Rene Peralta: Okay it's getting late; we are all tired, maybe… Do we still care about this topic here? Okay well I guess we have to do it. This is motivated largely by traffic on the discussion forum where there seemed to be much disagreement among yourselves about what we thought on regarding security and also about what our criteria were for evaluations of attacks. Not all the people who were more adamant on this discussion are here, some have dropped out. So if you at some point feel that we are all happy and content that we want to leave early, we can also do that. The slides, I guess I started somehow, the slides here have been produced by collectively with a lot of good input from John Kelsey and also from Ray Perlner and any flaw we'll just blame Mridul and all the good stuff is collective.

Mridul Nandi: Thank you Rene. Okay, so I will be giving a talk on NIST's view on SHA-3 security requirements and evaluation of attacks. Actually we divided into two talks, but I would like to give a combined talk so I put in the title together. Okay, so basically what I will try to do in the next say 20 or 30 minutes I will first try to give NIST's view point on SHA-3 security requirements and the related.. The next one is very much related to the first one, which is when you will come up with some attacks on some SHA-3 candidates, then how do we evaluate those attacks for the process of selection.

Okay so here is outline of the talk. The first security requirements, there are many security notions for the hash functions, definitely we are not going to consider all security notions, but definitely we will be going to consider the most studied, most important security notions, and then the cost of an attack. Attack requires several kinds of resources, those things will be studied. And also we will try to categorize all known attacks, or all possible future attacks. So based on the categorization of the attacks we can categorize the hash functions actually. We have four categories for these attacks. The first one is completely broken or it breaks completely, it wounds the hash functions, it undermines the confidence, and some attacks which are not that important. And finally we will try to give some summary, and we will try to have some questions from you, questions, answers, any kind of comments. Okay.. so let us move..

So the first one I want to give a quick view on the security definition of hash functions, most of you know the security definition, just want to make sure we are having we are thinking the same definitions so just to make sure that we know the definitions clearly. So the first one is collision attack, so it's just as simple as finding a collision pair like two different messages whose hash values are the same.

Next is a preimage attack. Okay very informally speaking, given h, find M such that the hash value of M is h. Now, one can ask how h is chosen. Is it randomly chosen or is it fixed, zero, one or any other fixed constant. Yeah those are actually under consideration.. means.. definitely if somebody can give attacks which runs for any h then definitely it's a good one, right? But if some attack is just able to find preimage for some constant zero say but not for other hash targets, it's still under consideration, it also says something bad for the hash function. Okay so this is the informal definition of preimage attack.

Second preimage attack is related to preimage attack, but here one message will be given and attacker has to find another message which is not same with the first message but the hash values are the same. Again we have the same question, right? How M will be chosen the first message the target message? We can again we can handle similarly like either we can choose randomly or if we can find the second preimage for particular message still fine.

The length extension attack is .. given the hash value of a message [M] the message is unknown, but the hash value [of M] is known, and the length [of M] is known, find a different hash value h' such that.. for some suffix z if you add the suffix to the unknown message you will .. you know what the hash value is even if you don't know what the prefix message M is. So this kind of attack can have some threat in some kind of password-based hash scheme, like when the password is hashed and with some prefix which is randomly chosen [a] randomly chosen string, in this scenario the length extension attack can be meaningful.

The next notion is eTCR, enhanced Target Collision Resistance, you probably may not find this term in the Federal Register Notice, what you'll find is that find M and then for a randomly chosen r, find M' and r' such that $H_r(M)$ equals to $H_{r'}(M')$. So what H_r is .. H_r is now a randomized hash. So you will see that most of the submi .. not many submissions have salt, and even if some submission does not have salt, there is some way to do that like RMX like randomized mixing, so there is a generic way to define randomized hash from .. from an unsalted hash function. This notion actually was introduced by Halevi and Krawczyk and it's termed as enhanced Target Collision Resistance, we found this interesting in signature scheme when we use hash then signature kind of protocols.

And finally PRF-attack. PRF attacks which try to distinguish HMAC based on a hash function from a random function. There can be different PRF constructions for different hash functions, but for the security requirement we'll be mostly interested in the HMAC based on a hash function.

Okay so there are.. I just like to highlight some best known generic attacks so that we can at least estimate what kind of security strength we should expect from a good hash function. So we should know what generic attacks are there, okay?

So for the collision, we know the classical birthday attack.. but the parallel collision search actually performs .. actually parallel collision search can behave like a classical birthday attack for certain choice of parameters, so the parallel collision search has one parameter for the number of processors, so if we choose the number of processors, you can make the parallel collision search as the classical birthday attack. So the complexity for the parallel collision search is like this. It takes the order of 2 power n by 2 computations, it needs order of one memory for each processor and this is a parallel algorithm .. so parallel algorithm does not mean that I can one can parallelize the algorithm as much as possible, so there is definitely a kind of speed up, how much speed up you can, do how much processor you can incorporate to get the linear speed-up, so the parallel collision search can have 2 power n by 4 parallelization. Okay, this is the parallel collision search complexity for the collisions.

Then the preimage attack, okay the trivial one like the trial and error attack I think is the best way to have a preimage attack which is definitely a parallel attack, it needs order of 2 power n computations and it needs order of one memory for each processor, and it is also parallelizable up to 2 power n by 2 factors. There are some time memory tradeoff attacks but if you look at the computation for the time memory you will see that it actually requires 2 power n computations. So when we see the time memory curve, something like say TM square equals to 2 power 2n, or capital N square something like that, that T actually represents the online computations, or online time actually more precisely, online time, not the offline time. So if we include the offline time, then actually you will find that the parallel attack will serve better. So if you include online time and offline time together then actually you can see that simple trial and error attack will be better than time memory tradeoff attack which we know like Hellman's time-memory tradeoff attack.

If the submission is something related to MD hash function or MD type hash function, then you will see that HMAC based on the hash function, one can distinguish if you have a collision.. so based on collision one can have a distinguishing attack. So the complexity for those HMAC based on hash function will have similar complexity as we have seen for collision. So these are the three mentioned generic attacks.

Now we are having some other security notions like eTCR, second preimage. We don't know when is generic preimage attack except the trivial one like in the order of 2 power n computation one can definitely have second preimage attack or enhanced target collision resistance, but there is no such known generic attack, so far not known. But if we consider some structure on the hash function as we mentioned for HMAC if we have Merkle-Damgard construction without any counter or without any permutation at the end like that then actually we can have the better second preimage attack as observed by Kelsey and Schneier. And Merkle-Damgard hash function has length extension attack so these are the best known generic attacks.

Now how we can compare with generic attacks. So definitely any attack requiring more computations than generic attack I think it should be ignored. But between generic attack with respect to both time and memory if you see that if you look at the collision attack and preimage attack, generic attack gives the maximum parallelization so you can reduce the computation time by the number of processors. So if you can beat a generic attack with respect to both time and memory then actually you are also beating the generic attack with respect to computations also.

So if some attack can not beat the time and memory with respect to the generic attack but if it beats in terms of computations shall we ignore those attacks? In our point of view the computation is one of the definitely one of the important parameters which actually reflects the security of the hash function or security of the construction. So if there is some attack which requires less computation than generic attack definitely it should be under consideration.

So here are the security requirements for the SHA-3 candidates. An n bit hash function expects a roughly n by 2 bit collision and PRF security for HMAC based hash, n bit security for preimage

and length extension attack, and n-k bit security for second preimage and eTCR. And if you consider the m bit truncation of the hash function so we can just replace n by m so we should expect the similar kind of security for m bit truncation. Just by replacing n by m. So this n by 2 bit or n bit security we have to or one should know what exactly it means, and it roughly means that the number of computations. So in n by 2 bit security it roughly means order of 2 power n by 2 computations and so on.

These are security requirements for SHA-3. And other security consideration like multicollision attack, you probably know the definition of multicollision attack, this is like a generalization of collision for more than two messages having the same collision value. If you see the narrow pipe design and those are actually vulnerable to multicollision attack due to Joux's multicollision attack. And the other security like as we have already observed we do not know any generic second preimage attack which requires less than 2 power n computations unless you assume some kind of structure like Merkle-Damgard hash function, then we can expect more than n-k bit security for second preimage attack.

Those attacks definitely will be viewed. Those attacks will be considered, but it does not mean that these are really threatful [threatening] attacks; but any submission which resists these attacks definitely will be viewed positively. So this is the first part of my talk. So I just left two questions like how we can measure n bit security, and the second question as I mentioned before. It means like the n by 2 bit security for collision we mentioned, but there are other security notions where we require more than n by 2 bit security, so what would be the reason for having different levels of security, like we can have one just n by 2 bit security for all the security notions, right? So this . can you help me.. Okay thank you .. so this is all about my first talk. Please hold on your questions to the end of my second talk so we can have question/answer session or discussion session after my second talk.

Now I want to move to my second talk. This is about evaluation of attacks. Okay so if we look at any attack algorithm, it has some kind of parameters like computation, time, memory, number of processors; so we should know what these parameters mean actually. The computation and time they are actually very closely related. For sequential algorithm they are actually identical. But for parallel algorithm they are not.

So here I want to time.. I want to mean the real clock time, so computation and time, these two complexities are closely related. We have two kinds of notions like offline, online. These can be meaningful in some situation, but if you do not want two different complexities then you can add up and then you can have a combined computation. As I mentioned that in sequential attack the computation is identical to time, but in parallel attack the computation is less than the time times the number of processors. And if you study all the attacks you will find that the success probability of an attack is closely related to computations. The attack can have several parameters, can tune the parameters, can change the number of time, the clock time, by increasing the number of processors, but eventually whatever they will do the attack.. the computation of the attack will remain the same.

Okay so to get to a certain amount of success probability they will have to do a number of computations that is fixed. So the number of computations actually reflects what it computes, what is its target to do, and the other parameters like parallel number of processors, how parallelizability .. those parameters actually [reflect] how the algorithm computes, not what the algorithm computes. So the algorithm can compute in parallel or can compute in sequential but not in parallel.. these are some kind of performance point of view parameters. So parallelizability, memory, these are definitely important factors for attack's performance consideration.

Now the next question is when we have several attacks for several hash functions how we can compare two attacks. It's always better to have one particular metric so one can definitely compare two attacks, there won't be any confusion about which attack is better and which attack is worse, but it is not true in our situation, in this situation. So it is very hard to compare two attack algorithms, but there are some.. something like if you know that these two attacks are really practically threatening attacks, then I think we don't need to compare these two attacks, right? So the comparison point is coming when those attacks are not practical in the sense that we can not run the algorithm and we can not get the collision pair and then we can say that okay this attack is better than this attack; otherwise what is the point of comparing two attacks which has 2 power 20 computations and another has 2 power 40 computations? We can do it in our time or in the feasible time, but definitely we need to compare when we have two attacks which are some kind of theoretical attack like which is not feasible but still that attack is doing better than much better than generic attack.

So we assume that there exists a parallel version of any attack. It seems true for mostly known cryptanalysis you will find their parallel version of the attack. So we will assume that there will exist a parallel version of an attack. That could help to compare two attacks. So unless if we get some strong evidence [against it] we can assume that.

These are the broad categorization of attacks, so the first one is completely breaks, it's really a practical threat. Some attack just wounds [the hash function], that means [it] is not a practical threat, but ok, you have designed some hash function so we should expect something from your hash function, but if your hash function fails to satisfy some security requirement, so that means it is wounded by those attacks. And there are some .. I should not say attacks I should say some kind of observations. It's not related to what security requirement is not exactly what security requirement but it corresponds to security requirement or somehow related to security requirement that can undermine our confidence. So I can say it comes as weakness of the hash function. And there are some attacks which we should not consider.

So I should first discuss what undermine confidence means, there are some attacks which are like near collision, pseudocollision or some low margin reduce round attacks, these are not actual attacks for the hash function, but still these are some observations which can be practical attack in the future or we should at least consider those attacks actually or those observations. But definitely reduced round attacks actually limit performance. So if you know that ten round is

broken then we are not going to use ten round hash function, so definitely we have to use more than ten rounds hash function so that gives some kind of limitation of performance. And there are some other unexpected properties which we should not expect from a good hash function, like some non-randomness behavior, like we can run some statistical test and it fails to satisfy the statistical test for randomness.

Then some hash function use some block cipher or some other structure, so if the hash function is block cipher then we usually expects the block cipher should be like a random permutation. So if it is not, then we should consider those observations, how that can be useful or helpful to get a real attack, or some kind of theoretical attack. Or there can be there are many submissions that are using S-box. They can have some properties on S-box how this can be used to obtain an attack. So those kinds of observations we should consider and we should study those observations.

Then there is something like flawed understanding of designers, so a designer provides some kind of proof which finds some flaws or some kind of reference which does not exist, that means the proof is not reliable. So in those cases there is some question on the submission on the security analysis of the algorithm. And there are some attacks or observations which do not directly violate the security notion, but we should still probably care about those attacks. But the guideline is that if there is some observation which requires more than 2 power n work, then definitely we should not care those observations.

But if an observation requires less than the theoretical limit, substantially below the theoretical limit, so some observations ideally should be done in say 2 power n by 2 works, but now we can do it in less than 2 power n by 2 works, then definitely we should consider those observations. Definitely we don't have any hard and first rule to measure how critical the observations are, so definitely we can pose this question to you that how we can evaluate the observations.

The next category is the wounded hash algorithms. So some attack algorithms which require less than the expected requirement as we mentioned, then we call those hash functions are wounded hash functions. For example, if some attack which beats 2 power n computations for the preimage, then we can say that it wounds the hash function, so we should consider this very carefully. Even if this attack may not be practical, may not be feasible right now, but still we should consider it. And similarly we should do for the other security requirements.

And if it is broken then definitely we are done with it there. So we should not take much time on those hash functions which are broken. But how can we define broken? If somebody gives me a collision pair of a hash function, okay definitely it is broken, right? Even if the collision pair does not have any textual English meaning representation, but still we consider that this hash function is broken, right?

Okay but there can be some other.. we have fixed some kind of table which says roughly how an attacked algorithm can be considered as broken. We have a table for computation and memory

so for every hash size we have a computation and memory bound. So if any attack requires less computation than [what's] mentioned in the table, less computation as well as less memory than mentioned in the table, then we consider that those corresponding hash functions are broken. And these numbers actually the computation number actually we found [is] very close [to] rounding off the collision bound; but the memory we estimate, we are having some simple conversion this is based on the current technology and [a near] future technology.

So we just convert one bit of memory, the cost of one bit of memory [to] roughly 2 power 20 to 2 power 40 computations, so that's why we choose 80, 100, 150, 200 [for] memory [bounds]. So this is just to make the cost of computation and cost of memory more or less similar. So these are actually considered as an analogy of AES and DES key sizes. It actually guides two things like if I have some attack if you have some attack which requires less computation less memory then okay the hash functions are broken, and the another thing is these numbers actually represent the security strength of the hash function.

When in the future suppose 2 power 100 computations are feasible, then definitely all hash functions will be under threat, so we should not be using 224 bit hash function, we should move to 256 hash function, or 384 or 512 bit hash function. So this is actually also giving some kind of time line of a hash function would be.

So the summary. At this point we evaluate attacks based on how they will affect our choice for the next round. So it means whenever we will come up with some attack how we can evaluate the attack, like: does it completely break the hash function, or does it violate NIST's security requirements, or does it undermine our confidence in the hash function, or does it require the hash function to be unacceptably slow to resist the attack like some reduced margin reduced round attack.

So this is all about my talk. So these questions we can ask or you can ask, we can try to answer. These are the following: how do we measure n bit security? What is the significance of having beyond n by 2 bit security? How do we compare two attacks? How do we evaluate observations? And any other questions that you would like to have. Thank you.

Mridul Nandi: Dan has some question.

Rene Peralta: Before you start throwing things this way, let me stress that maybe some blood was spilled in coming up with this, but nothing was etched in stone. Some of the blood is mine too I think. So there are already inputs that we have received here in previous discussions that may lead us to revise some of these things, for example the statement by Rich that clocks have plateaued is news to me and have to get back to our resident expert on such things with that as a question to him and say well that's true maybe you want to revise that table, for example since that table was predicated on things going getting a lot faster while the memory constraint not decreasing as fast. So Rich thank you for that. Okay, so shoot.

Donna Dodson: <inaudible> has a microphone.

Rene Peralta: Yeah so it's going to Dan there.

Dan Bernstein: Okay.

Rene Peralta: Oh Rich has one, oh sorry.

Rich Schroeppel: This is just a minor..

Dmitry Khovratovich: I would propose

Rich Schroeppel: Okay.

Dmitry Khovratovich: I would propose before agreement on how we measure the n-bit security.

Rene Peralta: Wait. Who is talking?

Mridul Nandi: I can't see who's talking. Can you just stand up? I can't see who's speaking.

Rene Peralta: Who is talking?

Dmitry Khovratovich: Here.

Mridul Nandi: Okay. Okay.

Rene Peralta: Ah okay, so let us point to you so that we know.. okay.

Dmitry Khovratovich: I'm here, not hiding.

Rene Peralta: Yeah yeah.

Dmitry Khovratovich: I propose before agreement on how we measure the n-bit security has some consensus on the computational model because in text <inaudible> people who do the attack have one computational model, designers have another computational model, reviewers have a third computational model stuff like these, so maybe first design a consistent model and in this model introduce one dimensional metrics.

Rene Peralta: Okay let me reply to that. That is one position that perhaps has been discussed, another position is well we care about all computer models. So you hit us with any attack under any computer model as long as you can count steps we are going to count the steps and if they are too many, meaning you know more than 2 to the n over 2 for collisions, more than 2 to the n ..oh I'm sorry, too few steps less than 2 to the n over 2 for collisions and less than 2 to the n for n

bit security we are going to look at that carefully. So restricting ourselves to one model computation may be, and if we do that perhaps we like to just simply say a random access machine and forget all this stuff about getting ourselves with all this complicated stuff.. particularly because we at this point we are not going to be comparing very minute differences between hash algorithms, we are trying to pick the set of fifteen that will contain the one that eventually should become SHA-3. But again, we will take that under advisement.

Dmitry Khovratovich: Thank you, but you may use many computation models, but the answer on the function is broken, wounded, not wounded will appear on your own website, so you have to state explicitly broken or not broken in decide which computational model you use for answering this question.

Rene Peralta: I am not sure that we'll be telling you why you did not make it to the second round in much detail, am I correct Bill? Yeah, there will be some arbitrariness and the and there will be the criteria are quite complex and they are not only related to security and performance. We can not lose track of the end goal, which is to end up with among the 15, 16 or whatever make it to the second round we want to make sure that the one that should eventually be SHA-3 should be among that group, and that means that taking many other things under consideration. Rich..you.

Rich Schroeppel: I just have a minor comment, you have no need to compare attacks.

Rene Peralta: Say that again.

Rich Schroeppel: You have no need to compare attacks.

Rene Peralta: No need to compare attacks. Yes yes, there's some of my blood there. I agree, but other people disagree strongly, so we will take that into advice. I think Dan has been waiting.

Dan Bernstein: OK. Dan Bernstein, University of Illinois Chicago. I would like to say first of all I've seen some clarification in the slides of what the rules of the game are and I think that's very useful for cryptanalysis. I think it would have been useful for the designers to found themselves subject to attacks of various types; I don't think for CubeHash I would have done anything differently, but I have seen a number of functions where if they have known the rules of the game, they would have either reduced the security margin or increase their security margin if they had known. Maybe NIST wants to let them tweak accordingly now that the rules of how to measure attacks are more clear. In any case, it's very helpful going forward to have some rules for how attacks are measured. I did have some questions about the details. You keep saying computations and steps versus memory, but what about success probability? Like an attack taking 2 to the 200 time or 2 to the 200 steps and 2 to the 200 memory, does it have to have success probability of one or can it get away with success probability of one in a thousand, is there any rule about that?

Rene Peralta: The complexity of the running time of your attack will be the running time of the algorithm divided by the probability of success.

Dan Bernstein: Okay so you're just taking you're just dividing it out and you are not taking any hard line. I mean suppose somebody has an attack that works extremely rarely very quickly.

Rene Peralta: Oh okay wait. So we are talking about two different things here. You mean if you run it again it will again fail or?

Dan Bernstein: Suppose somebody has an attack

Rene Peralta: with a probability of one half

Dan Bernstein: that finds preimages for 2 to the minus 100 of all possible outputs in time 2 to the minus 200 of what you want, then it's never going to happen, but according to what you just said about dividing, it is a problem, because that often says that you have to have a reasonable success probability.

Mridul Nandi: We can fix some success probability like half is a natural choice to have success so we can not consider definitely the attack algorithms should have some kind of parameters right? If we increase the computation then you can reduce you can increase the success probability, so we should see okay to achieve a success probability of [one] half, how much computation do you require?

Dan Bernstein: Okay that's clear enough. I had I guess one other part of this question which is for computations and steps, what do you mean? What is a step?

Mridul Nandi: Yeah basically computations and steps are more or less related. It depends on just as said like in different model. By step, I want to mean basic steps; but by computation, I want to mean in terms of hash computations. So just to make them uniform one computation corresponds to one hash computation.

Dan Bernstein: But what if an attack does not consist of hash computations?

Mridul Nandi: Sorry.

Dan Bernstein: I think most of the differential attacks the more sophisticated attacks that people come up with do not consist of hash computations; they do something other than hash computations. So what is the number of steps for those attacks? When cryptanalysts are saying I have an attack which runs in following number of steps, what are they supposed to do to count the number of steps?

Mridul Nandi: Counting the number of steps or the number of basic steps will be just will be it can not be straightforward to calculate, but I think given the attack, one can roughly estimate the number of hash function computations, even differential attack we are choosing different message pairs and we are computing hash values, hash I mean I am saying the underlying block which can be computation function, so we can just compute the number of compression function computations for the attack.

Dan Bernstein: But most attacks don't work that way. Most interesting attacks, differential attacks and so on don't do compression function evaluations; they do something more tricky where they are doing sub-compression function evaluations, they're doing table lookup…

Mridul Nandi: How would the attack be performed? Finally we have to compute M and M plus delta and you have to compute f of M and f of M plus delta to..

Dan Bernstein: But that's not the bottleneck. Most of the attacks, if you look at the SHA-1 attack that's going on, it does not consist primarily of doing SHA-1 evaluation.

Rene Peralta: Okay I think maybe an answer to that is that if by the time you're arguing whether your measure of a step should be this big or a thousand times that big your function is already wounded and we are not worried about that anymore.

Dan Bernstein: I'm not.

<applause>

Dan Bernstein: I'm sure that

Rene Peralta: Who else? Yeah Niels.

Niels Ferguson: Niels Ferguson, Microsoft. First I'll say that as far as I am concerned there is no use to have a security beyond n over 2 bits. Almost every system that uses the hash function somehow depends on collision resistance somewhere.

Man 7: Microphone.

Niels Ferguson: Sorry. I would urge to not consider things beyond n over 2 bits security. Almost any system that uses hash function at some point depends on collision resistance, and trying to go above the n over 2 bits security is academically interesting, but in practice not very relevant. My second comment would be for evaluating attacks. Traditionally, in cryptography we only looked at computational steps. That served us very well and that's a very simple measure. Adding memory is kind of interesting from if you actually have to build the machine, but it requires attackers to also start thinking about can I make a small memory version of this attack, and that's a completely separate specialty, and actually I don't think we know that much about how to

reduce memory requirement of an attack algorithm because we have very little literature about that. And I think it is dangerous if we start rejecting attacks because they use much memory, because that's kind of as assuming that someone does not find a way of reducing memory requirement. And all of complexity theory works on number of computational steps, and I think that's the measure for us to use.

Rene Peralta: Let me, again there are some internal discussion about that. I strongly agree with your position, and also I find that perhaps I have a benefit of being a little well quite ignorant of this field coming from a different side of cryptology. I find the notion that the parallelization problem the question is whether we can parallelize this attack rather than ask what is the parallel complexity of the problem; those are two very different things. And I see there's a lot of talk about the way that Pollard's rho has been parallelized, that makes lot of sense within the context of factoring. I look at that within the context of finding a collision, and I'm wondering why would I want to do that that much? Perhaps I wouldn't want to use Pollard's rho algorithm at all, I just go to this big parallel machine and do it some very easy way. So again another position is that one should not ask how do I parallelize this attack. The real question is what is the parallel complexity of this problem. And in complexity theory that's an extremely hard question to ask. And all the evidence, to answer, sorry, and all the evidence usually is of the type Yeah, after all, it could be paralle.. there was a parallel algorithm with the right complexity.

Rene Peralta: Danilo.

Danilo Gligoroski: You have given us several types of attacks on these slides. But since you are organizing this competition, you have to be aware about one another type of attack. And that type of attack is attack on the submission. Not attack on the hash function. If you have been to three days on Fast Software Encryption, you'll see the presentation of Mr. Weinmann, and he says we presented our attacks, but the submitters didn't concede that it is broken. So what would it mean if I concede that EDON-R is broken according to their attack and then today I saw your table and actually your attack is not an attack on my function. And this type of attitude is connected with also with International Association of Cryptographic Research. Dan Bernstein asked you previously how you're going to measure steps. On previous competition on e-stream there was an attack on EDON-80 where they said 2 to the 69 simple steps, wow the algorithm is broken. But then when you measure what is the complexity of the simple steps, suddenly you found out that the attack is more complex than the brute force. So this is sort of my comment to US organizers. You have to expect attacks on submissions just to persuade people to resign. Maybe it will make easy your life, you are dealing with 59, 51 submissions, but that is not completely on the line of scientific attitude. It is just forcing some of people to resign from this competition.

Rene Peralta: I can answer that, but Bill, did you want to answer that? I call the boss for that kind of questioning. We are the good guys.

Bill Burr: We haven't I mean. <Walking up the stage> I don't think at this point NIST itself has pronounced anything broken. Where people have conceded they were broken, we accepted that.

Now I guess I haven't thought in terms of people being I don't know what you were saying sort of bluffed into a concession or what. We've tried to be very careful not to say something was broken until we thought about it a long time, and carefully. And so at this point I don't believe we declare anything broken. I think it's likely that if we think when we get to the final selection, we do rule something out that otherwise looks very good because we think it is broken in some particular respect, we'll say that, okay. But remember what we are trying to do here is to pick the16 best, but that doesn't mean the remainder have anything broken in them necessarily. We are trying to pick I wouldn't say the 16 best, but the sort of best set that gives us that preserves a fair amount of options into the next round so that when we have more time to look at things. We are making this selection mainly because we think we need to focus the analysis of the community on fewer algorithms. We just don't think we are going to get a good hard analysis against really good algorithms with this many as we have got. So it does not it's not going to mean that if you are not selected in the next round, it is necessarily a bad algorithm or that we didn't have a hell of a time deciding this one instead of that one you know. So don't expect that we are going to give you some hard reasons why you weren't satisfactory. The chances are we will leave on the table algorithms that could have been perfectly good SHA-3 selections had we picked them. We are just trying to run a practical competition and make sure when we are done that the SHA-3 we wind up with is pretty darn good. We'll never know that it is the best choice, just as we'll never know that AES was necessarily the best choice the last time around.

<applause>

Adi Shamir: I would like to comment on the question that was raised how to measure what is a unit operation. So we have to remember that when we are comparing new attacks to the standard attacks, like the birthday bound attacks, we say that the complexity of the standard attack is 2 to the n over 2, measuring it in terms of how many operations of the hash function itself we are performing. And in my opinion that's only a reasonable way. So what you conclude from this is if someone demonstrates an attack which runs about 100 times faster than the evaluation of the hash function, then we will say that the cost of each one of those elementary steps was one hundredth, but when we compare attacks we have to compare in terms of complete evaluations of the hash function. I want to mention that this will not make a big difference with one exception. Almost all the functions are within the ratio of one to three or one to four in terms of their speed, so you might save one or two bits in one type of attack compared to the other. The only submission which might benefit from my definition is going to be Essence, because it has several kilo cycles per byte. So actually it will allow the attacker much longer kind of operation, ECOH, sorry ECOH, not Essence. ECOH.

Dan Bernstein: I would like to come back to the point of the memory if I could, the amount of memory used by an attack. The comment before from Rene was that it's okay if there is a little bit of ambiguity in the exact time taken by an attack, but for memory access that's not the case. For memory access it's not a small ambiguity, it's a large ambiguity. If you look at people doing super computer experiments for all sorts of computations, they take a huge amount of time for memory access. Now I would like to have something realistic, but I think it's more important for NIST to

make a decision and say here is how an attack is going to be measured, and it's going to be one array access is one step, then I think everybody can live with that. It's punishing some algorithms, it's rewarding others, but at least it would be clear. It's more important for NIST to have some consistent clear rules going forward than for NIST to try to take into account everything that people might be thinking for the next few years. NIST has to make a decision of how the rules of the game we're going to work, in particular for memory access, which some model of computation will say cost time one, and some model of computation say cost time 2 to the well 2 to the n over 2, depending on exactly how your attack works, it makes a huge difference how you count that. So I would like to hear NIST come to a decision about how much an attack costs, in particular memory access. If you just want to count lines of codes and say accessing an array costs one step, then people can work with that, that's the rules of the game. I don't think that it's going to be fair for submitters, as you're hearing before attacks on submitters. People who are facing attacks that take a lot of memory, and we heard quite a few attacks of that type, need to know whether NIST is going to take those seriously, cryptanalysts need to know whether those are going to be taken seriously.

Mridul Nandi: Yeah I can say one point in that. If I have, say 2 to the power 100 memory, my attack requires 2 power 100 memory, then definitely to set up that memory, we should have [on] the order of 2 power 100 time just to set up that memory. So the time should be at least 2 power 100 automatically, just to.. before starting the attack, the attacker should arrange the 2 power 100 memory, and to set up that 2 power 100 memory, there should be setup time, so that should be on the order of 2 power 100. So even if you even if an attack algorithm wants that I want the memory access in unit time, we should charge, or this should be reasonably to charge, 2 power 100 order of time, because of the setup cost, does it make sense?

Dan Bernstein: Well, at least that would be some rule that people can work with. Again it's more important to have some clarity. It would be nice to have some clarity a year ago. Again I think for a lot of submissions it doesn't matter, but for a lot of submissions it does. And I think the uncertainty in evaluating attacks is causing a lot more trouble for people than having some certainty, even if it is not the best rule, it would be nice to have some rule, something clear saying how we are measuring the number of steps in an attack.

Rene Peralta: Let me ask you, so essentially I think you are saying that there may be model of computation where access to memory is very expensive.

Dan Bernstein: Yeah, it's called reality, but I don't insist that NIST use that model of computation.

<laughter and some claps>

Rene Peralta: Say it again.

Dan Bernstein: The model of computation in which memory access is very expensive is called reality, but I don't insist that NIST use that model of computation.

Rene Peralta: Does that model include the 2 to the n cost access that you mention in your question reality?

Dan Bernstein: If you want to access 2 to the n memory, then speed of light on two dimensional circuit forces you to take time square root of 2 to the n.

Rene Peralta: Oh sorry, I thought you were saying that 2 to the n steps was equivalent to one memory access. So give us send us some numbers roughly how much you would charge your algorithm per memory access, and you know give us some numbers and we will take them into advice, okay.

Donna Dodson: Bart has had the microphone for quite a while.

Rene Peralta: Sorry that who?

Donna Dodson: Bart.

Rene Peralta: Bart.

Bart Preneel: I want to make the same point as Dan that I mean on the slides it only mentioned memory but not memory access. I think a good starting point, at least to get an idea for around 2 to the 50, 2 to the 60, then look at Mike Wiener's paper on full cost in general cryptology. At least he gives us an asymptotic analysis of what it may cost. I guess the problem is if you ask for numbers, they will fail, because if you read into 2 to the 32 memory that we have in our pc, then it costs one or it costs a few cycles, but if memory gets bigger, then I guess the constant grows, this is probably very hard to answer. But just ignoring memory access I think is a mistake.

Rene Peralta: I dint fully hear that, I don't think the audio is that good.. do you understand the question exactly or the comment?

Bill Burr: I think I think Bart's saying that memory access is different from the total memory.

Rene Peralta: Yes yes, in fact I am grateful. I hadn't thought of that, and Dan brought it up, and I realize what he was talking about finally, so thank you.

Niels Ferguson: I agree with the notion that memory access is the critical thing to count because that's part of the computation. I disagree with the saying that it's 2 to n over 2 due to the speed of light. We are talking about throughput not latency here. It takes the latency might be long, but the throughput can be very high, and most algorithms are parallelizable in some degree, so I don't think that the 2 to the n over 2 time is a reasonable model.

Rene Peralta: Yes, I'm hoping that the numbers that we get sent by Dan are more down to earth.

Bill Burr: I'm feeling a little bit like a judge in a court room that it's not wise to pre-commit yourself to the specifics of.. you know, so on one hand you want rules and laws and whatever so people have some guidelines about how they will be judged, whether they are likely to go to jail, or whatever; and on the other hand, no judge wants to rule in advance on a case until he has seen the case. And

<applause>

Bill Burr: I'm not real happy with saying more than rough principles, and I think that's generally true for more broadly than just this question. I want to look at the specific case. Somehow to me the numbers that we put up there as being broken seem pretty reasonable to me that if you are below those numbers, I would be pretty reluctant to go with it. It shouldn't be too hard to beat those numbers. I'm not sure that I want to make a really specific commitment even if it gives since we have well passed the point where it's easy to change your designs in time to do any good here, as far as the next selection goes. It may be that we made a mistake in not figuring this all out in advance, we just didn't know okay? And we are going to have to live with a certain amount of ambiguity at this point and a certain amount of discretion on our part, and we'll try very hard, we'll think about it the best we can, we'll listen to what you have to say, but honestly I'm not sure I see this likely to be a big controversy. I don't see a really great algorithm that looks to me like it's in real danger here you know. And if it's otherwise, it's not a strong contender anyhow, then I'm not terribly worried about the case.

Rene Peralta: So we have one minute.

Rene Peralta: Okay up there.

Man 12: I would like to have a question and a comment on the property of length extension. First I don't recall that it's a requirement for submission, but still it's listed as a requirement in the slides, in slide 6. And secondly in slide three, if you can go one backward, length extension is defined in a mistaken way, actually no hash function can be resistant against this definition.

Mridul Nandi: You are talking about the theoretical definition of length extension, right?

Man 12: Sorry?

Mridul Nandi: You are talking about the theoretical definition.

Man 12: The length extension definition on your slide, every hash function can not, can not at all, be resistant to it.

Rene Peralta: Did you notice the length

Man 12: Just a small mistake.

Rene Peralta: Are you missing the length of M and the

Man 12: It does not mention but you must not compute the beginning of M. It's a typo, but it should be corrected.

Rene Peralta: Okay we will check into that.

Rene Peralta: You are not given M and you are given the size of M as a constraint, I had the same problem.

Mridul Nandi: This is length, size of M, M and the two bar is.. represents the size of M. It's not M, it's the size, bit size, or length, or whatever the standard notion of.. So the attacker has only the hash value and the length of M, but not M.

Man 12: Sorry, you are right.

Rene Peralta: Now you made me feel really smart because I had the same problem, so thank you. Okay so oh one more.

Peter Schmidt-Nielsen: I have one thing to say, which is as to the extensibility of an attack, it seems to me that an attack that can say get through 8 rounds of a function of 16, but really looks like that you might be able to get through another 4, and maybe another 4 after that should be valued much more highly than one that can get through say 12 rounds and not being extended at all. That's a minor point, I just thought that it's worth mentioning, but that should factor into evaluation of attacks. I think Niels has a point

Rene Peralta: We will look into that, yes. So okay

Niels Ferguson: Maybe a final point

Rene Peralta: Yes last question.

Niels Ferguson: The requirements you have there are not sufficient to make the hash function safe in use for example in your hash DRBG SP 800-90. Really what everybody thinks a hash function is is a random mapping, and any deviation from that reason of expectation will lead to a security problem, so I think that's one of the requirements that we should have in mind that a hash function should be a random mapping, and I know that's formally not formalizable, but in practice, that's what everybody thinks it is.

Mridul Nandi: But these are taken from the Federal Register Notice actually.

Rene Peralta: Okay so thank you.

Mridul Nandi: Thank you.

<applause>

End of Session7.mp3