Edwards Curves RNG Implementation
Project Proposal/Abstract

Keith Avery  Emilie Menard Barnard
kpavery@cs.ucsb.edu  emiliebarnard@gmail.com

April 30, 2014

RNGs in Cryptography

Random number generators (RNGs) have many applications in cryptography, including various authentication protocols and the creation of session keys, signature keys, and parameters. They are also used to generate ephemeral keys, such as those used in the Digital Signature Algorithm (DSA) and as initialization vectors (IVs) for block ciphers. Ideally, a RNG should produce values that assume all admissible values with equal probability and are independent from predecessors and successors. An attacker with maximal knowledge and unlimited computational power attempting to “crack” such an ideal RNG has no better strategy than applying a brute force attack. Ideal RNGs are simply mathematical constructs, but RNG implementations are constructed with this vision.¹

Elliptic Curve DRNGs

Truly random values cannot be computed by deterministic machines.¹ Instead, deterministic RNGs (DRNGs) or pseudorandom number generators are used. Elliptic curves are a natural choice for implementing DRNGs, as they are characterized by inherent security proofs that provide both backward and forward security. In other words, they ensure that the knowledge of subsequences of random numbers does not allow one to compute or guess preceding or succeeding values with any higher probability of success.¹ These security proofs are based on the elliptic curve Discrete Logarithm Problem (DLP)²:

Given elliptic curve points \( P \) and \( Q = [k]P \), find \( k \).

¹http://cs.ucsb.edu/~koc/cren/docs/w06/rng.pdf
²Lange, Tanja, “Analysis of pseudo-random number generators based on elliptic curves.”
**Edwards Curves**

Elliptic curve RNG implementations commonly use curves in the standard Weierstrass form \((y^2 = x^3 + ax + b)\). In 2007, Harold Edwards introduced a new elliptic curve form.\(^3\) Bernstein and Lange then presented a simplified version of the general Edwards curve form:\(^4\)

\[
x^2 + y^2 = 1 + dx^2y^2
\]

In general, addition in Edwards curves is much simpler than standard Weierstrass curves and is also remarkably symmetric.\(^1\)

**Our UDOO Implementation**

We will implement a DRNG using Edwards curves on a small-scale platform. In doing so, we will take advantage of the simplicity of Edwards curves. This will be done on the UDOO, a small (11cm by 8.5cm) embedded system. It includes two ARM CPUs as well as other interesting features.\(^5\) We will use entropy sources with the UDOO to obtain the quadratic non-residues (QNRs) required at each step of our chosen algorithm.\(^1\) Our goal is to make the algorithm as fast as possible, while considering the space limitations of the UDOO. The runtime can then be compared to other RNG implementations.

**The Algorithm**

Given the following system parameters:

- Edward curve over \(\mathcal{F}_p\) with prime \(p\) and \(d_0 = d\)
- a point on the curve \(P_0 = (x_0, y_0)\)

An input seed vector:

\(k = (k_0, k_1, \ldots, k_n)\)

And the information provided by the entropy source:

\(T_0, T_1, \ldots, T_n\)

We will use the following algorithm to compute random points \(P_1, P_2, \ldots, P_n.\)^{1}

\[
\text{for } i=0 \text{ to } i=n, \text{ step 1:}
\]
\[
\begin{align*}
\text{Set } \ (a_i, b_i) &= [k_i]P_i \\
\text{Get } T_i \text{ and compute } t_i \text{ such that } a_i^2 + b_i^2 - t_i^2 \text{ is a QNR} \\
\text{Set } d_{i+1} &= \frac{t_i^2}{a_i^2} \left( a_i^2 + b_i^2 - t_i^2 \right) \\
\text{Set } \gamma_i &= t_i^{-1} \\
\text{Assign } P_{i+1} &= (\gamma_i a_i, \gamma_i b_i)
\end{align*}
\]

---


\(^4\)Daniel J. Bernstein, Tanja Lange, “Inverted Edwards coordinates”

\(^5\)http://www.udoo.org