Mark Weston

Professor Almstead: CSc 197

Capstone Project Proposal: A Cryptography Tool for Problem Generation

In its current state, the MTH121: Mathematical Cryptology course, like most courses, requires a lot of behind the scenes work to ensure its success. The example problems demonstrated in class, the homework problem sets, and the exam problems have all been carefully worked out beforehand. Effective teaching problems have the property of being "not too hot, not too cold, but just right." That is, they do not take an excessive amount of steps to solve, however, they are not trivially short either. As the course deals with a fair amount of modular arithmetic, especially modulo division, it is often desirable to have an answer "drop out" after a few runs of an encryption, decryption, or cryptanalysis algorithm. These qualities are currently produced by use of Excel macros and the guiding intelligence of an instructor to create appropriate problems. However, the Excel macros themselves are not especially extensible and require prior knowledge of which numbers to input in which cells in order to produce a problem. Although this setup works, there is room for improvement. My project aims to improve the problem generation needed for the MTH121 course in a way that makes it easy for another instructor to take up the course.

So, what exactly is the problem I intend to solve? Given problems from the course, I hope to research their structures and determine generic types of problems that are used in the course. By formalizing a structure to the problems and their desired properties, it should be possible to write a GUI-driven tool to generate those problems, taking some of the burden of intelligence off of the instructor. This system addresses the overall need for problem generation as well as resolving some of the inflexibility and monotony of working through an Excel driven system. I currently plan on working on a tool intended for the instructor of the course in the

context of a Java application. However, during development it may be decided that some features would be useful for students to use, such as showing the steps of a particular algorithm, or perhaps just allowing students to quickly do a computation they may have already done by a pen and paper process for many weeks. This would likely change the resulting tool into an applet based approach, where new features could be revealed or enabled as the course progressed.

Now that I have defined the problem, we will discuss some of its requirements by looking at the behavioral and functional needs of the solution, the non-functional requirements such as performance and flexibility, and the constraints on the solution. As mentioned above, there are two main functional needs that will be required. First, the problems of the course will need to be analyzed to determine if there is a certain structure to them. By structure I mean the defining characteristics that make an individual problem different from any other problem. For example, it is not enough, necessarily, to say separate the problems by crypto-system. For example, RSA problems share many of the algorithms used elsewhere in the course, such as the extended euclidean algorithm to generate modular inverses, prime number generation, generation of integers that are relatively prime to a modulus, and modular exponentiation by square-and-multiply. Looking at it this way, RSA problems can be defined by the number and types of algorithms they use and in what order and what way. Other problems could use a subset of the above algorithms, or require entirely new components such as the need for generators in the context of using the ElGamal crypto-system.

Once the structure of these problems has been defined, a second functional requirement of the project would then be to package the algorithms and processes that define the structure of a certain problem into a module that can be used for problem generation. In the simplest sense, this would be something like a black box to which the instructor can provide inputs for a desired problem. The internal system could then follow a process of using the various algorithms it

knows about to successfully produce a problem given those inputs.

If the system is appropriately flexible, it should be possible to combine small, functional pieces into a larger system to create a problem generation package. An additional requirement then, would be to produce a system that does not rely on a specific framework or set of inputs to work. The structures and implementation of the 'building-block' algorithms should be made as general as possible without sacrificing effectiveness.

Performance is not an explicit requirement, but it should be kept in mind. The problems the course usually requires for generation are meant to be done by hand, by students. As such, they do not deal with excessively large moduli. Still, this not should be a limitation of the program. Current computers efficiently generate and use RSA keys with 1024 bit moduli fairly quickly (although not fast enough for sustained, real-time encryption/decryption). This is related to flexibility in that the program should not be limited in the size or scope of the problems given to it. Likewise, it should be cleanly structured in such a way that those larger problems do not excessively scale in the amount of time it takes to generate them.

Constraints will likely be encountered as design and implementation of the project progress. Examples could include a limitation to the kinds and number of problems that can be effectively solved, as well time constraints for the number of basic algorithms that will need to be correctly implemented.

As mentioned above, the current solution for this problem is a functional, but somewhat inflexible and tedious process of using Excel macros that does not lend itself to being easily transferable to another instructor. There is an implicit use of convention and position (e.g. perform an operation in this cell, move the result to this cell) that would either need to be formalized or taught directly. This does not meant this solution is bad one. In fact, it *is* a solution and it worked beautifully when I took the course. All the problems served the needs of

the course. However, there was a cost of time and mental exercise on the instructor to create the needed problems. The solution I am proposing will be faster, more powerful, and easier to use. It is quite possible that software packages exist that could perform this exact need, but of course it will cost money. Careful (albeit humble) student work can just as easily fit certain needs, without the price tag of a professional development shop. Besides, technical support should be, at least in theory, free for life.

I intend to do as much of the work needed by the system as possible. This will give me more control over the final solution and will ensure the flexibility of the individual parts. However, this solution will require the correct implementation of many individual algorithms. The extended euclidean algorithm, the chinese remainder theorem, RSA key generation, factoring algorithms, square-and-multiply for modular exponentiation, and prime generation immediately come to mind. These alone could require a significant amount of time to implement, after which no progress will have been made towards actual problem generation. With this in mind, it may be preferable to trade some of the flexibility of an entirely self-developed codebase for the stability and correctness of methods of class libraries. As I intend to use Java for this solution, this will likely involve using some of the methods of the BigInteger class. Once these basic algorithms are in place, it will be possible to apply the research of problem structures (most likely done in the Fall term) to use them in certain orders or ways to generate a working problem. This is perhaps the most vague part of the project, perhaps because it is where the real core work lies. Normally a human would do the kind of work I am describing here, so getting a computer to do it may require some thought as to how one would procedurally go about generating a problem. If this is not possible, one can always consider brute-force, but a more elegant solution would be preferable. The exciting part for me about this project will be researching, defining, and working on a system that can generate the types of problems needed

for the course. This problem area is, as far as I know, unsolved, which makes attempting to solve it all the more challenging and interesting.

I chose Java because I am the most familiar with it out of the handful of languages I have experience in. Many of my courses here at Union have given me the knowledge to use Java in the context of data structures, algorithm implementation, and GUI development. This solution will at least require the latter two, if not all three. Using Java should enable a more successful project, both in terms of design and implementation.

Java is free and student work is free so I do not expect any extra costs for this project. Some books dealing with applied cryptography might be useful; an IEF grant should be able to cover those.

Now that we have defined the problem, discussed some of its requirements and sketched out a solution for it, are there any risks to this project? If so, should certain parts be given priority over others? Obviously, one major risk is that the program will not be complete or will not work come the end of Winter Term '06. Ignoring this possibility would be foolish. I certainly do not intend this to happen, though. Careful research in the Fall should be able to determine if the possibly lofty goals of this first solution are set too high. If so, a problem with a smaller domain, or a specific instance of the general problem could be analyzed and solved. Other than that, there are few actual risks to this project. The intended client, Professor Katherine Lesh in the Math department, already has a functional system in place that she has used to generate problems for the course. This project could crash and burn in an epic fashion, but she would remain unaffected. The goal then, would be a working system that she could use to either replace or use in conjunction with her existing system to create problems for the course more quickly and easily. With this in mind, there are few actual priorities that need to be discussed. The development will be basically linear, with each part relying on the proceeding

one. The intended order then is to research problem structures in the Fall as well as read up and start the basic algorithm implementations. The central work of creating problem generation packages will be done in the first half of Winter Term. The latter half of Winter Term will be used to create a GUI for the program and address any required written work.

Do I think this problem is feasible? I am a fairly optimistic person, so at this point I might as well admit that I think it can be done. However, as mentioned above, in the absolute worst-case scenario, a relaxed version of the general statement of problem generation, such as solving one kind of problem to produce one kind of answer, could be solved that would still be very useful. I have a fair amount of experience with Java and have taken MTH121 so, aside from private research, there is not much more that could be done to prepare me for this project. Attached is a Gantt chart that shows a sample schedule of how I expect this project to progress.

I am very pleased to be working on something that is not only interesting to me, but also serves to benefit the school in a non ephemeral way. In addition, I will be looking at a new problem that has not been solved before, which is quite appealing.

Previous research from my background paper was very useful in preparing this proposal:

[1] K.M. Pradosh, “Public key cryptography,” *Crossroads*, vol. 7, no. 1, 2000, pp. 14-22.

[2] P. DePalma et al, “Cryptography and Computer Security for Undergraduates,” 35th Technical Symposium on Computer Science Education (SIGCSE), pp. 94-95.

[3] “RSA Security – The New RSA Factoring Challenge” May 2005; http://www.rsasecurity.com/rsalabs/node.asp?id=2092

[4] R. D. Silverman, “Massively distributed computing and factoring large integers,” *Communications of the ACM* vol. 34, no. 11, Nov. 1991, pp. 95-103.

[5] E. Shade, “Ready for prime time?,” *Journal of Computing Sciences in Colleges* vol. 17, no. 3, Feb. 2002, pp. 282-289.

[6] "Sun Java API Documentation" May 2005; http://java.sun.com/j2se/1.5.0/docs/api/index.html