2006

Efficient encryption on limited devices

Roderic Campbell

Masters Project Proposal: Efficient Encryption on Limited Devices

Roderic Campbell
Department of Computer Science
Rochester Institute of Technology
Rochester, NY, USA
rmc8917@cs.rit.edu

June 24, 2004

```
----------------------------------------
Chair: Prof. Alan Kaminsky          Date

----------------------------------------
Reader: Prof. Hans-Peter Bischof    Date

----------------------------------------
Observer: Prof. Leonid Reznik       Date
```

1 Summary

As the capstone of my Master's education, I intend to perform a comparison of Elliptic Curve Cryptography(ECC) and The XTR Public Key System to the well known RSA encryption algorithm. The purpose of such a project is to provide a further understanding of such types of encryption, as well as present an analysis and recommendation for the appropriate technique for given circumstances.

This comparison will be done by developing a series of tests on which to run identical tasks using each of the previously mentioned algorithms. Metrics such as running time, maximum and average memory usage will be measured as applicable.

There are four main goals of Crypto-systems: *Confidentiality, Data Integrity, Authentication* and *Non-repudiation*[5]. This implementation deals only with confidentiality of symmetric key exchange. At no point do I consider, authentication of the public key with signatures. This is beyond the scope of this project.

The motivation for such research is the heightened use of small devices such as mobile phones, PDAs and embedded processors. These limited devices have reduced memory, smaller CPUs and less battery power, allowing much less electricity and computational power, while still performing at an acceptable level. These devices are utilized everyday and the need to secure communication between such devices grows increasingly important with the emergence of each new technology.

2 Encryption and Small Devices

With these limited devices being an ever growing portion of our daily lives, the need to secure communication becomes increasingly apparent. Soon, small devices will be used in applications of e-commerce, portable chat clients, as instruments of web banking, and countless other security sensitive uses.

Given the limited resources of the devices described and the complexity of key

encryption computation in RSA, it becomes apparent that today's encryption standards will prove to be unacceptably costly on a limited device.

2.1 Public Key Cryptography Foundation

Public key crypto-systems are based on the concept of a one way function. An example of a two way function is addition. With the equation $A+B=C$, and given A and B, it is easy to calculate C. Likewise, if we are given B and C, we could easily compute A by using the inverse of addition *subtraction*. This gives us the equation $C-B=A$. A one-way function on the other hand, is one where it is not computationally feasible to come up with one of the arguments, given the remaining arguments and the solution.

2.2 RSA

RSA is based on the one way function of modular exponentiation. It is easy to find the modulo of a number which has been raised to a large number, but given the solution and the large number, it is not easy to find the original number.

The use of RSA usually depends on extremely large prime numbers. The numbers must be this large to ensure sufficient security. This is where the problem of RSA comes in. While it is very secure, it is also very costly.

2.3 Elliptic Curve Cryptography

This leads us to Elliptic Curve Cryptography (ECC). It is believed that ECC gives remarkably similar levels of security at a fraction of the cost of RSA. With a further understanding of ECC, small devices may utilize the technology to exchange encryption keys. These keys may be used to encrypt bank transactions, credit card numbers, and other sensitive information.

Elliptic curves deal with the functions of the form $y^3 = x^2 + ax + b$. In order to determine a shared secret value, Alice and Bob exchange some public

information. Several calculations later, a private key is derived. The process is as follows [5]:

1. Bob and Alice agree on a non-secret elliptic curve of the form above and a curve point F.
2. Alice chooses a random integer A_k and Bob chooses a random integer B_k.
3. Alice publishes a point on the graph $A_P = A_k * F$ and Bob publishes $B_P = A_k * F$ where $*$ represents a special form of curve multiplication.
4. Alice calculates $A_k * B_P$ and uses this as the secret key. [1]
   (a) Note that Bob can determine the same number: by calculating $B_k * A_P$, since $Bk * AP = Bk * (Ak * F) = (Bk * Ak) * F = Ak * (Bk * F) = Ak * BP$.
   (b) Note that the security of this scheme is the inability to derive k from F and $k * F$.

The study of Elliptic Curve Cryptography will give us an introduction to, and a strong application for group oriented keys and the in depth concepts of Finite Fields and calculations within these fields.

2.4 XTR

Efficient Compact Subgroup Trace Representation [3] estimates that encryption using XTR is 21% faster than ECC, and that decryption can be up to 45% faster. This evidence, should it prove to be valid, provides powerful arguments for the widespread investigation of XTR. This significant drop in memory usage and running time could lead us to the next key distribution standard.

[3] states that XTR key selection is much faster than RSA and orders of magnitude easier and faster than ECC. For these reasons, [3] insists that XTR will find its home in small devices such as smart cards and wireless devices. This complements the goals and direction of this implementation.

Further examination of XTR and the experiments performed shall lead me to a conclusion about which situations or devices XTR would find an appropriate home in.

3 Functional Specification

3.1 Desktop Environment

The Desktop environment consists of a 2.4 Ghz Pentium 4 processor with 1 GB PC-2700 DDR RAM. The system will be running Java 1.4.2 key server.

3.2 Limited Device

The limited device will consist of a Samsung VGA 1000 Java enabled mobile phone. Java 2 Platform, Micro Edition with the Connected Limited Device Configuration(CLDC), and the Mobile Information Device Profile make up the Java Platform which will be used[2]. The device itself contains 1024KB of memory for Java Applications.

3.3 Software Implementation

The implementation will consist of the following tasks:

1. An Implementation of RSA, ECC, and XTR public key encryption and decryption in Java in a limited computing environment.
2. Implemented underlying algorithms like arbitrary precision integer arithmetic in Java in a limited computing environment described above.
3. Implement parameter generation for the above in Java in a desktop computing environment.
4. A system of metrics to analyze the following:

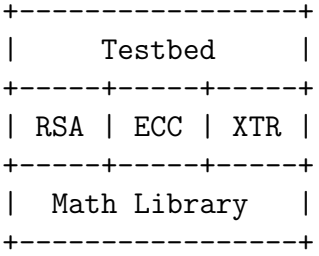
(a) Memory usage and running time of 2048-bit RSA in comparison to ECC and XTR, each with parameters which provide an equivalent level of security for the task of encrypting and decrypting a 256-bit message (a secret key), in the limited computing environment and the desktop computing environment. This encryption will be used for distributing the AES symmetric key for encryption and decryption of the target sensitive messages being sent between the clients.

(b) Measure the memory usage and running time of 4096-bit RSA and equivalent-sized ECC and XTR, in the task of encrypting and decrypting a 256-bit message (a secret key), in the limited computing environment and the desktop computing environment.

(c) The metrics will lead us to a recommendation of the best public key algorithm to use in a given limited computing environment.

4 Architectural Overview

The overall architecture of the system can be viewed with this simple diagram showing the relationship between the major subsystems of the implementation



The individual packages are described in the following sections.

4.1 RSA Implementation

The basic algorithm for RSA is quite simple and easy to understand. The math behind it deals with the understanding of the properties of modular exponentiation.

1. Given primes P, Q
2. $N = PQ$
3. $ed = 1\ mod(p-1)(q-1)$
4. $C = M^e mod(N)$
5. $M = C^d mod(n)$

The values of e and d represent the public and private keys. Using these gives us the ciphertext from the plaintext and vice versa. Step four is the actual encryption and step five is the decryption.

The arbitrary precision integer arithmetic required for these calculations is explained later in this section.

4.2 Elliptic Curve Implementation

The elliptic curve library will consist of the communication protocol to exchange the appropriate public information. When the public information is established, each of the parties will utilize methods in the math library to come up with random integers and calculations for the private key information. At this point, more communication will occur and the parties will be able to establish the public key.

4.3 XTR

The XTR implementation will use more of the same methods as described in the above section. The advantage will be the size of the parameters and

the ease of generation of these parameters while maintaining an acceptable level of security.

4.4 The Math Library

The following is a list of the known methods which will be used for the efficient implementation of the previously described algorithms.

1. Basic arithmetic: add, subtract, multiply, divide, bit manipulation
2. Pseudorandom number generator
3. Prime numbers: primality tests, random prime generation
4. Modular arithmetic: GCD, modular inverse, modular exponentiation

It is a well known fact that integer division as would normally be carried out by a person, reaches levels of unacceptable performance. There are known algorithms which reduce the complexity of this operation.

[4] has localized and summarized highly efficient arbitrary precision integer arithmetic algorithms for many of the above computations. I will use those implementations as a starting point for efficiency.

It has been brought to my attention by [4] that some multiple precision arithmetic may require hardware level manipulation. This could open the possibility of native C code for this implementation. It is unlikely that the target devices would have the specific hardware required for these calculations.

4.5 Test Bed

The test bed will consist of an automated means of executing the set of tasks described in the metrics portion of the deliverables list in the next section. The measurements of running time for the desktop and running time for the mobile device as appropriate will be recorded. The test bed shall be

command line driven from the desktop environment. This command line approach will allow for testing a subset of the entire metrics list.

5 Deliverables

The list of deliverables shall include:

1. Implementation of ECC, RSA and XTR algorithm in separate Java packages, bundled into a single executable jar file.
2. Metric results of the following tasks:
   (a) For RSA:
      i. 2048-bit security, parameter generation, code size
      ii. 2048-bit security, parameter generation, data size
      iii. 2048-bit security, parameter generation, desktop running time
      iv. 2048-bit security, encryption, code size
      v. 2048-bit security, encryption, data size
      vi. 2048-bit security, encryption, desktop running time
      vii. 2048-bit security, encryption, phone running time
      viii. 2048-bit security, decryption, code size
      ix. 2048-bit security, decryption, data size
      x. 2048-bit security, decryption, desktop running time
      xi. 2048-bit security, decryption, phone running time
      xii. 4096-bit security, parameter generation, code size
      xiii. 4096-bit security, parameter generation, data size
      xiv. 4096-bit security, parameter generation, desktop running time
      xv. 4096-bit security, encryption, code size
      xvi. 4096-bit security, encryption, data size
      xvii. 4096-bit security, encryption, desktop running time
      xviii. 4096-bit security, encryption, phone running time
      xix. 4096-bit security, decryption, code size
      xx. 4096-bit security, decryption, data size

xxi. 4096-bit security, decryption, desktop running time
xxii. 4096-bit security, decryption, phone running time

(b) For ECC:

i. 2048-bit security, parameter generation, code size
ii. 2048-bit security, parameter generation, data size
iii. 2048-bit security, parameter generation, desktop running time
iv. 2048-bit security, encryption, code size
v. 2048-bit security, encryption, data size
vi. 2048-bit security, encryption, desktop running time
vii. 2048-bit security, encryption, phone running time
viii. 2048-bit security, decryption, code size
ix. 2048-bit security, decryption, data size
x. 2048-bit security, decryption, desktop running time
xi. 2048-bit security, decryption, phone running time
xii. 4096-bit security, parameter generation, code size
xiii. 4096-bit security, parameter generation, data size
xiv. 4096-bit security, parameter generation, desktop running time
xv. 4096-bit security, encryption, code size
xvi. 4096-bit security, encryption, data size
xvii. 4096-bit security, encryption, desktop running time
xviii. 4096-bit security, encryption, phone running time
xix. 4096-bit security, decryption, code size
xx. 4096-bit security, decryption, data size
xxi. 4096-bit security, decryption, desktop running time
xxii. 4096-bit security, decryption, phone running time

(c) For XTR:

i. 2048-bit security, parameter generation, code size
ii. 2048-bit security, parameter generation, data size
iii. 2048-bit security, parameter generation, desktop running time
iv. 2048-bit security, encryption, code size
v. 2048-bit security, encryption, data size
vi. 2048-bit security, encryption, desktop running time

vii. 2048-bit security, encryption, phone running time
viii. 2048-bit security, decryption, code size
ix. 2048-bit security, decryption, data size
x. 2048-bit security, decryption, desktop running time
xi. 2048-bit security, decryption, phone running time
xii. 4096-bit security, parameter generation, code size
xiii. 4096-bit security, parameter generation, data size
xiv. 4096-bit security, parameter generation, desktop running time
xv. 4096-bit security, encryption, code size
xvi. 4096-bit security, encryption, data size
xvii. 4096-bit security, encryption, desktop running time
xviii. 4096-bit security, encryption, phone running time
xix. 4096-bit security, decryption, code size
xx. 4096-bit security, decryption, data size
xxi. 4096-bit security, decryption, desktop running time
xxii. 4096-bit security, decryption, phone running time

3. Design documentation for implementation and design decisions made along the way.
4. A user manual for executing the previously mentioned tests.
5. Maintenance manual for further work on this project.
6. A final recommendation of which algorithm is the most appropriate.

6 Schedule

The project can be broken up into five separate sections.

1. Research
2. Implementation
3. Testing/Metrics

4. Final Report

5. Defense

Each of the sections leaves a bit of breathing room for any unforeseen obstacles. This schedule will allow for a timely completion of all research required to meet all of the goals stated above.

Name	Start	Finish	Work
Research	**Jun 2**	**Jul 20**	**49d**
RSA	Jun 2	Jun 16	15d
ECC	Jun 14	Jul 13	30d
XTR	Jun 21	Jul 20	30d
Implementation	**Jun 21**	**Jul 20**	**30d**
Big Int Impl	Jun 21	Jun 27	7d
RSA Impl	Jun 28	Jul 1	4d
ECC Impl	Jul 5	Jul 13	9d
XTR Impl	Jul 12	Jul 20	9d
Testing/Metrics	**Jun 30**	**Jul 23**	**24d**
RSA Tests	Jun 30	Jul 3	4d
ECC Tests	Jul 12	Jul 16	5d
XTR Tests	Jul 19	Jul 23	5d
Final Report	Jul 8	Sep 10	65d
Defense	Sep 20	Sep 26	7d

Figure 1: Project Plan

6.1 Research

The research section is broken down into the three different types of encryption algorithms which I will be implementing. Namely RSA, ECC, and XTR. This is where the bulk of the experience of the project comes from. This is reflected in the Gantt chart as it takes the largest portion of time, if the final document is not considered.

I have allotted 15 days for the initial research of RSA, and 30 days for research on ECC and XTR. The difference in duration comes from my previous experience with RSA. I feel that this is a low risk area for me and I do not anticipate many problems.

The investigation of ECC and XTR is the significant portion of research and as such I have allotted a full 30 days of research with a few days of overlap with implementation.

6.2 Implementation

As far as the implementation goes, assuming that my research is thorough, I feel that implementation will be the smallest portion of my project. Implementing the underlying arbitrary precision integer arithmetic is going to be a difficult task. Efficiency is the key to this project and much of the work will be based on understanding the work of [4].

The time allotted for implementation of RSA is a bit generous. I do not foresee any problems in this portion as the algorithm is very straight forward. Assuming the previously mentioned arithmetic is bug free, there should not be any issues here.

Since I am unfamiliar with the algorithms of XTR and ECC, I have allotted 9 days for each of these steps of the implementation. I have allowed room for error and overlap as some of the problems for each may be similar and thus, may be solved at the same time.

6.3 Metrics Implementation

Metric implementation may begin in parallel with the completion of each of the above implementation tasks. I feel that it is ideal to have implemented the actual algorithms shortly before the metrics are developed. This will allow a certain sense of familiarity with the algorithms, thus allowing much more effective measuring of memory usage and running time.

6.4 Final Report

I feel as though the report is a large reflection of the lessons learned for this project. I shall begin this task as the previously mentioned tasks come to an end. This will allow me to reflect upon the project in an appropriate fashion, as opposed to trying to remember all of the problems and design decisions made along the way.

Doing the report in a piece-meal fashion will also allow for proper and accurate progress reports as to how far along the process I am. As soon as all of the research is done, I will be able to write the appropriate sections, and as such have an accurate measure of completion.

I expect that the report should take a full two and a half to three weeks, after implementation and testing is completed, to reach a final revision. Again, this will allow for multiple revisions.

6.5 Defense

I will aim to have the defense of the Master's Project in the third week of September. This will allow sufficient time after implementation and testing to gather all of the information pertinent to the final report. This also allows for several revisions of the final report and coordinates with the schedules of the advisors.

References

[1] G. Barwood. Elliptic curve cryptography faq v1.12. http://www.cryptman.com/elliptic.htm, December 1997. Retrieved June 3, 2004.

[2] S. M. Inc. Java 2 platform, micro edition - frequently asked questions. http://java.sun.com/j2me/faq.html, 2004. Retrieved June 18, 2004.

[3] A. K. Lenstra and E. R. Verheul. The XTR public key system. *CRYPTO 2000, Proceedings of the 20th Annual International Cryptology Conference*, 1880:1–19, 2000.

[4] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[5] L. C. Washington and W. Trappe. *Introduction to Cryptography: With Coding Theory*. Prentice Hall PTR, 2002.