The cr.yp.to blog

2014.02.13: A subfield-logarithm attack against ideal lattices

Background: post-quantum cryptography

Imagine that an attacker is *recording all your network traffic*. Yes, yes, I know this sounds paranoid, but the U.S. government said in October 2012 [mirror] that we have to watch out for Chinese-manufactured routers providing "Chinese intelligence services access to telecommunication networks" for "economic and foreign espionage by a foreign nation-state already known to be a major perpetrator of cyber espionage".

This recorded traffic includes your public keys, the public keys of everyone you're communicating with, and all the ciphertexts encrypted to those keys. Years from now the attacker will use Shor's algorithm to factor your RSA public keys and compute discrete logarithms of your ECC public keys. The attacker will then decrypt all the

ciphertexts. Furthermore, if you're still using RSA and ECC for signatures at that point, the attacker will forge valid signatures, for example on operating-system updates.

Shor's algorithm needs a large general-purpose quantum computer. When I say "general-purpose" I mean that the computer can reliably carry out certain standard quantum operations in any specified order. Beware that one heavily advertised corner of quantum computing, namely D-Wave's quantum-annealing computer, doesn't even try to be general-purpose. D-Wave's computer can't run most quantum algorithms, in particular can't run Shor's algorithm, and has very little evidence of actually being useful. Large general-purpose quantum computers will be much more useful, notably because they can run Shor's algorithm.

It isn't easy to build a large general-purpose quantum computer, but many people publicly working on it seem to think that they'll be successful next decade. Maybe there will be serious obstacles that slow down or stop this success, but there's clearly a serious risk to RSA and ECC, and it isn't sensible risk management to pretend that the problem doesn't exist.

Fortunately, there are post-quantum alternatives to RSA and ECC. My favorites are hash-based signatures and code-based encryption, both of which were introduced in the 1970s and inspire confidence after extensive security analysis.

Lattice-based cryptography

Lattice-based cryptography is another important candidate for post-quantum public-key cryptography, and is the main topic of this blog post. The main practical reason to consider lattice-based encryption rather than code-based encryption is that lattice-based systems advertise **much smaller public keys**.

Actually, there are two types of lattice-based encryption. The first type uses special types of lattices called **ideal lattices**; the classic example is the NTRU cryptosystem introduced almost twenty years ago. The second type uses lattices without this special ideal structure. The second type is much slower and has much larger public keys, so from now on I'll consider only the first type. Sometimes people write lattice-based papers of the second type because they find unstructured lattices simpler than ideals, but then there are followup papers adding ideals to the picture (e.g., switching from "LWE" to "Ring-LWE"), because ideals are critical for the efficiency of lattice-based cryptography.

Lattice-based encryption has also attracted attention from theoreticians who talk about

its "flexibility for realizing powerful tools like fully homomorphic encryption". The latest speed reports for fully homomorphic encryption are—let me use precise technical terminology here, since I'm a big fan of careful benchmarking—ludicrously slow, but without ideal lattices they would be *utterly* ludicrously slow. Lattices can also be used for constructing more important tools such as signatures.

The same theoreticians also say that lattice-based cryptography has "strong provable security guarantees". If this is taken literally then it is false advertising. The correct advertising is that a broad class of attacks against various lattice-based cryptosystems can be converted, with limited loss of speed and success probability, into attacks against certain standard lattice problems. But this begs the question of how hard those lattice problems are. (It also doesn't distinguish lattice-based cryptography from other areas of cryptography, but that's a topic for another blog post.)

The prototypical lattice problem is the **shortest vector problem** (SVP): finding the shortest nonzero vector in a lattice L, given as input a basis for L. There are many papers attacking this problem, and some papers attacking the variants of the problem that actually appear in lattice-based cryptography. The typical attack cost twenty years ago was exponential in n log n, where n is the lattice dimension. Newer attacks take time exponential in n, obviously an improvement if n is large enough. There have been various improvements in the base of these exponentials; subexponential speedups; polynomial-factor speedups; etc.

An attack against ideal lattices

Here's the main point of this blog post: an ideal-lattice attack strategy that, unlike traditional lattice attacks, exploits the multiplicative structure of ideals.

Setup. The prototypical ideal-lattice problem is as follows. Someone has a secret short nonzero element g of a ring R, and tells us a basis for the principal ideal gR. Our task is to find g. The ring R is public, the ring of integers of a standard degree-n number field.

Practically all of the literature on lattice-based cryptography takes specifically $R = \mathbf{Z}[\zeta_N]$, the smallest subring of the complex numbers containing $\zeta_N = \exp(2 \pi i/N)$; I'll also focus on this case, so I won't discuss the general theory of number fields. The number field in this case is $\mathbf{Q}(\zeta_N)$, the smallest subfield of the complex numbers containing ζ_N; this field has degree $n = \#(\mathbf{Z}/N)^*$. There is a ring isomorphism from $\mathbf{Q}[x]/\Phi_N(x)$ to the number field taking x to ζ_N, where Φ_N is the Nth cyclotomic polynomial. The same map restricted to $\mathbf{Z}[x]/\Phi_N(x)$ is a ring isomorphism from $\mathbf{Z}[x]/\Phi_N(x)$ to R.

NTRU takes N to be prime; then n = N-1 and $\Phi_N(x) = (x^N-1)/(x-1) = 1+x+x^2+...+x^n$. PASSSign, a recent lattice-based signature scheme, also takes a prime N, specifically 433 or 577 or 769 or 1153; in each case n factors as a product of powers of 2 and 3 to allow "very fast Fourier transform algorithms". In recent papers it is very common to consider only the cases N=2, N=4, N=8, N=16, etc.; then n = N/2 and $\Phi_N(x) = x^n+1$. For example, SWIFFT takes N=128, so $\Phi_N(x) = x^{64}+1$.

There are exactly n different ring morphisms from the number field into the complex numbers. Specifically, there are exactly n different roots of Φ_N in the complex numbers (namely all powers ζ_N^j for j in $(\mathbf{Z}/N)^*$), and for each root there is a unique ring morphism from the number field into the complex numbers taking ζ_N to that root. I'll write these n different maps as $\psi_1,\psi_2,...,\psi_n$; the order doesn't matter. Dirichlet's logarithm map takes each nonzero element v of the number field to the vector Log v = $(\log|\psi_1(v)|,\log|\psi_2(v)|,...,\log|\psi_n(v)|)$, where log is the natural logarithm.

Precomputation. The attack begins as follows. Write down many small elements s of R, and keep a list of principal ideals sR that factor as products of powers of small prime ideals. When there are more factorizations than small prime ideals, solve for the small prime ideals as products of (positive or negative) powers of these principal ideals. This is simple linear algebra on the exponents in the factorizations.

(There is a tiny algebraic obstruction to this, called the *class group* of R. A nontrivial class group means that a few tiny prime ideals won't be expressible as products of powers of the principal ideals; but all the remaining small prime ideals will be expressible as products of powers of the principal ideals and those few tiny prime ideals. One can easily trace the effect of this complication through the rest of the computation.)

Note that any excess factorizations end up writing R in various ways as products of powers of principal ideals, i.e., end up writing down units u as various products of powers of elements s of R. We can compute each Log u to any desired accuracy as a corresponding sum of multiples of Log s. Dirichlet's unit theorem tells us that Log R^* is a lattice of dimension n/2-1 (since, out of the maps $\psi_1,\psi_2,...,\psi_n$ above, 0 map to the reals and n don't), and also tells us the determinant of the lattice, so it's easy to recognize when we have enough units to generate this lattice.

Main computation: the logarithm attack. If the input ideal gR factors into small prime ideals then we immediately obtain an expression for gR as a product of powers of principal ideals, and thus for a generator of gR as a product of powers of elements of R. If the input ideal gR doesn't factor into small prime ideals then we search relatively small elements e of gR until the quotient ideal $eR(gR)^{-1}$ factors into small prime ideals,

again showing us a generator of gR as a product of powers of elements of R.

We're not done yet: we cannot expect this generator to be g or anything else small. The generator is gu for some unknown u in the unit group R^*. So let's look for elements of the lattice Log R^* close to Log gu; modulo roots of unity this means looking for u in R^* with $\log|\psi_1(u)|$ close to $\log|\psi_1(gu)|$, with $\log|\psi_2(u)|$ close to $\log|\psi_2(gu)|$, etc. This is a closest-vector problem for a lattice of dimension n/2-1.

The desired generator g (and -g and other roots of unity times g) will be found in this way: by hypothesis g is short, so $|\psi_1(g)|,|\psi_2(g)|,\ldots$ are short, so $\log|\psi_1(g)|,\log|\psi_2(g)|$ are small, so $\log|\psi_1(u)|$ is close to $\log|\psi_1(gu)|$ etc. The coefficients of g are easily interpolated from good approximations to $\psi_1(g),\psi_2(g),\ldots$, which in turn are easily computed from good approximations to $\log \psi_1(g),\log \psi_2(g),\ldots$ (with any branch cuts), which in turn are easily computed from the known power-of-product expressions of u and gu. One can alternatively compute g by computing u and gu modulo a large prime, or modulo enough not-so-large primes.

Improvement: the subfield-logarithm attack. At this point the lattice dimension has dropped from the original n by a factor of 2. But we can do even better, depending on the exact choice of N.

Consider, for example, N=128, and consider the ring automorphism m of R that takes ζ_{128} to $-\zeta_{128}$; this is equivalent to the ring automorphism of $\mathbf{Z}[x]/(x^{64}+1)$ that takes x to -x. Imagine that, along with the input ideal gR, we magically know the product g m(g). This makes the last step of the logarithm attack much easier: we know gu as a power product, so we know gu m(gu) as a power product, so dividing by the magic g m(g) we know u m(u) as a power product (as usual modulo roots of unity). This gives us many new linear constraints on Log u, effectively reducing the lattice dimension from 32 to just 16.

How do we find the magic g m(g)? Notice that the product g m(g) is in the subring S = $\mathbf{Z}[\zeta_{64}]$. Starting from gR, compute gR m(gR) = gm(g)R; compute gm(g)S as the intersection of S with gm(g)R; and compute gm(g) by applying the algorithm recursively to S.

Generalization: Compute all relative norms of gR down to all proper subfields of the original number field. Use the same algorithm recursively to compute each relative norm of g. These relative norms give various linear constraints on Log u; the remaining lattice dimension depends on the exact subfield structure. A large number of subfields of small relative degree should make the attack much faster; if n has enough small prime factors

then I would expect the lattice dimension to drop to something slightly sublinear in n, producing a slightly subexponential total attack cost.

For comparison, Gentry and Szydlo introduced a fast method to compute g from the product gg^*, where g^* is the complex conjugate of g; i.e., from the relative norm of g in the maximal real subfield of the original cyclotomic field. This can be viewed as reducing the security level n of the original field to the security level n/2 of the subfield. However, from the perspective of the logarithm attack, both of these fields already have the same n/2 security level and the reduction doesn't accomplish anything. The subfield-logarithm attack reduces the security level further, breaking the n/2 barrier.

Where does this algorithm come from?

The precomputation that I've described doesn't have anything new: it's one of the standard approaches to computing class groups and unit groups, the most fundamental tasks in computational algebraic number theory. There's one paper from 2009 on fully homomorphic encryption, by Smart and Vercauteren, that mentions but dismisses class-group computations, claiming incorrectly that the "best known algorithms" run in "exponential time in the degree of the field"; see below for more about the speed of these computations.

As for the logarithm attack, I think that this approach to finding generators is reasonably well known among computational algebraic number theorists. On the other hand, after considering the possibility of computing gu, Smart and Vercauteren dismiss gu as being useless since it "will be very large"; they don't seem to realize that one can use logarithms to move from gu to the target g.

I haven't seen the subfield-logarithm attack before. I think it's the first attack that can push the lattice dimension far below n/2. I came up with the idea in 2012 and have been discussing it with various people since then.

How fast is the precomputation?

This is certainly an important question. If the precomputation is too slow to carry out then we'll never even get to the lattice steps. But I find it clear that for large n the precomputation will be vastly less important than the reduced lattice dimension.

For number fields *of fixed degree* this precomputation is well known to take time subexponential in log|D|, where D is the discriminant. The argument is easy: there are many easy-to-find number-field elements of norm at most x, where x is not much larger than |D|; these norms have chance roughly 1/y of being y-smooth, where y is

subexponential in log|D|, roughly exp(sqrt(0.5 log x log log x)); all of the steps in the class-group computation then take time polynomial in y, and thus subexponential in log|D|. The same argument appears in analyzing the number-field sieve, an integer-factorization algorithm that works in the same way and that obviously produces as a side effect the class group of the underlying number field.

As the degree grows, one expects more and more separation between the discriminant and the norms. I haven't seen a careful analysis of this effect in the class-group literature. There are various papers stating that class-group computation takes time *at most* $n^{O(n)}$ times something subexponential in log|D|; but this is an upper bound, not a lower bound. To slow down the computation so much would require norms exponential in n^2, and as far as I can see this is very easy to beat.

In mid-2013 I chatted with an expert on class-group computations, Jean-François Biasse, and learned that for some big families of number fields he had an even better result (by fancier techniques), namely complexity $\exp((\log D)^{1/3+o(1)})$ no matter how large n is. Soon afterwards he told me that for Φ_N he expects subexponential complexity by at least two different techniques. Later in the year I discussed class-group computations with Steve Donnelly from the Magma computer-algebra group, and he told me that class-group computation for $x^{64}+1$ is "not very hard".

Has the complete attack been proven? Implemented? Tested?

No, no, and no. Carefully analyzing and optimizing the complete attack will require tons of serious effort. Maybe the attack will turn out to be very slow or non-functional for some reason. But my best guess at this point is that it will work and will force serious changes in parameters for lattice-based cryptography.

It also seems that the power-of-2 cyclotomic fields used in most recent lattice papers are quite far from the safest number fields, even though these fields are constantly advertised as allowing "proofs of security". Here's what's particularly troublesome: **The structures used in the "proofs of security", such as automorphisms, are also some of the structures exploited in this attack.** This is not an isolated phenomenon: I see many examples where the single-minded pursuit of "proofs of security" adds dangerous structures to cryptographic systems and ends up compromising actual security. Any competent cryptographer will **pay attention to the cryptanalytic algorithms** and recommend that cryptographic users avoid these dangerous structures.

Whether or not the attack actually turns out to work, it's clear that at this point there

has not been adequate security evaluation of ideal lattices. There is a vast body of work on computational algebraic number theory, including problems that are essentially identical to typical problems in lattice-based cryptography; this work has been given only the briefest consideration by lattice-based cryptographers and has been dismissed for frivolous reasons.

Wait a minute. Does solving this ideal-lattice problem actually break lattice-based cryptosystems?

Sometimes yes, sometimes no. Here are four examples to consider:

- My understanding is that solving this problem directly breaks the Smart–Vercauteren system.
- To break NTRU we want to solve a similar problem having two dimensions on top of a degree-n number field. I think this can be embedded into the same problem in a degree-2n number field.
- I don't think the embedding trick will work in general for larger relative dimensions. But maybe it's possible to generalize everything from nonzero ideals to projective modules.
- Many systems actually rely on CVP rather than SVP. The usual "embedding" trick for CVP doesn't respect the ideal-lattice structure unless one increases the *relative* dimension by 1.

People haven't actually been trying to break all these different lattice-based cryptosystems; people have been trying to break a few central lattice problems, most prominently SVP. If SVP is actually secure, and if ideal-lattice SVP is at least as secure as SVP, and if the cryptosystems are at least as secure as ideal-lattice SVP, then the cryptosystems are secure; but if ideal-lattice SVP *isn't* actually so secure then this logic collapses and we're left with cryptosystems whose security hasn't actually been evaluated.

Is there a defense against this attack?

Here's a concrete suggestion, which I'll call NTRU Prime, for eliminating the structures that I find worrisome in existing ideal-lattice-based encryption systems. This suggestion uses a number field of *prime* degree, so that the only subfield is **Q**; and uses an irreducible polynomial x^p-x-1 with a very large Galois group, so that the number field is very far from having automorphisms. The best CVP dimension seems to be about half the degree; this is optimal for number fields without many real embeddings. (It's hard to create many real embeddings while keeping coefficients small, and if coefficients are

large then there are other problems.) This suggestion also chooses its modulus q so that $(\mathbf{Z}/q)[x]/(x^p-x-1)$ is a field; this simultaneously avoids (1) NTRU's traditional 2-adic structure and (2) the linear splittings used in most recent papers.

Standardize a prime p larger than 10. Standardize a positive integer t. Standardize a prime q larger than 72t+3 such that p and q are different and x^p-x-1 is irreducible mod q. Define $R = \mathbf{Z}[x]/(x^p-x-1)$.

Alice's public key is a random-looking element h of $(R/q)^*$. Bob sends Alice a message as follows:

- Take a random m in R with all coefficients in {-1,0,1}.
- Take a random r in R with t coefficients equal to 1, t coefficients equal to -1, and the remaining coefficients all 0.
- Send c=m+rh to Alice.
- Use a hash of (m,r) as a secret key (for, e.g., AES-GCM) to encrypt and authenticate Bob's actual message to Alice.

Alice secretly computed the public key as follows:

- Take a secret f in R with t coefficients equal to 1, t coefficients equal to -1, and the remaining coefficients all 0. Note that 1+3f is nonzero, and thus invertible, in the field R/q.
- Take a secret g in R with floor(p/3) coefficients equal to 1, floor(p/3) coefficients equal to -1, and the remaining coefficients all 0. Note that g is invertible in R/q.
- Compute the public key h=3g/(1+3f) in R/q.

Given c=m+rh, Alice recovers m and r as follows:

- Compute (1+3f)c = (1+3f)m+3rg in R/q.
- Lift to R with coefficients in {-(q-1)/2,...,(q-1)/2}.
- Reduce modulo 3 to obtain m.
- Compute r=(c-m)/h in R/q.

This works because the lifting produces exactly (1+3f)m+3rg in R: i.e., each coefficient of (1+3f)m+3rg is in {-(q-1)/2,...,(q-1)/2}. Indeed, multiplying m by f in $\mathbf{Z}[x]$ (with the usual lifts) produces a polynomial of degree at most 2p-2, with each coefficient bounded in absolute value by 2t. Each coefficient of mf in R is thus bounded in absolute value by 6t: note that reducing modulo x^p-x-1 adds the coefficient of x^{2p-2} to the coefficients of x^{p-1} and x^{p-2}, adds the coefficient of x^{2p-3} to the coefficients of x^{p-2} and x^{p-3}, etc. Each coefficient of rg is similarly bounded in absolute value by 6t. Each coefficient of

m+3fm+3rg is thus bounded in absolute value by 1+18t+18t = 36t+1, and q was chosen so that (q-1)/2 is at least ((72t+3)-1)/2 = 36t+1. Typically the coefficients of m+3fm+3rg are considerably smaller, but I prefer to avoid the mess of figuring out whether an attacker can trigger decryption failures.

Of course, I don't recommend actually using NTRU Prime unless and until it survives years of serious cryptanalytic attention, including quantitative evaluation of specific parameter choices. I could have screwed up something small, or something big, or there could be something even more dangerous about the entire concept of using ideal lattices in cryptography.

Version: This is version 2014.02.13 of the 20140213-ideal.html web page.