

## Data Sheet: Welcome to MIRACL



January 2012

## Table of Contents

<b>MIRACL: Introduction</b> .....	<b>2</b>
<b>Why is MIRACL different?</b> .....	<b>2</b>
<b>Why is MIRACL better?</b> .....	<b>2</b>
<b>Where is MIRACL appropriate?</b> .....	<b>3</b>

---

## MIRACL: Introduction

Multiprecision Integer and Rational Arithmetic C++/C/Assembly Language Library (MIRACL) specializes in fast, efficient, compact implementations of number theory-based cryptographic algorithms.

### Why is MIRACL different?

Today's eCommerce requires strong cryptography. Since the amounts of money involved are typically large, the motive for robbery is proportionally large as well. That's why cryptography has to be bullet proof. Modern cryptography depends largely on what is known as a "One Way Function" - a calculation easy to perform in one direction but impossible to perform in the other.

Integer factorization is a classic example. It is easy to multiply 21649 by 513239; that is called long multiplication and we could do it by hand. However it's a much harder problem if asked what two numbers equal 1111111111 when multiplied together. This is integer factorization. If the numbers involved are much bigger, then long multiplication remains easy (if tedious) even by hand, but integer factorization becomes effectively impossible, even on the most powerful computer.

The classic RSA method which supports 95% of e-commerce today is based on this particular one-way function. The challenge is that numbers must be very big for this "one-wayness" to fully kick in. Computers don't usually handle very large (typically 1024-bit) numbers, as they normally work with much smaller 32-bit or 64-bit numbers. These are more than adequate for most run-of-the-mill applications; a typical PC these days is described as "64-bit."

### Why is MIRACL better?

MIRACL is a technology that can handle more. It contains efficient algorithms for what is known as multi-precision arithmetic: the ability to handle very large numbers with complete accuracy. It also supports cryptographic techniques built on top of other less well known one-way functions, like the "discrete logarithm" problem, and one way functions based on much more elaborate mathematical structures, such as Elliptic Curves.

The optimal algorithms that need to be implemented correctly are among the most complex in all of mathematics. Great care and expertise is required. There is no other library out there that offers the same range and depth of support for number theoretic cryptography, including the very recently discovered methods based on bilinear pairings.

Not surprisingly, these algorithms can be slow to compute. For maximum efficiency, assembly language patches are required to alleviate computational bottlenecks that arise. This requires specialized code that needs to be written for every different computer architecture.

## Where is MIRACL appropriate?

MIRACL doesn't need separate codes for different architecture. MIRACL uses a special purpose technique for automatic generation of optimal assembly language that is grounded on a bespoke macro-based system. These macros are already available for the majority of common architectures, and typically it takes CertiVox less than one day to develop macros for a new processor, should they be required. This means performance is typically boosted times three.

MIRACL supports various options, setting the gold standard for keeping code size to a minimum. While there are many libraries on the market that support larger processors, MIRACL also targets the smallest environments, where processing power, RAM and ROM are all severely constrained. In this mode of operation, all memory can be allocated exclusively from the stack so that no fragmenting of precious RAM resources is required. New configuration options further reduce the amount of program code.

MIRACL is particularly adept at methods based on Elliptic Curves, and the new paradigm of Pairing-Based Cryptography.

