chosen. Approach B will present a primary target for research, and is best in this regard.

Cost of implementation

Any approach that mandates more than one algorithm to be implemented will raise the development costs proportionally. Approach A particularly, and to a lesser extent C, have the most impact. Where development costs are the overriding concern, approach B is as good as a single-algorithm selection.

The cost of implementation can be reduced dramatically, however, given good reference materials to accompany the standard. It is to be hoped that the various software implementations made available during the selection process will also be available to accompany the standard itself. This will strongly reduce the cost of producing a correct implementation.

It may be necessary to rewrite the descriptions of one or more algorithms to use a consistent set of terminology - particularly, for instance, with respect to bit-numbering and byte-ordering conventions.

Architectural implications

Multiple algorithms, and the process required to select one, will undoubtedly add to the architectural changes required for the new standard. In many situations, where a negotiation of cipher suite is already part of the protocol, this will have minimal additional impact.

Approaches A and B allow the selection to be made fixed, but approach C necessitates some form of negotiation, and this may be impossible in some circumstances.

The issue of additional algorithm parameters needs careful consideration. The table below gives some potential variation in parameters for each of the current AES candidates:

Cipher	Variations
MARS	Key size 4-39 32-bit words
RC6	Word size *w* , no. of rounds *r*, key size 0-255 bytes
Rijndael	Block length of 128, 192 or 256 bits
Serpent	Key size 0..256 bits
Twofish	Key size 0..32 bytes

All block ciphers can accommodate a 128-bit block, and 128, 192 and 256-bit keys as per the AES requirements, but beyond this the functionality differs substantially. In fact no two candidates have exactly the same set of allowed keys. It is poor software engineering practice to expose this to the user of the cipher; any variant in algorithm should *exactly* match the functional interface of the other, including rejecting the same set of invalid keys.

Similarly, if any variations in the number of rounds is proposed to allow a speed/security trade-off to be chosen by the user, it is poor design to let the user choose the number of rounds directly. Apart from the dangerous possibility of a round-by-round attack, it requires the user to know ‘good’ and ‘bad’ values for each algorithm. An acceptable solution would be to allow, say, three security levels - ‘minimum’, ‘medium’, and ‘maximum’, which is translated to a number of rounds appropriate to the algorithm in use.

Any AES which recommends more than one algorithm must address these issues, to remove ambiguity and promote interoperability.

Legal Issues

Clearly, multiple algorithms may increase the legal complications for a developer. In an ideal case, *any* algorithm offered as an option in the final AES will be free for use without restriction. As a minimum, sufficient algorithms should be available to construct a conforming implementation, without any patent or similar restriction.

Approach A requires all algorithms to have no restrictions; approach C requires the majority to have no restrictions, and approach B requires the primary algorithm alone to have no restrictions.

Performance - ideal case

Multiple algorithms give the best opportunity to maximise absolute speed, especially as evolving technology changes the balance between operations in different algorithms.

Approach A is good for this; the communicating parties will negotiate the fastest algorithm, and this is guaranteed to be available.

Approach B has some merit; the primary algorithm may not be the fastest on the chosen platform, but the implementer can add the secondary algorithms if they improve speed. In the ideal case, both communicating parties will do this and speed is maximised.

Approach C also allows implementers to choose the subset of algorithms which are most efficient on the chosen platform. Two communicating parties should then be able to pick their mutually fastest choice.

Performance - best worst-case

This benefits greatly from a choice of algorithms. Most worst-cases will be a particular feature of an algorithm which behaves poorly on a particular platform, and often any alternative will help.

The same strategy for choosing algorithms can be adopted as in the ‘ideal case’ scenario, with similar results. The worst case in Approach B is when the primary algorithm has poor performance on a given platform, and one of the communicating parties does not support any secondary algorithms. This is still an improvement, however, because it will only occur in those few cases where there is an overwhelming need for cost saving.

Minimum size requirements

This is impeded by the requirement for multiple algorithms; any standard which mandates more than one to be implemented will severely impact costs for low-end system developers.

To a certain extent this can be mitigated where two algorithms share common large functional blocks, or memory requirements which can be overlaid. To demonstrate this, the table below summarises the major functional requirements for each AES candidate. For comparison, triple-DES is also listed.

The table sizes given are 'minimum' requirements, with a 128-bit key, and will not be for the most efficient possible implementation; the RAM size given does not include that required for working purposes. The ROM size may be misleading as it does not include code size, which is in some cases traded off against lookup table size.

Candidate	ALU Operations					Table size (bytes)	
	logic / fixed shift	add/sub	data-dep shift	GF (2^P) ops	mult	ROM (S-boxes, etc.)	RAM (key schedule, etc.)
MARS	X	X	X		X	2048	160
RC6	X	X	X		X	none	176
Rijndael	X			X		512	16
Serpent	X					128	32
Twofish	X	X		X		64	24
Triple-DES	X					256	24

So it can be seen, for instance, that adding RC6 to a chip designed to implement MARS would have relatively little impact, but adding it to one optimised for Rijndael might be difficult.

Approach A is the worst of all worlds for minimum-size implementations. Approach C is better (only the smallest algorithms should be selected) but would still be typically double the best-case cost. Approach B allows very resource-limited implementations to implement solely the primary algorithm, and is as good as the single-algorithm case.

Summary of results

The effect of the various possibilities for a multiple-algorithm standard can be summarised in the table below, where "++" indicates the most positive impact, '0' indicates no impact compared to a single-algorithm standard, and "--" is the most negative impact.

Category	A	B	C	Notes
Security	++	+	--	1
Impl. Cost	--	0	-	2
Architecture	-	-	--	2
Legal issues	--	0	-	2
Best-case speed	++	++	++	
Worst-case speed	++	+	++	
Minimum size	--	0	-	

Notes:

1. Based on the ability of the standard to continue given failure of a cipher.
2. Can be mitigated by a good standardisation process.

Conclusions

A number of approaches to specifying multiple algorithms have been presented. This suggests that approach B - to specify a required 'primary' algorithm and one or more optional 'secondary' algorithms - has advantages over other approaches, and allows potential speed and security improvements over a single algorithm selection.

This approach means that outright performance can be eliminated from the criteria for primary algorithm selection - this can be left for the secondary algorithm. Security (or in practice, safety margin and conservative design) should be the primary algorithm's main requirement, followed by a modest resource requirement for minimum-size implementations.

Similarly, resource requirements can be ignored when making the secondary algorithm selection; implementers seeking a lowest-cost solution can simply omit these. An algorithm more aggressively optimised for performance is ideal here.

Provided that the legal and functional differences between the algorithms are mitigated by a well-written standard, there is no reason that this approach should not offer the best of all worlds.