Implementation and analysis of the Rijndael encryption algorithm in different programming languages

Daniel Fowles

Bachelor of Science in Computer Science with Honours
The University of Bath
April 2008
This dissertation may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signed:
Implementation and analysis of the Rijndael encryption algorithm in different programming languages

Submitted by: Daniel Fowles

COPYRIGHT
Attention is drawn to the fact that copyright of this dissertation rests with its author. The Intellectual Property Rights of the products produced as part of the project belong to the University of Bath (see http://www.bath.ac.uk/ordinances/#intelprop).

This copy of the dissertation has been supplied on condition that anyone who consults it is understood to recognize that its copyright rests with its author and that no quotation from the dissertation and no information derived from it may be published without the prior written consent of the author.

Declaration
This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed:
Abstract

The purpose of this project is to investigate and analyze the implementation and performance of the Rijndael encryption algorithm in several different programming languages, with the aim of being able to make recommendations on its use in the context of larger systems with their own requirements.
Contents

CONTENTS .................................................................................................................I
ACKNOWLEDGEMENTS ....................................................................................II
INTRODUCTION .................................................................................................1
  1.1 THE RIJNDAEL CIPHER .................................................................1
  1.2 OBJECTIVES ......................................................................................1
LITERATURE SURVEY .....................................................................................3
  2.1 CRYPTOGRAPHY ...............................................................................3
  2.2 ENCRYPTION STANDARDS5
    2.2.1 Data Encryption Standard (DES) ......................5
    2.2.2 Triple DES .................................................................5
    2.2.3 Advanced Encryption Standard (AES) ..............6
  2.3 RIJNDAEL ALGORITHM .................................................................6
    2.3.1 Existing Implementations ........................................7
    2.3.2 Performance .................................................................7
  2.4 PROGRAMMING LANGUAGES ......................................................7
    2.4.1 Java ...............................................................7
    2.4.2 C ...............................................................7
    2.4.3 Perl .............................................................8
    2.4.4 JavaScript ......................................................8
CONCLUSIONS .............................................................................................9
BIBLIOGRAPHY ............................................................................................10
Acknowledgements

- My project supervisor, Dr Russell Bradford.
Chapter 1

Introduction

For some computer systems, encryption is an important requirement due to the nature of the data the system handles, or the channels through which the data will be sent. However, the process of encryption and decryption can adversely affect the performance of a system due to the time taken by these operations. These can be affected by factors such as the strength of the encryption, the size of the data being encrypted, or the programming language being used. These consequences of using encryption need to be carefully considered when designing a system. The aim of this project is to analyse the implementation and performance of the Rijndael algorithm across several different programming languages.

The Rijndael Cipher

The Rijndael algorithm (Daemen and Rijmen, 2002) is a cipher that was invented by Vincent Rijmen and Joan Daemen. The Rijndael algorithm is a block cipher which means that for a given encryption key, the algorithm performs a series of transformations on a block of data. The same encryption key can be used to reverse these transformations in the process of decryption. Rijndael was selected as the Advanced Encryption Standard (AES) in 2001 by the National Institute of Standards and Technology (NIST). As a result the AES specification (FIPS 197, 2001) was created, although it should be noted that this is limited to 128-bit blocks of data, using 128, 192 or 256-bit keys.

Objectives

In order to achieve the aims outlined above, the main objective in this project is to create implementations of Rijndael in at least 4 different programming languages. The initial set of
languages I will develop the algorithm for are:

- C
- Java
- JavaScript
- Perl

These languages are to be used because they cover a range of intended usages, which reflects the different types of applications which may require encryption. For example, Java and JavaScript are used extensively in programming for web-based systems.

In practice, implementing the algorithm means creating a set of functions which allow for the encryption and decryption according to the Rijndael specification. My implementations will go beyond the AES specification, and support the variable key and block sizes mentioned in the original Rijndael specification (Daemen and Rijmen, 2002). To facilitate the performance analysis of the different versions, each implementation must be capable of outputting relevant data, such as the time taken to perform different stages of the encryption or decryption process. I will use this data to analyse the performance of the different implementations. This will be done by monitoring factors such as the time taken to complete encryption and decryption using certain key and block sizes. I will also investigate how the performance of different operations in the algorithm varies between the languages.

For each language, I will also create a small application which uses these functions. These applications will make full use of the functions provided by that language's implementation of the algorithm, and each application will represent a typical use of the chosen language. This means that the C implementation may be used to produce a fast application for file encryption and decryption, whereas the Java implementation may be used on a web applet. The process of developing these applications should be used to assess any practical issues which need to be considered when working in the different languages, such as how the algorithm is adapted to handle the encryption of files larger than the specified blocks.

By comparing the different versions of Rijndael in the selected languages and analysing their performance, I hope to be able to make recommendations on the choice of language for an implementation based on performance requirements. The data gathered while analysing the systems and any subsequent discussions of my implementations should be written so that they may be of use to developers who need to implement the Rijndael algorithm as part of a larger system. My project cannot cover all possible scenarios for implementing Rijndael, but should identify and specify the performance of the encryption in the selected languages.
Chapter 2

Literature Survey

This project aims to investigate the practical implementation and performance of the Rijndael encryption algorithm in a selection of different programming languages. This will be done through the development of several small systems which make use of it. This process should give an insight into the ease with which Rijndael can be implemented in the languages and thus the suitability of using this encryption in certain types of system. Secondly, by analysing the performance of the different implementations, it should be possible to make recommendations on the use of Rijndael and how it operates as part of a larger system. These results could be useful to software developers designing a system which makes use of Rijndael encryption, where the requirements for effective cryptography need to be balanced with the other system requirements.

Before beginning this project, it is important to realise that development of correct and useful software requires an understanding of the aims and requirements of the system being developed. In this case, it is therefore important to not only understand how Rijndael works and why it should be used, but also why we use encryption at all. For this reason, my literature review will begin by briefly examining the reasons for using cryptography, the different types of encryption, and some issues to be considered when building systems which use encryption.

2.1 Cryptography

Cryptography is defined by Konheim (1981) as:

"the art and science of making communications unintelligible to all except the intended recipient(s)."

In practice, cryptography is used as a method of ensuring the privacy of data communicated between two parties. We consider the case of two individuals who need to communicate with each other, but need to use channels which cannot guarantee that their communications
remain private. A method of encryption known to both individuals may be used to ensure that their communications cannot be read by a third party. A relevant example may be that of two people communicating using the Internet, where their data may have to travel through computers outside of their control. A successful method of encryption facilitates their communications yet renders the data useless to anyone else.

Encryption is the process of applying an encryption algorithm (known as the cipher) to the initial data (known as the plaintext) to produce a ciphertext. The ciphertext should be unreadable to anyone unable to decrypt the message – an ability which commonly depends on the knowledge of a special piece of data known as a key. A key is a parameter to the algorithm which is used in the encryption and decryption process. Typically the key is shared privately between the users; although not always, as is the case with public key cryptography. The use of a key means that the encryption algorithm itself does not need to be kept secret, because knowledge alone of the method used is not enough to decrypt the message. It also means that while an algorithm may be widely used, data encrypted with it can still only be read by a party with the correct key.

The size of a key in an encryption algorithm is an important factor in the overall security of the cipher. If the key size is too small then an attacker may be able to decrypt a message by trying all possible keys until the right one is found – this is known as a brute force attack. However, depending on the design of the algorithm there may be quicker ways to find the correct key. These methods of cryptanalysis can depend on the mathematical properties of a cipher and analysis of cipher texts. It is important to note here that the breaking of an encryption algorithm can have different meanings. In a strict mathematical sense, an encryption may be classified as broken if there is an attack better than an exhaustive key search (brute force). Even if this attack is impractical, it would still be theoretically possible given the appropriate input and computing power. However from a software engineering perspective, an attack needs to be realistic and practical given the circumstances within which it is implemented. For some systems, the data may not exist long enough that very strong encryption is required. This is a good reason to investigate the Rijndael algorithm at various strengths. Although it may seem more sensible to use the strongest encryption possible, this approach does not take into account the other requirements a system may have, such as the speed with which encryption and decryption must take place.

With regard to system design, it is important to note here that the security of the data within a system does not just depend on the strength of the encryption used. A system must be careful that it does not unnecessarily expose the data during encryption or decryption, and that the system itself does not provide an attacker with opportunities for more efficient cryptanalysis. As demonstrated by Bernstein (1998) in his paper on the AES algorithm (Rijndael) is still vulnerable to attacks which are unrelated to the strength of the cipher. Bernstein’s paper shows that it is possible for an attacker to retrieve a key through analysis of the computer that is performing the encryption and decryption. In reference to Bernstein, Schneier (1998) refers to these as side-channel attacks and observes that:

“These attacks don't necessarily generalize. A fault-analysis attack just isn't possible against an implementation that doesn't permit an attacker to create and exploit the required faults.”
But these attacks can be much more powerful. For example, differential fault analysis of DES requires between 50 and 200 ciphertext blocks (no plaintext) to recover a key. Contrast this with the best non-side-channel attack against DES, which requires just under 64 terabytes of plaintext and ciphertext encrypted under a single key.” (Schneier, 1998).

While analysing my implementations of Rijndael with respect to these particular attacks falls outside the scope of this project, it does raise the issue of the extent to which an implementation of an algorithm is weakened by the choice of language or hardware. An algorithm may seem secure in terms of standard mathematical cryptanalysis, but the process of interpreting it through a programming language and a piece of hardware could create weaknesses which were not necessarily predictable.

2.2 Encryption Standards

Konheim (1981) notes that while the study of cryptography has been around for hundreds of years, it has typically been used only for government or military purposes. However, as the use of insecure methods of communication such as the Internet increases, so there is a growing need for the use of encryption in day-to-day exchanges. We can also note that as ever more information is stored on computers, maintaining the integrity and security of personal or otherwise valuable data becomes more important. Clearly it would be much harder to meet these needs if every new system was required to invent and implement its own unique encryption algorithm. However, due to the creation of open standards in encryption, strong encryption algorithms can now be much more readily available.

2.2.1 Data Encryption Standard (DES)

The first major standard in encryption came with the arrival of the Data Encryption Standard (DES). DES is a block cipher that was developed in the United States by IBM in 1974 adopted as a national standard by the United States National Institute of Science and Technology (NIST) in 1977 (Coppersmith, 1994). With DES’ subsequent reaffirmation as a standard in 1983, 1988, 1993, it saw widespread across organisations and systems where encryption was required. However, in the 1990s it became clear that the DES algorithm was no longer secure enough, as attacks using the methods of differential and linear cryptanalysis were used to successfully break DES keys.

2.2.2 Triple DES

In 1999, DES was reaffirmed again, but with the recommendation that a new variant of DES was used instead. This is known as the Triple Data Encryption Algorithm (TDEA) or Triple DES. This algorithm used DES as a component to provide greater security through increased key length. While still an accepted standard, the Triple DES is being gradually replaced in favour of the Advanced Encryption Standard.
2.2.3 Advanced Encryption Standard (AES)

The process for developing a new encryption standard known as the Advanced Encryption Standard was announced by the US NIST in 1997. The open selection process invited the submission of ciphers, which the cryptology community would be encouraged to analyse and evaluate. Through a series of conferences over the following years, the candidates would undergo further testing and evaluation until the Rijndael algorithm was announced by NIST as the new AES in 2000.

NIST originally announced that it was “looking for a block cipher as secure as Triple DES, but much more efficient” (Daemen and Rijmen, 2002). The selection process did not just take into consideration the security of the candidate algorithms, but other factors which were relevant to future usage of the algorithm. As noted by Daemen and Rijmen (2002) candidate algorithms were evaluated against criteria such as their potential implementation on different physical platforms, the time required to setup a key, and the simplicity of the algorithm. The conclusion of the choice of Rijndael was summarised as follows (report by Nechvatal et al. (2000, cited by Daemen and Rijmen, 2002):

“Rijndael appears to be consistently a very good performer in both hardware and software across a wide range of computing environments regardless of its use in feedback or non-feedback modes. Its key setup time is excellent, and its key agility is good. Rijndael’s very low memory requirements make it very well suited for restricted space environments, in which it also demonstrates excellent performance.”

2.3 Rijndael Algorithm

Having now identified the motivations for using cryptography, some of the key issues to be considered, and the origins of Rijndael, it is suitable to examine the algorithm in more detail and look at existing implementations of it.

As previously mentioned, the Rijndael algorithm is an iterative block cipher. It has a variable key and block length, which gives it a degree of flexibility when choosing an appropriate implementation. The block and key sizes can be any multiple of 32 bits between 128 bits and 256 bits. This is the key difference between the Rijndael algorithm and the version of it specified as the AES, as the AES fixes the block length at 128 bits and key sizes of 128, 192 or 256 bits. My project will look at the Rijndael algorithm with varying key and block sizes, because while the algorithm remains the same, there will be a larger set of test results to evaluate. This should lead to more accurate implementation-specific recommendations.

As an iterative block-cipher, the Rijndael algorithm takes a block of data as the input and performs a number of ‘Round Transformations’ on it. The Round transformation is actually a series of four separate transformations. These operations are applied in order for a set number of rounds, followed by a slightly altered final round which completes the encryption.
2.3.1 Existing Implementations

As Rijndael has existed as the Advanced Encryption Standard since 2000, there are already many implementations of the algorithm. These implementations are available in many different programming languages. As the AES standard is open, organisations or users which wish to implement the Rijndael algorithm are free to do so.

2.3.2 Performance

As previously described in the AES section, one of the key reasons for Rijndael’s selection was its good performance across different platforms, from 8-bit smart cards to standard 32-bit computer processors, as well as having the potential for fast dedicated hardware.

Some research has already been carried out into the speed of the Rijndael algorithm in Java and Orgill (2005) is a demonstration of this. In his paper he finds that the speed of a Java implementation can actually be affected by the different type of Java Virtual Machine that is used to run the code. Although this research is not directly relevant to my project, it shows that there is definitely potential for speeding up Rijndael depending on how it is implemented in the various languages.

2.4 Languages

As previously stated, this project requires the implementation of the Rijndael algorithm in systems constructed with several different programming languages. Therefore it is important to examine the history and nature of these languages, so as to understand how they might be used and how cryptography might be used in a system developed in that language. The languages to be used in this project are Java, C, JavaScript and Perl.

2.4.1 Java

Java is an object-oriented language that was created and developed by software engineers at Sun Microsystems during the early 1990s. Code written in Java is compiled into bytecode which can then be executed on the Java Virtual Machine. In practice this means that programs written in Java can be run on any computer which supports a JVM, regardless of the underlying hardware or operating system, which is a key feature of the language. While Java can be used to create desktop applications, it can also be used to create Java applets. These are applications which can be distributed over the internet and run within a web browser. Therefore a potential application for cryptography would be an applet which encrypts data on the client’s computer before submitting it to a server.

2.4.2 C

The programming language C was originally designed in 1972 and used as the systems language for UNIX. It gradually evolved in the years following until in 1990 the American National Standards Institute (ANSI) set the standard for the language, known as ANSI C.
C is a relatively low level programming language that gives the programmer more control over aspects such as memory allocation. Like Java, C has been widely used and has a variety of potential applications, but for this language, a suitable program might be one that encrypts or decrypts files from a command prompt.

2.4.3 Perl
Perl is a scripting language like JavaScript, and grew because of its strengths in text manipulation. Perl is often regarded as a general purpose programming language which can be used to automate a variety of different tasks. While there is no ‘typical’ Perl system, one which used cryptography might do so for the encryption of data being passed between other systems.

2.4.4 JavaScript
JavaScript is a scripting language that was created in the early 1990s. It is commonly used on the internet to provide client-side functionality in web pages. It has no technical relation to Java, although both languages have a similar syntax. A system using JavaScript on the internet might make use of cryptography when displaying encrypted content to a user.
Chapter 3

Conclusions

In this Literature Review we have explored the reasons behind the use of cryptography which justify the importance of research and development into various cryptographic techniques. It has also been acknowledged that encryption algorithms can still be vulnerable to attacks not directly related to their key strength, which demonstrates the importance of considering the overall security of a system, as opposed to treating encryption like a separate function within a system. This justifies the research of my project into practical implementations of the Rijndael algorithm.

Unfortunately, the task of developing and analysing the different implementations of the Rijndael algorithm that comprises the remainder of this project is beyond my current capabilities, and has not been completed. I have not developed the systems required in the previously specified languages or recorded the data required for analysing the performance of the algorithm. Therefore this project ends here with the literature review.
Bibliography


