Constant-Time Big Numbers (for Go)

Lúcás Críostóir Meier

School of Computer and Communication Sciences

Decentralized and Distributed Systems lab (DEDIS)

BSc Semester Project

May 2021

Responsible and Supervisor
Prof. Bryan Ford
EPFL / DEDIS

Contents

1 Introduction

Explain the importance of public-key cryptography, and why big numbers are used.

Explain why timing leaks are a problem, and that big numbers are hard to implement in constant-time.

Explain what work we've done to address this, and what performance results we've gotten.

2 Background

Explain the necessary math and concerns that are relevant for our project.

2.1 Big Numbers in Cryptography

Explain how Big Numbers are used in Public Key Cryptography. Mention DSA, RSA, Elliptic curves.

2.1.1 Big Numbers in `go/crypto`

Explain how things are used in Go's standard library.

2.2 Side-Channels

Explain the concept of side-channels in cryptography, and give some of the fundamental types we're concerned with.

2.2.1 Actual Attacks

Explain how these concerns actually lead to vulnerabilities in systems.

2.2.2 Our Threat-Model

Explain the threat model we have, and what side-channels we aren't concerned about.

2.3 Vulnerabilities in `big.Int`

Explain what vulnerabilities are potentially lurking in bigInt.

2.3.1 Padding and Truncation

Explain how big.Int truncates numbers internally.

Explain potential issues with padding in cryptography.

2.3.2 Leaky Algorithms

Explain how most algorithms are potentially leaky.

2.3.3 Mitigations

Explain what mitigations are deployed in Go.

3 Implementation

Describe at a high level what we've done.

3.1 Strategies for numbers in Cryptography

Describe different strategies in place for providing numbers for Cryptography.

3.2 The `safenum` library

Describe at a high level what the library provides.

3.2.1 Handling Size

Describe how we handle sizing of numbers.

3.3 Some Basic Techniques

Describe some basic techniques for constant-time operation.

3.4 Some Algorithm Choices

Describe the algorithm choices we've used for different things.

4 Results

Describe what results we've managed to perform.

4.1 Comparison with `big.Int`

Describe the final performance results we've managed to achieve.

4.2 Comparison with `go/crypto`

Describe the benchmarks on actual code.

5 Further Work

5.1 Upstreaming to `go/crypto`

Describe our work in providing a patch for RSA, and what results we've managed to achieve.

6 Conclusion

Summarize the things we put in the introduction.

Acknowledgements

References