mike's blog

Joshua fit the battle of Jericho

After one week of Trump, it is clearer than ever that Americans must stand against the incoming administration, and for what is right. But take heart: God's justice is so inevitable that, as long as we are standing with Him and against injustice, it is as though we have already won.

— Mike Hamburg, January 29, 2017, 11:54 AM

it's been forever

Wow, it's been a long time since I updated. Let's see, what's new...

Life is mostly the same as ever. I'm still working at Cryptography Research. I'm still living on San Francisco's Potrero Hill. I'm still editing this page in a text editor. A few new papers up on the site today.

My side project these days is Ed448-Goldilocks, particularly the Decaf branch. Will I get it to 1.0 this summer? Time will tell.

Hopefully between biking around the city, fun with Lydia and some casual rock climbing, I can get in better shape. Meanwhile, life goes on.

— Mike Hamburg, July 3, 2015, 4:30 PM

faster 𝔽_q arithmetic

I've been working on a crypto math library, which I'm intending to post (in alpha form) and whitepaperize before too long, but in the meantime I'm going to mention some of the tricks I've found for math over $\mathbb{F}_q$. These tricks are probably known already, so if they're yours, email me and I'll cite you. I'm mainly comparing my results to MPFQ and MIRACL. In my limited testing, I was unable to coax competitive results out of MIRACL, but I probably have the compilation flags wrong, so I've been comparing more to MPFQ.

multiplication: Techniques for Montgomery multiplication have been studied extensively; I relied most on the work of Koç, Acar and Kaliski. In their terminology, I found a variant of FIPS (Finely Integrated Product Scanning) to be fastest on x86-64 hardware. Product scanning — focusing on a given place-value in the product, instead of in one of the operands — avoids juggling carries, which is painful on x86 architectures. Furthermore, at least in theory it plays nicely with the deeply pipelined multiplier on some Intel chips, because it does not block results using low-order bits on those using high-order bits. The result is a multiplier which is around twice as fast as MPFQ on my Sandy Bridge laptop, costing just over 100 cycles for a 256-bit Montgomery multiply (includes function call latency, but the timings are confused some by TurboBoost).

I conjecture, but have not yet verified, that an operand-scanning mechanism like FIOS would be faster on ARM. This is because carries can be handled with the UMAAL (unsigned multiply double accumulate long) instruction, which lets you use an entire word as a carry flag. Almost the entire multiplication can be done using UMAAL, without the need for any other form of carry tracking.

reduction: It has been noted before (eg by Hachez and Quisquater) that the final subtraction of a Montgomery multiply is not strictly necessary. This is, I think, more true than most cryptographers realize. Montgomery multiplication on numbers a and b computes $(ab+xp)/2^w$, where w is the size of the multiply in bits and x is some fudge factor less than 2^w. This means that if a and b are both at most $2^{w/2}\sqrt{p}$ then the result will be at most $2p$. Thus if p is sufficiently less than 2^w and the algorithm has all its addition and subtraction chains broken up by multiplications, the operands will remain bounded. Note that this makes reductions after additions and subtractions unnecessary as well, which is important as constant-time reductions are expensive. The results will not of course be fully reduced, but they will be correct modulo p. On a 64-bit machine this works very well for p near 2^{160} or 2^{224} where w is 192 or 256, respectively. Signed wide multiplication is expensive, so negative numbers can be dealt with by adding a suitable multiple of p before multiplication.

Something more might be gained by lazy reduction, i.e. by accumulating several products and then reducing mod p. However, except in special cases, this disrupts integrated reduction algorithms (especially FIPS). I have yet not tested performance with lazy reduction, so there may be significant gains to be had, especially on ARM where FIPS is probably unprofitable, or in quadratic extension multiplies where special cases can occur.

extension fields: For quadratic extension fields, complex squaring and Karatsuba multiplication seem to work best, as previously reported by Devegili, hÉigeartaigh, Scott and Dahab. However, there is another optimization opportunity for cubic-over-quadratic fields. To construct the quadratic field, adjoin $i=\sqrt{-1}$ instead of $\sqrt{-2}$ (this makes some operations slightly faster anyway). Then to multiply over the cubic field, use Toom-Cook: not on $(0,\pm1,2,\infty)$ but on $(0,\pm1,\pm i)$. The result looks like a Fourier multiplication algorithm, complete with butterfly gates. By my count this algorithm uses 27 add/subs instead of 35 add/subs for standard Toom-Cook-3 and 15 for Karatsuba-3, so Toom-Cook trades 12 adds for a multiply instead of 20. This trick may even outperform Chung-Hasan for squaring because it uses 5 squarings and no multiplies, and the subfield's "complex" squaring algorithm is much faster than multiplication. What's more, when using the above reduction trick, the structure of this "Toom-Cook-3i" algorithm allows us to make all the inputs to the subfield multiplication algorithm positive by adding a single bias on the subfield term of each operand (though this doesn't work quite as well as we'd like because Karatsuba doubles the bias).

The above algorithm computes 4 times the product instead of 6 times, so the actual product can be recovered more quickly with two divisions by 4. What's more, as mentioned by DhÉSB such divisions may not be necessary. In the case of pairings, the extra factor will be canceled by the final exponentiation, but even when we are not computing pairings, an extra factor can be compensated for by dividing all cubic-extension elements by 4. This trick works exactly the same way as compensating for Montgomery multiplication's division by 2^w. It does not affect multiplication by subfield elements, but it does affect addition, subtraction and conversion of subfield elements.

Similar algorithms may be used in other cases. Multiplication in quartic-over-quadratic fields can use a 6th root of unity (if it is not in the base field), and sextic-over-quartic

can use a 5th root of unity. It may also be practical to use a direct sextic-over-quadratic extension, speeding up Toom-Cook simply taking advantage of the additional low-norm elements in the subfield.

I've also tried this trick (on paper) with cube roots of unity, the golden ratio and $\sqrt{\pm 2}$, and so far $\sqrt{-1}$ is best except in the quartic-over-quadratic case.

I hope to soon finish implementing elliptic-curve arithmetic and pairings using these techniques. Then we'll see if I can get a significant speedup over the current state of the art.

— Mike Hamburg, August 2, 2011, 2:50 AM

update

It's about time I updated this blog, so here goes. I'm interning at Google this summer, and will soon leave and enter the 5th year of my PhD. Google is nice, but the code I'm working on is frustrating and messy. And yes, the free food is great. I also like that it's (difficult) biking distance from my house, so when I'm not feeling lazy I can have a nice morning and evening bike ride.

It seems like everyone is getting married. My roommate Eric is engaged, and there are two other weddings coming up in September, including my grandfather! (Next on my to-do list: figure out travel plans for those weddings.) I guess it's just that time of life, though I feel a little sad to be left out of the party (and for a while, too, since I don't have a girlfriend right now). But good things come to those who wait...

My health hasn't been great this summer, but hopefully it's just a passing phase and I'll be all back to normal this fall.

A few months ago, Emily, Dan and I released version 0.8 of the Stanford Javascript Crypto Library, the smallest, cleanest, fastest, awesomest Javascript crypto library out there. On this weekend's to do list: finish importing Emily's elliptic curve crypto code. The trickiest thing about this, and one which I probably didn't get quite right, is defense against timing attacks. On the one hand, Javascript is a hard environment to defend against this attack on, and also an environment that you don't expect to fall under attack this way. But on the other hand, it seems silly not to offer any protection. So I've made some effort to protect SJCL, but I expect that it's not perfectly tight. I'm not sure this is the right trade-off: the code might be faster or simpler without these protections.

I've been playing Starcraft II since the mid beta. It's interesting to see how reviews tend to go wild in one direction or the other. To me, the game seems pretty good, not especially original but a worthy sequel of Starcraft I. The multiplayer is a lot of fun. The campaign is OK... not great but not terrible. I suppose it's annoying that the campaign is only Terran (with a little Protoss sprinkled in), but it does allow them to go into depth with Terran-specific campaign mechanics. Also, I'm sure that almost everyone who's considering buying the game knows about the campaign by now.

Similarly, I'm sure everyone knows that it's a trilogy, so that Blizzard is planning to sell you another game next year. This is interesting psychologically: would you be happier if the same game weren't part of a trilogy? (I'm sure there are legitimate reasons, such as the 3-part campaign or the fact that your friends will upgrade.)

The DRM has the usually annoyance/convenience tradeoff: you can just download and play on any machine, but you can't play offline multiplayer, and you can't create multiple instances of the campaign. What's more, you can't make a learning account. I'd kind of like to play Zerg in a low league and Terran or Protoss in a higher one, but as it is I've postponed learning Zerg, which is sad because this way I can't go random. Oh well.

— Mike Hamburg, August 21, 2010, 11:20 AM

aes with vector permutations online

My quals paper was on making AES implementations more secure by using vector permutations to compute the S-box. Only recently did I get around to cleaning up and releasing the code. It's hosted at Stanford's crypto site for ~~export control~~ stupid reasons. It's in the public domain, so use it for whatever. Except not anything too important, since it's version 0.5 and needs more testing.

In completely unrelated news, I misbooked my Thanksgiving flight. Delta wants $180-$200 to change it... might as well just get another ticket. This is probably the point, though you'd expect them to make it more worthwhile to fly Delta again instead of rebooking on some other carrier.

— Mike Hamburg, November 10, 2009, 6:47 PM

back online

It took a while, but catameringue is back online, now hosted from the Riddlair. Thanks, Robert and Shaddin!

CRYPTO 2009 is going well, though I didn't manage to make slides in time for the rump session. I've also been playing too much Civ IV and doing too little mingling between sessions. But I've gotten to see many friends and acquaintances that I haven't seen in a long time.

— Mike Hamburg, August 18, 2009, 7:05 PM

rebuilt server

Long post because I like to geek out over this sort of thing.

I'm in the process of rebuilding the shiftleft.org server, meringue, into a smaller, more power-efficient machine, with the goal of moving it out of my apartment into someone's closet or something. I'm using an M350 case and a pair of notebook hard drives for storage. Because of the way the dual hard-drive mounts look, and because I like dumb puns, I'm calling it “catameringue”.

I've had a few problems so far. First up, I got a 24-pin PSU by mistake instead of the 20-pin version, so the whole thing is currently hooked up to a ginormous ATX PSU. I botched the install (turns out the Hardy /etc/passwd and my earlier Ubuntu /etc/passwd have different UIDs for certain daemons...) and it took me several hours to figure out what was wrong and correct the problem. So that's why the server was down for an evening. What's more, there are lignering stability issues. They might be caused by powernowd, which doesn't meaningfully reduce power consumption anyway,