`Cry` – Project 2
(Software Requirements Specification): Interview

Daniel Dunning, Michael Degraw, Vu Phan

2017-02-13

Contents

1 Questionnaire

1.1 Introduction

- We are `Team Crybabies` (Daniel Dunning, Michael Degraw, Vu Phan).
- We would like know what you think about our `Cry` cryptoframework.
- This interview is part of the 2017 Spring Texas Tech University Computer Science Capstone class instruced by Dr. Sunho Lim.
- The log and summary of this interview would be accessible only by members of the class (no object-oriented pun intended).
- Your participation is entirely voluntary.
- We appreciate your help.

Question 1. Would you participate in this interview?

Question 2. May we record the audio of this interview? (The recorded interview would be accessible only by `Team Crybabies`.)

1.2 Planned Functionality

Question 3. The `Cry` cryptoframework will briefly describe whether your created cryptographic algorithm is breakable within a given time frame. Would it be helpful to have a more detailed report? If so, what other infomation should be on that report?

Question 4. `Cry` will have some built-in cryptographic algorithms, including *RSA* and *ElGamal*. What other cryptographic algorithms (such as hashing algorithms and network stream ciphers) should be included and why?

1.3 Requested Functionality

Question 5. If `Team Crybabies` were to add additional functionality to `Cry`, what one feature would be most beneficial to you and why?

Question 6. If you could change one thing about the current state of `Cry`, what would it be and why?

1.4 Conclusion

Thank you so much.

2 Log

2.1 Interviewer: Daniel Dunning

Answer 1. Yes

Answer 2. Yes

Answer 3. Yes, it would be helpful. The report should include a statement about whether the cryptographic strength being evaluated is weak, adequate, or strong. This could be accompanied by a numeric value. For example, this algorithm is a 6 on a scale of 1-10. Another feature that would be helpful would be a statement of whether the algorithm could be broken by something other than brute force. Was there a glaring weakness in the encryption method that made it so that a brute force attack is unnecessary? A last feature might be a statement about what effect key size has on the algorithm being evaluated. That is, would a slightly larger key size greatly increase the strength of the encryption?

Answer 4. AES encryption support is desired. SHA-2 (and possibly SHA-1) hash function support is desired. In addition, legacy encryption and hash function support would be helpful. Many legacy encryption algorithms and hash functions are still in use. Legacy encryption algorithms desired are RC4, RC5, and 3DES. Legacy hash functions desired are MD4, MD5, and SHA-1. RC4 and MD4 are specified since these may still be in use on old Windows systems like Windows 2000. Also, many people are still in the process of upgrading from SHA-1 to SHA-2.

Answer 5. Assuming that `Cry` finds a problem with the encryption function, the following feature would be desirable. This feature would be a determination of whether the algorithm can be quickly strengthened (e.g. just increase key size), whether a moderate amount of work is needed to strengthen the encryption (e.g. quickly swap out encryption algorithm), or a determination that a complete rewrite is needed.

Answer 6. The best way to improve `Cry` is by having one or more examples which illustrate how to use this cryptoframework. Even though the interview question did not show the command line prompts for input, make sure the question is clear with an example of the answer. The following might be an example prompt:

```
Size of key (enter a number, for example 50):
```

2.2 Interviewer: Michael Degraw

Answer 1. Yes, I will participate.

Answer 2. Yes, you may record the audio.

Answer 3. As opposed to simply outputting ``` ``yes, the algorithm can be broken in the given timeframe'' ``` or ``` ``no, it cannot'' ```, forgo the timeframe altogether and simply display how long it would take to break the algorithm.

Answer 4. It would be foolhardy to not include AES as one of the standard algorithms.

Answer 5. It would be nice to have as detailed a statistical report as possible. For instance, include the number of operations completed in the 1 minute you run the program, include the standard deviation of the data you collected. Also, a graph wouldn't necessarily be a bad thing. Basically, I would want to know that the data collected was a good sample.

Answer 6. I don't think anything needs to be taken out, just add more statistical information. I don't think a GUI would be beneficial.

2.3 Interviewer: Vu Phan

Answer 1. Yes. You may personally identify me.

name	Chris Monico, PhD
job	math professor, Texas Tech University
IT experience	`C` expert
age range	40–60
gender	male

Answer 2. Yes.

Answer 3. Yes:

- for probabilistic algorithms: probability of sucessful cryptanalysis within time limit
- usage of RAM and CPUs

Answer 4. Include:

- a symmetric cryptographic algorithm such as *AES*
- cryptographic algorithms for hashing and digital signature

Answer 5. Add a random number generator with a good source of entropy for seeds (system time is not good enough).

Answer 6. Use these libraries:

- for big integers: `GMP` (GNU Multiple Precision Arithmetic Library)
- for integer factorization: `msieve`, a mostly embarrassingly parallel implementation of `GNFS` (General Number Field Sieve)

3 Summary

(by Daniel Dunning)

All participants were ok with being interviewed as well as having audio recorded. All participants agreed that it would be helpful to have more detail on the report, as oppose to a simple measurement of time. The additional report features included a simple number scale (1-10), a specific amount of time specified by the user, possible weaknesses of the algorithm (i.e. long binary strings of 1's or 0's or possibly whether the algorithm was particularly susceptible to attacks other than brute force). In addition, one participant stated that they would like information regarding the RAM and CPU power that it took (or would take) to break the given cryptosystem. All participants unanimously agreed that other algorithms would be helpful to have as a baseline or comparison, with AES being specifically named by all. Additional algorithms were legacy encryption algorithms such as RC's and DES as well as legacy hashing algorithms such as MD's and previous SHA's. The additional features of `Cry` were different among the interviewees and added more features to consider on implementation. One popular opinion was additional features of the report. Certain baselines (calculations per minute, etc.) as well as helpful graphs were mentioned. Another additional feature proposed was to have a good random number generator within the program which uses high entropy sources for seeds (not system time like traditional RNGs). Also possible feedback on potential improvements was voiced. For instance, upon completion of a test, a user would be notified whether a small, medium, or large amount of work and changes to their system would be able to solve certain vulnerabilities found. These would hopefully provide possible solutions or, in extreme cases, suggest a total rewrite of the system. Finally, suggestions for changes were made by participants. Most were small and in addition to other suggestions, such as providing more statistical data or uses specific, useful libraries. One interesting comment was made in regards to our possibility of adding additional GUI to the framework. They said that they did not think this was necessary of helpful, which was useful information for us as developers. Another suggestion was to be sure to include samples or example of how to run the framework, including details such specifying the key length.