Rounded Gaussians
Fast and Secure Constant-Time Sampling for Lattice-Based Crypto

Andreas Hülsling, Tanja Lange, Kit Smeets

26 Sep 2017
Lattice-based signatures

- Bimodal Lattice Signature Scheme (BLISS) (CRYPTO ’13 by Léo Ducas and Alain Durmus and Tancrède Lepoint and Vadim Lyubashevsky)
- Pretty short and efficient; already included in strongSwan (library for IPsec-based VPN).
- Needs noise from discrete Gaussian distribution.
- ACM-CCS 2017: To BLISS-B or not to be – Attacking strongSwan’s Implementation of Post-Quantum Signature by Pessl, Groot Bruinderink, and Yarom.
Simplified BLISS

- Work in $R = \mathbb{Z}[x]/(x^n + 1)$, $n = 2^r$, and $R_q = (\mathbb{Z}/q)[x]/(x^n + 1)$ for $q$ prime.
- Secret key $S = (s_1, s_2) = (f, 2g + 1) \in R_q^2$, $f, g$ sparse in $\{0, \pm1\}^n$.
- Public key $A = (a_1, a_2) \in R_{2q}^2$, with key equation $a_1s_1 + a_2s_2 \equiv q \mod 2q$.
- Computed as $a_q = (2g + 1)/f \mod q$ (restart if $f$ is not invertible); then $A = (2a_q, q - 2) \mod 2q$. 

- Can verify key guess for $f$ with key equation; $g$ computable.

- To sign, sample $y$ from discrete $n$-dim Gaussian $D_{\mathbb{Z}^n}$.
- $c = H(a_1, y, \text{public stuff})$ // $H$ special hash function.
- Choose a random bit $b$.
- Signature: $(z, c)$ with $z = y + (-1)^b s_1 \cdot c \mod 2q$.
- Can get $\pm s_1 = (z - y)/c \in R_q$ if we know $y$, the error vector/polynomial; ($c$ needs to be invertible).
Simplified BLISS

- Work in $R = \mathbb{Z}[x]/(x^n + 1)$, $n = 2^r$, and $R_q = (\mathbb{Z}/q)[x]/(x^n + 1)$ for $q$ prime.
- Secret key $S = (s_1, s_2) = (f, 2g + 1) \in R_q^2$, $f, g$ sparse in $\{0, \pm 1\}^n$.
- Public key $A = (a_1, a_2) \in R_q^2$, with key equation $a_1s_1 + a_2s_2 \equiv q \mod 2q$.
- Computed as $a_q = (2g + 1)/f \mod q$ (restart if $f$ is not invertible); then $A = (2a_q, q - 2) \mod 2q$.
- Can verify key guess for $f$ with key equation; $g$ computable.
- To sign, sample $y$ from discrete $n$-dim Gaussian $D_{\mathbb{Z}^n, \sigma}$.
- $c = H(a_1, y, \text{public stuff})$ // $H$ special hash function.
- Choose a random bit $b$.
- Signature: $(z, c)$ with $z = y + (-1)^bs_1 \cdot c \mod 2q$.
- Can get $\pm s_1 = (z - y)/c \in R_q$ if we know $y$, the error vector/polynomial; ($c$ needs to be invertible).
Discrete Gaussian

- Continuous Gaussian

\[ \rho_{v,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{\|s - v\|^2}{2\sigma^2} \right) \]

with center \( v \) and variance \( s \).

- Take integer values of continuous Gaussian; sample \( x \) with probability

\[ D_{v,\sigma}(x) = \frac{\rho_{v,\sigma}(x)}{\rho_\sigma(Z)} \]

where \( \rho_\sigma(Z) = \sum_{z \in Z} \rho_\sigma(z) \).

- Complicated to do in practice; relatively nice to analyze.

- Do this \( m \) times for \( m \)-dimensional discrete Gaussian.
Rounded Gaussian

- Idea: Sample $z \in \mathbb{R}$ from continuous Gaussian, round to nearest integer $x \in \mathbb{Z}$; output $x$. 

In math:

$$
R_{\mu,\sigma}(x) = \int_{x - 0.5}^{x + 0.5} \rho_{\mu,\sigma}(s) \, ds.
$$

- Easy to implement, harder to analyze.
- Hard part: needed to redo all proofs for BLISS etc.

Box-Muller sampling is $<$50 lines of code on top of Fog’s VCL (constant-time vectorized implementation of sin, cos, log, sqrt, ...).
Rounded Gaussian

▶ Idea: Sample \( z \in R \) from continuous Gaussian, round to nearest integer \( x \in Z \); output \( x \).

▶ In math:

\[
R_{\nu,\sigma}(x) = \int_{x-0.5}^{x+0.5} \rho_{\nu,\sigma}(s) \, ds.
\]

▶ Easy to implement, harder to analyze.

▶ Hard part: needed to redo all proofs for BLISS etc..

▶ Box-Muller sampling is \(<50\) lines of code on top of Fog’s VCL (constant-time vectorized implementation of \text{sin}, \text{cos}, \text{log}, \text{sqrt}, \ldots\)