From: Perlner, Ray A. (Fed)
To: Cooper, David (Fed); Liu, Yi-Kai (Fed)
Cc: internal-pqc
Subject: RE: Including memory access costs in gate count metric
Date: Wednesday, June 24, 2020 2:17:15 PM

I'm not sure I know a good reference to cite off the top of my head, but here's a quick summary of the various models under discussion

1) The basic gate model:

This model of computation treats an algorithm as a sequence of simple operations where each operation takes one or two input bits and produces an output bit that is a simple deterministic function of the input bits (e.g. the AND or XOR of the inputs). Each operation is called a gate. The cost is then reckoned as the total number of gates that need to be performed to get from the algorithm's input (e.g. a Kyber public key) to its output (e.g. a Kyber private key). Additionally, we can measure the width and depth of the computation. The width is the maximum number of bits that need to be stored at any given time and the depth is the number of gates that need to be performed in series. Apart from any hard limits that we may place on the computation's width, this metric effectively ignores the cost of memory access, because it allows a gate to be performed between any input bits or intermediate bits that have already been computed, regardless of how far apart they might be stored.

We've generally used this metric to evaluate the cost of classical attacks, because it is relatively easy to compute for known attacks, and for most attacks (other than lattice reduction via sieving), the attack can be set up in such a way that almost all the memory access is done locally. We have not explicitly endorsed any specific strategy for getting a lower bound on the amount by which the gate model underestimates the true cost of lattice sieving, but in my first response to Kyber, I did give an outline for such an approach which I would find acceptable, if the Kyber team or someone else made a credible effort to optimize known attacks taking this lower bound on the cost of memory into account. So effectively our requirement for level 1 at this point is "show that all the known attacks on your scheme require at least as many gates as key search on AES, or give us a reason why we shouldn't care (e.g. by showing that a conservative estimate of the memory costs implies that any attack will cost more than AES key search in practice)."

2) Dan's favored modification of the basic gate model (to account for communication costs) -- I know this mostly from previous interactions with him.

This model of computation is the same as the basic gate model, except, at all times during the computation all bits that will be used later in the computation must be stored at a specific, stationary location, and the inputs and output of any gate must be next to each other in the array. This at a minmum would make the cost of a random access memory query scale as the square root of the size of the memory (both in terms of number of gates and circuit depth). I think this is likely to overestimate the cost of memory as implemented by both current and future computing technology.

3) Dan's favored modification of the basic gate model (but 3d)

Exactly what is says on the tin. This is exactly the same as Dan's model except we replace 2 dimensions with 3 dimensions, and the cost of random access memory would then scale as the cube root of the size of the memory (both in number of gates and depth). This is less likely to underestimate the cost of memory access, but I can't confidently say it isn't still an underestimate, since the only way in this model to move a bit of memory around is to perform gates every few nanometers, which I doubt is going to be as efficient as a fiber optic link, with any plausible technology for implementing arbitrary gates. In the other direction, Dan doesn't like this metric because the amount of heat a system can dissipate scales as its surface area rather than its volume, and even when this issue is resolved, cooling a 3 dimensional system efficiently requires convective rather than radiative cooling, which might not scale as well.

4)Yi Kai's suggested model (as I understand it)
Same as the 3d version of Dan's gate model, but relax the requirement that inputs and outputs of gates must be next to each other, e.g. by allowing them to be something like 2^20 times as far apart as nearest neighbors in the 3d array.

5) The model I suggested in the forum
Attacks must be implemented mostly locally (either in 2 or 3 dimensions, not sure which I like better), but they will have access to magic memory boxes that have a volume proportional to their storage capacity, and every time you read or write from them, you additively increase the effective number of gates in the system by the log of the storage capacity, and you increase the effective depth of any serial computation including the read or write instruction by the cube root of the memory's storage capacity.

-----Original Message-----
From: David A. Cooper <david.cooper@nist.gov>
Sent: Wednesday, June 24, 2020 12:05 PM
To: Liu, Yi-Kai (Fed) <yi-kai.liu@nist.gov>
Cc: internal-pqc <internal-pqc@nist.gov>
Subject: Including memory access costs in gate count metric

I'm at a bit of a loss here, since I don't understand what the "gate count" metric is. Could someone point me to a brief explanation of this?

In one of his messages to the mail list Ray noted the following:
> There is also at least a logarithmic gate cost in the literal sense
> [for memory access], since you need logarithmically many branch points
> to get data to and from the right memory address.
For this reason, I don't think we should assume that each gate can access some huge amount of memory for free. Even operations that can be performed in a single clock cycle require many gate operations. Using numbers from https://gcc02.safelinks.protection.outlook.com/?url=https%3A%2F%2Fbench.cr.yp.to%2Fresults-stream.html&data=02%7C01%7Cray.perlner%40nist.gov%7C15ee39adc5cc42dcd14e08d818586df7%7C2ab5d82fd8fa4797a93e054655c61dec%7C1%7C0%7C637286115364507907&sdata=KHK4m7jA%2B8%2BCAF%2FoEAC07hgxiYsa3Z%2B4QOL%2BqCcGUsl%3D&reserved=0 I calculated that one processor can perform an AES 128 block operation in (at most)
48 cycles (about 15.5 ns); something that our CFP seems to says requires
2^15 gate operations.

Dan did not dispute Ray's claim about requiring at least a logarithmic gate count to access memory, but he seems to still be insisting that the classical gate metric requires treating memory accesses as free, regardless of how large the memory is. As Ray noted, it is theoretically impossible to construct a computer that can read from memory without doing some work (gate operations). So, the question should be how to model the cost of memory access, not whether it is acceptable to do so.

As for how one might estimate memory access costs, I don't know anything about the algorithms being discussed, but I'd want to know more than the total amount of memory required. There is a big difference between an attack involving a huge number of processors each accessing a small amount of memory and an attack which requires every processor to access every part of the memory.

Given that the more locally cached the data is the faster the access, when I think about the cost of memory access I think about entropy. If an algorithm requires 2^89 memory and each memory access could be equally likely to read from any memory address, then the cost of each read will be higher than if memory access patterns are more predictable so that some sort of caching strategy could be used. It seems that for each memory access the cost should be proportional to the uncertainty of what the memory address will be.

On 6/23/20 4:57 PM, Liu, Yi-Kai (Fed) wrote:
> Hey,
>
> For what it is worth, I would be open to replacing the naive "gate count" metric with a more elaborate "gate count" metric that assumes the computer's memory has a 3-D layout, and each gate can access a spherical region that contains at most M bits of memory, where M is some large number like 2^60, 2^80 or 2^100.
>
> The important part of this definition is the parameter M, which basically says that we're giving everyone M bits of memory for free, and only after that do we start keeping track of the cost of memory. I think this is in agreement with our intuition that the cost of memory is not a strong obstacle to cryptanalysis, until you start using truly enormous quantities of it.
>
> The interesting thing is, I think you can actually try to justify the choice of M, using fairly simple arguments about latency, i.e., the time needed to access a large memory. I don't know if we all agree on the reasons *why* memory is expensive, but latency is definitely one of the reasons, and this seems to be enough to decide on a plausible value for M. Here are some arguments that suggest we should set M to be somewhere between 2^60 and 2^100:
>
> - We assume each gate executes in 1 nanosecond (i.e., the computer runs at 1 GHz clock speed). This is to be consistent with our "real world" notions of what a gate is.
>
> - We assume that signals cannot travel faster than the speed of light. I think this is uncontroversial. So each gate can access a spherical region of radius at most 30 cm, which has volume at most (4/3) * pi * (30cm)^3, which is roughly 10^5 cm^3.
>
> - We assume an upper bound on the density of memory, in bits per cubic centimeter. One possible upper bound, which might be too low, is 1 terabit (10^12 bits) per cm^3. Then each gate can access at most M bits of memory, where M = 10^12 cm^(-3) * 10^5 cm^3 = 10^17, which is pretty close to 2^57.
>
> - Another possible upper bound on memory density, which might be on the high side, is Avogadro's number of bits per cubic centimeter. Then each gate can access at most M bits of memory, where M = 6 * 10^23 cm^(-3) * 10^5 cm^3 = 6 * 10^28, which is pretty close to 2^96.
>
> Anyway, I found this useful as intuition about when the cost of memory becomes important...
>
> --Yi-Kai