UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MOBILEIRON, INC.,
Petitioner

v.

BLACKBERRY LIMITED,
Patent Owner

CASE NO. IPR2020-01741
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) 2
C. Lead and Back-Up Counsel under §42.8(b)(3) 6
II. Fee Payment 7
III. Requirements Under 37 C.F.R. §§ 42.204 7
A. Grounds for Standing 7
B. Identification of Challenge and Statement of Relief Requested 7
C. Considerations under 35 U.S.C. §325(d) 8
IV. Overview of the '961 Patent 8
A. Level of Ordinary Skill in the Art 8
B. Technology Overview 8
1. Cryptography and Digital Signatures 8
2. Modular Arithmetic, Prime Numbers, and Groups 10
3. Hash Functions and Random Number Generators 12
4. Rejection Sampling 13
C. The '961 Patent 14
D. The Challenged Claims 18
V. Claim Construction 19
VI. Overview of the Prior Art 20
A. Miyaji 20
B. Bellare 23
C. LEDA 24
VII. The Challenged Claims are Unpatentable 25
A. Ground 1: Miyaji 25
1. Claims 2, 16, 24 40
2. Claims 3, 17, and 25 41
3. Claims 4, 18 and 26 41

Table of Contents
(continued)

Page

B. Ground 2: Miyaji in view of Admitted Knowledge in the Art 42
1. Claims 1, 15, and 23 43
2. Claims 2, 16 and 24 44
3. Claims 3, 17 and 25 46
4. Claims 4, 18 and 26 47
5. Claims 5, 19 and 27 50
6. Claims 6, 20 and 28 51
7. Motivation to Combine 52
C. Ground 3: Bellare in View of LEDA 53
2. Claims 2, 16, and 24 66
3. Claims 3, 17, and 25 66
4. Claims 4, 18, and 26 67
5. Claims 5, 19 and 27 68
6. Claims 6, 20 and 28 68
7. Secondary Considerations 69
8. Motivation to combine 69
VIII. CONCLUSION 72

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 Professor Michael Goodrich, 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, *Digital Signature Standard (DSS)* (May 19, 1994)
1005	U.S. Patent 6,697,946 to Miyaji (PCT Application filed January 28, 1997, PCT Application published July 30, 1998, U.S. Patent issued February 24, 2004)
1006	"'Pseudo-random' Number Generation Within Cryptographic Algorithms: The DDS case," by Mihir Bellare, et al. (published in Proceedings of the Annual International Cryptology Conference (CRYPTO), pp. 277-291, Springer, 1997, as a part of the *Lecture Notes in Computer Science* (*LNCS*) book series, volume 129).
1007	*LEDA: a Platform for Combinatorial and Geometric Computing*, by Kurt Mehlhorn & Stefan Näher, Cambridge University Press, 1999
1008	Original Complaint filed April 27, 2020 in *MobileIron. Inc. v. Blackberry Corp., et al.*, Case No. 3:20-cv-02877 (N.D. Cal.)
1009	First Amended Complaint filed June 29, 2020, in *MobileIron. Inc. v. Blackberry Corp., et al.*, Case No. 3:20-cv-02877 (N.D. Cal.)
1010	First Amended Complaint filed in *BlackBerry Ltd. v. Facebook, Inc. et al.*, Case No. 2:18-cv-01844-GW-KS (C.D. Cal.), Facebook's Exhibit 1033 from *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)
1011	Excerpts from Menezes et al., Handbook of Applied Cryptography (1997)

Exhibit No.	Description of Document
1012	Dkt. Entry 157, *Corrected* Final Ruling on Claim Construction/*Markman* Hearing, filed April 5, 2019, entered April 11, 2019, in *BlackBerry Ltd. v. Facebook, Inc. et al.*, Case No. 2:18-cv-01844-GW-KS (C.D. Cal.)
1013	Dkt. Entry 652, Civil Minutes entered February 13, 2020, in *BlackBerry Ltd. v. Facebook, Inc. et al.*, Case No. 2:18-cv-01844-GW-KS (C.D. Cal.)
1014	Dkt. Entry 655, Sealed Minutes of Motion Hearing held February 27, 2020, before Judge George H. Wu, in *BlackBerry Ltd. v. Facebook, Inc. et al.*, Case No. 2:18-cv-01844-GW-KS (C.D. Cal.)
1015	Dkt. Entry 656, Joint Request To Vacate Pretrial Conference and Trial Date, filed March 19, 2020, in *BlackBerry Ltd. v. Facebook, Inc. et al.*, Case No. 2:18-cv-01844-GW-KS (C.D. Cal.)
1016	Dkt. Entry 657, Order dated March 24, 2020 Vacating Pretrial Conference and Trial Date, in *BlackBerry Ltd. v. Facebook, Inc. et al.*, Case No. 2:18-cv-01844-GW-KS (C.D. Cal.)
1017	Dkt. Entry 673, In Chambers – Order dated July 23, 2020, in *BlackBerry Ltd. v. Facebook, Inc. et al.*, Case No. 2:18-cv-01844-GW-KS (C.D. Cal.)
1018	Petition for *Inter Partes* Review, filed by Facebook, Inc., et al. in *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)
1019	Paper No. 14, Decision entered November 5, 2019, in *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB Nov. 5, 2019)
1020	Excerpts from Schneier, Applied Cryptography (2d ed. 1996), Exhibit 1008 from *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)
1021	Rose, Re: "Card-shuffling" algorithms, USENET, sci.crypt, sciath h 10, 1993), Facebook's Exhibit 1006 from *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)
1022	Excerpts from Rao, Error Coding for Arithmetic Processors (1974), Facebook's Exhibit 1018 from *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)
1023	Excerpts from Floyd, Essentials of Data Processing (1987), Facebook's Exhibit 1019 from *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)

Exhibit No.	Description of Document
1024	Declaration of Daniele Micchiancio
1025	Affidavit of Elizabeth Rosenberg
1026	Declaration of Sylvia Hall-Ellis, Ph.D, in Support of Petitioner's Petition for *Inter Partes* Review
1027	Excerpts from Blackberry's Supplemental Responses and Objections to Facebook's Interrogatories, Facebook's Exhibit 1033 from *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)
1028	Declaration of Dr. Jonathan Katz, Facebook's Exhibit 1002 from *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)
1029	Patent Owner's Preliminary Response filed in *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)
1030	Declaration of Markus Jakobsson, Exhibit 2001 filed by Patent Owner in *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)
1031	Excerpts from *Microsoft Computer Dictionary, Fourth Edition,* 1999
1032	Federal Information Processing (FIPS) Publication 186-2, *Digital Signature Standard (DSS)* (December 15, 1998)
1033	Federal Information Processing (FIPS) Publication 186-2, *Digital Signature Standard (DSS)* (January 27, 2000)
1034	John von Neumann, *Various Techniques Used in Connection with Random Digits*, 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, Facebook's Exhibit 1017 from *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)
1035	Excerpts from M.T. Goodrich and R. Tamassia, *Algorithm Design and Analysis,* Wiley, 2015

Exhibit No.	Description of Document
1036	Federal Information Processing (FIPS) Publication 186-3, *Digital Signature Standard (DSS)* (June 2009), Patent Owner's Exhibit 2003 from *Facebook, Inc., et al. v. BlackBerry Ltd.*, IPR2019-00923 (PTAB)

MobileIron, Inc. ("Petitioner") seeks *Inter Partes* review of claims 1-6, 15-20, and 23-28 of U.S. Patent No. 7,372,961 ("the '961 patent") (Ex. 1001). Petitioner's request is supported by the Expert Declaration of Professor Michael Goodrich, Ph.D. (Ex. 1002), and other exhibits submitted herewith. The challenged claims cover a method for reducing bias when generating a cryptographic key. The prior art anticipates and renders the claimed subject matter obvious.

The Board should institute review because there is at least a reasonable likelihood that Petitioner will prevail with respect to at least one challenged claim. 35 U.S.C. § 314(a). This is Petitioner's first petition challenging any claim of the '961 patent, and the petition cites prior art and raises arguments that have not previously been presented to the Office.

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

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

MobileIron, Inc. is the real party-in-interest to this IPR petition.[1]

[1] Ivanti, Inc. publicly announced that it has signed "definitive agreements" to acquire Petitioner. *See* https://www.ivanti.com/company/press-releases/2020/ivanti-announces-strategic-acquisitions-of-mobileiron-and-pulse-secure (last visited October 2, 2020). As of the filing of this Petition, the acquisition by Ivanti, Inc. of Petitioner has not occurred.

B. Related Matters under §42.8.(b)(2)

The '961 patent is the subject of a declaratory judgment claim brought by Petitioner against BlackBerry Corporation (Patent Owner) and BlackBerry Ltd. (collectively, Defendants): *MobileIron. Inc. v. Blackberry Corp., et al.*, Case No. 3:20-cv-02877 (N.D. Cal.) (the "Northern District Action"). Petitioner filed the original complaint (Ex. 1008, ¶¶83-89) containing that declaratory judgment claim and other claims on April 27, 2020. Petitioner filed a First Amended Complaint (Ex. 1009, ¶¶84-90) on June 29, 2020. Petitioner has not asserted in those complaints that any claim of the '961 patent is invalid. Defendants have not yet answered, and no trial date has been set.

On September 17, 2020, Patent Owner filed five IPR petitions against Petitioner related to patents asserted by Petitioner against Patent Owner in the Northern District Action: (1) *BlackBerry Ltd. v. MobileIron, Inc.*, IPR2020-01519 (PTAB) (2) *BlackBerry Ltd. v. MobileIron, Inc.*, IPR2020-01593 (PTAB); (3) *BlackBerry Limited v. MobileIron, Inc.*, IPR2020-01594 (PTAB); (4) *BlackBerry Limited v. MobileIron, Inc.,* IPR2020-01604 (PTAB); and *BlackBerry Limited v. MobileIron, Inc.*, IPR2020-01634 (PTAB). The Board has not rendered any ruling on these petitions, and none of these petitions relates to the '961 patent.

The '961 patent is one of several patents involved in pending litigation in the Central District of California, *BlackBerry Ltd. v. Facebook, Inc. et al.*, Case No.

2:18-cv-01844-GW-KS (C.D. Cal.) ("the *Facebook* litigation"). BlackBerry's First Amended Complaint alleges that Facebook and its affiliates infringe the '961 patent. Ex. 1010, ¶¶89-132. The district court in the *Facebook* litigation issued a Corrected Final Ruling on Claim Construction/*Markman* Hearing, where it construed one term of the '961 patent, addressed in Section V below. (Ex. 1012, 29-35.) The district court has stayed all claims in the *Facebook* litigation, including with respect the '961 patent, pending the outcome of IPR proceedings directed to other patents. *See* Ex. 1017, 1-2 & n.1; *see also* Exs. 1015-1016.

Facebook previously filed a Petition for *Inter Partes* Review of the '961 patent, relying on different prior art than that raised in the present Petition: *Facebook, Inc., et al. v. BlackBerry Ltd.*, 2019IPR-00923 (PTAB) (Ex. 1018). Facebook raised four obviousness challenges: (a) claims 1, 2, 5, 15, 16 and 19 were obvious in view of Federal Information Processing (FIPS) Publication 186, Digital Signature Standard (DSS) (May 19, 1994) (Ex. 1004) ("DSS"), in view of Schneier, Applied Cryptography (2d ed. 1996) (Ex. 1020) ("Schneier") and Rose, Re: "Card-shuffling" algorithms, USENET, sci.crypt, sciath h 10, 1993) (Ex. 1021) ("Rose"); (b) claims 1, 2, 5, 15, 16 and 19 were obvious in view of DSS, Schneier, and Menezes et al., Handbook of Applied Cryptography (1997) (Ex.

1011) ("Menezes")[2]; (c) claims 23, 24 and 27 were obvious in view of DSS, Schneier, Rose, Rao, Error Coding for Arithmetic Processors (1974) (Ex. 1022) ("Rao") and Floyd, Essentials of Data Processing (1987) (Ex. 1023) ("Floyd"); and (d) claims 23, 24 and 27 were obvious in view of DSS, Schneier, Menezes, Rao and Floyd.

The Board issued a decision denying institution of Facebook's Petition on November 5, 2019. *Id.*, Paper 14. Ex. 1019. The Board agreed with the District Court that the limitation in the independent claims requiring "reducing mod q" should be construed to mean "computing the remainder of dividing a value by q." *Id.*, 7. Nonetheless, the Board found that none of the combinations taught the steps in each of independent claims 1, 15 and 23 of evaluating the output of a hash function performed on a random number to determine if that output is less than a group of order q, and, if it is not, rejecting that value and repeating the method – what the Board called the "the "determining," "accepting," "rejecting," and "repeating" steps. *See id.,* 13-15. The Board found that DSS performs a modulo operation on the output of the hash of a random value and stores that result as ephemeral private key k, but does not contemplate evaluating the output of the hash function prior to performing the modulo operation or iteratively repeating the

[2] Petitioner has attached excerpts of Menezes at Ex. 1011 that include all pages cited by Facebook in its IPR petition, as well as some additional pages referred to by Petitioner and its expert, Dr. Goodrich, in his declaration at Ex. 1002.

process of selecting a new random number and performing a hash function on that number until a value less than q is obtained. *Id.,* 14. Similarly, the Board concluded that Rose does not (1) disclose hashing the random number, (2) contemplate application to a cryptographic key generation algorithm, (3) contemplate avoiding bias in the output of a hash function performed on a random number, or (4) contemplate that avoiding modulo bias requires iteratively repeating both the selection of a random number and hashing that number. *Id.*, 14-15.

The Board likewise concluded that Menezes failed to disclose the "determining," "accepting," "rejecting," and "repeating" steps performed on the output of a hash function. *Id.*, 19. Menezes does not mention hashing, and does not contemplate accepting or rejecting the result of performing a hash function on a randomly generated seed value, or iterative repetition of generating a random number and performing a hash function on that number when the output of the hash function falls outside of a desired range. *Id.*, 20.

The prior art that Petitioner raises in this Petition, which was not before the Examiner during prosecution of the '961 patent, overcomes the issues identified by the Board in the *Facebook* IPR. Miyaji, Bellare and LEDA all explicitly teach repeatedly applying a hash function to a seed value generated from a random number generator, until a value less than q is obtained, prior to reducing modulo q, and storing the result as a key k. *See* Sections VII.A-C, *infra*.

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

Petitioner provides the following designation of counsel.

LEAD COUNSEL	BACK-UP COUNSEL
Sanjeet Dutta Registration No. 46,145 Goodwin Procter LLP 601 Marshall Street Redwood City, CA 94063 Email: sdutta@goodwinlaw.com Tel: (650) 752-3100 Fax: (650) 853-1038	Rachel Walsh (admission pro hac vice pending) Goodwin Procter LLP Three Embarcadero Center, 24th Floor San Francisco, California 94111 rwalsh@goodwinlaw.com Telephone: (415) 733-6000 Facsimile: (415) 677-9041
	I. Neel Chatterjee (admission pro hac vice pending) Goodwin Procter LLP 601 Marshall Street Redwood City, CA 94063 Email: nchatterjee@goodwinlaw.com Tel: (650) 752-3100 Fax: (650) 853-1038
	Monte M.F. Cooper (Admission *pro hac vice* pending) Goodwin Procter LLP 601 Marshall Street Redwood City, CA 94063 Email: mcooper@goodwinlaw.com Tel: (650) 752-3100 Fax: (650) 853-1038

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

Petitioners authorize the U.S. Patent & Trademark Office to charge Deposit Account No. 506989 for the fee set in 37 C.F.R. § 42.15(a) for this Petition and further authorizes any additional fees to be charged to this Deposit Account.

III. REQUIREMENTS UNDER 37 C.F.R. §§ 42.204

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 Relief Requested

Petitioner requests institution of IPR based on the following grounds:

Ground	Claims	Basis for Rejection
1	1-4, 15-18, and 23-26	Anticipated pursuant to 35 U.S.C. § 102 by Miyaji (Ex. 1005)
2	1-6, 15-20, and 23-28	Obviousness in view of Miyaji (Ex. 1005) and admitted knowledge of a person of ordinary skill in the art
3	1-6, 15-20, and 23-28	Obviousness in view of Bellare (Ex. 1006) and LEDA (Ex. 1007)

Submitted with this Petition is the Declaration of Professor Michael Goodrich ("Goodrich"), a qualified technical expert. Ex. 1002, ¶¶2-12, Ex. A.

C. Considerations under 35 U.S.C. §325(d)

This Petition does not present "the same or substantially the same prior art or arguments previously were presented to the Office." 35 U.S.C. §325(d). None of the prior art references used herein were cited during prosecution of the '961 patent, or in the Facebook IPR.

IV. OVERVIEW OF THE '961 PATENT

A. Level of Ordinary Skill in the Art

A person of ordinary skill in the art as of the '961 patent's December 27, 2000 priority date would have possessed a bachelor's degree in computer science, mathematics, applied mathematics, or a related field, and (1) two or more years of experience in applied cryptography or computer security software engineering, and/or (2) an advanced degree in computer science or a related field. Ex. 1002, ¶41.

B. Technology Overview

The '961 patent, titled "Method of Public Key Generation" relates to eliminating a potential bias that can occur when generating a private key for use in a cryptographic function. Ex. 1001, Abstract.

1. Cryptography and Digital Signatures

Cryptography is the study of mathematical techniques for performing functions related to information security. Ex. 1002, ¶50. For example, suppose Alice and Bob share a secret number, called a "key," k. Such cryptographic keys

are numbers or strings of bits used for encrypting or decrypting information. *Id.* Alice can scramble a "plaintext" message, M, using the key k, to produce a "ciphertext," C, and send it to Bob, who can unscramble C using k to retrieve the message. *Id.* This process is called "symmetric" cryptography, because the same key, k, is used for both encryption and decryption. *Id.*, ¶51.

Before Alice and Bob can engage in symmetric encryption/decryption, they must share the secret key, k. *Id.*. Alternatively, in public-key cryptography, or "asymmetric" cryptography, Bob generates two keys—a public key, K_{pub}, and a private key, K_{priv}. Ex. 1002, ¶52. Bob keeps the private key secret and does not share it with anyone, whereas he openly shares his public key. *Id.* Alice can encrypt a plaintext message using Bob's public key and send him the resulting ciphertext. *Id.* Bob can then decrypt the ciphertext using his private key to retrieve the message. *Id.* Without knowing Bob's private key, it is difficult for a third party to decrypt the plaintext from a ciphertext encrypted with Bob's public key. *Id.*

A variety of cryptosystems have been developed using such public-key cryptosystems. *Id.*, ¶53; Ex. 1001, 1:17-20. Some of these protocols use both long term and short term, or "ephemeral," key pairs. *See* Ex. 1001, 1:38-42. An ephemeral private key is generated at the start of a session, usually by a random number generator (RNG). *Id.* 1:42-44. The ephemeral public key also is generated, and the resultant key pair used for the duration of the session. *Id.* 1:44-

47; *see also* Ex. 1002, ¶¶53-54. Long-term public keys are utilized across multiple sessions. Ex. 1001, 1:47-49. Public-key cryptography also is used for digital signatures. Ex. 1002, ¶55. Here, Bob can perform a form of encryption on a message, M, using his private key to produce a digital signature, S, such that anyone knowing Bob's public key can verify that S is a digital signature for M. *Id.* Digital signatures are designed so that it is difficult for someone to forge a signature for a message without knowledge of Bob's private key. *Id.*; Ex. 1001, 1:26-32. Thus, signing a message involves encryption with a private key and verifying a signature involves decrypting a signature with a public key. Ex. 1002, ¶56. Prior art protocols for both applied cryptography and digital signatures are described in the '961 patent, and include the Digital Signature Algorithm ("DSA"). *See* Ex. 1001, 1:52-56.

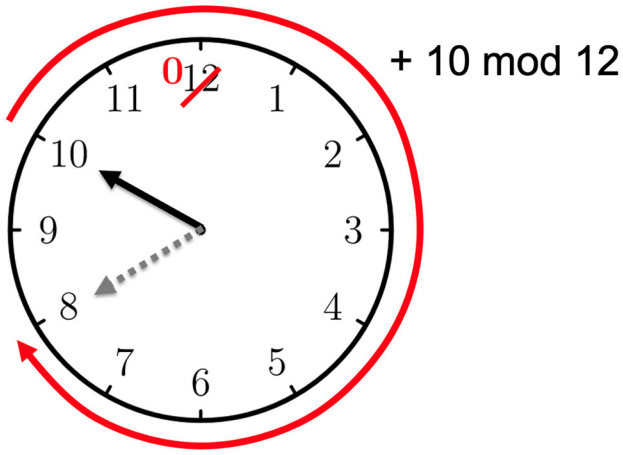
2. Modular Arithmetic, Prime Numbers, and Groups

Messages can be represented as integers, which in turn can be represented as strings of bits. Ex. 1002, ¶¶58-59. A set of such integers, {0, 1, …, n-1}, is denoted as "$\mathbf{Z}_n$." *Id.*, ¶66.

Representing messages as an integers permits modular arithmetic operations on messages, which can be used for encryption or decryption. *Id.*, ¶¶60, 62-63. In modular arithmetic, the ***modulo*** operation, "x mod n," returns the remainder after

dividing x by n. *Id.*, ¶60. Thus, 10 mod 3 is 1 (i.e. 10 divided by three has a remainder of 1). *Id.*

Modular arithmetic can be analogized as "clock arithmetic." *Id.*, ¶61. For example, if "12:00" is replaced on a clock with the number "0," adding hours going clockwise around the clock equates to "addition mod 12." *Id.* Thus, if a mother asks her son at 10:00 to set an alarm for 10 hours later, he would calculate "10+10 mod 12 = 8" and set the alarm for 8:00:



Id.

Modular arithmetic also can be used to define the order for any element, a, in the multiplicative group $\mathbf{Z}^*_n$, where n is a prime number, to be the smallest positive number, k, such that $a^k \pmod n = 1$. *Id.*, ¶¶64-68. The following table shows the order of each number in $\mathbf{Z}^*_{21} = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$:

$a \in \mathbb{Z}^*_{21}$	1	2	4	5	8	10	11	13	16	17	19	20
order of a	1	6	3	6	2	6	6	2	3	6	6	2

Id., ¶69. In this scheme, orders of the elements 4, 8, 13, 16, and 20 in $\mathbf{Z}^*_{21}$ are prime. *Id.*, ¶70. As a result of a theorem known as "Cauchy's theorem," if G is a finite group and q is a prime number dividing the order of G, then G contains an element of order q. *Id.* For each prime number, q, that divides (p-1), where p is a prime number, there is an element x in $\mathbf{Z}^*_p$ such that x has order q (i.e. $x^q \pmod p = 1$). *Id.* This concept is used to establish parameters for the Digital Signature Algorithm (DSA) described in the '961 patent. *Id.*, ¶¶70-71.

3. Hash Functions and Random Number Generators

A "hash function," h(x), is used to produce a relatively short digest or fingerprint for a message, such as by mapping a value x to a 160-bit digest. *Id.*, ¶72. One type of hash function is a "multiplicative hash function" of the following form:

$$h(x) = (a*x + b) \bmod n,$$

where a, b, and n, are positive integers (e.g., 160-bit numbers). *Id.* Another type of hash function, known as an "exponential hash function" or "discrete logarithm hash function," is reflected by the equation: $h(x) = g^x \bmod n$, for suitable parameters g and n. *Id.* The composition of two hash functions is itself a hash function. *Id.*, ¶73. That is, if f( ) and g( ) are hash functions, then h(x) = f(g(x)) also is a hash function. *Id.*

In cryptography, it is often useful for the result of a hash function, h(x), applied to an input, x, to appears random, so that it is difficult to determine the value of x from the value of h(x). *Id.*, ¶74.

Random number generators ("RNGs") can generate a sequence of numbers that have the properties of random numbers. *Id.*, ¶¶75-76. Such RNGs typically output numbers that are uniformly distributed within a large, specified range, such as 32-bit numbers with values ranging from zero to 2^{32}-1. *Id.*75.

A pseudo-random number generator is an RNG that takes a number, s, known as a "seed," and produces a sequence of numbers from that seed in which the output numbers seem random. *Id.*, ¶77. The seed, s, can either be truly random or may come from an RNG. *Id.* A pseudo-random number generator can repeatedly apply a hash function to the output of the previous iteration, starting from a seed value, such as from user input or an output from an RNG. *Id.*, ¶78. For instance, repeatedly applying a multiplicative hash function to the output of the previous iteration is known as a "linear congruential generator." *Id.*

4. Rejection Sampling

Many cryptographic applications also need random numbers that fall uniformly within a smaller range than that of an underlying RNG, so as to avoid "bias" in the key generation. *Id.*, ¶79. Bias results because performing a reduction modulo q to get a number within a smaller range favors the lower

numbers within the range. *Id.*, ¶¶ 79-80. For example, to generate random numbers between 0 and 150, one could first generate an 8-bit random number (which would be in the range from 0 to 255), then return the value r mod 151. In performing this reduction, the number 0 is twice as likely to be output than the number 150, even if the random number generators are truly random, resulting in bias toward the smaller numbers. *Id.* ¶80.

A well-known way to address this is "rejection sampling," which is a simple method for generating random numbers in a smaller range, by uniformly generating a random in a larger range and rejecting the point if it does not belong to the smaller range. *Id.*, ¶¶81-88. For example, to generate a random number uniformly chosen from the interval from 1 to q-1, one could first generate an L-bit random number (where $q<2^L-1$), and then reject the number if it is not less than q. *Id.*, ¶81. This process repeats until it achieves a random number less than q. *Id.* Unlike the biased choice of reducing the result modulo q, this process is guaranteed to produce a random number uniformly chosen to be less than q. *Id.*

C. The '961 Patent

The '961 patent claims priority to Canadian Patent Application No. 23229590, filed on December 27, 2000. Ex. 1001, Foreign Publication Data. The underlying U.S. application was filed on December 26, 2001, originally with 14 proposed claims. After the claims were repeatedly rejected in several office

actions, resulting in multiple amendments (the final of which added issued claims 15-30) by the applicant, the patent issued on May 13, 2008. *Id.*, Face Page; Ex. 1003, at 66-76, 81-95, 110-121, 126, 129-133, 138-151, 163, 182-198, 252-257.

The '961 patent describes a method for avoiding potential bias in the generation of a cryptographic key. Ex. 1001, 3:1-3. The patent purports to remedy modulo bias in the generation of a key k in connection with DSA, described in a prior art standard known as the Digital Signature Standard (DSS) (Ex. 1004) published by the National Institute of Standards and Technology ("NIST"). Ex. 1001, Abstract, 2:32-36; Ex. 1002, ¶¶111-113. The '961 patent describes how a version of DSS known as FIPS 186-2, published in 2000, selects an integer for use as a private key. Ex. 1001, 2:36-46; Ex. 1002, ¶114.

The '961 patent focuses on how DSS generates the ephemeral cryptographic key, *k*. Ex. 1002, ¶¶112, 115. 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. Ex. 1001, 2:40-43. The bit string represents an integer between 0 and 2^{160}-1. *Id.* 2:43-44. However, because this integer can be greater than the prime q, DSS reduces the integer by computing "SHA-1(seed) mod q," which ensures that the value of the key k will fall within the desired range of 0 to q-1. Ex. 1002, ¶117.

Using a "mod q" operation could introduce some modulo bias in favor of smaller values. *Id.*, ¶118. The '961 patent admits that such potential bias was known. Ex. 1001, 2:53-55; Ex. 1002 ¶118. The '961 patent avoids this bias by employing a well-known prior-art hashing and rejection sampling techniques whereby hashed seed values that are not less than q are rejected, and new hashed seed values are produced until one less than q is obtained. Ex. 1001, 4:11-17; Ex. 1002 ¶¶119-121.

Figure 2 illustrates the rejection sampling technique described in the '961 patent:

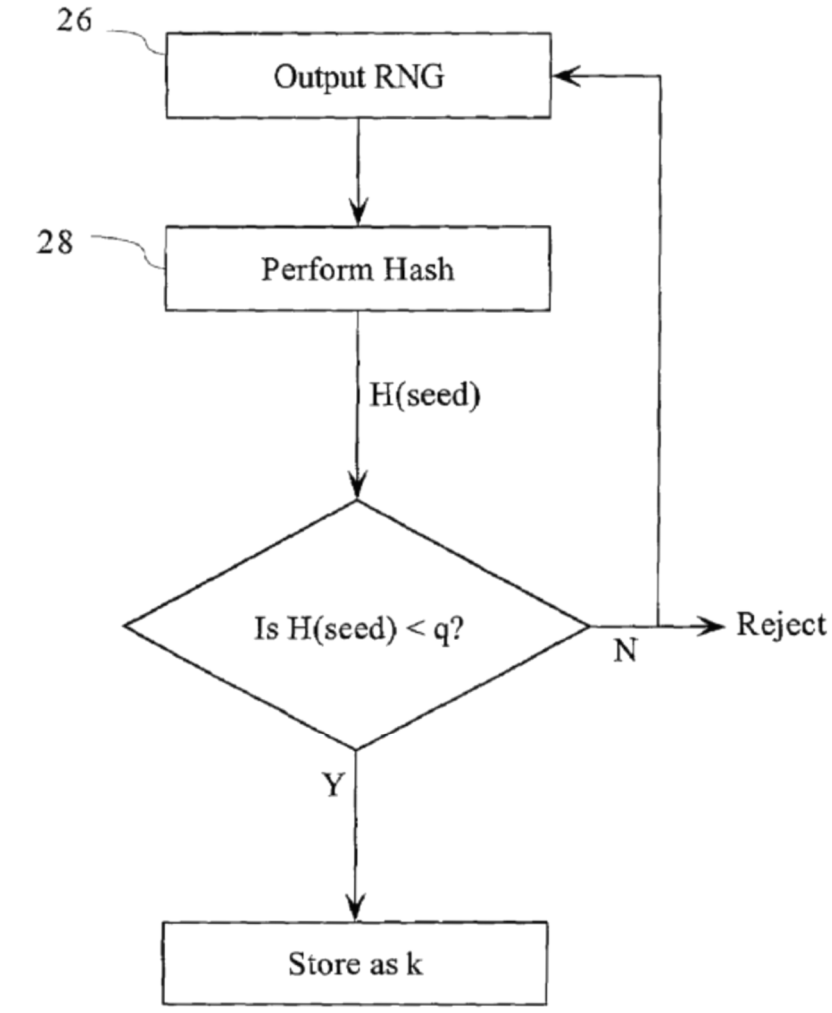


Fig. 2

Ex. 1001, Fig. 2; Ex. 1002, ¶121. The method shown in Figure 2 "originates by obtaining a seed value (SV) from the random number generator 26." Ex. 1001, 3:64-66. The number output from the RNG then is provided to hash function 28 in order to produce an output – H(seed) – of the correct length. *Id.*, 4:6-10. This is the same value represented as "SHA-1(seed)" in the Background description of DSA. *Id.*, 2:40-46; Ex. 1002, ¶121. The hashing function is used "to yield a bit string of predetermined length, typically 160 bits." Ex. 1001, 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. Ex. 1002 ¶121. If the answer is "yes," H(seed) is accepted and used as the cryptographic key k. Ex. 1001, 4:11-13. 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." *Id.*, 4:13-17; Ex. 1002, ¶121.

D. The Challenged Claims

Petitioner challenges claims 1-6, 15-20, and 23-28, of which claims 1, 15, and 23 are independent. Claims 1-6 are representative of all of the claims:

> 1**[a]** 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[b]** generating a seed value SV from a random number generator;
>
> **1[c]** performing a hash function H( ) on said seed value SV to provide an output H(SV);
>
> **1[d]** determining whether said output H(SV) is less than said order q prior to reducing mod q;
>
> **1[e]** 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;
>
> **1[f]** rejecting said output H(SV) as said key if said value is not less than said order q;
>
> **1[g]** if said output H(SV) is rejected, repeating said method; and

> **1[h]** 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).
>
> 2. The method of claim 1 wherein another seed value is generated by said random number generator if said output is rejected.
>
> 3. The method of claim 1 wherein the step of accepting said output as a key includes a further step of storing said key.
>
> 4 The method of claim 1 wherein said key is used for generation of a public key.
>
> 5. The method of claim 1 wherein said order q is a prime number represented by a bit string of predetermined length L.
>
> 6. he method of claim 5 wherein said output from said hash function is a bit string of predetermined length L.

Ex. 1001, 5:33-62. Many limitations in the claims recite features that are admitted prior art from the DSA/DSS. Ex. 1002, ¶133. The other two groups of challenged claims (*i.e.,* claims 15-20 and claims 23-28) recite computer readable medium and apparatus claims corresponding to the subject matter in claims 1-6, respectively.

V. CLAIM CONSTRUCTION

As noted above, Patent Owners have not yet responded to Petitioner's declaratory judgment claim directed to the '961 patent. Accordingly, the District Court has not yet construed any term from the patent. However, Petitioner is aware that the parties to the pending *Facebook* litigation dispute the meaning of "**reducing mod q**" as recited in each independent claim. In that action, the parties agreed that the operation "**mod q**" determines the remainder of dividing a number by **q** but disagreed on whether the operation must ***always*** result in a lower value.

Here, therefore, the Board should adopt the constructions for this term reached by both the District Court in the *Facebook* litigation and the Board in its denial of Facebook's IPR petition. Namely, the Board should construe "reducing mod q" as "determining the remainder of dividing a number by q." As both the District Court and the Board previously recognized, 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. Ex. 1012, 29-35; Ex. 1019, 7; *see also* Ex. 1002, ¶¶134-144.

VI. OVERVIEW OF THE PRIOR ART

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. This concept, however, was well-known and described verbatim in the prior art.

A. Miyaji

U.S. Patent 6,697,946 ("Miyaji") derives from a PCT application filed on January 28, 1997, which published on July 30, 1998. Ex. 1005. The reference therefore is prior art to the '961 patent pursuant to at least 35 U.S.C. § 102(b).

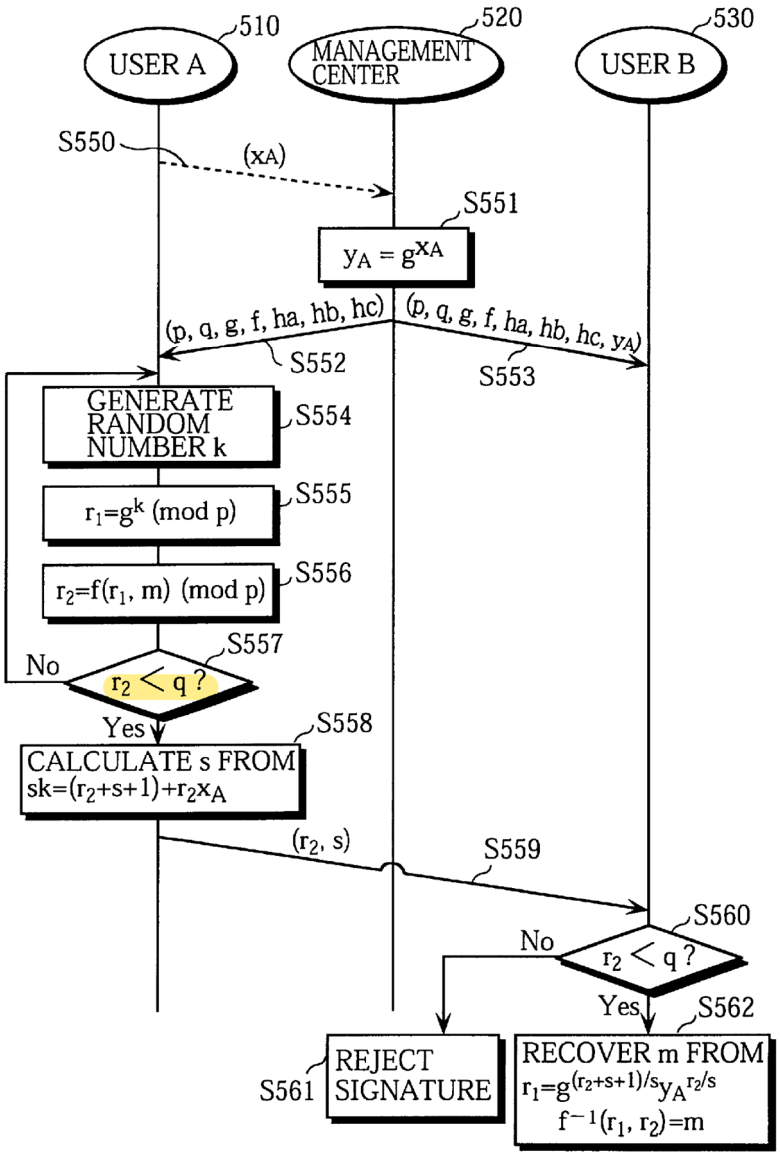
Miyaji discloses several cryptographic functions involving public-key cryptography, including the generation of public keys and private keys,

encrypting/decrypting messages, and signing messages. Ex. 1002, ¶91. For example, Miyaji uses random numbers, and the difficulty of the discrete logarithm problem, to overcome security vulnerabilities of previous digital signature schemes. *Id.* Significantly, Miyaji discloses the same random number generation, hashing, and rejection sampling methods disclosed and claimed in the '961 Patent. *Id.* ¶¶92-93.

Specifically, Miyaji discloses that a management center determines a public key y_A of user A using A's secret key x_A and announces the public key y_A to user B. Ex. 1005, Abstract; 7:58-8:7; FIG 5. User A repeats generation of a random number k and calculation of a "commitment" r_1 until the "masked message" r_2 and q meet a condition $r_2<q$. *Id.* Abstract; 8:24-45; FIG. 5. If the condition is met, user A finds s using a mod q calculation, and sends a ciphertext (r_2,s) to user B. *Id.* Abstract; 8:47-56; FIG. 5. User B rejects the ciphertext if $q \leqq r_2$. *Id.* Abstract; 8:43-45; FIG. 5. If $r_2<q$, user B recovers a message m by calculating r_1 using a mod p calculation. *Id.* Abstract; 8:47-54; FIG. 5.

The similarity of Miyaji's algorithm to that disclosed in the '961 patent is shown by Figure 5 of Miyaji, illustrated below:

FIG. 5



Id., FIG. 5 (highlighting added). The step 557, in which a calculation is made whether $r_2<q$, is the same step as reflected in Figure 2 of the '961 patent, where it is determined whether or not H(seed) is less than q. Ex. 1002, ¶94. This is reflected in a side-by-side comparison of step 557 from Miyaji's Figure 5 with Figure 2 of the '961 patent:

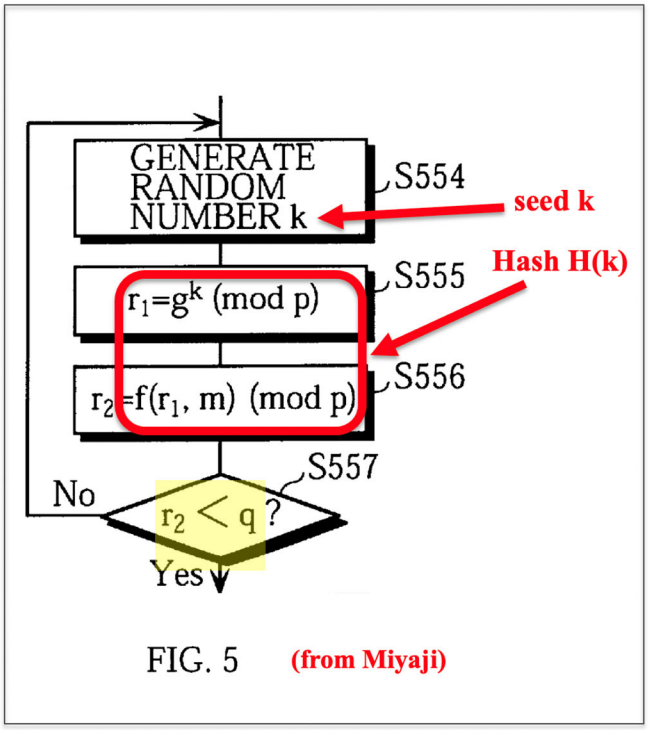


FIG. 5 **(from Miyaji)**

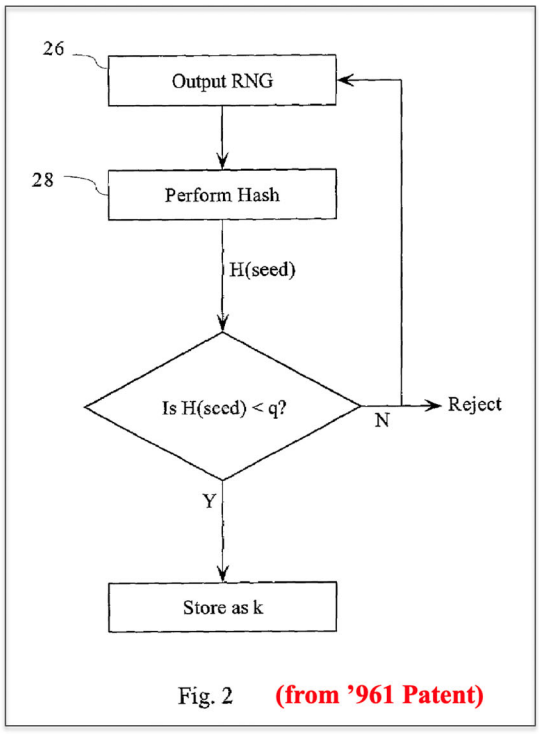


Fig. 2 **(from '961 Patent)**

Ex. 1005, FIG. 5 (annotations added); Ex. 1001, Fig. 2 (annotations added); *see also* Ex. 1001, 4:11-17; Ex. 1002, ¶¶93-94, 167-169.

B. Bellare

"'Pseudo-random' Number Generation Within Cryptographic Algorithms: The DDS case," by Mihir Bellare, *et al.* ("Bellare") (Ex. 1006), was published in Proceedings of the Annual International Cryptology Conference (CRYPTO), pp. 277-291, Springer, 1997, as a part of the *Lecture Notes in Computer Science* (*LNCS*) book series, volume 1294. Ex. 1024, ¶¶1-10; Ex. 1026, ¶¶52-58 *see also* Ex. 1002, ¶95. Bellare shows that if the random numbers generated in DSS's Digital Signature Algorithm (DSA) are generated according to a linear congruential random number generator, then signatures generated by the algorithm

are vulnerable to an attack that can reveal the secret key used. Ex. 1002, ¶96. Bellare states, "This illustrates the high vulnerability of the DSS to weaknesses in the underlying random number generation process." Ex. 1006, 277.

Bellare describes "the scheme" of DSA, which uses a prime number p, a prime number q which divides p-1, and an element $g \in \mathbf{Z}^*_p$ of order q (chosen a $g = h^{(p-1)/q}$ where h is a generator of the cyclic group $\mathbf{Z}^*_p$). *Id.*, 281. The secret key of a user is a random integer x in the range $\{O, \ldots, q-1\}$, and the corresponding public key is $y = g^x \bmod p$. *Id.* The signer generates a random number $k = \{1, \ldots, q-1\}$, which Bellare calls a "*nonce,*" and the dependence of DSA on the randomness of this nonce is described as a "drawback." *Id.* This drawback is what Bellare evaluates using hashing and modular calculation. *Id.*; Ex. 1002, ¶¶97-98.

C. LEDA

The textbook *LEDA: a Platform for Combinatorial and Geometric Computing*, by Kurt Mehlhorn and Stefan Näher, Cambridge University Press, 1999 ("LEDA") (Ex. 1007) was publicly available in 1999 prior to the priority date for the '961 Patent. Ex. 1025, ¶¶1-7; Ex. 1026, ¶¶59-64. LEDA describes methods for implementing data structures and algorithms in the programming language C++. It includes disclosures on basic data types, including random sources and hash functions, among other operations. Ex. 1002, ¶¶101-102. LEDA discloses the same algorithm as the '961 Patent. *Id.* ¶102; Ex. 1007, 92. LEDA's

equations use an RNG and a hash function, which is initialized with a random seed. Ex. 1002, ¶103. Starting with the first number generated, each number returned by the "random source," which is denoted "internal_source," is generated from a hash function applied to a seed value from an RNG. *Id.*; Ex. 1007, 93. LEDA performs rejection sampling and repetition as reflected by the following "while" operation:

```
while ( x > diff) x = internal_source() & pat;
```

Id.; Ex. 1002, ¶103. The "& pat" portion is an "and" function that performs a bitwise-and with the mask "pat," so that the result is a hash value in the range 0 to 2^p-1 – that is, a p-bit number. *Id.* ¶104.

VII. THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. Ground 1: Miyaji

Miyaji anticipates claims 1-4, 15-18, and 23-26 of the '961 patent. Ex. 1002, ¶¶145-182.

1. 1[a]: 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:

15[a]: 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:

23[a]: 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:

Miyaji discloses the preambles of independent claims 1, 15, and 23. Ex. 1002, ¶¶146-147. Miyaji states:

> The present invention relates to an apparatus for performing secret communications and digital signatures by a public key cryptosystem, and especially relates to a message recovery signature apparatus whose security is based on the discrete logarithm problem.

Ex. 1005, 1:5-10. Miyaji involves public-key cryptographic functions performed over a group of order q, in which Miyaji's methods generate cryptographic keys. Ex. 1002, ¶149.

Miyaji's method employs a group of order q:

> The message recovery signature scheme used in this system is a public key cryptosystem that uses the discrete logarithm problem as the founding principle for the security, and is based on operations over a finite field GF(p) where p (512 bits long) is a prime number, g is an element, and q (256 bits long) is the order of g.

Ex. 1005, 4:12-34; *see also id.* 7:9-23. Miyaji's signature method first sets up system parameters that include a prime number, p, and an element, g, that is a member of the field GF(p), where all arithmetic is modulo p, such that q is the order of g. *Id.,* Ex. 1002, ¶149. To digitally sign a message, m, the method begins

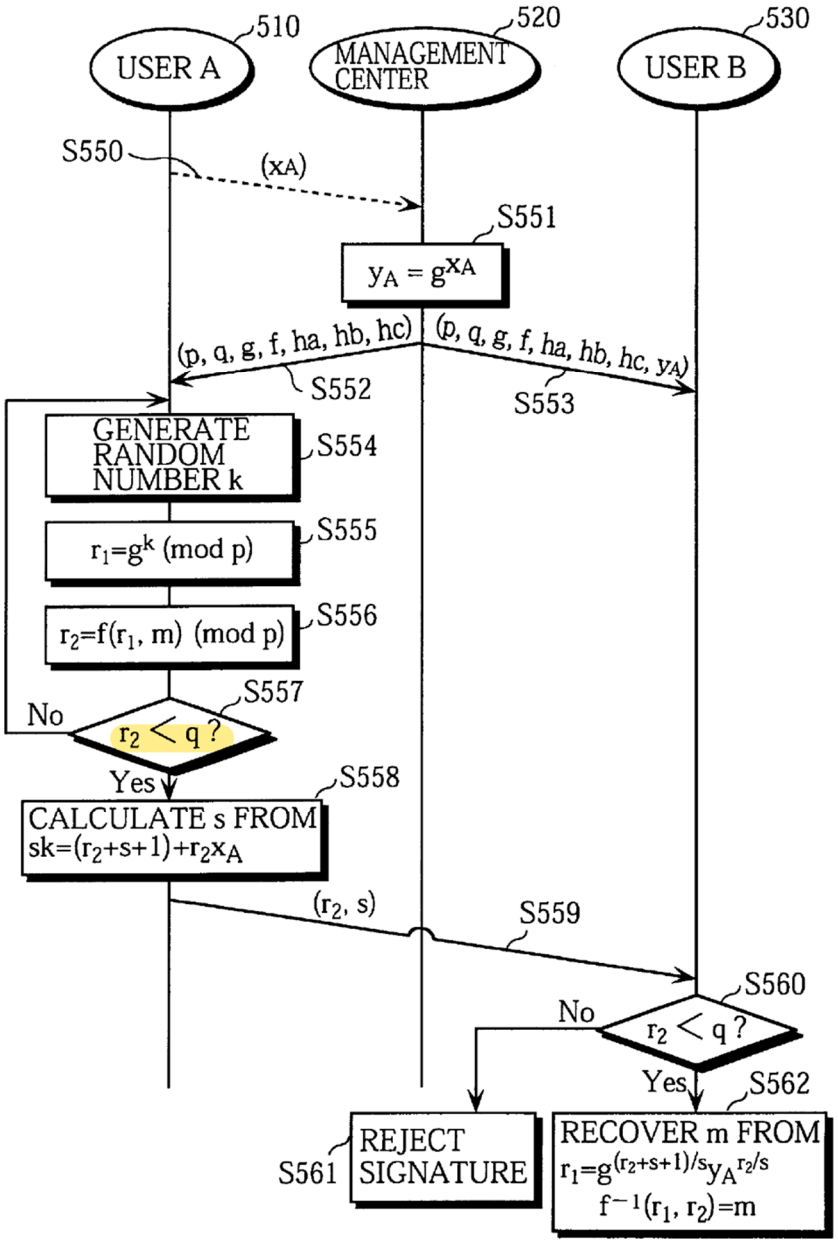
by generating a random number, k, from an RNG, which serves as a seed. Ex. 1005, 7:16-21; Ex. 1002, ¶149.

Next, Miyaji computes a hash of the seed, k, according to the following formula:

$r_2 = H(k) = g^k + m \pmod p$.

Ex. 1005, 8:32-41. Miyaji generates a key k for use in a cryptographic function – that is, Miyaji generates a key (identified as a "masked message"), r_2, and the digital signature. Ex. 1002, ¶¶150-153. Figure 5 reflects that Miyaji evaluates whether r_2, is less than order q, like the '961 patent:

FIG. 5



Ex. 1005, FIG. 5 (highlighting added). A comparison of Figure 5 from Miyaji and Figure 2 from the '961 Patent reflects they both (a) generate a seed value k from RNG, perform a hash function on the seed value k to provide an output hash H(k) stored in the masked message r_2 (or key k); (b) determine whether the output r_2 (or

k) is less than order q prior to reducing mod q; (c) accept the output r_2 (or k) for use as said key k if the value of r_2 (or k) is less than said order q; (d) reject r_2 (or k) if its value is not less than order q; (e) repeat the method if it isn't, and (f) provide the key k for use in performing the cryptographic function if r_2 (or k) is accepted and r_2 (or k) is equal to H(k):

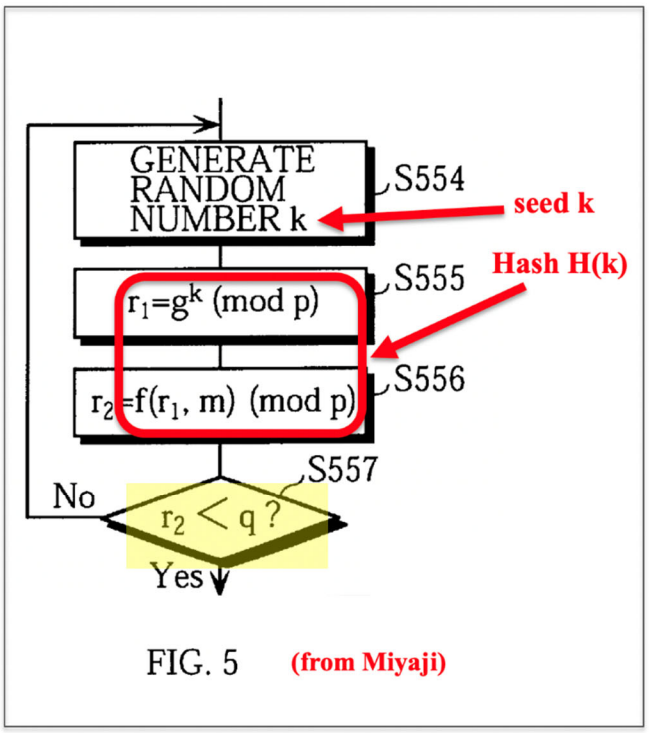


FIG. 5 (from Miyaji)

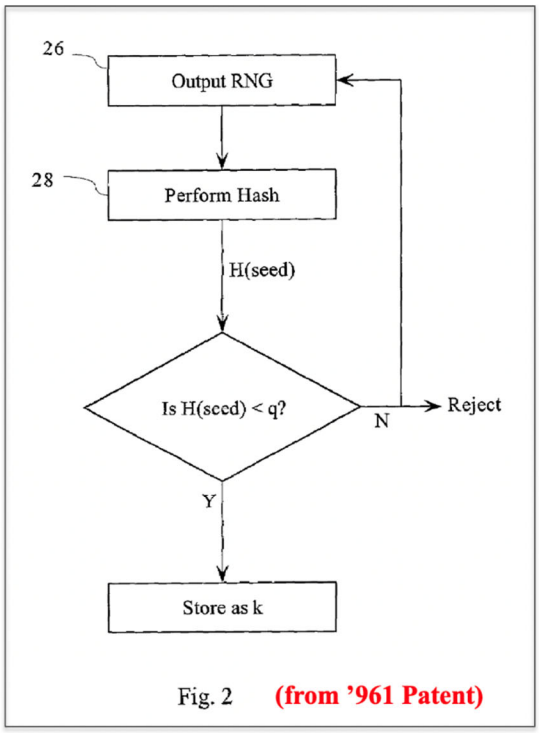


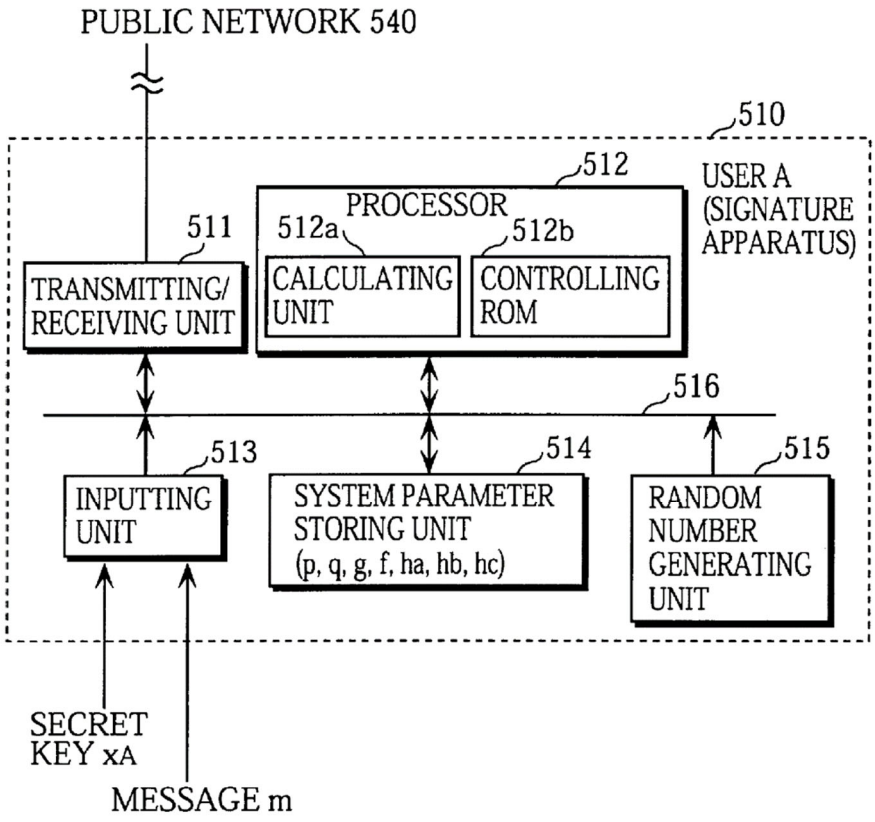
Fig. 2 (from '961 Patent)

Compare Ex. 1005, FIG. 5, *with* Ex. 1001, Fig. 2 (annotations added); Ex. 1002, ¶¶155-158.

Finally, Miyaji discloses the additional elements in claims 15 and 23 of a computer-readable medium or cryptographic unit for executing computer-executable and computer-readable instructions. Ex. 1002, ¶159. Miyaji discloses a processor 512 comprising a "ROM 512b storing a control program unique to the

signature apparatus 510, and performs calculations and transmission control for signatures …, as well as controlling each component of the signature apparatus 510." Ex. 1005, 6:12-16. Figure 2 reflects the processor and memory storing the control program, including the relevant instructions.

FIG. 2

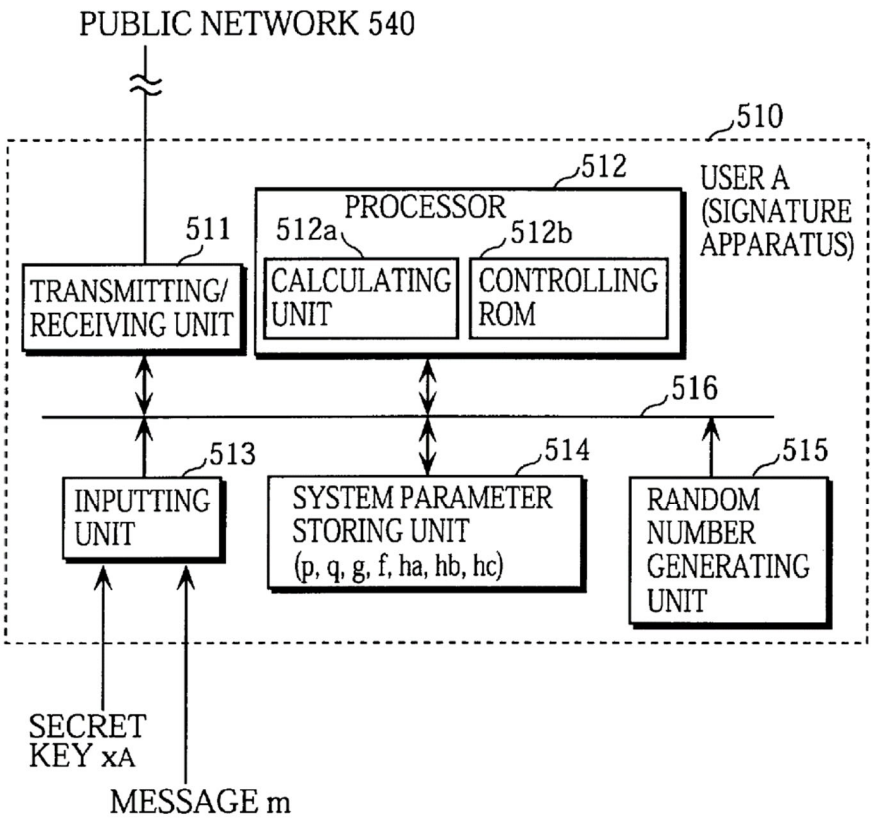


Ex. 1005, FIG. 2; Ex. 1002, ¶159. Further, Miyaji's cryptographic algorithm is performed "in accordance with the program in the controlling ROM 512b." Ex. 1005, 8:11-30. The program controlled by ROM 512b requires both computer-readable and computer-executable instructions. Ex. 1002, ¶159.

2. [23][b]: "an arithmetic processor"

Miyaji discloses an arithmetic processor, as recited in independent claim 23. The processor used to encrypt messages is illustrated in Figure 2 and includes a "calculating unit" 512a:

FIG. 2



Ex. 1005, FIG. 2. The calculating unit 512a performs the "exponentiation modulo p through use of g and the random number k," and is an arithmetic unit. *Id.* 8:26-30; Ex. 1002, ¶¶160-161.

3. 1[b], 15[b], [23][c]: "generating a seed value SV from a random number generator;"

Miyaji discloses "generating a seed value SV from a random number generator." Ex. 1002, ¶¶162-163. Miyaji uses random number k as a seed for subsequent computations, and discloses the following regarding step S554:

> To place an order with the user B 530 for the product, the user A 510 first stores the received system parameters in the system parameter storing unit 514 and **generates a random number k (512 bits long)**. Specifically, in accordance with the program in the controlling ROM 512b, the processor 512 stores the system parameters received via the transmitting/receiving unit 511 in the system parameter storing unit 514, has the **random number generating unit 515 generate the random number k** on GF(p), and acquires the random number k.

Ex. 1005, 8:12-21 (emphasis added). Thus, the value k in Miyaji is disclosed as being generated by a "random number generator." Ex. 1002, ¶163. Further, k is a "seed value," because it is used as an input to a subsequent hash function. *Id.*; *cf.* Ex. 1001, 2:40-44, 3:64-66, 4:6-10, Fig. 5.

4. 1[c], 15[c]. 23[d]: "performing a hash function H( ) on said seed value SV to provide an output H(SV);"

Miyaji discloses "performing a hash function H( ) on said seed value SV to provide an output H(SV)." Ex. 1002, ¶¶164-166. Miyaji discloses performing a

hash function, H(k), on the seed value k to provide an output H(k), which is stored in the masked message r_2. *Id.* ¶164. Miyaji derives the output r_2 from the hash function H(k):

$$r_2 = f(r_1, m)$$

$$= r_1 + m \pmod p$$

$$= g^k \pmod p + m \pmod p$$

$$= g^k + m \pmod p$$

$$= H(k).$$

Ex. 1005, 8:23-35. The function, $f(r_1,m)$, denoted here as "H(k)," is a hash function because it converts the input random value, k, that is 512 bits long, (i.e., a number in the range from 0 to 2^{512}-1), into a number in the range from 0 to p-1 (where p is a prime number 512 bits long). *See, e.g., id.* 7:17-23, 8:12-15; Ex. 1002, ¶165. H(k) is itself the composition of an exponential/discrete-logarithm hash function and a multiplicative hash function, as shown below in Figure 5 of Miyaji.

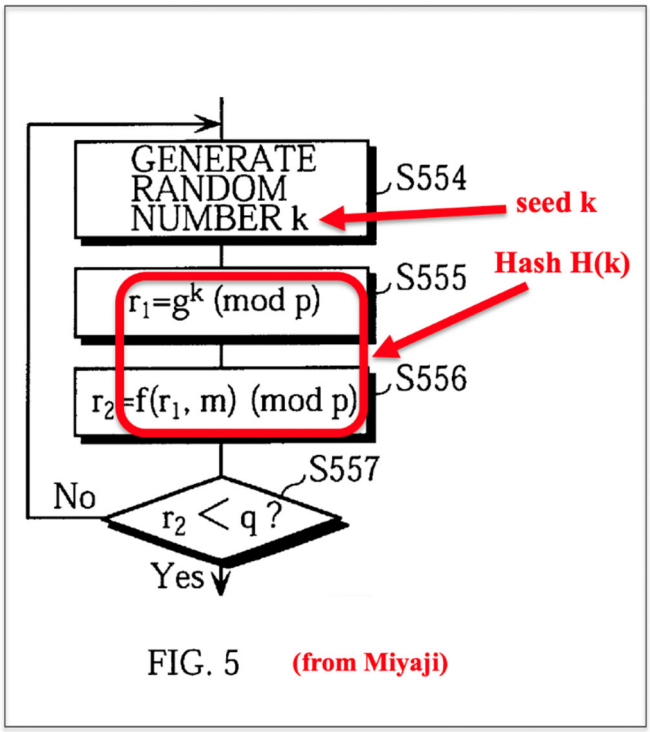


FIG. 5 (from Miyaji)

Ex. 1005, FIG. 5 (annotations added); Ex. 1002, ¶¶73, 165-166. Thus, Miyaji discloses "performing a hash function H( ) on said seed value SV to provide an output H(SV)." *Id.*, ¶166.

5. 1[d], 15[d], 23[e]: "determining whether said output H(SV) is less than said order q prior to reducing mod q;"

Miyaji discloses "determining whether said output H(SV) is less than said order q prior to reducing mod q." Ex. 1002, ¶¶167-169. Miyaji discloses determining whether r_2, which stores the output value, H(k), is less than q, which is the order of the element g in GF(p). *See, e.g.,* Ex. 1005, 7:17-23. In Miyaji's Figure 5, this step occurs at step S557 where the processor 512 "compares r_2 and q according to the program in the controlling ROM 512 *b*," such that "[i]f $q \leqq r_2$, steps

S554-S556 are repeated." *Id.* 8:42-45. Step 557 is illustrated in Figure 5 with the corresponding steps from the '961 patent highlighted:

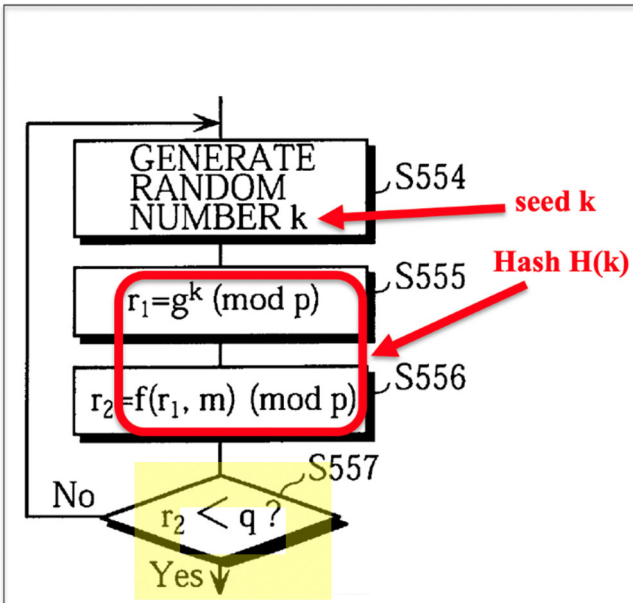


FIG. 5 **(from Miyaji)**

Id. FIG. 5 (annotations added); Ex. 1002, ¶167. Further, step 557 occurs prior to reducing mod q. Ex. 1002, ¶168. For example, step S558, which comes after step S557, solves the following equation for the signature s mod q:

$$sk=(r_2+s+1)+r_2x_A \pmod q \qquad \text{(Equation 2.11)}$$

Ex. 1005, 8:53; *see also id.* 8:54-56. Thus, Miyaji discloses determining whether said output H(SV) is less than said order q prior to reducing mod q. Ex. 1002, ¶169.

6. 1[e], 15[e], 23[f]: "accepting said output H(SV) for use as said key k if the value of said output is less than said order q;"

Miyaji discloses "accepting said output H(SV) for use as said key k if the value of said output is less than said order q." Ex. 1002, ¶¶170-171. If r_2<q, where r_2 stores the output value, H(k), Miyaji accepts the key r_2 and continues to the next step to encrypt the message m using the key r_2 so as to produce the signature, s. *Id.* ¶170. This "accepting" step is disclosed in Miyaji as follows:

> (Step S**558**)
> If, on the other hand, r_2<q, the user A **510** solves s from the signature equation
>
> $$ha(r_2',s,\mathbf{1})k=hb(r_2',s,\mathbf{1})+hc(r_2',s,\mathbf{1})x_A(\text{mod } q) \quad \text{(Equation 2.10)}$$
>
> that is
>
> $$sk=(r_2+s+1)+r_2x_A(\text{mod } q) \quad \text{(Equation 2.11)}$$
>
> More specifically, the processor **512** receives the secret key x_A via the inputting unit **513** and has the calculating unit **512***a* solve s modulo q.

Ex. 1005, 8:46-46. It is further reflected by Figure 5:

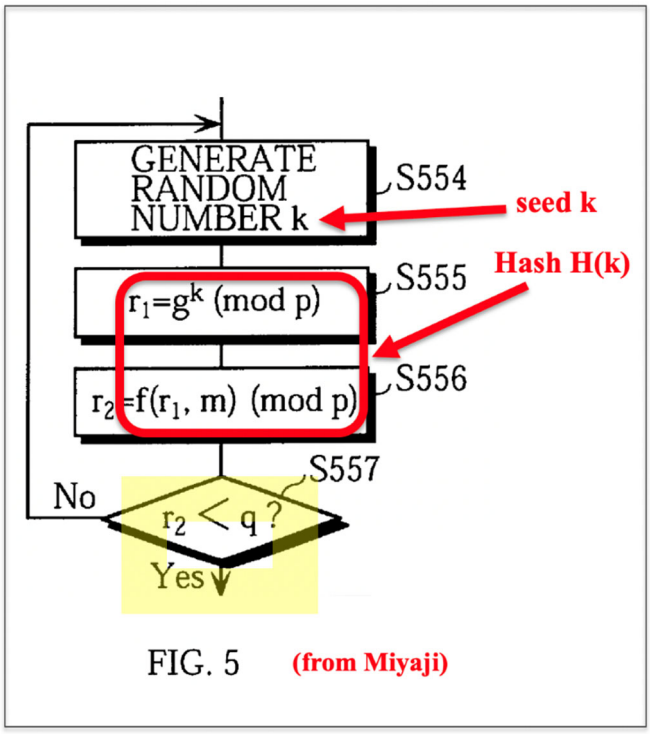


FIG. 5 (from Miyaji)

Id. FIG. 5 (annotations added). Thus, Miyaji discloses accepting r_2, the result of the hash function, for use as a key to encrypt a message if the value is less than the order q. Ex. 1002, ¶171.

7. 1[f], 15[f], 23[g]: "rejecting said output H(SV) as said key if said value is not less than said order q;"

Miyaji discloses "rejecting said output H(SV) as said key if said value is not less than said order q." Ex. 1002, ¶172. If r_2, which stores the output value, H(k), is not less than q, then the value stored in r_2 is rejected, and the steps for computing the key r_2 are repeated, starting from the generation of the random number k. Ex. 1005 8:42-45 (Step 557); Ex. 1002, ¶172. The processor 512 then compares r_2 and q according to the program in the controlling ROM 512b, and "[i]f $q \leq r_2$, steps S554-S556 are repeated." Ex. 1005, 8:43-45. Again, this process is shown Figure

5, which includes a "No" choice, representing when the stored value in r_2 is not less than q:

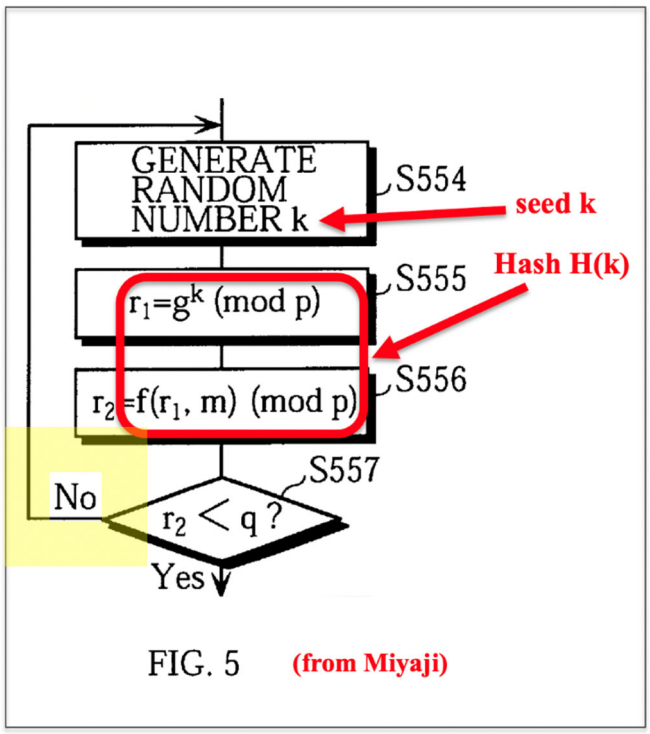


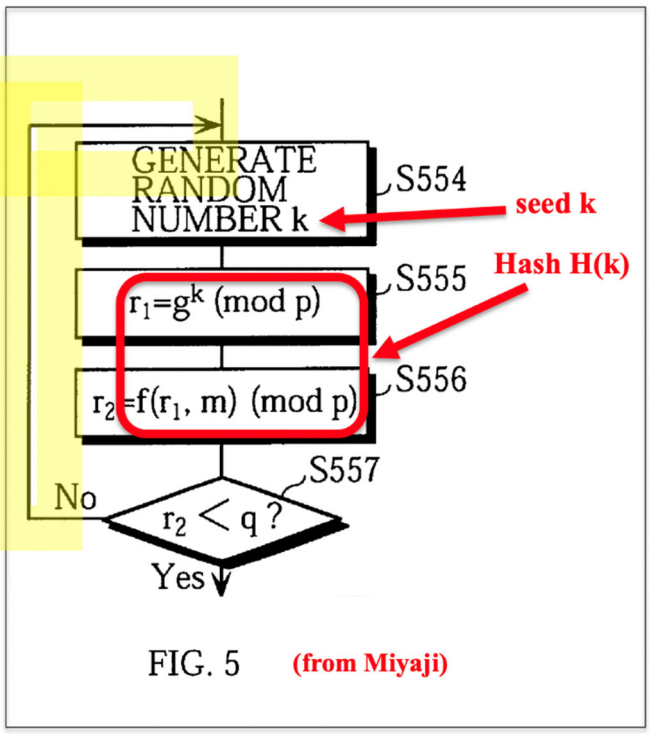
FIG. 5 (from Miyaji)

Id. FIG. 5 (annotations added); Ex. 1002, ¶172.

8. 1[g], 15[g], 23[h], "if said output H(SV) is rejected, repeating said method;"

Miyaji discloses that "if said output H(SV) is rejected, repeating said method." Ex. 1002, ¶173. If r_2, which stores the output value, H(k), is not less than q, then the value stored in r_2 is rejected, and the steps for computing the key r_2 are repeated, starting from the generation of the random number k. Ex. 1005 8:42-45 (Step 557); Ex. 1002, ¶173. The processor 512 then compares r_2 and q according to the program in the controlling ROM 512b, and as Miyaji notes, "[i]f

$q \leq r_2$, **steps S554-S556 are repeated**." Ex. 1005 8:42-45 (emphasis added). The step also is illustrated in Figure 5:



Ex. 1005, Fig. 5 (annotations added); Ex. 1002, ¶173.

9. 1[h], 15[h], 23[i]: "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)."

Miyaji discloses the final element of the independent claims that "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)." Ex. 1002, ¶¶174-175. If the value stored in r_2 is accepted, then the key r_2, which stores the output H(k), is used to produce the signature s, as shown with respect to step S558. *Id.* ¶174. In Miyaji, producing the signature s is a cryptographic function,

because the next step, S559, involves user A sending to user B the *ciphertext* (r_2,s), consisting of the key, r_2, and signature, s. *Id.*; Ex. 1005, 8:46-61. Miyaji thus anticipates claims 1, 15 and 23 of the '961 Patent. Ex. 1002, ¶175.

1. Claims 2, 16, 24

Miyaji discloses the limitation "wherein another seed value is generated by said random number generator if said output is rejected." Ex. 1002, ¶176. In the case where the output r_2, which stores the output, H(k), is rejected, Miyaji repeats the process for computing r_2 (i.e., steps S554-S556) starting from the step (S554) of producing another seed value k according to an RNG:

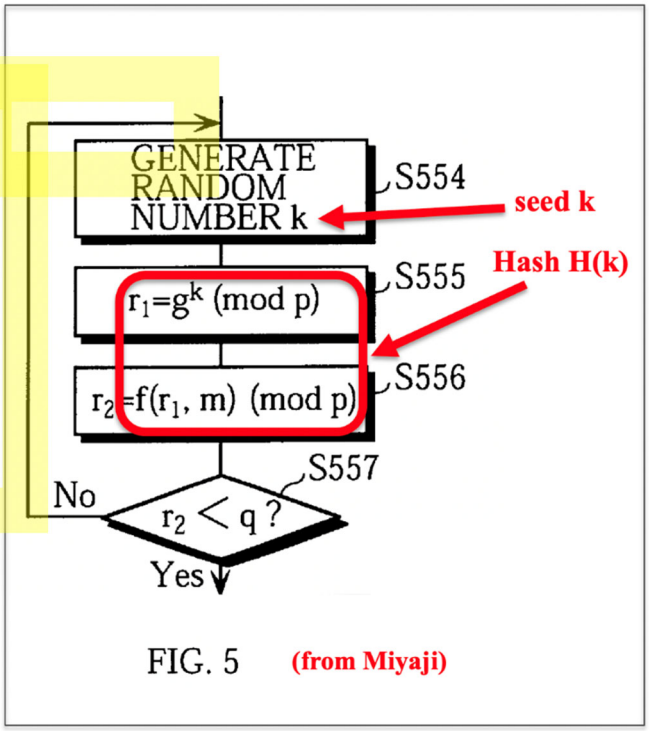


FIG. 5 (from Miyaji)

Ex. 1005, FIG. 5 (annotations added); *see also, e.g., id.*, 8:11-15, 8:42-45; Ex. 1002, ¶176.

2. Claims 3, 17, and 25

Miyaji discloses "wherein the step of accepting said output as a key includes a further step of storing said key." Ex. 1002, ¶177. Miyaji employs a "system storing unit 514" that "is a RAM or the like," which "temporarily stores system parameters downloaded from the management center 520." Ex. 1005, 6:18-20. The system parameters are those "necessary for the message recovery scheme" and include the output H(k). *Id.* 6:20-23; Ex. 1002, ¶177. As noted above with respect to the discussion of elements 1[d], 15[d] and 23[e], Miyaji further discloses storing the output, H(k), in the variable r_2. *See supra* Section VII.A.5. Thus, Miyaji also discloses "wherein the step of accepting said output as a key includes a further step of storing said key." Ex. 1002, ¶177.

3. Claims 4, 18 and 26

Miyaji also discloses "wherein said key is used for generation of a public key" as recited in dependent claims 4, 18 and 26. Ex. 1002, ¶¶178-182. Miyaji stores the output, H(k), in the key, r_2, which is used by user B to decrypt the signature, s. *See* Ex. 1005, 8:62-9:34; Ex. 1002, ¶178. Miyaji describes r_2 as a "masked message" that is generated from the "commitment" r_1 and the message "m." Ex. 1005, 4:23-26, 5:5-6; claim 1. Miyaji also identifies r_2 as being part of the "ciphertext (r_2,s)" that is sent to user B for decryption. *Id.* Abstract, 2:12-13, 2:31-32. A POSA would recognize that the "masked message" or key r_2 is public

(e.g., it is sent to user B from user A), that it has an associated private key, k, and it is used to decrypt the digital signature, s. Ex. 1002, ¶180. For example, the key r_2 is an ephemeral public key used by user B to decrypt a digital signature, s, to get the resulting original plaintext message, m. *Id.*; *see also* Ex. 1001, 1:26-51.

Miyaji also discloses that user B recovers the message, m, using r_2:

(Step S**562**)
If $r_2<q$, the user B **530** finds r_1 by computing

$$r_1 = g^k = g^{(r2+s+1)/s} y_A^{r2/s} \pmod{p} \quad \text{(Equation 2.12)}$$

and recovers the message m according to

$$f^{-1}(r_1,r_2)=m(\text{mod } p) \quad \text{(Equation 2.13)}$$

which can be expanded to

$$m = f^{-1}(r_1, r_2) = r_2 - r_1 = r_2 - g^k = r_2 - g^{(r2+s+1)/s} y_A^{r2/s} \quad \text{(Equation 2.14)}$$

Ex. 1005, 9:4-25 (highlighting added). Thus, r_2 is a public key. Ex. 1002, ¶¶181-182.

B. Ground 2: Miyaji in view of Admitted Knowledge in the Art

Miyaji in view of ordinary skill in the art renders obvious claims 1-6, 15-20, 23-28 of the '961 Patent. *Id.* ¶¶183-212.

1. Claims 1, 15, and 23

As shown above in Section VII.A, Miyaji anticipates claims 1, 15 and 23 of the '961 Patent. Miyaji teaches generating a seed, k (which can be mapped to SV), from an RNG, applying a hash function to get a result, r_2, rejecting r_2 if it is not less than q, and repeating the method until r_2 is less than q, and then using r_2 as an ephemeral key for a signature, s, for a message, m. Ex. 1002, ¶¶185-186. As further reflected by the discussion of why Miyaji anticipates claims 4, 18 and 26 in Section VII.A.3, r_2 is a public key that is sent to user B from user A that has an associated private key, k. Ex. 1002, ¶186.

Miyaji in view of the admitted knowledge in the art thus also renders each of independent claim 1, 15, 23 obvious. *Id.* ¶¶183-188. That is because the '961 Patent acknowledges that a POSA would be aware of the DSA signature standard and the bias in how the (NIST) FIPS 186 standard computes the key k used in DSA, and would readily apply such knowledge to modify r_2 and use it as its own key k. *Id.* ¶¶186-187; Ex. 1001, 1:52-64; 2:32-55. A POSA would find it obvious to use Miyaji to generate the same key k of DSA, e.g., by setting $k=r_2$, where r_2 is an ephemeral public key for a null message, m=0. Ex. 1002, ¶187.

A POSA would have an expectation of success from this use of Miyaji to generate the public key k of DSA, because this combination Miyaji and admitted skill would predictably eliminate the known bias in the generation of DSA's public

key k. *Id.*, ¶188. The expectation of success from this use of Miyaji and ordinary skill in the art would be further supported by the fact that Miyaji uses the well-known rejection sampling technique to generate r_2. *Id.* That is, a POSA would recognize that this method provides a random key, k, for use in a protocol like that of DSA that is uniformly distributed in the range from 1 to q-1. *Id.*

2. Claims 2, 16 and 24

Miyaji in combination with admitted knowledge in the art discloses the limitation "wherein another seed value is generated by said random number generator if said output is rejected. Ex. 1002, ¶¶189-192. Miyaji discloses that in the case when the output r_2, which stores the output, H(k), is rejected, Miyaji repeats the process for computing r_2 (i.e., steps S554-S556) starting from the step (S554) of producing another seed value k according to an RNG:

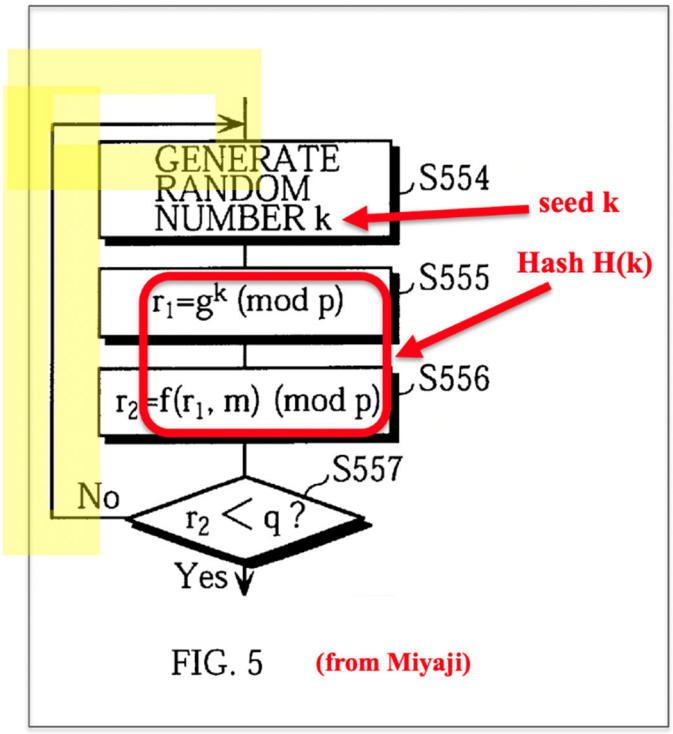


FIG. 5 (from Miyaji)

Ex. 1005, FIG. 5 (annotations added); *see also id.*, 8:11-15 , 8:42-45; Ex. 1002, ¶190. Given this disclosure, a POSA would find it obvious to apply Miyaji to generate a public key k using the admitted knowledge of DSA by setting $k=r_2$, where r_2 is an ephemeral public key for a null message, m=0. Ex. 1002, ¶¶190-191.

A POSA would have an expectation of success from this combination of Miyaji and the admitted knowledge in the art reflected by the DSA key generation method of FIPS 186 described in the '961 patent. *Id.* ¶191; Ex. 1001, 1:52-64, 2:32-55. A POSA would have the expectation of success of eliminating the bias in the generation of DSA's key k, since the method of Miyaji uses the well-known rejection sampling technique to generate r_2. *Id.* ¶191. That is, a POSA would

recognize that the combination of Miyaji and admitted knowledge in the art provides a random key, k, for use in a context like DSA that is uniformly distributed in the range from 1 to q-1. *Id.*

3. Claims 3, 17 and 25

Miyaji in view of admitted knowledge in the art renders obvious claims 3, 17 and 25. Ex. 1002, ¶¶193-195. As noted in Section VII.A.5, explaining why Miyaji discloses elements 1[d], 15[d] and 23[e], Miyaji discloses storing the output, H(k), in the variable r_2, satisfying the requirement that "the step of accepting said output as a key includes a further step of storing said key." *Id.* ¶193.

A POSA also would find it obvious to use the method disclosed in Miyaji to generate a key k like that used in DSA by setting $k=r_2$, where r_2 is an ephemeral public key for a null message, m=0. *Id.* ¶194. A POSA would have an expectation of success from combining Miyaji with the admitted knowledge of the DSA key generation method from FIPS 186. *Id.* A POSA would expect success from the combination by eliminating the bias in the generation of DSA's key k, since the method of Miyaji uses the well-known rejection sampling technique to generate r_2, thereby eliminating the bias in the way that the FIPS 186-1 and 186-2 documents describe. *Id.* A POSA would recognize that this combination of Miyaji and

admitted knowledge in the art provides a random key, k, for use in a protocol like DSA that is uniformly distributed in the range from 1 to q-1. *Id.*

4. Claims 4, 18 and 26

Miyaji and admitted knowledge in the art renders obvious the requirement of claims 4, 18 and 26 "wherein said key is used for generation of a public key." Ex. 1002, ¶¶196-202. A POSA would recognize that the masked message/key r_2 is public and is sent to user B from user A. *Id.* ¶197. A POSA also would know that that r_2 has an associated private key, k, and that it is used to decrypt the digital signature, s. *Id.* The key r_2 is an ephemeral public key used by user B to decrypt a digital signature, s, producing the resulting original plaintext message, m. *Id.* ¶198. A POSA would recognize these are all properties of a public key. *Id.*; *see also* Ex. 1001, 1:26-51.

Miyaji also discloses that user B recovers the message, m, using r_2 as follows:

(Step S562)
If r_2<q, the user B **530** finds r_1 by computing

$$r_1 = g^k = g^{(r2+s+1)/s} y_A^{r2/s} \pmod{p} \quad \text{(Equation 2.12)}$$

and recovers the message m according to

$$f^{-1}(r_1,r_2)=m(\text{mod } p) \quad \text{(Equation 2.13)}$$

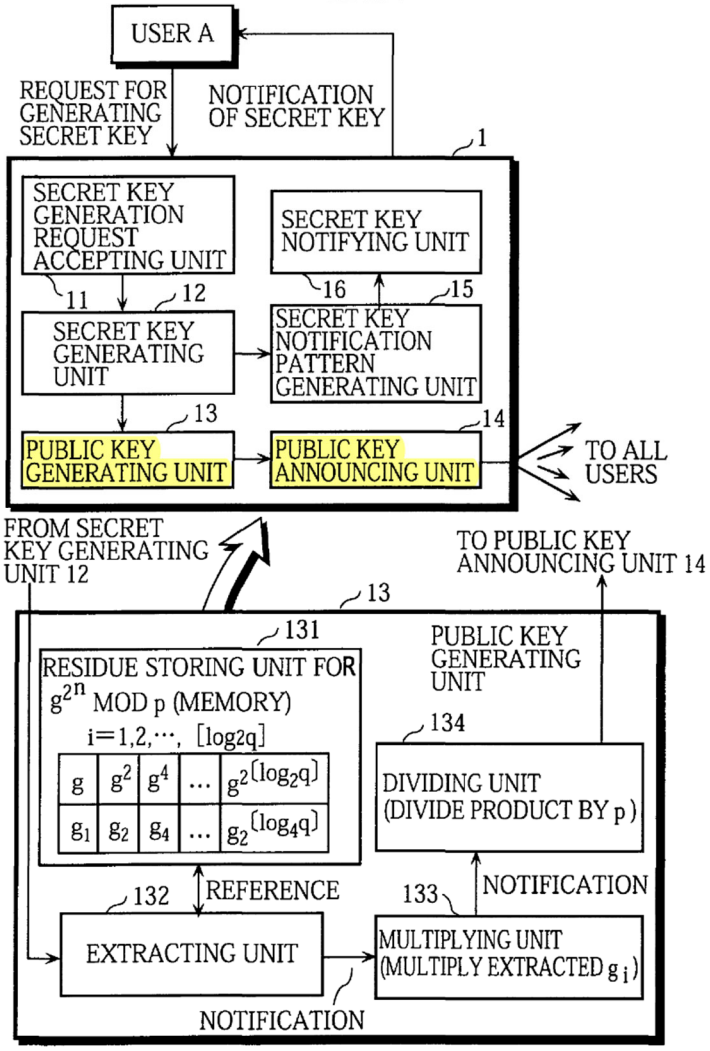
which can be expanded to

$$m = f^{-1}(r_1, r_2) = r_2 - r_1 = r_2 - g^k = r_2 - g^{(r2+s+1)/s} y_A^{r2/s} \quad \text{(Equation 2.14)}$$

Ex. 1005, 9:4-25 (highlighting added). Thus, a POSA would understand r_2 to be a public key. Ex. 1002, ¶¶197-198.

It also would be obvious to use the method of Miyaji with the admitted knowledge to generate a (long-term) public key. *Id.* ¶¶199-201. Miyaji discloses a public key generating unit and a public key announcing unit, as reflected by Figure 7:

FIG. 7



Ex. 1005, FIG. 7 (highlighting added). The '961 Patent admits that ordinary skill includes the knowledge that in the ECDSA elliptic curve digital signature algorithm, a public key kP is generated from a random number key in the range from 0 to n-1. Ex. 1001, 2:2-11. Miyaji also discloses elliptic-curve public keys of this form. Ex. 1005, 11:45-58, 12:20-29.

It thus would be obvious to a POSA to use the method of Miyaji for generating a random value k in a given range 0 to n-1, by setting $k=r_2$ for a null

message, m=0, and to then apply admitted knowledge in the art and use this value k to generate the public key kP in ECDSA. Ex. 1002, ¶200. A POSA would have an expectation of success from generating a random value that is uniform in a given interval, and would apply this admitted knowledge to take advantage of the public-key generation method of ECDSA. *Id.*; *see also* Ex. 1005, 7:15-8:61. Further, this expectation of success from combining the admitted knowledge of generating a random key in ECDSA with Miyaji's masked message key r_2 to perform the same function produces better uniformity in the result. Ex. 1002, ¶200. Hence, there would be an expectation of success that using Miyaji in combination with EDCSA's technique for generating a key via ordinary knowledge eliminates any bias in generating the random key used in ECDSA. *Id.*

5. Claims 5, 19 and 27

Miyaji in view of admitted knowledge in the art discloses the limitation "said order q is a prime number represented by a bit string of predetermined length L." Ex. 1002, ¶¶202-207.

Miyaji discloses a method for a group with order q. *Id.* ¶203; Ex. 1005, 2:28-29. As a result, it also would be obvious to a POSA that the order of this group q can be a prime number. Ex. 1002, ¶204. Such ordinary knowledge is exemplified by the POSA's knowledge of the mathematics behind DSA, including Cauchy's theorem. *Id.*; Ex. 1001, 1:52-2:14 (describing known art and explaining

signature generation in DSA). For instance, a POSA would know that DSA requires an element of order q where q is a prime number, which supports its security. *Id.* ¶¶70, 204-205. Cauchy's theorem guarantees success in finding such an element. *Id.* ¶204. Hence, a POSA would have an expectation of success to make q be a prime number. *Id.* ¶206. For example, a POSA would find it obvious to use the method of Miyaji to generate a key like the k of DSA, which is required to be less than a number, q, which is prime. *Id.* ¶¶204, 206.

6. Claims 6, 20 and 28

Miyaji in view of admitted knowledge in the art discloses the limitation the "output from said hash function is a bit string of predetermined length L." Ex. 1002, ¶¶208-210. The hash function, $f(r_1,m)$, denoted as "H(k)" in the comparison of Miyaji to the '961 patent in Section VII.A.5, produces a bit string that is a predetermined length, or 512 bits long. *Id.* ¶209. The 512-bit output is produced from the seed "k" and stored in r_2 and is computed as mod p, where p is a bit string that is 512 bits long. *Id.*; Ex. 1005, 7:17-23; 8:11-41. Thus, for the same reasons a POSA would appreciate that the combination of Miyaji and admitted knowledge produces an order q that is a prime number represented by a bit string of predetermined length L as required by claims 5, 19 and 27, a POSA also would understand that this output p and r_2 are represented by a bit string of predetermined length L, where L is 512 bits. Ex. 1002, ¶¶209-210. A POSA would have an

expectation of success from combining Miyaji with the admitted ordinary skill in the art reflected by the '961 patent, because it provides security by using a large prime number. *Id.* ¶210.

7. Motivation to Combine

A POSA would be motivated combine the teachings of Miyaji with the ordinary skill of a POSA as evidenced by the art known and described in the '961 patent. Ex. 1002, ¶¶211-212. A POSA would have been motivated to combine Miyaji's rejections sampling technique employing the public key k of DSS or the public key kP in ECDSA disclosed in the '961 patent, since a POSA would have a reasonable expectation of success in doing so. *Id*. A POSA would recognize this combination of Miyaji and ordinary skill offers predictable results. *Id.* ¶211.

For instance, a POSA would recognize that it would be beneficial to employ the public key k from DSS described in the '961 patent with Miyaji's rejection sampling technique using the masked message r_2 from Miyaji (which already is a public key), because doing so reduces bias in the selection of the key. *Id.* ¶212*; cf. In re Nilssen,* 851 F.2d 1401, 1403 (Fed. Cir. 1988) (a known technique of using a cutoff switch would have predictably resulted in protecting the inverter circuit, such that it would have been within the skill of the ordinary artisan to use a cutoff switch in response to the actuation signal to protect the inverter). To a POSA, using the rejection sampling technique and r_2 of Miyagi and the known techniques

and generation of a public key from DSS would simply require substitution of one known element (generating of a public key using the techniques described as prior art in the '961 patent) for another (the rejection sampling technique that reduces bias described in Miyaji). Ex. 1002, ¶212; *see also* MPEP § 2143(I); *Fout,* 675 F.2d 297, 301 (CCPA 1982) ("[b]ecause both Pagliaro and Waterman teach a method for separating caffeine from oil, it would have been *prima facie* obvious to substitute one method for the other"; an "express suggestion to substitute one equivalent for another need not be present to render such substitution obvious"). This combination would have additionally constituted the use of a known technique (generation of a public key k or kP from DSS and/or ECDSA) to improve an algorithm directed to reducing bias, such that the known technique would function in the same way as before, with predictable results. Ex. 1002, ¶212.

C. Ground 3: Bellare in View of LEDA

Claims 1, 15 and 23 of the '961 Patent are rendered obvious by Bellare in view of LEDA. Ex. 1002, ¶¶213-270.

1. Claims 1[a], 15[a], and 23[a]

1[a]: 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:

15[a]: 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:

23[a]: 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:

Bellare in view of LEDA discloses "generating a key k for use in a cryptographic function performed over a group of order q." Bellare discloses a method of generating a key for use in a cryptographic function. Ex. 1002, ¶216. Bellare generates a key k for use in the DSA cryptographic function, and the function is performed over the group $\mathbf{Z}_{q.}$ *Id.* This group has order q, as reflected by Bellare's use of the "scheme" of the DSA algorithm. Ex. 1006, 281. This scheme uses the prime number p, prime number q which divides p-1, and element $g \in \mathbf{Z}^*_p$ of order q (chosen a $g = h^{(p-1)/q}$ where h is a generator of the cyclic group $\mathbf{Z}^*_p$). *Id.* Bellare reflects that a subgroup G is generated by g, where g "has prime order," and its exponents are from a field, namely $\mathbf{Z}_{q.}$ *Id.* Using this scheme, "[t]he signer generates a random number $k \in \{1, \ldots, q-1\}$," which Bellare calls the "nonce." *Id.* Because $\mathbf{Z}_q$ is a group, Bellare discloses that the exponentiation operations in DSA, as well as operations in $\mathbf{Z}_q$, are performed over a group of order q. *Id.*, 281-282; Ex. 1002, ¶¶217-218.

Further, Bellare in view of LEDA disclose the requirements in the preambles to claims 15 and 23 of "a computer-readable medium comprising instructions" and

a "cryptographic unit having access to computer readable instructions." Ex. 1002, ¶219. LEDA employs C++ source code, and it would be obvious to a POSA to implement "the scheme" of Bellare and the C++ source code disclosed in LEDA by storing computer executable instructions in a computer readable medium or a cryptographic unit for performing the instructions, and then executing the instructions. *Id.*; Ex. 1007, 281; Ex. 1007, 93.

2. 23[b]: "an arithmetic processor"

Bellare in view of LEDA discloses an arithmetic processor, as required by independent claim 23. Ex. 1002, ¶220. Bellare discloses that to perform "the scheme," it must compute an exponentiation mod p and a reduction mod q, and further compute a multiplicative inverse in $\mathbf{Z}^*_q$. *Id.* A POSA would understand that each of these operations requires an arithmetic processor. *Id.*

3. 1[b], 15[b], 23[c]: "generating a seed value SV from a random number generator"

Bellare in view of LEDA discloses "generating a seed value SV from a random number generator." Ex. 1002, ¶¶221-226. Bellare discloses that the random nonce k in "the scheme" for DSA is generated when the "signer generates a random number k [in] {1, …, q-1}". Ex. 1006, 281. Bellare discloses that such a nonce k is generated by a pseudo-random number generator that inputs a seed σ:

2.2 Pseudo-Random Number Generators

Each time DSA is used to digitally sign a message m, a nonce k is needed. Ideally k should be a truly random number. In practice the nonces k are pseudo-random numbers produced by a pseudo-random number generator.

A pseudo-random number generator is a program $\mathcal{G}$ that on input a *seed* σ, generates a seemingly random sequence of numbers $\mathcal{G}(\sigma) = k_1, k_2, \ldots$.

Id., 282 (highlighting added); Ex. 1002, ¶¶222-223.

Bellare also teaches an RNG that Bellare calls a "truncated linear congruential generator" that takes an initial seed $\sigma_0 \in \mathbf{Z}_m$ and produces a sequence of pseudo-random numbers in the range $\{0, \ldots, 2^{h-l}-1\}$. Ex. 1006, 282. Bellare computes a sequence σ_i according to a linear recurrence employing modulo M. *Id.*; Ex. 1002, ¶223. A POSA would understand that each value σ_1, σ_2, …, is a ***seed*** generated by a random number generator, because each value is produced by the truncated linear congruential generator and is used as seed input to the next iteration in the truncated linear congruential generator. *Id.* In Bellare, each seed is identified as σ_i, for i=0,1,2,…, and Bellare says that σ denotes a "seed." Ex. 1006, 282. This same analysis applies to other RNGs that use hash functions, including Bellare's teachings on the use of the SHA-1 hash function. Ex. 1006, 278, 282; Ex. 1002, ¶223.

LEDA also discloses an RNG that applies a hash function to a seed X_0 so that each subsequent value is a seed value. Ex. 1007, 92; Ex. 1002, ¶224. LEDA teaches discarding the first 320 such seeds; hence, each random number that LEDA

outputs is produced as a hash function applied to a seed. *Id.* ¶224. Each value of the RNG that is output is a seed (for the next iteration) that is generated by an RNG that produces what LEDA calls "random sources." Ex. 1007, 79, 92; Ex. 1002, ¶224. LEDA teaches using one hash function to generate the first 31 values, then using a different hash function to generate the additional values after that:

Implementation of random_source: Our implementation of random sources follows the description in [Knu81, Vol2, section 3.2.2]. We first give the mathematics and then the program. Internally, we always generate a sequence of integers in the range $[0\,..\,2^{31}-1]$. We define 32 unsigned longs $X_0, X_1, \ldots, X_{31}$ by

$$X_0 = seed$$

and

$$X_i = (1103515245 \cdot X_{i-1} + 12345) \bmod m$$

for $1 \leq i \leq 31$. Here $m = 2^{32}$. We extend this sequence by

$$X_i = (X_{i-3} + X_{i-32}) \bmod m$$

for $i \geq 32$. In this way an infinite sequence $X_0, X_1, \ldots$ of unsigned longs is obtained. Following [Knu81, Vol2, section 3.2.2], we discard the first 320 elements of this sequence (they are considered as a warm-up phase of the generator) and we also drop the right-most bit of each number (since it is the least random). Thus, the i-th number output by the internal generator is

```
(X[i + 320] >> 1) & 0x7fffffff.
```

Hash functions

LEDA, 92 (highlighting added); Ex. 1002, ¶225. Based on these teachings, a POSA would find it obvious to combine Bellare and LEDA to take advantage of the mutually beneficial schemes each reference employs. *Id.* Both schemes generate a seed value from an RNG in essentially the identical fashion, although each method is independently useful. *Id.* Combining Bellare with LEDA thus

leads to an expectation of success that the combination can produce seed values from either a truncated linear congruential generator like that in LEDA or a SHA-1 RNG like that in Bellare, which advantageously doubles the user's options for generating the seed value. *Id.*

4. 1[c], 15[c], 23[d]: "performing a hash function H( ) on said seed value SV to provide an output H(SV)"

Bellare in view of LEDA discloses "performing a hash function H( ) on said seed value SV to provide an output H(SV)." Ex. 1002, ¶¶227-229. Bellare discloses that a truncated linear congruential generator can be used as an RNG. *Id.* ¶227. Each value σ_{i+1} is computed by performing a hash function on a seed value, σ_i, consisting of the previous value generated as a part of a truncated linear congruential generator. *Id.*; Ex. 1006, 282. The hash function is a multiplicative hash function, $\sigma_{i+1} = a*\sigma_i + b \pmod M$, where a, b, and M are parameters of the RNG. Ex. 1002, ¶227. Bellare thus discloses applying a hash function, H(x) = a*x+ b (mod M), to said seed value. *Id.* Bellare also teaches the use of "the use of a pseudo-random generator based on SHA-1 or DES." Ex. 1006, 278. A POSA would understand that SHA-1 is a hash function, and that Bellare discloses it should be applied as hash function in the RNG for generating the nonce k. *Id.* ¶227.

Likewise, LEDA also teaches the use of a hash function to generate random numbers, where each hash function is applied to a seed (after an initial seed X_0) that is generated by an RNG:

Implementation of random_source: Our implementation of random sources follows the description in [Knu81, Vol2, section 3.2.2]. We first give the mathematics and then the program. Internally, we always generate a sequence of integers in the range $[0\,..\,2^{31}-1]$. We define 32 unsigned longs $X_0, X_1, \ldots, X_{31}$ by

$$X_0 = seed$$

Hash functions

and

$$X_i = (1103515245 \cdot X_{i-1} + 12345) \bmod m$$

for $1 \leq i \leq 31$. Here $m = 2^{32}$. We extend this sequence by

$$X_i = (X_{i-3} + X_{i-32}) \bmod m$$

for $i \geq 32$. In this way an infinite sequence $X_0, X_1, \ldots$ of unsigned longs is obtained. Following [Knu81, Vol2, section 3.2.2], we discard the first 320 elements of this sequence (they are considered as a warm-up phase of the generator) and we also drop the right-most bit of each number (since it is the least random). Thus, the i-th number output by the internal generator is

```
(X[i + 320] >> 1) & 0x7fffffff.
```

Id. ¶228; Ex. 1007, 92 (highlighting added). Combining Bellare with LEDA thus leads to an expectation of success that the combination can produce seed values from either a hash-based random number generator like that in LEDA or a SHA-1 RNG like that in Bellare, which advantageously doubles the user's options for generating the seed value. Ex. 1002, ¶ 228.

5. 1[d], 15[d], 23[e]: "determining whether said output H(SV) is less than said order q prior to reducing mod q;"

Bellare in view of LEDA discloses "determining whether said output H(SV) is less than said order q prior to reducing mod q." Ex. 1002, ¶¶230-238. Bellare discloses that in "the scheme," the signer generates a random number k in the range from 1 to q-1, called a *nonce*. Ex. 1006, 281. In addition, Bellare teaches the use of possible RNGs based on hash functions applied to a seed value (generated from an RNG in a previous iteration) to generate the key k. Ex. 1002, ¶¶230-231.

Bellare discloses that the nonce k used in "the scheme" for DSA needs to be as random as possible in order for "the scheme" to be secure. Ex. 1006, 278. Thus, a POSA would look to LEDA for methods that improve the randomness and uniformity of the nonce k. Ex. 1002, ¶232. LEDA discloses that to get a random value in the range from low to high, simply reducing a random value modulo high-low+1 will produce a biased result. Ex. 1007, 92; Ex. 1002, ¶233. LEDA instead applies rejection sampling to a value, x, produced by a hash function, internal_source(), which is based on using a seed generated by an RNG of type random_source. Ex. 107, 92; Ex. 1002, ¶234.

LEDA discloses a random source based on a hash function that takes as an input seeds the output random values from previous iteration, to generate random numbers:

Implementation of random_source: Our implementation of random sources follows the description in [Knu81, Vol2, section 3.2.2]. We first give the mathematics and then the program. Internally, we always generate a sequence of integers in the range $[0 .. 2^{31} - 1]$. We define 32 unsigned longs $X_0, X_1, \ldots, X_{31}$ by

$$X_0 = seed$$

and

$$X_i = (1103515245 \cdot X_{i-1} + 12345) \bmod m$$

for $1 \leq i \leq 31$. Here $m = 2^{32}$. We extend this sequence by

$$X_i = (X_{i-3} + X_{i-32}) \bmod m$$

for $i \geq 32$. In this way an infinite sequence $X_0, X_1, \ldots$ of unsigned longs is obtained. Following [Knu81, Vol2, section 3.2.2], we discard the first 320 elements of this sequence (they are considered as a warm-up phase of the generator) and we also drop the right-most bit of each number (since it is the least random). Thus, the i-th number output by the internal generator is

```
(X[i + 320] >> 1) & 0x7fffffff.
```

Hash functions

Ex. 1007, 92 (highlighting added); Ex. 1002, ¶234. LEDA uses this hash value to generate a random number uniformly distributed in a given range, [low.high]. Ex. 1002, ¶235. LEDA computes a difference, diff=high-low, which in the case of the range [0,q-1] implies that diff=q-1. *Id.* If x is greater than diff (that is, greater than or equal to q in this case), LEDA repeats the hash-function process (in a "while" loop). *Id.* That is, rather than reducing mod q, which LEDA teaches would be biased, LEDA teaches rejecting a value of x that is too large and repeating the hash function computation. *Id.* The comparison "x > diff" discloses "determining whether said output H(SV) is less than said order q prior to reducing mod q." *Id.*; Ex. 1007, 93 (highlighting added). The "& pat" expression referenced in LEDA is

a bit-wise "AND" of the value returned by the hash function "internal_source()," which is LEDA's implementation of "random_source," and is a hash function. Ex. 1002, ¶236; Ex. 1007, 93.

A POSA would expect success from substituting this method from LEDA (which applies rejection sampling with a hash value applied to a seed produced by a random number generator) with that of Bellare (which uses a hash value applied to a seed produced by a random number generator and reducing it modulo q). Ex. 1002, ¶237. There would be an expectation of success from the combination of reducing mod q with using a "while" loop based on rejection sampling that terminates when x is not greater than diff=q-1 (that is, it terminates when x is less than q). *Id.* Since LEDA teaches that reducing modulo q (which is equal to high-low+1 in this case) is "not uniformly distributed," a POSA would have an expectation of success that nonce k would be uniformly distributed in the range [0,q-1], instead of being biased. *Id.*; Ex. 1007, 92. A POSA would understand that this result advantageously applies where the hash function used in LEDA is replaced with either the truncated linear congruential generator disclosed in Bellare (Ex. 1006, 282), or with SHA-1, as Bellare recommends. *Id.*, 278; Ex. 1002, ¶237.

6. 1[e], 15[e], 23[g]: "accepting said output H(SV) for use as said key k if the value of said output is less than said order q;"

Bellare in view of LEDA discloses "accepting said output H(SV) for use as said key k if the value of said output is less than said order q." Ex. 1002, ¶¶239-240. Bellare in light of LEDA compares the output of the hash function (applied to the seed) to q in a "while" loop that terminates when the value x is less than q (which is the order of the group disclosed in Bellare). *Id.* ¶239. LEDA then returns low+x, which is simply x in the case when low=0, i.e., for the interval [0,q-1]. *Id.*; 1007, 93. Because the "while" loop terminates when the value x is less than q, the output is accepted for use as a key when this condition is met. Ex. 1002, ¶239. Further, this random-number generation method applies equally to the embodiment of Bellare and LEDA using the truncated linear congruential generator of Bellare, or with the SHA-1 cryptographic hash function recommended by Bellare, leading a POSA to expect success from the combination. *Id.*

7. 1[f], 15[f], 23[g]: "rejecting said output H(SV) as said key if said value is not less than said order q;"

Bellare in view of LEDA discloses "rejecting said output H(SV) as said key if said value is not less than said order q." Ex. 1002, ¶¶241-242. Bellare in view of LEDA teaches comparing the output of the hash function (applied to the seed) to q in a "while" loop that terminates when the value x is less than q. When the value x is greater than q, the loop continues. *Id.* ¶241. Thus, the hash value returned by

"internal_source()" is rejected if the value x is greater than diff, i.e., when x is not less than q (since diff=q-1 when applying the teachings of LEDA for the interval [0,q-1]). *Id.* This random-number generation method using a comparison with q advantageously applies equally to the embodiment of Bellare-LEDA using the truncated linear congruential generator of Bellare or with the SHA-1 cryptographic hash function recommended by Bellare. *Id.*

8. 1[g], 15[g], 23[h]: "if said output H(SV) is rejected, repeating said method;"

Bellare in view of LEDA discloses "if said output H(SV) is rejected, repeating said method." Ex. 1002, ¶¶243-244. Bellare in light of LEDA compares the output of the hash function (applied to the seed) to q in a "while" loop that terminates when the value x is less than q. *Id.* ¶243. The loop continues if the value of x is greater than q. *Id.* Thus, the hash value returned by "internal_source()" is rejected if the value x is greater than diff, i.e., when x is not less than q (since diff=q-1 when applying the teachings of LEDA for the interval [0,q-1]), and the method is repeated, calling internal_source() again, if the output is rejected. *Id.* Further, this random-number generation method using rejection sampling applies equally to the embodiment of Bellare and LEDA using the truncated linear congruential generator of Bellare or with the SHA-1 cryptographic hash function recommended by Bellare, providing an expectation of success to a POSA from the combination. *Id.*

9. 1[h], 15[h], 23[i]: "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)."

Bellare in view of LEDA discloses "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)." Ex. 1002, ¶¶245-247. Bellare in view of LEDA teaches comparing the output of the hash function (applied to the seed) to q in a "while" loop that terminates when the value x is less than q (i.e. the order of the group disclosed in Bellare). *Id.* ¶245. Further, the value low+x (which is equal to x when low=0) is returned by the method of LEDA when the while-loop terminates, i.e., when the hash value returned by "internal_source()" is accepted. Ex. 1007, 93. Since this value would then be used in "the scheme" by Bellare, a POSA would expect success from substituting the method of LEDA for the reduction modulo q taught in Bellare, as the returned value, x, would then be used as the key k in "the scheme" of Bellare. Ex. 1002, ¶245. This random-number generation method applies equally to the embodiment of Bellare-LEDA using the truncated linear congruential generator of Bellare, or with the SHA-1 cryptographic hash function recommended by Bellare, providing additional expectation of success from the combination. *Id.*

2. Claims 2, 16, and 24

Bellare in view of LEDA discloses "wherein another seed value is generated by said random number generator if said output is rejected." Ex. 1002, ¶¶248-249. In the "while" loop disclosed in LEDA, the random number generator, "internal_source()", is called again when the previous value is rejected. Ex. 1007, 93. According to the truncated linear congruential generator taught in Bellare, the hash function taught in LEDA, as well as using SHA-1 as taught in Bellare, the next call to "internal_source()" is based on using the previous generated value now used as a seed. Ex. 1002, ¶¶249; Ex. 1006, 278, 282; Ex. 1007, 92. That is, the next seed to "internal_source()" is generated by the random number generator if the output is rejected. Ex. 1002, ¶249.

3. Claims 3, 17, and 25

Bellare in view of LEDA discloses "wherein the step of accepting said output as a key includes a further step of storing said key." Ex. 1002, ¶¶250-251. LEDA teaches storing the value returned from "internal_source()" in a variable, x, and returning this value (when low=0, as in the case for an interval, [0,q-1]), to the calling function, which in this combination is "the scheme" of Bellare. *Id.*, ¶251; Ex. 1006, 281; Ex. 1007, 93. This returned value is then stored in the variable k according to "the scheme" of Bellare. Ex. 1002, ¶251; Ex. 1006, 281.

4. Claims 4, 18, and 26

Bellare in view of LEDA discloses "wherein said key is used for generation of a public key." Ex. 1002, ¶¶252-257. Bellare discloses that the key is used for the generation of a public key. *Id.* ¶¶253-254. Bellare discloses that in "the scheme" of DSA for signing a sequence of message, m_i, each key, r_i, which is used by user B to decrypt the signature, s_i, that "the secrecy of the nonces is crucial" and "the cryptanalyst only sees $r_i = (g^{ki} \bmod p) \bmod q$ from which he cannot recover k_i short of computing discrete logarithms…." Ex. 1006, 278, 279; Ex. 1002, ¶253. As a result, the "cryptanalyst" can see each key r_i. Ex. 1002, ¶254. Thus, a POSA would recognize that each key r_i is a (short-term, ephemeral) public key (e.g., it is sent to the signature verifier to decrypt a digital signature), and that it has an associated private key, k_i; hence, a POSA would understand each r_i to be a public key. *Id.*

Alternatively, Bellare teaches that the public key, y, used in DSS is generated from a random secret key, x, which is in $\mathbf{Z}_q$, i.e., x is a secret key chosen randomly in the range from 1 to q-1. *Id.* ¶255 Ex. 1006, 278. Thus, a POSA would find it obvious to use the same method to generate x as would be used to generate the random nonce, k, disclosed in "the scheme" of DSA. Ex. 1002, ¶256.

5. Claims 5, 19 and 27

Bellare in view of LEDA discloses "wherein said order q is a prime number represented by a bit string of predetermined length L." Ex. 1002, ¶¶258-259. LEDA teaches storing the value returned from "internal_source()" in a variable, x, and returning this value (when low=0, as in the case for an interval, [0,q-1]), to the calling function, which in this combination is "the scheme" of Bellare. *Id.* ¶259; Ex. 1006, 281; Ex. 1007, 93. This returned value is then stored in the variable k according to "the scheme" of Bellare. Ex. 1006, 281. Further, "the scheme" disclosed in Bellare teaches that the order q is a prime number represented by a bit string of predetermined length 160. Ex. 1006, 278, 281. This predetermined length of 160 bits applies equally and advantageously to the embodiment of Bellare-LEDA using the truncated linear congruential generator of Bellare, or with the SHA-1 cryptographic hash function recommended by Bellare. Ex. 1002, ¶259.

6. Claims 6, 20 and 28

Bellare in view of LEDA discloses "wherein said output from said hash function is a bit string of predetermined length L." Ex. 1002, ¶¶260-261. Bellare's hash function used in conjunction with its truncated linear congruential generator produces outputs represented by a bit string of predetermined length *h-l*, where *h* and *l* are input indices, and the resulting nonce k has 160 bits. Ex. 1006, 282. Bellare discloses that the recommended SHA-1 hash function for computing the

key k in DSA produces an output that is a bit string of predetermined length 160 bits. *Id.*, 278, 281. This use of a 160-bit hash function applies equally to the embodiment of Bellare-LEDA using the truncated linear congruential generator of Bellare or with the SHA-1 cryptographic hash function recommended by Bellare. Ex. 1002, ¶261.

7. Secondary Considerations

Petitioner is not aware of any evidence of secondary considerations of nonobviousness, such as commercial success, alleged long felt but unresolved need, alleged failure of others, or copying by others. Ex. 1002, ¶¶271-288.

8. Motivation to combine

A POSA would be motivated combine the teachings of Bellare with the teachings of LEDA, at least because a POSA would be motivated to improve the randomness and uniformity of the seed using techniques from LEDA, as applied to the cryptographic methods in Bellare. Ex. 1002, ¶262. This combination would be recognized by a POSA to have a reasonable expectation of success. *Id.*

A POSA would understand the disclosures in Bellare to teach the steps of the DSA algorithm (in Bellare, referred to as the DSS algorithm), as disclosed in "the scheme" described in Bellare. *Id.* ¶263; Ex. 1006, 281. A POSA would understand that Bellare teaches that the random number used for the nonce k in the DSA "scheme" should be uniformly chosen in the interval from 1 to q-1 as

randomly as possible. Ex. 1002, ¶264; Ex. 1006, 278. Bellare likewise stresses that the security of DSA depends on the secrecy and randomness of the nonce, k. Ex. 1002, ¶265; Ex. 1006, 281.

Thus, a POSA reading Bellare would be motivated to seek methods to improve the randomness and uniformity of the nonce, k, which Bellare discloses should be a random number uniformly chosen in the range from 1 to q-1. Ex. 1002, ¶266. LEDA describes how to use a random-bit generator to generate random numbers that are uniformly distributed in a given interval. *See* Ex. 1007, 92-93. Thus, a POSA would look to LEDA for such methods. Ex. 1002, ¶266. Further, LEDA teaches that reducing the function modulo (high-low+1) to generate a random number in a range [low,high] will produce a biased result. Ex. 1007, 92. Thus, a POSA would understand that LEDA's method of generating a random number (based on hashing and rejection sampling) instead generates a random number that is uniformly distributed in the range. Ex. 1002, ¶266; Ex. 1007, 92.

Bellare also teaches the use of hash functions with respect to the random number generator used to generate the nonce k in Bellare's description of "the scheme" for the DSS standard. Ex. 1002, ¶267. For example, Bellare teaches the use of a "truncated linear congruential generator," which uses a multiplicative hash function composed with a hash function that truncates the bits of a multiplicative hash function. Ex. 1006, 282. Bellare also teaches the use of a random generator

based on SHA-1 to generate the nonce k of "the scheme" for the DSS standard. *Id.*, 278.

As a result, a POSA would be motivated to replace the hash-function random number generator taught in LEDA (which is implemented in the "internal_source()" function) with either the truncated linear congruential generator or the cryptographic hash function, SHA-1, and then use this method to generate the nonce k disclosed in "the scheme" method taught in Bellare. Ex. 1002, ¶269. For example, LEDA discloses source code for generating a random number uniformly in the range from low to high. Ex. 1007, 93. The source code disclosed in the LEDA permits internal_source() to be implemented using any RNG, and gives a hash-function-based method for this. Ex. 1002, ¶270. A POSA thus would have an expectation of success from implementing internal_source() with SHA-1 or the truncated linear congruential generator of Bellare, and that this combination also would have a predictable result. *Id.* A POSA would be motivated to substitute one type of RNG (the hash-function based generator disclosed in LEDA) for internal_source() with a cryptographic hash function, such as the truncated linear congruential generator of Bellare (which uses a longer bit string of 160 bits than the 32-bit generator of LEDA) or with SHA-1 (which Bellare presents as an even better option), to achieve the expected result of uniformly generating a random number in a given range, thereby improving the

randomness of the nonce k. *Id.* Further, since the method taught in LEDA begins with a random seed, a POSA would also be motivated to use the same seed in this substitution. *Id.*; Ex. 1007, 92. Thus, a POSA would have a reasonable expectation of success of improving the uniformity and randomness of the nonce k with this combination and it would be obvious to try this combination of Bella. re with LEDA. Ex. 1002, ¶270.

VIII. CONCLUSION

Petitioner respectfully requests institution of IPR on the challenged claims.

Dated: October 2, 2020

Respectfully submitted,

/s/ *Sanjeet Dutta/*

Sanjeet Dutta
Registration No. 46,145
Goodwin Procter LLP
601 Marshall Street
Redwood City, CA 94063
Email: sdutta@goodwinlaw.com
Tel: (650) 752-3100
Fax: (650) 853-1038

CERTIFICATE OF COMPLIANCE WITH WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), the undersigned certifies that this Petition for *Inter Partes* Review complies with the type-volume limits of 37 C.F.R. § 42.24(a)(1)(i) because it contains 12,411 words according to the word-processing system used to prepare this Petition, excluding the words in the Table of Contents, Table of Authorities, List of Exhibits, Mandatory Notices, Certification Under §42.24(d), and Certificate of Service, as set forth in 37 C.F.R. § 42.24(a)(1).

October 2, 2020

/s/ *Sanjeet Dutta* /

Sanjeet Dutta
Registration No. 46,145
Goodwin Procter LLP
601 Marshall Street
Redwood City, CA 94063
Email: sdutta@goodwinlaw.com
Tel: (650) 752-3100
Fax: (650) 853-1038

CERTIFICATE OF SERVICE

The undersigned certifies that a true and correct copy of the Petition together with all exhibits identified in the above Table of Exhibits and Petitioner's Power of Attorney, have been served on the Patent Owner via Priority Mail Express or by means at least as fast and reliable as Priority Mail Express on the below date, at the following address:

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

Respectfully submitted,
GOODWIN PROCTER LLP

Date: October 2, 2020

By: */Sanjeet Dutta/*

Sanjeet Dutta
Registration No. 46,145
Goodwin Procter LLP
601 Marshall Street
Redwood City, CA 94063
Email: sdutta@goodwinlaw.com
Tel: (650) 752-3100
Fax: (650) 853-1038