

Petition for *Inter Partes* Review of
U.S. Patent No. 7,372,961

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

FACEBOOK, INC., INSTAGRAM, LLC, and WHATSAPP INC.,
Petitioners

v.

BLACKBERRY LIMITED,
Patent Owner

U.S. Patent No. 7,372,961
Issue Date: May 13, 2008

Title: METHOD OF PUBLIC KEY GENERATION

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 7,372,961**

Table of Contents

	Page
I. Mandatory Notices under §42.8(A)(1).....	1
A. Real Party-In-Interest under §42.8.(b)(1).....	1
B. Related Matters under §42.8(b)(2).....	1
C. Lead and Back-Up Counsel under §42.8(b)(3).....	2
D. Service Information.....	3
II. Fee Payment.....	3
III. Requirements under §§ 42.104 and 42.108 and Considerations under §325(d).....	3
A. Grounds for Standing	3
B. Identification of Challenge and Statement of Precise Relief Requested	3
C. Considerations under §325(d)	4
IV. Overview of the '961 Patent.....	6
A. Level of Ordinary Skill in the Art.....	6
B. Technology Overview	6
1. Cryptography and Cryptographic Keys	6
2. Random Number Generation	7
3. Generating Random Numbers Within a Desired Range By Using Modular Reduction	7
4. Alternative to Modular Reduction – Rejection Sampling	10
C. '961 Specification.....	12
V. Claim Construction.....	16
VI. The Challenged Claims are Unpatentable	18
A. Overview of Grounds	18
B. Ground 1: Obviousness of Claims 1, 2, 5, 15, 16 and 19 Over DSS in view of Schneier and Rose	19
1. Summary and Date Qualification of Prior Art.....	19

Table of Contents (continued)

	Page
(a) DSS (Ex. 1004).....	19
(b) Rose (Ex. 1006)	23
(c) Schneier (Ex. 1008)	27
2. Application of the Prior Art to the Challenged Claims	27
(a) Claim 1.....	27
(b) Claim 2: “The method of claim 1 wherein another seed value is generated by said random number generator if said output is rejected.”	45
(c) Claim 5: “The method of claim 1 wherein said order q is a prime number represented by a bit string of predetermined length L.”	46
(d) Claim 15.....	46
(e) Dependent Claims 16 and 19.....	47
C. Ground 2: Obviousness of Claims 1, 2, 5, 15, 16 and 19 Over DSS in view of Schneier and Menezes	47
1. Summary and Date Qualification of Prior Art.....	48
(a) Menezes (Ex. 1005).....	48
2. Application of the Prior Art to the Challenged Claims	49
(a) Claim 1.....	49
(b) Claim 2: “The method of claim 1 wherein another seed value is generated by said random number generator if said output is rejected.”	56
(c) Claim 5: “The method of claim 1 wherein said order q is a prime number represented by a bit string of predetermined length L.”	56
(d) Claim 15, 16, and 19.....	57
D. Grounds 3 and 4: Obviousness of Claims 23, 24, and 27 Over Prior Art from Grounds 1, in Further View of Rao and Floyd	57
1. Summary and Date Qualification of Prior Art.....	58

Table of Contents
(continued)

	Page
(a) Rao (Ex. 1018).....	58
(b) Floyd (Ex. 1019).....	58
2. Application of the Prior Art to the Challenged Claims	59
(a) Independent Claim 23.....	59
(b) Claims 24 and 27	62
VII. Conclusion	63

List of Exhibits

Exhibit No.	Description of Document
1001	U.S. Patent No. 7,372,961 B2 to Scott A. Vanstone et al. (filed December 26, 2001, issued May 13, 2008)
1002	Declaration of Jonathan Katz, Ph.D.
1003	Prosecution History for U.S. Patent Application No. 10/025,924, which issued as U.S. Patent No. 7,372,961
1004	Federal Information Processing (FIPS) Publication 186, <i>Digital Signature Standard (DSS)</i> (May 19, 1994)
1005	Excerpts from Alfred J. Menezes et al., <i>Handbook of Applied Cryptography</i> (1997)
1006	USENET article by Greg Rose to sci.crypt and sci.math, <i>Re: "Card-shuffling" algorithms</i> (March 10, 1993)
1007	Excerpts from Donald E. Knuth, <i>The Art of Computer Programming</i> , Vol. 2 (1998)
1008	Excerpts from Bruce Schneier, <i>Applied Cryptography</i> (2d ed. 1996)
1009	USENET article by Steve Brecher, <i>Re: How can I generate random numbers?</i> (October 12, 1996)
1010	USENET article by Steve Brecher, <i>Re: Help: Random numbers?</i> (November 13, 1996)
1011	USENET article by Trevor L. Jackson, III, <i>Re: Random Generator</i> (April 7, 2000)
1012	Declaration of Aviel Rubin, Ph.D Regarding Claim Construction of the '961 Patent, filed in Blackberry Limited v. Facebook, Inc. et al., No. 2:18-cv-01844-GW-KS (C.D. Cal.), filed February 28, 2019

List of Exhibits

Exhibit No.	Description of Document
1013	U.S. Patent No. 5,732,138 to Landon Curt Noll et al. (filed January 29, 1996, issued March 24, 1998)
1014	Excerpts from Jenny A. Frstrup, <i>USENET: Netnews for Everyone</i> (1994)
1015	Federal Information Processing (FIPS) Publication 186-2, <i>Digital Signature Standard (DSS)</i> (January 27, 2000)
1016	Excerpts from Brian W. Kernighan & Dennis M. Ritchie, <i>The C Programming Language</i> (2d ed. 1988)
1017	John von Neumann, <i>Various Techniques Used in Connection with Random Digits</i> , Summary by G.E. Forsythe, National Bureau of Standards Applied Math Series, 12 (1951), pp 36-38, reprinted in von Neumann's Collected Works, 5 (1963), Pergamon Press pp 768-770.
1018	Excerpts from Thammavarapu R. N. Rao, <i>Error Coding for Arithmetic Processors</i> (1974)
1019	Excerpts from Nancy A. Floyd, <i>Essentials of Data Processing</i> (1987)
1020	Amendments to Claims and Remarks, October 30, 2007, filed in U.S. Appl. Ser. No. 10/025,924 (issuing as the '961 patent)
1021	M. Horton et al., <i>Standard for Interchange of USENET Messages</i> , Request for Comments (RFC) 1036, December 1987
1022	K. Mieszkowski, <i>The Geeks Who Saved Usenet</i> , January 8, 2002, published at < https://www.salon.com/2002/01/07/saving_usenet/ >
1023	Google Inc., <i>20 Year Usenet Archive Now Available</i> , Internet Archive < https://web.archive.org/web/20011212191924/http://www.google.com/googlegroups/archive_announce_20.html >

List of Exhibits

Exhibit No.	Description of Document
1024	Expanded version of Exhibit 1006, USENET article by Greg Rose to sci.crypt and sci.math, <i>Re: "Card-shuffling" algorithms</i> (March 10, 1993), posted at: < https://groups.google.com/forum/#!searchin/sci.crypt/modulus\$20random\$20number\$20generator\$20bias%7Csort:date/sci.crypt/T0C1hfhbI_w/7DwT2RE6eFQJ >
1025	Excerpts from G. Todino et al., <i>Using UUCP and Usenet</i> (1991)
1026	B. Reid, <i>USENET Readership Summary Report for Mar 1993</i> (April 7, 1993), posted at: < https://groups.google.com/forum/message/raw?msg=news.lists/3vrSZPb0OLY/JgXOXAkYb7MJ >
1027	B. Reid, <i>USENET Readership Report for Mar 1993</i> (April 7, 1993), posted at: < https://groups.google.com/forum/#!original/news.lists/fQ3jYgDNc-A/wI1h_tc-P0oJ >
1028	Excerpts from <i>Merriam Webster's Collegiate Dictionary</i> (10th ed. 1998)
1029	Excerpts from <i>The American Heritage Dictionary of the English Language</i> (4th ed. 2000)
1030	Declaration of Greg Rose
1031	Declaration of Keith Moore
1032	Declaration of Sylvia D. Hall-Ellis, Ph.D.
1033	First Amended Complaint for Patent Infringement from <i>BlackBerry Limited v. Facebook, Inc. et al.</i> , No. 2:18-cv-01844-GW (C.D. Cal.), ECF No. 15, filed on April 4, 2018

List of Exhibits

Exhibit No.	Description of Document
1034	Certificates of Service from <i>BlackBerry Limited v. Facebook, Inc. et al.</i> , No. 2:18-cv-01844-GW (C.D. Cal.), ECF Nos. 20-23, showing that service on Petitioners was effected on April 6, 2018
1035	Tentative Ruling on Claim Construction in <i>BlackBerry Limited v. Facebook, Inc. et al.</i> , No. 2:18-cv-01844-GW (C.D. Cal.), ECF No. 152, filed April 1, 2019

I. MANDATORY NOTICES UNDER §42.8(A)(1)

A. Real Party-In-Interest under §42.8(b)(1)

Facebook, Inc. and subsidiaries, Instagram, LLC and WhatsApp Inc., are the real parties-in-interest to this IPR petition. For ease of reference, this Petition will refer to Facebook, Instagram, and WhatsApp collectively as “Petitioner” (singular).

B. Related Matters under §42.8(b)(2)

The ’961 patent is the subject of pending litigation involving Petitioner: *BlackBerry Ltd. v. Facebook, Inc. et al.*, Case No. 2:18-cv-01844-GW-KS (C.D. Cal.). Petitioner was first served on April 6, 2018. (Ex. 1034.)¹ The First Amended Complaint in that action alleges that Facebook, Instagram, and WhatsApp infringe the ’961 patent. (Ex. 1033, ¶¶89-132.)

As of the filing of this Petition, the district court has not issued any *final* claim construction rulings. On April 1, 2019, the district court issued its written Tentative Ruling on Claim Construction with respect to a disputed term of the ’961 patent, which is also addressed in **Part V** below. (Ex. 1035 at pp.30-34.) The district court did not indicate when it would issue its final order on claim construction. The

¹ The initial Complaint in that action was filed on March 6, 2018, and a First Amended Complaint on April 4, 2018. Service on Petitioner first took place on April 6, 2018, after the filing of the First Amended Complaint. (Ex. 1034.)

Petition for *Inter Partes* Review of
U.S. Patent No. 7,372,961

Petitioner is accordingly filing the present Petition at this time, but will update these Mandatory Disclosures to provide a copy of the district court's final claim construction order when it issues.

No trial date has been set in the pending litigation. Petitioner is aware of no other IPR petitions with respect to the '961 patent.

C. Lead and Back-Up Counsel under §42.8(b)(3)

Petitioner provides the following designation of counsel.

LEAD COUNSEL	BACK-UP COUNSEL
Heidi L. Keefe (Reg. No. 40,673) hkeefe@cooley.com COOLEY LLP ATTN: Patent Group 1299 Pennsylvania Ave. NW, Suite 700 Washington, DC 20004 Tel: (650) 843-5001 Fax: (650) 849-7400	Andrew C. Mace (Reg. No. 63,342) amace@cooley.com COOLEY LLP ATTN: Patent Group 1299 Pennsylvania Ave. NW, Suite 700 Washington D.C. 20004 Tel: (650) 843-5287 Fax: (650) 849-7400
	Mark R. Weinstein (Admission <i>pro hac vice</i> to be requested) mweinstein@cooley.com COOLEY LLP ATTN: Patent Group 1299 Pennsylvania Avenue NW Suite 700 Washington D.C. 20004 Tel: (650) 843-5007 Fax: (650) 849-7400
	Yuan Liang (Admission <i>pro hac vice</i> to be requested) yliang@cooley.com

LEAD COUNSEL	BACK-UP COUNSEL
	COOLEY LLP ATTN: Patent Group 1299 Pennsylvania Avenue NW Suite 700 Washington D.C. 20004 Tel: (202) 728-7132 Fax: (202) 842-7899

D. Service Information

This Petition is being served by Federal Express to the attorney of record for the '961 patent, BlackBerry Limited - Direct Practice - (US Team), ATTN: PATENT TEAM, 2200 University Avenue, E. Waterloo ON N2K 0A7. Petitioner consents to electronic service at the addresses provided above for lead and back-up counsel.

II. FEE PAYMENT

Petitioner requests review of nine claims, with a \$30,500 payment.

III. REQUIREMENTS UNDER §§ 42.104 AND 42.108 AND CONSIDERATIONS UNDER §325(D)

A. Grounds for Standing

Petitioner certifies that the '961 patent is available for IPR and that Petitioner is not barred or otherwise estopped.

B. Identification of Challenge and Statement of Precise Relief Requested

Petitioner requests institution of IPR based on the following grounds:

Petition for *Inter Partes* Review of
U.S. Patent No. 7,372,961

Ground	Claims	Challenge under §103(a)
1	1, 2, 5, 15, 16, 19	DSS (Ex. 1004), in view of Schneier (Ex. 1008) and Rose (Ex. 1006)
2	1, 2, 5, 15, 16, 19	DSS, in view of Schneier and Menezes (Ex. 1005)
3	23, 24, 27	DSS, in view of Schneier and Rose, in further view of Rao (Ex. 1018) and Floyd (Ex. 1019)
4	23, 24, 27	DSS, in view of Schneier and Menezes, in further view of Rao and Floyd

Submitted with this Petition is a Declaration of Jonathan Katz, Ph.D. (Ex. 1002) (“Katz”), a qualified technical expert. (Katz, ¶¶1-7, Ex. A.)

C. Considerations under §325(d)

This Petition does not present a scenario in which “the same or substantially the same prior art or arguments previously were presented to the Office.” §325(d). With the exception of Schneier (Ex. 1008), none of the prior art references cited above was cited during prosecution of the ’961 patent.

Schneier is a lengthy textbook on cryptography, and only eight pages (483-490) were considered during prosecution of the ’961 patent. (Ex. 1001, References Cited (56).) Those pages contain a summary and description of the Digital Signature Algorithm (DSA), which is also admitted prior art in the ’961 patent. (Schneier, pp.483-490; ’961, 1:52-2:55; Katz, ¶¶61-63.) There is no evidence that any other portion of Schneier was considered during prosecution.

Petitioner does not rely on Schneier's description of the DSA or pages 483-490 to show limitations of the challenged claims. Petitioner instead relies on other portions of Schneier describing techniques for generating true random numbers. (Schneier, pp.421-428; Katz, ¶104.) The random number discussion cited by Petitioner appears approximately sixty pages earlier in Schneier, and there is no evidence that the Examiner considered that portion or any other portion outside pages 483-490.

Additionally, Petitioner cites Schneier for a narrow and different purpose – to account for any attempt by Patent Owner to distinguish the prior art by arguing for an exceedingly narrow construction of “random number generator” as excluding pseudorandom number generators. *See, e.g., Wirtgen Am., Inc. v. Caterpillar Paving Prods. Inc.*, IPR2018-01202, Paper 10 at 12-13 (P.T.A.B. Jan. 8, 2019) (declining to invoke §325(d) against petitioner for relying on a reference considered during prosecution where the reference was previously considered for different reasons). Petitioner thus cites Schneier to foreclose any speculation that the disclosures of the primary DSS reference (Ex. 1004) are somehow insufficient to disclose the claimed random number generator.

As discussed below, the purported novelty of the '961 patent centers around determining “whether said output $H(SV)$ is less than said order q prior to reducing mod q ,” and the steps that follow as recited in the independent claims. Petitioner

cites Rose (Ex. 1006) in **Grounds 1 and 3**, and Menezes (Ex. 1005) in **Grounds 2 and 4**, for those features. There is no evidence that Rose and Menezes (or any other reference cited in **Grounds 1-4**) were cited or considered during prosecution.

IV. OVERVIEW OF THE '961 PATENT

A. Level of Ordinary Skill in the Art

A person of ordinary skill in the art as of December 2000 (the '961 patent priority date) would have possessed at least a bachelor's degree in electrical engineering or computer science (or equivalent degree) and at least two years of experience with computer-based cryptographic techniques. (Katz, ¶¶11-13.)

B. Technology Overview

The '961 patent, entitled "Method of Public Key Generation," purports to disclose and claim a method for generating a key for use in a cryptographic function. The '961 patent focuses primarily on a single aspect of cryptographic functions – the generation of an appropriate cryptographic key.

1. Cryptography and Cryptographic Keys

Cryptography is often associated with *encryption* (which can be used to transform information into scrambled or otherwise unintelligible information) and *digital signatures* (which can be used to verify that a message was not altered in transit). (Katz, Ex. 1002, ¶¶34-37.) Cryptographic algorithms typically rely on "keys," which are often represented as very large numbers. (*Id.*, ¶38.)

But in order for a cryptographic technique to be secure, the cryptographic key needs to be generated such that it is all but impossible to guess. (*Id.*, ¶39.) Many cryptographic techniques thus generate keys using **random number generators**, which attempt to generate numbers that are unpredictable and follow no discernable pattern. (*Id.*) Random number generation was a highly developed field in computer science prior to the '961 patent. (*Id.*, ¶¶31, 40-41.)

2. Random Number Generation

A simple example useful for understanding random number generation is to imagine a die with six sides showing 0, 1, 2, 3, 4, and 5.² (*Id.*, ¶32.) Each roll of the die generates a random number in the range from 0 to 5. If the die is unweighted, then each side is equally likely, *i.e.*, each roll is equally likely to result in any of the numbers 0, 1, 2, 3, 4, or 5. When each value in a given set of possible values is equally likely to occur, the random number generator is said to have a “uniform distribution” in that set. (*Id.*)



3. Generating Random Numbers Within a Desired Range By Using Modular Reduction

Many applications need random numbers that fall within a much *smaller range* than the range of the random number generator. To use a simple example,

² This discussion will use a die with faces showing 0-5 instead of 1-6 because the smallest nonnegative value representable in a computer is zero. (Katz, ¶32, n.1.)


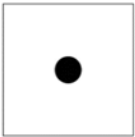
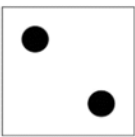
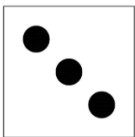
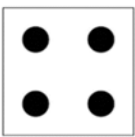

suppose someone wants to generate random numbers from 0-3 by throwing the hypothetical six-sided die mentioned above, having sides showing 0-5. Some of the throws of the die will show sides from 0-3, within the desired range. But others will result in values from 4-5, falling *outside* the desired range. (*Id.*, ¶¶43, 47.) This raises the obvious question – how does one handle these out-of-range values?

One well-known solution is to perform a mathematical operation known as a *modular reduction*, which computes the remainder when one number is divided by another. (*Id.*, ¶¶44-46.) The concept behind modular reduction would have been familiar to elementary schoolchildren: for example, when dividing 5 by 3, the quotient is 1 *with a remainder of 2*. Modular reduction returns this remainder. This operation is often expressed using the notation “**mod**,” so the example earlier would be expressed as “**5 mod 3 = 2**.” The divisor (3 in this example) is called the *modulus*. (*Id.*, ¶44.)

Modular reduction is used by random number generators to output numbers that fall within a desired range because the modular reduction will *always* yield values less than the modulus. For example, “**x mod 4**” can only yield four possible values – 0, 1, 2, or 3 – regardless of the value of “x.” (*Id.*, ¶¶ 45, 47.)

The table below illustrates an example of how modular reduction can generate random numbers within a desired range, using our earlier example of generating numbers from **0-3** using a six-sided die showing sides from **0-5**. We will thus rely

on the operation “**x mod 4**” to yield values from 0-3:

Roll of the Die	Modulo Operation (Die Value Mod 4)	Resulting Value
	$0 \bmod 4 = 0$	0
	$1 \bmod 4 = 1$	1
	$2 \bmod 4 = 2$	2
	$3 \bmod 4 = 3$	3
	$4 \bmod 4 = 0$	0
	$5 \bmod 4 = 1$	1

(Katz, ¶47.) As shown, the “**x mod 4**” operation transforms any roll of the die outside the desired range – namely 4 and 5 – into 0 and 1, respectively. (*Id.*, ¶¶45-47.) This illustrates how modular reduction can be used to output random numbers within a desired range, even if the random number generator has a larger range.

But using modular reduction here created a potential problem – this random

number generator does **not** provide a *uniform distribution* of numbers from 0-3. This is because *two* positions on the die (0 and 4) yield 0, *two* positions (1 and 5) yield 1, but only *one* position yields 2 and **only one** position yields 3. This random number generator thus exhibits **modulo bias** because it is *twice* as likely to output a 0 or a 1 as it is to output a 2 or a 3. (*Id.*, ¶48.)

4. Alternative to Modular Reduction – Rejection Sampling

Modular reduction is not the only way to output a random number that falls within a desired range. Another well-known technique is to simply throw out, or **reject**, any number output by the random number generator that lies outside the desired range. (*Id.*, ¶50.) To use the example above, if one rolled the die and it showed a “4” or “5”, she could simply reject the value and roll again (and continue rolling if needed) until the die showed a value between the desired range of 0-3.

This technique avoids modulo bias because it does not rely on modular reduction to obtain values within the desired range – it simply discards out-of-range values. (*Id.*) The technique of rejecting random values that fall outside a desired range, commonly known as “rejection sampling,” dates back to work during the early 1950s by famous mathematician John von Neumann. (*Id.*, ¶¶51-53.)

This technique was also described in Alfred J. Menezes et al., *Handbook of Applied Cryptography* (1997), whose authors include two of the ’961 patent’s named inventors. (Ex. 1005.) Menezes explains that a random bit generator can be used to

generate random numbers in the desired range of 0 to n by rejecting any values that exceed n . (Menezes, p.170, §5.2 (“[I]f the resulting integer exceeds n , one option is to discard it and generate a new random bit sequence.”).) In other words, one can generate random numbers within a desired range by accepting only values that fall within the desired range. This exceedingly simple technique is what was later claimed in the ’961 patent, as shown below. (Katz, ¶¶55, 66.)

At least three programmers on the Internet, apparently working independently, came up with and published this same technique. (*Id.*, ¶¶56-59.) First, on March 10, 1993, Greg Rose posted an article disclosing this technique to the `sci.crypt` and `sci.math` newsgroups on USENET. (Ex. 1006.) Rose’s article responded to an earlier article by another programmer who attempted to generate random numbers from 0 to 2 using a modular reduction technique. (Katz, ¶56 (citing Rose, Ex. 1006, at 001).) Rose explained that this approach creates a modulo bias problem, and Rose provided the solution:

Technically, there IS a problem with the above generator... When the desired interval doesn’t exactly divide the interval you already have, the smaller numbers are more likely than the larger ones. What you really have to do is throw away any result from the original generator that is too big.

(Rose at 002 (underlining added)³.) Rose goes on to disclose exemplary source code that generates a random number and throws it away if it is “too big,” *i.e.*, if it exceeds a desired maximum. (*Id.*; Katz, ¶¶56-57.) Petitioner will discuss Rose in more detail for **Ground 1** below.

Rose was not the only one to recognize the modulo bias problem and to identify this solution. In 1996, a programmer named Steve Brecher posted an article to USENET identifying the same problem and proposing substantially the same solution as Rose. (Katz, ¶58 (citing Brecher articles, Exs. 1009, 1010).) And on April 7, 2000, more than three years after Brecher, Trevor Jackson III posted an article to the Internet identifying the same problem and substantially the same solution as Rose and Brecher. (Katz, ¶59 (citing Jackson article, Ex. 1011).)

The articles posted by Rose, Brecher, and Jackson show three programmers on the Internet who independently identified and provided remarkably similar solutions to the modulo bias problem. This is hardly surprising considering that Rose, Brecher, Jackson – and the ’961 patent – all employ a simplified variation of the decades-old rejection method. (Katz, ¶60.)

C. ’961 Specification

The ’961 patent describes a method for avoiding potential bias in the

³ Unless noted otherwise, all emphasis and annotation has been added by Petitioner.

generation of a cryptographic key. The patent purports to remedy modulo bias in the generation of a key k in connection with the Digital Signature Algorithm (DSA), described in a well-known standard known as the **Digital Signature Standard (DSS)**. DSS was published by the National Institute of Standards and Technology (NIST), which is part of the U.S. Department of Commerce. (Katz, ¶61; DSS, Ex. 1004.)

By December 2000, NIST had already published versions of DSS in 1994, 1998, and 2000. (Katz, ¶62.) The Background of the '961 specification cites and discusses FIPS 186-2, the 2000 version of DSS. ('961, 2:36-46.) The '961 patent focuses on one aspect of DSS – the generation of the ephemeral cryptographic key, k . The patent explains:

One such implementation [of the DSA] is the DSS mandated by the National Institute of Standards and Technology (NIST) FIPS 186-2 Standard. The DSS stipulates the manner in which an integer is to be selected for use as a private key. A seed value, SV, is generated from a random number generator which is then hashed by a SHA-1 hash function to yield a bit string of predetermined length, typically 160 bits. The bit string represents an integer between 0 and $2^{160}-1$. However this integer could be greater than the prime q and so the DSS requires the reduction of the integer mod q , i.e. $k = \text{SHA-1}(\text{seed}) \bmod q$.

('961, 2:36-46.) To summarize, DSS first specifies generating a random number **seed**, and then computing a hash of the random number **SHA-1(seed)**. Because the

value of $\text{SHA-1}(\text{seed})$ may be larger than the desired range (as determined by the “prime q ”), a “ $\text{mod } q$ ” operation is performed to compute the value of the cryptographic key k , *i.e.*, $k = \text{SHA-1}(\text{seed}) \bmod q$. (’961, 2:44-46.) Computing “ $\text{SHA-1}(\text{seed}) \bmod q$,” as explained, ensures that the value of the key k will fall within the desired range required by the DSA, *i.e.*, from 0 to $q-1$.

But as detailed above, using a “ $\text{mod } q$ ” operation could introduce some modulo bias in favor of smaller values. The ’961 patent acknowledges that the possibility of such bias was known. (’961, 2:53-55 (“With this algorithm it is to be expected that more values will lie in the first interval than the second and therefore there is a potential bias in the selection of k .”); Katz, ¶¶64-65.)

The ’961 patent purports to avoid this bias by employing the same rejection technique well-known in the prior art – simply *reject* any random numbers that are not less than q , and keep generating new random numbers until one is obtained that is less than q . (’961, 4:11-17.) Figure 2 shows this technique:

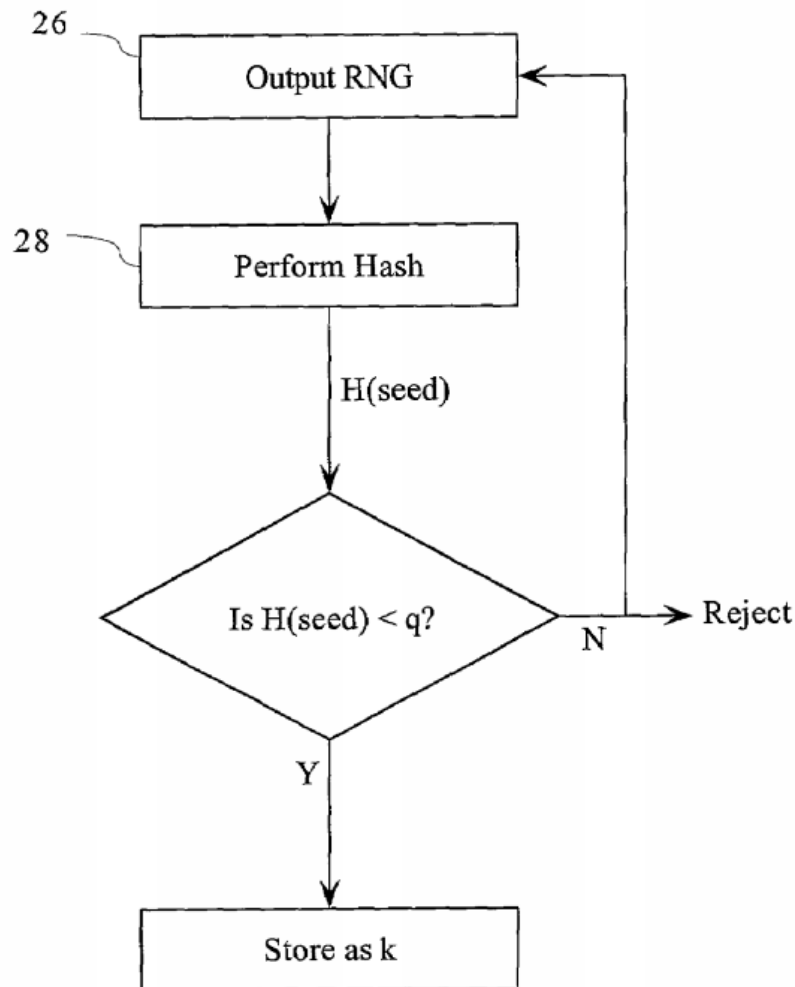


Fig. 2

(’961, Fig. 2.) The method shown “originates by obtaining a seed value (SV) from the random number generator **26**.” (’961, 3:64-66.) The number output from the random number generator (RNG) is then provided to hash function **28** in order to produce an output – **H(seed)** – of the correct length. (’961, 4:6-10.) This is the same value represented as “SHA-1(seed)” in the Background description of DSA. (’961, 2:40-46.) The hashing function is used “to yield a bit string of predetermined length, typically 160 bits.” (’961, 2:40-43, 3:38-41.)

The remaining steps in Figure 2 perform the simplified rejection technique of the prior art – namely, they check if **H(seed)** is less than **q**, where the maximum desired value is $q-1$. If the answer is “yes,” **H(seed)** is accepted and used as the cryptographic key **k**. Otherwise the value **H(seed)** “is rejected and the process starts over with generation of a new random number. This loop will continue indefinitely until a satisfactory value is obtained.” (’961, 4:13-17; Katz, ¶¶66-67.)

V. CLAIM CONSTRUCTION

The parties to the pending litigation dispute the meaning of “**reducing mod q**” as recited in each independent claim. The parties agree that the operation “**mod q**” determines the remainder of dividing a number by **q** but disagree on whether the operation must *always* result in a lower value.

Patent Owner has argued that the term “reducing mod q” requires an operation that *always* outputs a lower value, whereas Petitioner disagrees. As noted in **Part I**, on April 1, 2019, the district court issued its tentative claim construction ruling agreeing with Petitioner. (Ex. 1035, pp.30-34.)⁴ Petitioner has identified this dispute in the interest of completeness, but respectfully submits that the Board need not

⁴ The district court’s final ruling on claim construction was not available as of the filing of this Petition. Petitioner will provide the Board with a copy of that ruling when it becomes available.

resolve this dispute to apply the prior art to the challenged claims. The analysis in **Part VI** shows that the challenged claims are rendered obvious under either view.

But to the extent the Board deems a construction warranted, it should construe “**reducing mod q**” as determining the remainder of dividing a number by q. The term “reducing mod q” refers to the modulo operation itself – it does not require lowering of a numerical value based on a modulus q. (Katz, ¶¶70, 75-79.)

A modular reduction, expressed for example as “**a mod b**,” returns the remainder when **a** is divided by **b**. The operation *will* generate a lower number when $a \geq b$, but *will not* generate a lower number when $a < b$. A modular reduction therefore may or may not result in a lower value, depending on the inputs to the operation. (*Id.*, ¶¶71, 75.) Nonetheless, persons of ordinary skill refer to “a mod b” as performing a “modular reduction” or a “reduction modulo b,” regardless of whether or not the resulting value is less than the original “a” value. (*Id.*, ¶71.)

This is consistent with the specification, the claim language, and the definition of “modular reduction” provided by two learned treatises on cryptography cited as prior art herein (Schneier and Menezes) – neither of which define the operation as requiring a lower value. (*Id.*, ¶¶76-79.) The term “**reducing mod q**” therefore simply means calculating the remainder of dividing a first number by q.

VI. THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. Overview of Grounds

The '961 patent is not really about cryptography. It does not disclose any new cryptographic techniques and admits that “[t]he basic structure of a public key cryptosystem is well known and has become ubiquitous with security in data communication systems.” ('961, 1:10-12.) The '961 specification freely admits that the specific claim limitations actually relating to cryptographic techniques were admitted prior art and disclosed in the well-known Digital Signature Standard (DSS), as further demonstrated below. (Katz, ¶¶68, 108-111, 114, 125.)

The '961 patent instead addresses a narrower topic – generating a random number for use as a cryptographic key. The patent discloses an exceedingly simple technique for generating a random number whose essence is captured by the maxim, “if first you don’t succeed, try, try again.” The challenged claims recite a way to generate a random number within a desired range (“q”) by obtaining a random number and, after performing a hashing function on it, checking to see if it falls within the desired range. If it does, the number is accepted; if not, it is rejected and the process starts over, *i.e.*, try, try again. The concept recited in the claim can be readily analogized to the hypothetical presented in **Part IV.B.3** of attempting to obtain a random number from 0-3 by throwing a six-sided die showing sides from 0-5 – if the die shows a 4 or 5, keep throwing the die until it shows a 0 to 3.

Petitioner's grounds present two different ways of exposing the obviousness of this technique. **Ground 1** relies on Rose (Ex. 1006), an article posted to the `sci.crypt` and `sci.math` USENET newsgroups in 1993 in which Greg Rose described the same modulo bias problem identified in the '961 patent and provided exemplary source code to solve it. His solution: reject any random number greater than the desired range. (Rose, at 002 ("What you really have to do is throw away any result from the original generator that is too big.").) **Ground 2** relies on Menezes (Ex. 1005), a 1997 cryptography textbook that also describes a way to generate a random number falling within a particular range. Menezes explains that if the computer obtains a random number exceeding the desired range, "one option is to discard it and generate a new random bit sequence." (Menezes, p.170, §5.2.) Both of these techniques render the challenged claims obvious as explained below.

Grounds 3-4 mirror **Grounds 1-2**, respectively, but add additional references to address the "arithmetic processor" limitation recited in independent claim 23. These grounds otherwise provide the same prior art mappings as **Grounds 1-2**.

B. Ground 1: Obviousness of Claims 1, 2, 5, 15, 16 and 19 Over DSS in view of Schneier and Rose

1. Summary and Date Qualification of Prior Art

(a) DSS (Ex. 1004)

DSS, the Digital Signature Standard (DSS) describes the Digital Signature Algorithm (DSA) and was published in 1994 and adopted as a standard by the United

States Department of Commerce. (DSS, p.1; Katz, ¶¶83-84.) The DSA is one of the most well-known and influential cryptographic techniques ever created. (Katz, ¶84.) For clarity, “**DSA**” here refers to the Digital Signature Algorithm, and “**DSS**” to the U.S. Government publication describing it (Ex. 1004). DSS qualifies as a printed publication under §102(b). (Hall-Ellis Decl., Ex. 1032, ¶¶42-49.)

DSS explains that “[d]igital signatures are used to detect unauthorized modifications to data and to authenticate the identity of the signatory.” (DSS, Ex. 1004, Abstract.) **Exhibit 1004** contains the 1994 published version of DSS (FIPS 186), an earlier version of the DSS document (FIPS 186-2) specifically referenced in the ’961 patent.⁵ (’961, 2:36-44; Ex. 1015.)

Like the ’961 patent, this Petition focuses on a very small aspect of DSS – the process of generating a particular cryptographic key, **k** – the subject matter of the challenged claims. **Appendix 3** of DSS describes the process of generating **k** in detail, and **Appendix 5** provides a concrete example using real values. (DSS, pp.13-16, App. §3.2, 18-20.) As explained in Appendix 3.2:

⁵ Petitioner relies on the first version of DSS (FIPS 186/1994). The techniques for generating the ephemeral key **k** in DSS did not materially change from FIPS 186 (Ex. 1004) to FIPS 186-2 (Ex. 1015), and as such, the description provided in the ’961 patent mirrors the techniques in FIPS 186 as shown below. (Katz, ¶84 n.11.)

3.2. ALGORITHM FOR PRECOMPUTING ONE OR MORE k AND r VALUES

This algorithm can be used to precompute k , k^{-1} , and r for m messages at a time. Algorithm:

Step 1. Choose a secret initial value for the seed-key, KKEY.

Step 2. In hexadecimal notation let

$t = \text{EFCDAB89 98BADCFE 10325476 C3D2E1F0 67452301.}$

This is a cyclic shift of the initial value for $H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4$ in the SHS.

Step 3. For $j = 0$ to $m - 1$ do

a. $k = G(t, \text{KKEY}) \bmod q.$

b. Compute $k_j^{-1} = k^{-1} \bmod q.$

c. Compute $r_j = (g^k \bmod p) \bmod q.$

d. $\text{KKEY} = (1 + \text{KKEY} + k) \bmod 2^b.$

(*Id.*, p.14, App. §3.2.) Most of the steps shown in Appendix 3.2 (including the ones omitted above) describe features not relevant to the challenged claims, such as generating keys for multiple messages. The two yellow highlighted steps represent the steps actually reflected in the challenged claims. (Katz, ¶¶85-91.)

“**Step 1**” states that to compute k , one must “[c]hoose a secret initial value for the seed-key, KKEY.” (DSS, p.14, App. §3.2.) This step directly corresponds to step [a] of claim 1, “generating a seed value SV from a random number generator.” DSS explains that “[a]ny implementation of the DSA requires the ability to generate random or pseudorandom integers. Such numbers are used to derive a user’s private keys, x , and a user’s per message secret number, k .” (*Id.*, p.13; Katz, ¶86.)

“**Step 2**” requires little discussion. It assigns a value to t , a constant used when hashing KKEY in Step 3(a) described below. Step 2 is not directly reflected in the challenged claims, but under the DSA, it is a necessary prerequisite to the particular hashing function used, as described below. (Katz, ¶87.)

“**Step 3**,” subpart (a), states: “ $k = G(t, KKEY) \bmod q$ ” (DSS, p.14.) In plain English, this step performs a hashing function on the seed-key from Step 1 (KKEY), performs a “mod q ” operation on the output of the hashing function, and then assigns the resulting value to k . DSS explains that “ $G(t, KKEY)$ ” refers to a one-way hashing function using, for example, the Secure Hash Algorithm (SHA). (DSS, p.13; Katz ¶89.)⁶ The term “hashing function” refers to a computational algorithm that uses input data to generate an output “hash” of a fixed length (*e.g.*, 160 bits). (Katz, ¶89.)

The operation “ $G(t, KKEY)$ ” in DSS therefore generates a hash of the seed-key, KKEY, from Step 1. (*Id.*, ¶90.) This directly corresponds to step [b] of claim 1, “performing a hash function $H(\)$ on said seed value to provide an output $H(SV)$.”

⁶ DSS (Ex. 1004) uses the “SHA” one-way hash function while the later version of DSS referenced in the ’961 patent uses “SHA-1.” The differences between SHA and SHA-1 are immaterial to this Petition as the challenged claims do not require any particular hashing function. (Katz, ¶89 n.12.)

Step 3(a) also specifies that a “mod q ” operation is performed on the hash, *i.e.*, “ $k = G(t, KKEY) \bmod q$.” (DSS, p.14, App. §3.2.) As explained, the purpose of this “mod q ” operation is to ensure that the result falls within the desired range, *i.e.*, from 0 to q minus 1.

It is at this point that the ’961 patent steps in and purports to augment the DSS key generation process. Claim 1[c] recites the step of “determining whether said output $H(SV)$ [*i.e.*, $G(t, KKEY)$ in DSS] is less than said order q prior to reducing mod q .” The claim therefore requires that the claimed determination take place *before* any reduction modulo q takes place. DSS does not disclose the determining step, but the recited determination – and all the subsequent steps involved in accepting and rejecting an output $H(SV)$ – would have been obvious over DSS in view of Rose (for **Ground 1**), and Menezes (for **Ground 2**). (Katz, ¶¶92, 100, 102.)

(b) Rose (Ex. 1006)

Rose is an article authored by Greg Rose dated March 10, 1993, published to two newsgroups (`sci.crypt` and `sci.math`) on a global service known as USENET. Rose qualifies as a printed publication under §102(b) because it was sufficiently accessible to persons of ordinary skill. (Moore Decl. (Ex. 1031), ¶¶48-57; Rose Decl. (Ex. 1030), ¶¶4-5; Katz, ¶¶94-97.)

Rose addresses a previous USENET article that had suggested generating random numbers from 0-2 by performing a “mod 3” operation on the output of the

random number generator. (Rose, Ex. 1006, at 001; Katz, ¶¶98-99.) Rose explains that this method of generating random numbers causes modulo bias:

Technically, there IS a problem with the above generator. The example above demonstrates it. When the desired interval doesn't exactly divide the interval you already have, the smaller numbers are more likely than the larger ones. What you really have to do is throw away any result from the original generator that is too big.

(Rose at 002.) Rose then provides source code that, after generating a random number, determines whether the generated number is greater than or equal to a given bound. If so, the code rejects the random number and obtains another. (Katz, ¶99 (quoting Rose at 002).) Rose thus discloses a simplified rejection method for avoiding modulo bias in random number generation, and renders the challenged claims obvious in view of DSS as shown below. (Katz, ¶¶100, 150-160.)

Authenticity and Public Accessibility of the Rose

USENET refers to a global communications system for reading and disseminating messages (often called “articles” or “posts”) to users through the Internet. USENET was widely available throughout the world and regarded as a useful resource for researchers and professionals to share technical information with others in the field. (Moore Decl., Ex. 1031, ¶¶25-27; Rose Decl., Ex. 1030, ¶¶2-4; Katz, ¶¶94, 97.) As explained in the declaration of Mr. Rose, **Exhibit 1006** is an authentic copy of the USENET article he posted to the “sci.crypt” and

“sci.math” newsgroups on March 10, 1993. (Rose Decl., Ex. 1030, ¶¶1, 4-7.)⁷

Mr. Rose posted the article while working at the Australian Computing and Communications Institute. (*Id.*, ¶¶5-6.) He regularly followed “sci.crypt”, which he recalls at that time receiving approximately 100 new messages each day. (*Id.*, ¶¶2-4.) As he explains, “people generally followed the newsgroup to learn about what others are doing in the cryptography area, and from time to time, educate others and answer questions posted by other users.” (*Id.*, ¶4.)

The Federal Circuit has held that articles posted to USENET can qualify as prior art because each newsgroup is organized hierarchically, allowing interested persons to easily obtain a list of posts within the newsgroup. *See Suffolk Techs., LLC v. AOL Inc.*, 752 F.3d 1358, 1364-65 (Fed. Cir. 2014). Indeed, discussions on USENET are organized into a series of newsgroups identified by a hierarchical naming structure. (Moore, ¶26; Rose Decl., ¶2; Katz, ¶95.) For example, “sci.crypt” identifies the newsgroup to which the article was posted; “sci” refers to the top-level hierarchy of science and the “crypt” refers to a newsgroup relating to cryptography. (Katz, ¶95; Moore, ¶26.)

Each USENET newsgroup was essentially an electronic newsletter delivered

⁷ The Rose Declaration attaches its own copy of the Rose USENET article as **Exhibit A** (Ex. 1030 at 006-008), the same document as **Exhibit 1006**.

to interested readers. (Moore, ¶74.) Anyone could subscribe to any newsgroup without restrictions. (*Id.*) Today USENET articles from 1981-2000 are available through the Google Groups service, which collected those articles from a variety of different sources. (*Id.*, ¶¶39-41, 46.)

As explained by Keith Moore, who has decades of experience with USENET technology, the Rose article would have been widely accessible to tens of thousands of interested persons all over the world. (*Id.*, ¶¶12-13, 76-77.) In March 1993, the readership for “sci.crypt” and “sci.math” had reached 47,000 and 71,000 users, respectively. (Moore, ¶76.) Rose is a typical USENET article from the 1993 timeframe, and there is no evidence that any of its header fields or contents were materially altered. (*Id.*, ¶¶48-56.)⁸ Rose would have been publicly accessible through tens of thousands of USENET sites worldwide within a few days of March 10, 1993. (*Id.*, ¶¶57, 79.)

As noted, the Federal Circuit has held that articles posted to USENET can qualify as prior art. *See Suffolk Techs.*, 752 F.3d at 1364-65. In that case, the Federal Circuit observed that a paper delivered orally at a conference, during which at least

⁸ The only alterations relate to shortening email address strings in header fields and the signature block, a practice employed to thwart harvesting of email addresses. (Moore, ¶¶47, 55; Rose Decl., ¶6 n.1.)

six copies were distributed, qualified as a printed publication. *Id.* at 1365. The evidence here is far more compelling and overwhelming, and it establishes that Rose would have been publicly disseminated to tens of thousands of people interested in cryptography within a few days of its submission on March 10, 1993.

(c) Schneier (Ex. 1008)

Schneier is a book that provides “both a lively introduction to the field of cryptography and a comprehensive reference.” (Schneier, p.xx (Preface).) Schneier describes various techniques for generating random and pseudorandom numbers. (Schneier, pp.421-428.) Schneier qualifies as prior art under §102(b). (Hall-Ellis Decl., Ex. 1032, ¶¶50-56; Katz, ¶103.)

Petitioner cites Schneier in the event Patent Owner argues that the “**random number generator**,” as recited in claim 1[a], requires generation of true (as opposed to pseudo) random numbers. Schneier provides a description of true random numbers and explains their benefit over pseudorandom numbers in cryptographic key generation. (Schneier, pp.421-22, §17.14; Katz, ¶104.)

2. Application of the Prior Art to the Challenged Claims

(a) Claim 1

- (i) “A method of generating a key k for use in a cryptographic function performed over a group of order q, said method including the steps of:”
(1[Pre])**

To the extent limiting, the preamble is fully disclosed by DSS.

Petitioner respectfully submits that the Board need not even consult DSS because the '961 specification admits that the subject matter of the preamble was prior art. *See, e.g., PharmaStem Therapeutics, Inc. v. ViaCell, Inc.*, 491 F.3d 1342, 1362 (Fed. Cir. 2007) (“Admissions in the specification regarding the prior art are binding on the patentee for purposes of a later inquiry into obviousness.”).

The “Background of the Invention” of the '961 patent describes DSA and refers to it as a “popular” digital signature protocol. ('961, 1:52-56.) The specification explains that “[t]he DSA requires the signatory to select an ephemeral key k lying between 1 and $q-1$.” ('961, 1:63-64.) The specification goes on to describe how DSS describes generating key k :

One such implementation [of DSA] is the DSS mandated by the National Institute of Standards and Technology (NIST) FIPS 186-2 Standard. The DSS stipulates the manner in which an integer is to be selected for use as a private key. A seed value, SV, is generated from a random number generator which is then hashed by a SHA-1 hash function to yield a bit string of predetermined length, typically 160 bits. The bit string represents an integer between 0 and $2^{160}-1$. However this integer could be greater than the prime q and so the DSS requires the reduction of the integer mod q , i.e. $k = \text{SHA-1}(\text{seed}) \bmod q$.

('961, 2:36-46.) The passage confirms that the prior art DSS disclosed generation of the key k – the DSS formula “ $k = \text{SHA-1}(\text{seed}) \bmod q$ ” ('961, 2:46) as drafted computes the value for k . (Katz, ¶109.)

The Background also explains that k is an “ephemeral private key,” which can be used to generate the digital signature for a message. (’961, 1:26-47; Katz, ¶¶109-110.) The Background confirms that the key k is directly used to calculate the two DSA signature components (r and s) for a particular message: (Katz, ¶110 (citing ’961, 1:64-67).) The patent specification therefore admits that the prior art DSS disclosed **“generating a key k for use in a cryptographic function.”**

The Background also confirms that the prior art DSA generates a key k for use in a cryptographic function **“performed over a group of order q .”** The term **“group of order q ”** simply refers to a set of integer (whole) numbers between 0 and q minus 1, *i.e.* $\{ 0, 1, 2, \dots q-1 \}$. (Katz, ¶¶111-112.) The two signature components (r and s) generated by DSA are elements of the set of integers between 0 and $q-1$. (*Id.*, ¶112.) This is because the formulas for generating r and s (as set forth in the Background) both recite a “mod q ” operation. (’961, 1:64-67 (“A first signature component r is generated from the generator α such that $r=(\alpha^k \bmod p) \bmod q \dots$ A second signature component s is generated such that $s=k^{-1}(H(m)+dr) \bmod q \dots$ ”).) Because of the “mod q ” operation, the values r and s output by these functions will result in integer values between 0 and $q-1$, *i.e.*, **“over a group of order q .”** (Katz, ¶112.) The preamble of claim 1 is thus admitted prior art. (*Id.*, ¶¶109-112.)

Even putting aside the admissions in the ’961 specification, the preamble is disclosed by **DSS**, which discloses precisely the same technique described in the

Background of the '961 patent. (*Id.*, ¶113.) The step of “**generating a key k for use in a cryptographic function**” is shown in Appendix 3.2:

3.2. ALGORITHM FOR PRECOMPUTING ONE OR MORE k AND r VALUES

This algorithm can be used to precompute k , k^{-1} , and r for m messages at a time. Algorithm:

Step 1. Choose a secret initial value for the seed-key, KKEY.

Step 2. In hexadecimal notation let

$t = \text{EFCDAB89 98BADCFE 10325476 C3D2E1F0 67452301.}$

This is a cyclic shift of the initial value for $H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4$ in the SHS.

Step 3. For $j = 0$ to $m - 1$ do

a. $k = G(t, \text{KKEY}) \bmod q.$

b. Compute $k_j^{-1} = k^{-1} \bmod q.$

c. Compute $r_j = (g^k \bmod p) \bmod q.$

d. $\text{KKEY} = (1 + \text{KKEY} + k) \bmod 2^b.$

(DSS, Ex. 1004, p.14, App. §3.2.) The claimed generation of k is shown in Step 1, which assigns k the value of the hash of KKEY (*i.e.*, $G(t, \text{KKEY})$), $\bmod q$. Section 5 further discloses that k is used to generate the signature components r and s :

5. SIGNATURE GENERATION

The signature of a message M is the pair of numbers r and s computed according to the equations below:

$$r = (g^k \bmod p) \bmod q \quad \text{and}$$

$$s = (k^{-1}(\text{SHA}(M) + xr)) \bmod q.$$

(DSS, p.6, §5.) As shown in yellow, the generated key k is used in the formulas in Section 5 for computation of signature components r and s (“**for use in a cryptographic function**”). Those computations also include a “ $\bmod q$ ” operation

in blue to ensure that the signature values result in integers in the range of 0 to $q-1$, and thus the resulting values for r and s are integers lying in the range from 0 to $q-1$ (“performed over a group of order q ”). DSS therefore discloses the preamble for the same reasons as the admissions in the ’961 patent. (Katz, ¶113.)

(ii) **“generating a seed value SV from a random number generator;” (1[a])**

The ’961 specification admits that this step was also disclosed by DSS:

The DSS stipulates the manner in which an integer is to be selected for use as a private key. A seed value, SV, is generated from a random number generator which is then hashed by a SHA-1 hash function to yield a bit string of predetermined length, typically 160 bits.

(’961, 2:38-43.) The ’961 patent therefore admits that DSS disclosed **“generating a seed value SV from a random number generator.”**

This is confirmed by DSS itself. Appendix 3.2 identifies **Step 1** of the algorithm for computing the value of k as: “Choose a secret initial value for the seed-key, KKEY.” (DSS, p.14, App. §3.2.) The claimed **“seed value SV”** thus takes the form of the seed-key, “KKEY,” in DSS.

It would have been obvious that the seed-key “KKEY” would be generated by a random number generator. (Katz, ¶¶116-120.) Indeed, the first two sentences of Appendix 3 state that “[a]ny implementation of the DSA requires the ability to generate random or pseudorandom integers. Such numbers are used to derive a

user's private key, x , and a user's per message secret number, k ." (DSS, p.13, App. §3.) It would have been obvious that a random number "used to derive... a user's per message secret number, k " (*id.*) refers to the value of the seed-key, KKEY in **Step 1** of the algorithm for computing k . (Katz, ¶116.) In fact, Step 1 is the only step in the computation of k in which a random value can be input into the process – the remaining steps simply act upon the value of KKEY and thus do not independently generate random numbers. (*Id.*)

Random vs. Pseudorandom Number Generators: Random number generators generally fall into two categories – "true" and "pseudo" random number generators. (*Id.*, ¶118.) A "true" random number generator relies on an external source of unpredictability or randomness (such as random noise) to generate true random output. (*Id.*, ¶¶118, 41.) A "pseudorandom" number generator, on the other hand, uses a deterministic function applied to some initial input value, resulting in output that appears random. (*Id.*, ¶118.) The '961 specification does not appear to suggest that the claimed "**random number generator**" covers only *true* random number generators. (*Id.*, ¶119.)

And even if Patent Owner argues that the '961 patent excluded pseudorandom number generators, DSS itself discloses the generation of "**random** or pseudorandom integers" (DSS, p.13, App. §3), thus disclosing claim 1[a] even if the claim is construed narrowly. A person of ordinary skill would have been motivated

to use random (instead of pseudorandom) numbers because they can generate stronger cryptographic keys as explained below. (Katz, ¶¶120-121.)

This is confirmed by **Schneier** (Ex. 1008), which explains that true random numbers are preferred for key generation over pseudorandom numbers:

Sometimes cryptographically secure pseudo-random numbers are not good enough. Many times in cryptography, you want real random numbers. Key generation is a prime example... A random-sequence generator's sequences cannot be reproduced. No one, not even you, can reproduce the bit sequence out of those generators.

(Schneier, pp.421-22, §17.14.) Schneier goes on to describe several known techniques for generating real random numbers, including hardware techniques for harvesting random noise, using the computer's system clock and keyboard latency, among others. (*Id.*, pp.423-425, 426-428; Katz, ¶121.)

Rationale and Motivation to Combine DSS and Schneier: It would have been obvious to combine the teachings of DSS with Schneier, predictably resulting in the technique for generating key *k* in DSS in which the seed-key, KKEY, was always generated using a real (and not pseudo) random number generator. (Katz, ¶¶122-124.) DSS and Schneier are analogous references in the field of cryptography and key generation. A person of ordinary skill would have found it natural to consult Schneier in implementing DSS. (*Id.*, ¶122.)

Schneier provides an express motivation to combine: "Sometimes

cryptographically secure pseudo-random numbers are not good enough. Many times in cryptography, you want real random numbers. Key generation is a prime example.” (Schneier, p.421, §17.14.) A person of ordinary skill would have been motivated to use a real random number for the seed-key, KKEY, in DSS to create a stronger key for a digital signature. (Katz, ¶123.) A person of ordinary skill would have had every expectation of success, as techniques for generating “real” random numbers were known and readily available. (*Id.*, ¶¶124, 41.)

(iii) “performing a hash function $H()$ on said seed value SV to provide an output $H(SV)$,” (1[b])

The '961 specification admits that this step was also disclosed by DSS:

The DSS stipulates the manner in which an integer is to be selected for use as a private key. A seed value, SV, is generated from a random number generator which is then hashed by a SHA-1 hash function to yield a bit string of predetermined length, typically 160 bits. The bit string represents an integer between 0 and $2^{160}-1$. However this integer could be greater than the prime q and so the DSS requires the reduction of the integer mod q , i.e. $k = \text{SHA-1}(\text{seed}) \bmod q$.

(’961, 2:38-46.) The '961 patent therefore admits that DSS disclosed “**performing a hash function $H()$ on said seed value SV to provide an output $H(SV)$.**” The claimed “output $H(SV)$ ” corresponds to “SHA-1(seed)” (an integer between 0 and $2^{160}-1$) in the above-quoted passage. (Katz, ¶125.)

And the '961 specification is correct on this point – DSS does disclose this

step. Appendix 3.2 identifies **Step 3(a)** of the algorithm for computing the value of **k** as: “ $k = G(t, KKEY) \bmod q$.” (DSS, p.14, App. §3.2.) The claimed “**H(SV)**” thus takes the form of the hash of KKEY, **G(t, KKEY)**, in DSS. The DSS explains that “G(t,c)” refers to a one-way hashing function using, for example, the Secure Hash Algorithm (SHA). (DSS, p.13 (“The algorithms employ a one-way function G(t,c).... One way to construct G is via the Secure Hash Algorithm (SHA)....”).)

As explained in **Part VI.B.1(a)**, a “hashing function” uses input data to generate an output “hash” of a fixed length (*e.g.*, 160 bits). (Katz, ¶89.) The “t” in the “G(t,c)” hashing function refers to a fixed 160-bit constant in **Step 2** of DSS. (DSS, p.13.) The “c” refers to the input on which the hash operates. “G(t, KKEY)” therefore generates a hash of the seed-key, KKEY, from Step 1. (Katz, ¶¶127-128, 89-90.) Accordingly, DSS discloses “**performing a hash function H() on said seed value SV [i.e. KKEY] to provide an output H(SV).**”

- (iv) “**determining whether said output H(SV) is less than said order q prior to reducing mod q, accepting said output H(SV) for use as said key k if the value of said output H(SV) is less than said order q, rejecting said output H(SV) as said key if said value is not less than said order q, if said output H(SV) is rejected, repeating said method**” (1[c]-1[f])

DSS does not disclose these claim limitations. As mentioned, **Step 3(a)** of the algorithm for generating *k* in DSS states: “ $k = G(t, KKEY) \bmod q$.” (DSS, p.14, App. §3.2 (emphases added showing modular reduction of G(t, KKEY) to obtain the

value k .) The '961 patent similarly states that “the DSS require[d] the reduction of the integer mod q , i.e. $k = \text{SHA-1}(\text{seed}) \bmod q$.” ('961, 2:44-46.) DSS performs the “**mod q** ” operation to ensure that the resulting key k falls within the range of 0 to $q-1$. (Katz, ¶¶129-130.)

As explained in **Part IV.B**, the '961 patent employs a different approach. Instead of using modular reduction (**mod q**) to ensure that “the output $H(SV)$ ” falls within the desired range (from 0 to $q-1$), claim 1[c] simply rejects values falling outside that range. In overview, these claim steps first determine whether output $H(SV)$ is less than the desired range q . If the answer is “yes” ($H(SV) < q$) the method accepts output $H(SV)$ as the key k (claim 1[d]), otherwise **rejects** the number and repeats the method (claim 1[e]-[f]).

These limitations are obvious in view of Rose. Because claims 1[c]-1[f] are closely intertwined, Petitioner will address them together.

Rose discloses a technique for generating a random number within a specified range. Rose is a USENET article that responded to a previous article that had suggested generating random numbers from 0-2 using the following line of code (in the C programming language): “ $r = \text{rand48}() \% 3$ ” (where the percent sign (%) operator represents a modulo operation in C). Rose explained this code introduces a modulo bias problem:

Technically, there IS a problem with the above generator. The

example above demonstrates it. When the desired interval doesn't exactly divide the interval you already have, the smaller numbers are more likely than the larger ones. What you really have to do is throw away any result from the original generator that is too big.

(Rose, at 002.) Rose provides source code that obtains a random number and determines whether it is less than an upper range value (called "biggest"). As explained below, "biggest" will have a value equal to "q" under the combination with DSS. (Katz, ¶¶140, 151, 193.) If the candidate random number is less than biggest, it is accepted; otherwise it is rejected and the process starts over. The code, and Dr. Katz's annotations explaining it, are set forth below:

```
/*
 * Roll a random number [0..n-1].
 * Avoid the slight bias to low numbers.
 */
int
roll(int n)
{
    long    rval;
    long    biggest;
#define BIG 0x7FFFFFFFL /* biggest value returned by random() */

    if (n == 0)
        return 0;
    biggest = (BIG / n) * n;
    do {
        rval = random();
    } while (rval >= biggest);
    return rval % n;
}
```

Comment to the code explaining that the code below will generate a random number in the range from 0 to n-1, and avoid the modulo "bias"

Define function roll() which takes input "n", where the desired range for random numbers generated by roll will be from 0 to n-1

BIG identifies the largest number the underlying random number generator ("random()") may output

biggest is assigned to the largest multiple of "n" that is less than or equal to BIG.

This is a loop that (1) generates a random number rval using random number generator random(); (2) compares rval against biggest. If rval is greater than or equal to biggest, rval is rejected and the loop continues and generates another rval

At this point, the program has broken free from the loop because rval is less than biggest. The last step is to reduce rval modulo n (i.e., to compute rval mod n), to guarantee that the returned number falls within the desired range from 0 to n-1. This introduces no modulo bias because "biggest" (which was used in the loop above) was selected as a whole number multiple of "n".

(Rose, at 002 (annotations from Katz, ¶132); Katz, ¶¶133-142.)

The most critical portions are the following three lines:

```
do {  
    rval = random();  
} while (rval >= biggest);
```

(Katz, ¶141.) In plain English, these three lines (1) obtain a random number by calling the “**random()**” function, and assign that number to a variable called “**rval**,” and (2) check if “**rval**” is greater than or equal to “**biggest**,” our desired range as described below. If $rval \geq biggest$, the code rejects the value and starts over, looping back to step (1) to get another random number. Otherwise the code exits the loop and the number may be used. (*Id.*)

The “**determining**” step under the combination of DSS and Rose is reflected at the bottom of the loop with “while (rval >= biggest)”, which causes the computer to check if $rval \geq biggest$, and if so, loop back and obtain another random number. Conversely, if $rval < biggest$, the loop terminates and the random number is accepted. (Katz, ¶¶141, 145.) This corresponds to the next step of claim 1 under the combination of DSS and Rose, “**accepting said output H(SV) for use as said key k if the value of said output H(SV) is less than said order q,**” as explained further below. This is because, if $rval < biggest$, the loop terminates and no new random numbers need be obtained. (*Id.*, ¶145.)

Immediately after these three lines comes the final line of the **roll** function, “**return rval % n**”, which performs a modular reduction of “**rval**” based on the desired range “**n**” to obtain a value from 0 to n-1. (*Id.*, ¶147.) Because modular reduction in Rose only occurs after the “while (rval >= biggest)” operation (which performs the claimed “**determining**” step in combination with DSS), Rose satisfies the requirement of “determining whether said output H(SV) is less than said order q prior to reducing mod q.” (*Id.*, ¶¶147-148.)⁹

This code therefore accomplishes the same functions as the loop in claims 1[c]-1[f] – determine if a candidate random number falls within the desired range

⁹ In the pending litigation, Patent Owner has taken the position that “prior to reducing mod q” does not actually *require* performance of a “mod q” operation. (Rubin Decl., Ex. 1012, ¶57.) This position is correct – the claim language instead specifies that ***if*** a “reducing mod q” operation occurs, it cannot take place ***before*** the determining step. (Katz, ¶181 n.25.) Accordingly, even if the Patent Owner argues that the last line of **roll()** in Rose does not perform the claimed “reducing mod q” operation under its narrow district court construction (*see Part V*), this makes no difference – all that matters for the claim is that “reducing mod q” never occur *before* the claimed determining step, a requirement that Rose indisputably satisfies.

prior to modular reduction. If so, accept it; otherwise reject the number and try again by generating a new random number.

The code in Rose also uses the variable of “**biggest**” to determine whether the generated random number falls within the desired range. As explained by Dr. Katz, the code assigns to “biggest” the largest multiple of **n** (the desired range) that is less than or equal to BIG (the largest value the underlying random number can generate). (*Id.*, ¶¶137-140.) This computation is essentially an optimization intended to reduce the likelihood of rejection when the maximum value of the underlying random number generator is two or more times larger than the maximum value of the desired range. (*Id.*, ¶¶138-139, 193.)

An extended discussion of “**biggest**” is unnecessary because when applied to DSS, the value of biggest will always be the same as **q**, our desired range. (*Id.*, ¶¶140, 151, 193.) This is because the DSS requires that the value of **q** (which defines the top of the desired range) be more than half of the maximum value the random number generator can output. (*Id.*, ¶151.)

Manner, Rationale and Motivation to Combine: Rose illustrates a simple and generic acceptance-rejection technique that could readily be applied to any situation requiring generation of random numbers within a specified range, including DSS. A person of ordinary skill would have found it obvious to combine the techniques of Rose with the cryptographic key generation technique described in

DSS. (Katz, ¶¶150-160.) This combination would have predictably resulted in the technique of generating k described in DSS, enhanced with the technique in Rose, resulting in the following sequence of steps:

- (1) Generate a random number for KKEY in accordance with DSS alone or in combination with Schneier (the claimed “**seed value SV**”), as explained for claim 1[a];
- (2) perform a hash of KKEY, $G(t, KKEY)$, according to DSS (the claimed “**output H(SV)**”), as explained for claim 1[b];
- (3) determine whether $G(t, KKEY)$ (“**output H(SV)**”) is less than **order q** (*i.e.* the value q),¹⁰ according to the teachings of Rose;

¹⁰ The term “**said order q** ” is satisfied simply by the value q as disclosed in DSS, which is a large prime number. (*See* ’961, 1:54-62 (“The DSA domain parameters are preselected. They consist of... a prime number q of a predetermined bit length....”), 2:44-46; *see also id.*, 3:66-4:2.) This is consistent with claim 5, which recites “wherein said order q is a prime number represented by a bit string of predetermined length L .” The reference to “**said order q** ” in the claim thus simply refers to the value of prime number q . (Katz, ¶143; DSS, p.13, App. §3 (“These randomly or pseudorandomly generated integers are selected to be between 0 and the 160-bit prime q (as specified in the standard).”).)

(4) accept $G(t, \text{KKEY})$ (“**output $H(SV)$** ”) if $G(t, \text{KKEY})$ **is** less than q , according to the teachings of Rose;

(5) reject $G(t, \text{KKEY})$ if it is **not** less than q , by repeating the method and going back to step (1), according to the teachings of Rose; and

(6) if $G(t, \text{KKEY})$ (“**output $H(SV)$** ”) was accepted in step (4), perform a “mod q ” operation on $G(t, \text{KKEY})$, pursuant to the teachings of Rose, and provide it as a key k for use in performing a cryptographic function, in accordance with DSS.

(Katz, ¶150; *see also id.*, ¶¶143-148.)¹¹ A person of ordinary skill would have regarded this as a trivially simple combination that could have been successfully implemented in only a few lines of code. (*Id.*, ¶¶151-154, 159-160.) DSS, Schneier, and Rose are analogous references in the field of random number generation and cryptography, and Rose contains disclosures reasonably pertinent to the problems to be solved in DSS – generating a random number for a cryptographic key that falls

¹¹ Petitioner’s proposed combination relies on Rose for its determine-accept-reject logic, but not for generation or hashing of a random number. As explained in the text, instead of using the “random()” function in Rose, the system under the combination of Rose, Schneier and DSS would use the random number generator from DSS and Schneier (as described in claim 1[a]), each time a random number was needed, and hash that value in accordance with DSS. (Katz, ¶152.)

within a desired range. (*Id.*, ¶154.)

A person of ordinary skill would have been motivated to adapt Rose to DSS in order to avoid the “modulo bias” problem described earlier, which is expressly described in Rose. Rose’s comments preceding the “roll” function describe its purpose as “[a]void[ing] the slight bias to low numbers.” (Rose, at 002.) As further explained by Rose: “When the desired interval doesn’t exactly divide the interval you already have, the smaller numbers are more likely than the larger ones.” (*Id.*) Rose continues by providing the solution described in the ’961 patent: “What you really have to do is throw away any result from the original generator that is too big.” (*Id.*) By identifying the problem and the solution, Rose provides express motivations to adapt his technique for any application that requires random numbers within a specified range, such as DSS. (Katz, ¶155.)

A person of ordinary skill would have been motivated to use Rose’s technique to generate a key k as described in DSS. (*Id.*, ¶157.) A person of ordinary skill would have understood that modulo bias could, to some extent, weaken the strength of a cryptographic key by making some numbers more likely to occur than others. (*Id.*) The ’961 patent admits it was known that the “mod q ” operation in the DSA/DSS could create a bias towards lower values in the range of 0 to $q-1$. (’961, 2:53-61.) Accordingly, as of the filing of the ’961 patent, the modulo bias problem for DSS was known and the motivation for avoiding it was clear. (Katz, ¶157.) A

skilled artisan would thus have been motivated to use the technique in Rose to maintain a uniform distribution of random numbers over the range specified by q , resulting in a cryptographically stronger key k , in turn increasing the strength of the resulting digital signature scheme. (*Id.*)

Indeed, as explained in **Part IV.B.4**, rejecting values that fall outside a desired range was a well-known way of obtaining a uniform distribution of random numbers within the range. (Katz, ¶¶156, 53, 55-59.) At least two other programmers on the Internet – Brecher and Williams – independently made the same observation as Rose, and proposed solving it using remarkably similar techniques. (*Id.*, ¶¶58-60; Exs. 1009-1011.) The technique described in Rose and recited in the '961 patent, in fact, is a highly simplified version of the rejection method well known since at least the early 1950s. (Katz, ¶¶156, 51-54.) Thus, Rose's technique would also have been obvious to try. (*Id.*, ¶158.)

- (v) **“if said output $H(SV)$ is accepted, providing said key k for use in performing said cryptographic function, wherein said key k is equal to said output $H(SV)$.” (1[g])**

This final step does not add anything not already addressed in limitations 1[c]-1[f] above. As explained, DSS and Schneier, in combination with Rose, disclose a key generation system that accepts the hash of KKEY (“**said output $H(SV)$** ”) if it is less than order q .

Appendix 3.2 of the DSS identifies **Step 3(a)** of the algorithm for computing

the value of **k** as: “**k**=G(t,KKEY) mod q.” (DSS, Ex. 1004, p.14, App. §3.2.) Under the combination of DSS and Rose, therefore, after the hash of KKEY (*i.e.*, G(t,KKEY)) is accepted, it is assigned as “**key k**” and used in performing a cryptographic function. (Katz, ¶162.) As explained in the preamble, “key k” is used in the formula for computing the two digital signature components **r** and **s**, and thus, “key k” is “**use[d] in performing said cryptographic function,**” as claimed. (DSS, *e.g.*, pp.6-7, §5, p.14, App. §3.2.)¹²

(b) Claim 2: “The method of claim 1 wherein another seed value is generated by said random number generator if said output is rejected.”

This claim adds nothing that was not already addressed for claim 1. (Katz, ¶164.) Under the combination of DSS and Rose, if “**output H(SV)**” is rejected (*i.e.*,

¹² The proposed combination of DSS and Rose performs a modulo reduction before returning the resulting value, in accordance with the last line of the function in Rose (“return rval % n”). (Katz, ¶¶147-148.) But under the combination of DSS and Rose, the modulo reduction would *not* impact the value of G(t,KKEY) because, as explained in the text, its value would only be accepted if $G(t,KKEY) < q$, in which case “G(t,KKEY) mod q” would simply equal G(t,KKEY). The value provided as “key k” in claim 1[g] is thus the same value “accepted” in claim 1[d] above – G(t,KKEY) is unaffected by reducing mod q. (Katz, ¶163.)

because $G(t, KKEY) \geq q$), another random number is generated for KKEY (“seed value”) by the underlying random number generator as explained in claim 1[a], and the process continues to claim 1[b], thus repeating the steps of the claim. (*Id.*)

(c) Claim 5: “The method of claim 1 wherein said order q is a prime number represented by a bit string of predetermined length L.”

This claim also adds nothing of significance. The ’961 patent admits that the DSA/DSS specifies “a prime number q of a predetermined bit length, by way of example 160 bits....” (’961, 1:57-62.) DSS similarly refers to “q” as “160-bit prime q (as specified in the standard).” (DSS, p.13, App. §3.) DSS thus discloses that “**said order q is a prime number represented by a bit string of predetermined length L,**” with “L” being a length of 160 bits. (Katz, ¶¶165-166.)

(d) Claim 15

Claim 15 is substantially similar to claim 1, with the exception that claim 15 recites a “computer readable medium.”

(i) “A computer readable medium comprising computer executable instructions for generating a key k for use in a cryptographic function performed over a group of order q, said instructions including instructions for:” (Claim 15[Pre])

Claim 15[Pre] is identical to 1[Pre] of claim 1 except that it recites “a computer readable medium comprising computer executable instructions” that performs the method steps of claim 1. The recitation of “[a] computer readable

medium comprising computer executable instructions” adds nothing of patentable significance. DSS explains that “[t]he DSA may be implemented in software, firmware, hardware, or any combination thereof.” (DSS, p.3.) It would also have been apparent and obvious to a person of ordinary skill that the proposed combination of DSS, Schneier and Rose, described for claim 1, could have been implemented in software containing computer executable instructions provided on a computer readable medium. (Katz, ¶169.)

The remaining limitations of claim 15 are identical to claim 1. (*Id.*, ¶170 (showing side-by-side comparison chart).) Claim 15 is therefore obvious.

(e) Dependent Claims 16 and 19

These claims are substantially similar to claims 2 and 5, respectively. The only differences are the addition of “computer readable medium” (based on their dependency from claim 15), but this adds nothing of significance as noted.

C. Ground 2: Obviousness of Claims 1, 2, 5, 15, 16 and 19 Over DSS in view of Schneier and Menezes

Ground 2 is similar to **Ground 1** except that, instead of Rose, **Ground 2** relies on the rejection sampling technique of Menezes with respect to limitations reciting “determining whether said output $H(SV)$ is less than said order q prior to reducing mod q ,” and the subsequent limitations relating to accepting or rejecting said output $H(SV)$. With respect to all other claim limitations, the mapping from **Ground 1** remains the same. With respect to dependent claims 2, 5, 16, and 19, the

analysis from **Ground 1** remains unchanged and applies here.

1. Summary and Date Qualification of Prior Art

Ground 2 relies on DSS and Schneier, already summarized for **Ground 1** above, and further adds Menezes.

(a) Menezes (Ex. 1005)

Menezes, entitled *Handbook of Applied Cryptography* (1997), is a textbook “intended as a reference for professional cryptographers, presenting the techniques and algorithms of greatest interest to the current practitioner, along with supporting motivation and background material.” (Menezes, p.xxiii (Preface).) Two of its authors (Alfred J. Menezes and Scott A. Vanstone) are inventors on the ’961 patent, but there is no evidence that the book was cited or considered by the Examiner during prosecution of the ’961 patent. Menezes qualifies as prior art under §102(b). (Hall-Ellis Decl., Ex. 1032, ¶¶57-63, Katz, ¶101.)

Like Rose, Menezes discloses the ability to generate uniformly distributed random numbers by rejecting any numbers that fall outside a desired range. Menezes explains that the security of many cryptographic systems depends upon the ability to generate random or unpredictable sequences. (Menezes, p.169, §5.1.) Menezes explains that “[a] *random bit generator* is a device or algorithm which outputs a sequence of statistically independent and unbiased binary digits.” (Menezes, p.170, §5.1 (*italics in original*)). Menezes describes how to employ this technique to

generate a random number between 0 and n :

A random bit generator can be used to generate (uniformly distributed) random numbers. For example, a random integer in the interval $[0, n]$ can be obtained by generating a random bit sequence of length $\lfloor \lg n \rfloor + 1$, and converting it to an integer; if the resulting integer exceeds n , one option is to discard it and generate a new random bit sequence.

(*Id.*, §5.2 (italics in original).) Menezes thus discloses the purported point of novelty of the '961 patent – the ability to obtain random numbers within a desired range by rejecting random numbers that fall outside that range.

2. Application of the Prior Art to the Challenged Claims

(a) Claim 1

- (i) “A method of generating a key k for use in a cryptographic function performed over a group of order q , said method including the steps of:” (Preamble)**
- (ii) “generating a seed value SV from a random number generator;” (1[a])**
- (iii) “performing a hash function $H()$ on said seed value SV to provide an output $H(SV)$;” (1[b])**

With respect to 1[Pre] and claims 1[a]-1[b], the mapping from **Ground 1** remains unchanged. For the reasons previously stated, the preamble (to the extent limiting) and claims 1[a]-[b] are obvious over the prior art admissions in the '961 patent, and over DSS in view of Schneier. (Katz, ¶175.)

- (iv) “determining whether said output $H(SV)$ is less than said order q prior to reducing mod q , accepting said output $H(SV)$ for use as said key k if the value of said output $H(SV)$ is less than said order q , rejecting said output $H(SV)$ as said key if said value is not less than said order q , if said output $H(SV)$ is rejected, repeating said method” (1[c]-1[f])

As explained for **Ground 1**, DSS does not disclose these limitations. But they are obvious over DSS in view of Menezes. Because claim limitations 1[c] through 1[f] are very closely intertwined, Petitioner will address them together.

Menezes describes how to employ a random bit generator to generate a random number in a desired range from 0 to n :

A random bit generator can be used to generate (uniformly distributed) random numbers. For example, a random integer in the interval $[0, n]$ can be obtained by generating a random bit sequence of length $\lfloor \lg n \rfloor + 1$, and converting it to an integer; if the resulting integer exceeds n , one option is to discard it and generate a new random bit sequence.

(Menezes, p.170, §5.2 (italics in original).) Menezes thus discloses generating uniformly distributed random numbers ranging from 0 to n by (a) generating a random number in a larger range, and (b) rejecting the number if it lies outside the desired range, *i.e.*, if it exceeds n . This technique does not use any modulo arithmetic at all, thus avoiding modulo bias. (Katz, ¶179.)

Menezes, in combination with DSS, discloses “**determining whether said output $H(SV)$ is less than said order q prior to reducing mod q .**” Under the combination with DSS, the “resulting integer” in the passage of Menezes quoted above corresponds to “**said output $H(SV)$,**” and the interval from 0 to n corresponds to the interval from 0 to $q-1$ specified by the “**order q .**” (*Id.*, ¶180.)

It would have been obvious that implementing the simplified rejection technique in Menezes would have required that the software *determine* whether the random number is less than or equal to n , or greater than n , and make a decision based on that determination. The above-quoted passage explains that “if the resulting integer exceeds n , one option is to discard it and generate a new random bit sequence,” which could not occur without a comparison of the resulting integer against the value of n . (*Id.*)

With respect to the requirement of determining whether said output $H(SV)$ is less than said order q “**prior to reducing mod q ,**” this is satisfied because the technique above avoids the need to perform any “mod q ” operation altogether. There is no need to perform such an operation because, under the technique in Menezes as combined with DSS, the value of “ $G(t, KKEY)$ ” (“**said output $H(SV)$** ”) will only be accepted if $G(t, KKEY) < q$, in which case $G(t, KKEY) \bmod q$ would simply return

the same value, $G(t, KKEY)$.¹³ (Katz, ¶181.)

The above-quoted text from Menezes, in combination with DSS, also discloses “**accepting said output $H(SV)$ for use as said key k if the value of said output $H(SV)$ is less than said order q .**” This is because Menezes states that “if the resulting integer exceeds n , one option is to discard it and generate a new random bit sequence.” (Menezes, p.170, §5.2.) It would have been obvious that a value not

¹³ As noted in footnote 9, *supra*, “prior to reducing mod q ” does not require performance of a “mod q ” operation – it instead specifies that a “reducing mod q ” operation, if it occurs, cannot take place *before* the determining step. The claim therefore does not exclude a system in which “reducing mod q ” is not performed at all. (Katz, ¶181 n.25.) And even if the claim were interpreted to require performance of a “mod q ” operation, under the DSS-Menezes combination, that operation could be performed *after* it was determined whether $G(t, KKEY) < q$. (*Id.*, ¶182.) It would have been obvious to adapt the DSS-Menezes combination such that any “mod q ” operation would not occur until after it was determined that an acceptable value was attained. (*Id.*) The performance of “mod q ” in the context of the DSS-Menezes combination, although guaranteed to not result in a reduction in value, would require fewer modifications to existing DSA implementations, and thus allow them more closely track the description in the DSS standard. (*Id.*, ¶78 n.10.)

exceeding n would not be “discarded,” but instead, accepted. (Katz, ¶¶182-183.)

And turning to the step of “**rejecting said output H(SV) as said key if said value is not less than said order q, if said output H(SV) is rejected, repeating said method,**” this occurs under the combination of DSS and Menezes because if the resulting random integer is not less than or equal to n , “one option is to discard it and generate a new random bit sequence.” (*Id.*, ¶184; Menezes, p.170, §5.2.)

Manner, Rationale and Motivation to Combine: It would have been obvious to combine the techniques of DSS and Schneier with Menezes. (Katz, ¶¶185-193.) This combination would have predictably resulted in the technique of generating k in DSS with the following steps:

- (1) Generate a random number for KKEY according to DSS alone or in combination with Schneier (the claimed “**seed value SV**”), as explained for claim 1[a];
- (2) compute a hash of KKEY, $G(t, KKEY)$, according to DSS (the claimed “**output H(SV)**”), as explained for claim 1[b];
- (3) determine whether $G(t, KKEY)$ (“**output H(SV)**”) is less than **order q**,¹⁴ according to the teachings of Menezes;
- (4) accept $G(t, KKEY)$ (“**output H(SV)**”) if $G(t, KKEY)$ is less than q , according to the teachings of Menezes; and

¹⁴ As explained in footnote 8, the claimed “**said order q**” is satisfied by the value of q in DSS, a large prime number. (Katz, ¶143.)

(5) reject $G(t, KKEY)$ if it is **not** less than q , by repeating the method and going back to step (1), according to the teachings of Menezes.

(Katz, ¶185.) A person of ordinary skill would have found this to be a simple combination for many of the same reasons applicable to Rose, and would have had every expectation of success. (*Id.*, ¶¶186, 192-193.) DSS, Schneier, and Menezes are analogous references, and Menezes contains disclosures reasonably pertinent to a problem faced in DSS – generating a random number falling within a desired range, for use as a cryptographic key. (*Id.*, ¶¶186-187.)

Menezes does not specifically describe the potential bias that can occur from performing a modulo reduction on a random number. But this distinction is legally irrelevant because “the skilled artisan need not be motivated to combine [] for the same reason contemplated by [the inventor].” *In re Kahn*, 441 F.3d 977, 990 (Fed. Cir. 2006); *see also, e.g., In re Beattie*, 974 F.2d 1309, 1312 (Fed. Cir. 1992).

Moreover, the '961 patent admits that modulo bias was a known issue. ('961, 2:53-61; Katz, ¶157.) Accordingly, the modulo bias problem, and the reasons to avoid it, were already known. (Katz, ¶¶157, 191.)

Even if one assumes that a person of ordinary skill would have been unaware of the modulo bias issue, she would still have been motivated to use the technique in Menezes to solve a particular problem in implementing DSS – ensuring that the resulting k value falls within the desired range of 0 to $q-1$. (*Id.*, ¶188.) There were

a finite number of known and predictable solutions for obtaining a random number within a desired range, and the rejection technique described in Menezes was one of them, and a known alternative to the “modulo” technique. (*Id.*, ¶189.)

Moreover, Menezes expressly states that its technique generates “uniformly distributed” random numbers. (*See* Menezes, p.170, §5.2 (“A random bit generator can be used to generate (uniformly distributed) random numbers. For example, a random integer in the interval $[0, n]$ can be obtained... if the resulting integer exceeds n , one option is to discard it and generate a new random bit sequence.)).) It was well-known that uniformly distributed random numbers (*i.e.*, in which every number within the desired range is equally likely) are preferable for cryptographic keys. (Katz, ¶190 (citing Schneier, pp.421-22, §17.14; *see also id.*, p.173 (“Good keys are random-bit strings... If the key is 64 bits long, every possible 64-bit key must be equally likely.”))).) A person of ordinary skill would thus have been motivated to use the technique in Menezes to obtain uniformly distributed random numbers, resulting in a cryptographically stronger key k for DSS and, in turn, a stronger digital signature. (Katz, ¶¶189-190.)

- (v) **“if said output $H(SV)$ is accepted, providing said key k for use in performing said cryptographic function, wherein said key k is equal to said output $H(SV)$.” (1[g])**

This step adds little if anything that was not already addressed in limitations 1[c]-1[f]. As explained, DSS and Schneier, in combination with Menezes, disclose

a key generation system that accepts the hash of KKEY, $G(t, KKEY)$ (“**said output $H(SV)$** ”), if it is less than order q .

Appendix 3 of DSS identifies **Step 3(a)** of the algorithm for computing the value of k as: “ $k = G(t, KKEY) \bmod q$.” (DSS, Ex. 1004, p.14, App. §3.2.) Under the combination of DSS and Menezes, therefore, after the hash of KKEY (*i.e.*, $G(t, KKEY)$) is accepted, it is assigned as “**key k** ” and used in a cryptographic function. The “ $\bmod q$ ” operation specified in Step 3(a) of DSS need not be performed for the reasons explained above. Finally, as explained, “key k ” is used in the formula for computing the two signature components r and s , and thus “key k ” is “**use[d] in performing said cryptographic function.**” (DSS, *e.g.*, p.6, §5; Katz, ¶¶194-195.)

(b) Claim 2: “The method of claim 1 wherein another seed value is generated by said random number generator if said output is rejected.”

As explained for **Ground 1**, if “**output $H(SV)$** ” is rejected (*i.e.*, because $G(t, KKEY) \geq q$), another random number is generated for KKEY (as explained for claim 1[a]) and the process starts over. (Katz, ¶¶164, 196.)

(c) Claim 5: “The method of claim 1 wherein said order q is a prime number represented by a bit string of predetermined length L .”

As explained for **Ground 1**, DSS discloses that, in the context of generating random numbers for ephemeral key k , that “[t]hese randomly or pseudorandomly

generated integers are selected to be between 0 and the 160-bit prime q (as specified in the standard).” (DSS, p.13, App. §3; *see also id.*, p.6, §4 (defining q as “a prime ...”).) DSS thus discloses claim 5 for the reasons explained for claim 1. (Katz, ¶¶165, 197.)

(d) Claim 15, 16, and 19

Claims 15, 16, and 19 are substantially similar to claims 1, 2, and 4, with the exception that they recite a “computer readable medium” instead of methods. As explained for **Ground 1 (Part VI.C.2(d))**, a computer readable medium adds nothing of significance and is disclosed by DSS. These claims are obvious for the same reasons as claims 1, 2, and 4. (Katz, ¶¶169-173.)

D. Grounds 3 and 4: Obviousness of Claims 23, 24, and 27 Over Prior Art from Grounds 1, in Further View of Rao and Floyd

Grounds 3-4 address claims 23, 24, and 27, which recite similar limitations as claims 1, 2, and 5. Independent claim 23 adds a cryptographic unit comprising “**an arithmetic processor**” for performing the steps recited in independent claim 1. The ’961 specification devotes only one sentence to the “arithmetic processor,” and it is depicted as a nondescript box (**24**) in Figure 1. (’961, 3:33-38.)

DSS, Schneier, Rose, and Menezes do not appear to expressly disclose use of “an arithmetic processor,” but such a processor would have been obvious. Petitioner has cited Rao and Floyd, which provide information about computer architecture and organization that would have formed the basic background knowledge of a

person of ordinary skill. (Katz, ¶¶105-106.)

These references confirm that the claimed “**arithmetic processor**” merely describes the processing functionality built-in to commercially available processors available long before 2000. **Grounds 3 and 4** thus cite Rao and Floyd for the claimed arithmetic processor, and rely on the prior art mapping from **Grounds 1 and 2**, respectively, for all other claim limitations.

1. Summary and Date Qualification of Prior Art

(a) Rao (Ex. 1018)

Rao is a 1974 textbook that explains how digital computers are divided up into functional units or subsystems, including an “arithmetic processor” (AP). (Rao, p.39.) Rao explains that “[a]n arithmetic processor is the part of a computer that provides the logic circuits required to perform arithmetic operations such as ADD, SUBTRACT, MULTIPLY, DIVIDE, SQUARE-ROOT, etc.” (*Id.*, p.41.) Rao qualifies as prior art under §102(b). (Hall-Ellis Decl., Ex. 1032, ¶¶71-76.)

(b) Floyd (Ex. 1019)

Floyd explains that in more modern computers with a microprocessor, the arithmetic processing is performed by a portion of a computer’s central processing unit (CPU) known as an arithmetic/logic unit (ALU). (Floyd, pp.45-46.) Floyd qualifies as prior art under §102(b). (Hall-Ellis Decl., Ex. 1032, ¶¶64-70.)

2. Application of the Prior Art to the Challenged Claims

(a) Independent Claim 23

- (i) “A cryptographic unit configured for generating a key k for use in a cryptographic function performed over a group of order q , said cryptographic unit having access to computer readable instructions and comprises:” (23[Pre])**

The preamble of claim 23 is similar to independent claims 1 and 15, described in connection with **Ground 1** above. The “**cryptographic unit**” takes the form of a computing device (such as a computer) having a processor for executing computer readable instructions (software) for performing the DSA according to the teachings of DSS, adapted in accordance with the teachings of Schneier and Rose (for **Ground 3**) and Schneier and Menezes (for **Ground 4**), as detailed for **Grounds 1-2** above. (Katz, ¶202.) This mapping of the “cryptographic unit” to a computer is consistent with the ’961 specification, which explains that each function of the cryptographic unit (shown as cryptographic function **20** in Figure 1) “is controlled by a processor executing instructions to provide functionality and inter-operability as is well known in the art.” (’961, 3:41-44.)

DSS discloses that “[t]he DSA may be implemented in software, firmware, hardware, or any combination thereof.” (DSS, p.3.) It would have been obvious that an implementation of the DSA on a computer would include hardware (such as a processor) and computer executable instructions stored on some type of computer

readable medium (such as volatile or non-volatile memory). (Katz, ¶202.) It would also have been apparent and obvious to implement the proposed combinations of DSS, Schneier, and Rose (for **Ground 3**), or DSS, Schneier, and Menezes (for **Ground 4**), in software containing computer readable instructions. (*Id.*)

The remaining language of the preamble (“generating a key k for use in a cryptographic function performed over a group of order q ”) is identical to that of claims 1 and 15. For the reasons stated for **Grounds 1-2**, these portions of the preamble would have been obvious.

(ii) “an arithmetic processor for:” (23[a])

The ’961 patent devotes only one sentence to the “**arithmetic processor**,” and depicts it in Figure 1 as a nondescript black box (24). (*See* ’961, 3:33-38, Fig. 1.) The claimed “**arithmetic processor**” merely requires a processing unit capable of carrying out basic arithmetic and logical operations. (Katz, ¶204.)

It therefore would have been obvious – without even having to refer to any additional references – that the “cryptographic unit” used to carry out the combination of DSS, Schneier, and Rose (**Ground 3**), or DSS, Schneier, and Menezes (**Ground 4**), would have contained an “arithmetic processor” in order to carry out the functions described by those combinations. (*Id.*, ¶205.) The claimed “arithmetic processor” was a built-in feature of substantially all computer processors as of December 2000. (*Id.*, ¶¶207-208.) Performing steps 23[a]-23[g] via an

arithmetic processor would have been obvious.

To the extent there is any question, Rao and Floyd provide information about computer architecture and organization that would have formed the basic background knowledge of a person of ordinary skill. (*Id.*, ¶206.) Floyd and Rao are analogous references to DSS, Schneier, Rose, and Menezes in that Floyd and Rao provide information reasonably pertinent to the problem of how to implement the DSA in hardware and/or software. (*Id.*, ¶208.)

Rao is a 1974 textbook that explains how digital computers are divided into functional units or subsystems, including an “arithmetic processor” (AP). (Rao, p.39.) Rao explains that “[a]n arithmetic processor is the part of a computer that provides the logic circuits required to perform arithmetic operations such as ADD, SUBTRACT, MULTIPLY, DIVIDE, SQUARE-ROOT, etc.” (*Id.*, p.41.) **Floyd** explains that in a more modern computer that includes an integrated microprocessor, the arithmetic processor functionality is contained within a portion of a computer’s central processing unit (CPU) known as an **arithmetic/logic unit (ALU)**. (Floyd, pp.45-46.) “The arithmetic/logic unit (ALU), as the name implies, contains all of the circuitry necessary to perform the computer’s arithmetic and comparison operations.” (*Id.*) Floyd also explains that conventional microprocessors (such as Intel microprocessors) used on personal computers (PCs) included an ALU. (*Id.*, pp.107, 108 (Fig. 6.2) (“A microprocessor chip contains the ALU as well as the

control unit for a microcomputer (photo courtesy of Intel Corporation).”).)

It therefore would have been obvious to carry out the random number generation technique as described in the combinations of **Ground 1** and **Ground 2** using an “**arithmetic processor**,” such as an off-the-shelf, general purpose microprocessor available in December 2000 that carries out arithmetic and logical operations.¹⁵ (Katz, ¶207.) It therefore would have been obvious that implementing the DSA technique described in DSS, in the manner described in **Ground 1** and **Ground 2** – on any modern computer – would have resulted in the performance of its steps by an arithmetic processor as claimed. (*Id.*)

The remaining limitations of claim 23 are identical to claim 1. (Katz, ¶209 (showing side-by-side comparison chart).) These remaining limitations are therefore disclosed by and obvious over DSS, Schneier and Rose (**Ground 1**), and DSS, Schneier, and Menezes (**Ground 2**), as explained.

(b) Claims 24 and 27

Claims 24 and 27 are substantially similar to claims 2 and 5, respectively,

¹⁵ In a modern microprocessor, the ALU functionality is physically and functionally integrated with other functions of the processor. Petitioner is therefore not mapping the claimed “arithmetic processor” to just the ALU portion of a processor, but rather, to the microprocessor itself that includes the ALU functionality. (Katz, ¶207 n.27.)

Petition for *Inter Partes* Review of
U.S. Patent No. 7,372,961

except that they recite a “cryptographic unit,” which was addressed for claim 23[a].

Thus, these claims would have been obvious.

VII. CONCLUSION

Petitioner respectfully requests institution of IPR on the challenged claims.

Dated: April 3, 2019

Respectfully submitted,

COOLEY LLP
ATTN: Patent Group
1299 Pennsylvania Avenue NW
Suite 700
Washington, DC 20004
Tel: (650) 843-5001
Fax: (650) 849-7400

By: / Heidi L. Keefe /
Heidi L. Keefe
Reg. No. 40,673
Counsel for Petitioner

CERTIFICATE OF COMPLIANCE WITH WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), I certify that this petition complies with the type-volume limits of 37 C.F.R. § 42.24(a)(1)(i) because it contains 13,422 words, according to the word-processing system used to prepare this petition, excluding the parts of this petition that are exempted by 37 C.F.R. § 42.24(a) (including the table of contents, a table of authorities, mandatory notices, a certificate of service or this certificate word count, appendix of exhibits, and claim listings).

DATED: April 3, 2019

COOLEY LLP
ATTN: Patent Docketing
1299 Pennsylvania Avenue NW
Suite 700
Washington, D.C. 20004
Tel: (650) 843-5001
Fax: (650) 849-7400

/ Heidi L. Keefe /
Heidi L. Keefe
Reg. No. 40,673

CERTIFICATE OF SERVICE

I hereby certify, pursuant to 37 C.F.R. Sections 42.6 and 42.105, that a complete copy of the attached **PETITION FOR INTER PARTES REVIEW OF U.S. PATENT NO. 7,372,961**, including all exhibits (**Nos. 1001-1035**) and related documents, are being served via Federal Express on the 3rd day of April, 2019, the same day as the filing of the above-identified document in the United States Patent and Trademark Office/Patent Trial and Appeal Board, upon Patent Owner by serving the correspondence address of record with the USPTO as follows:

BlackBerry Limited - Direct Practice - (US Team)
ATTN: PATENT TEAM
2200 University Avenue E.
Waterloo ON N2K 0A7

And, via Federal Express upon counsel of record for Patent Owner in the litigation pending before the U.S. District Court for the Central District of California entitled *Blackberry Limited LLC v. Facebook, Inc.*, Case No. 2:18-cv-01844-GW-KS as follows:

James R. Asperger
Quinn Emanuel
865 S. Figueroa Street, 10th Floor
Los Angeles, CA 90017

DATED: April 3, 2019

/ Heidi L. Keefe /
Heidi L. Keefe
Reg. No. 40,673
COOLEY LLP
1299 Pennsylvania Ave. NW,
Suite 700
Washington, D.C. 20004
Tel: (650) 843-5001