Souradyuti Paul: So this is the third NIST session, excluding Bill's opening remark. Like the other two sessions, we have a sort of exciting discussions on the system priorities and the evaluation of attacks. And in the same spirit, in this session also, we will take a few issues on which we have a lot of inquiries put forward to us. And there is a good deal of postings in the hash forum too. So before I start, one thing I would like to mention that this competition of such a large magnitude, of such a large scale, it's quite natural that it will run into disagreements. It's more so because we have been dealing with a large number of parameters, a hash function which is very fast in hardware, which is using long message block. Analyzing for the other one may not be good in that respect, but it is good maybe in some other respect. So it is quite natural and quite expected that we may not have consensus on every point. So there will be disagreement. But the one thing that I would like to make clear, that we are not rigid. We have started out with some guiding principles. And we cannot let this competition go on forever. We are actually working under a very tight schedule. But we have always kept avenues for discussions. There is a hash forum. You can always let your opinion known through this. You can send email to us, so that we can revise ourselves and put ourselves back on the right track and go ahead. So that is what the purpose of these sessions are made for. So we'll be mostly in the listening mode and less in the talking mode so that we can get the best out of you. And that is what actually happened in those other earlier sessions.

And in this session, I will take two issues which have actually generated a lot of sort of questions from you that what are our plans of how to handle, particularly the issues of tunable parameters and the role of performance in the first cut. Mind you that now we are in the first round of evaluation. In future, in a couple of months maybe, we'll make the announcement of second round candidates, then we'll go forward with the second round. So now we're in the first round. And in this talk, there will be two talks really, and I'll give it back to back. And after that, we'll have a discussion. There you have the chance to air your concern, remarks, questions to us. And we can probably have a discussion on a variety of issues. And it will let our opinion clear to you to some extent, as far as possible.

Okay. So that lists the NIST's plan for handling tunable parameters. Everybody is aware of this issue already. But those who are still not introduced to these tunable parameters, these are nothing but expressing a hash algorithm in terms of some parameters, say, u,v,w. So I can always express the hash in terms of this $H_{u,v,w}$, and it will give the message. So that is how I express a hash. And if I take several values of u,v,w, we see that it has impact on the security performance. This is kind of you are trading security for performance. If you want more security, you lose performance, like that. I believe that everybody in the audience is well aware of all these things. Just for the sake of completeness. Some examples, there's a very a good example, the number of rounds. In every hash function, I don't say every, but ideally in most of the hash functions we'll see there is a compression function. And this compression function is iterative. So how many rounds should we allow? And block length, and there are various things you can use as a tunable parameter. So this is just a very brief overview of all the tunable parameters.

Now at this point, I think it is worth noting that what was NIST's initial announcement of a tunable parameter. Precisely, the Federal Register Notice, Volume 72, number 212 that had been published Friday, November 2, 2007, which reads-- it's a long paragraph, I don't want you to read this. However, you might like to be reminded of those lines in black. It says that "the submitted algorithm may include a tunable security parameter." And then "the submission should also provide any bounds that the designer feels are appropriate for the parameter, including a bound below which the submitter expects cryptanalysis to become practical." That is how you give recommended values for your tunable parameters. And then, which is maybe of some importance that NIST has perhaps made it a little bit clear about how it wants to handle tunable parameters. "NIST will consult with the submitter of the algorithm if it plans to select that algorithm for SHA–3, but with a different parameter value than originally specified by the submitter."

So these three points are to be noted that it is possible, if a candidate algorithm is selected for the second round, NIST might like to ask the designer to change the values of the tunable parameter. Okay. We shall take this issue more elaborately later.

Now the question is, what was NIST's motivation for encouraging the designers to provide tunable parameters in the algorithm? Mainly the motivation is actually two fold. It gives the algorithm flexibility. That is, you can tune your parameter and make it suitable for various applications, which may require more speed but less security, more security, less performance. And that is very important for all of us who are actually involved in the evaluation process that you can make a weakened version of this algorithm and then try to study it. It always gives more insight if you have a weakened version of the algorithm and then study it, even if it is not the full version. For example, suppose it is found that the tunable parameter t, which takes value five, it comes out to have some weakness. Then we just increase it, or maybe decrease it. I just, without loss of generality, you say that I put it five plus k. Now the discussion goes around heavily that what k should be. And we'll perhaps seek your opinion on this issue also, that how much security margin should be kept between the weakest, least performance broken algorithm and the algorithm version that we select.

It is nice to, when we find that many of the submissions really have provided us with tunable parameters. At the same time, I would like to mention that the designers' recommendations are also very important. Because we believe that the algorithm designer knows his or her algorithm the best. So this recommendation is also very important.

Now this is the most important announcement perhaps of some interest to you, that NIST will allow the submitters to adjust tunable parameters for the second round. Mind you that now we are in the first round. After the selections are done, the designs which have actually been selected, they have the chance to change their tunable parameters. They can provide us with different recommended values, they can even change the ranges of values that they have provided. Is something wrong over there?

Man 1: No.

Souradyuti Paul: Okay. Now, what are the other changes that we are going to entertain? I mean, this is quite obvious, the submitters can make editorial changes, add analysis, more analysis, or examples. And again, the submitters may make relatively minor changes to the algorithms selected for the second round. See the operative word here is the "relatively minor algorithm changes." We don't really know, but we have the plan to allow for relatively minor algorithm changes. But one has to explain really the impact of those changes.

Now when we make changes, I mean those who have been doing cryptanalysis, they know very well that those changes have potential risks. I mean, if your algorithm change is in response to a very good cryptanalysis, you know that we have less time to really analyze the changed version. The changed version could as well be interpreted as a new one. So it takes a long time. So changes to the algorithm may invalidate existing cryptanalysis, which hurts its chances. So any change that we see that is given directly in response to the existing cryptanalysis should not be taken very positively. And the effect of change, the change that you make, the designer should be able to very clearly specify the impact of the change. That is, how much impact does it have with respect to the original one? So this is what we expect along with the changes that will be provided with the algorithms selected for the second round. And it has to be very clearly specified so that we clearly know how much it has affected the original algorithm, assuming that the original algorithm is actually not broken. Now there is again, another important announcement, that is after the second round selections are made, the designers will have very little time to make changes. So it's better that if you're going to make changes, you start preparation now.

Okay, this is what we want to talk about our plan for handling tunable parameters. And we will take questions and discussion after the second talk. So now I would talk about the NIST's plan for role of performance in the first cut.

In the first round, that is now, we'll mainly focus on these two instruction sets, or architectures. That is, Intel architecture 32-bit, IA-32, and AMD64. We consider these to be very common platform, and they have plenty of use and applications. And on those two platforms, we have a lot of data. We are not primarily focusing on other architectures. However, if we receive performance data on other platforms through public analysis, these will not be overlooked.

Now at this point, it is quite natural that SHA-2 forms a very important benchmark. If we adjust tunable parameters to run an algorithm as fast as SHA-256, SHA-512, on these two specified platforms, IA-32 and AMD64, we ask is the algorithm secure if we tune the parameters to run it as fast as SHA-2? If it is not, I'm afraid that it hurts its chances to be advanced to the second round. This is our initial thinking on how to handle tunable parameters to have an impact on this next selection. Beyond that, we are not very much paranoid about performance in other platforms. But this is our sort of threshold. This we mainly focus on.

And when we talk about performance, the memory requirements closely come on the heels. For the first round, we mainly focus on software performance implementations, specifically, the RAM

requirement, how much RAM I need, and the code size. However, we again ask, in view of the constrained environment, is the algorithm implementable on constrained environments? But unfortunately, we do not have much data on that, but we expect.

Regarding hardware issues, we obviously consider the obvious hardware issues like the total memory and all this. However, in the second round, we expect to have more time for hardware details. Exact implementations and the gate counts on both FPGA and ASIC. So this is our initial standpoint on these issues.

There are some other issues under consideration that is the parallelizability, whether the algorithm is parallelizable or not. It has some importance in view of future processors which will have a lot of parallel processors. And again, that we have seen in a lot of other descriptions of the candidate algorithms, that whether the algorithm is easily analyzable. A very complex algorithm which does not really invite a lot of cryptanalysis will require a lot of analysis. And so it is important that the algorithm description is very clear so that it invites a lot of analysis and makes our life a bit easier to select. So the design should be easily analyzable. And of course comes the embedded systems, are applications on embedded systems too?

Some additional considerations are also there. If it is found very convincingly that a particular algorithm is exceptionally good on constant hardware environments or smartcard processors, we would view it very positively. But we do not have much data on that at present. And we encourage analysis by the public, and we depend very much on you regarding these issues.

So that is how I conclude this talk from our side. And we'll now move on to discussion phase, which is very important, where we'll deal mainly with these issues, maybe many more also. So now we are into the discussion phase, or question/answer. Okay. Niels?

Niels Ferguson: Hi, Niels Ferguson, Microsoft. There are some candidates that have-- a lot of candidates have like one primary tunable parameter, where it's a straight tradeoff between security and speed. There are also candidates that have two or three parameters, and there has been some discussion, people arguing about the slowest broken version or the fastest unbroken version. And it's not really clear what the right guideline is. And you mentioned you were interested in the slowest broken version?

Souradyuti Paul: I told that the slowest broken version of course would not be selected. But the fastest unbroken version may not give the reasonable security margin that we may like.

Niels Ferguson: The problem that I have with the fastest unbroken version is that there are so many variants of these algorithms that you break one and they say, "Well, just move sideways to another one that you haven't broken yet." And that might be an issue.

Souradyuti Paul: So this is a very important issue that I actually sometimes confuse this. If you have any opinion on this issue, I mean, just stand up and say if you have any opinion on this

issue, because as Niels pointed out, many of us already know that there are some sideways fastest unbroken, which is very difficult for us to handle. Adi.

Bill Burr: Niels, I don't expect this is going to be a huge problem in the sense that it's going to boil down to if we look at that and whatever we can tune, if it looks to us like it's slow compared to SHA-2, it's going to be very hard to see how we would ever sell it to the public. And we're not going to split hairs here really finely.

Adi Shamir: The easiest way to deal with the multiple parameters is to ask the designers of a cryptosystem to tie all of them into a single master parameter. So there will be one parameter which can be raised or lowered, and the designer himself will say how to tie all the other parameters to it in the best possible way. Otherwise, you'll get into never ending problems, like you mentioned.

Bill Burr: Was that a question or a statement?

Souradyuti Paul: It is a statement.

Adi Shamir: <inaudible>

Man 5: It was a question followed by its own answer.

Souradyuti Paul: It's regarding tunable parameters, right? So you want to say, "I just meant of tunable parameter."

Man 6: <inaudible>

Souradyuti Paul: No. Rich.

Rich Schroeppel: This is on issue two, on minor changes. I was completely puzzled as to whether you're allowing people to fix breaks or not.

Souradyuti Paul: Sorry, the question is?

Rich Schroeppel: I'm completely puzzled as to whether you're allowing people to fix breaks.

Bill Burr: If we decide that an algorithm's-- that for example, if you consider it a break that they did not block the length extension attack, if we decide that one of the algorithms that didn't block the length extension attack is otherwise interesting and good enough that we want to include it in the second round, then we would expect that to be fixed, actually. It's not going to be to your advantage for the second round selection to have any kind of a break. We obviously wouldn't pick it if we didn't think whatever-- if we do pick something for the second round that way, we wouldn't pick it unless we thought it was very straightforward or apparent, okay? And so it depends on

what you call a break or whatever. But the field here is so good that a break is certainly a big disadvantage, even a very technical, didn't fix the length extension, is certainly a big disadvantage.

Dan Bernstein: Dan Bernstein, University of Illinois at Chicago. I was wondering about when you say "faster than SHA-2," what does this mean for, for example, MD6? Because if MD6 is a few times slower than SHA-512 in 64-bit mode, but runs on four cores, then it's suddenly faster. So does that make it qualify already with the full rounds? I mean without tuning any parameters? Or do you need them to say in one core, we need to tune the parameters? What does MD6 need to do to convince you that yeah, it's better than SHA-2 performance-wise, and still secure? I'm taking MD6 as an example, because I think it's an interesting submission which I don't have any personal plus or minus feelings about it, it's interesting and they say that they're reasonably fast with parallelization and they've got a lot of rounds, which they don't need that many rounds. So what does MD6 need to do to convince you they're okay?

Souradyuti Paul: So is it not sufficiently clear what we said about that if-- is it not sufficiently clear in the talk that we just adjust the tunable parameter to run it as fast as SHA-2. And then we see that if it is secure or not.

Dan Bernstein: I don't think it's quite clear. One part is that it's not clear if you're taking account of parallelizability when you adjust it to be as fast as SHA-2. MD6 is saying, "Hey, we're already as fast as SHA-2 because we can use multiple cores, and that gives us the parallelization advantage." And another thing is that, well, when you adjust the tunable parameters, do you need implementations from people? Or can you just figure that out for yourselves? Does the MD6 team need to do any more effort to convince you that they are fast enough?

Souradyuti Paul: Okay. We shall think about that.

Bill Burr: I don't want to get too much into particular cases here. I've thought a good bit about MD6. And I'll think a good bit more about it in the weeks that come. And so will the other members of our team. And there are several others I think in that category. And with something like MD6, I suppose the obvious thing you can do is try to account for the effects of multiple cores, or for the effects of producing the number of rounds. And then if you do that, you tamper I guess with the proof on differential analysis. So that's a pretty complicated case, and I don't want us to commit ourselves at this point of just how we're going to deal with that. MD6 is not the only algorithm, I think, that has these kinds of considerations and also considerable strengths. And so we just don't know at this point, and we welcome your inputs.

Rich Schroeppel: This is another question in the same vein. I think Dan's can be summarized as what are you going to do about counting multiple cores? And the other issue is, there's a big performance improvement on the horizon for S-box mapping with Intel coming out with an instruction that maps S-boxes. And the issue of how you're going to figure in performance for that and for the XOR [ph?] multiplying instruction and so on, AES, seems to me relevant. And I'm not

asking for an answer now, but it seems something that people are going to be wondering about. Because the performance can go way up or way down, depending on do you have these magic instructions or not.

Dmitry Khovratovich: I have a comment on minor issues, on minor changes. I would like to propose somehow divide changes on the hash function on the mode and changes on the compression function. Because many attacks that do not explicitly exploit some flaws in compression function, countermeasures to this attack can be made with some minor changes. Introducing small door counters, permutations, something else, until those changes are quite minor and easy to understand. While you attack the compression function, the changes will likely to be not so minor. And it won't be easy to explain why the change will strengthen the hash function, but not to reduce the security. Maybe only increasing this tunable parameter will obviously increase the security of hash function. And I propose when talking on the minor changes, somehow distinguish between changes on the compression function and changes on the mode of hash function in general.

Souradyuti Paul: So this is your view right? On minor changes?

Dmitry Khovratovich: Yes.

Souradyuti Paul: Okay. Wherever you change, in the mode of operation or in the compression function, we have no plan of make a distinction between changes on the mode of operation or compression function. But as we have already made clear that we should try to measure or evaluate the changes against these issues, that whether it invalidates the existing cryptanalysis or what's the effect of change really on the whole algorithm? So one should be very, I mean, clear about this one. But we have no plan of making-- I think you are proposing or you are in favor of having a distinction between compression function and mode of operation. Is that correct?

Dmitry Khovratovich: No. I just introduced some clarification that there will be surely problems on introducing changes on the compression function. And on the mode, there will be more easy and clear to change them.

Souradyuti Paul: Okay. Thank you. Hongjun.

Hongjun Wu: In your presentation, I saw to compare performance you will use the fastest version, you will try to reduce the parameter to say whether the reduced version equivalent to SHA-2.

Souradyuti Paul: Yes, yes.

Hongjun Wu: Then use that for performance, right? But I think it's just too difficult to do it in practice. How do we make sure that the reduced version is the equivalent to SHA-2 in security? I think it might take several years to do it.

Souradyuti Paul: No, actually, we have talked about in terms of tunable parameter. So if your algorithm has a tunable parameter which actually trades performance against security, then I can as well change the tunable parameter to run it as fast as SHA-2, possibly. If I can't really, I mean, for any possible value of the tunable parameter, if it is lower than SHA-2, I think it will have some impact on the selection.

Hongjun Wu: But you can always tune the parameter to be faster than SHA-2. There's no problem.

Souradyuti Paul: No, if we have the concrete tunable parameter, I can always do that by removing some operations, right? But I can't really, unless you provide us with a tunable parameter.

Hongjun Wu: Yeah, so--

Souradyuti Paul: So, we'll all of us do it legally. I mean, I can always change the tunable parameter if it is there, and then I can do it possibly, I believe.

Hongjun Wu: Yeah, you can reduce the parameters so that it will run as fast as SHA-2. But it's not easy to tell whether its security is still the same as SHA-2 or what's the security. I think still you have to admit, the most difficult thing is it's difficult for you to analyze the security of the reduced version, right?

Souradyuti Paul: Okay. Okay. Thank you.

Hongjun Wu: Another question is you specified platforms, then I would like to ask what kind of compilers and the compiler options were you using?

Souradyuti Paul: I think we have made it very clear in the Federal Register Notice that we'll be using ANSI C, right Bill? The compiler that we're using is ANSI C.

Bill Burr: The compiler? You're asking compiler? We don't expect to recompile anything. I think if you look at the submissions, some folks are a lot more conservative than others, okay? And some people have chosen to put a stake in the ground that seems very, very safe. The truth is that we're going to have to look at the available cryptanalysis on whatever it is and make a guess about what would be reasonable. I'm not very good at those kinds of guesses, but I'll rely heavily on John Kelsey and Soura and others who are. And if it looks like that odds still be pretty good, then we may go ahead. I do accept the principle that there has to be a pretty good argument that it's at least as fast as SHA-2, because I don't see how we're going to get people to use an algorithm that is slower than SHA-2. And other arguments will have to deal with how do you account for multi-threaded implementations and modes? And we haven't figured that out yet, to be honest.

Souradyuti Paul: We have taken input from you and we have taken a note of that. We'll try to address this issue internally and let you know.

Bill Burr: So I mean, I have the impression that MD6 is example that people always talk about, you could break the proof and still have quite a margin on what anybody is likely to be able to analyze. And so there's some reason to think that maybe it's reasonable to talk about reducing the number of rounds and seeing what it looks like when it runs as fast as SHA-2 on a single core.

Souradyuti Paul: Danilo.

Danilo Gligoroski: I would like to return on the question of multiple cores, performing hash on multiple cores. And we have to be aware that a vast majority of hash applications, or use of the hash, will hash not extremely long messages where multiple cores can give you advantage. For example, Internet traffic is not so huge. And if you have one 1500 bytes and you want to hash them, multiple cores will not help you. Actually, they will drag you down with the speeds if you try to compute hash of 1500 bytes over multiple cores. So it is nice to-- I would like to see, let's say, MD6 or any other type of mode of operations of hash functions where multiple cores can be used for any winner or any SHA-3. But especially just to promote the multiple cores as a big advantage comparable to the SHA-2 performance is, I don't think it's completely correct.

Souradyuti Paul: Thank you. I hope there is no question, that it is a statement.

Danilo Gligoroski: _____.

Souradyuti Paul: Okay.

Man 12: I have a comment for the smartcard implementation. So I think there are two types of implementation for smartcard. One is using assembly language. The second is using a common platform, such as a Java Card platform. And the situation for both environments is different. And in some cases, we would like to implement it, the application was cryptographic function to Java Card platforms. Please consider such a situation. I think this is an issue of the second round, but I thought I would like to consider that point.

Rich Schroeppel: Are you allowing teams to join their submissions?

Souradyuti Paul: A bit louder, please?

Rich Schroeppel: Are you allowing teams to join their submissions?

Man 13: <inaudible>

<laughter>

Bill Burr: To join their submissions?

Rich Schroeppel: Some of the AES hashes for example, were very similar. And one might imagine that two of the teams would decide to work together to produce an intermediate version.

Bill Burr: I'd really have to think about how we could make that work mechanically. I don't know what would happen if two teams got together and they said, "Well, if we both got picked, then we want to make a minor change to merge the two."

Souradyuti Paul: This is kind of a nice tradeoff.

Bill Burr: If somebody wants to do a merge, let us know sooner, rather than later.

Souradyuti Paul: But we might like to consider whether this falls under minor changes or not. So if it is minor change, then we'd probably have to think about this.

Hilarie Orman: I was curious about what feedback from AES NIST had gotten about hardware applications, constrained environments, parallelizability? What was the feedback from industry about the AES choice and its effect on their ability to implement in constrained environments?

Bill Burr: About the AES choice?

Hilarie Orman: Yeah. I mean, during AES, this whole subject of constrained environments and specialized hardware and everything kind of hit people as a surprise. They didn't know how they should think about it, and they didn't know what you were going to think about it. And it was all sort of up in the air. But then you went ahead, you made the AES choice. People have done implementations, what feedback did you get about how that choice affected things?

Bill Burr: Well I don't know that we've got a tremendous amount of specific feedback on that. People seem pretty happy with the choice, in general. I wasn't around for the whole AES process, but only for the end of it. I do have the distinct impression that the whole constrained platform thing kind of took us a little by surprise, and that we acted to it. But the fact was that we had choices that didn't seem like we gave up very much to cover the constrained platforms. And so why not? And if for some reason we really had to make a choice, you couldn't do a good job on the big machines and the little machines at the same time, then you'd have to make a choice, okay? But at least in the AES, in the block cipher arena, it seemed like either Serpent or Rijndael could do pretty well on constrained platforms in some ways, although Serpent might have been a little slow just in terms of the number of rounds it had. But they're both very efficient in hardware. And so the argument raised itself and certainly in making the final selection we paid I think a lot more attention to constrained platforms than we had, I believe, thought we would have going into the competition. There's a lesson in that, and I think that is that you learn a lot while you're doing

the competition about what you want. And you shouldn't be too rigid. I think it would have been a big mistake in this competition to have tried to lay out rigid rules that we were going to follow for the duration of the competition, rather than adjust to what we saw. If we don't have any more questions, maybe we should go to lunch 15 minutes early.

Niels Ferguson: Second.

Bill Burr: Okay, let's do that. Oh wait, do you have a-- No. Okay.

<applause>

Souradyuti Paul: Thank you.

End of Session9.mp3