Efficient and Secure Outsourcing of Modular Exponentiation

Degree programme: BSc in Computer Science | Specialisation: IT-Security
Thesis advisor: Prof. Dr. Rolf Haenni
Expert: Dr. Federico Flueckiger

Many modern cryptographic techniques make extensive usage of modular exponentiation as underlying mathematical operation. Due to security requirements, they generally operate on very large numbers, making them impractical on typical devices of today, ranging from microcontrollers to smartphones. We present a thorough analysis of literature for outsourcing such calculations and provide a full implementation showcasing the technology and enabling benchmarks.

Modular Exponentiation, Practical Problems and Use Cases

Modular exponentiation, called modexp for short, is the operation x^y modulo n; or simply the remainder of an exponentiation modulo a given number.
Even though efficient algorithms for calculating modexps exist, they remain computationally expensive operations. Applications become rapidly impractical, when a large number of modexps has to be calculated, possibly in a web browser, on a smartphone or an embedded system.
The eVoting group of Bern University of Applied Sciences has developed two protocols for electronic voting which make extensive use of modexps (up to 300 000 per voter, with exponents ranging from 1024 bits upwards). While electronic voting is by far not the only application requiring such vast calculations, these two protocols are the primary motivation for our efforts in this thesis. They serve as use cases and as validation for our work.

Solving the Problem by Outsourcing

One possible solution for the calculation of modexps on limited devices is the outsourcing to one or multiple, computationally strong peers. While this may sound straightforward at first sight, problems emerge while taking a closer look.
Typically, in the cryptographic context, at least some of the involved numbers have to be kept secret and are known only to the client; they cannot simply be sent to the server for calculation. Another issue, depending on the application, might be checkability: is the result, that has been obtained from the server, really the correct answer to the outsourced modular exponentiation? There might have been an error or a server may maliciously return incorrect results. Obvious and simple solutions for these problems can be found using only basic arithmetic. In general, however, these are not optimal; either regarding the amount of required servers or concerning checkability.
To find better algorithms, we have conducted a thorough analysis of the current state of the art. Algorithms using two servers started to appear in 2005, while recent research focuses on cases with only a single server. We have unified the algorithm descriptions and present a comparative overview regarding number of required servers, secrecy of parameters and checkability.

Presenting a Working Solution

For our use cases, practical application and benchmarking of modexp outsourcing, are very important objectives. We have thus developed a fully working system for outsourcing modular exponentiations with the properties required by the use cases. Furthermore, our system allows for direct performance comparison between outsourced and locally calculated modexps, directly in any web browser.
Our system consists of three self-contained components, which cover specific aspects of outsourcing modexps. We release the full implementation under the permissive MIT open-source license. The components are:
A stand-alone server, implemented as a Java application based on JAX-RS. It offers a RESTful interface for clients to submit modexp calculations. We focused on having robust and clean code which is thoroughly tested and which can be easily deployed.
As the use cases specifically target the web browser, the client side requires an implementation in JavaScript. For this, we provide a library, aimed at developers of web applications, with only a minimum of external dependencies. The library provides methods for calculating individual or multiple modexps either locally or on remote servers, hiding complexity and offering a consistent interface.
As third and last component, we provide a demonstrator application which can be used to test the functionality of our system. It consists of a web application including a web server which can directly be started. Using it, the functionality of the client library can be tested and benchmarks against locally calculated modexps can be made with a single click.

Pascal Mainini
pascal@mainini.ch