

Threshold Lattice Cryptography

Sara Krehbiel; June 30, 2011

Threshold cryptography and discrete Gaussian sampling.

In threshold cryptosystems, a secret key is distributed among several parties, who must cooperate in order to sign messages or decrypt ciphertexts. Furthermore, no small enough set of corrupted parties can recover any useful information about the secret key or sabotage honest parties in signing or decrypting. Threshold cryptosystems thus provide a way to decentralize trust and increase robustness against malicious participants. A well-designed threshold cryptoscheme should have sign or decrypt algorithms that are very efficient and require as little interaction as possible.

Lattice-based cryptography has attracted recent attention for several reasons. While some NP problems are thought to be hard only in the worst case, fundamental approximation problems on lattices (such as the Shortest Vector Problem and others) are provably hard on average, assuming only their worst-case hardness [Ajt96]. This makes them of great utility in cryptographic schemes. Furthermore, lattice-based public key cryptoschemes are among the few that are believed to be secure against quantum attacks.

Several important lattice-based cryptoschemes make use of two fundamental randomized algorithms. First, a key generation algorithm is required to produce a lattice with a short “trapdoor” basis of that lattice, which serves as a secret key for the public lattice. Second, a sampling algorithm uses this trapdoor basis to produce lattice points with a Gaussian-like distribution over the lattice. This Discrete Gaussian sampling (DGS) algorithm is used as a building block in several lattice-based crypto applications, such as digital signatures, identity-based encryption, and zero knowledge proofs.

In particular, DGS can be used as the core step in the signing algorithm in a digital signature scheme. This is analogous to raising a message to a secret exponent in an RSA-style scheme. Exponentiation is nicely compatible with linear secret sharing schemes such as [Sha79] because linear operations such as Lagrange interpolation can be done non-interactively in the exponent. Consequently, several natural threshold cryptoschemes are based on RSA (see eg [Sho00]). By contrast, the known DGS algorithms (eg [GPV08, Pei10, MP11]) are much more complicated and include inherently non-linear operations, so it is hard to see how they could be securely emulated by efficient protocols. This makes the question of threshold key generation and discrete Gaussian sampling an interesting and nontrivial one.

Proposed research.

The central goal of the proposed research is to “thresholdize” the most important operations in lattice-based cryptography. To that end, we hope to develop efficient and secure

protocols for generating shares of secret trapdoors for lattices, discrete Gaussian sampling, and other fundamental lattice operations. Simple threshold signature schemes could easily be built on top of a pair of protocols for threshold key generation and DGS. While general multiparty computation results could be applied in a black box manner to existing standalone key generation and DGS algorithms to solve this problem, this solution would have high communication and round complexity and would provide no theoretical insights.

The first step towards this goal is to choose centralized key generation and Gaussian sampling algorithms upon which to base a multiparty protocol. The first DGS solution was proposed in [GPV08] and inspired by the nearest plane algorithm of [Bab86]. This algorithm is inherently sequential and therefore does not appear amenable to thresholdizing. Since intermediate results in a threshold version of the algorithm would need to be kept secret from the parties, it appears that maintaining secrecy would require many rounds of interaction. A second style of solution is given in [MP11] and builds on work in [Pei10]. In contrast to earlier trapdoor generation algorithms [Ajt99, AP09], key generation in [MP11] is simple and linear. Furthermore, the [MP11] solution to DGS pushes all the non-linearity to the preprocessing phase so that responding to online requests for Gaussian samples is fast, parallel, and linear. Our goal is to harness this aspect of the standalone algorithm to construct a threshold implementation in which all of the precomputation is done interactively and then online sample requests are answered quickly and without interaction.

Research challenges.

To implement the [Pei10, MP11] algorithms, we must first devise a scheme for secret sharing over lattices. The first and most widely used secret sharing scheme [Sha79] only handles sharing secrets in a finite field, and lattices have infinitely many points. However, since we are working with Gaussian distributions with small tails, we believe we will be able to share points as elements of sufficiently large but finite quotient groups.

The [Pei10, MP11] DGS algorithm breaks down samples into two components: a random perturbation is effectively responsible for achieving the target sample parameter, and a correction term ensures that the final sample lands in the desired coset of the lattice. The main difficulty in generating perturbations seems to be that the distribution of these components depends on the secret key, which in a threshold algorithm will be shared across the parties. However, perturbations can be generated completely offline, so we will be satisfied with an interactive solution to this task. Similarly, we hope to use the parallel structure of the [Pei10, MP11] algorithms to prepare the correction terms with an offline, interactive protocol so that the online portion of the overall protocol is non-interactive.

Thresholdizing the [Pei10, MP11] key generation algorithm should be easy, as the algorithm amounts to multiplying a public matrix by a secret matrix, which is a common and easy secret sharing operation. However, the algorithm appears to require generating shares of Gaussian samples over the integers, which seems to be a fundamental but nontrivial task. Our ultimate goal is for every step of the key generation and Gaussian sampling protocols to reduce to a protocol that produces shares of integers sampled from a Gaussian distribution. This very specific task is fundamental enough that even if we cannot reach a thoroughly satisfying solution, it may attract interest from other researchers. Any efficiency gains in this core threshold operation would immediately improve our overall protocols.

References

- [Ajt96] M. Ajtai. Generating hard instances of lattice problems. *Proc. 28th Annual ACM Symposium on the Theory of Computing*, 1996.
- [Ajt99] M. Ajtai. Generating hard instances of the short basis problem. *Proc. ICALP*, 1999.
- [AP09] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *STACS*, pages 75–86, 2009.
- [Bab86] L. Babai. On lovaszs lattice reduction and the nearest lattice point approximation. *Combinatorica*, 6(1):1–13, 1986.
- [GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. *STOC*, pages 197–206, 2008.
- [MP11] D. Micciancio and C. Peikert. Trapdoors for lattices: simpler, tighter, faster, smaller. *In preparation*, 2011.
- [Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. *CRYPTO*, pages 80–97, 2010.
- [Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.
- [Sho00] V. Shoup. Practical threshold signatures. *Eurocrypt*, 2000.