Comments on the Transition Paper

From: hugokraw@gmail.com [Hugo Krawczyk]
Sent: Wednesday, July 08, 2009
Hi,

I am reading the document and was wondering what distinguishes "data authentication" from "entity authentication". For example, when IKE applies a signature it is certainly doing entity authentication but it is also signing data, for example, negotiated algorithms.

As another example, is a signature in a certificate "entity authentication" or "data authentication"?

A clue to what you mean by differentiating between the two cases seems to be the following text in page 5: "signature verification for entity authentication is performed immediately after signature generation; therefore. there is no requirement to retain a signature for later verification. "

Would I be correct to say that the actual differeniation you are doing is between signatures with long-term verification needs and those with short-term (or ephemeral) needs?

This may still require an understanding of what is short-term and long-term (*) but still seems to be better defined than "entity vs data".

(*) Deciding on short term vs long term is not straightforward. In the IKE example, an application may record IKE exchanges for the sake of auditing or archiving and may need to validate signatures well after the protocol run is complete.

One additional question, only tangentially related to your paper. In table 4 you specify compliance with SP 800-56A. I was recently asked the following question to which I did not have an answer. Is it relevant to talk about MQV in the context of IKE given that IKE has its own signature-based mechanism for key exchange? What should be the answer to this question?

Thanks

Hugo

From: Ratcliffe, Steve [mailto:sratcliffe@icsalabs.com]
Date: Wednesday, July 08, 2009

Page 1, line 1, "providing": Perhaps you should change this word to authoring. NIST did not just start doing/providing key management guidance 9 years ago!!

Page 1, lines 2-3, "management guidance": This included lessons learned over many years of dealing with key management issues": It might sound better to say NIST began the task of providing cryptographic key management guidance via SP 800-57 (or any other document that shows you management guidance) to the US Federal Government and assistance to the commercial community, in the form of technical understanding. Remove century. That was only 9 years ago and key management has been going on a lot longer.

Page 1, line 6, "algorithm breaks": Is this actual breaks or academic breaks. I am not aware of any actual breaks so it might be safer to say academic this way no one gets the impression that any algorithm has actually been broken.

Page 1, line 6, "techniques": This is a really long sentence. "over many years" is that 2 years or 50 years? "dealing with key management issues" is this government issues or has this been government and commercial? The later carries more weight if this paper is going to get any acceptance from the commercial/civilian community. "attempts to encourage the definition and implementation", this sounds like NIST has no idea so they kept pushing the world for a definition which goes against the many years of dealing with key management statement in the second sentence.

Page 1, line 7, "Publication (SP) 800-57": This is a three part document. so you should at least mention that here and list them separately in the back. You can add the term DRAFT.

Page 1, para.2, "definition of security strengths, the association of the approved algorithms with these security strengths, and a projection of the time frames during which the algorithms could be expected to provide adequate security": This information isn't really cut and dry within the document so it might be a good idea to state that this comes from section 5.6 of SP 800-57 part 1. Such as Some of the guidance provided in SP 800-57 Part 1 specifically section 5.6 includes the definition of the security strengths, the association of the approved algorithms as they are associated with the security strengths and finally a projection o f the time frames....

Page 1, para. 3, line 2, "discovering the key": This is a brute force attack. so this is a measurement of the difficulty of discovering the key during a brute force attack.

Page 1, para. 4, line 5, "so easily done": It might be good to site a few examples as to why you believe this is not so easily done. Such as cost, breakdown in interoperability, vast number of changes required to maintain continuity and so on.

Page 1, para. 4, line 6, “class of algorithm”: I do not understand what you mean by class of algorithm. I understand class as it might pertain to hash, cipher, mode of operation. I also understand algorithm as it might be SHA, AES, Triple DES. What exactly are you going to examine. That is what you should say here.

Page 1, para. 4, line 6, “sometimes make some adjustments”: Are you talking about strength or algorithm implementation? Sometimes and some used here does not sound right. I would remove one of the words.

Page 1, para. 5, line 2, “NIST’s”: The CMVP is both NIST and CSEC. So for political reason you might want to remove NIST's

Page 1, para. 5, lines 3-4, “which is responsible for validations cryptographic modules for conformance to FIPS 140-2”: Does not sound right. which is responsible for validating cryptographic modules against FIPS 140-2 standards
or
which is responsible for validations of cryptographic modules against FIPS 140-2 standards.

Page 1, para. 5, line 5, “approved cryptographic algorithms”: The phrase "approved cryptogrphic algorithms" is not used in Annex A. It might be better to say i.e.; AES, Triple-DES, SHA, DSA, HMAC to name a few.

Page 1, para. 5, lines 6-10, “CMVP is the vehicle used for testing conformance to FIPS 140-2 and the approved algorithm specifications. In some cases, an algorithm or protocol that has not been approved in a FIPS or NIST Recommendation is “allowed”; an algorithm is indicated as allowed by means of the FIPS 140-2 Implementation Guidance document”: Going off CMVP web site under What is the purpose of the CMVP. You might change this statement to read something like.

The CMVP was establish to validate cryptographic modules to the current FIPS 140-2 standard. Likewise the CAVP was established to validate cryptographic algorithms to the appropriate algorithm standard. CMVP is not a vehicle. It is a process.

Recommend change this sentence to something like:
Algorithms and/or protocols that are not listed as approved in FIPS or NIST Recommendation are sometimes permissible, in those cases FIPS 140-2 Implementation Guidance will provide the required details for those algorithms and/or protocols. In those cases any deviation from the Implementation Guidance is not allowed. Use of algorithms and/or protocols not listed in any NIST recommendation or FIPS would require permission from the Security Technology Group within NIST.

Page 1, para. 5, last line, “can”: Remove this word.

Page 2, Section 2, line 1, “confidentiality of sensitive information”: What about integrity, authentication and non-repudiation. These are all part of encryption.

Page 2, Section 2, line 1, “by the Federal government”: The CMVP is NIST and CSEC. Do you really want to say Federal?? Why not end the sentence following sensitive information.

Page 2, bullet 1, line 1, “two key sizes”: This sounds more like the keys vary in actual size such as 64 and 128 bits when in fact what this should be is number of different keys. Two key Triple DES uses two different
keys while three key Triple DES uses three different keys.

Page 2, bullet 2, “was”: In keeping with the other two bullets recommend changing this word to "is".

Page 2, bottom paragraph, “Federal government”: Again the use of Federal government. This may be a NIST publication but the program is CMVP and has both NIST and CSEC. Recommend removing.

Page 2, bottom para., line 3, “performed”: Change this word to issued

Page 2, bottom para., line 3, “these algorithms after that date”: This part of the sentence only talks about the CAVP and algorithm validation. Then the second part of the sentence talks about CMVP certificates issued will be modified to
remove these algorithms. The algorithms are issued certificates (actually only numbers) by the CAVP and the cryptographic module certificates are issued from the CMVP. Two different certificates and two different programs. Both will need to be modified or pulled.

Page 3, lines 1-2, “that module will no longer be valid”: This is a nice statement but how will a user of one of these modules know that the system they are using is no longer FIPS valid.

Page 3, last line, “between”: Between does not sound right. Suggest change between to from since the transition is from one FIPS to another.

Page 4, lines 1-2: On page one you say this paper is to bring attention to government and the public. While this statement is aimed more at the labs and not the government or the public. Perhaps you should say what the new tests are so as to better inform the government and public.

Page 4, para. 2, last sentence: Recommend changing this sentence to:
FIPS 186-3 features which currently have no tests implemented will require vendor affirmation when submitting for validation.

Page 4, para. 3: New implementation submitted for testing and validated Dec 1 2010 under FIPS 186-2 may find their validation being pulled come Jan 1 2011. I would highly recommend in order to cut down on confusion. wasted time and expenses that a statement

be added here and possible other parts advising vendors and end users to move to FIPS 186-3 long before December 31, 2010.

Page 5, bullet 1, lines 1-2, "New implementations must generate digital signatures with a security strength that is equal to or greater than 80 bits through December 31": Although this statement is correct it is really impractical. A new implementation entering the CMVP process would not gain final validation until, at the earliest, November/ December 2009. 13 months later it would lose its validation. Users that procured any system with 80-bit security would have to replace the system or be forced to use a system that is not FIPS validated. I would propose that new implementations entering the process today have both 80 and 112 bits security available so as not to jeopardize the validation. This point should be driven home to both users and vendors.

Page 6, Section 4, line 2, "random number generators (RNGs)": NIST needs to determine what name they want to use and stick to it. I read one source that refers to random bit generators and another source like this says random number generators.

Page 6, 1st line under Table 3, "RNGs specified in SP 800-90": Recommend this be RBG and you should add in deterministic since 800-90 deals with deterministic random bit generators. I also suggest adding in the following information:
The DRBG mechanisms specified in this Recommendation support four security strengths: 112, 128, 192 or 256 bits. The actual security strength supported by a given instantiation depends on the DRBG implementation and on the amount of entropy provided to the instantiate function. A DRBG mechanism requires instantiate, uninstantiate, generate, and health testing functions. A DRBG mechanism may also include a reseed function. A DRBG **shall** be instantiated prior to the generation of output by the DRBG. This info comes from section 8. Specifics on HASH, HMAC, CTR, and DUAL_EC is in section 10.

Page 6, last line "SP 800-56A": I believe the doc is actually 800-56A Revised.

Page 7, line 5, "MQV primitives": (section 5.7 pertains).

Page 7, line 5, "approved KDFs": (section 5.8 pertians).

Page 7, Para. beginning with "Protocols...", lines 2-3, "capability to interoperate with the older protocols": Commonly referred to as backwards compatibility.

Table 4, last 8 rows: I would recommend listing besides each protocol one source (RFC, FIPS, SP, IETF doc). Remove any confusion an Engineer might have and a tester have if everyone is working from the same source.

Page 8, Section 6, para. 2, sentence 2: I recommend removing this statement. It is true in the sense that we are not required to test but when ever I get a product that identifies using one of these I at least verify that it

is used by running a sniffer. Any good analyst will at least verify that something is at least implemented if it is humanly possible.

Page 8, Section 7, line 3, "RNG": RNG or RBG???

Page 8, Section 7, para. 2, lines 4-5, "a new section of the FIPS 140-2 Implementation Guidance will be developed": This is an interesting statement. Will be developed, leads me to believe that nothing has been started. So is this 30 days, 90 day or 6 months away?? It might be better to say
that it is being worked on and expected to be released sometime over the next what ever period.

Page 8, last line, "Note that since Two-key Triple DES will be disallowed after": I can not find this statement in 800-57 relating to key wrapping and effective dates. If I am wrong please identify the source. Otherwise I would recommend changing this statement to something like:
Note that since Two-key Triple DES will be disallowed after December 31, 2010 (see section 2 above), it can be derived that Two-key Triple DES will not be allowed for key wrapping after that date.

Page 9, bullet 2: You mention the use of protocols like IKEv1 and IKEv2 above which are reference in RFC's. These references list SHA1 as MUST. It is possible the RFC will get changed but highly doubtful. I would recommend that a caveat be added somewhere here or in the section above about protocols or both that says the allowance of security functions as stated here or in another SP/FIPS document will be followed over the MUST statements within the protocol documents. This is not to say those protocol documents are at fault or in error but fro security considerations those documents are not updated to meet today's ever changing security needs.

From: Stan Kladko [mailto:kladko@bkpsecurity.com]
Sent: Tuesday, July 07, 2009
Folks,

Here are some comments for the whitepaper

1. The document does not include a discussion of HMAC which is important since HMAC is used in a number of security protocols.

2. The document does discuss IKE but does not discuss IPSec. IPSec uses truncated HMAC-SHA-1 which is only 96 bits long. This means that for long-lived IPSec connections, there will be two messages with the same HMAC after approximately 2^48 IPSEC messages are transmitted. If IPSEC is used to secure a terabit optical link, sending 2^48 messages is actually feasible. The fate of IPSEC and specifically the truncated HMAC-SHA-1 needs to be discussed.

3. The issue of key transport using SSH or TLS needs to be addressed since it is of widespread use. As an example can an administrator use a remote SSH connection to import cryptographic keys into a module?

There are actually two options:

a) SSH is allowed after 2010 and key transport using SSH is allowed after 2010
b) SSH is allowed after 2010 but key transport using SSH is NOT allowed after 2010 since SSH is not an approved key transport method and since SSH provides less than 112 bits of security.

The document needs to articulate whether a) or b) is true.

In general which key transport methods using symmetric algorithms are going to be allowed after 2010?

4. The document splits signatures into signatures used for "non-repudiation" and signatures used for "authentication". I am sure that there will be many discussions on this subject in the future, since in many cases a signature will serve both purposes. A user may use a smartcard with an RSA private key to authenticate to a bank and issue a transaction. Later the bank may use the fact that the user signed in using the smartcard to show that the transaction was authorized by the user - this is non-repudiation. Therefore, the distinction between non-repudiation and authentication is in many cases quite blurred.

5. The wireless 802.11i protocol uses HMAC-SHA-1 to derive session keys during the four-way handshake. Is this use of HMAC-SHA-1 going to be allowed after 2010?

6. The document mentions two RSA signature types - PKCS#1 and ANSI X9.31. There is actually another RSA signature type which is implicitly approved by NIST but not mentioned in the document. In particular, during TLS 1.0 handshake the ephemeral Diffie-Hellman parameters may be signed by an RSA signature. This signature is "TLS-RSA" signature and is neither PKCS#1 nor
ANSI X9.31. The signature generation for "TLS-RSA" signatures is described in the TLS 1.0 standard specification.

The document needs to mention this signature type. Many people think that TLS 1.0 uses PKCS#1 signatures. This is not exactly true. There are actually two RSA signature types used in TLS 1.0. RSA PKCS#1 may be used to sign certificate chains. "TLS-RSA" is used to sign ephemeral Diffie-Hellman parameters.

With best regards,
Stan Kladko, Aspect Labs

From: Paul Hoffman [mailto:paul.hoffman@vpnc.org]
Sent: Monday, July 06, 2009

a) The title is incomplete in that it is too general. This document is, as it points out later, from the view of the CMVP. That should be reflected in the title of the document, such as "FIPS 140 and the Transitioning...".

b) Although two-key Triple DES is discussed in FIPS 140 and SP 800-56A, it is almost never seen in deployed products. For example, I see no certificates in <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm> that list two-key Triple DES. Its inclusion, particularly in Table 1, could be confusing to someone who thinks "Triple DES" means three-key, as it does nearly everywhere. As much as I hate footnotes, I suggest that you relegate three-key Triple DES to a footnote to keep the focus on three-key TripleDES. (As a picky side-note, you should be consistent about the usage of "2-key" vs "two-key".)

c) Section 3.2 says that SP 800-57 Part 1 "needs to be revised". This should be committed to: "will be revised soon".

d) Table 2 has a very deep problem in that current signatures that use SHA-1 as the hash algorithm are much weaker than 80 bits today. The current collision attacks on SHA-1 have reduced it to 52 or 62 bits, depending on who you listen to. NIST has said that the security strength of a signature (particularly in certificates) is affected by the collision strength of the hash function used. Thus, using the number "80" in the "New Validations" column is terribly misleading because current validations cover signature algorithms that are significantly weaker than that.

e) There is no rationale for the 2015 date in Table 3, but there should be, similar to the way you explained the extended date for entity authentication earlier in the document.

f) The future testability of IKEv1, IKEv2, TLS, SSH, and so on in Section 5 is quite muddled. The last paragraph is particularly confusing, given that earlier you say "However, the KDFs that do not comply with the KDFs in SP 800-56A will not be tested." Similarly, it is not clear whether "Non-tested DH and MQV primitives" in Table 4 refers to those from IETF protocols or something else. You need to use consistent wording throughout this section so readers can understand whether you are talking about IETF protocols or something else. I recognize that this is a tricky area for NIST, but it is particularly important because many US Federal agencies mandate the use of some of these protocols and need to know NIST's long-term plans for those protocols.

g) Similarly, in Section 5, the sentence "However, any new versions of these protocols using DH and MQV key agreement must be designed to conform to SP 800-56A" is just plain silly. NIST has no power over the IETF or other SDOs. You could threaten "any new versions of these protocols using DH and MQV key agreement that are not designed to conform to SP 800-56A will not be tested", but that is also shooting yourself in the foot because everyone is using those protocols, not the ones that meet 800-56A. I propose

that NIST instead deal with the topic more directly by simply saying that it strongly suggests that all future KDFs conform to 800-56A but it will continue to choose to test KDFs in FIPS 140 on a case-by-case basis, and tie this stated desire in with the text from bullet point (f).

I hope these comments help the process. Again, it is great to see that NIST is going to publish this guide!

--Paul Hoffman, Director
--VPN Consortium

From: Ian Simmons [mailto:ian@athena-scs.com]
Sent: Wednesday, July 22, 2009

Dear NIST,

Thank you for publishing the paper *The Transitioning of Cryptographic Algorithms and Key Sizes*. We have the following questions.

1. Section 2 on the use of 80 bits of security strength for encryption states: "CMVP certificates that were previously issued will be modified to remove these algorithms from the approved list for the FIPS mode". If we look at one of our certificates (for example, http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm#925) it states - *FIPS-approved algorithms:* Triple-DES (Cert. #560). If you find this on the CAVP DES Validation List (http://csrc.nist.gov/groups/STM/cavp/documents/des/tripledesval.html#560) you will see it is here that KO 2 is specified.

   Are the actual actions then:
   - KO 2 will be removed from all CAVP certificates on the Triple DES Validation List.
   - Any CAVP certificates on the Triple DES Validation List that only had KO 2 will be removed.
   - Any removed CAVP certificates will be removed from CMVP certificates on the Validated FIPS 140-1 and FIPS 140-2 Cryptographic Modules list.
   - CMVP certificates on the Validated FIPS 140-1 and FIPS 140-2 Cryptographic Modules that only had this approved algorithm will be removed.

   For the given example this will mean:
   - KO 2 will be removed from CAVP Triple DES Validation List certificate #560.
   - The certificate will remain valid because it is also for KO 1.
   - The CMVP Validated FIPS 140-1 and FIPS 140-2 Cryptographic Modules certificate #925 will therefore be unchanged.

2. Section 3.2 states: "the generation of digital signatures with less than 112 bit of security strength will no longer be considered valid for the FIPS mode on a FIPS 140-2 validation certificate". Will existing certificates be modified in the same way as for encryption? That is, will all certificates on the CAVP RSA Validation List with less than 112 bit of security strength (due to SHA-1 or 1024-bit RSA) be removed and the changes echoed through to the CMVP list?

3. What is the position for digital signatures for data authentication with Triple-DES MAC (from the IG rather than FIPS 186-3)? Is it the same that for 80 bits of security strength generation will not be supported after 2010 but verification will be allowed?

4. Same as question 2 but for SHA-1 in Section 9 Hash Functions. That is, will all certificates on the CAVP SHS Validation List with less than 112 bit of security strength (SHA-1) be removed and the changes echoed through to the CMVP list?

There are also some very minor typos:

1. Page 1, paragraph 5 superfluous word “can” at end.
2. Page 2 uses “2-key” and “3-key”, and page 3 uses “two-key” and “three-key”.
3. Page 4: Table 2 caption is orphaned at bottom of page.

Best,
Ian

From: Wallner, Debbie M [mailto:dmwalln@tycho.ncsc.mil]
Sent: Monday, July 27, 2009 5:41 PM

- Table entries:

Suggested to use a different term than "OK" as an entry, perhaps "Allowed".

- Page 1, Section 1, paragraph 3:

Describing security strength "only" as a measure of the difficulty of discovering "the key" is too simplistic.

Perhaps NIST could say that the security strength of a cryptographic mechanism is a measure of the difficulty of discovering any secret key(s) associated with the mechanism or otherwise defeating the intended purpose of the mechanism.

In that same paragraph, replace reference to "RSA using a key length of 1024 bits..." by "RSA using a modulus whose length is 1024 bits..."

There are too many possible interpretations of what constitutes the public/private RSA key -- and for a 1024-bit modulus, most of those keys would require more than 1024 bits. (E.g., e, (e,N), d, (d,N), (e,d,N).)

- Page 4, Section 3.1, penultimate paragraph:

There is a statement here that invalidation of certificates for implementations providing only 80 bits of security will affect "all DSA

(currently) validated implementations" is contradicted by Table 2 and the discussion that follows it.

- Page 4, Section 3.1, penultimate paragraph:

The meaning of the last sentence of this paragraph is not clear "Note that the invalidation ..... for non-repudiation purposes."

- Page 4, Section 3.2:

The security strength of the digital signature key is not the same thing as the strength of the scheme that uses it. FIPS 186-3 allows the use of under-sized hash functions which could lower the strength of the scheme below that which might be expected by looking only at the cryptographic subgroup.

- Pages 4-6, Section 3.2:

The issue is more over authentication that needs to be maintained over a period of time vs one-time authentcation rather than between data authentication, entity authentication, and software/firmware integrity. Entity authentication is probably always in the latter case - One only cares about authenticating the entity they are communicating with right before they communicate with that entity. Or in the case of an email, the first time they read the email is the when it matters. Software/firmware integrity probably always falls in the first case - one may wish to verify that the software has not been modified every time they use it. The data case can go either way depending on the lifespan of the data.

- Page 5, Table 2:

It is not clear as to the difference between the treatment of "already validated implementations" w/r/t signature generation for data authentication and signature generation for "software and firmware integrity testing." (E.g., isn't software a type of data? Why is it OK to generate new, weaker signatures on software after 2010 but not OK to generate such signatures on other forms of data? Malware is a BIG problem.)

- Pages 6-8, Section 5:

This is an especially important section, but it is not entirely clear. It might help if this section was split into two sub-sections, one on DH/MQV primitives, the other addressing KDFs.

- Page 7, First full paragraph:

RE: the talk of eventually testing the DH or MQV primitives used by commonly-used protocols for SP 800-56A compliance (independent of the KDF). Be aware that SP 800-56A uses cofactor DH for elliptic-curve-based implementations, so implementers not using prime-order curves should be warned. Also, how will compliance be tested? By comparing Z values generated by NIST against those generated by the IUT? The failure by the IUT to zeroize Z would constitute a violation of SP 800-56A (let alone outputting the value!) It's a catch-22.
If testing is non-invasive, how are they going to test the output of the primitives when a non-NIST KDF is used? We don't want operational devices able to output Z.

- Page 7, Second full paragraph:

What constitutes a "new" version of an old protocol? When does the "new" timer start?

- Page 7, Table 4:

What about protocols that explicitly make reference to carrying out a TLS-like handshake to produce a master secret? Are they covered by the TLS entry? (That is, what about things like DTLS, EAP-TLS, and maybe even some wireless protocols.)

- Page 7, Table 4:

KDFs for S/MIME, SSH and other established IETF protocols are not addressed.

- Page 9, (the other) Table 4:

Isn't this Table 5? Anyway, more guidance has to be given to distinguish beween "hash-only" and "all other applications." For example, one could argue that HMAC is a "hash-only" application. (What else does it use?)

- Page 9, Bullets on SHA-1.

Something should be said about the (80-bit) security level associated with SHA-1 (when used with signature schemes) before pointing back to table 2. It's a little confusing, since that table is in a section talking about security strengths for signature keys. Actually that table doesn't even say what the 80- and 112-bit strengths refer to. It really should refer to the signature schemes, not just the keys. One could use a 128-bit-strong key (i.e., 224-bit cryptographic subgroup) but if it is combined with a SHA-1 hash, they have an 80-bit-strong digital signature.

From: Chang, Shu-jen H. [mailto:shu-jen.chang@nist.gov]
Sent: Tuesday, July 28, 2009 11:32 AM

Regarding the Discussion Paper: *The Transitioning of Cryptographic Algorithms and Key Sizes*, I have a few comments:

- Page 1, Paragraph 5, Line 3: the first word should be "validating" rather than "validations".
- Same paragraph: Shouldn't the current standard FIPS 140-3 be cited instead of FIPS 140-2? If so, other occurrences may need to be changed too.
- Same paragraph, the last sentence: Is the last word in the sentence ("can") needed?

Other than these, the draft looks fine to me.

Shu-jen

From: Natarajan Vijayarangan [n.vijayarangan@tcs.com]
Sent: Tuesday, July 28, 2009 1:24 PM

SHA-1 hash algorithm can be strengthened via Message Preprocessing methods (MP). Until SHA-3 algorithm is finalised, we can use SHA-1 via MP. This MP approach shows that SHA-1 is efficient. Michael Szydlo (RSA Security), Yiqun Lisa Yin (Independent Consultant) and Natarajan Vijayarangan (TCS Innovation labs) proved that SHA-1 via MP is simple and secure.

In page 9, the following change has to be incorporated in Table 4: Hash functions transitioning

Hash algorithm	New Validations
SHA-1 via MP	OK for all hash function applications SHA-1 via MP through December 31, 2012

Regards

Natarajan Vijayarangan
TCS Innovation labs Hyderabad
India

From: Reiter, Robert [mailto:rwreite@tycho.ncsc.mil]
Sent: Wednesday, July 29, 2009 10:12 AM

I did not go through in detail to review and did not get comments to Debby.
I will note that one portion I did need to refer to, the treatment of KDFs, was hard to understand. I note that others had similar comments. It is not clear how KDFs from established IETF protocols would be handled, near term and longer term.

It was good to see that the outside crypto folks had good comments and were complimentary to NIST for responding to need for this paper.

Thanks,
Bob

From: Kim Schaffer [mailto:KSchaffer@cygnacom.com]
Sent: Thursday, July 30, 2009 4:07 PM

Corrections:

Section 3 – Digital Signatures

All ECDSA implementations that have been validated to FIPS 186-2 (all current ECDSA validations) use only SHA-1, so all of them will be affected, along with all current DSA validations.

Section 4 – Random Number Generation:

Footnote 3 is Footnote 3 is incorrect. The ANSI X9.31 RNG always used 2-key Triple DES (described in terms of multiple passes of DES). The NIST pdf referenced extended that to 3-key Triple-DES and also 128, 192, and 256 AES.

The Approved ANSI X9.62 RNG is not listed. It should be listed with the FIPS 186-2 RNGs on lines 2 & 3 of table 3.

Comments:

Section 3.2 – Security Strength for Digital Signature Keys

Given that Signature Verification of ≥ 80-bit is going to be permitted past 2010 and Signature Generation for Entity Validation of ≤ 112-bit is going to be permitted through 2013; reaffirm that CAVS algorithm testing for new DSA and RSA implementation using 1024-bit keys and/or SHA-1 will continue for key & signature generation through 2013 and continue for digital signature verification indefinitely.

Using the same logic that permits 80-bit signature verification after 2010, recommend reinstituting FIPS-approved digital signature verification, and associated algorithm testing, for DSA keys of < 80-bit security. (The 512, 576, 640, 704, 768, 832, 896, and 960-bit keys were revoked and algorithm testing removed as of May 19, 2007)

Clarify the definition of "Entity Authentication" and, if appropriate, point out its impact on PIV Validated modules.

Consider issuing new FIPS 140-2 guidance that requires vendors to add guidelines to the Security Policy for recommended digital signature strength vs. planned digital signature lifetime (based on the key strength table and recommendation for SP 800-57)

Section 4 - Random Number Generation

What is the basis for the claim that non SP800-90 RNGs support < 112-bit security strength? Note that NIST Special Pub (SP) 800-57, table 3 claims that SHA-1 used for random number generation provides 128-bit security.

Section 5 – Key Agreement Using Diffie-Hellman and MQV

A reference to minimum allowable key size should be added for Diffie-Hellman and MQV for both FF and EC based keys. SP 800-57, table 2, shows how key-pair size maps to security strength. (For example DH needs a 2048-bit public & 224-bit private key to achieve 112-bit security strength)

Section 6 – Key Agreement and Key Transport Using RSA

A reference to minimum allowable key size should be added for RSA. SP 800-57, table 2, shows how key size maps to security strength. (For example RSA needs a 2048-bit key to achieve 112-bit security strength)

Section 9 – Hash Functions

Recommend that, due to the complexity of determining if SHA-1 is used for any functions that are permitted after 2010 (Digital Signature Verification, Entity Authentication, Integrity Test, Key Derivation) that no SHA-1 certificate be revoked. Instead state that a warning will be applied on the CAVP validation list to all SHS certificates that contain SHA-1, declaring that SHA-1 is non-conformant for digital signature generation or hashing after 2010.

General - Message Authentication

This discussion paper needs to cover what, if any, impact the December 31, 2010 deadline has on FIPS-approved message authentication algorithms.

* CCM
* CMAC
* HMAC: all HMAC algorithms (including HMAC SHA-1) are allowed past 2010. [Based on SP 800-57, Table 3: Hash function security strengths for cryptographic applications]
* Triple-DES MAC

From: n.vijayarangan@tcs.com [mailto:n.vijayarangan@tcs.com]
Sent: Friday, July 31, 2009 9:32 AM

We are happy that NIST has come up with The Transitioning of Cryptographic Algorithms and Key sizes. We have the following comments:

Page 1, Paragraph 5, lines 7-9, "In some cases, an algorithm or protocol that has not been approved in a FIPS or NIST Recommendation is "allowed"; "

We recommend this sentence to be:

Algorithms or protocols that are not listed in FIPS or NIST Recommendation are exceptionally allowed through Public evaluation.

SHA-1 hash algorithm can be strengthened via Message Preprocessing methods (MP). Until SHA-3 algorithm is finalised, we can use SHA-1 via MP. This MP technique shows that SHA-1 is efficient. Michael Szydlo (RSA Security), Yiqun Lisa Yin (Independent Consultant) and Natarajan Vijayarangan (TCS Innovation labs) have contributed that Hash algorithms via MP are simple and secure.

In page 9, the following modification is to be incorporated in Table 4: Hash functions transitioning

Hash algorithm New Validations

SHA-1 via MP OK for all hash function applications SHA-1 via MP through December 31, 2012

Page 3, under the heading ECDSA, "FIPS 186-3 allows the generation of alternative curves, using methods specified in ANS X9.62."

We recommend this sentence to be:

FIPS 186-3 allows the generation of other Recommended Elliptic curves, using methods specified in ANS X9.62.

Regards

Natarajan Vijayarangan
TCS Innovation labs Hyderabad
India

From: David McGrew [mailto:mcgrew@cisco.com]
Sent: Friday, July 31, 2009 6:18 PM

Thanks to NIST for writing this tranition document and for soliciting comments on it. It is very useful to have all of the issues gathered together. I hope that the following comments are useful.

Section 3.2, "Security Strengths for Digital Signature Keys"

It is not clear what the rationale is behind the allowance of 80-bit signature verification beyond the time that 80-bit signatures may be generated. Certainly it is expedient, because it will be difficult for users to re-sign data with larger keys. But it seems inconsistent to forbid the generation of 80-bit signatures, since it is only during the verification process that an adversary can perpetrate a forgery. Preventing legitimate users from generating signatures with 80-bit strength will not prevent an attacker from doing so. Perhaps there is some unstated intent that would remove thia apparent inconsistency, such as a plan to disallow the verification of signatures with 80-bit strength in the future. I suggest that the rationale be added or referenced.

An issue related to the preceeding one is the quesiton of whether or not an implementation performing a signature validation would be required to ascertain when a digital signature was created.

It would be good to clarify what Entity Authentication means. Would this include signatures in public key certificates, for example?

Section 5, "Key Agreement Using Diffie-Hellman and MQV"

Page 7 says that "... any new versions of these protocols using DH and MQV key agreement must be designed to conform to SP 800-56A." It seems from the context that "these protocols" would include protocols like IKE and TLS, whose normative definition is controlled by standards-generating organizations that have international scope. It would be expecting far too much to require that implementers who aim to conform to NIST's crypto specificiations will somehow influence the generation of future standards in such a way as to ensure that they all conform to SP 800-56A. It seems better to replace this statement, perhaps with one such as "... the suitability of any new versions of these protocols, or any completely new protocols, which do not coform to SP 800-56A, will need to be individually assessed. The use of SP 800-56A is strongly encouraged."

Table 4 omits Secure RTP. The KDF for this protocol is allowed in the current Implementation Guidance, but it is not listed in this table, perhaps as an inadvertent omission. It is highly desirable to include the SRTP KDF in the transition plan.

I would guess that the comments on the TLS KDF are meant to apply to its use in DTLS as well as TLS; it would be useful for their document to clarify this.

Section 8, "Key Wrapping". The CMVP Implementation Guidance allows the use of key wrapping as defined in the GDOI Group Key Management Protocol, but this method is not mentioned in the transition document. It would be useful to develop more detailed guidance on the use of GDOI key wrapping, as related to key sizes and cryptographic parameters. It would also be valuable to add other key wrapping mechanisms, such as those using encryption or authenticated encryption methods that are already well specified and commonly supported, though in making this recommendation we are probably going beyond the scope of the transistions document.

Best regards,

David McGrew, Fellow
Cisco Systems

From: Hsieh, George [mailto:George_Hsieh@sra.com]
Sent: Monday, August 03, 2009 8:06 AM

This discussion paper offers little information on NIST's plans around key establishment schemes briefly described in the open paragraph of section 4 in SP 800-56A. Draft SP 800-56B offers more information about key agreement scheme (KAS) and key transport scheme (KTS) in sections 8 and 9 respectively.

Now that CAVP has added a KAS validation list, will a KTS validation list materialize soon? Perhaps the next iteration of this paper (or a different discussion paper) can offer guidance for mapping KAS or KTS validation results in conjunction with FIPS 140-2 (or 140-3) validation for IPSec, TLS and other protocols used to establish a secure cryptographic channel.

I hope the final release of this document would provide a transition plan for key management use cases as described in draft SP 800-57 Part 3: Application-Specific Key Management Guidance. In the case of transport layer security (TLS), section 4.2.2 *Recommended Cipher Suites for Federal Government Use* contains tables of cipher suites combinations. These tables are more descriptive than section 4.2.2 of SP 800-113 mentioning "ephemeral Diffie-Hellman key establishment used by TLS version 1.0 and later do not meet requirements of NIST SP 800-56A". This section also mentions NIST's exception for SSL until at the end of 2010 and subject to further analysis. Will the final release of this paper state confirm the exception for SSL will terminate at the end of 2010 or the exception will be extended further some number of years?

Respectfully,
George Hsieh

From: Vijay Bharadwaj [mailto:Vijay.Bharadwaj@microsoft.com]
Sent: Monday, August 03, 2009 5:41 PM

Thank you for preparing this document, and for the opportunity to provide feedback. It answers a number of questions about how the 112-bit transition will work in practice. However, we do have a few comments and questions:

- At a high level, there are at least two audiences that are affected by the 112-bit transition – the vendors who go to CMVP to get their products validated, and the various government agencies that must comply with the FIPS guidelines as part of their FISMA obligations. This document has a fair bit of information for the former, but it lacks any guidance regarding next steps for the latter. What should agencies be thinking about when procuring new products between now and 2011? What should they be doing about existing products which may be affected by the 2011 transition, and how will this affect their FISMA compliance? Will FDCC be updated to account for the transition?
    - It may be that NIST considers such information to be more appropriate for a separate document or publication. Any information clarifying NIST's thinking in this regard would be appreciated.
- Related to the above point, the document should do more to address the issue of access to existing data that was protected using algorithms that are now retired. Specifically, while Section 3.2 does say that verifying signatures with 80-bit strength is okay in FIPS mode, Section 2 does not say whether 2-key 3DES decryption be allowed in FIPS mode, and Section 9 does not say if SHA-1 will be permitted for verification purposes in FIPS mode. We believe this is necessary, since expecting all existing data to be re-protected with new algorithms in 2011 would be impractical.
- The document touches on all the classes of FIPS-approved algorithms, but leaves out the message authentication algorithms such as HMAC. Even if no algorithms of this class are affected by the transition, it may be useful to add a note to that effect for clarity.
- The paper talks about security strengths and how it determines what protection needs to be used for a given piece of data. However, it does not offer any guidance as to how and when security strength of algorithms is reassessed and what people should do when that happens. This is particularly relevant today, as new attacks are found against AES-256, which would seem to reduce its security strength.
- Section 3.2 makes a distinction between different uses of digital signature algorithms, and Section 9 makes a similar distinction for hash functions. How should we apply these guidelines to generic crypto modules (such as software modules) which are typically not aware of the use for which they are being called?
    - Will the CMVP require applications for future certificates to identify the specific uses for which they use signature algorithms? If not, how does the CMVP plan to identify how each cryptographic module uses these algorithms?

- Section 3.2 seems to imply that existing certifications for RSA and DSA key lengths less than 2048 bits, as well as ECC key lengths less than 224 bits, will be invalidated starting in 2011 for all uses other than entity authentication. However, this is not spelled out explicitly. More details on which certifications will be affected, in terms of hashes and key sizes, would be welcome.
- Some of the later sections, such as Section 6 and Section 7, do not appear to be related to the crypto transition at all. Instead, they seem to be talking about various crypto-related work items that NIST may take on in future. It may be useful to add some explanation of why these are related to the crypto transition, or to move them to a different document altogether.
- The various sections are not consistent in their level of explanation about how existing certifications will be impacted by the transition. Section 2 and Section 3.1 are very specific, Section 3.2 and Section 9 are less so. It may make the document easier to read if an explanation of the proposed process (i.e. stop new certifications, remove existing ones) was moved to its own section and it was clarified that this applies to all the affected classes of algorithms.

Regards,

--
Vijay Bharadwaj
Senior Program Manager, Windows Security
Microsoft Corporation
One Microsoft Way
Redmond WA 98052

From: Iorga, Michaela [mailto:michaela.iorga@nist.gov]
Sent: Monday, August 03, 2009 6:26 PM

DISCUSSION PAPER:
The Transitioning of Cryptographic Algorithms and Key Sizes

Comment [m1]: Suggest enhance title to be more explicit: e.g. Guidance to Federal Gov. on Transitioning …

1 Background and Purpose

At the beginning of the century, NIST began the task of providing cryptographic key management guidance. This included lessons learned over many years of dealing with key management issues, and attempts to encourage the definition and implementation of appropriate key management procedures, to use algorithms that adequately protect sensitive information, and to plan ahead for possible changes in the use of cryptography because of algorithm breaks or the availability of more powerful computing techniques. This guidance is provided in NIST Special Publication (SP) 800-57.

Comment [m2]: At the beginning of the century the organization was named NBS – it became NIST only in 1988.

Comment [m3]: Suggest rewriting it with more explanations (known attacks, theoretical/academic attacks/breaks, etc). "algorithms breaks" reads very … discouraging, concerning.

Some of the guidance provided in SP 800-57 includes the definition of security strengths, the association of the approved algorithms with these security strengths, and a projection of the time frames during which the algorithms could be expected to provide adequate security. The guidance often relies on the length of the cryptographic keys as an integral part of these determinations.

The security strength is measured in bits and is, basically, a measure of the difficulty of discovering the key. The understood security strength for each algorithm is listed in SP 800-57. For example, RSA using a key length of 1024 bits (i.e., 1024-bit RSA) has a security strength of 80 bits, as does two-key Triple DES, while 2048-bit RSA and three-key Triple DES have a security strength of 112 bits. See Table 2 in Part 1 of SP 800-57 for further security strength information.

Comment [m4]: Suggest more information added. Based on the difficulty of the brut force attack or known theoretical or practical attacks.

The selection of an algorithm that provides the adequate security strenghtdepends on the sensitivity of the data being protected, and needs to be determined by the owner of that data. For the Federal government, a minimum security strength of 80 bits is currently required. However, a minimum security strength of 112 bits is planned in 2011 as indicated in Table 4 of SP 800-57, Part 1. We've learned that this is not so easily done. The reality is that we need to examine each class of algorithm and sometimes make some adjustments.

One of the means for enforcing the algorithm and security strength requirements is NIST's Cryptographic Module Validation Program (CMVP), which is responsible for validations cryptographic modules for conformance to FIPS 140-2. To be validated, each module requires at least one cryptographic algorithm that has been approved for Federal government use. Approved security functions (i.e., approved cryptographic algorithms) are listed in Annex A of FIPS 140-2. n some cases, an algorithm or protocol that has not been approved in a FIPS or NIST Recommendation is "allowed"; an algorithm is indicated as allowed by means of the FIPS 140-2 Implementation Guidance document. The CMVP has defined two classes of modes for cryptographic modules: the FIPS mode and the non-FIPS modes for cryptographic module operation; in general, the FIPS mode uses only approved or allowed algorithms.

Comment [m5]: CMVP is a join program between NIST and CSE Canada. It validates modules for conformance, but it is nit a "mean" of enforcing the security strength. It is a program not the legislative or normative act or document. Please rewrite the statement.

Comment [m6]: How about CAVP – the program that validates the algorithms before CMVP can validate the module.

Comment [m7]: Or latest (FIPS 140-3 is ready for the second public review)

This paper is intended to bring some of the transition issues associated with the use of cryptography to the attention of the Federal government and the public, and to obtain feedback about the proposed approaches. **Please provide comments to CryptoTransitions@nist.gov by August 3, 2009.**
The general approach for transitioning from one algorithm or key size to another is addressed in SP 800-57, Part 1. The remainder of this paper addresses transition issues from the point of view of the CMVP.

2 Encryption

Encryption is used to protect the confidentiality of sensitive information. Several algorithms are currently approved for the encryption of sensitive information by the Federal government:

- Triple DES -specified in SP 800-67. Triple DES generally uses three distinct keys, and it is referred to as three-key Triple DES. When only two distinct keys are used, the algorithm is referred to as two-key Triple DES. The security strength of the two-key Triple DES has been assessed at 80 bits[1], whereas the security strength of the three-key Triple DES has been assessed at 112 bits.
- SKIPJACK - specified in FIPS 185. The security strength of this cryptographic algorithm has been assessed at 80 bits.
- AES - specified in FIPS 197. This cryptographic algorithm has three approved key sizes: 128, 192 and 256 bits. The security strength of AEShas been assessed at 128 bits for AES-128, 192 bits for AES 192, and 256 bits for AES-256.

Comment [m8]: Encryption is used for more than just confidentiality. If section addresses only encryption for confidentiality, the title should indicate it and a separate statement, prior to it, should correctly list all cases where encryption is used: confidentiality, integrity, authentication and non-repudiation (basically introduce the reader to the following sections and paper's organization)

Comment [m9]: The suggested changes intend to keep the reader's focus on the fact that this is a list of currently approved crypto algorithms for Federal use

NIST is proposing the following transition schedule (see Table 1).

Table 1: Encryption Transitions

Encryption Algorithm	New Validations	Already Validated Implementations
Two-key Triple DES	Through 2010	Disallow after 2010
Three-key Triple DES	OK	OK
SKIPJACK	Through 2010	Disallow after 2010
AES-128	OK	OK
AES-192	OK	OK
AES-256	OK	OK

As of December 31, 2010, two-key Triple DES and SKIPJACK will no longer be validated for conformance to the lates FIPS 140 and accepted for use by the Federal government to protect sensitive data (see SP 800-57, Part 1). All CMVP certificates that have been previously issued will be modified to remove these algorithms from the approved list for the FIPS mode. Note that if no other approved algorithms are included in a cryptographic module, the certificate for that module will no longer be valid.

Comment [m10]: Is this paper co-authored by CMVP (NIST and CSE) to be able to talk about CMVP programmatic issues? I suggest to make it a recommendation to CMVP.

3 Digital Signatures

3.1 Transition from FIPS 186-2 to FIPS 186-3

Federal Information Processing Standard (FIPS) 186-3 specifies three algorithms for the generation and verification of digital signatures: DSA, ECDSA and RSA. FIPS 186-3 also includes methods for generating key pairs and domain parameters, as required. FIPS 186-3 incorporates the following changes:

- General:
  - Specifies the use of all hash functions provided in FIPS 180-3, rather than just SHA-1,
  - Provides requirements for obtaining assurances of domain parameter validity (DSA and ECDSA only), public key validity, and private key possession,
  - References SP 800-57 for guidance on key management, including the key sizes and security strengths to be used,
  - Provides guidance on domain parameter and key pair management,
  - References SP 800-90 for random number generation, rather than including RNGs in the Standard, either explicitly or by reference to ANSI Standards,
  - Provides more guidance on the use of RNGs to generate key pairs,
  - Provides revised primality test guidance.

- DSA:
  - Specifies larger key sizes,
  - Replaces the domain parameter generation routine with new methods,
  - Includes explicit methods for the validation of domain parameters,

- RSA:
  - Approves the use of both ANSI X9.31 and PKCS #1, and provides guidance for their use,
  - Provides multiple explicit methods for the generation of key pairs,
  - Limits the key sizes and provides criteria for the generation of key pairs to be used for Federal government use.

- ECDSA:
  - Allows the generation of alternative curves, using methods specified in ANSI X9.62, while it still includes the Recommended Elliptic Curves

Since FIPS 186-3 only recently became official, a period of time must be defined for transitioning between FIPS 186-2 and 186-3. Some of the new tests required for testing against FIPS 186-3 are now available, while others are under development, and will be made available as soon as possible.

Comment [m11]: This is written as a mandatory requirement but no indication of whom is responsible to determine it, or what is the period of time

Implementations claiming conformance to FIPS 186-3 may now be submitted for validation by the CMVP/CAVP testing laboratories. However, those features for which tests have not been completed will be validated by vendor affirmation until the tests are available. New implementations designed to conform to FIPS 186-2 may be tested by the laboratories until December 31, 2010, after which only implementations claiming conformance to FIPS 186-3 will be accepted for testing and validation.

Certificates for implementations that have been validated against FIPS 186-2 will continue to be valid, subject to the requirements for appropriate security strengths, as discussed in

Comment [m12]: Changed for consistency across the entire document re: the tense.

Section 3.2. For example, implementations that provide security strengths of 112 bits or more will continue to be valid and operable in the FIPS mode, but those that provide only 80 bits of security will not2. Note that the invalidation of certificates will affect all DSA (currently) validated implementations, and those implementations of RSA and ECDSA that only use SHA-1 for digital signature generation for non-repudiation purposes.

In order to reach a larger audience, a Federal Register Notice will be published that requests comments about other issues that need to be considered during the transition from FIPS 186-2 to FIPS 186-3. Readers of this paper are encouraged to identify any issues that they foresee in order to prepare the Federal Register Notice with a realistic transition strategy.

3.2 Security Strengths for Digital Signature Keys

Digital signatures are used for several different purposes, such as:

- Data authentication (i.e., providing assurance about the authenticity of the signed data),
- Entity authentication (i.e., authenticating the identity of an entity (e.g., an entity that participates in a communication protocol),
- Determining that software or firmware has not been modified (see the integrity test on software and firmware in Section 4.6.1 in FIPS 140-2).

Comment [m13]: Non-repudiation is missing

When SP 800-57, Part 1 was written, the difference between these purposes was not fully addressed. While discussing the various uses of digital signatures in the government's identity cards and the timeframes in which currently existing implementations could be updated to be compliant with the NIST-recommended security strengths, and in the design and validation of FIPS 140-2-compliant cryptographic modules, the different use cases of digital signatures have been identified. The guidance provided in SP 800-57, Part 1 will be revised to recognize these differences.

Comment [m14]: Are these the only instances or are these examples of instances where the

NIST is proposing the following for the CMVP to address the aforementioned nuances for using digital signatures (see Table 2).

Comment [m15]: CMVP is NIST and CSE. So

Purpose	**Digital Signature Process**	**New Validations***	**Already Validated Implementations***
Software and firmware integrity test	Signature generation	≥ 80 bits through 2010 ≥ 112 bits after 2010	Validated implementations continue to be OK
Data Authentication	Signature verification	≥ 80 bits is OK	< 112 bits disallowed after 2010 ≥ 112 bits OK
	Signature generation	≥ 80 bits OK through 2010 ≥ 112 bits OK after 2010	
	Signature verification	≥ 80 bits is OK	Validated implementations continue to be OK
Entity Authentication	Both signature generation and verification	≥ 80 bits OK through 2013 ≥ 112 bits OK after 2013	< 112 bits disallowed after 2013 ≥ 112 bits OK

Table 2: Digital Signatures Transitions

Comment [MI16]: How was this date determined? I personally believe that a malicious user that gets access to the data abusing a security breach in the user authentication protocol can compromise more than one piece of data...

* Given in bits of security (i.e., security strength)

Comment [MI17]: Suggest to move it to footnotes

Digital signatures that are intended to provide data authentication:

Comment [m18]: Is this a subtitle or a member of a list? Please make the intent clear

- Signature generation: New implementations must generate digital signatures with a security strength that is equal to or greater than 80 bits through December 31, 2010; thereafter, the digital signatures must be generated with a security strength that is equal to or greater than 112 bits. Beginning in 2011, the generation of digital signatures with less than 112 bit of security strength will no longer be considered valid for the FIPS mode on a FIPS 140-2 validation certificate.

- Signature verification: New implementations may verify a signature that provides a security strength that is equal to or greater than 80 bits. Already validated implementations may continue to verify signatures at security strengths that are equal to or greater than 80 bits in the FIPS mode for the foreseeable future.

Digital signatures that are intended to provide only entity authentication: Until December 31, 2013, the minimum security strength required is 80 bits for the generation or verification of digital signatures for entity authentication. Beginning in 2014, only those digital signatures that provide at least 112 bits of security can be used in the FIPS mode. Note that signature verification for entity authentication is performed immediately after signature generation; therefore. there is no requirement to retain a signature for later verification.

Comment [m19]: Is this a subtitle or a member of a list? Please make the intent clear

Comment [MI20]: What "immediately" means? Next millisecond, next second, next action as part of a hand-shake protocol?

Digital signatures that are intended to determine the integrity of software and firmware:

Comment [m21]: Is this a subtitle or a member of a list? Please make the intent clear

- *Signature generation*: New implementations must generate digital signatures with a security strength that is equal to or greater than 80 bits through December 31, 2010; thereafter, the digital signatures must be generated with a security strength that is equal to or greater than 112 bits. Already validated implementations that generate digital signatures with a security strength of 80 bits may continue to be used by a cryptographic module operating in the FIPS mode for the foreseeable future.

- *Signature verification*: New implementations that verify digital signatures may verify a signature that provides a security strength that is equal to or greater than 80 bits. Already validated implementations that verify digital signatures with a security strength of 80 bits or more may continue to be used in the FIPS mode for the foreseeable future.

.

4 Random Number Generation

Random numbers are used for various purposes, such as the generation of keys, nonces and challenges. Several random number generators (RNGs) have been approved for use by the Federal government. Until relatively recently, FIPS 186-2 was used as an approval vehicle for three of these RNGs: a generator based on the use of the SHA-1 hash algorithm, a generator based on a symmetric block cipher algorithm[3], and a generator that was specified in a standard developed by the American National Standards (ANS) Institute (i.e., in ANS X9.31, and previously in ANS X9.17). In 2007, a new set of RNGs were approved in SP 800-90 that provide higher levels of security than the older RNGs. NIST proposes the following transition schedule (see Table 3).

Comment [MI22]: I am not sure if ANSI is not more accurate than ANS… If googled, or searched on Wikipedia, ANS comes with everything but ANSI …ANSI search brings the right links fast. Is there a way to determine the

Table 3: Random Number Generation Transitions

Description	New Validations	Already Validated Implementations
SP 800-90 (HASH, HMAC, CTR, DUAL_EC)	OK	OK
SHA-1 (FIPS 186-2)*	OK through 2010	Disallow after 2015
Sym. Alg. (FIPS 186-2)*	OK through 2010	Disallow after 2015
X9.31 = X9.17*	OK through 2010	Disallow after 2015

[3] DES was specified as the symmetric algorithm for the RNG in FIPS 186-2. Since DES has been withdrawn, two and three-key Triple DES and AES can be used in place of DES as the core engine of the RNG (see http://csrc.nist.gov/groups/STM/cavp/documents/rng/931rngext.pdf).

* Design or guidance does not support 112-bit security strength

Cryptographic modules that implement the RNGs specified in SP 800-90 can be validated and used for the foreseeable future in cryptographic modules operating in the FIPS mode. Cryptographic modules that implement the three older RNGs will only continue to be accepted for validation until December 31, 2010. Cryptographic modules that have been submitted for validation prior to this date and validated as conforming to these RNGs can be used in the FIPS mode of operation until December 31, 2015.

Comment [MI23]: The implementation can only be "conforming to a standard or specification".

Comment [MI24]: How was this date determined?

5 Key Agreement Using Diffie-Hellman and Menezes-Qu-Vanstone

Key agreement techniques are used to establish keys between two entities that intend to communicate (e.g., the keys may be used later for encryption or message authentication, or for the generation of additional keys). Two families of key agreement schemes have been approved in SP 800-56A Revision 1: Diffie-Hellman (DH) and Menezes-Qu-Vanstone (MQV). Each has been defined over two different mathematical structures: finite fields (FF) and elliptic curves (EC).

6

Key agreement includes at least two steps: the use of an appropriate DH or MQV "primitive" to generate a shared secret, and the use of a key derivation function (KDF) to generate one or more keys from the shared secret. SP 800-56A Revision 1, contains approved DH and MQV primitives, and approved KDFs.
Many commonly-used protocols that perform key agreement use DH or MQV. Until recently, no conformance testing was performed on implementations of these protocols in cryptographic modules. Testing for validation purpose is now available for implementations that claim conformance to SP 800-56A Revision 1 (i.e., conformance to one of the DH or MQV primitives used to generate a shared secret and a KDF specified in SP 800-56A Revision 1). Other protocols that implement DH and MQV are allowed, but testing for conformance is not supported by NIST/CAVP. In the future, NIST will require that the labs test implementations of the DH or MQV primitives during cryptographic algorithm validation testing managed by NIST/CAVP. However, the KDFs that do not comply with the KDFs specified in SP 800-56A Revision 1 will not be tested.

Protocols are used for a very long time. When new versions of a protocol are designed and implemented, a vendor may need to include a capability to interoperate with the older protocols. Because of this, the older protocols (and the KDFs in particular) will continue to be allowed. However, any new version of these protocols using DH and MQV key agreement must be designed to conform to SP 800-56A Revision 1.

Comment [MI25]: Please rewrite. What is "very long time" one year, 5 years, 10 years?

NIST is proposing the following set of transition rules (see Table 4):

Table 4: Key Agreement (DH and MQV) Transitions

Scheme	New Validations	Already Validated Implementations
SP 800-56A Revision 1 primitives	OK[1]	OK
Non-tested DH and MQV primitives	OK through 2010	Test by 2014
KDFs:		
SP 800-56A Revision 1	OK[2]	OK
IKEv2	OK	OK
X9.42	OK	OK
X9.63	OK	OK
IKEv1	OK	OK
SSH	OK	OK
TLS (1.0, 1.1, 1.2)	OK	OK

Comment [MI26]: What does it mean? "submitted for conformance testing by SOME-DETERMINED-DATE, 2014

~~1~~ Currently, a primitive is tested only when the KDF complies with SP 800-56A. Plan to test all DH and MQV implementations of the primitives in the future.
[2] Now tested.

Comment [MI27]: Suggest to move to footnotes

Implementations that fully comply with SP 800-56A Revision 1 (i.e., both the DH or MQV primitive and the KDFs) will be conformance tested and validated for use by cryptographic modules in the FIPS mode of operation for the foreseeable future.

Other implementations of the DH and MQV primitives (i.e., non-SP 800-56A Revision 1 implementations) that have already been validated must have the primitive(s) tested by 7

Comment [MI28]: By whom? Re-tested? Validation already implies testing.

December 31, 2013 for the cryptographic module to use them in a key agreement scheme in the FIPS mode of operation after 2013.

Implementations of IKEv2, IKEv1, and the current versions of X9.42, X9.63, SSH, and TLS are allowed to be used by a cryptographic module in the FIPS mode of operation for the foreseeable future. Their DH and MQV primitives will be tested for conformance to the primitives in SP 800-56A Revision 1.

6 Key Agreement and Key Transport Using RSA

SP 800-56B specifies the use of RSA for both key agreement and key transport processes. Like key agreement, key transport is a form of key establishment; however, the method for establishing keys is somewhat different. Refer to SP 800-57, Part 1 and SP 800-56B for definitions. SP 800-56B is currently in draft form, but is expected to be published as complete in the near future. The transition issues associated with the validation of SP 800-56B implementations have not yet been addressed.

The requirements for protocols containing key transport schemes are addressed in the FIPS 140-2 Implementation Guidance, which states that the key transport schemes in SSL v3.1, TLS, PEAP, EAP-FAST and EAP-TLS may be used by a cryptographic module operating in the FIPS mode. Note that these schemes are not actually tested for conformance during cryptographic module validation. The key transport schemes use by these protocols listed above are based on the RSA algorithm. The continued acceptability of the key transport schemes in these protocols will be reassessed when SP 800-56B is completed.

Comment [MI29]: As per SP 800-56B, eventhough "protocols" would sound better to me.

Comment [MI30]: The dec 2008 version from our website is not marked as draft???

7 Deriving Additional Keys from a Single Key

SP 800-108 specifies key derivation functions that use a key (i.e., a key derivation key) to generate additional keys. The key derivation key could be:

- a key that was manually distributed (e.g., by a courier),.
- generated using an approved RNG, or
- obtained using a key agreement or key transport scheme (see Sections 5 and 6).

This specification allows a large variety of KDFs. At the present time, conformance tests against this specification have not been developed. In addition, FIPS 140-2 Implementation Guidance does not include a provision to allow key derivation using the SP 800-108 KDFs. Therefore, a new section of the FIPS 140-2 Implementation Guidance will be developed to allow key derivation using the KDFs specified in SP 800-108.

8 Key Wrapping

Key wrapping is the encryption of a symmetric key by another symmetric key for integrity protection (called a key wrapping key). Symmetric keys are used with algorithms such as Triple-DES and AES. See SP 800-57 for further information. At the present time, neither a FIPS nor a NIST Recommendation have been developed for key wrapping, although a specification for key wrapping using AES is available at http://csrc.nist.gov/groups/ST/toolkit/documents/kms/AES_key_wrap.pdf.

The FIPS 140-2 Implementation Guidance (IG) addresses key wrapping as defined above. The IG states that AES or Triple DES may be used to wrap keys using the above referenced specification. If Triple DES is used, then it **shall** be used in exactly the same way that is defined for AES, and both Two-key and the Three-key Triple DES can be used for key

Comment [MI31]: It has 3 parts

Comment [MI32]: Why is this in bolt? What is the value of the "shall" statement here if this is not a normative document?

wrapping. Note that since Two-key Triple DES will be disallowed after December 31, 2010 (see Section 2 above), it will also not be allowed for key wrapping after that date
8

.

9 Hash Functions

Five approved hash functions are specified in FIPS 180-3. The security strength for a hash function, which depends on its design and use, is provided in SP 800-57, Part 1. Discussions about the different use cases are provided in SP 800-107.

NIST is proposing the following transition rules for hash functions (see Table 4):

Table 4: Hash Function Transitions

Hash Function	New Validations	Already Validated Implementations
SHA-1	OK for all hash function applications through December 31, 2010	Digital signatures: see Table 2 Hash-only: Disallow after 2010 OK for all other applications
SHA-224	OK for all hash function applications	OK for all hash function applications
SHA-256		
SHA-384		
SHA-512		

The five hash functions listed above can be submitted for validation to NIST/CAVP and can be used for all applications through December 31, 2010. In the case of implementations that have already been validated by CAVP:

- When SHA-1 is used by a cryptographic module in the generation of digital signatures, see Table 2 for the continued use of the implementation.
- When SHA-1 is used by a cryptographic module for hash-only applications, the use of already-validated implementations is disallowed after 2010 in the FIPS mode of operation of the cryptographic module.
- For all other SHA-1 applications and for all applications using the other hash functions, the implementations that have previously been validated by NIST/CAVP may continue to be used in the FIPS mode of operation of the cryptographic module.

10 References

All references documents are available at http://csrc.nist.gov/publications/.

FIPS 140-2 Security Requirements for Cryptographic Modules, with Change Notices, December 2002.
FIPS 180-3 Secure Hash Standard (SHS), October 2008.
FIPS 185 Escrowed Encryption Standard, Feb 1994.
FIPS 197 Advanced Encryption Standard, November 2001.

9 10

SP 800-56A Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, March 2007.
SP 800-56B Recommendation for Pair-Wise Key Establishment Using Integer Factorization, DRAFT, December 2008.
SP 800-57 Part 1, Recommendation for Key Management: General, March 2007.
SP 800-67 Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, May 2008.
SP 800-107 Recommendation for Applications Using Approved Hash Algorithms, February 2009.
SP 800-108 Recommendation for Key Derivation Using Pseudorandom Functions, November 2008.
Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program
FIPS 140-2 Annex A, *Approved Security Functions*, Draft June 2009
FIPS 140-2 Annex C, *Approved Random Number Generators,* Draft October 2007
FIPS 140-2 Annex D, *Approved Key Establishment Techniques,* Draft January 2008
Available at http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf.

From: Joan Lozano [mailto:jlozano@infogard.com]
Sent: Monday, August 03, 2009 7:50 PM

InfoGard Comments

White Paper: The Transitioning of Cryptographic Algorithms and Key Sizes

(Posted July 2, 2009)

1 How will the information from this whitepaper be incorporated into or reconciled with FIPS 140-2 Implementation guidance, SP 800-57 or related NIST documents?

2 Please clarify whether the terms "new validations" and "already validated implementations" refer to algorithm validations through CAVP or module validations through CMVP. Page 2, paragraph 1 says "this paper addresses transition issues from the point of view of CMVP." But page 2, final paragraph says that "no new validations will be performed on these algorithms" which indicates the validations in the tables are for CAVP.

3 Please provide further clarification for the use of the term "already validated implementations" in Tables 1-4. Does "already validated implementations" cover the case where a vendor wishes to revalidate an existing implementation as described in Implementation Guidance G.8? For example, if a cryptographic module has a non-tested DH primitive (see Table 4), can that module continue to get revalidated with software upgrades conforming to IG G.8 until 2014 without the DH primitive test? Or will any revalidation trigger the DH test requirement, even if there are no security relevant changes?

4 A module with an existing validation makes security relevant changes as described in IG G.8, could those changes trigger a "new validation" which would in turn trigger the requirements under "new validation" in Tables 1-4. For example in an embedded software/hardware module, if a vendor wanted to upgrade a cryptographic library but use the same algorithms, or if a vendor wanted to upgrade a cryptographic library using the same algorithms as before but added an SP800-90 compliant RNG to replace an older RNG, could that trigger a "new validation" that would then bring in the additional requirements show in Tables 1-4 under "new validations?

5 Please clarify whether NIST plans to provide a standalone DH and MQV test program under CAVP. CAVP currently covers a Key Agreement test for DH/MQV with KDF. Will the CAVP provide a standalone DH test that allows the vendor to test DH with an existing non-tested KDF (e.g., IKEv1)?

6 Please clarify Table 4 with respect to the KDFs being OK for new and already validated implementations. For example, if 2-key Triple DES were used in one of the listed KDFs (perhaps it is not), does this mean the KDF is still “OK”? Does Table 4’s “OK” to KDFs override the transition status of the underlying algorithms?

7 Page 8, near the top states: “Implementations of IKEv2 and IKEv1 … are allowed in the FIPS mode for the foreseeable future.” Table 4 is consistent with this, since the IKEv1 KDF is shown as OK for New Validations and Already Implemented Validations. However, IG 7.1 (pg. 71/72) lists IKEv1 KDF (Item C under Bullet 3) as having a 12/31/2010 time limit. How should readers interpret these two documents regarding IKEv1 KDF? What policy or recommendation change occurs in 12/31/2010, if any? Will the IG change?

8 Since SHA-1 will be allowed after December 31, 2010, but not for every scenario, how will the CAVP handle new SHA-1 certificates? How will the CAVP control or limit the new SHA-1 certificates?

9 There is no mention of the transition for Message Authentication algorithms (e.g., HMAC with SHA-1 or CMAC with 2-key TDES). These algorithms are discussed in SP 800-57, but for clarity, please add a section for them in the transition document.

10 In Section 4 Table 3, please add a row for the ANSI x9.62 RNG. The table would then show all RNGs that are currently listed in Annex C.

11 There is a contradiction regarding RNG transition. SP 800-57 Table 3 states that RNGs using SHA1 are OK for 2011, Table 3 of this paper states that old RNGs using SHA-1 will be discontinued in 2011, and Table 4 of this paper states "OK for all other applications" for SHA-1 (this includes RNGs). If SP 800-57 is incorrect in this case, please reiterate in Table 4 of this paper that some RNGs using SHA-1 won't be allowed in 2011.

12 The paper states, "…No new validations will be performed on these algorithms after that date, and CMVP certificates that were previously issued will be modified to remove these algorithms from the approved list for the FIPS mode. Note that if no other approved algorithms are included in a cryptographic module, the certificate for that module will no longer be valid." Is it correct to infer that CAVP validation testing ends on December 31, 2010? Is there a transition period prior to December 31, 2010 that CAVP validation of these algorithms will no longer be available? Is it correct to infer that CMVP acceptance of reports submitted with these algorithms listed as Approved ends on December 31, 2010? Is there a transition period prior to December 31, 2010 where reports will no longer be accepted? When will CMVP accept validation reports with TDES (KO =2) and Skipjack as Approved Algorithms with a caveat (e.g., "For support of legacy systems until December 31, 2010")?

From: Frankel, Sheila E. [mailto:sheila.frankel@nist.gov]
Sent: Friday, August 07, 2009 12:59 PM

1. Section 3.2: Definitions of data authentication and entity authentication: not a good idea to use the word that's being defined (authenticating) within the definition/description.

   Also: what is the difference between #1 (data authentication) and #3 (determining that ... has not been modified). Doesn't authenticating data provide integrity protection? Isn't #3 a special case of #1? if they are different, please clarify.

   Since the whole doc hinges on these terms (and different crypto communities use these terms in slightly different ways), more specific definitions would be good, possibly with examples of use.

   data authentication: should mention that it's also called integrity protection

   Later in the doc, data authentication is referred to once as "message authentication" - confusing.

2. HMAC is only mentioned under RNGs.

   Since there has been a lot of confusion about the use of SHA-1 vs. HMAC-SHA-1, it would be good to mention the use of MACs (as opposed to hash algs) for integrity-protection, and mention that HMAC-SHA-1 is OK.

   The hash function table (Table 4) says that SHA-1 is allowed for all other apps in already validated implementationss - I assume that that includes HMAC-SHA-1. But nowhere in the table does it state that new validations are OK for HMAC-SHA-1 after 2010.

3. FIPS mode - might want to mention that this is distinct from encryption algorithm "modes of operation"

4. Is 2-key triple DES used today in any known products/protocols? If not, should mention that fact.

On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography

version 2, August 7, 2009

Joppe W. Bos[1], Marcelo E. Kaihara[1], Thorsten Kleinjung[1], Arjen K. Lenstra[1,2], and Peter L. Montgomery[3]

[1] EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland
[2] Alcatel-Lucent Bell Laboratories
[3] Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

Abstract. Meeting the requirements of NIST's new cryptographic standards means phasing out usage of 1024-bit RSA and 160-bit elliptic curve cryptography (ECC) by the end of the year 2010. This write-up comments on the vulnerability of these systems to an open community attack effort and aims to assess the risk of their continued usage beyond 2010. We conclude that for 1024-bit RSA the risk is small at least until the year 2014, and that 160-bit ECC over a prime field may safely be used for much longer – with the current state of the art in cryptanalysis we would be surprised if a public effort can make a dent in 160-bit prime field ECC by the year 2020. Our assessment is based on the latest practical data of large scale integer factorization and elliptic curve discrete logarithm computation efforts.

Keywords: NIST Special Publication 800-57, Suite B Cryptography, 80-bit security, RSA, integer factorization, NFS, ECC, Elliptic curve discrete logarithm, Pollard rho

1 Introduction

In February of 2005, the United States' National Security Agency (NSA), arguing that *"The sustained and rapid advance of information technology in the 21st century dictates the adoption of a flexible and adaptable cryptographic strategy for protecting national security information"*, announced its *Suite B of Cryptographic Protocols* (cf. [31]). Along similar lines, the United States' National Institute of Standards and Technology (NIST) provides guidance *to use algorithms that adequately protect sensitive information, and to plan ahead for possible changes in the use of cryptography* in *Special Publication 800-57, Part 1* (cf. [29]; see also [30] and [27]). These initiatives are targeted broadly enough that the majority of vendors and major enterprises – worldwide, not just in the USA – will be more or less forced to follow them.

Roughly speaking, "Suite B" adopts 128-bit and higher security levels, leaving behind the current widely-employed 80-bit security level. It also prescribes a precise set of technologies to be used. NIST's "SP 800-57 (1)" allows more flexibility in the technology used and settles for 112-bit security at the lowest level from the year 2011 on. Although any migration – to other technologies, to a new security level, or to both – will be a logistical nightmare and has the potential for substantial security breaches unrelated to the cryptographic security levels, there can be little question that, ultimately, migration to a higher security level than

80 bits is mandatory. The migration should, however, be orchestrated with utmost care and too hurried adoption of the new standards should be avoided; even for the NSA itself it will take many years to comply (cf. [16]) and NIST, in [30], comments that *We've learned that this is not so easily done*. Thus, though both "Suite B" and "SP 800-57 (1)" are in principle welcome initiatives whose implementation by the year 2010 can be recommended, an important question is how much time companies and other complying practitioners can reasonably allow themselves before fully adopting either of the new standards, in a manner that is as cost effective and security conscious as possible.

Both initiatives affect all three pillars of modern cryptography: block ciphers, cryptographic hash functions, and public key cryptography. For block ciphers "Suite B" requires the Advanced Encryption Standard (AES), which was designed to provide 128-bit security or more. Although there are doubts about the adversarial model in which the intended security level of AES is reached, there is consensus that single DES (the classical Data Encryption Standard) can no longer be recommended even for low risk applications. "SP 800-57 (1)" also allows limited usage of three key triple DES. This write-up does not further deal with the subject of block ciphers or migration to AES. It does not deal with cryptographic hash functions either. But it may be good to remind the reader that the situation with respect to cryptographic hash functions is disconcerting. In the first place, there are concerns about the security provided by the current standard, the 160-bit hash function SHA-1 which is designed to provide 80-bit security. Convincing arguments have been presented that the security level of SHA-1 is quite a bit lower than that: right now it is not clear if the bottom is in sight yet. If chosen-prefix collisions as presented in [37] for MD5 could be made to work for SHA-1, attacks may be feasible that could have an undesirably negative impact on the security of the Internet as a whole. SHA-256, as required by "Suite B", may be regarded as an improvement over the common use of SHA-1. However, given the design similarities between SHA-1 and SHA-256 and the currently unforeseeable outcome of the NIST hash competition, the situation with respect to cryptographic hashing is unclear.

Here we concentrate on the third pillar mentioned above, the migration away from 80-bit security for public key cryptography. NSA restricts the use of public key cryptography in "Suite B" to elliptic curve cryptography (ECC). Compliance will be a challenge for applications that do not use ECC yet, because implementing ECC is non-trivial for a variety of reasons – because it involves mathematical structures that are much harder to understand and implement than those involved in RSA (cf. [34]), but also because various higher level system decisions need to be made, the consequences of which not even the *cognoscenti* seem to agree upon (cf. [20]). Finally, once one understands all mathematics and has decided which type of elliptic curves to use, it is still a challenge to figure out the best computational approach given one's platforms.

Where "Suite B" adopts 256-bit ECC as the lowest strength asymmetric system, NIST's "SP 800-57 (1)" allows 224-bit ECC and, more remarkably, continued use of RSA, but with 2048-bit moduli. Estimates vary as to what level of security is offered by 2048-bit RSA moduli. For the intended 112-bit security level, 2048-bit RSA may be regarded as satisfactory, depending on one's view on the security provided by 1024-bit RSA (cf. below). The security level of any of the newly standardized systems is, however, not the subject of this paper – indeed, we will not even try to define what precisely is meant by 'security level' (but see the remark at the end of this paper). Instead, we address a short term but nevertheless important question that

many practitioners will face the next few years, namely until when the current standards, 1024-bit RSA and 160-bit ECC, can responsibly be used: do businesses indeed have to adopt the new standards by the end of 2010, or can they allow for a smoother transition? For RSA this question, which includes the important practical concern by when digital certificates based on 1024-bit RSA should be phased out, is addressed in Section 2, for ECC in Section 3. For ECC we limit ourselves to curves over prime fields, since that is the focus of NSA's recommendations as well: throughout this paper, ECC refers to ECC over a prime field.

With a question of this sort we must say something about the adversarial model. Throughout this write-up we restrict our attention to work that has been or may be performed in the open research community. We are not privy to activities that take or may take place behind closed doors, such as at government laboratories, and therefore cannot include them in our considerations. When we say that a system can "responsibly be used", we mean that it must be considered unlikely that an academic, open community effort will be able to mount a successful attack. Obviously, this is by no means a formal definition and leaves many issues unaddressed (such as timing and funding level). Nevertheless, despite this vagueness, we strongly believe that a paper of this sort is useful for the cryptographic user community. We present the relevant current cryptanalytic state of the art and, combined with past developments, give our view on what can and what cannot be expected in the near future. More importantly, though, we present these data so that the reader can develop an understanding of the effort involved in breaking RSA or ECC for relevant parameter choices and draw his or her own conclusions.

For the purposes of the present write-up, we may assume that precautions are taken to avoid common pitfalls when implementing RSA or ECC (as surveyed for RSA in [6]). This implies that we may concentrate on attacks that put a firm upper boundary on the security of any implementation of RSA or ECC: attacks that are commonly referred to as *brute-force attacks* that attempt to factor the public RSA modulus or to compute a discrete logarithm in an elliptic curve group.

2 The security of 1024-bit RSA

Since the introduction of the RSA cryptosystem, the effectiveness of brute-force attacks has improved considerably, with noticeable impact on the choice of RSA modulus sizes. The most effective method that has been published, the *Number Field Sieve* (NFS, cf. [22]), is already more than 20 years old. Its development, impact, and properties relevant for the assessment of the security of 1024-bit RSA moduli are reviewed in this section. Since 1989, essentially nothing has happened in integer factorization, except for a seemingly endless stream of relatively small improvements. All these tweaks, particularly when combined, influence the effectiveness of NFS, but none of them represents a major new idea or breakthrough: factoring-wise, research is at a disconcerting or reassuring standstill. Notable exceptions are polynomial time factoring on a quantum computer [36], so far without practical implications, and polynomial time primality proving [1].

The original version of NFS, now referred to as the *Special Number Field Sieve* (SNFS), was expressly invented to fully factor the ninth Fermat number, $F_9 = 2^{2^9} + 1$, and succeeded doing so in 1990 (cf. [23]). SNFS crucially relies on the peculiar, "special" shape of the numbers it can handle, such as F_9, and in particular cannot be used to factor RSA moduli. It took a few

more years before SNFS was generalized to the somewhat slower method now known as NFS that *can* handle RSA moduli. The first public announcement of the NFS-factorization of a 512-bit RSA modulus was in 1999 (cf. [9]).

The latest record of SNFS factorization was that of the "special" number $2^{1039}-1$, found in 2007 (cf. [2]). An NFS-effort to factor a 768-bit RSA modulus is currently underway and is expected to finish in 2010. It may turn out to be a new NFS record that would beat the current factoring record of the 663-bit, 200-digit RSA modulus RSA-200 by more than 100 bits (cf. [3]). It is expected that the 768-bit factorization, when finished, will have required about 10 to 15 times the resources required for $2^{1039}-1$. This estimate is based on current practical findings, not on theoretical estimates or guesswork: both are unreliable when switching between SNFS and NFS. Extrapolating 768-bit NFS to 1024-bit NFS (as required for a 1024-bit RSA modulus) can be done using the heuristic asymptotic runtime estimate for NFS. Doing so in the usual "$o(1)$-less" fashion (cf. [25] and [21]), one finds that 1024-bit RSA moduli are about 1200 times harder to crack than 768-bit ones, the first example of which is "about" to be completed. We round this down to one thousand based on the argument that there is a practical discontinuity in the NFS runtime for 'small' numbers that is not adequately reflected in the asymptotic runtime estimate when the $o(1)$ is omitted. Numbers of 768 bits are relatively close to the lower end of the range where number fields of extension degree 6 are optimal, whereas 1024-bit numbers are fairly close to the middle of that same range. As a consequence, the $o(1)$-less estimate for 768 bits is low compared to the actual runtime, where for 1024 bits they can be expected to be much closer. Thus, it may be expected that the 1200 is too pessmistic and that one thousand is closer to reality. Furthermore, combining the numbers, it is estimated that a 1024-bit RSA modulus requires resources that are at most 10 to 15 thousand times larger than for the already completed $2^{1039}-1$ SNFS effort.

It may be argued that it is a bit of a stretch to extrapolate NFS factoring estimates from 768 to 1024 bits. But it is supported by a similarly calculated and relatively speaking even larger extrapolation from 512 to 768 bits: "$o(1)$-less" theoretical runtime extrapolation suggests that 768-bit RSA moduli are about 6 thousand times harder to crack than 512-bit ones, whereas empirical evidence based on best efforts using current software and hardware, suggests an average figure of 4 to 8 thousand.

Below, we estimate the resources currently required to tackle a 1024-bit RSA modulus, assuming rough familiarity with the major steps of an (S)NFS factorization. This is followed by a discussion of the security of 1024-bit RSA moduli in the near and not too distant future.

Polynomial selection. The result of this step (needed just for NFS, not for SNFS) greatly influences the time spent on the later steps, but the amount of computation spent here is small compared to the others: so far about 3% to 5% of the overall computation. It is fully parallelizable. There is still ample room for improvement in current methods (cf. [8]).

Sieving. Sieving allows full and virtually communication-free parallelization and can be run at any number of locations on any number of independent processors. It is and has always been the step that requires the bulk of the computation. For $2^{1039}-1$ it required about two hundred years of computing on a single 3GHz Pentium D core, for a 768-bit RSA modulus we have established that ten times that effort suffices. For a 1024-bit RSA modulus the sieving time extrapolates to 2 to 4 million core years.

For a 768-bit RSA modulus 2 gigabytes of RAM per core works nicely, half of it still works too but is noticeably less efficient. For 1024 bits, 2 gigabytes of RAM per core would cut it

too closely, but memories of 32 or 64 gigabytes per multi-core or multi-threaded processor (or even per mainboard) would work. These estimates take into account current processor and memory access speeds and are based on the fact that memory requirements grow roughly with the square root of the sieving time.

Filtering. Transforming the sieving data into a matrix is one of the least challenging steps of a factorization and does not need to be discussed here. For a 1024-bit RSA modulus it will require a few hundred terabytes of disk space, which is relatively easy to manage compared to the previous and next steps.

Matrix. The matrix requires time and memory growing as fast as it is for sieving. Although for all factorization attempts to date the matrix time has been modest compared to the sieving time, it has always been more cumbersome due to the distinctly smaller degree to which this step allows parallelization. The first time the step was parallelized over multiple locations was for the factorization of $2^{1039} - 1$ reported in [2], namely on two sets of clusters, one in Japan and one in Switzerland. Overall it took about 35 core years on a 3GHz Pentium D. For a 1024-bit RSA modulus between half a million and a million core years should suffice. For each tightly coupled cluster participating a combined memory of 10 terabytes should be adequate.

Final square root. The runtime of this step has always been insignificant compared to all other steps. There is no reason why it would be different for a 1024-bit RSA modulus.

We stress that the figures above are based not just on extrapolation of the data from [2], but on analysis and extrapolation of data of an ongoing 768-bit RSA modulus factorization effort for which the sieving has been completed. These figures thus represent the most accurate estimates published so far for a 'realistic' 1024-bit RSA modulus factoring effort.

We consider what it means for the security of 1024-bit RSA – now, for the next five years, and for the next decade. At this point in time a brute-force attack against 1024-bit RSA would require about two years on a few million compute cores with many tens of gigabytes of memory per processor or mainboard. Though substantial, this is not an inconceivably large effort. We conclude that the computation is in principle already feasible: processors with enough memory for the sieving do exist, and there are clusters that have the combined memory required for the matrix (though most will not have enough nodes). The total budget required is large, but smaller than what was spent on other dedicated scientific endeavors.

Nevertheless, as an "open community" effort a 1024-bit RSA factorization is currently out of reach. For how long will this be the case? As usual when we try to predict the cryptanalytic future, we study past trends and use those as a basis for our predictions. Thus, as factoring breakthroughs have not occurred for several decades, and polynomial time factoring on an actual quantum computer (cf. [36]) still seems to be several decades away, both these possibilities are discounted.

Next considering special purpose hardware, the most optimistic approach suggests that sieving for a 1024-bit RSA modulus can be done in a year for about US $10,000,000, plus a one-time development cost of about US $20,000,000, and with a comparable time and cost for the matrix [35]. In our opinion, (common) skepticism met by such designs is beside the point. Such figures should not be interpreted as upper bounds, i.e., "Be careful, 1024-bit RSA can be broken in two years for about twenty million bucks (assuming free development)," but as lower bounds, i.e., "No reason to worry too much: even under very favorable attack conditions,

factoring a 1024-bit RSA modulus still requires enormous resources." The estimates should thus not be read as threatening but as confidence-inspiring.

The above point aside, even if one manages to build special purpose hardware, by that time commodity hardware has already caught up, speed-wise or cost-wise – at least that has consistently been the case in the past, and is predicted to be the case until at least 2030 (cf. [17]). Eventually, this will change because special purpose hardware will be the only way to stay on Moore's curve, but that is in a future that is too distant for our purposes (cf. [17]).

Turning back to reality, the following events are relevant for our question when a 1024-bit RSA modulus can be expected to fall:

1988: First factorization of a hard 100-digit integer (cf. [24]).
1999: First factorization of a 512-bit RSA modulus (cf. [9]).
2010: Expected first factorization of a 768-bit RSA modulus.
20??: Expected first factorization of a 1024-bit RSA modulus.

Based on NFS runtime comparisons the three consecutive "difficulty gaps" between the above four factorizations are factors of about 2,000, 6,000, and 1,000. The last two numbers were already given above and the 6,000 was shown to be a close match to empirical evidence. It should be noted that the 100-digit factorization was not obtained using NFS but using its asymptotically slower predecessor 'Quadratic Sieve'. Thus, in reality, a larger hurdle than just a factor 2,000 was overcome in the period 1988-1999, mostly due to the invention and development of NFS starting in 1989. In any case, the first three entries above suggest that a thousand-fold increase in factoring difficulty can traditionally be overcome in a decade. This reasoning would naively lead to an expected first factorization by the open community of a 1024-bit RSA modulus approximately by the year 2020. But is it reasonable to expect that the developments that took place between 1988 and now will continue in the decades to come?

To explain the factor of more than one thousand per decade – as opposed to just observing it twice in a row in the past – a factor of one hundred every ten years can fairly consistently be attributed to Moore's law. The other factor is due to lots of gradual improvements in the polynomial selection step and the sieving and matrix software, probably combined with greater persistence and patience among the researchers involved. The type of compute power required for large scale NFS efforts will keep following Moore's law at least for the next two or three decades (cf. [17]), and realistic progress on algorithmic fronts has been announced as well (cf. [8]). Thus, there is enough reason to expect that over the next decade we will be able to overcome yet another factor of one thousand in factoring difficulty.

This explanation of the observed progress bolsters the confidence in the above naive prediction that in about 10 years the first open community 1024-bit RSA modulus factorization can be expected – assuming by then volunteers can be found to take up the considerable challenge. In the next paragraph, and to conclude this section, we consider the question whether we can expect a 1024-bit RSA modulus factorization much sooner.

Assume that the 768-bit RSA modulus factoring attempt will be completed in the year 2010. The factor of one hundred provided every decade by Moore's law implies a factor of ten every five years. Thus, to be able to factor a 1024-bit RSA modulus in 2015 ($= 2010 + 5$) the combined algorithmic improvements would have to accumulate to a factor of about one hundred well within the next five years. Such progress would be unprecedented unless a major

breakthrough occurs. Given the unlikelihood of the latter, an open community effort that would factor a 1024-bit RSA modulus cannot be expected by the year 2015. This conclusion is compounded by the observation that the current 768-bit effort already stretches the resources of open community factoring enthusiasts: it is virtually inconceivable that a thousand-fold larger effort would be feasible in just five years. Obviously not a hard argument, but a more intuitive one – steeped in decades of leading edge integer factorization experience.

3 The security of 160-bit ECC

The most effective brute-force method to compute a discrete logarithm in a prime order elliptic curve group is Pollard's rho method (cf. [33]). We assume that this group order and the order of the underlying prime field have the same bitlength (as is for instance the case for the curves in [28, Section D.1.2]). Pollard rho comes with a whole lot of embellishments, some with major, others with relatively minor impact. Most importantly, the method allows full parallelization on any number of very loosely coupled processing elements that require only occasional communication with a central server (cf. [39]). Denoting the group order by q, each Pollard rho process needs memory for a small constant number of $(\log_2 q)$-bit numbers. Thus – and this in huge contrast to NFS-based brute force attacks against RSA – virtually any processing element can, in one way or another, contribute to an ECC attack. Furthermore, if a processing element allows multi- or hyperthreading or any other fancy type of parallelization or pipelining, this can in principle be taken advantage of by running as many Pollard rho processes in parallel as required to keep the device fully occupied.

This very 'lightweight' parallelization changes the applicable threat model compared to our estimates of the resources required to attack 1024-bit RSA. In particular we need to examine whether the most recently published result on ECC brute-force attacks still adequately represents current possibilities: in 2002 it took "10^4 computers (mostly PCs) running 24 hours a day for 549 days" to solve an elliptic curve discrete logarithm problem over a 109-bit prime field (cf. [11]). The large team performing this calculation used Chris Monico's Pollard rho implementation, with highy optimized assembly code tuned to a wide variety of PC platforms. Naively accommodating for 5 'Moore doublings' over a period of 5 times 18 months, but increasing the difficulty by 100M (which approximates $\left(\frac{160}{109}\right)^2 \cdot 2^{(160-109)/2}$, taking into account that the number of group operations grows with the square root of the group size, combined with the quadratically growing group operation cost itself) Monico's result extrapolates to about 50 billion years to attack 160-bit ECC on a current core. Obviously, this would be a daunting task. Our question is if this estimate is valid given current multi-core processors, and how the required computational effort can be realized in the most economical fashion. For this purpose we consider, in the remainder of this section, a number of different platforms that may be used to attack 160-bit ECC using Pollard's rho method.

Pollard's rho method was designed to calculate discrete logarithms in multiplicative groups of prime fields. We already mentioned that it allows ideal parallelization. This is based on the use of *distinguished points* as described in [39]. Furthermore, the *negation map* (cf. [14], [40]), *adding walks* (cf. [38]) and *tag-tracing* (cf. [13]) make the calculation somewhat faster. As described in [13] tag-tracing does not immediately apply to elliptic curve groups, but has been shown in [7] to give a moderate speed-up there as well. According to [7], tag-tracing does not seem to be compatible with usage of the negation map or, more in general, the use

of equivalence classes (applicable mostly to curves over small characteristic extension fields, and thus of no concern here). For elliptic curves over prime fields each of these improvements does not affect the overall expected runtime by more than 40%, and is therefore hardly of concern for our primary purpose to get an order of magnitude estimate.

With elliptic curves there is the issue how to represent group elements: affinely requiring only a few multiplications plus an inversion (all in the prime field) per group operation, or projectively to avoid the inversion at the cost of more multiplications. Recognition of distinguished points, however, enforces a unique representation and thereby an inversion. In circumstances such as the present one where many processes can be run simultaneously, their inverses can be shared at the cost of, roughly, 3 additional multiplications per curve, using the simultaneous inversion from [26]. As a result, when processing N processes in parallel and using affine Weierstrass representation, point addition takes four multiplications, one squaring and $\frac{1}{N}^{\text{th}}$ inversion (all in the prime field) for the x-coordinate, plus one multiplication for the y-coordinate.

For a variety of platforms, we consider best-effort implementations, either from the literature or derived by ourselves, of Pollard's rho method in elliptic curve groups, while making use of full parallelization, adding walks, affine point representation with simultaneous inversion, and possibly other improvements if compatible with the platform and other implementation aspects. The platforms in question, in rough order of user-friendliness, are the following: regular desktops, Sony PlayStation 3 game consoles (PS3), graphics cards, FPGAs, and ASICs.

Desktop. The fastest actually working desktop implementation of Pollard rho on elliptic curves that we are aware of is reported in [7]. It focuses on the 131-bit prime field ECC challenge from [10], implemented as indicated above and processing 200 walks per core. For generic 131-bit primes, an expected overall runtime of 10^5 years is reported on an AMD Phenom 9500 quad-core 64-bit processor with a clock frequency of 2.2GHz, i.e., about $4 \cdot 10^5$ core years. A 25% speed-up can be obtained due to the particular size of the 131-bit prime used. Both the regular and the speeded-up implementation use assembly code. It should be noted that [7] uses tag-tracing as opposed to the negation map. Swapping these may lead to an overall speedup of about 20%.

Extrapolation of the generic runtime to 160-bit primes, in the same manner as the runtime reported by Chris Monico was extrapolated, yields $\left(\frac{160}{131}\right)^2 \cdot 2^{(160-131)/2} \cdot 4 \cdot 10^5 \approx 10^{10}$ core years. We stress that this runtime is realistic and that 2.5 billion current workstations should be able to crack 160-bit ECC within a time-frame of 6 to 24 months (i.e., give or take a factor of 2, depending on which embellishments are used or which unexpected snags will occur). This is about 5 times faster than what naive extrapolation of Monico's result made us believe, but still not a task that is feasible anytime soon for an open community effort – indeed, even for a well sponsored agency it would be a formidable task. Below we consider whether it can be done 'cheaper.' The factor 5 may be attributed to the fact that our type of application heavily relies on integer multiplication which is, relatively speaking, much faster on the current 64-bit processors than on the 32-bit ones from back in 2002.

PS3. The Sony PlayStation 3 game console is powered by a Cell processor, a powerful unit that is described in more detail below. The Cell's processing power can be unlocked for general applications, i.e., not just games, using Sony's hypervisor, thereby making the PS3 a relative inexpensive source of processing power. Actually using this processing power, however, in

particular for the type of application that we are interested in here, is not straightforward: it is hard to optimize multi-precision integer arithmetic on regular processors, it is not easier on the peculiar Cell architecture. A first approach is to use IBM's off-the-shelf MPM library for long integer arithmetic (cf. [18]). Unfortunately the performance did not live up to what we had expected given the Cell's specifications, lagging behind by about an order of magnitude.

Thus, we set out to design our own implementation, geared toward optimal performance for our application to Pollard rho for elliptic curve groups over relatively small prime fields. We give a rough outline of the design of our arithmetic functions. For further details we refer the interested reader to the Appendix.

Omitting many details, the Cell has a main processing unit (the PPE, a dual-threaded 64-bit Power Processing Element) which can offload work to eight Synergistic Processing Elements (SPEs). The for us relevant part of the SPE is the Synergistic Processing Unit (SPU). Each SPU runs independently of the others at 3.2 GHz and works on its own 128 registers of 128 bits each and has its own 256 kilobyte Local Store for its instructions and data. The 128-bit registers allow SIMD operation on sixteen 8-bit, eight 16-bit, or four 32-bit integers. Finally, the SPU has two pipelines, an odd and an even one, which means that two instructions (one odd, one even) can be dispatched per clock cycle per SPU. Although it is rich in boolean operations, its integer multiplication facilities are somewhat limited, cf. Appendix. There is no smart branch prediction on the SPU. It is therefore advisable, as customary in SIMD anyhow, to avoid branching as much as possible.

Of the 8 SPEs one is disabled and one is reserved by Sony's hypervisor. Thus, when running Linux (under the hypervisor), 6 of the 8 SPEs are accessible. Because each SPU can be regarded as a four-way SIMD 32-bit processor (with limited multiplication possibilities) and runs at 3.2 GHz, it may under favorable circumstances be possible to squeeze performance equivalent to $4 \times 6 = 24$ 'regular' (but 32-bit) cores out of a single PS3. As reported in [37], sometimes even more is possible. We did not manage to achieve this for our elliptic curve Pollard rho application, but we got quite close, as implied by the numbers below.

More precisely, we obtained the following results. We focussed our attention on *curve number 6* from [12], which is a curve over a 112-bit prime field and which is, as of 2009, the 'smallest' elliptic curve standardized for cryptographic purposes. To minimize the inversion overhead, we processed as many walks per SPU as permitted by the size of the Local Store. This turned out to be 400 for the 112-bit prime field, thus we could use $N = 400$ for the simultaneous inversion. The 400 walks are organized in 100 sequential (but synchronized at inversion time) batches of 4 walks that are processed simultaneously by performing the underlying field arithmetic in a 4-way SIMD fashion. It turned out that for the specific 112-bit prime certain computational advantages could be obtained (cf. Appendix) that do not translate to generic primes of the same size. For generic primes the arithmetic is at most 20% slower.

For the special 112-bit curve we found that the overall calculation can be expected to take approximately 60 PS3 years[4] . Since we use just 6 SPUs, better performance can be obtained if we would use the PPU too, or if we would be able to access the 7^{th} SPU. Extrapolating while including a 20% generic slowdown (and also accounting for a lower value of N for the larger prime), we find $\left(\frac{160}{112}\right)^2 \cdot 2^{(160-112)/2} \cdot 1.2 \cdot 60 \approx 2.5 \cdot 10^9$ PS3 years to crack 160-bit ECC.

[4] As reported on `http://lacal.epfl.ch/page81774.html` this calculcation was completed on July 8, 2009, taking the predicted amount of time. A description of the full details of the computation is in preparation.

Comparing to the number of desktops calculated earlier, we again find that a year on 2.5 billion devices suffices. For the current application 6 SPUs are computationally apparently more or less equivalent to a quad-core 64-bit processor, i.e., a single SPU is equivalent to 66% of a single 64-bit core. The latter is better at integer multiplication, which outweighs, for this application, the advantage of the 4-way 32-bit SIMD SPU architecture. As mentioned above, for other types of applications the advantage may be on the other side.

It is debatable which of the two options – desktops or PS3s – is more economical. The relevant parts for a desktop can be purchased for about the same price as a PS3, but the latter is ready to go and the former needs to be assembled, billions of times if one gets serious about an ECC attack. The PS3 comes with a Blu-ray player that is not needed, but is attractively priced just because of it, so getting the non-assembled relevant parts of a PS3 may actually turn out to be more expensive (cf. relatively high cost of Cell-blades). We leave quantum-discounts for an order of several billion devices of either type to the reader's negotiation skills. Note that the PS3 also comes with a rather powerful graphics card which is unfortunately, with the present version of the hypervisor, not accessible to applications other than gaming. If it were accessible, it could double the effectiveness of a PS3 (cf. below). As is, the balance may already tip in favor of the PS3, compared to a desktop, if we would also involve the PS3's PPE, and in particular the PPE's AltiVec SIMD instruction set, in the calculation. This is under investigation.

We refer to the Appendix for some of the details of our implementation and in particular for an interesting 4-way SIMD modular inversion method that is ideally suited to the SPU. An interesting aspect of our implementation is that we allow occasional errors by omitting the tests and branching that would have caught and fixed them. This implies that of the many parallel streams that we are processing, occasionally one may produce an erroneous result. Obviously, this would be recognized and the result thrown out – except that despite months of computations on many machines it has not occurred yet. Overall, we gain a very considerable speed-up using this approach.

Graphics card. As far as we can tell at this point, graphics cards do not dramatically change the above picture yet. In [4] it is reported that a GTX 295, containing two graphics processing units, can perform about 42 million modular multiplications per second for 280-bit moduli. Extrapolating the generic SPU modular multiplication times from the Appendix to this same modulus size (cf. last paragraph of Section A.1), we find that a single SPU can do almost 8 million modular multiplications per second for 280-bit moduli, for a total of 47 million per second per PS3. These figures are comparable, and we are not aware of a reason why that would not be the case for 160-bit moduli. The price and power consumption of graphic cards, however, both look less attractive than those of a PS3.

FPGA. For FPGAs we consult [15, Table V]: using a single XC3S1000 chip it would take about $3.1 \cdot 10^{11}$ years to crack 160-bit ECC. Furthermore, according to [15, Section 6.1], this time can be reduced by a factor 120 to $2.6 \cdot 10^9$ years when 120 of such chips are purchased for US$10,000. Note that $2.6 \cdot 10^9$ is close to the number of PS3 years required, as analysed above. Also note that about 25 PS3s can be purchased for US$10,000. It follows that FPGAs are not competitive. This hardly changes if we allow for a 'Moore factor' of about 2 in favor of FPGAs (since [15] was published in 2007).

ASIC. All performance figures presented so far are based on actual implementations or realistic extrapolations thereof. For ASICs we have not been able to find performance figures that are backed up by actual implementations. Nevertheless, from [15, Table VII] we conclude that ASIC performance is estimated to be about 200 times more economical than FPGAs for 160-bit ECC. Combined with the above factor 25 in favor of PS3s over FPGAs, we find that price-wise ASICs should outperform PS3s by about a factor 8. Thus, for about an eighth of the cost of 2.5 billion PS3s, one should be able to crack 160-bit ECC in about a year. As we did in the case of RSA, we argue (also based on [17]) that the time is not yet ripe for special purpose hardware, and do not consider ASICs in our further analysis below. It should be noted, though, that large scale special purpose hardware attacks against ECC look easier to realize than the designs proposed in [35].

From the above we conclude that, computationally speaking, cracking 160-bit ECC is at least three orders of magnitude harder than cracking 1024-bit RSA. For 1024-bit RSA we argued that the open community may have a chance by the year 2020, but that before 2015 nothing can be expected. These estimates, however, took into account continuation of the usual steady stream of algorithmic improvements, contributing a factor 10 over a decade. For ECC the contribution of algorithmic improvements has been much smaller. Even if they occur at the same rate as for RSA, we are still looking at the same difficulty gap of a factor one thousand by the year 2020, i.e., the year that cracking 1024-bit RSA may be feasible for the open community.

On the other hand, running a whole lot of Pollard rho processes is quite a bit less cumbersome than running the NFS sieving and matrix steps. Indeed, Chris Monico for his 109-bit ECC effort back in 2002 managed to attract a number of contributing computers that is about an order of magnitude larger than the current large scale factoring effort for a 768-bit RSA modulus. Given the large number of PS3s 'out there,' can the required number of compute cycles not conveniently be harvested using some type of PS3-BOINC project (cf. [5])? Right now most certainly not: 2.5 billion PS3s or equivalent devices (such as desktops) for a year is way out of reach. In a decade, very optimistically incorporating 10-fold cryptanalytic advances, still millions of devices would be required, and a successful open community attack on 160-bit ECC even by the year 2020 must be considered very unlikely.

4 Conclusion

Moving from 80-bit security to one of the proposed higher security levels is a good idea. There is, however, no reason to be concerned that usage of 1024-bit RSA until the year 2014 carries a major risk. More specifically, there is no indication that certificates relying on 1024-bit RSA will have to be revoked so long as they naturally expire by 2014. Similarly, there does not seem to be any reason to be concerned about continued usage of 160-bit prime field ECC during the next decade.

Recommendations of this sort can be made by anyone, in particular if they are, as the one concerning 1024-bit RSA, not particularly surprising. The value of our work is, however, that we are deeply involved in all the new theoretical and practical algorithmic research and grunt work required to be confident to make them. Although we are simply driven by algorithmic curiosity and interest in the underlying mathematical and computational problems, we are also willing to stick out our necks to interpret and publish the likely practical implications of

our activities, not in our own immediate interest but of potential use to the RSA and ECC user communities. We remark that our 1024-bit RSA estimate supports the recommendation proposed in [30]. We expect that our 160-bit prime field ECC estimate is more controversial – and may even trigger an out of the ordinary effort to prove it wrong.

Remark on security levels. Defining "56-bit security" as the level of protection offered by single DES (cf. [25] and [21]), we conclude from [32] that 56-bit security can on average be broken on a single PS3 (using 6 SPUs) in about 1.75 year. Combining that figure with the above PS3 effort to crack 160-bit ECC and observing that $2^{30} < 2.5 \cdot 10^9/1.75 < 2^{31}$, we find that 160-bit ECC provides between 86 and 87 bits of security. It would be reasonable to conclude that 1024-bit RSA provides about 76-bit security. The figures for ECC in [25, Table 1] are similar, but 1024-bit RSA looks a bit more secure than suggested there: apparently improvements in NFS have not lived up to the expectations from [25]. Note that this 10-bit gap allows one to use 160-bit ECC with a 10-bit cofactor (as in [28, Table 1] but as explicitly excluded here) without getting security less than 1024-bit RSA.

Acknowledgements

We gratefully acknowledge comments by Chris Babel and his team and by Paul Hoffman. This work was supported by the Swiss National Science Foundation under grant numbers 200021-119776 and 206021-117409 and by EPFL DIT. Travel related to the activities described in Section 2 was supported by the European Commission through the EU ICT program ECRYPT II.

References

1. M. Agrawal, N. Kayal, N. Saxena, Primes is in P, Annals of Mathematics 160, pp. 781–793 (2004).
2. K.Aoki, J. Franke, T. Kleinjung, A.K. Lenstra, D.A. Osvik, A kilobit special number field sieve factorization, Asiacrypt 2007, LNCS 4833, pp. 1–12, 2007.
3. F. Bahr, M. Boehm, J. Franke, T. Kleinjung, Factorization of RSA-200, May 2005, `http://www.loria.fr/~zimmerma/records/rsa200`.
4. D.J. Bernstein, T.-R. Chen, C.-M. Cheng, T. Lange, B.-Y. Yang, ECM on graphics cards, Eurocrypt 2009, LNCS 5479, pp. 483–501, 2009.
5. BOINC, Open-source software for volunteer computing and grid computing. `http://boinc.berkeley.edu/`.
6. D. Boneh, Twenty years of attacks on the RSA cryptosystem, Notices of the American Mathematical Society, 46: 203–213, 1999. `http://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf`.
7. J.W. Bos, M.E. Kaihara, T. Kleinjung, Pollard rho on elliptic curves, manuscript, submitted for publication, May 2009.
8. CADO workshop on integer factorization, presentations by T. Kleinjung, P.L. Montgomery, J. Papadopoulos, P. Stach, and others, Nancy, France, October 7–9, 2008: `http://www.sigsam.org/bulletin/issues/issue167.html`.
9. S. Cavallar, B. Dodson, A.K. Lenstra, P. Leyland, P.L. Montgomery, B. Murphy, H. te Riele, P. Zimmermann, et al., Factoring a 512-bit RSA modulus, Proceedings Eurocrypt 2000, Springer-Verlag LNCS 1807, 1–18, 2000.
10. Certicom. Certicom ecc challenge. `http://www.certicom.com/images/pdfs/cert_ecc_challenge.pdf`, 1997.
11. Certicom. Press release: Certicom announces elliptic curve cryptosystem (ECC) challenge winner. See `http://www.certicom.com/index.php/2002-press-releases/38-2002-press-releases/340-notre-dame-mathematician-solves-eccp-109-encryption-key-problem-issued-in-1997`, 2002.

12. Certicom Research. Standards for efficient cryptography 2: recommended elliptic curve domain parameters. Standard SEC2, Certicom, 2000.
13. J.H. Cheon, J. Hong, M. Kim, Speeding up the Pollard rho method on prime fields, Asiacrypt 2008, LNCS 5350, pp. 471–488 (2008).
14. R.P. Gallant, R.J. Lambert, and S.A. Vanstone, Improving the parallelized pollard lambda search on anomalous binary curves, Math. Comp. 69, pp. 1699–1705 (2000).
15. T. Güneysu, C. Paar, J. Pelzl, Special-Purpose Hardware for Solving the Elliptic Curve Discrete Logarithm Problem, ACM Trans. Reconfigurable Technol. Syst., pp. 1–21 (2008).
16. K. Hickey, The cutting edge of defense IT, Encrypting the future, Government Computer News, August 2007: `http://www.gcn.com/print/26_20/44796-5.html`.
17. P. Hofstee, personal communication, September 2008.
18. IBM, Example Library API Reference, Version 3.0, `http://www.ibm.com/developerworks/power/cell/documents.html`, 2007.
19. B.S. Kaliski, Jr., The Montgomery inverse and its applications, IEEE Trans. Computers, 44, pp. 1064–1065, 1995.
20. A.H. Koblitz, N. Koblitz, A. Menezes, Elliptic curve cryptography, the serpentine course of a paradigm shift, September 2008, `eprint.iacr.org/2008/390`.
21. A.K. Lenstra, Unbelievable security, Proceedings Asiacrypt 2001, Springer-Verlag LNCS 2248, 67–86.
22. A.K. Lenstra, H.W. Lenstra, Jr. (editors), The development of the number field sieve, Springer-Verlag LNM 1554, August 1993.
23. A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse, J. Pollard, The factorization of the ninth Fermat number, Math. Comp. 61: 319–349, 1994.
24. A.K. Lenstra, M.S. Manasse, Factoring by electronic mail, Proceedings Eurocrypt 1989, Springer-Verlag LNCS 434, 355–371, 1989.
25. A.K. Lenstra, E.R. Verheul, Selecting cryptographic key sizes, J. of Cryptology 14: 255–293, 2001.
26. P.L. Montgomery, Speeding the Pollard and elliptic curve methods of factorization, Math. Comp. (1987), pp. 519–521.
27. National Institute of Standards and Technology, Suite B Cryptography: `http://csrc.nist.gov/groups/SMA/ispab/documents/minutes/2006-03/E_Barker-March2006-ISPAB.pdf`.
28. National Institute of Standards and Technology, FIPS Pub 186-3, Digital Signature Standard (DSS), `http://csrc.nist.gov/publications/fips/fips186-3/fips_186_3.pdf`.
29. National Institute of Standards and Technology, Special Publication 800-57: Recommendation for Key Management Part 1: General (Revised), `http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf`.
30. National Institute of Standards and Technology, Discussion paper: the transitioning of cryptographic algorithms and key sizes, `http://csrc.nist.gov/groups/ST/key_mgmt/documents/Transitioning_CryptoAlgos_070209.pdf`.
31. National Security Agency, Fact sheet NSA Suite B Cryptography: `http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml`.
32. D.A. Osvik, E. Tromer, Cryptologic applications of the PlayStation 3: Cell SPEED: `http://www.hyperelliptic.org/SPEED/slides/Osvik_cell-speed.pdf`.
33. J.M. Pollard, Monte Carlo methods for index computation (mod p), Math. Comp (1978), pp. 918–924.
34. R. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public key cryptosystems, Commun. of the ACM, 21: 120–126, 1978.
35. A. Shamir, E. Tromer, Factoring large numbers with the TWIRL device, Proceedings Crypto 2003, Springer-Verlag LNCS 2729, 1–26, 2003.
36. P.W. Shor, Algorithms for quantum computing: discrete logarithms and factoring, Proceedings of the IEEE 35th ann. symp. on foundations of computer science, 124–134, 1994.
37. M. Stevens, A. Sotirov, J. Appelbaum, A.K. Lenstra, D. Molnar, D.A. Osvik, B. de Weger, Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate, `http://eprint.iacr.org/2009/111`, Crypto 2009, Springer-Verlag 2009, LNCS XXXX, YY–ZZ.
38. E. Teske, On random walks for Pollard's rho method, Math. Comp. (2001), pp. 809–825.
39. P.C. van Oorschot and M.J. Wiener, Parallel collision search with cryptanalytic applications, Journal of Cryptology 12 (1999), pp. 1–28.
40. M.J. Wiener and R.J. Zuccherato, Faster attacks on elliptic curve cryptosystems, SAC 1998, LNCS 1556, pp. 190–200 (1998).

A Multi-precision integer arithmetic on the SPU

We present some of the details of our SPU multi-precision integer arithmetic implementation on the PS3 SPU. This implementation is designed for the relatively short integers of at most a few hundred bits as used for ECC over prime fields. For these sizes we can, given the SPU architecture and because our Pollard rho application allows any level of parallelism, profit from 4-way SIMD parallelism by processing four identical or similar long integer operations simultaneously. For applications requiring larger integers (such as RSA) or where parallelism cannot be used, entirely different implementations are required. Those have been developed as well, but will not be described here.

A.1 Representation of relatively short long integers

Each SPU has 128 registers of 128 bits each. Each 128-bit SPU register v can be interpreted as a vector of eight 16-bit values $(v_{0,0}, v_{0,1}, v_{1,0}, \ldots, v_{3,1})$ or as a vector of four 32-bit values (v_0, v_1, v_2, v_3). When doing modular multiplication, n registers $X[0], X[1], \ldots, X[n-1]$ are interpreted as four $16n$-bit integers $X_{0,h}, X_{1,h}, X_{2,h}, X_{3,h}$ for $h \in \{0, 1\}$ with $X_{j,h} = \sum_{i=0}^{n-1} X[i]_{j,h} 2^{16i}$ with the understanding that if $h = 0$ then the $X[i]_{j,1}$ are zero and if $h = 1$ then the $X[i]_{j,0}$ are zero. While doing modular subtraction or inversion, m registers $X[0], X[1], \ldots, X[m-1]$ are interpreted as four $32m$-bit integers X_0, X_1, X_2, X_3 with $X_j = \sum_{i=0}^{m-1} X[i]_j 2^{32i}$. It is understood that any four-tuple $[A, B, C, D]$ of $16n$-bit or $32n$-bit integers can be represented in this way using n 128-bit registers consisting of four *short* integers (with $h = 0$ or $h = 1$) or m 128-bit registers consisting of four *long* integers, respectively. SIMD instructions are then applied to the four-tuple $[A, B, C, D]$ by applying the relevant SIMD instructions to the corresponding n or m registers.

The short representation is necessitated by the fact that the SPU has just a $16 \times 16 \to 32$-bit 4-way SIMD multiplier, the long one is to take advantage of fast 32-bit 4-way SIMD operations (such as addition and subtraction). Because switching back and forth between the two representations is relatively easy using shuffle operations (which rearrange bytes), and because furthermore the shuffle operations can be dispatched almost for free since they are odd pipeline instructions as opposed to the even pipeline arithmetic ones, this turned out to be the most efficient approach.

Thus, the 128-bit register width implies that four long integers can be processed simultaneously in SIMD mode, but this fixed width does not impact the efficiency when dealing, for different applications, with equal length moduli of another size: all one needs to do is use a different value for n: for instance, $n = 4$ for a 128-bit modulus, and $n = 5$ for a 160-bit one. When moving to larger moduli, various overheads will have a relatively smaller effect on the cycle count, so that the usual quadratic cost extrapolation for schoolbook multiplication and such will be on the pessimistic side.

A.2 Modular arithmetic

Let $R = 2^{128}$ and $\tilde{p} = R - 3$. Curve number 6 from [12] is defined over the prime field of characteristic p, where p is the 112-bit prime $\tilde{p}/(11 \cdot 6949)$. We show that it is computationally

advantageous to use a redundant representation modulo $\tilde{p}$ as opposed to p, while only reducing results to a unique representation in $\{0, 1, \ldots, p-1\}$ when so required (cf. Section A.4).

Let $\mathfrak{R}(x) = x \bmod R + 3\left\lfloor \frac{x}{R} \right\rfloor$ for non-negative x, then $x \equiv \mathfrak{R}(x) \bmod \tilde{p}$. Furthermore, with high likelihood $\mathfrak{R}$ can be used to quickly reduce values modulo $\tilde{p}$. Because $0 \leq \mathfrak{R}(x) < 4R$ for any x with $0 \leq x < R^2$, it follows that $0 \leq \mathfrak{R}(\mathfrak{R}(x)) < R + 9$. It is easily seen that $R + 9$ can be replaced by $R + 6$. Assuming that all values have more or less the same probability to occur, the result will actually most likely be $< \tilde{p}$. Although counterexamples are simple to construct and we have no formal proof, we can confidently state the following.

Proposition 1. *For independent random 128-bit non-negative integers x and y there is overwhelming probability that $0 \leq \mathfrak{R}(\mathfrak{R}(x \cdot y)) < \tilde{p}$.*

Thus, our modular multiplication works as follows. We have two four-tuples of 128-bit integers $[X_0, X_1, X_2, X_3]$ and $[Y_0, Y_1, Y_2, Y_3]$, where each tuple is represented using four 128-bit registers of 4 long (i.e., 32-bit) integers per register. Using shuffle operations the corresponding short (i.e., 16-bit) representations are extracted, per tuple requiring eight 128-bit registers (of 4 short integers per register). Next, the 4-way SIMD multiply-and-add instructions are used along with ordinary schoolbook multiplication to compute a four-tuple of 256-bit integers $[Z_0, Z_1, Z_2, Z_3]$, represented using eight 128-bit registers of 4 long integers per register, such that $Z_i = X_i \cdot Y_i$ for $i = 0, 1, 2, 3$. Multiply-and-add works nicely, because addition of a short value d to the product of shorts a and b while including a short carry c does not cause overflow: if $0 \leq a, b, c, d < 2^{16}$, then $d + (a \cdot b) + c < 2^{32}$. The product four-tuple $[Z_0, Z_1, Z_2, Z_3]$ is then reduced modulo $\tilde{p}$ by twice applying $\mathfrak{R}$ in 4-way SIMD fashion (cf. Prop. 1), where in most cases the second application of $\mathfrak{R}$ requires only a single multiply-and-add.

Our implementation takes care to fill both the even and the odd pipelines, thereby considerably reducing the overall latency. The average number of clock cycles required per 4-way SIMD multiplication modulo $\tilde{p}$ is about 215, i.e., 54 clock cycles per Pollard rho process.

Modular subtraction also uses the 4-way SIMD approach and can easily be made to work on the long representations, i.e, just four 128-bit registers to represent a four-tuple of 128-bit integers. Expensive branching is avoided by appropriate use of mask instructions. Overall, a 4-way SIMD subtraction modulo $\tilde{p}$ takes 16 instructions in the odd and 20 in the even pipeline, for a total of 20 clock cycles. This becomes 5 clock cycles per Pollard rho process.

A.3 Modular inversion

The calculation of the modular inverse of a positive integer x in the residue class of the odd modulus p is done by the algorithm depicted in Alg. 1. The method used is the one as described in [19], as it is very suitable for implementation on the 4-way SIMD architecture of the SPU. It proceeds in two phases:
1) The computation of the almost Montgomery inverse $x^{-1} \cdot 2^k \bmod p$ for some k.
2) A normalization phase where the factor $2^k \bmod p$ is removed.
Because 4 variables are employed on which operations can be carried out in SIMD-mode, they are grouped together as $[A_1, B_1, A_2, B_2]$, as set forth in Section A.1. In the algorithm, $X >> t$ ($X << t$) means that variable X is shifted by t bits towards the least (most) significant bit position (the SPU lets each of the elements of a four-tuple have its own shift amount). Note

Algorithm 1 4-SIMD Extended Binary GCD

Input: $r = 2^{32}$, $p : r^{n-1} < p < r^n$ and $\gcd(p, 2) = 1$
$x : 0 < x < r^n$ and $\gcd(x, p) = 1$
Output: $z \equiv \frac{1}{x} \bmod p$
Algorithm:

```
[A1, B1, A2, B2] := [p, 0, x, 1] and [k1, k2] := [0, 0]
while (true)
    /* Start of shift reduction. */
    Find t1 such that 2^t1 | A1
    Find t2 such that 2^t2 | A2
    [k1, k2] := [k1 + t1, k2 + t2]
    [A1, B1, A2, B2] := [A1 >> t1, B1 << t2, A2 >> t2, B2 << t1]

    /* Start of subtraction reduction. */
    if (A1 > A2) then
        [A1, B1, A2, B2] := [A1 − A2, B1 − B2, A2, B2]
    elseif (A2 > A1) then
        [A1, B1, A2, B2] := [A1, B1, A2 − A1, B2 − B1]
    else
        Return z := B2 · (2^−(k1+k2)) mod p
    endif
endwhile
```

that, in the algorithm, operations $A_1 >> t_1$ and $A_2 >> t_2$ shift out only zero bits. The arithmetic is unsigned.

Let $g = \gcd(x, p)$. Let y be a solution of $xy \equiv g \pmod{p}$. The algorithm has invariants

$$\begin{aligned} k_j \geq 0, \quad A_j > 0, \\ A_j(2^{k_1+k_2}y) \equiv B_j g \pmod{p} \quad (\text{for } j = 1, 2), \\ \gcd(A_1, A_2) = g, \\ A_1B_2 - A_2B_1 = p, \\ 2^{k_1}A_1 \leq p, \quad 2^{k_2}A_2 \leq x, \\ B_1 \leq 0 < B_2. \end{aligned} \tag{1}$$

When the loop exits, a modular multiplication by a table look-up removes powers of 2 from the output. We can bound the subscript $k_1 + k_2$ by

$$2^{k_1+k_2} \leq (2^{k_1}A_1)(2^{k_2}A_2) \leq px.$$

We will have $A_1 = A_2 = \gcd(A_1, A_2) = g$. If $A_2 > 1$ then we report an error to the caller. Otherwise $g = 1$. The output $z = B_2 \cdot (2^{-k_1-k_2})$ satisfies

$$z = zg \equiv B_2 \cdot (2^{-k_1-k_2})g \equiv (A_2 2^{k_1+k_2}y)2^{-k_1-k_2} \equiv A_2 y = y \pmod{p}.$$

A shift reduction always starts with at least one of A_1 and A_2 being odd, by (1). We do not know which of these might be even, but can examine both, in a SIMD fashion. If we pick t_1 and t_2 as large as possible during a shift reduction, then the new A_1 and A_2 will both be odd. In that case the next subtraction and shift reductions will reduce $A_1 + A_2$ by at least a factor

of 2. The trailing zero bit count of a positive integer A is the population count of $\overline{A} \wedge (A-1)$. The SPU's population count instruction acts only on 8-bit data, so our t_1 and t_2 may not be maximal.

The values of A_1 and A_2 are bounded by p and x, respectively. The invariant $p = A_1B_2 - A_2B_1 \geq B_2 - B_1$ bounds B_1 and B_2. For $n = 4$ these fit in 128 bits.

Within the subtraction reduction, the four 128-bit differences $A_1 - A_2$, $B_1 - B_2$, $A_2 - A_1$, and $B_2 - B_1$ are evaluated in parallel. We exit the loop if neither $A_1 - A_2$ nor $A_2 - A_1$ needs a borrow. Otherwise we update $[A_1, B_1, A_2, B_2]$ appropriately. Subtracting 1 from each element of the borrow vector gives masks of -1 or 0 depending on the sign of $A_1 - A_2$ or $A_2 - A_1$. A shuffle of these masks builds a selector which determines which parts of $[A_1, B_1, A_2, B_2]$ are updated.

The final multiplication with 2^{-k} is done by first looking up this value in a table and next computing the modular multiplication as outlined in Prop. 1. Hence, the modular inversion implementation takes as input an integer x and outputs $z \equiv \frac{1}{x} \bmod p$ with $0 \leq x, z < \widetilde{p}$.

Computation of a single modular inverse takes fewer than 5000 clock cycles on average. With $N = 400$ Pollard rho processes running in parallel, it boils down to about 12 clock cycles per Pollard rho process (plus the time required for the additional modular multiplications required per process in order to be able to use simultaneous inversion).

A.4 Incorporation into the Pollard rho iteration

Given elliptic curve group elements P and Q, we wish to calculate an integer m such that $Q = mP$. Since we use 16-adding walks (cf. [38]) we fix 16 group elements R_j, $0 \leq j < 16$, calculated as $R_j = \tilde{c}_j P + \tilde{d}_j Q$ for random integers $\tilde{c}_j, \tilde{d}_j$ and share those among all Pollard rho processes that will be participating in the calculation, i.e., across all streams on the same SPU, but also among all SPUs, and among all participating PS3s and any other devices. Furthermore, we fix and share an easy-to-calculate function h from the elliptic curve group more or less uniformly to $\{0, 1, 2, \ldots, 15\}$.

For each Pollard rho process integers c_0 and d_0 are randomly selected, and the startpoint $X_0 = c_0P + d_0Q$ is calculated. Using elliptic curve point addition (using doubling as well for the calculation of the R_j and all X_0's, but *never* using doubling during the iteration, cf. below), while sharing the inversion cost among $N = 400$ processes on the same SPU, the new point is then calculated as $X_{i+1} = X_i + R_{h(X_i)} = c_{i+1}P + d_{i+1}Q$ for $i = 0, 1, 2, \ldots$ until a distinguished point is found (cf. [39]). At the cost of 16 counters of 32 bits each per process, updating the values c_{i+1}, d_{i+1} can be postponed until a distinguished point is found. Both the calculation of h and the check for the distinguished point property require unique representation of the group element, whereas we use a redundant representation modulo $\tilde{p}$. This is relatively easy to fix by calculating the unique partial Montgomery reduction $x \cdot 2^{-16} \bmod p$ of the x-coordinate of the group element in question, since it will be unique in $\{0, 1, \ldots, p-1\}$. Identical distinguished points with, most likely, different constants c and d may lead to the desired solution to $Q = mP$.

As noted above, several things can go wrong: we may have dropped off the curve because we should have used curve doubling (in the unlikely case that $X_i = R_{h(X_i)}$, or in the unlikely case of incorrect reduction modulo $\tilde{p}$, cf. Prop. 1), or a wrong point may by accident again

Operation	Average #cycles per operation	Quantity per iteration	Average #cycles per iteration
Modular multiplication	54	6	322
Modular subtraction	5	6	30
Modular inversion	4941	$\frac{1}{400}$	12
Montgomery reduction	24	1	24
Miscellaneous	68	1	68
Total			456

Table 1. *Average clock cycle count for the operations during a Pollard rho iteration on one SPU.*

have landed on the curve, and have nonsensical c_i, d_i values. Just as the correct iterations, these wrong points will after a while end up as distinguished points. Thus, whenever a point is distinguished, we check that it indeed lies on the curve and that the equation $x_i = c_i P + d_i Q$ holds for the alleged c_i and d_i. Only correct distinguished points are collected. If we hit upon a process that has gone off-track, all 400 concurrent processes on that SPU are terminated and restarted, each with a fresh startpoint. This type of error-acceptance leads to enormous time-savings and code-size reduction at negligible cost: we have not found even a single incorrect distinguished point yet.

Enforcing uniqueness using the partial Montgomery reduction takes on average 24 clock cycles per iteration per process. The other issues mentioned above (retrieving the correct R_j's, shuffling them into the right places, distinguished point checking, branching overhead) contribute another 68 clock cycles, on average.

A.5 Timings

Table 1 combines the average clock cycle requirements per Pollard rho process. The total expected number of iterations to solve a discrete logarithm problem in the elliptic curve group over the 112-bit prime field is $\sqrt{\frac{\pi \cdot q}{2}} \approx 8.4 \cdot 10^{16}$, where q is the prime group order and where we do not use the negation map. Disregarding the possibility of tag-tracing, we need 456 clock cycles per iteration per SPU. Since an SPU runs at 3.2GHz and 6 SPUs are available per PS3, we expect about 60 PS3 years to complete the calculation. A moderate speed-up can be expected if one of the two improvements is used.

With a 24-bit distinguishing property we expect $5 \cdot 10^9$ distinguished points. Storing each distinguished point along with its c and d value requires $4 \cdot 112$ bits, i.e., 56 bytes, for a total of about $56 \cdot 5 \cdot 10^9$ bytes, i.e., 260 gigabytes.

Given the breakdown of the 54 cycle count for modular multiplication into (regular schoolbook) multiplication and (fast, due to Prop. 1) reduction, we expect that for generic 112-bit moduli only the 322 in Table 1 needs to be changed, namely to approximately 420. As a result the overall cycle count would grow by about 20%.

More precise extrapolation of the iteration cycle count for a generic 112-bit modulus to a generic 160-bit one than the rough "$\left(\frac{160}{112}\right)^2 \approx 2$" used in the PS3-part of Section 3, results in $N = 320$ and an overall iteration cycle count less than 900. The 160-bit ECC PS3-estimate from Section 3 is therefore on the high side (cf. also last paragraph of Section A.1).