Lightweight Crypto in Reverse

Abstract

The aim of this thesis is to look at cryptography in a reversible computing environment. Reversibility works on the principle that no information can be destroyed during the computation, so any computation can be traced to the beginning. This feature has the implication that any residual information is either known (e.g. null-initialized memory stays zero at the end of the computation), or the encryption implementation can be potentially attacked by using the residual information to lower the searched entropy for brute-force decryption.

In an emerging classification of encryption algorithms, we consider *lightweight encryption* algorithms for several reasons: reversible computation has a direct relation to low-power applications, and also the current breed of programming languages is still rather experimental; standard practices have not evolved yet, so preferably simpler algorithms shall be considered first.

Project Description

With the ongoing boom of small devices, often related to terms like "Internet of Things" and "smart home", we see a new requirement in the area of security, particularly cryptography. These devices have too little processing power (e.g. due to small physical sizes or limitations in terms of electrical power consumption) that they cannot run a complex encryption scheme that is normally required by today's standards when, for example, browsing the web. These require what is now known as LIGHTWEIGHT CRYPTOGRAPHY - a set of hashing algorithms, ciphers and other that are optimized for as little CPU processing as possible.

An emerging area in computer science is the concept of REVERSIBLE COMPUTING, where programs can be executed in both directions of the program flow, forwards and backwards. Two languages that have existing interpreters are Janus (imperative, C-like) and RFun (functional, Haskell-like). Both languages are still experimental and have not yet established a programming culture or community and their features are still being extended. These facts may become an obstacle, and so this alone is interesting topic to explore.

However, the combination of cryptography and reversibility gives us the opportunity to look for weaknesses in the implementation of cryptographic functions. Reversibility implies that no piece of information is destroyed. Since reversible programs can be executed in both directions, this also means that no information can be created (otherwise it would be destroyed when run in the other direction). So, for example, when given a string and a key, encryption outputs an encrypted string and a key, but nothing else. This lack of additional information means there is no secondary forgotten output that could be used to weaken the encryption. Furthermore, as a nice bonus we get a decryption function just by reversing the program flow on the encryption.

Learning Objectives

At the end of the project I shall be able to:

1. write functionally correct Janus / RFun code
2. analyse difficulties in writing code in reversible languages from programmer's perspective
3. analyse potential issues when writing security-sensitive code in reversible languages
4. implement several cryptography-related functions; block ciphers and hash functions
5. assess the implementation from security perspective
6. suggest improvements in development of the reversible languages

References

[1] Alex Biryukov and Léo Perrin. Lightweight cryptography lounge, 2015.

[2] Alex Biryukov and Léo Paul Perrin. State of the art in lightweight symmetric cryptography, 2017.

[3] Michael Kirkedal Thomsen and Holger Bock Axelsen. Interpretation and programming of the reversible functional language rfun. In *Proceedings of the 27th Symposium on the Implementation and Application of Functional Programming Languages*, IFL '15, pages 8:1–8:13, New York, NY, USA, 2015. ACM.

[4] Tetsuo Yokoyama and Robert Glück. A reversible programming language and its invertible self-interpreter. In *Proceedings of the 2007 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation*, PEPM '07, pages 144–153, New York, NY, USA, 2007. ACM.