Lattice Basis Reduction and Cryptography
Werner Backes and Susanne Wetzel
Center for the Advancement of Secure Systems and Information Assurance (CASSIA)
http://www.stevens.edu/provost/research/securitycenter/

Introduction
What:
• Improve lattice basis reduction algorithms in terms of performance and quality.
• Develop algorithms for multi-processor, multi-core architectures.

Why:
• Lattice basis reduction provides effective means for cryptanalysis.
• Lattice-based attacks on cryptosystems and signature schemes, e.g., NTRU cryptosystem or special instances of RSA.
• Lattice based cryptographic primitives are believed to exhibit strong security even in the presence of quantum computers.

How:
• Analyze runtime behavior of lattice basis reduction algorithm to find methods to parallelize the computations.
• Develop multi-threaded algorithms for a shared memory environment using POSIX threads.

Lattice Basis Reduction
Definition: Let $n, k \in \mathbb{N}$ with $k \leq n$. A lattice $L \subset \mathbb{R}^n$ is a discrete, additive subgroup of $\mathbb{R}^n$ such that $L = \{ \sum x_i b_i | x_i \in \mathbb{Z}, i = 1, ..., k \}$ where $b_1, ..., b_k \in \mathbb{R}^n$ are linearly independent vectors. $B = (b_1, ..., b_k) \subset \mathbb{R}^n$ is a basis of the k-dimensional lattice $L$.

Lattice basis reduction: Find a basis $B' = (b'_1, ..., b'_k)$ for lattice $L(B)$ with $B' = B \cdot U$ ($U$ unimodular) and as short and orthogonal basis vectors as possible.

Example: $B_1 = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$, $B_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ are two bases of the same lattice.

Definition: The determinant of a lattice is an invariant. It is defined as $\det(L) = |\det(B^*B)|^{1/n}$. The defect of a lattice basis $B$—which allows the comparison of the quality of different lattice bases—is defined as $\delta_h(B) = \frac{\det(B^*B)}{\det(L)}$.

Goal: Determine a better lattice basis $B$ with a smaller defect.

Lattice basis reduction algorithms:
• LLL reduction. Developed by Lenstra, Lenstra and Lovasz. Practical variants introduced by Schnorr and Euchner.
• BKZ reduction. Practical variant of Korkine-Zolotarev reduction developed by Schnorr and Hoerner.

Parallel LLL
Implementation:
• Divide main loop of the LLL algorithm into parts that can be parallelized.
• Synchronize threads using locks and barriers to handle dependencies within the algorithm.
• Use control thread to prepare for the parallel computation and to balance the work load among all threads.

Experiments:
• Knapsack lattice bases $B_k$, Goldstein-Mayer random lattice basis $B_r$, and unimodular lattice bases.

$B_k = \begin{pmatrix} 2 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 1 \end{pmatrix}$, $B_r = \begin{pmatrix} p & x_1 & \cdots & x_{n-1} & x_n \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$

• Dimension $n \in [50,1500]$, maximum bit length $b \in (1000,2500)$.

Results:
• Speed-up and overhead for the 2 and 4 thread version of the multi-threaded LLL.

Stronger Reduction
Implementation:
• Uses parallel approach to find short basis vectors based on our conjecture.
• Incorporates the short vectors into a new lattice basis.
• Performs lattice basis reduction on this improved lattice basis.

Experiments:
• We are using “challenge lattices” from the TU Darmstadt to verify our conjecture and test our algorithm (work in progress).

Future Work
• Fine-tune our multi-threaded LLL for more than 4 threads.
• Improve our stronger reduction and try more than 2 iterations.
• Verify findings on stronger reduction with other types of bases.