Idempotent Factorizations: A New Addition to the Cryptography Classroom

Barry Fagin
US Air Force Academy
USAFA, CO
barry.fagin@usafa.edu

ABSTRACT

While it is commonly believed RSA requires two primes p and q, that is incorrect. Infinite examples of RSA encryption moduli $n = pq$ exist with p and/or q composite that generate correct RSA keys. This can be explained in the undergraduate cryptography classroom with support from public domain technologies like the python numbthy library [4] and the Gephi graph processor [3].

KEYWORDS

computer science education, cryptography, RSA, abstract algebra

ACM Reference Format:
Barry Fagin. 2019. Idempotent Factorizations: A New Addition to the Cryptography Classroom. In *Innovation and Technology in Computer Science Education (ITiCSE '19), July 15–17, 2019, Aberdeen, Scotland, UK*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3304221.3325557

1 INTRODUCTION

The RSA cryptosystem [1] ostensibly requires two primes p, q with $pq = n$ and two integers e, d chosen such that $ed \underset{\phi(n)=(p-1)(q-1)}{\equiv} 1$, where ϕ denotes Euler's Totient function . Its security is based on the time required to factor $n = pq$ without knowing p or q, since no polynomial-time factoring algorithms are known.

Since large primes are tested probabalistically, students may ask what happens if one or both of p, q is composite. In fact, there are infinite examples of $n = pq$ with p and/or q composite where RSA continues to function correctly.

2 IDEMPOTENT FACTORIZATIONS AND CLASSROOM EXAMPLES

A factorization of n into pq is *idempotent* if $\lambda(n) \mid (p-1)(q-1)$, where λ is the Carmichael lambda function. Note p or q may be composite. The p and q of an idempotent factorization generate correctly functioning RSA keys [2].

There are infinite n with idempotent factorizations. A list of all such $n < 2^{26}$ is available at [5]. Rather surprisingly, integers exist for which all of their bipartite factorizations are idempotent. We call these integers *maximally idempotent* [2]. Examples of these are also available at [5].

ACM ISBN 978-1-4503-6301-3/19/07.
https://doi.org/10.1145/3304221.3325557

Maximally idempotent integers can be constructed by choosing a prime p, identifying all divisors a_i of $\lambda = p - 1$ such that $p_i = a_i + 1$ is prime, and constructing the *divisor graph* for λ. A divisor graph has nodes for all a_i, with edges from a_i to a_j if $\lambda/a_i \mid a_j$. A f-clique corresponds to an f-factor maximally idempotent integer [2].

Divisor graphs can be visualized with Gephi [3]. The divisor graph for $\lambda = 36$ is shown in Figure 1. It has 6 3-cliques and one 4-clique, corresponding to 7 maximally idempotent integers from 2109 to 63973. Choosing $n = 2109, p = 57, q = 37$ or $p = 111, q = 19$ will generate valid RSA keys, even though p is composite.

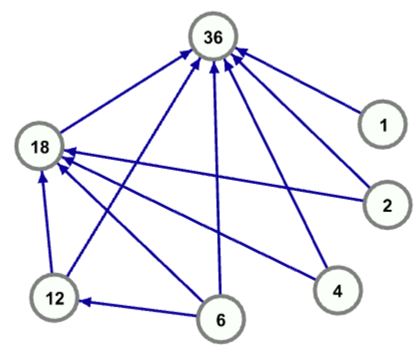

Figure 1: Divisor graph for $\lambda = 36$

3 CONCLUSIONS

Although it is common to tell students two primes are required for RSA to work, that is not strictly true. Any p, q for which $\lambda(pq) \mid (p-1)(q-1)$ will generate working RSA keys.

Knowledge of number theory is not required to present this material. For students and instructors in a more applied setting, the examples in [5] can be presented directly to demonstrate that primality is not required for RSA to function. Python code is also available at [6]. Teachers in a more theoretical setting can use the analysis here and in [2] to present these concepts in more detail.

REFERENCES

[1] Rivest, R.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystem. *Commun. ACM* **1978**, *21*, 120–126.
[2] Fagin, B. Idempotent Factorizations of Square-free Integers, *Mathematics of Computation*, under review. Contact barry.fagin@usafa.edu.
[3] https://gephi.org
[4] Campbell, R. https://github.com/Robert-Campbell-256/Number-Theory-Python/blob/master/numbthy.py, Number Theory functions in Python
[5] Fagin, B. Sequences A306330 and A306812 in *The On-Line Encyclopedia of Integer Sequences* 2019, https://oeis.org
[6] https://github.com/barryfagin/idempotency.py/archive/v1.0.zip