**From:** Apon, Daniel C. (Fed)
**To:** (b) (6)
**Cc:** Perlner, Ray A. (Fed)
**Subject:** Re: ABC stuff
**Date:** Monday, December 2, 2019 10:24:28 AM

---

I am!

I'm doing something for the next 30 minutes or so, then I will have the rest of the day to work on Overleaf.

If you get a minute, could we chat on the phone briefly about what I should be doing? Otherwise, I will go through your long email from before and start editing accordingly

---

**From:** Daniel Smith (b) (6)
**Sent:** Monday, December 2, 2019 10:23 AM
**To:** Apon, Daniel C. (Fed) <daniel.apon@nist.gov>
**Cc:** Perlner, Ray A. (Fed) <ray.perlner@nist.gov>
**Subject:** Re: ABC stuff

Ah, I see you're there!

On Mon, Dec 2, 2019 at 10:22 AM Daniel Smith (b) (6) wrote:

> Hi, Daniel,
>
> I think that Javier invited people to an overleaf thing. I'll ask him if he forwarded it to you and to do it if he didn't.
>
> Cheers,
> Daniel
>
> On Mon, Dec 2, 2019 at 9:47 AM Apon, Daniel C. (Fed) <daniel.apon@nist.gov> wrote:
>
> > Hey Daniel (cc Ray),
> >
> > My personal email address is (b) (6)
> >
> > However, since I don't want my personal email address potentially FOIA'd, I make a concerted effort to keep work emails on my work email address.
> > If it's critical, you can send email there, but I may not check it more quickly than here anyway.. For example, this past week, I was on holiday vacation and completely off-the-grid. (In those situations, sending me a text message is the best way to contact me.)
> >
> > Anyway-- Yes, I will work all day on this.
> > Is there LaTeX I can edit? (If it's in a prior email, I'm currently going through my emails from this past week while I was on vacation and will find it shortly.)

--Daniel

---

**From:** Daniel Smith (b) (6)
**Sent:** Thursday, November 28, 2019 12:56 AM
**To:** Apon, Daniel C. (Fed) <daniel.apon@nist.gov>; Perlner, Ray A. (Fed) <ray.perlner@nist.gov>
**Subject:** ABC stuff

HI, Daniel,

Is there another email address I can reach you at? It's a bit irritating dealing with the NIST email when you're remote.

So the situation is this. We have submitted the paper since the deadline was Tuesday. We still have until next Tuesday to finish the paper and make it nice and squeaky clean.

The attack works and Dustin has some experimental evidence to back it up. There are several things we need to write up proofs for, and I think that that might be the best place for you to contribute. Here is the outline of the attack.

Recall that ABC works like this. (Any other way of expressing it is equivalent to this.) Take n variables, x_1,..,x_n. Make three matrices of random linear forms in the x's: A which is r x s, B which is s x u and C which is s x v. So a random entry in any of these matrices might look like:
a_1x_1+...+a_nx_n where the a_i's are random constants in some field, GF(2^d).

Then multiplying AB and AC we get two matrices of quadratic forms of size ru and rv, respectively. We call these E_1 and E_2. We vectorize these and apply a linear transformation T mixing them all up to get the public key P.

Encryption is just evaluating P at a vector x=(x_1,...,x_n). Decryption is more complicated.

First, apply T^-1 to get the output of the E_1 and E_2. We then parse it into E_1 and E_2, so that we have two matrices of known coefficients of size ru and rv. Next, we assume that when the correct values of x are plugged in that the matrix A has a left inverse, that is, there is a W of size s x r such that WA=I_s. So we make a matrix of variables representing W and multiply it by the known values of E_1 and E_2 like this: WE_1 and WE_2. This gives us an expression that is linear in the unknown coefficients of W. If W is the correct left inverse of A, though, then we know that the result of these products should be B and C. Since we know the linear forms in x in each entry of these matrices, we set these equal to WE_1 and WE_2 and have a system of su+sv equations in the n+rs variables x_i and w_i (the entries of W). We use Gaussian elimination to get rid of the w variables and recover enough relations on the x's to nearly solve the system. Whatever system is left over (for example, x_{13}=x_{19}+x_{20} and x_{14}=x_{19}+2x_{20}) we plug back into AB and AC to get a very small quadratic system that is easy to solve: e.g. (AB)(1,...,3, x_{19}+x_{20}, x_{19}+2x_{20},2,...,x_{19},x_{20})=E_1...

The attack attempts to find a low rank polynomial in the span of the public key. We can express any homogeneous quadratic map as a matrix (for example (x_1,x_2)->

(x_1^2+2x_1x_2) can be written:
[[1 1]
[1 0]].)
So we express all m public equations as matrices and try to find a linear combination of rank 2s. We know that such maps exist because any linear combination of the public matrices that produces a map generated by a single row of A (for example) will depend on the value of those coordinates of A. So, for example, if all of those values were zero, the map would evaluate to zero. If we find such a map, then we have found a part of T that will help us to recover an equivalent A.

How do we do the minrank attack? There are lots of ways, but the most appropriate here is linear algebra search. In these examples, typically m is about 2n, so let's pretend that is the case for a moment. We pick 2 vectors and solve for when they are both in the kernel of the same map.
\sum_{i=0}^mt_iP_ix_1=0
\sum_{i=0}^mt_iP_ix_2=0.
(These x's are representing vectors now... bad notation.) This is 2n equations in m unknowns, t_i. If we can solve for the t's then we check the solution space for a matrix of rank 2s. When we find one, we are done with the most important part of the attack. (The reason we need two vectors is that each of the above equations is actually n equations, one for each coordinate of the 0 vector. To solve linearly for m variables we need about m equations.)

Here is what we know so far. When A is square, so when r=s, the decryption failure rate is high. So it is necessary to make r>s. Now we have learned that when r>s that it makes the minrank attack above faster by about a factor of q^{r-s-1}. When there is a decryption failure, it means that there is no left inverse of A, and so in effect it is as if s is one smaller, so the attack speedup is q^{r-s}. The details are written in the overleaf document which hopefully you have access to.

There is a slight problem that m is bigger than 2n. If we need to guess an x_3 also, then that really hurts the attack. The thing is that m is barely bigger than 2n, so I propose to eliminate extra equations to get m=2n. it so happens that we have some extra degrees of freedom in the key whenever u+v is bigger than 2s. The reason is a bit difficult to write out. It has to do with the matrix with R's in it in Theorem two (equation 12) in https://eprint.iacr.org/2014/399.pdf. Instead of being 2s x 2s, the matrix in our case is 2s x (u+v). And for us u+v is bigger than 2s by at least as much as m-2n. So we can just throw away equations and still expect that the space of maps of the right form is 2s dimensional so that this matrix has a decent change of having a nontrivial right kernel.

If m-2n=(u+v)-2s, then the complexity of the reaction attack is q^{2s-r} times linear algebra stuff that amounts to about s^7. In the case that m-2n<(u+v)-2s, then the complexity of the reaction attack is q^{2s-r-1} times linear algebra stuff.

So what you can prove and write up explicitly is an analogue of Theorem 2 from https://eprint.iacr.org/2014/399.pdf, where instead of an s^2 dimensional space where the variables take their values and a 2s dimensional band space, we have n dimensional variables and a u+v dimensional band space. Because we don't want to have more than two vector equations of the form of equation (9), we throw away m-2n variables, and then (you have to argue) the expected intersection between the u+v

dimensional band space and the remaining 2n dimensional space is of dimension u+v+2n-m. When this quantity is 2s, then the proof is completely identical to that Theorem 2 and we get a probability of 1/q, but when u+v+2n-m is bigger than 2s, then there are more columns in the matrix of equation (12) than rows and there is always a right kernel, so the probability is 1.

I think that is all we will need in terms of proving something and proof righting. There may be some other things, but Ray can help figure out where you can help, too, since it's easier for him to communicate with you than me.

Cheers,
Daniel