DEFENDANTS' RESPONSIVE BRIEF ON CLAIM CONSTRUCTION

EXHIBIT 8

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re reexam of: U.S. Patent 5,490,216 to RICHARDSON, III	Confirmation No.: 2214
Reexam Control No.: 90/010,831	Art Unit: 3992
Filed: January 22, 2010	Examiner: HENEGHAN, Matthew E.
For: **System for Software Registration**	Atty. Docket: 2914.001REX0

Declaration of William R. Rosenblatt Under 37 C.F.R. § 1.132

Mail Stop *Ex Parte* Reexam
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

I, William R. Rosenblatt, declare as follows:

Retention:

1. I have been retained as a technical expert witness on behalf of Uniloc USA, Inc. and Uniloc Singapore Private Limited (collectively "Uniloc" herein) for the above-captioned reexamination. I understand that this reexamination involves U.S. Patent No. 5,490,216 ("the '216 patent") which resulted from Application No. 08/124,718, filed on September 21, 1993 on behalf of Frederic B. Richardson, III. I further understand that the '216 patent is currently assigned to Uniloc Singapore Private Limited.

Introduction:

2. Over the past seven years I have been retained as a technical expert witness in nine cases and have testified in that capacity in Federal Court.

Analysis of Claim 1 Over Hellman in View of Grundy:

34. Claim 1 of the '216 patent stands rejected on grounds of obviousness over Hellman in view of Grundy. Claim 1 recites local and remote "licensee unique ID" generation by the same algorithm on both local and remote systems.

35. The Office action asserts that "Hellman discloses several algorithms for local licensee unique ID and remote licensee unique ID generation" and then relies on Grundy to supply an algorithm for unique ID generation, noting that "it would be obvious to one of ordinary skill in the art at the time the invention was made to modify Hellman to use Grundy's checksum for ID generation, because it is easier to implement" than the summer disclosed in the '216 patent. ('216 Office action, p. 7).

Neither Hellman Nor Grundy Teach a Local or Remote Licensee Unique ID

36. To form an opinion on the above assertion in the Office action, first I consider whether either Hellman or Grundy teaches the generation of "licensee unique ID" (hereinafter "LUID") in any form. My opinion is that they do not. In support of the rejection, the '216 Office action cites to Hellman at 10:14-18, which refers to the "base unit" in Hellman's Figure 7, and equates that to the "local platform" in the '216 patent. The Examiner also cites to Hellman at 6:62-7:2, which refers to the "authorization and billing unit" in Hellman's Figure 2, and equates that to the "remote platform" in the '216 patent.

37. My showing that Hellman in combination with Grundy does not teach LUID generation follows three steps: First, I show that Hellman does not teach "***licensee*** ID generation." Second, I show that Hellman does not teach "***unique*** ID generation" (at

either local or remote locations). Third, I show that Grundy does not cure the deficiencies of Hellman such that the resulting combination teaches local or remote LUIDs.

38. First, regarding Hellman and "***licensee*** ID generation": the process in Hellman cited by the examiner above shows that the "check value C" (Hellman at 10:17), which the Examiner equates to LUID, is generated from four inputs, designated as K, N, R, and H. Of these: K is a ***key*** to a cryptographic function, which is an indicium of the ***computer*** on which the software is intended to be run. N is the ***number of usages*** of the software requested by the user (see Hellman at 5:65-66); R is a ***random number*** (Hellman at 5:66); H is an indicium of the ***software package*** being authorized for use (Hellman at 6:31-35), which is computed by means of a hash function. A hash function produces a value that serves as a mathematical "shorthand" for some data that has properties described appropriately in Hellman at 6:38-61, including that it is efficient to calculate and store.

39. None of these four inputs is an indicium of the ***licensee*** of the software, i.e., the user intending to run the software on the computer. Therefore Hellman does not teach "licensee ID generation."

40. I have reviewed Hellman's sworn testimony at trial. It reinforces my conclusion. The following is an excerpt from examination of Hellman at trial:

> [Attorney] Question: If you wanted to indicate that information associated with the user, unique information was input into the cryptographic function, you certainly had the ability to disclose that in the figures, if you so chose.
> [Hellman] Answer: Correct.
> [Attorney] Question: And you didn't?
> [Hellman] Answer: Correct.

> [Attorney] Question: And you also had the ability to describe in the patent, if you so chose?
> [Hellman] Answer: In the specification? Yes.
> [Attorney] Question: And you didn't?
> [Hellman] Answer: Correct.
> (March 31, 2009 Trial Transcript: p. 61, ll 17 - p. 62, ll 4, Uniloc USA, Inc. et al. v. Microsoft Corp., C.A. No. 03-440 (D.R.I.))

41. Second, I show that Hellman does not teach "***unique*** ID generation" at either local or remote locations. Three figures in Hellman depict the generation of the ID that is to be checked for validity: Figures 2, 6, and 7 (shown below). Figure 2 shows a Cryptographic Function with inputs SK, N, R, and H that is used on the base unit (equivalent to the local system in the '216 patent); its output is marked A and labeled 23. Figure 6 shows a more complete picture of the base unit. Figure 7 shows a Cryptographic Function with inputs K, N, R, and H and output C, which is part of the authorization and billing unit (equivalent to remote system in the '216 patent). These have outputs designated A and C respectively. Because these are intended to be equal if the software is authorized to run, I use the designation C hereinafter to refer to either of the outputs. The Examiner equates C to the LUID in the '216 patent.

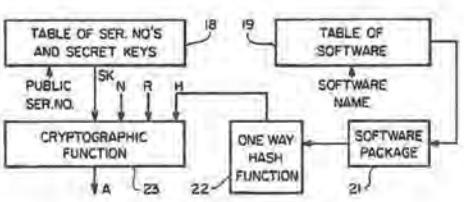


FIG_2

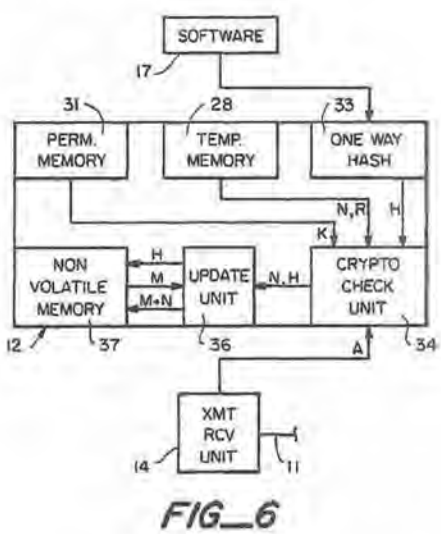


FIG_6

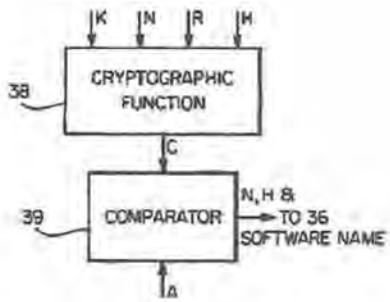


FIG_7

42. Hellman does not teach that C is unique. Here are descriptions of C (or A) as disclosed in the patent specification: "Generator 23 has the property that new authorizations cannot be inferred from old authorizations." (Hellman at 7:3-4 describing Figure 2.) This property is not the same thing as "authorizations must be unique." Authorizations need not be unique to satisfy this property, nor would a POSA infer uniqueness from this property.

43. For Figure 6, Hellman refers the reader to the ensuing description of Figure 7 for a disclosure of the output of the Cryptographic Function: "In a manner to be described shortly, the cryptographic check unit 34 determines if the authorization A is valid for the software package 17 being presened to the base unit 12 for he current authorization request." (Hellman at 9:54-57)

44. For Figure 7, Hellman discloses: "FIG. 7 depicts an implementation of the cryptographic check unit 34. Signals representing K. N, R and H are applied as inputs to a cryptographic function generator 38 which generates a check value C as an output signal." (Hellman at 10:14-18)

45. These are the only disclosures in Hellman that describe any properties of C. None of them teach that C must be unique, nor would they suggest such teaching to a POSA.

46. Therefore Hellman does not teach "***unique*** ID generation."

47. Because Hellman teaches neither "licensee ID generation" or "unique ID generation," Hellman does not teach local LUID generation, nor does Hellman teach remote LUID, and thus cannot by itself satisfy the "licensee unique ID generating means" element of claim 1 of the '216 patent.

Neither Hellman Nor Grundy Disclose or Teach Any Means for Generating Unique IDs:

48. The Office action claims that the combination of Hellman with Grundy teaches "local or remote licensee unique ID generation." The Office action relies on Hellman to teach ID generation, and introduces Grundy in order to add Grundy's checksum algorithm for generating unique IDs. As I will show, Grundy's checksum algorithm does not do this; therefore Grundy does not cure the acknowledged deficiency in Hellman. In short, neither Hellman nor Grundy disclose or teach any "local or remote licensee unique ID generation."

49. In support, I will now show that Grundy does not supply an adequate algorithm for generating unique IDs. The Examiner cites to Grundy, including "the checksum originally calculated … from the owner data as entered by the user during the registration process" (See, Grundy 15:11-13, figure citation omitted) and "a checksum of the user data" (See, Grundy 18:25-26). ('216 Office action p. 7.)

50. First, it is worth noting that the Order granting reexamination of the '216 patent characterizes checksums as non-unique: "***Checksums are not unique fields***, even if there [sic] are at least in part derived from unique data." (Order Granting/Denying Request for Ex Parte Reexamination, p. 9, emphasis added)

51. I concur with that characterization of checksums. A checksum is not usable as a generator of unique IDs. Grundy does not define "checksum" in the specification in the context of "a checksum of the user data," but rather suggests definitions elsewhere in his specification that are consistent with both dictionary definitions and a definition that a POSA would use, as I will now show.

52. A POSA would understand "checksum" to mean a small number of check digits that are typically appended to data in order to ensure the data's integrity when it is copied, entered by a user, or transmitted. To calculate a checksum of some data, the data can be added up (e.g., broken up into C-byte chunks, where C is a small number such as 1, 2, 4, or 8, and summed); the sum is chopped to a fixed length (e.g., C bytes) and appended to the data before storage or transmission. Checksum algorithms used in practice are variations on this scheme. When the data is received or retrieved later, the checksum is re-calculated to ensure that the result is the same as the original checksum; if the result differs then the data must have been corrupted.

53. The Grundy patent was filed in April 1992 and issued in March 1994. Definitions of "checksum" from dictionaries of computer terms from that era concur with the POSA's definition that I have provided above.

54. For example: *The Computer Glossary: The Complete Illustrated Desk Reference,* 6th Edition, 1993 defines "checksum" as a "Value used to ensure data is

transmitted without error. It is created by adding the binary value of each alphanumeric character in a block of data and sending it with the data. At the receiving end, a new checksum is computed and matched against the transmitted checksum. A non-match indicates an error."

55. As another example, the *IBM Dictionary of Computing*, 10th Edition, 1994 defines "checksum" as: "(1) The sum of a group of data associated with the group and used for checking purposes. (2) In error detection, a function of all bits in a block. If the written and calculated sums do not agree, an error is indicated. (3) On a diskette, data written in a sector for error detection purposes; a calculated checksum that does not match the checksum of data written in the sector indicates a bad sector. The data are either numeric or other character strings regarded as numeric for the purposes of calculating the checksum."

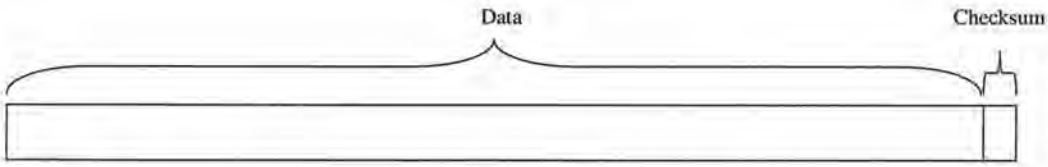
56. In all of these cases, a checksum is much smaller in length than its input data. For example, a 16-bit (2-byte) or 64-bit (8-byte) checksum may be calculated on thousands, millions, or billions of bytes of data. This fulfills the checksum's intended purpose well, given that most errors in data storage or transmission are small and localized, making it highly likely that the resulting checksum will differ from the one originally calculated, and extremely unlikely that corrupted data will produce the same checksum as the original one. For example, if one or two bits are altered, the checksum will differ.

57. A few examples highlight this point. Most credit card numbers contain check digits that serve to make sure that a credit card holder enters the number properly (e.g., on a website). The check digits are often calculated by means of an

algorithm called the Luhn algorithm, which is a type of checksum algorithm. Because it is a single digit (with values 0-9), it is clearly impossible for the check digit to be unique, given that most credit card numbers are 15 digits long (not counting the check digit) and therefore that there are up to 10^{15} or 1 quadrillion possible credit card numbers.

58. As another example, consider social security numbers, which are 9 digits long. Suppose we wanted to calculate the checksum of a social security number. We can use the most basic checksum algorithm, in which all the digits in the input data are added and the result is the checksum. All checksum algorithms in practical use are derived from this basic one. The social security numbers 123-45-6789 and 987-65-4321 are clearly different, but they both have the same checksum of 45; moreover, it is impossible to reconstruct (preserve the uniqueness of) the original social security number given the checksum.

59. As a final example, electronic devices often have memory chips in them that contain data stored at the factory. These are known as PROMs (Programmable Read-Only Memory). A typical PROM will contain a checksum of 8 bytes in length, even though the capacity of the PROM may be in the thousands or millions of bytes, as illustrated in the diagram below. The data in the PROM can be added up byte by byte, and the lowest 8 bytes of the result can be used as the checksum. The checksum is used to ensure that the data in the PROM was copied correctly during manufacturing. It works well for that purpose, because errors are typically limited to a few bytes, which would result in a different checksum being calculated. However, many different sets of PROM data would lead to the same checksum, meaning that uniqueness of the data is not preserved.



60. Grundy himself suggests the use of checksums as error-checking means in disclosures of checksums in his invention other than the "checksum of the user data" used to generate the code that the Office action alleges satisfies the LUID element of claims in the '216 patent. Here are some examples:

- "If there is a match, the ***anti-virus checksum*** as originally calculated during the Franking process retrieved from the central database is compared with the checksum of the host software product as calculated in the process of generating the registration code. If these two checksums do not match, it indicates that ***the host software product has in some way been deliberately or accidentally corrupted***." (Grundy at 15:28-36, figure citations omitted, emphasis added.)
- "If the user data validity check passes, a checksum of the user data is created. This checksum will be used during the Authorization process as a cross-reference to ***validate the user data*** as communicated to the Manufacture Control Agency." (Grundy at 18:26-29, figure citations omitted, emphasis added.)
- "A checksum is then calculated from the bits in the array and added to the packed bit array. This checksum will be used by the Manufacture Control Agency to ***avoid operator data-entry errors***." (Grundy at 19:4-8, figure citations omitted, emphasis added.)

61. From the foregoing, Grundy understood well the use of checksums according to the POSA's and dictionary definitions described above; therefore one may assume that his understanding of checksums applies to all of the disclosures of that term in his patent, including the "checksum of the user data" as referred to above.

62. Grundy shows the input data to this checksum routine in Fig. 2, 212, "ENTER NEW USER DETAILS." This is "new user data, such as the user's name, address and telephone number" (Grundy at 12:37-38). Such data might take up roughly a hundred bytes of data. As indicated above, a checksum of this data would not preserve its uniqueness; many different sets of user data could produce the same checksum.

63. For the above reasons, a checksum cannot possibly preserve whatever uniqueness the input data may possess. In particular, a POSA would not ascribe any reasonable definition of "unique" to the output of a checksum routine.

64. Therefore the checksum as disclosed in Grundy cannot function as a generator of unique identifiers, which is a required characteristic of the claimed LUID. Grundy thus does not overcome the acknowledged deficiency in Hellman.

65. I therefore disagree with the factual conclusions stated in the Office action with respect to the teachings of Hellman and Grundy—neither of them teach or disclose licensee unique ID generation.

Extension of Arguments to Independent Claim 12:

66. As with claim 1, the '216 Office action also rejects claim 12 of the '216 patent on grounds of obviousness with respect to Hellman in view of Grundy. Claim 12 of the '216 patent recites a "registration system" to generate a "security key"

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the '216 patent.

Executed this 23rd day of November 2010 in New York, NY.

Respectfully submitted,

William R. Rosenblatt