

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTERNATIONAL BUSINESS MACHINES CORPORATION

Petitioner

v.

SECURITY FIRST INNOVATION, LLC

Patent Owner

Case No. IPR2025-01201

U.S. Patent No. 8,904,194

Claims 1–20

DECLARATION OF DR. EREZ ZADOK, PH.D.

TABLE OF CONTENTS

	<u>Page</u>
I. Background and Qualifications	1
II. Materials Considered.....	12
III. Summary of Opinions.....	15
IV. Relevant Law	15
A. Claim Construction.....	16
B. Obviousness.....	16
C. Ordinary Skill in the Art.....	18
V. Technology Overview	19
A. General Computer Operations.....	19
B. Networking.....	24
1. Networked and Distributed Storage Systems	28
C. Data Security and Privacy	32
1. Hash Functions and Their Uses	39
2. Secret Splitting and Sharing.....	41
D. Recovering from Data Loss.....	44
1. Selecting The Fastest-Responding Servers.....	50
VI. Overview Of The '194 Patent.....	54
A. Prosecution History	62
VII. Level Of Ordinary Skill In The Art.....	63
VIII. Claim Construction.....	64
IX. Specific References And Grounds For Challenge.....	64

A.	Ground I: Dickinson And Hardjono Render Obvious Claims 1–	
	20.....	64
1.	Overview of Dickinson (EX1003).....	65
2.	Overview of Hardjono (EX1004)	71
3.	Overview of Dickinson in View of Hardjono ("Dickinson/Hardjono").....	76
i.	Fastest Responding Storage Devices in View of Hardjono	77
a.	Reasons to Combine Dickinson with Hardjono’s Teaching of Fastest Responding Storage Devices	77
ii.	Sending Assembled Data to the Requester in Response to a Request in View of Hardjono.....	82
a.	Reasons to Combine Dickinson with Hardjono’s Teaching of Sending Assembled Data to the Requester in Response to a Request.....	84
iii.	Encrypting Data Portions with Separate Keys in View of Hardjono	93
a.	Reasons to Combine Dickinson with Hardjono’s Teaching of Encrypting Data Portions with Separate Keys	95
4.	Color-Coding Scheme for Dickinson/Hardjono	98
5.	Claim 1: A method for securely storing and retrieving data, the method comprising: generating, using an electronic computing system that includes processing circuitry, a plurality of shares by performing a cryptographic operation on a data set and distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of shares; storing the plurality of shares at a plurality of storage devices; receiving, at the electronic computing	

system, request to retrieve the data set; identifying from the plurality of storage devices a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices; retrieving from the set of fastest-responding storage devices, the minimum number of shares; reconstructing the data set using the minimum number of shares; and sending the data set responsive to the request.99

i. [1Pre] A method for securely storing and retrieving data, the method comprising:.....99

ii. [1A] generating, using an electronic computing system that includes processing circuitry, a plurality of shares by performing a cryptographic operation on a data set and distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of shares;.....100

a. [1A-1] generating, using an electronic computing system that includes processing circuitry, a plurality of shares by performing a cryptographic operation on a data set and100

b. [1A-2] distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of shares;107

iii. [1B] storing the plurality of shares at a plurality of storage devices;.....111

iv. [1C] receiving, at the electronic computing system, request to retrieve the data set;112

v. [1D] identifying from the plurality of storage devices a set of fastest-responding storage devices necessary to retrieve the minimum number of

shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices;113

vi. [1E] retrieving from the set of fastest-responding storage devices, the minimum number of shares;115

vii. [1F] reconstructing the data set using the minimum number of shares; and.....117

viii. [1G] sending the data set responsive to the request.118

6. Claim 2.....119

i. [2Pre] The method of claim 1, wherein storing the shares comprises:119

ii. [2A] storing a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center; and.....119

iii. [2B] storing a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.125

7. Claim 3: The method of claim 1, further comprising establishing secure connections between the electronic computing system and the plurality of storage devices.127

8. Claim 4: The method of claim 1, wherein the plurality of data blocks share contain a substantially random distribution of the data set.....128

9. Claim 5: The method of claim 1, wherein the data set can be reconstructed from shares received from at least two storage devices.129

10. Claim 6: The method of claim 1, wherein the shares are encrypted with a corresponding number of different keys.130

11. Claim 7.....131

- i. [7Pre] An electronic computing system for securely storing and retrieving data, the electronic computing system comprising:.....131
 - ii. [7A] a processing unit; and.....132
 - iii. [7B] a system memory comprising instructions that, when executed by the processing unit, cause the processing unit to:.....133
 - iv. [7B-1] generate a plurality of shares by performing a cryptographic operation on a data set and distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of the shares and such that the data set cannot be reconstructed using any subset of the shares that includes fewer than the minimum number of the shares;135
 - v. [7B-2] store the plurality of shares at a plurality of storage devices;.....136
 - vi. [7B-3] receive, via the primary interface, a request to retrieve the data set;.....136
 - vii. [7B-4] identify, from the plurality of storage devices, a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices, retrieve from the set of fastest-responding storage devices the minimum number of shares;138
 - viii. [7B-5] reconstruct the data set using exclusively the minimum number of shares; and139
 - ix. [7B-6] send the data set responsive to the request.140
12. Claim 8: The electronic computing system of claim 7, wherein the instructions cause the processing unit to store

a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.141

i. [8Pre]: The electronic computing system of claim 7, wherein the instructions cause the processing unit to 141

ii. [8A]: to store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and..... 142

iii. [8B]: to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center..... 142

13. Claim 9143

i. [9] The electronic computing system of claim 7, wherein the instructions further cause the processing unit to generate the plurality of shares in response to receiving a request to store the data set.143

14. Claim 14147

i. [14Pre] A non-transitory computer-readable storage medium comprising instructions that, when executed at an electronic computing device, cause the electronic computing device to:..... 147

ii. [14A] receive a request to write a data set to a storage location;..... 149

iii. [14B] generate a plurality of shares by performing a cryptographic operation on the data set and distributing the data set in the plurality of shares

such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of the shares and such that the data set cannot be reconstructed using any subset of the shares that includes fewer than the minimum number of the shares;152

iv. [14C] store the plurality of shares at a plurality of storage devices;.....152

v. [14D] receive a request to retrieve the data set;153

vi. [14E] identify, from the plurality of storage devices, a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices;153

vii. [14F] retrieve from the set of fastest-responding storage devices, the minimum number of shares;153

viii. [14G] reconstruct the data set using exclusively the minimum number of shares; and153

ix. [14H] send the data set responsive to the request.153

15. Claim 16: The non-transitory computer-readable storage medium of claim 14, wherein the instructions further cause the processing unit to generate the plurality of shares in response to receiving the request to write the data set to the storage location.....154

16. Claims 10–13, 15, and 17–20154

B. Ground II: Dickinson, Hardjono, And Moulton Render Obvious Claims 2, 8, And 15.....155

1. Overview of Moulton (EX1018).....155

2. Overview of Dickinson in Combination with Hardjono and Moulton157

i.	Reasons to Combine Dickinson/Hardjono with Moulton.....	158
3.	Claim 2.....	163
i.	[2Pre] The method of claim 1, wherein storing the shares comprises:	163
ii.	[2A] storing a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center; and.....	163
iii.	[2B] storing a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.	164
4.	Claim 8: The electronic computing system of claim 7, wherein the instructions cause the processing unit to store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.	167
i.	[8Pre]: The electronic computing system of claim 7, wherein the instructions cause the processing unit to	167
ii.	[8A]: to store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and.....	168
iii.	[8B]: to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.	169

5. Claim 15: The non-transitory computer-readable storage medium of claim 14, wherein the instructions cause the processing unit to store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.169

X. Conclusion169

XI. Availability for Cross-Examination169

XII. Right to Supplement170

XIII. Jurat.....170

LISTING OF CLAIMS

Claim 1

[1Pre] A method for securely storing and retrieving data, the method comprising:

[1A-1] generating, using an electronic computing system that includes processing circuitry, a plurality of shares by performing a cryptographic operation on a data set and

[1A-2] distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of shares;

[1B] storing the plurality of shares at a plurality of storage devices;

[1C] receiving, the electronic computing system, request to retrieve the data set;

[1D] identifying from the plurality of storage devices a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices;

[1E] retrieving from the set of fastest-responding storage devices, the minimum number of shares;

[1F] reconstructing the data set using the minimum number of shares; and

[1G] sending the data set responsive to the request.

Claim 2

[2] The method of claim 1, wherein storing the shares comprises:

[2A] storing a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center; and

[2B] storing a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.

Claim 3

[3] The method of claim 1, further comprising establishing secure connections between the electronic computing system and the plurality of storage devices.

Claim 4

[4] The method of claim 1, wherein the plurality of data blocks shares contain a substantially random distribution of the data set.

Claim 5

[5] The method of claim 1, wherein the data set can be reconstructed from shares received from at least two storage devices.

Claim 6

[6] The method of claim 1, wherein the shares are encrypted with a corresponding number of different keys.

Claim 7

[7Pre] An electronic computing system for securely storing and retrieving data, the electronic computing system comprising:

[7A] a processing unit; and

[7B] a system memory comprising instructions that, when executed by the processing unit, cause the processing unit to:

[7B-1] generate a plurality of shares by performing a cryptographic operation on a data set and distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of the shares and such that the data set cannot be reconstructed using any subset of the shares that includes fewer than the minimum number of the shares;

[7B-2] store the plurality of shares at a plurality of storage devices;

[7B-3] receive, via the primary interface, a request to retrieve the data set;

[7B-4] identify, from the plurality of storage devices, a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the

response time of the storage devices, retrieve from the set of fastest-responding storage devices the minimum number of shares;

[7B-5] reconstruct the data set using exclusively the minimum number of shares;
and

[7B-6] send the data set responsive to the request.

Claim 8

[8Pre] The electronic computing system of claim 7, wherein the instructions cause the processing unit to

[8A] store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and

[8B] to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.

Claim 9

[9] The electronic computing system of claim 7, wherein the instructions further cause the processing unit to generate the plurality of shares in response to receiving a request to store the data set.

Claim 10

[10] The electronic computing system of claim 7, wherein the instructions further cause the processing unit to establish secure connections between the electronic computing system and the plurality of storage devices

Claim 11

[11] The electronic computing system of claim 7, wherein the plurality of shares contain a substantially random distribution of the data set.

Claim 12

[12] The electronic computing system of claim 7, wherein the data set can be reconstructed from shares received from at least two storage devices.

Claim 13

[13] The electronic computing system of claim 7, wherein the shares are encrypted with a corresponding number of different keys.

Claim 14

[14Pre] A non-transitory computer-readable storage medium comprising instructions that, when executed at an electronic computing device, cause the electronic computing device to:

[14A] receive a request to write a data set to a storage location;

[14B] generate a plurality of shares by performing a cryptographic operation on the data set and distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of the shares and such that the data set cannot be reconstructed using any subset of the shares that includes fewer than the minimum number of the shares;

[14C] store the plurality of shares at a plurality of storage devices;

[14D] receive a request to retrieve the data set;

[14E] identify, from the plurality of storage devices, a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices;

[14F] retrieve from the set of fastest-responding storage devices, the minimum number of shares;

[14G] reconstruct the data set using exclusively the minimum number of shares; and

[14H] send the data set responsive to the request.

Claim 15

[15] The non-transitory computer-readable storage medium of claim 14, wherein the instructions cause the processing unit to store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.

Claim 16

[16] The non-transitory computer-readable storage medium of claim 14, wherein the instructions further cause the processing unit to generate the plurality of shares in response to receiving the request to write the data set to the storage location.

Claim 17

[17] The non-transitory computer-readable storage medium of claim 14, wherein the instructions further cause the processing unit to establish secure connections between the electronic computing device and the plurality of storage devices.

Claim 18

[18] The non-transitory computer-readable storage medium of claim 14, wherein the plurality of shares contain a substantially random distribution of the data set.

Claim 19

[19] The non-transitory computer-readable storage medium of claim 14, wherein the data set can be reconstructed from shares received from at least two storage devices.

Claim 20

[20] The non-transitory computer-readable storage medium of claim 14, wherein the shares are encrypted with a corresponding number of different keys.

I, Erez Zadok, declare as follows:

1. My name is Erez Zadok.
2. I have been retained as an expert witness in the field of data storage, file systems, and data security protocols on behalf of Petitioner International Business Machines Corporation (“IBM”) to provide expert opinions on the validity of U.S. Patent No. 8,904,194 (the “’194 Patent”).

I. Background and Qualifications

3. My *curriculum vitae*, which includes a more detailed summary of my background, experience and publications, is attached as EX1016.

4. I am a Professor in the Computer Science Department at Stony Brook University (part of the State University of New York (“SUNY”) system). I direct the File-systems and Storage Lab (FSL) at Stony Brook’s Computer Science Department. My research interests include file systems and storage systems, operating systems, transactional systems including database technologies, information technology and system administration, security/privacy and information assurance, networking, energy efficiency, performance and benchmarking, virtualization, cloud systems, compilers, applied machine learning, and software engineering.

5. I studied at a professional high school in Israel, focusing on electrical engineering (“EE”), and graduated in 1982. I spent one more year at the high

school's college division, receiving a special Certified Technician's degree in EE. I then went on to serve in the Israeli Defense Forces for three years (1983–1986). I received my Bachelor of Science degree in computer science (“CS”) in 1991, my Master's degree in CS in 1994, and my PhD in CS in 2001—all from Columbia University in New York.

6. When I began my undergraduate studies at Columbia University, I also started working as a student assistant in the various campus-wide computer labs, eventually becoming an assistant to the head labs manager, who was managing all public computer labs on campus. During that time, I also became more involved with research within the CS Department at Columbia University, conducting research on operating systems, file and storage systems, distributed and networked systems, security, and other topics. I also assisted the CS department's computer administrators in managing the department's computers, which included storage, IT, networking, and cyber-security related duties.

7. In 1991, I joined Columbia University's CS department as a full-time systems administrator, studying towards my MS degree part-time. My MS thesis topic is related to file system reliability, fault tolerance, replication, and failover in mobile networked storage systems using file virtualization. My main duties as a systems administrator involved installing, configuring, and managing many networked servers, proxies, and desktops running several operating systems, as well

as network devices setup; this included many software and hardware upgrades, device upgrades, and BIOS firmware/chipset updates/upgrades. My duties also included ensuring reliable, secure, authenticated access to networked systems/storage and licensed software, as well as software updates, security and bug fixes. Examples of servers and their protocols included email (SMTP), file transfer (FTP), domain names (DNS), network file systems (NFS), network news systems (NNTP), and Web (HTTP).

8. In 1994, I left my systems administrator position to pursue my doctoral studies at Columbia University. My PhD thesis topic was on versatile file system development using stackable (virtualized) file systems, with examples in the fields of security and encryption, efficiency, reliability, and failover. I continued to work part-time as a systems administrator at the CS department, and eventually I was asked to serve as manager to the entire information technology (“IT”) staff. From 1991 to 2001, I was also a member of the faculty-level Facilities Committee that oversaw all IT operations at the CS department.

9. As part of my PhD studies at Columbia, I collaborated on projects to develop advanced AI-like techniques to detect previously unknown viruses (a.k.a. “zero-day malware”), using data mining and rule-based detection. This work led to several highly cited papers (over 1,600 citations for one of the papers alone) and two patents. I also became a Teaching Assistant (“TA”) for a first-ever Computer

Security course given at Columbia University's CS department with Dr. Matt Blaze as instructor.

10. From 1990 to 1998, I consulted for SOS Corporation and HydraWEB Technologies as a systems administrator and programmer, managing data storage use and backup/restore duties, databases, web servers, as well as information assurance and cyber-security (*e.g.*, malware protection, software licensing). From 1994 to 2000, I led projects at HydraWEB Technologies, and then became the Director of Software Development—overseeing the development of several products and appliances such as stateful firewalls and HTTP load-balancers, utilizing network-virtualization and high-availability techniques. From 2009 to 2019, I have consulted for Packet General Networks, a startup specializing in secure, virtualized, network storage and applications' data security in the cloud.

11. In 2001, I joined the faculty of Stony Brook University, a position I have held since that time. In 2002, I joined the Operations Committee, which oversees the IT operations of the CS department at Stony Brook University. From 2006 to 2010, I was the Director of IT Operations of the CS department. My day-to-day duties included setting policies regarding computing, hiring and training new staff, assisting any staff with topics of my specialty, defining requirements for new software/hardware, and purchasing. From 2010 to 2015, I had served as the Co-Chair to the Operations Committee. From 2016 to 2019, I oversaw the IT Operations

as the Chair of the Operations Committee. A significant component of these duties included defining and helping implement policies for data management, to ensure the security of users and their data, and data reliability and availability, while minimizing the inconvenience and performance impact to users. I personally helped setup and maintain an initial virtual-host infrastructure in the department. Since late 2019, I've been a member of the department's Executive Committee that also oversees all IT operations.

12. In 2017, I became the department's Graduate Academic Adviser, advising all Master students (over 400 annually on average) and many other graduate students on an assortment of academic matters. In 2024, I took over as the department's Graduate Program Director, overseeing the entire graduate CS program (700-800 students annually on average).

13. Since 2001, I have personally configured and managed my own research lab's network. This includes setting up and configuring multiple storage systems (*e.g.*, NFS, CIFS/SMB, NAS), virtual and physical environments, applications such as database (*e.g.*, MySQL, PostgreSQL), Web servers (*e.g.*, Apache), and mail servers; user access control (*e.g.*, NIS, LDAP), backups and restores, snapshot policies, and more. I have personally installed, configured, changed, replaced parts, and upgraded components in numerous devices including mobile devices, laptops, desktops, and servers, both physical and virtual.

14. Since 1995, I have taught courses on operating systems, storage and file systems, advanced systems programming in Unix/C, systems administration, data structures, data/software security, and more. My courses often use storage, file systems, distributed systems, and system/network security as key teaching principles and practical examples for assignments and projects. I have taught these concepts and techniques to my students, both to my direct advisees as well as in my courses. For example, in my graduate Operating Systems course, I often cover Linux's kernel mechanisms to protect users, applications, and data files, virtual file systems, as well as distributed storage systems (*e.g.*, NFS). And in the System Administration undergraduate course, I covered many topics such as networking, storage, backups, and configuring complex applications such as mail, web, and database servers.

15. My research often investigates computer systems from many angles: security, efficiency, energy use, scalability, reliability, portability, survivability, usability, ease-of-use, versatility, flexibility, and more. My research gives special attention to balancing five often-conflicting aspects of computer systems: performance, reliability, energy use, security, and ease-of-use.

16. Since joining Stony Brook University in 2001, my group in the File-systems and Storage Lab ("FSL") has developed many file systems and operating system extensions; examples include a highly-secure cryptographic file system, a portable copy-on-write ("COW") versioning file system, a tracing file system useful

to detect intrusions, a replaying file system useful for forensics, a snapshotting and sandboxing file system, a namespace unification file system (that uses stackable, virtualized, file-based COW), an anti-virus file system, an integrity-checking file system, a load balancing and replication/mirroring file system, network file system extensions for security and performance, distributed secure cloud-based storage systems, transactional key-value stores and file systems, OS-level embedded databases, a compiler to convert user-level C code to in-kernel efficient yet safe code, GCC plugins, stackable file system templates, and a Web-based backup system. Many of these projects used one form of virtualization or another (storage, network, host, etc.). I continue to maintain and release newer versions of some of these file systems and software.

17. I have published over 120 refereed publications (in ACM, IEEE, USENIX, and more). To date, my publications have been cited more than 10,000 times (as per Google Scholar as of June 13, 2025). My papers cover a wide range of related technologies such as file systems, storage systems, transactional systems, security, clouds and virtualization, performance benchmarking and optimization, energy efficiency, system administration, web systems, and more. I also published a book titled “Linux NFS and Automounter Administration” (Sybex, 2001), covering systems administration topics related to network storage and data security.

18. Some of my research has led to public software releases that have been used worldwide. I have publicly maintained the Amd Berkeley Automounter in a package called “am-utils” since 1992; this software helps administrators manage the multitude of file system mounts on dozens of different Unix systems, especially helping to automate access to multiple NFS/NAS storage volumes. Since 1997, I have maintained and released several stackable (virtualized) file system software projects for Linux, FreeBSD, and/or Sun Solaris, in a package called FiST. One of my stackable file system encryption projects, called Cryptfs, became the basis for IBM’s public release of eCryptfs, now part of Linux. Packet General Networks, for whom I have provided consulting services between 2009 and 2019, licensed another encryption file system called Ncryptfs. Another popular file system released in 2003, called Unionfs, offers virtual namespace unification, transparent shadow copying (a.k.a. copy-on-write or COW), file system snapshotting (*e.g.*, useful for forensics and disaster recovery), and the ability to save disk space by sharing a read-only copy of data across several computers, among other features.

19. My research and teaching make extensive use of data security features. For example, each time I taught the graduate operating system course, the first homework assignment includes the creation of a new system call that performs new or added functionality, often for encrypting a file or verifying its integrity; many of my other assignments cover topics of user/process access control, anti-virus filtering,

and more. Since 2001, over 1,000 graduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

20. Moreover, in an undergraduate course titled “Advanced Systems Programming in Unix/C,” I cover many topics of system security and vulnerabilities, such as the structure of UNIX processes, and memory segments such as the heap and stack. This course covers details of several hundred Linux system calls. Often, the first assignment for this course is to develop a tool to encrypt/decrypt files using advanced ciphers, use digital signatures to certify the cipher keys used, and reliably recover files in case of failures. Since 2001, several hundred undergraduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

21. In another undergraduate course, System Administration, I taught network configuration, security, and storage configuration and reliability. In a special topics course on Storage Systems, I covered many topics such as data deduplication, RAID, transactional storage, storage hardware including modern Flash based ones, virtual storage, backup/restore, snapshots and continuous data protection (“CDP”), NAS and SAN, and NFS.

22. Overall, in addition to the aforementioned experience, my technical experience relevant to the ’194 Patent at the time of the alleged invention included the following: I studied, researched, developed code, published, taught, and/or

released software for the following: encryption and integrity software using logical operations such as XOR; encryption file systems including my own (Cryptfs and a licensed NCryptfs); file system integrity using CRC codes and hashes such as MD4 and MD5; digital signatures; RAID software with parity codes such that data can be recovered from a subset of the data copies; distributed and networked storage and file systems (*e.g.*, NFS, SMB/CIFS); a mobile, networked file system that where clients can move across large geographic distances, then measure, locate, and [re]connect to the fastest responding file systems (this was part of my MS Thesis); and a file system that splits files across multiple locations.

23. My research has been supported by many federal and state grants as well as industry awards, including an NSF CAREER award, two IBM Faculty awards, two NetApp Faculty awards, a Western Digital award, a Facebook award, several Dell-EMC awards, and several equipment gifts. I received the 2008 SUNY Chancellor's Excellence in Teaching award, and the 2022 SUNY Chancellor's Award for Excellence in Scholarship and Creative Activities (both awards can be given only once in a lifetime). In 2021, I was named an ACM Distinguished Member for "Outstanding Scientific Contributions to Computing."

24. My service record to the community includes serving as the co-chair for the USENIX Annual Technical Conference in 2020 (ATC'20); serving as the co-chair for USENIX File and Storage Technologies (FAST'15) in 2015 and on the

FAST Conference Steering Committee from 2015 to 2023; serving on the ACM HotStorage Steering Committee since 2021; and serving as the co-chair in 2012 and on the Steering Committee of the ACM SYSTOR conference since 2012. I have served as an Associate Editor to the ACM Transactions on Storage (“TOS”) journal from 2009 to 2022; in 2022, I was named the Editor-in-Chief for ACM’s TOS journal.

25. I am a named inventor on four patents, two titled “Systems and Methods for Detection of New Malicious Executables” (U.S. Patent No. 7,487,544, issued February 3, 2009; and U.S. Patent No. 7,979,907, issued July 12, 2011); and two more titled “Multi-Tier Caching,” (U.S. Patent No. 9,355,109, issued May 31, 2016; and U.S. Patent 9,959,279, issued May 1, 2018).

26. I have been disclosed as a testifying expert in 22 cases (including *inter partes* review (“IPR”) proceedings) in the past four years. I have been deposed 13 times and testified in trial twice.

27. I have personal knowledge of the facts and opinions discussed in this declaration and believe them to be true. If called upon to do so, I would testify competently thereto. I have been warned that willful false statements and the like are punishable by fine or imprisonment, or both. I have had no contact with any of the named inventors of the ’194 Patent.

28. My consulting company, Zadoks Consulting Services, is being compensated for my time at my current standard consulting rate. I am also being reimbursed for expenses that I may incur during the course of this work. My compensation is not contingent upon the results of my study and analysis, the substance of my opinions, or the outcome of any proceeding involving the Challenged Claim(s).

29. I have no financial interest in the Petitioner. I have no financial interest in the outcome of this matter or in any litigation involving the '194 Patent.

II. Materials Considered

30. I understand that the '194 Patent has a filing date of May 10, 2012, claiming priority to U.S. Provisional Application Serial Nos. 60/622,146, filed October 25, 2004; and 60/718,185, filed September 16, 2005. EX1001 ('194 Patent); EX1009 (Provisional Application No. 60/622,146); EX1010 (Provisional Application No. 60/718,185).

31. Therefore, I assume that the earliest possible priority date of the '194 Patent is October 25, 2004. I have considered issues from the perspective of a person of ordinary skill in the art as described in §VII below [Level Of Ordinary Skill In The Art] as of October 25, 2004.

32. I have reviewed the following documents in preparing this declaration. I understand that these are prior art to the '194 Patent.

- PCT Patent Application Publication No. WO2001/022322 (“Dickinson”) (EX1003)
- U.S. Patent No. 6,363,481 (“Hardjono”) (EX1004)
- U.S. Patent Application Publication No. US2001/0034795 (“Moulton”) (EX1018)

33. I have also reviewed all of the exhibits cited in this declaration, including the following:

Exhibit No.	DESCRIPTION
1001	U.S. Patent No. 8,904,194 (“’194 Patent”)
1003	PCT Patent Application Publication No. 01/022322 (“Dickinson”)
1004	U.S. Patent No. 6,363,481 (“Hardjono”)
1005	Claudia Canali et al., Performance Comparison of Distributed Architectures for Content Adaptation and Delivery of Web Resources, 25 IEEE Int’l Conf. on Distributed Comput. Sys. Workshops 331 (2005) (“Canali”)
1006	File History of U.S. Patent No. 8,904,194 (“’194 File History”)
1007	File History of U.S. Patent Application Serial No. 11/258,839 (“’839 App. File History”)
1008	Reserved

Exhibit No.	DESCRIPTION
1009	U.S. Provisional Application Serial No. 60/622,146 (“Provisional Application No. 60/622,146”)
1010	U.S. Provisional Application Serial No. 60/718,185 (“Provisional Application No. 60/718,185”)
1011	Microsoft Computer Dictionary, 5 th ed., 2002, excerpts (“Microsoft Computer Dictionary”)
1012	Bruce Schneier, Applied Cryptography, 2nd ed., 1996, excerpts (“Schneier”)
1013	Highlighted version of the ’194 Patent’s specification where the highlighted portion appears verbatim in Dickinson
1014	Yitzhak Birk, “Random RAIDs with Selective Exploitation of Redundancy for High Performance Video Servers,” IEEE 1997 (“Birk”)
1015	U.S. Patent Publication No. 2003/0016596A1 (“Chiquoine”)
1016	Curriculum Vitae of Dr. Erez Zadok
1017	John Kubiawicz, et al., “OceanStore: An Architecture for Global-Scale Persistent Storage,” ACM 2000 (“Kubiawicz”)
1018	U.S. Patent Publication No. 2001/0034795 (“Moulton”)

Exhibit No.	DESCRIPTION
1019	U.S. Patent Publication No. 2003/0046551 (“Brennan”)
1020	U.S. Patent Publication No. US2002/0049655 (“Bennet”)
1021	<i>Google, LLC v. Security First Innovations, LLC</i> , IPR2024-00212, Exhibit 1043 (Patent Owner’s Proposed Claim Constructions In <i>Security First Innovations, LLC v. Google, LLC</i> , No. 2:23-cv-00097 (E.D. Va.)) (“PO’s Google Litigation Construction”)

III. Summary of Opinions

34. It is my opinion that claims 1–20 of the ’194 Patent are obvious in view of the Ground identified below:

Ground	References	Challenged Claims
I	Dickinson, Hardjono	1-20
II	Dickinson, Hardjono, Moulton	2, 8, 15

IV. Relevant Law

35. I am not an attorney. For the purposes of this declaration, I have been informed about certain aspects of the law that are relevant to my opinions. My understanding of the law is as follows:

A. Claim Construction

36. I have been informed that claim construction is a matter of law and that the final claim construction will ultimately be determined by the Board. For the purposes of my analysis in this proceeding and with respect to the prior art, I have been informed that IPRs are currently reviewed under “the *Phillips* standard.”

37. I have been informed that under the *Phillips* standard, claim terms are given their plain and ordinary meaning as understood by a person of ordinary skill in the art at the time of the invention in light of the claim language and the patent specification.

38. I have been informed that embodiments described in the specification are generally encompassed by the claims.

39. I have been informed that the patentee can serve as their own lexicographer. As such, if a claim term is provided with a specific definition in the specification, then I should interpret that claim term in light of the particular definition provided by the patentee.

B. Obviousness

40. I have been informed and understand that a patent claim can be considered to have been obvious to a person of ordinary skill in the art at the time the application was filed. This means that, even if all of the requirements of a claim are not found in a single prior art reference, the claim is not patentable if the

differences between the subject matter in the prior art and the subject matter in the claim would have been obvious to a person of ordinary skill in the art at the time the application was filed.

41. I have been informed and understand that a determination of whether a claim would have been obvious should be based upon several factors, including, among others:

- the level of ordinary skill in the art at the time the application was filed;
- the scope and content of the prior art;
- what differences, if any, existed between the claimed invention and the prior art; and
- any relevant objective considerations of non-obviousness.

42. I have been informed and understand that the teachings of two or more references may be combined in the same way as disclosed in the claims, if such a combination would have been obvious to one having ordinary skill in the art.

43. In determining whether a combination based on multiple references would have been obvious, it is appropriate to consider, among other factors:

- whether the teachings of the prior art references disclose known concepts combined in familiar ways, which, when combined, would yield predictable results;
- whether a person of ordinary skill in the art would have implemented a predictable variation, and would have seen the benefit of doing so;
- whether the claimed elements represent one of a limited number of known design choices, and would have a reasonable expectation of success by those skilled in the art;
- whether a person of ordinary skill would have recognized a reason to combine known elements in the manner described in the claim;

- whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent; and
- whether the claim applies a known technique that had been used to improve a similar device or method in a similar way.

44. I understand that one of ordinary skill in the art has ordinary creativity and is not an automaton.

45. I understand that in considering obviousness, it is important not to determine obviousness using the benefit of hindsight derived from the patent being considered.

C. Ordinary Skill in the Art

46. I understand that, in the context of an invalidity analysis, a person having ordinary skill in the art (“POSITA”) is a hypothetical person who looks to prior art at the time of the invention.

47. I further understand that the factors that may be considered in determining the level of ordinary skill in the art include: (1) the problems encountered in the art; (2) the prior art solutions to the problems encountered in the art; (3) the rapidity of innovation; (4) the sophistication of the technology; and (5) the education level of active workers in the field. I understand that these factors need not all be taken into account for the analysis and that one or more of these factors may control.

V. Technology Overview

48. In this section, I provide a brief overview of the background art as understood by a POSITA as of the earliest possible priority date of the '194 Patent, *i.e.*, October 25, 2004.

A. General Computer Operations

49. Computers are hardware and software devices that execute instructions. The hardware often includes a central processing unit (“CPU”) or other microprocessors, typically called “processors,” required volatile memory (*e.g.*, RAM (random-access memory), or DRAM (dynamic random-access memory)), non-volatile (persistent) storage (*e.g.*, an internal hard disk or flash drive), and input/output (“I/O”) devices (*e.g.*, network interface card, mouse, keyboard, display). RAM, hard disks, and flash drives are considered, generally, “storage” or “memory” devices because they can store and hold bits of information (whether persistently or not). Computers can have multiple storage devices. Processors are coupled to main memory using a memory bus and coupled to peripherals such as hard disks and network interface cards using one or more I/O busses.

50. Memory and storage devices have considerable differences in their speeds, capacities, and costs. Non-volatile storage devices such as hard disks have the largest capacity, are least expensive per gigabyte, and are slowest (typically operating at millisecond speeds). Volatile DRAM devices are smaller, more

expensive, and faster than hard disks (typically operating at microsecond speeds). A CPU's own memory caches and operations are the smallest, most expensive, and fastest (typically operating at nanosecond speeds). Overall, CPUs are about 1,000× faster than DRAM, which in turn is about 1,000× faster than hard disks.

51. Computer programs are often written in a high-level human-readable language (*e.g.*, C, C++, Java, Perl, PHP, Python), then translated using a compiler or script processor to machine instructions understood by the CPU (*e.g.*, Intel or AMD processor). Machine instructions—generally called “software”—are stored in files on persistent media (*e.g.*, HDD or ROM), loaded into DRAM, and then loaded into the CPU where they can be executed. All modern computers operate in this manner, regardless of their purpose: a small laptop, a server in a data center, a mainframe computer, a personal workstation, a proxy device, a gateway, a firewall, a router, an appliance, a virtual machine, a handheld or mobile device, a smart card, an SD card, etc. Computer systems may run multiple software programs, from background servers and daemons to user applications.

52. Software itself may be distributed or located in a single location, use various modules or units such as libraries and other components. In general, “software” is “[c]omputer programs; instructions that make hardware work. Two main types of software are system software (operating systems), which controls the workings of the computer, and applications, such as word processing programs,

spreadsheets, and databases, which perform the tasks for which people use computers. Two additional categories, which are neither system nor application software but contain elements of both, are network software, which enables groups of computers to communicate, and language software, which provides programmers with the tools they need to write programs. In addition to these task-based categories, several types of software are described based on their method of distribution. These include packaged software (canned programs), sold primarily through retail outlets; freeware and public domain software, which are distributed free of charge; shareware, which is also distributed free of charge, although users are requested to pay a small registration fee for continued use of the program; and vaporware, software that is announced by a company or individuals but either never makes it to market or is very late. *See also* application, canned software, freeware, network software, operating system, shareware, system software, vaporware. *Compare* firmware, hardware, liveware.” EX1011 (Microsoft Computer Dictionary), 489.

53. Users (including administrators) can login to one or more computers (*e.g.*, desktop, laptop, server) and execute one or more applications (*e.g.*, word processing, web browsing, backup tool). Some computers offer external interfaces and APIs to communicate with certain devices and especially peripherals. For

example, both a USB storage device as well as a printer use a well-defined “protocol” to allow users to communicate with them through the main computer.

54. All computers include internal volatile memory (*e.g.*, RAM). A “primary storage” (or “internal memory”) is “Random access memory (RAM); the main general-purpose storage region to which the microprocessor has direct access. A computer’s other storage options, such as disks and tape, are called secondary storage or (sometimes) backing storage.” EX1011 (Microsoft Computer Dictionary), 281, 419. A “secondary storage” is “[a]ny data storage medium other than a computer’s random access memory (RAM)—typically tape or disk. *Compare* primary storage.” EX1011 (Microsoft Computer Dictionary), 469.

55. A “file server” is “[a] file-storage device on a local area network that is accessible to all users on the network. Unlike a disk server, which appears to the user as a remote disk drive, a file server is a sophisticated device that not only stores files but manages them and maintains order as network users request files and make changes to them. To deal with the tasks of handling multiple—sometimes simultaneous—requests for files, a file server contains a processor and controlling software as well as a disk drive for storage. On local area networks, a file server is often a computer with a large hard disk that is dedicated only to the task of managing shared files. *Compare* disk server.” EX1011 (Microsoft Computer Dictionary), 212-213.

56. File servers can transmit audio/video files. A “video server” is “[a] server designed to deliver digital video-on-demand and other broadband interactive services to the public over a wide area network. EX1011 (Microsoft Computer Dictionary), 552.

57. There are many audio/video file formats, comprising a sequence of segments or frames. For example, the “.mpeg” format designates “[t]he file extension that identifies video and sound files compressed in the MPEG format specified by the Moving Pictures Experts Group. *See also* MPEG.” EX1011 (Microsoft Computer Dictionary), 350. “MPEG-2” is “[a]n extension of the MPEG-1 standard designed for broadcast television, including HDTV. MPEG-2 defines a higher bandwidth of up to 40 Mbps, five audio channels, a wider range of frame sizes, and interlaced video. *See also* HDTV, MPEG (definition 1). *Compare* MPEG-1, MPEG-3, MPEG-4.” EX1011 (Microsoft Computer Dictionary), 350.

58. Similarly, an “.avi” is “[t]he file extension that identifies an audiovisual interleaved data file in the Microsoft RIFF format.” EX1011 (Microsoft Computer Dictionary), 46. An “AVI” denotes the “[a]cronym for Audio Video Interleaved. A Windows multimedia file format for sound and moving pictures that uses the Microsoft RIFF (Resource Interchange File Format) specification.” EX1011 (Microsoft Computer Dictionary), 46.

B. Networking

59. A Local Area Network (LAN) is typically a network at a local site where network latencies are short and physical distances between computers are small. A LAN and its computers are often protected from the rest of the Internet using a firewall. A LAN is “Acronym for **l**ocal **a**rea **n**etwork. A group of computers and other devices dispersed over a relatively limited area and connected by a communications link that enables any device to interact with any other on the network. LANs commonly include PCs and shared resources such as laser printers and large hard disks. The devices on a LAN are known as nodes, and the nodes are connected by cables through which messages are transmitted. *See also* baseband network, broadband network, bus network, collision detection, communications protocol, contention, CSMA/CD, network, peer-to-peer architecture, ring network, star network. *Compare* WAN.”). EX1011 (Microsoft Computer Dictionary), 304, 315.

60. Another form of a local area network is one that operates inside a single computer. A “localhost” is “[t]he name that is used to represent the same computer on which a TCP/IP message originates. An IP packet sent to localhost has the IP address 127.0.0.1 and does not actually go out to the Internet. *See also* IP address, packet (definition 1), TCP/IP.” EX1011 (Microsoft Computer Dictionary), 316. Often, when users/processes want to communicate with other users/processes, using

TCP/IP communications on the same machine, they can use the localhost address: it is considered a trusted, high-speed internal network.

61. Conversely, a Wide-Area Network (WAN) refers to computers across the entire Internet, which are often distant from each other, exhibit longer latencies, and afford little protection. A “wide area network” is “[a] geographically widespread network, one that relies on communications capabilities to link the various network segments. A WAN can be one large network, or it can consist of a number of linked LANs (local area networks).” EX1011 (Microsoft Computer Dictionary), 561, 566.

62. A firewall typically intercepts all traffic between computers in a LAN and the WAN, monitors and controls access, and can filter out undesired traffic, and thus acts as a proxy device. Often, LAN computers are far more trusted than WAN computers; therefore, a firewall can be said to separate the trusted network (LAN or localhost) from the untrusted network (WAN). EX1011 (Microsoft Computer Dictionary), 214-215 (“**firewall** *n.* A security system intended to protect an organization’s network against external threats, such as hackers, coming from another network, such as the Internet. Usually a combination of hardware and software, a firewall prevents computers in the organization’s network from communicating directly with computers external to the network and vice versa. Instead, all communication is routed through a proxy server outside of the

organization's network, and the proxy server decides whether it is safe to let a particular message or file pass through to the organization's network. *See also* proxy server.”).

63. A server is a computer or software that provides a service to other computers, called clients. There are many servers on the Internet, for example, Web servers and Mail servers. EX1011 (Microsoft Computer Dictionary), 474 (“**server** *n.* 1. On a local area network (LAN), a computer running administrative software that controls access to the network and its resources, such as printers and disk drives, and provides resources to computers functioning as workstations on the network. 2. On the Internet or other network, a computer or program that responds to commands from a client. For example, a file server may contain an archive of data or program files; when a client submits a request for a file, the server transfers a copy of the file to the client. *See also* client/server architecture. *Compare* client (definition 3).”

64. A client is typically an end-point computer or software, sometimes used by a user, that communicates across LAN and WAN networks; a client may contact servers to request some service. A user's desktop, laptop, and handheld device are examples of clients. (Note that a computer can be both a client and a server, a form of a proxy.)

65. EX1011 (Microsoft Computer Dictionary), 102 (“**client** *n.* [] 2. A process, such as a program or task, that requests a service provided by another

program—for example, a word processor that calls on a sort routine built into another program. The client process uses the requested service without having to ‘know’ any working details about the other program or the service itself. *Compare* child (definition 1), descendant (definition 2). 3. On a local area network or the Internet, a computer that accesses shared network resources provided by another computer (called a *server*). *See also* client/server architecture, server.”).

66. A client/server architecture describes how client computers interact with server computers. EX1011 (Microsoft Computer Dictionary), 102 (“**client/server architecture** *n.* An arrangement used on LANs (local area networks) that makes use of distributed intelligence to treat both the server and the individual workstations as intelligent, programmable devices, thus exploiting the full computing power of each. This is done by splitting the processing of an application between two distinct components: a ‘front-end’ client and a ‘back-end’ server. The client component is a complete, stand-alone personal computer (not a ‘dumb’ terminal), and it offers the user its full range of power and features for running applications. The server component can be a personal computer, a minicomputer, or a mainframe that provides the traditional strengths offered by minicomputers and mainframes in a time-sharing environment: data management, information sharing between clients, and sophisticated network administration and security features. The client and server machines work together to accomplish the processing of the

application being used. Not only does this increase the processing power available over older architectures but it also uses that power more efficiently. The client portion of the application is typically optimized for user interaction, whereas the server portion provides the centralized, multiuser functionality. *See also* distributed intelligence.”).

1. Networked and Distributed Storage Systems

67. File systems can appear as local volumes (*e.g.*, the C:\ drive in Windows), but they can also be networked, where the physical storage resides on a remote file server. Network file systems allow files and volumes to be shared and accessible from multiple networked computers. EX1011 (Microsoft Computer Dictionary), 363 (“**Network File System** *n.* *See* NFS.”), 365 (“**NFS** *n.* Acronym for **Network File System**. A distributed file system that allows users to access remote files and directories on a network as if they were local. NFS is compatible with Microsoft Windows and UNIX-based systems, including Linux and Mac OS X.”).

68. NFS is an example of a network-attached storage, used to hold, transmit, and stream any files. A “NAS” is “[a]cronym for **Network-Attached Storage**. A platform-independent storage appliance connected to a network. NAS uses a storage unit with a built-in server that can communicate with clients over a

network. NAS devices are popular for ease of maintenance, manageability, and scalability. *Compare SAN.*” EX1011 (Microsoft Computer Dictionary), 357, 362.

69. For larger, more demanding network storage systems, a storage-area network is used. A “storage area network” or “SAN” is “[a] high-speed network that provides a direct connection between servers and storage, including shared storage, clusters, and disaster-recovery devices. A storage area network, or SAN, includes components such as hubs and routers that are also used in local area networks (LANs), but it differs in being something of a ‘subnetwork’ dedicated to providing a high-speed connection between storage elements and servers. Most SANs rely on fiber-channel connections that deliver speeds up to 1000 Mbps and can support up to 128 devices. SANs are implemented to provide the scalability, speed, and manageability required in environments that demand high data availability. *Acronym: SAN. Also called: system area network.*” EX1011 (Microsoft Computer Dictionary), 463, 498-499.

70. A NAS, NFS, or SAN storage can be used by any computer over a network. Such network storage is designed for sharing, or for multiple computers to access the same network storage concurrently. However, it is also possible for such network storage or portion of it to be dedicated for user by a single computer. In that case, such network storage acts similarly to a dedicated internal storage such as an internal hard disk.

71. When the amount of data to access grew large enough, and especially if the data had to be concurrently accessed from multiple users/applications, storage researchers and companies leveraged the principles of distributed systems to improve the performance, resiliency, and fault-tolerance of the storage system.

72. A single computer's performance is limited by the physical maximum speeds of its network, processor, memory, and I/O devices. A distributed system, however, improves overall performance by using multiple computers concurrently to process many requests, thus increasing the aggregate throughput manyfold.

73. A single computer also represents a single point of failure: if that one computer is down, off, or broken into, then the service it offers is unavailable. Even regular networks can fail and packets get lost. A distributed system, conversely, improves resiliency and security by ensuring that the failure of any one (or even a few) components do not affect the overall service provided. This principle is called "fault tolerance." EX1011 (Microsoft Computer Dictionary), 207 ("**fault tolerance** *n.* The ability of a computer or an operating system to respond to a catastrophic event or fault, such as a power outage or a hardware failure, in a way that ensures that no data is lost and any work in progress is not corrupted. This can be accomplished with a battery-backed power supply, backup hardware, provisions in the operating system, or any combination of these. In a fault-tolerant network, the system has the ability either to continue the system's operation without loss of data

or to shut the system down and restart it, recovering all processing that was in progress when the fault occurred.”).

74. NFS is but one example of a networked or distributed file system. Many complex software systems have been designed to operate in a distributed manner, where they are more resilient to failures and scale better than software running on a single node.

75. Generally, a “distributed network” is “[a] network in which processing, storage, and other functions are handled by separate units (nodes) rather than by a single main computer.” EX1011 (Microsoft Computer Dictionary), 168.

76. The “Distributed Computing Environment” is “[a] set of standards from the Open Group (formerly the Open Software Foundation) for development of distributed applications that can operate on more than one platform. *Acronym*: DCE. *See also* distributed processing.” EX1011 (Microsoft Computer Dictionary), 167.

77. “[D]istributed processing” is “[a] form of information processing in which work is performed by separate computers linked through a communications network. Distributed processing is usually categorized as either plain distributed processing or true distributed processing. Plain distributed processing shares the workload among computers that can communicate with one another. True distributed processing has separate computers perform different tasks in such a way that their combined work can contribute to a larger goal. The latter type of

processing requires a highly structured environment that allows hardware and software to communicate, share resources, and exchange information freely.”

EX1011 (Microsoft Computer Dictionary), 168.

78. Finally, a “distributed file system” is “[a] file management system in which files may be located on multiple computers connected over a local or wide area network.” EX1011 (Microsoft Computer Dictionary), 168.

C. Data Security and Privacy

79. All modern systems include measures of security to protect their data from unauthorized access. EX1011 (Microsoft Computer Dictionary), 212 (“**file protection** *n.* A process or device by which the existence and integrity of a file are maintained. Methods of file protection range from allowing read-only access and assigning passwords to covering the write-protect notch on a disk and locking away floppy disks holding sensitive files.”).

80. The POSIX standard, first started in the late 1980s, codifies the access application program interfaces (APIs) that operating systems export to user applications. EX1011 (Microsoft Computer Dictionary), 414 (“**POSIX** *n.* Acronym for **Portable Operating System Interface for UNIX**. An Institute of Electrical and Electronics Engineers (IEEE) standard that defines a set of operating-system services. Programs that adhere to the POSIX standard can be easily ported from one

system to another. POSIX was based on UNIX system services, but it was created in a way that allows it to be implemented by other operating systems.”).

81. A simple data protection scheme defines whether users can only read files, only write files, or have both read and write access. EX1011 (Microsoft Computer Dictionary), 398 (“**permission** *n.* In a networked or multiuser computer environment, the ability of a particular user to access a particular resource by means of his or her user account. Permissions are granted by the system administrator or other authorized person. Several levels of access can be given: read only, read and write (view and change), or read, write, and delete. *Also called:* Access permission.”).

82. In the basic POSIX model, for example, a file has three sets of permissions (read, write, or execute) for three classes of users (owner, group member, and everyone else).

83. More sophisticated schemes generalize these simpler permission models to allow for lists of users, lists of groups, as well as logical operations on lists (*e.g.*, negation, conjunction, and disjunction). These are called “access control lists,” or ACLs. EX1011 (Microsoft Computer Dictionary), 14 (“**access control** *n.* The mechanisms for limiting access to certain items of information or to certain controls based on users’ identities and their membership in various predefined groups. Access control is typically used by system administrators for controlling

user access to network resources, such as servers, directories, and files. *See also* access privileges, system administrator.”), 14 (“**access control list** *n.* A list associated with a file that contains information about which users or groups have permission to access or modify the file. *Acronym:* ACL.”). ACLs existed in Unix systems since at least the 1990s (and I personally used them, see §I above).

84. Another common technique, useful to protect user’s privacy, is “encryption,” or “[t]he process of encoding data to prevent unauthorized access, especially during transmission. Encryption is usually based on one or more keys, or codes, that are essential for decoding, or returning the data to readable form. The U.S. National Bureau of Standards created a complex encryption standard, Data Encryption Standard (DES), which is based on a 56-bit variable that provides for more than 70 quadrillion unique keys to encrypt documents. *See also* DES.” EX1011(Microsoft Computer Dictionary), 142, 192.

85. To “encrypt” is “[t]o encode (scramble) information in such a way that it is unreadable to all but those individuals possessing the key to the code. Encrypted information is known as cipher text. *Also called:* encipher, encode.” EX1011 (Microsoft Computer Dictionary), 192.

86. A “block cipher” is “[a] private key encryption method that encrypts data in blocks of a fixed size (usually 64 bits). The encrypted data block contains

the same number of bits as the original. *See also* encryption, private key.” EX1011 (Microsoft Computer Dictionary), 65.

87. “DES” is an “[a]cronym for Data Encryption Standard. A specification for encryption of computer data developed by IBM and adopted by the U.S. government as a standard in 1976. DES uses a 56-bit key. *See also* encryption, key (definition 3).” EX1011 (Microsoft Computer Dictionary), 142, 153.

88. Most encryption schemes use an “encryption key,” “[a] sequence of data that is used to encrypt other data and that, consequently, must be used for the data’s decryption. *See also* decryption, encryption.” EX1011 (Microsoft Computer Dictionary), 192.

89. Some encryption schemes do not use a key, such as “ROT13 encryption,” “[a] simple encryption method in which each letter is replaced with the letter of the alphabet 13 letters after the original letter, so that A is replaced by N, and so forth; N, in turn, is replaced by A, and Z is replaced by M. ROT13 encryption is not used to protect messages against unauthorized readers; rather, it is used in newsgroups to encode messages that a user may not want to read, such as sexual jokes or spoilers. Some newsreaders can automatically perform ROT13 encryption and decryption at the touch of a key.” EX1011 (Microsoft Computer Dictionary), 458.

90. Even schemes that use a key require that the key be protected to maintain privacy. The principle of “privacy” is “[t]he concept that a user’s data, such as stored files and e-mail, is not to be examined by anyone else without that user’s permission. A right to privacy is not generally recognized on the Internet. Federal law protects only e-mail in transit or in temporary storage, and only against access by Federal agencies. Employers often claim a right to inspect any data on their systems. To obtain privacy, the user must take active measures such as encryption. *See also* encryption, PGP, Privacy Enhanced Mail. *Compare* security.” EX1011 (Microsoft Computer Dictionary), 422.

91. Indeed, in public-key cryptography, the “private key” is “[o]ne of two keys in public key encryption. The user keeps the private key secret and uses it to encrypt digital signatures and to decrypt received messages. *See also* public key encryption. *Compare* public key.” EX1011 (Microsoft Computer Dictionary), 422.

92. There are many forms of verification that can be applied to data files. One example is to verify the integrity of data using cryptographic hashing algorithms such as SHA. “SHA” is an “[a]cronym for **S**ecure **H**ash **A**lgorithm. A technique that computes a 160-bit condensed representation of a message or data file, called a *message digest*. The SHA is used by the sender and the receiver of a message in computing and verifying a digital signature, for security purposes. *See also* algorithm, digital signature.” EX1011 (Microsoft Computer Dictionary), 477. The

term “data integrity” represents “[t]he accuracy of data and its conformity to its expected value, especially after being transmitted or processed.” EX1011 (Microsoft Computer Dictionary), 143.

93. Users can also verify the authenticity of files’ sources using digital signatures. A “digital signature” is “[a] security mechanism used on the Internet that relies on two keys, one public and one private, that are used to encrypt messages before transmission and to decrypt them on receipt.” EX1011 (Microsoft Computer Dictionary), 159.

94. Digital signatures often combine cryptographic techniques and hashing algorithms to enable bi-directional authentication and secure communications. “Public key cryptography” or “public key encryption” is “[a]n asymmetric scheme that uses a pair of keys for encryption: the public key encrypts data, and a corresponding secret key decrypts it. For digital signatures, the process is reversed: the sender uses the secret key to create a unique electronic number that can be read by anyone possessing the corresponding public key, which verifies that the message is truly from the sender. *See also* private key, public key.” EX1011 (Microsoft Computer Dictionary), 429.

95. In fact, digital signatures were so common that in 2000 the U.S. Government published a standard describing their use. The “Digital Signature Standard” is “[a] public key cryptographic standard issued in 1994 by the United

States National Institute of Standards and Technology (NIST) to authenticate electronic documents. The DSS uses a Digital Signature Algorithm (DSA) to generate and verify digital signatures based on a public key, which is not secret, and a private key, which is known or held only by the person generating the signature. A digital signature serves to authenticate both the identity of the signer and the integrity of the transmitted information. *Acronym: DSS. See also public key encryption.* EX1011 (Microsoft Computer Dictionary), 159-160.

96. Data protection techniques such as encryption often use XOR operations. The “exclusive OR” or “XOR” is “[a] Boolean operation that yields ‘true’ if and only if one of its operands is true and the other is false. See the table. *Acronym: EOR. Also called: XOR. See also Boolean operator, truth table. Compare AND, OR.* EX1011 (Microsoft Computer Dictionary), 199-200, 579 (including table below).

Table E.1 Exclusive OR.

<i>a</i>	<i>b</i>	<i>a XOR b</i>
0	0	0
0	1	1
1	0	1
1	1	0

97. Indeed “XOR encryption” is “[s]hort for Exclusive-OR encryption. A simple encryption scheme using the ‘exclusive-or’ concept, in which a decision is based on only one of two conditions being met. Using a provided key, XOR

encryption performs an exclusive-or process on each byte of data to be encrypted. Because XOR encryption is not a strong security tool used alone, it is typically used as an additional level of security for Internet transmission of sensitive information.” EX1011 (Microsoft Computer Dictionary), 579.

1. Hash Functions and Their Uses

98. Hashing algorithms are mathematical functions that take as input (typically) a long piece of data and produce a shorter output number that can (almost always) uniquely identify the input data. Depending on the specific context, hash functions are sometimes called checksums, fingerprints, message digests, or digital signature functions.

99. Hash functions are one-way: it is typically considered cryptographically hard to produce the original input data from the hash itself (*e.g.*, invert it). *See* EX1011 (Microsoft Computer Dictionary), 247-248 (“**hash**² *vb.* To be mapped to a numerical value by a transformation known as a hashing function. Hashing is used to convert an identifier or key, meaningful to a user, into a value for the location of the corresponding data in a structure, such as a table. []”), 477 (“**SHA** *n.* Acronym for **Secure Hash Algorithm**. A technique that computes a 160-bit condensed representation of a message or data file, called a *message digest*. The SHA is used by the sender and the receiver of a message in computing and verifying a digital signature, for security purposes. *See also* algorithm, digital signature.”),

133 (“**CRC** *n.* Acronym for cyclical (or cyclic) redundancy check. A procedure used in checking for errors in data transmission. CRC error checking uses a complex calculation to generate a number based on the data transmitted. The sending device performs the calculation before transmission and includes it in the packet that it sends to the receiving device. The receiving device repeats the same calculation after transmission. If both devices obtain the same result, it is assumed that the transmission was error free. The procedure is known as a redundancy check because each transmission includes not only data but extra (redundant) error-checking values. Communications protocols such as XMODEM and Kermit use cyclical redundancy checking.”).

100. Hash functions have been used for many years in various settings from security, cryptography, integrity, data deduplication, and others. For example, the Tripwire system I personally used in the early 1990’s hashes files and their meta-data and compares those hashes against the files later on; if a mismatch is detected, it means that the file was changed, which could indicate that an intruder broke into the system and installed malware such as Trojans. *See* §I above (Background and Qualifications).

101. In another example, vendors hash known malicious binaries and distribute those to customers. The hashes are small, can easily be distributed, and also do not identify the actual binary (because customers do not like exposing their

own files even to their own security vendors). On the customer's computers, existing and incoming binaries can be compared: any matches found would indicate that the customer has or received malware. This malware can be cleaned (if possible), quarantined (*e.g.*, saved in a special folder), deleted, and/or reported.

102. Hashes are useful beyond security applications. Storage systems and file systems can store numerous files. Many of those files, or parts thereof, can be duplicates (*e.g.*, due to users copying files repeatedly, backup procedures, and more). Storage systems therefore have employed data de-duplication techniques to remove these duplicates: this can save a lot of space and reduce costs. Because files' names do not have to be unique and can change over time, deduplication systems often use a different identifier for the file's actual data—unique hash computed from the actual data (or parts thereof). In that way, the hash can uniquely identify the file's data regardless of the file's name or location.

2. Secret Splitting and Sharing

103. Secret splitting is a technique to break up a piece of data and distribute it such that no one person has the entire secret.

104. “Imagine that you've invented a new, extra gooey, extra sweet, cream filling or a burger sauce” and that “you have to keep it secret. You could tell only your most trusted employees the exact mixture of ingredients, but what if one of them defects to the competition? There goes the secret....” EX1012 (Schneier), 70.

105. “This calls for **secret splitting**. There are ways to take a message and divide it up into pieces [551]. Each piece by itself means nothing, but put them together and the message appears. If the message is the recipe and each employee has a piece, then only together can they make the sauce. If any employee resigns with his single piece of the recipe, his information is useless by itself.” EX1012 (Schneier), 70.

106. Schneier describes a simple secret-splitting scheme using XOR among several people: Alice, Bob, Carol, Dave, and Trent. EX1012 (Schneier), 70-71. Schneier then outlines problems with this simple scheme.

107. “However, this protocol has a problem: If any of the pieces gets lost and Trent isn’t around, so does the message. If Carol, who has a piece of the sauce recipe, goes to work for the competition and takes her piece with her, the rest of them are out of luck. She can’t reproduce the recipe, but neither can Alice, Bob, and Dave working together. Her piece is as critical to the message as every other piece combined. All Alice, Bob, or Dave know is the length of the message-nothing more.” EX1012 (Schneier), 71.

108. Schneier describes an improve scheme called “Secret Sharing.” Suppose “[y]ou’re setting up a launch program for a nuclear missile. You want to make sure that no single raving lunatic can initiate a launch. You want to make sure

that no two raving lunatics can initiate a launch. You want at least three out of five officers to be raving lunatics before you allow a launch.” EX1012 (Schneier), 71.

109. “This is easy to solve. [] Give each of the five officers a key and require that at least three officers stick their keys in the proper slots before you’ll allow them to blow up whomever we’re blowing up this week.” EX1012 (Schneier), 71.

110. “We can get even more complicated. Maybe the general and two colonels are authorized to launch the missile, but if the general is busy playing golf then five colonels are required to initiate a launch. Make the launch controller so that it requires five keys. Give the general three keys and the colonels one each. The general together with any two colonels can launch the missile; so can the five colonels. However, a general and one colonel cannot; neither can four colonels.” EX1012 (Schneier), 71.

111. “A more complicated sharing scheme, called a **threshold scheme**, can do all of this and more-mathematically. At its simplest level, you can take any message (a secret recipe, launch codes, your laundry list, etc.) and divide it into n pieces, called **shadows** or shares, such that any m of them can be used to reconstruct the message. More precisely, this is called an **(m,n) -threshold scheme**.” EX1012 (Schneier), 71.

112. “With a (3,4)-threshold scheme, Trent can divide his secret sauce recipe among Alice, Bob, Carol, and Dave, such that any three of them can put their

shadows together and reconstruct the message. If Carol is on vacation, Alice, Bob, and Dave can do it. If Bob gets run over by a bus, Alice, Carol, and Dave can do it. However, if Bob gets run over by a bus while Carol is on vacation, Alice and Dave can't reconstruct the message by themselves." EX1012 (Schneier), 71.

113. "General threshold schemes are even more versatile. Any sharing scenario you can imagine can be modeled." EX1012 (Schneier), 72.

114. The need for secret-splitting was so great that "[t]his idea was invented independently by Adi Shamir [1414] and George Blakley [182] and studied extensively by Gus Simmons [1466]. Several different algorithms are discussed in Section 23.2." EX1012 (Schneier), 72.

D. Recovering from Data Loss

115. Storage systems are supposed to be reliable: ideally data should never be lost or corrupted. In reality, however, the risk of data loss depends on many factors including hardware, software, configurations, workloads, and even human errors.

116. Hardware can fail. Memory bits can be corrupted by magnetic fields and even cosmic rays. Temperatures affect the stability of bits in volatile memory. Electro-magnetic storage media are susceptible to heat, humidity, vibrations (even loud music played in close proximity to a storage device), magnetism, the quality of power feeds, and more. Mechanical parts inside disks wear out over time and with

use. Poor quality electronic components anywhere on a computer system can ultimately affect data reliability elsewhere.

117. Software may contain bugs. Such bugs can result in returning the wrong data to users, writing the wrong data in the wrong place, writing only some of the data, or even not writing user-requested data at all (aka “lost writes”). Bugs can reside everywhere: inside the firmware of any component, in any of the operating system components, in libraries, and applications.

118. User applications and workloads exercise the system in different ways, whether directly or indirectly. Such access can trigger bugs and affect how quickly the hardware wears out. Generally, higher levels of I/O activity tend to increase wear-out more quickly.

119. Finally, disasters can destroy data on storage hardware, from man-made disasters (*e.g.*, wars, terrorism acts) to natural disasters (*e.g.*, earthquakes, fires, or floods). It should be noted that human errors also play a crucial role in data reliability: a significant portion of errors that result in lost data can be attributed to misconfigured systems and human errors.

120. To reduce the risk of data loss, users can use various techniques to survive one or more failures (*i.e.*, recover from disasters). The general idea is to provide extra levels of data redundancy: the more levels of redundancy that are employed, the higher the probability of recovering such data in the event of a

disaster. That is partially what RAID was designed for. Indeed, RAID stands is “[a]cronym for **r**edundant **a**rray of **i**ndependent (or **i**nexpensive) **d**isks. A data storage method in which data is distributed across a group of computer disk drives that function as a single storage unit. All the information stored on each of the disks is duplicated on other disks in the array. This redundancy ensures that no information will be lost if one of the disks fails. RAID is generally used on network servers where data accessibility is critical and fault tolerance is required. There are various defined levels of RAID, each offering differing trade-offs among access speed, reliability, and cost. *See also* disk controller, error-correction coding, Hamming code, hard disk, parity bit, server (definition 1).” EX1011 (Microsoft Computer Dictionary), 437.

121. Moreover, RAID level 5 (RAID5) provides one level of redundancy by consuming one of N drives’ capacity to store parity information, distributed across all N drives. If one drive fails, the system is said to operate in “degraded mode.” Then, RAID5 software uses the parity information to reconstruct the lost drive’s data, thus no data appears lost to the user. The cost of operating in degraded mode is that overall storage system performance is lower because RAID5 has to work harder to reconstruct lost data on the fly.

122. In RAID5 systems, it is, therefore, important to replace a failed drive promptly so that the RAID system can reconstruct the redundant data (as discussed

below) and stop operating in degraded mode. If a RAID5 system that is operating in degraded mode loses *another* drive, then *all* data on the storage system is now lost—a catastrophic data loss event (a disaster). These RAID variants provide a safe window of time in which to replace failed drives. Recovering such data now requires restoring from backups, if any.

123. RAID6 offers two levels of redundancy. Here, up to two out of N drives can fail without catastrophic data loss. Other forms of RAID offer three levels of redundancy: up to 3 of N drives can fail without catastrophic data loss.

124. One can design storage systems with any level of redundancy, even up to an extreme where all data is replicated on all N drives (*i.e.*, RAID1 mirroring). RAID1 provides $N - 1$ levels of redundancy but only $1/N^{\text{th}}$ of the overall storage capacity. RAID1 can also improve read performance because data-read requests can be load-balanced across N drives.

125. Conversely, RAID0 “striping” effectively concatenates N drives into a single logical drive whose capacity is the sum of all N drives. RAID0 maximizes storage capacity and can improve performance by distributing writes across all drives. RAID0, however, provides *no* levels of redundancy. RAID0’s risk of data loss is even higher than that of a single drive because with RAID0, if *any* single drive fails, all data is lost.

126. Users can combine RAID levels to try to trade off capacity, performance, and risk of data loss—at the cost of increased complexity. For example, RAID10 is often a combination of RAID0 (mirroring) and RAID1 (striping), attempting to balance the pros and cons of both.

127. Generally, for a given set of drives, the more levels of redundancy, the less overall usable capacity users get, as redundancy information consumes some space. Also, higher levels of redundancy tend to require more computationally complex and intensive techniques, possibly increasing CPU utilization.

128. When a drive fails, it has to be replaced. Users have to purchase a suitable replacement drive or have purchased one ahead of time and placed it on reserve. Once a suitable replacement drive is available, users have to remove the failed drive from the storage system and insert the new drive. It should be noted that a fatal human error that sometimes happens at this stage is when users remove the *wrong* (*i.e.*, “good”) drive in a degraded array—potentially causing catastrophic data loss.

129. Once a replacement drive is inserted, the RAID software has to initialize the new drive with appropriate data and parity information. That process is called *resilvering*. Resilvering can take time and consume significant amount of I/O: data often has to be read from other drives, parities calculated, and then written to the replacement drive. During resilvering time, a degraded array may continue to

serve I/O requests to users and applications. The more user activity there is, the more it interferes with and slows the resilvering process; this can lengthen the time to complete resilvering and hence increase the risk of catastrophic data loss.

130. Until a drive is physically replaced and fully resilvered, the storage system is still running in degraded mode and hence vulnerable to catastrophic data loss should enough additional drives fail. For that reason, users who are more concerned with the risk of data loss would employ higher levels of redundancy and also reserve several drives in the storage system as *hot spares*. Hot-spare drives are automatically brought into service when one drive fails, without requiring time-delayed, manual human intervention.

131. RAID disks are not the only way to implement different disaster recoverability levels. In its simplest form, the ability to recover data after a disaster is a probability function that increases with two factors. First, the more copies that are kept, the higher the chance that one could recover at least one copy of the data. Second, the more copies that are distributed distant from each other (even geographically), the higher the chance that one could recover at least one: it makes no sense to keep source data and all backups in the same physical facility, because a building fire, earthquake, major flood, and even a terrorism act could destroy all of them.

132. The parities that RAID techniques use are examples of a more generalized theory of error correction. An “error-correcting code” (ECC), used in “error-correction coding,” is “[a] method for encoding that allows for detection and correction of errors that occur during transmission. Data is encoded in such a way that transmission errors may be detected and corrected by examination of the encoded data on the receiving end. Most error-correction codes are characterized by the maximum number of errors they can detect and by the maximum number of errors they can correct. Error-correction coding is used by most modems. *Also called:* error-correcting code. *See also* error detection and correction. *Compare* error-detection coding.” EX1011 (Microsoft Computer Dictionary), 196-197.

133. RAID techniques and ECC schemes can be configured to be able to recover an original data with only a subset (or threshold) number of parts available. These techniques are the basis for secret splitting as discussed in §V.C.2 above.

1. Selecting The Fastest-Responding Servers

134. Performance or resource consumption has often been a trade-off with reliability. For example, one could make more mirror copies of a piece of data and distribute them geographically: that would increase reliability of the data and its resilience to failures; however, it would consume more storage resources and run slower as more copies have to be retrieved from different nodes on the network

135. RAID is a form of error-correcting codes. When one uses RAID with multiple storage devices, especially slower hard disks, it is possible that some drives would run much slower than others. In that case, one optimization implemented was to retrieve just a subset of the RAID data blocks, from those drives that have responded fastest—meaning to skip the drives that are the slowest to respond; and, once enough blocks are read, the full data could be reconstructed in memory before returning it to the user or application.

136. For example, Birk recognizes that “[b]y using the redundant information to avoid accessing an overloaded disk drive, the occasional transient imbalance in disk load due to the randomization is partly prevented and, when occurring, can be circumvented.” EX1014 (“Birk”), 13. Birk considers this problem in the context of “High Performance Video Servers”. EX1014 (“Birk”), 13.

137. Birk discusses techniques for “Exploitation of redundancy for load balancing”. EX1014 (“Birk”), 16-17. He notes that, “in 1975, Maxemchuk proposed to partition a message into m packets, compute r redundant ones such that the original message can be reconstructed from any sufficiently large subset of the $m + r$ packets, send those packets over different paths and use the earliest arrivers to reconstruct the message [11]” and that “this substantially reduced delay and provided fault-tolerance.” EX1014 (“Birk”), 16-17. Moreover, “[s]imilar

observations were later made by Rabin [12] and dubbed ‘information dispersal algorithm’.” EX1014 (“Birk”), 17.

138. Birk contemplates that an “approach, cast in terms of disk arrays, would entail submitting requests to all the disk drives holding a given parity group, and using the first ones that arrive to reconstruct the data.” EX1014 (“Birk”), 17.

139. Birk, however, improve on that approach to minimize the number of read requests that have to be issued in the first place. Birk “exploit[s] the redundant chunk if and only if the queue length to one of the disks holding a data chunk is longer than a certain value and the queue to the disk holding the parity chunk is sufficiently shorter. We refer to our approach as *selective exploitation of redundancy*, and apply it in conjunction with randomized layout.” EX1014 (“Birk”), 17-18.

140. Similarly, Chiquoine recognizes that disks in a RAID array may be slow (or degraded). “Individual disks of a RAID storage configuration will occasionally stall or respond slowly to an access request due to disk surface defects and bad block re-vectoring.... During a slow response, the entire RAID configuration may wait while one disk transmits requested data. Thus, a single slowly responding disk can cause a long latency for a read operation from the RAID configuration.” EX1015 (“Chiquoine”), [0004].

141. Chiquoine, like Birk, also considers the problem of slow storage for video servers. “For digital video and cable systems, one slowly responding disk can cause a disaster, because data needs to arrive at a video receiver at a substantially constant rate to keep the receiver’s input buffer full. ... If a slow RAID configuration causes a transmission gap so that the receiver’s input buffer empties[,] a viewer may perceive a noticeable pause in the video being viewed. Defect-free transmission of video requires that such pauses be absent.” EX1015 (“Chiquoine”), [0005].

142. “A reconstructor performs an XOR operation on parity and data to reconstruct the missing data corresponding to the bad or reconstructing drive. Thus, by skipping the bad drive and not waiting for a slow or degraded node or drive to respond before trying to reconstruct data improves the performance of the RAID array. ...” EX1015 (“Chiquoine”), [0010].

143. “Thus, by skipping the bad drive and not waiting for a slow drive to respond before trying to reconstruct data, will improve the performance of the RAID array towards normal performance.” EX1015 (“Chiquoine”), [0038].

144. The idea of reconstructing data by retrieving data from the fastest-responding servers was not limited to RAID but also use in distributed storage systems where data is protected using encryption and redundancy techniques.

145. “OceanStore is a utility infrastructure designed to span the globe and provide continuous access to persistent information. Since this infrastructure is comprised of untrusted servers, data is protected through redundancy and cryptographic techniques. To improve performance, data is allowed to be cached anywhere, anytime. Additionally, monitoring of usage patterns allows adaptation to regional outages and denial of service attacks; monitoring also enhances performance through pro-active movement of data.” EX1017 (“Kubiatowicz”), 1 (Abstract).

146. “To reconstruct archival copies, OceanStore sends out a request keyed off the GUID of the archival versions. Note that we can make use of excess capacity to insulate ourselves from slow servers by requesting more fragments than we absolutely need and reconstructing the data as soon as we have enough fragments. As the request propagates up the location tree (Section 4.3), fragments are discovered and sent to the requester. This search has nice locality properties since closer fragments tend to be discovered first.” EX1017 (“Kubiatowicz”), 8.

VI. Overview Of The '194 Patent

147. The '194 Patent alleges that previous methods for securing data (such as passwords, encryption keys, and biometry) were flawed. *See generally* EX1001 ('194 Patent), 1:22–2:26. The '194 Patent purports to improve systems for “securing data from unauthorized access or use” in contrast to previous authentication methods

well known in the art. *See* EX1001 ('194 Patent), 1:17–18. For example, password management “has proven to be quite costly” and passwords are susceptible to “inappropriate access.” EX1001 ('194 Patent), 1:30–36. Likewise, encryption keys are “highly reliant on the user for security”: Users may store unsecured keys on publicly accessible locations (such as storage connected to the Internet); and users often backup or archive their data, creating multiple key copies which can be used by unwanted parties. EX1001 ('194 Patent), 1:37–2:3. And biometric authentication (such as the user’s fingerprints or speech) often relies on specialized equipment (such as a fingerprint scanner or voice recorder), without which a user cannot access secured systems. *See* EX1001 ('194 Patent), 2:4–26.

148. Figure 1 of the '194 Patent depicts a “cryptographic system 100.” EX1001 ('194 Patent), 8:42–47. As shown, this system includes a “user system 105” and a “vendor system 120,” both of which are connected to a “trust engine 110” through a “communication link 125.” EX1001 ('194 Patent), 8:42–47, FIG. 1 (below). The trust engine 110 delivers “complete cryptographic functionality,” which encompasses operations like encryption and decryption. EX1001 ('194 Patent), 10:4–17.

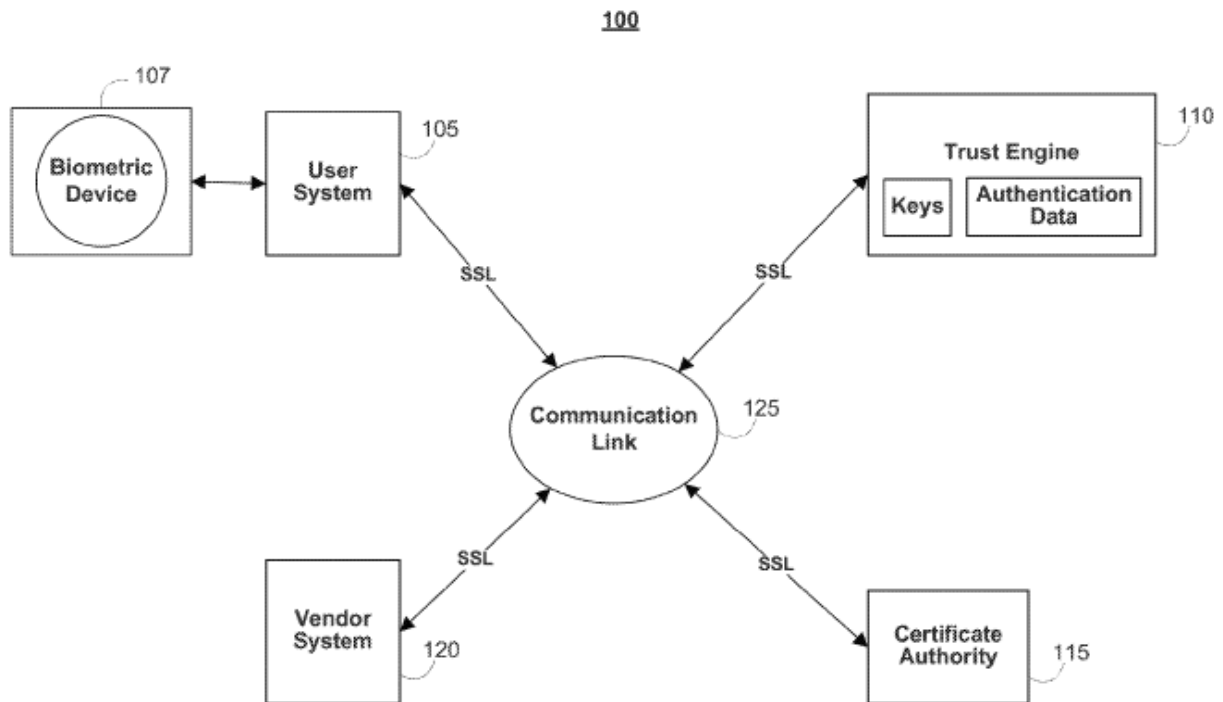


FIG. 1

EX1001 ('194 Patent), FIG. 1.

149. Within the trust engine is a component referred to as a “depository 210,” which “comprises one or more data storage facilities, such as, for example, a directory server, a database server, or the like.” EX1001 ('194 Patent), 12:1–24, 13:20–34, FIG. 2 (below). Additionally, the trust engine 110 contains both an “authentication engine 215” and a “cryptographic engine 220.” EX1001 ('194 Patent), 12:1–24. These components “employ their respective data splitting modules to divide sensitive data into undecipherable portions; they then transmit one or more

undecipherable portions of the sensitive data to a particular data storage facility”
housed within the depository 210. EX1001 ('194 Patent), 18:32–46.

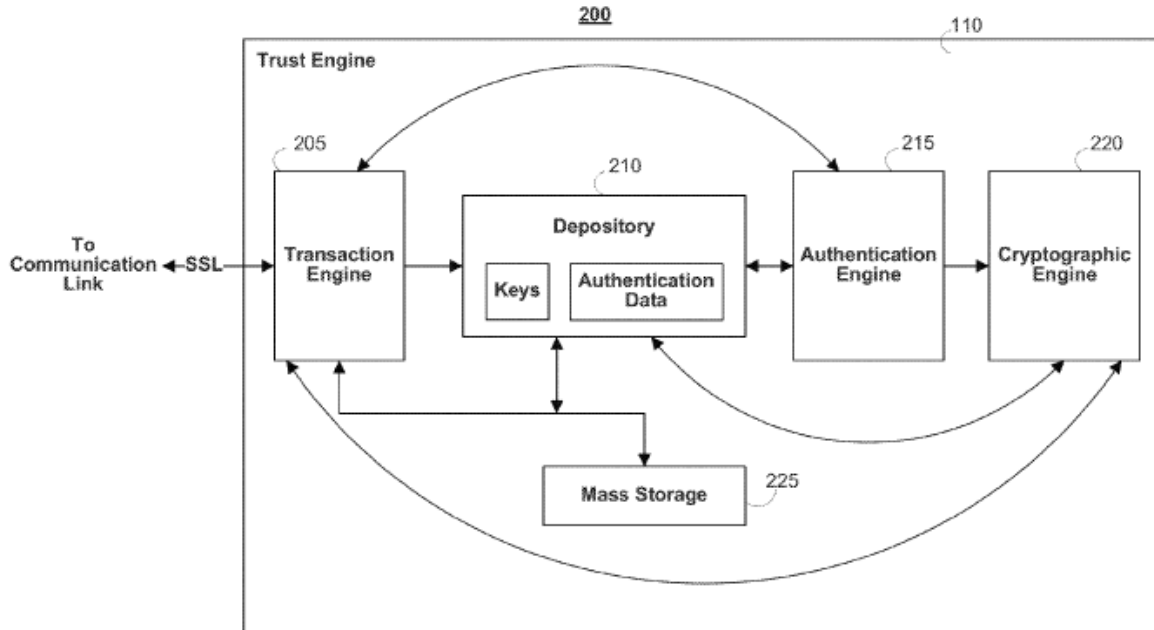


FIG. 2

EX1001 ('194 Patent), FIG. 2.

150. According to the '194 Patent, the “depository system ... advantageously comprises multiple data storage facilities.” EX1001 ('194 Patent), 18:8–31. Figure 7 (below) shows how the “depository 210” within the trust engine 110 protects a data set by creating shares from that data and distributing those shares across several data storage facilities, labeled D1 through D4. EX1001 ('194 Patent), 18:47–19:5.

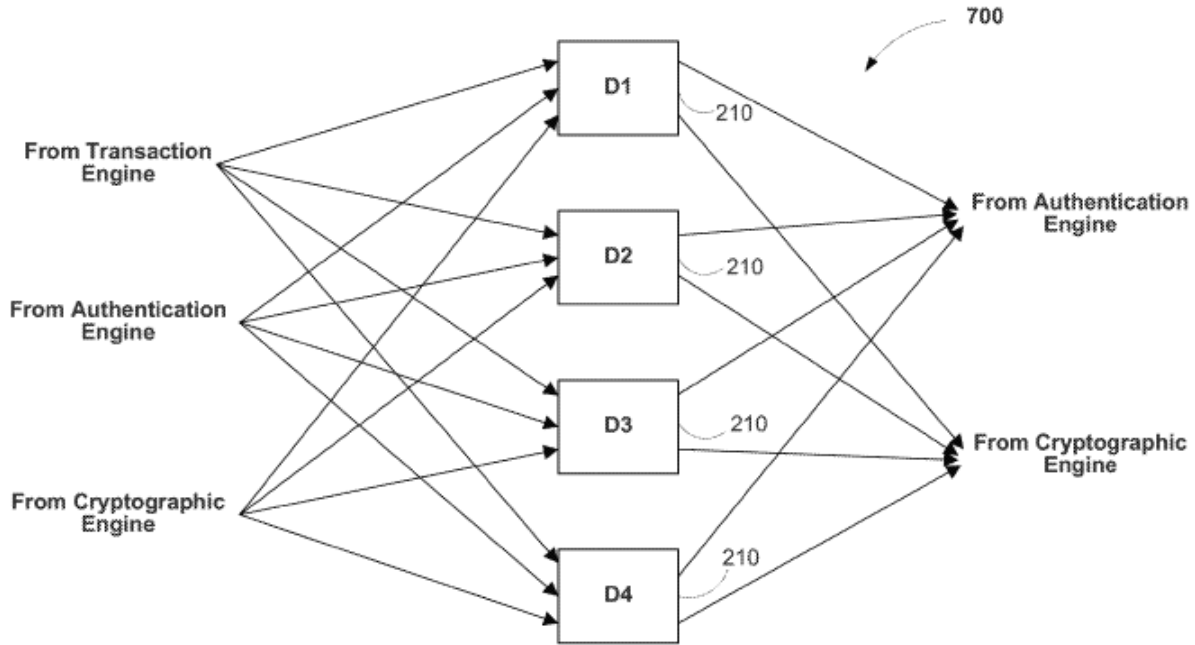


FIG. 7

EX1001 ('194 Patent), FIG. 7.

151. The system illustrated in Figure 7 (above) can carry out a method for securely storing and retrieving data, as outlined in Figure 8 (below). EX1001 ('194 Patent), 19:37–67. As one example, the system may begin by receiving a request to write or store a particular data set. EX1001 ('194 Patent), 18:8–31. It then applies a cryptographic operation to that data set in order to generate several data shares. EX1001 ('194 Patent), 19:16–36. These shares are then distributed in a way that allows the original data set to be reconstructed from any subset that includes at least a minimum number of shares, which is fewer than the total number created. EX1001 ('194 Patent), 20:1–9, 20:38–60. For example, the system may generate two random

values, strings, numbers, or bit sequences identified as “A” and “C.” EX1001 (’194 Patent), 19:16–67. It then uses these in combination with the sensitive data “S” to derive two new values, “B” and “D,” respectively, using operations such as exclusive/or (XOR): $B = A \text{ XOR } S$, $D = C \text{ XOR } S$. EX1001 (’194 Patent), 19:37–67.

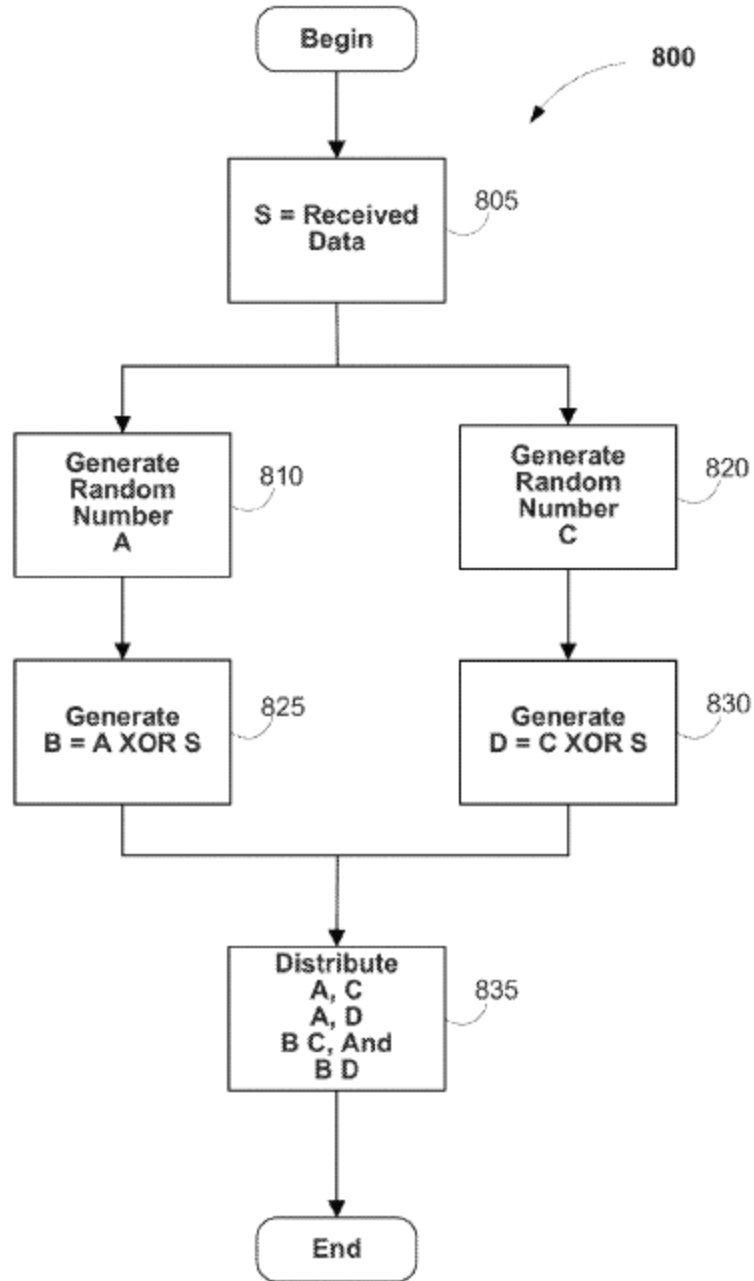


FIG. 8

EX1001 ('194 Patent), FIG. 8.

152. Next, the system creates pairings of A, B, C, and D “such that none of the pairings contain sufficient data, by themselves, to reorganize and decipher the original sensitive data S.” EX1001 (’194 Patent), 19:37–67. For example, it may store the data in paired combinations like AC, AD, BC, and BD. EX1001 (’194 Patent), 19:37–67. These are organized such that any pair provides one element from (A and B) and one from (C and D). EX1001 (’194 Patent), 20:38–60. Each of these pairings is then stored in one of the four storage facilities D1 through D4, as depicted in Figure 7 (above). EX1001 (’194 Patent), 19:37–20:9.

153. To recover the original data set, Dickinson’s system and method obtain the necessary data shares from a group of “fastest-responding” storage devices that together contain at least the minimum number of shares required for reconstruction. Specifically, the system determines which storage devices respond most quickly and selects a subset that can supply the minimum number of shares needed. EX1001 (’194 Patent), 18:8–31. It then retrieves the required shares from those selected devices. EX1001 (’194 Patent), 18:8–31, 20:1–9, 20:38–60. In order to identify which devices are the “fastest-responding,” the system “may broadcast requests to particular data storage facilities based on a wide number of criteria, such as, for example, response time, server loads, maintenance schedules, or the like.” EX1001 (’194 Patent), 18:8–31. Subsequently, the system combines the retrieved shares to reconstruct the original data set. EX1001 (’194 Patent), 20:38–60.

A. Prosecution History

154. The '194 Patent claims priority to U.S. Patent Application Serial No. 13/468,562 (referred to as the "'562 Application"), which was filed on May 10, 2012. EX1006 ('194 File History), 4. This application is a continuation of U.S. Patent Application Serial No. 11/258,839 (referred to as the "'839 Application"), filed on October 25, 2005. EX1007 ('839 App. File History), 63. The '839 Application, in turn, claims the benefit of two U.S. Provisional Applications: Serial No. 60/622,146, filed October 25, 2004, and Serial No. 60/718,185, filed September 16, 2005. EX1007 ('839 App. File History), 240.

155. The claims of the '194 Patent were allowed following three rounds of Office Actions and corresponding amendments. On June 5, 2014, Examiner Lemma issued a Non-Final Rejection, rejecting claims 1–20 under 35 U.S.C. §102(b) on the grounds that they were anticipated by U.S. Patent Application Publication No. 2010/0162003 ("Dodgson"). EX1006 ('194 File History), 771–84.

156. In response, on September 15, 2014, the Patent Owner's predecessor, Security First Corp. ("SFC"), revised the independent claims. One such revision replaced the previously recited limitation, "automatically identifying, at the electronic computing system, a set of fastest-responding storage devices in the set of storage devices," with the new language, "identifying, from the plurality of storage devices a set of fastest-responding storage devices necessary to retrieve the

minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices.” EX1006 (’194 File History), 793–99.

157. SFC also modified the claims by removing the step that involved “sending, from the electronic computing system to the storage devices in the set of fastest-responding storage devices, secondary read requests to retrieve data stored at secondary storage locations.” EX1006 (’194 File History), 793–99. After these amendments were made, Examiner Lemma approved the claims. EX1006 (’194 File History), 820–26.

VII. Level Of Ordinary Skill In The Art

158. In my opinion, a person of ordinary skill in the relevant field or art (“POSITA”) as of the earliest claimed priority date of the ’194 Patent would have had a Bachelor’s degree in Computer Science, Computer Engineering, Electrical Engineering, or an equivalent field, and about 2-3 years of experience in the fields of data storage and security. Less professional experience can be substituted by additional education, and vice versa.

159. I met and/or exceeded these requirements for a POSITA at the time of the earliest claimed priority date of the ’194 Patent.

VIII. Claim Construction

160. I have reviewed the claims, specification, and the file history of the '194 Patent. EX1001 ('194 Patent); EX1006 ('194 File History). I have also reviewed the file history of the '839 Application, and U.S. Provisional Applications 60/622,146 and 60/718,185. EX1007 ('839 App. File History), EX1009 (Provisional Application No. 60/622,146), EX1010 (Provisional Application No. 60/718,185).

161. I have been asked to determine if there are any particular claim terms that should be construed for the Board to understand how the prior art references read on the challenged claims. I have not identified any such claim terms.

162. Thus, I have afforded the claim terms their ordinary and customary meaning.

IX. Specific References And Grounds For Challenge

A. Ground I: Dickinson And Hardjono Render Obvious Claims 1–20.

163. Claims 1–20 would have been obvious in view of Dickinson and Hardjono.

164. In this Ground, I introduce illustrations to show how the Dickinson/Hardjono combination renders Claims 1–20 obvious.

1. Overview of Dickinson (EX1003)

165. The content disclosed in Dickinson is largely the same as what appears in the '194 Patent. Specifically, Figures 1 through 20 in Dickinson, along with their corresponding descriptions, closely mirror Figures 1 through 20 and the related descriptions found in the '194 Patent. *See* EX1013 (providing a highlighted version of the '194 Patent's specification where the highlighted portion appears in Dickinson). Since the figures and descriptions in the '194 Patent support a majority of its claim limitations, Dickinson likewise discloses material that renders many of those same limitations unpatentable.

166. As shown in Figure 1 of Dickinson, the system features a “cryptographic system 100.” EX1003 (Dickinson), 9:22–24. This system includes elements such as a “user system 105” and a “vendor system 120,” which are each connected to a “trust engine 110” through a “communication link 125.” EX1003 (Dickinson), 9:22–24, FIG. 1 (below).

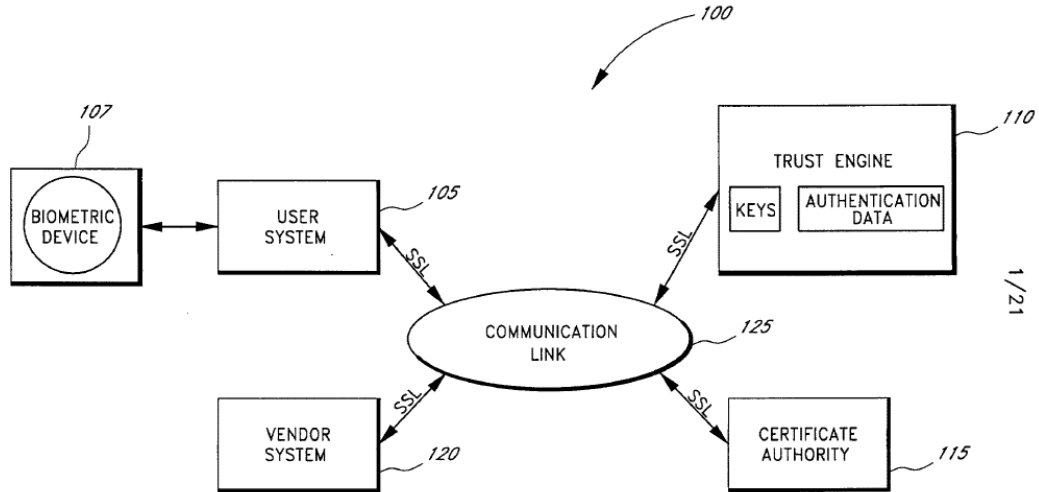


FIG. 1

EX1003 (Dickinson), FIG. 1.

167. The trust engine 110 delivers “complete cryptographic functionality,” including tasks like encryption and decryption. EX1003 (Dickinson), 11:3–9. Inside the trust engine is a “depository 210,” which “comprises one or more data storage facilities, such as, for example, a directory server, a database server, or the like.” EX1003 (Dickinson), 13:5–17, 14:17–24, FIG. 2 (below).

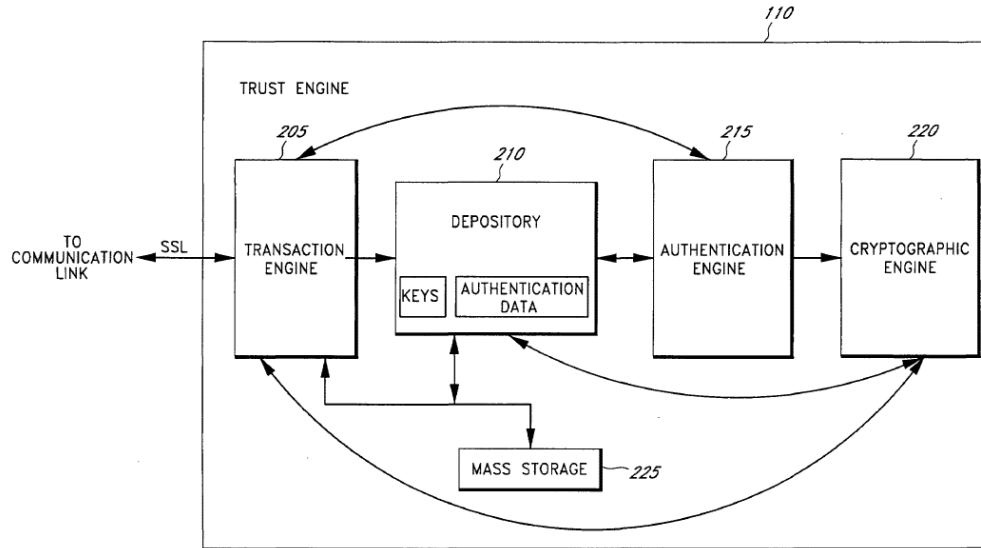


FIG. 2

EX1003 (Dickinson), FIG. 2.

168. Additionally, the trust engine 110 incorporates both an “authentication engine 215” and a “cryptographic engine 220.” EX1003 (Dickinson), 13:5–17. These components “employ their respective data splitting modules to divide sensitive data into undecipherable portions, and then transmit one or more undecipherable portions of the sensitive data to a particular data storage facility” located within the depository 210. EX1003 (Dickinson), 19:27–34.

169. Dickinson explains that the “depository system ... advantageously comprises multiple data storage facilities.” EX1003 (Dickinson), 19:16–26. As shown in Figure 7 below, the “depository 210” within the trust engine 110 protects a data set by creating multiple shares from it and then distributing those shares across

several data storage facilities, labeled D1 through D4. EX1003 (Dickinson), 20:1–14.

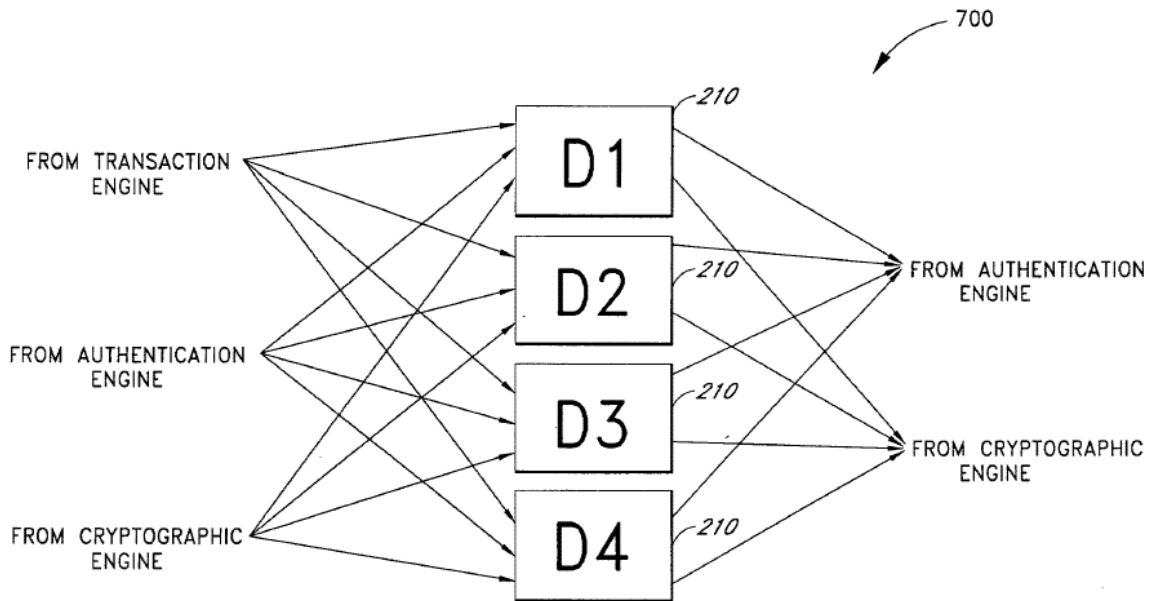


FIG. 7

EX1003 (Dickinson), FIG. 7.

170. The system shown in Figure 7 (above) can execute a method for the secure storage and retrieval of data, as further detailed in Figure 8 (below). EX1003 (Dickinson), 20:30–21:9.

8/21

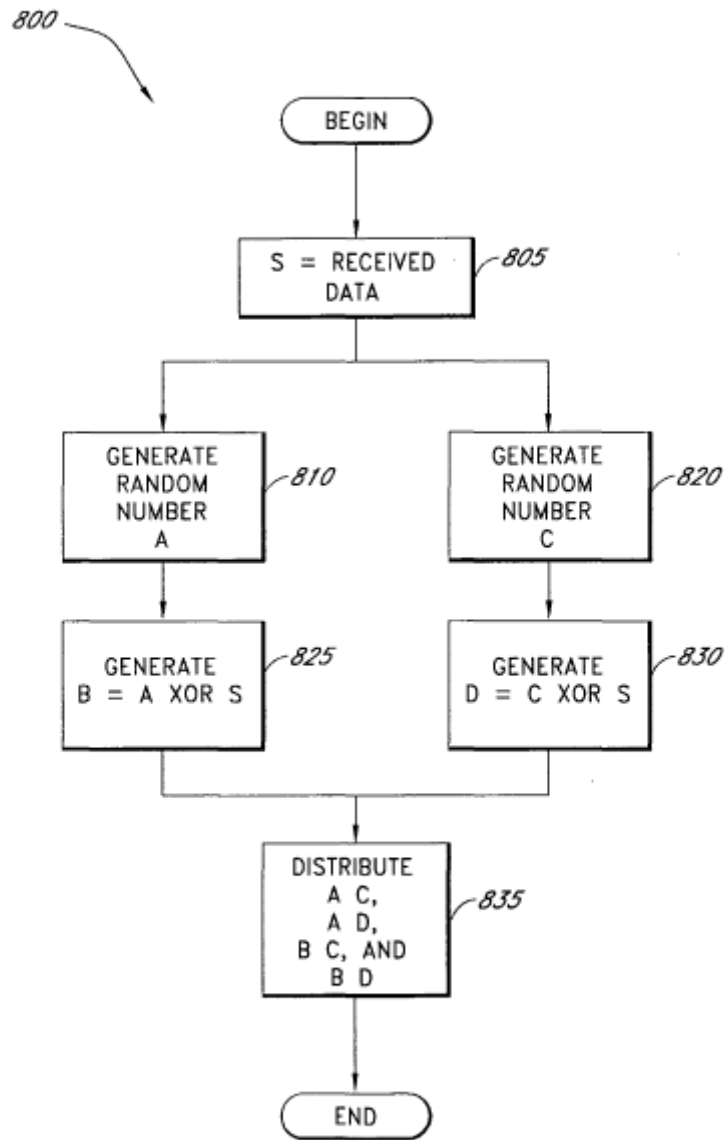


FIG. 8

EX1003 (Dickinson), FIG. 8.

171. As an illustration, the system begins by receiving a request to write or store a data set. EX1003 (Dickinson), 19:16–34. It then applies a cryptographic

process to that data in order to produce multiple data shares. EX1003 (Dickinson), 20:20–21:9. These shares are distributed in such a way that the data set can later be reconstructed from any subset that includes at least a minimum number of shares—fewer than the full set. EX1003 (Dickinson), 21:10–14, 21:21–32. For example, the system may generate two random values, numbers, bit sequences, or strings labeled “A” and “C.” EX1003 (Dickinson), 20:20–21:9. The system then uses “A” and “C” in conjunction with the sensitive data “S” to derive new values “B” and “D,” respectively. EX1003 (Dickinson), 20:30–21:9. This can be done, for example, using an exclusive/or (XOR) operation: $B = A \text{ XOR } S$, $D = C \text{ XOR } S$. EX1003 (Dickinson), 20:30–21:9.

172. The system then forms pairings of A, B, C, and D “such that none of the pairings contain sufficient data, by themselves, to reorganize and decipher the original sensitive data S.” EX1003 (Dickinson), 20:30–21:9. For example, the data may be organized into the pairs AC, AD, BC, and BD, in such a way that any two of them provide one element from (A and B) and one from (C and D). EX1003 (Dickinson), 21:21–32. Each of these pairings is stored across the four data storage facilities D1 through D4, as depicted in Figure 7 (above). EX1003 (Dickinson), 20:30–21:14.

173. When a user initiates recovery of the original data set, Dickinson’s system and method obtain the required data shares from a group of “fastest-

responding” storage devices that together hold the minimum number of shares needed for reconstruction. The system first determines which storage devices meet both the responsiveness criteria and the share threshold. EX1003 (Dickinson), 19:16–26. It then retrieves the necessary number of shares from those selected devices. EX1003 (Dickinson), 19:16–26, 21:10–14, 21:21–32. To identify the appropriate “fastest-responding” devices, the system “may broadcast requests to particular data storage facilities based on a wide number of criteria, such as, for example, response time, server loads, maintenance schedules, or the like.” EX1003 (Dickinson), 19:16–26.

174. Once the required shares have been retrieved, the system assembles them to recreate the original data set. EX1003 (Dickinson), 21:21–32.

2. Overview of Hardjono (EX1004)

175. Hardjono describes a system and method designed to securely store and retrieve data. EX1004 (Hardjono), Abstract, 7:65–8:2. Like the ’194 Patent, Hardjono’s approach involves dividing the data into multiple segments and saving those segments across several storage devices. EX1004 (Hardjono), 3:28–39 (“Server 110 implements the secure data storage according to one embodiment of the present invention. Server 110 securely stores data by creating a plurality of ‘shares’ of the data. A share of data is calculated in the illustrated embodiment by applying a ‘threshold scheme’ to the data. A threshold scheme generates multiple

(n) ‘shares’ based on the document of which at least a subset (k) are required in order to re-construct the document. Each share of data is based on the data, however, no portion of the document can be re-constructed based on the share alone (rather, k shares are required.”).

176. Hardjono provides details of its threshold scheme’s operation and benefits. EX1004 (Hardjono), 5:49–6:14 (“In the illustrated embodiment, share generation process 230 uses a ‘threshold scheme’ to create the multiple shares of data. A threshold scheme is one in which multiple (n) shares are generated based on the block of data. ... However, in order to re-create the block of data from the shares, only a subset (k) of the shares is required, with k being less than or equal to n and greater than or equal to 2. ... Each individual share generated by a threshold scheme in and of itself provides virtually no information about the original block of data. In fact, even obtaining k–1 shares would provide virtually no information about the original block of data. Thus, the distributed database environment of the present invention provides for secure storage of a block of data because at least k shares of such data must be compromised before the security of the storage is broken. By way of example, assuming the original block of data is a two-page letter and the share generation process 230 uses a (20, 39) threshold scheme, then thirty-nine different shares are generated based on the two-page letter, twenty of which are required to re-create the letter. Thus, if an individual were to attempt to ‘hack’ into the

distributed database environment of the present invention, he or she must compromise no fewer than twenty of the distributed databases in order to re-create the letter. Even if nineteen of the distributed databases were comprised, no text from the letter could be read by the hacker.”).

177. Also consistent with the '194 Patent, Hardjono's system can reconstruct the original data even when only a portion of the stored pieces is available. EX1004 (Hardjono), 6:53–67 (“It is to be appreciated that, given the distribution of multiple shares to multiple distributed databases, in combination with only k shares being required to re-create the block of data, the secure data storage of the present invention provides increased fault tolerance. By way of example, if n shares are generated based on a block of data and only k shares are required to re-create the block of data, then $n-k$ of the databases 131-133, or the couplings 145, of FIG. 1 can be temporarily or permanently unavailable ... and the data can still be re-created. ...”).

178. Hardjono describes using a “threshold scheme” to divide data into shares that are stored across different databases, with only a subset of those shares being necessary to reconstruct the original data. Under this approach, Hardjono applies a “threshold scheme” to generate a plurality of data shares from the original data. EX1004 (Hardjono), 3:28–38 (“A share of the data is calculated in the illustrated embodiment by applying a “threshold scheme” to the data. A threshold

scheme generates multiple (n) ‘shares’ based on the document of which at least a subset (k) are required in order to re-construct the document. Each share of the data is based on the data, however, no portion of the document can be re-constructed based on the share alone (rather, k shares are required).”).

179. Hardjono then explains how the shares are then “distributed to multiple separate databases.” EX1004 (Hardjono), 3:40–50 (“Once generated, the separate shares are distributed to multiple separate databases 131-133. In one embodiment, each of the n shares is sent to a different one of n or more databases. Upon subsequent receipt of a request for the data, server 110 verifies that the requester is entitled to access the data. Once the requestor’s access is verified, server 110 obtains at least k shares from the databases and re-constructs the original data using these k shares. The re-constructed data is then returned to the requester. In the illustrated embodiment, each of the databases 131-133 is situated in a different physical location than the others.”).

180. To retrieve the original data, a user sends a request through Hardjono’s network/communication interface. EX1004 (Hardjono), FIG. 5, 8:18-25 (“These elements 502-522 perform their conventional functions known in the art. In particular, network/communication interface 524 is used to provide communication between system 500 and any of a wide range of conventional networks, such as a LAN (e.g., using an Ethernet or token ring), the Internet, etc. It is to be appreciated

that the circuitry of interface 524 is dependent on the type of network the system 500 is being coupled to.”). This interface enables Hardjono’s system to connect with external networks, including a LAN or the Internet. *Id.* As a result, a user can interact with the system over these networks using the network/communication interface, thereby submitting a request for the stored data. *Id.*

181. When a user makes a request for data, Hardjono’s system determines which of the distributed databases can respond most quickly. Hardjono describes “identify[ing] databases where k shares of the block are stored,” and explains that this identification is done “based on determining which of the distributed databases are currently, most easily, or *most quickly accessible*.” EX1004 (Hardjono), 7:57–64. A POSITA would have understood that the term “most quickly accessible” refers to the storage devices that have the fastest response times.

182. Once the shares have been retrieved from the databases, Hardjono’s system recombines them to reconstruct and return the original data to the requester. The reconstruction process requires at least k of the data pieces. EX1004 (Hardjono), 6:53–67. Hardjono explains that “[u]pon subsequent receipt of a request for the data,” “[t]he re-constructed data is then returned to the requester.” EX1004 (Hardjono), 3:40–50, 7:65–8:2 (“Once the databases where k shares are located are identified, server 110 retrieves the k shares for the block of data step 425. Server

110 then re-creates the block from the k shares, step 430, and returns the re-created data block to the requester, step 435.”).

3. Overview of Dickinson in View of Hardjono (“Dickinson/Hardjono”)

183. Both Dickinson and Hardjono describe systems and methods for securely storing data by dividing the data into multiple segments and saving those segments across several storage devices and then retrieving data by substantively reversing the process. *See* §§IX.A.1 above (Dickinson) and IX.A.2 above (Hardjono). Although Dickinson does not expressly teach certain details recited in the claims challenged below, those details would have been obvious in view of Hardjono. A POSITA would have been motivated to combine Dickinson with Hardjono to implement Hardjono’s teachings in Dickinson’s system and would have had a reasonable expectation of success in doing so, as described next.

184. For convenience, I discuss the combination of Dickinson and Hardjono, as well as the motivations to combine them, along three separate topics:

- Using the fastest-responding storage devices, in §IX.A.3.i below;
- Sending assembled data to a requested in response to a request, in §IX.A.3.ii below; and
- Encrypting data portions with separate keys, in §IX.A.3.iii below.

i. Fastest Responding Storage Devices in View of Hardjono

185. As part of the data retrieval process, both Dickinson and Hardjono describe identifying storage devices based on characteristics of those devices. For example, Dickinson explains that “the transaction engine 205 may broadcast requests to particular **data storage facilities** based on a wide number of criteria, such as, for example, **response time**, server loads, maintenance schedules, or the like.” EX1003 (Dickinson), 19:24–26. Similarly, Hardjono teaches “**identify[ing] databases** where **k shares** of the block are stored” including “**based on** determining which of the distributed **databases** are currently, most easily, or **most quickly accessible**.” EX1004 (Hardjono), 7:57–64. As detailed next, it would have been obvious to combine Dickinson’s system with Hardjono’s teachings such that Dickinson’s trust engine identifies, from among its storage facilities D1–D4, “where **k shares** of the block are stored,” “based on [] which of the distributed databases are currently, most easily, or **most quickly accessible**.” EX1004 (Hardjono), 7:57–64. *See also* §V.D.1 above

a. Reasons to Combine Dickinson with Hardjono’s Teaching of Fastest Responding Storage Devices

186. In my opinion, a POSITA would have been motivated to combine Dickinson with Hardjono’s teaching of identifying fastest responding storage

devices and retrieving shares from the identified storage device, for at least the following five reasons.

187. **First**, both Dickinson and Hardjono are directed to a substantially similar technical issue and a substantially similar solution. For example, Dickinson explains that “the transaction engine 205 may broadcast requests to particular **data storage facilities** based on a wide number of criteria, such as, for example, **response time**, server loads, maintenance schedules, or the like.” EX1003 (Dickinson), 19:24–26. Similarly, Hardjono teaches “**identify[ing] databases** where **k shares** of the block are stored” including “**based on** determining which of the distributed **databases** are currently, most easily, or **most quickly accessible**.” EX1004 (Hardjono), 7:57–64. Accordingly, a POSITA would have understood that the teachings of Hardjono—using the fastest-responding storage devices to reconstruct the original data—would have been readily applicable to Dickinson.

188. **Second**, a POSITA would have been particularly motivated to implement Hardjono’s teachings in Dickinson’s trust engine to improve the user experience. It was well known that data storage systems with a faster response time—like the databases identified using Hardjono’s technique “based on [] which of the distributed databases are currently, most easily, or **most quickly accessible**” (EX1004 (Hardjono), 7:57–64)—would improve the user experience by not requiring the user to wait for the data to be retrieved. Accordingly, a POSITA would

have been motivated to use Hardjono's technique to reduce the data retrieval response time of Dickinson's trust engine.

189. **Third**, a POSITA would have been motivated to use Hardjono's technique to identify and use the "most quickly accessible" storage devices because a POSITA would have understood that such technique improves the efficiency of the networked system. A POSITA would have understood that the "most quickly accessible" storage devices tend to be closer geographically, and thus tends to require packets to travel shorter distances across the network. EX1005 (Canali), 5 ("This [response time] performance gain is due to the placement of the adaption servers close to the clients[.]"). By requiring packets to travel less distance, the system as a whole would have been more efficient.

190. **Fourth**, a POSITA would have understood that identifying and using the "most quickly accessible" storage devices would have improved the network load distribution. For example, it was well understood in the art that selecting the fastest responding servers "contribut[es] to evenly distribute network load[.]" See EX1005 (Canali), 6. By evenly distributing the network load, the system as a whole would have been able to serve a larger number of requests more quickly and more efficiently, with less failures and improved user experience. Thus, a POSITA would have been motivated to identify and use the fastest-responding servers for improved

system efficiency. Indeed, techniques to mitigate high network latencies, such as load balancing, were well known in the art. *See* §V.D.1 above.

191. **Fifth**, a POSITA would have recognized that this combination simply involves applying a known technique—Hardjono’s method of identifying databases that are currently, most easily, or most quickly accessible—to a known system, namely Dickinson’s storage architecture. This system was already suitable for enhancement, and the application of Hardjono’s technique would have predictably improved the speed of data retrieval.

192. Moreover, in my opinion, a POSITA would have had a reasonable expectation of successfully implementing Hardjono’s technique of identifying the most quickly accessible databases within Dickinson’s system, for at least three reasons.

193. **First**, it would have been obvious to combine Dickinson with Hardjono so that the criteria used by Dickinson’s data storage facilities included identifying the “most quickly accessible” databases, as taught by Hardjono. Both Dickinson and Hardjono describe methods for selecting databases or shares of data based on performance-related characteristics that enhance the efficiency of the data reconstruction process. *Compare* EX1003 (Dickinson), 19:24–26 (“On the other hand, the transaction engine 205 may broadcast requests to particular data storage facilities based on a wide number of criteria, such as, for example, response time,

server loads, maintenance schedules, or the like.”) *with* EX1004 (Hardjono), 7:57–64 (“However, if access is permitted, then server 110 identifies databases where k shares of the block are stored, step 420. This identification process may be based on determining where shares are located (e.g., there are more databases in the distributed database environment than there are shares), or alternatively may be based on determining which of the distributed databases are currently, most easily, or most quickly accessible.”).

194. The result of this combination would have been just as expected; it would simply make explicit that one of the “criteria” referenced in Dickinson (EX1003 (Dickinson), 19:24–26) is the ability to identify database locations that are “most quickly accessible” (EX1004 (Hardjono), 7:57–64).

195. **Second**, the combination would have simply involved adding Hardjono’s identification step—specifically, identifying the databases that are currently, most easily, or most quickly accessible—after the step in Dickinson where the system receives a request to retrieve the data. Implementing this additional step would predictably enhance the speed of data recombination. *See* EX1004 (Hardjono), 7:57–64.

196. **Third**, incorporating Hardjono’s teachings into Dickinson’s system would have involved simply writing program code, since Dickinson’s trust engine modules are implemented in software. *See, e.g.*, EX1003 (Dickinson), 18:20–21

(“The authentication engine 215 also includes the data splitting module 520,” which “advantageously comprises a software, hardware, or combination module[.]”), 18:23–24 (“The data assembling module 525 advantageously comprises a software, hardware, or combination module[.]”), 19:3–5 (“The cryptographic engine 220 also comprises a cryptographic handling module 625” that “may comprise software modules or programs, hardware, or both.”). Writing such code to implement a function already identified—such as the one disclosed in Hardjono—would have been well within the ordinary skill in the art.

197. Thus, modifying Dickinson with Hardjono’s step of identifying the databases that are currently, most easily, or most quickly accessible would have been simple and would not have required extensive experimentation or imposed undue burden, because both references already disclose secure distributed storage systems. For the same reasons, the combination would have yielded predictable results (because identifying the databases that are currently, most easily, or most quickly accessible is a well-established technique). Lastly, a POSITA would have had a reasonable expectation of success because Hardjono merely adds implementation detail to functionality already aligned with Dickinson’s goals.

ii. Sending Assembled Data to the Requester in Response to a Request in View of Hardjono

198. As part of the data retrieval process both Dickinson and Hardjono describe re-constructing data by substantively reversing the data splitting process

after receiving a request. See §§IX.A.1 above (Dickinson), IX.A.2 above (Hardjono).

199. For example, Dickinson states that “the vendor system 120 ... forwards ... the authentication *request* to the *trust engine 110*[.]” EX1003 (Dickinson), 28:6–14. Furthermore, Dickinson explains that the *trust engine 110* includes a transaction engine 205.” EX1003 (Dickinson), 13:5–17. Dickinson also teaches that “transaction engine 205 receives the [*vendor request*] ... and generates a *request* for the user’s enrollment authentication data to be assembled from the *data storage facilities D1 through D4*.” EX1003 (Dickinson), 28:6–14. Dickinson also indicates that “the *sensitive data S* includes enrollment authentication data.” EX1003 (Dickinson), 21:33–22:2.

200. Similarly, Hardjono states that, “[u]pon subsequent receipt of a *request for the data*,” “[t]he *re-constructed data* is then *returned* to the requester.” EX1004 (Hardjono), 3:40–50; *see also* EX1004 (Hardjono), 7:65–8:2 (“Server 110 then re-creates the block from the *k shares*, step 430, and *returns* the *re-created data block* to the *requester*, step 435.”).

201. It would have been obvious to combine Dickinson’s system with Hardjono’s teachings such that Dickinson’s trust engine sends assembled data in response to the requester “[u]pon subsequent receipt of a *request for the data*.” EX1004 (Hardjono), 3:40–50.

a. Reasons to Combine Dickinson with Hardjono's Teaching of Sending Assembled Data to the Requester in Response to a Request

202. In my opinion, a POSITA would have been motivated to combine Dickinson with Hardjono's teaching of sending assembled data in response to a request, for at least two reasons.

203. **First**, the trust engine in Dickinson assembles the data set in response to receiving a request to retrieve the assembled data. However, assembling the data but not sending it in response to the request to the requester would represent an inefficient use of the trust engine's resources.

204. A POSITA would readily recognize that Hardjono specifically teaches the step of sending the assembled data in response to the request to the requester. *See* EX1004 (Hardjono), 3:40–50 (“Once generated, the separate shares are distributed to multiple separate databases 131-133. In one embodiment, each of the n shares is sent to a different one of n or more databases. Upon subsequent receipt of a request for the data, server 110 verifies that the requester is entitled to access the data. Once the requestor's access is verified, server 110 obtains at least k shares from the databases and re-constructs the original data using these k shares. The re-constructed data is then returned to the requester. In the illustrated embodiment, each of the databases 131-133 is situated in a different physical location than the others.”), 7:65–8:2 (“Once the databases where k shares are located are identified,

server 110 retrieves the k shares for the block of data step 425. Server 110 then re-creates the block from the k shares, step 430, and returns the re-created data block to the requester, step 435.”). A POSITA would have appreciated that this return step is a fundamental and expected part of any functional data storage and retrieval system. *See also* §V.B above.

205. Therefore, a POSITA would have been motivated to combine Dickinson and Hardjono such that Dickinson’s trust engine transmits the assembled data set back to the requester in response to the request, consistent with Hardjono’s teachings.

206. **Second**, the combination would have been especially apparent to a POSITA because it simply involves applying a known technique—Hardjono’s step of returning the assembled data—to a similar system, namely Dickinson’s data storage architecture. The purpose and effect of the combination are consistent: enabling the system to deliver information in response to a request, thereby making Dickinson’s data storage system fully functional in the same manner.

207. Moreover, in my opinion, a POSITA would have had a reasonable expectation of successfully implementing Hardjono’s technique of sending assembled data, in response to a request, to the requester within Dickinson’s system, for at least the following five reasons.

208. **First**, both Dickinson and Hardjono are focused on the secure storage and retrieval of data.

209. Dickinson describes a system to store secure data on servers. EX1003 (Dickinson), 9:7–12 (“One aspect of the present invention is to provide a cryptographic system where one or more secure servers, or a trust engine, stores cryptographic keys and user authentication data. Users access the functionality of conventional cryptographic systems through network access to the trust engine, however, the trust engine does not release actual keys and other authentication data and therefore, the keys and data remain secure. This server-centric storage of keys and authentication data provides for user-independent security, portability, availability, and straightforwardness.”).

210. Dickinson provides details on how these secure servers operate. EX1003 (Dickinson), 10:20–32 (“FIGURE 1 illustrates the trust engine 110. According to one embodiment, the trust engine 110 comprises one or more secure servers for accessing and storing sensitive information, such as user authentication data and public and private cryptographic keys. According to one embodiment, the authentication data includes data designed to uniquely identify a user of the cryptographic system 100. For example, the authentication data may include a user identification number, one or more biometrics, and a series of questions and answers generated by the trust engine 110 or the user, but answered initially by the user at

enrollment. The foregoing questions may include demographic data, such as place of birth, address, anniversary, or the like, personal data, such as mother's maiden name, favorite ice cream, or the like, or other data designed to uniquely identify the user. The trust engine 110 compares a user's authentication data associated with a current transaction, to the authentication data provided at an earlier time, such as, for example, during enrollment. The trust engine 110 may advantageously require the user to produce the authentication data at the time of each transaction, or, the trust engine 110 may advantageously allow the user to periodically produce authentication data, such as at the beginning of a string of transactions or the logging onto a particular vendor website.”).

211. Dickinson then provides details on how the secure servers are authenticated. EX1003 (Dickinson), 38:16–21 (“The current authentication data provided by the user and the enrollment data retrieved from the depository 210 are received by the authentication engine 215 in STEP 1600 of FIGURE 16. Both of these sets of data may contain data which is related to separate techniques of authentication. The authentication engine 215 separates the authentication data associated with each individual authentication instance in STEP 1605. This is necessary so that the authentication data is compared with the appropriate subset of the enrollment data for the user (e.g. fingerprint authentication data should be compared with fingerprint enrollment data, rather than password enrollment data).”).

212. Similarly, Hardjono describes secure network servers (databases). EX1004 (Hardjono), Abstract (“A method and apparatus for secure data storage using distributed databases generates a first plurality of shares, using a first threshold scheme, based on a block of data, with at least a subset of the first plurality of shares being needed to re-create the block of data. The first plurality of shares are then distributed to a plurality of distributed databases.”), 7:65–8:2 (“Once the databases where k shares are located are identified, server 110 retrieves the k shares for the block of data step 425. Server 110 then re-creates the block from the k shares, step 430, and returns the re-created data block to the requester, step 435.”).

213. **Second**, each system also describes dividing data into pieces and storing those pieces across multiple storage devices. *Compare* EX1003 (Dickinson), 18:21–22 (“The data splitting module 520 advantageously comprises a software, hardware, or combination module having the ability to mathematically operate on various data so as to substantially randomize and split the data into portions.”), 38:16–21 (“The authentication engine 215 separates the authentication data associated with each individual authentication instance in STEP 1605.”) *with* EX1004 (Hardjono), 3:28–39 (“Server 110 securely stores data by creating a plurality of ‘shares’ of the data.”), 5:49–6:14 (“Thus, if an individual were to attempt to ‘hack’ into the distributed database environment of the present invention, he or she must compromise no fewer than twenty of the distributed databases in order to

re-create the letter.”), 6:53–67 (“It is to be appreciated that, given the distribution of multiple shares to multiple distributed databases, in combination with only k shares being required to re-create the block of data, the secure data storage of the present invention increased fault tolerance.”).

214. **Third**, both references teach that only a subset of the data pieces is needed to reconstruct the original data. *Compare* EX1003 (Dickinson), 20:1–14 (“For example, according to one embodiment, only data from two of the multiple data storage facilities, D1 through D4, are needed to decipher and reassemble the sensitive data.”) *with* EX1004 (Hardjono), 3:28–39 (“A threshold scheme generates multiple (n) ‘shares’ based on the document of which at least a subset (k) are required in order to re-construct the document.”), 5:49–6:14 (“However, in order to re-create the block of data from the shares, only a subset (k) of the shares is required, with k being less than or equal to n and greater than or equal to 2.”), 6:53–67 (“It is to be appreciated that, given the distribution of multiple shares to multiple distributed databases, in combination with only k shares being required to re-create the block of data, the secure data storage of the present invention increased fault tolerance.”).

215. **Fourth**, Dickinson and Hardjono address the same technical challenge and offer similar structural and functional solutions, so a POSITA would have readily recognized that Hardjono’s technique of sending the reassembled data in

response to the request to the requester is readily applicable to Dickinson's storage system.

216. Dickinson begins by describing a secure storage server with authentication keys. EX1003 (Dickinson), 9:7–12 (“One aspect of the present invention is to provide a cryptographic system where one or more secure servers, or a trust engine, stores cryptographic keys and user authentication data. Users access the functionality of conventional cryptographic systems through network access to the trust engine, however, the trust engine does not release actual keys and other authentication data and therefore, the keys and data remain secure. This server-centric storage of keys and authentication data provides for user-independent security, portability, availability, and straightforwardness.”).

217. Dickinson gives an example of distributing the secure information across multiple servers. EX1003 (Dickinson), 10:20–32 (“According to one embodiment, the trust engine 110 comprises one or more secure servers for accessing and storing sensitive information, such as user authentication data and public and private cryptographic keys.”).

218. Dickinson gives another specific example of splitting and distributing the authentication information and keys. EX1003 (Dickinson), 18:20–19:2 (“According to one embodiment, the authentication engine 215 employs the data splitting module 520 to randomize and split enrollment authentication data into

portions, and employs the data assembling module 525 to reassemble the portions into usable enrollment authentication data.”), 20:1–14 (“According to one embodiment, each data storage facility, D1 through D4, comprises a separate and independent storage system, such as, for example, a directory server.”), *see also* 20:20–29.

219. Moreover, Dickinson describes different user authentication methods. EX1003 (Dickinson), 38:16–21 (“The current authentication data provided by the user and the enrollment data retrieved from the depository 210 are received by the authentication engine 215 in STEP 1600 of FIGURE 16. Both of these sets of data may contain data which is related to separate techniques of authentication. The authentication engine 215 separates the authentication data associated with each individual authentication instance in STEP 1605. This is necessary so that the authentication data is compared with the appropriate subset of the enrollment data for the user (e.g. fingerprint authentication data should be compared with fingerprint enrollment data, rather than password enrollment data.”).

220. Similarly, Hardjono describes distributed servers to store and secure data and keys, where a subset of the shares is needed to recover the data. EX1004 (Hardjono), Abstract (“A method and apparatus for secure data storage using distributed databases generates a first plurality of shares, using a first threshold scheme, based on a block of data, with at least a subset of the first plurality of shares

being needed to re-create the block of data. The first plurality of shares are then distributed to a plurality of distributed databases.”), 3:28–39 (“Server 110 securely stores data by creating a plurality of ‘shares’ of the data. A share of the data is calculated in the illustrated embodiment by applying a ‘threshold scheme’ to the data.”), 5:49–6:14 (“However, in order to re-create the block of data from the shares, only a subset (k) of the shares is required, with k being less than or equal to n and greater than or equal to 2.”), 6:53–67 (“It is to be appreciated that, given the distribution of multiple shares to multiple distributed databases, in combination with only k shares being required to re-create the block of data, the secure data storage of the present invention provides increased fault tolerance.”), 7:65–8:2 (“Once the databases where k shares are located are identified, server 110 retrieves the k shares for the block of data (step 425). Server 110 then re-creates the block from the k shares (step 430) and returns the re-created data block to the requester (step 435).”).

221. **Fifth**, incorporating Hardjono’s teachings into Dickinson’s system would have involved no more than writing program code, since Dickinson’s trust engine modules are implemented in software. *See* EX1003 (Dickinson), 18:20–21 (“The authentication engine 215 also includes the data splitting module 520 and the data assembling module 525.”), 18:23–24 (“The data assembling module 525 advantageously comprises a software, hardware, or combination module”), 19:3–5 (“The cryptographic engine 220 also comprises a cryptographic handling

module 625 configured to perform some or all of a wide number of cryptographic functions. According to one embodiment, the cryptographic handling module 625 may comprise software modules or programs, hardware, or both.”). Developing software to carry out a function that has already been identified—such as Hardjono’s step of sending the reassembled data—would have been well within the capabilities of a POSITA.

222. Thus, modifying Dickinson with Hardjono’s step of returning the assembled data would have been simple and would not have required extensive experimentation or imposed undue burden, because both already disclose secure distributed storage systems.

223. For the same reasons, the combination would have yielded predictable results (because returning the assembled data is a well-established technique).

224. Lastly, a POSITA would have had a reasonable expectation of success because Hardjono merely adds implementation detail to functionality already aligned with Dickinson’s goals.

iii. Encrypting Data Portions with Separate Keys in View of Hardjono

225. Dickinson discloses a trust engine that enforces data usage policies by controlling access to and encrypting portions of data. EX1003 (Dickinson), 3:13–25 (“The method comprises associating a user from multiple users with one or more keys from a plurality of private cryptographic keys stored on a secure server.”).

226. Hardjono discloses encrypting each piece of data separately using different encryption keys. EX1004 (Hardjono), 1:30-41 (“One solution to this problem is to Separate a document into multiple pieces and encrypt each piece Separately using the same or different encryption keys. This solution provides an additional level of Security because possibly multiple keys must be compromised in order to access the entire data. However, this solution can still be problematic because compromising of a single key allows an entire piece of data to be accessible to the unauthorized user. For example, one piece of a document may be the most important (e.g., the body of a letter), so that having that one piece compromised and accessible to an unauthorized individual circumvents this additional level of security.”).

227. Hardjono further discloses that this approach enhances security because multiple keys must be compromised to access all the data. EX1004 (Hardjono), 1:30-41 (“This solution provides an additional level of security because possibly multiple keys must be compromised in order to access the entire data.”).

228. Hardjono also teaches that a wide variety of encryption techniques can be used, including RSA, which employs one or more encryption keys. EX1004 (Hardjono), 6:24–33 (“Various embodiments of the present invention utilize encryption in order to provide further security for data Storage. Encryption can be introduced at any one or more of multiple points in the present invention, as

discussed in more detail below. Any of a wide variety of conventional encryption processes can be used by the present invention, including different encryption processes at different points in the secure document storage process. Examples of such encryption processes include the RSA encryption scheme which employs one or more encryption keys to encrypt the data.”).

229. It would have been obvious to combine Dickinson with Hardjono so that Dickinson’s trust engine encrypts data portions with separate keys in view of Hardjono.

230. This is because Hardjono teaches that doing so enhances data security, a goal shared by Dickinson’s system. Incorporating Hardjono’s method into Dickinson would involve adapting the trust engine’s encryption functionality so that instead of using a single key for multiple shares or fragments, it uses a distinct encryption key for each share. This improves confidentiality by ensuring that compromising one key would not expose the entire dataset.

a. Reasons to Combine Dickinson with Hardjono’s Teaching of Encrypting Data Portions with Separate Keys

231. In my opinion, a POSITA would have been motivated to combine Dickinson with Hardjono in a manner where each of Dickinson’s data portions is encrypted using a separate key, for at least two reasons.

232. **First**, Hardjono expressly states that the use of “different encryption keys” “provides an additional level of security because possibly multiple keys must be compromised in order to access the entire data.” EX1004 (Hardjono), 1:30–41. Based on this teaching, a POSITA would have been motivated to apply different encryption keys to each portion of a data set to enhance overall security.

233. **Second**, a POSITA would have recognized that this combination simply involves applying a known method—namely, Hardjono’s approach of using different encryption keys for individual data pieces—to an existing system, such as Dickinson’s storage system. This combination targets a system that is already suitable for enhancement and would predictably lead to improved secrecy of the stored information.

234. Moreover, in my opinion, a POSITA would have had a reasonable expectation of successfully implementing Hardjono’s method of using different encryption keys within Dickinson’s system, for at least three reasons.

235. **First**, both Dickinson and Hardjono explicitly disclose the use of multiple encryption keys. For example, Dickinson states: “The method comprises associating a user from multiple users with *one or more keys* from a plurality of private cryptographic keys stored on a secure server.” EX1003 (Dickinson), 3:13–25 (“The method comprises associating a user from multiple users with one or more keys from a plurality of private cryptographic keys stored on a secure server.”).

Similarly, Hardjono teaches “encrypting each piece separately using ... *different encryption keys*.” EX1004 (Hardjono), 1:30–41 (“One solution to this problem is to separate a document into multiple pieces and encrypt each piece separately using the same or different encryption keys. This solution provides an additional level of security because possibly multiple keys must be compromised in order to access the entire data.”). Given that Dickinson’s system already incorporates the use of multiple keys, a POSITA would readily apply Hardjono’s approach of encrypting individual data shares with separate keys, thereby enhancing the security of Dickinson’s system.

236. **Second**, integrating Hardjono’s encryption technique—where each data portion is encrypted with a dedicated key—into Dickinson’s system would simply involve applying this encryption step after the data set has been divided. This combination would predictably enhance the confidentiality of the resulting data portions. As Hardjono explains, this approach “provides an additional level of security because possibly multiple keys must be compromised in order to access the entire data.” EX1004 (Hardjono), 1:30–41.

237. **Third**, incorporating Hardjono’s approach into Dickinson’s system would have primarily required writing program code, since the modules in Dickinson’s trust engine are software-based. Implementing a known function—

such as the encryption method described in Hardjono—through programming would have been well within the capabilities of a POSITA.

238. Thus, modifying Dickinson with Hardjono’s encrypting using different encryption keys for individual data pieces would have been simple and would not have required extensive experimentation or imposed undue burden, because both already disclose secure distributed storage systems. For the same reasons, the combination would have yielded predictable results (because using different encryption keys for individual data pieces a well-established technique). Lastly, a POSITA would have had a reasonable expectation of success because Hardjono merely adds implementation detail to functionality already aligned with Dickinson’s goals.

4. Color-Coding Scheme for Dickinson/Hardjono

239. The illustrations in this ground have been color-coded in accordance with the color-coding scheme summarized in the table below.

Color	Disclosure in Dickinson and Hardjono
purple	Dickinson’s trust engine
red	Dickinson’s portions
	Hardjono’s shares
forest green	Dickinson’s various cryptographic operations
dark orange	Dickinson’s data
pink	Dickinson’s desires to enroll/transmitting enrollment data
light blue	Dickinson’s forward
light green	Dickinson’s split

5. **Claim 1: A method for securely storing and retrieving data, the method comprising: generating, using an electronic computing system that includes processing circuitry, a plurality of shares by performing a cryptographic operation on a data set and distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of shares; storing the plurality of shares at a plurality of storage devices; receiving, at the electronic computing system, request to retrieve the data set; identifying from the plurality of storage devices a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices; retrieving from the set of fastest-responding storage devices, the minimum number of shares; reconstructing the data set using the minimum number of shares; and sending the data set responsive to the request.**

- i. **[1Pre] A method for securely storing and retrieving data, the method comprising:**

240. To the extent the preamble is limiting, Dickinson discloses preamble [1Pre].

241. Dickinson discloses “[a] *method for securely storing and retrieving data.*” For example, Dickinson describes a system that “provide[s] a cryptographic system where one or more *secure* servers, or a trust engine, *stores cryptographic keys and user authentication data.*” EX1003 (Dickinson), 9:7–12. Additionally, Dickinson states that “the trust engine 110 comprises one or more *secure* servers for accessing and *storing sensitive information.*” EX1003 (Dickinson), 10:20–32. *See*

also EX1003 (Dickinson), 38:13 (“assembling the *enrollment data* 410 *retrieved* from the depository 210”).

242. **In sum**, Dickinson discloses “[a] *method for securely storing and retrieving data* [Dickinson’s providing a cryptographic system and accessing and storing sensitive information].”

- ii. **[1A] generating, using an electronic computing system that includes processing circuitry, a plurality of shares by performing a cryptographic operation on a data set and distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of shares;**

243. Dickinson discloses claim limitation [1A]. For convenience, next I discuss this limitation in two parts.

- a. **[1A-1] generating, using an electronic computing system that includes processing circuitry, a plurality of shares by performing a cryptographic operation on a data set and**

244. **First**, Dickinson discloses “*using an electronic computing system that includes processing circuitry*[.]” For example, Dickinson describes a “*trust engine 110*,” which comprises components such as a transaction engine 205, a depository 210, an authentication engine 215, a cryptographic engine 220, and mass storage 225, and notes that it “provides the user with complete cryptographic functionality.” EX1003 (Dickinson), 11:5–6; *see also* EX1003 (Dickinson), 13:5–7 (explaining that FIG. 2 is “a block diagram of the trust engine 110 of FIG. 1”).

245. Dickinson further explains that these components interact through mechanisms that rely on *processing circuitry*—for example, by transmitting XML documents to IP addresses (EX1003 (Dickinson), 13:10–17) and routing data using standard HTTP routing techniques such as URLs or URIs (EX1003 (Dickinson), 13:29–30).

246. Indeed, Dickinson discloses that the modules inside the trust engine 110 may be implemented in “*software, hardware, or combination.*” EX1003 (Dickinson), 18:20-28 (“The authentication engine 215 also includes the data splitting module 520 and the data assembling module 525. The data splitting module 520 advantageously comprises a software, hardware, or combination module having the ability to mathematically operate on various data so as to substantially randomize and split the data into portions. According to one embodiment, original data is not recreatable from an individual portion. The data assembling module 525 advantageously comprises a software, hardware, or combination module configured to mathematically operate on the foregoing substantially randomized portions, such that the combination thereof provides the original deciphered data. According to one embodiment, the authentication engine 215 employs the data splitting module 520 to randomize and split enrollment authentication data into portions, and employs the data assembling module 525 to reassemble the portions into usable enrollment authentication data.”). A POSITA would have understood that the trust engine, as

described, constitutes an example of an electronic computing system. *See also* §V.A above.

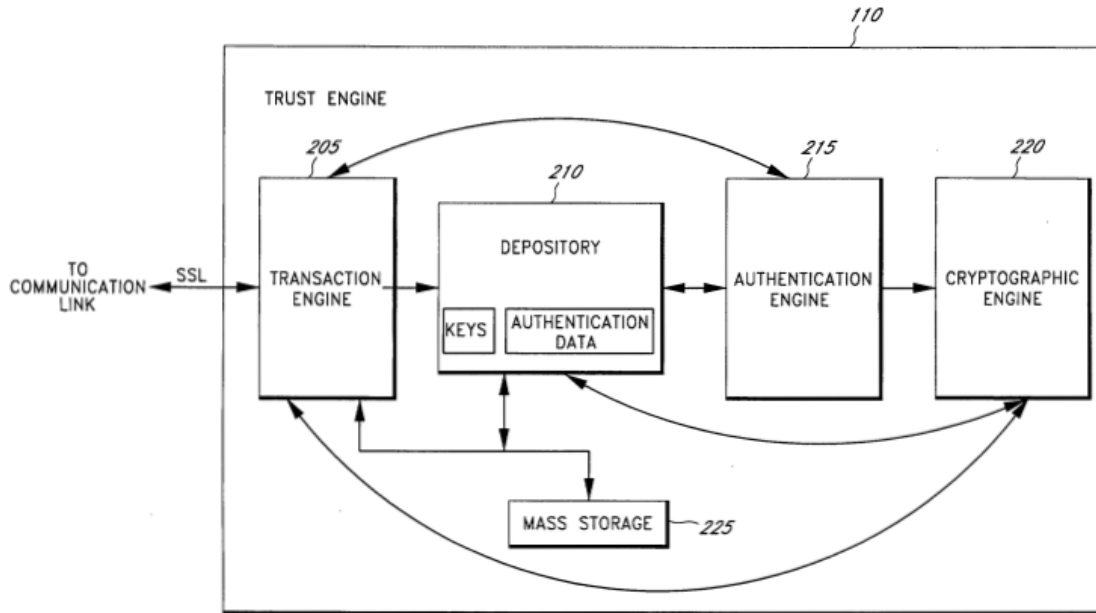


FIG. 2

EX1003 (Dickinson), FIG. 2.

247. **Second**, Dickinson discloses the step of “generating, using an electronic computing system that includes processing circuitry, a plurality of shares by performing a cryptographic operation on a data set.” For example, Dickinson explains that “the authentication engine 215 and the cryptographic engine 220,” both components of the trust engine 110, “may advantageously employ their respective data splitting modules to divide sensitive data into undecipherable portions, and then transmit one or more undecipherable portions of the sensitive

data to a particular data storage facility.” EX1003 (Dickinson), 19:31–34; *see also* EX1003 (Dickinson), 18:20–28, 20:20–21:9.

248. Dickinson further describes the cryptographic operations involved in this process. For example, Figure 5 (below) shows that “[t]he data splitting module 520 advantageously comprises a software, hardware, or combination module having the ability to **mathematically operate on various data** so as to ... **split the data** into **portions**.” EX1003 (Dickinson), 18:20–28.

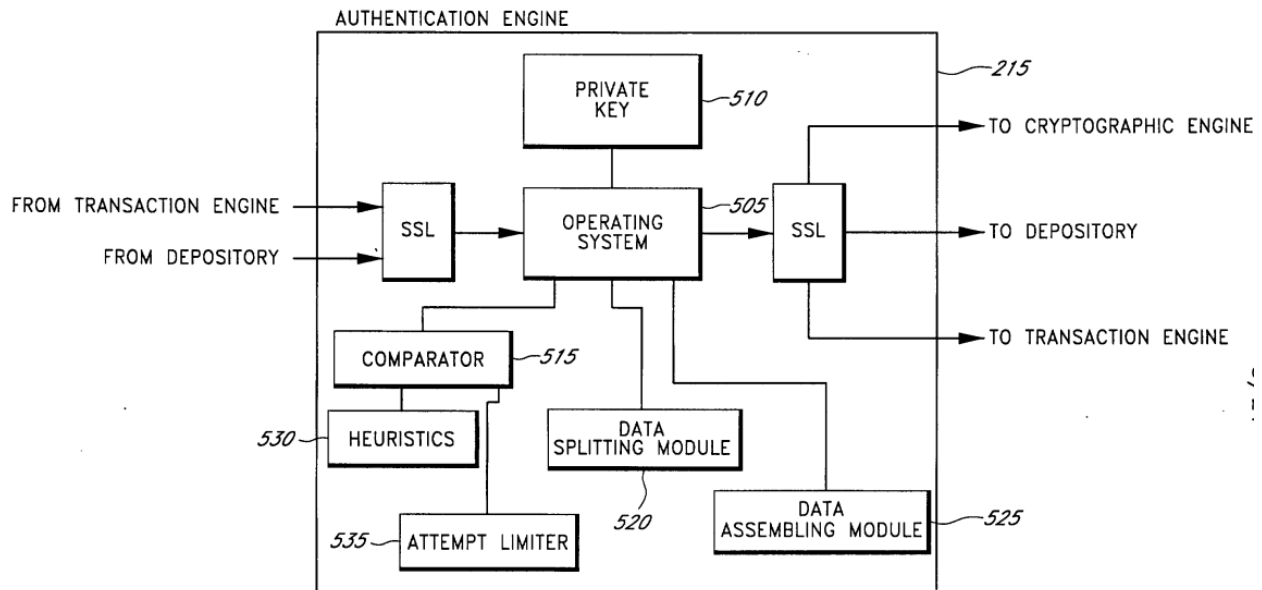


FIG. 5

EX1003 (Dickinson), FIG. 5.

249. In another example, Dickinson describes that “the authentication engine 215 and the *cryptographic* engine 220 may advantageously employ their respective data splitting modules [520 and 610, respectively] to **divide sensitive data** into **undecipherable portions**[.]” EX1003 (Dickinson), 19:27–34. This process is

depicted in Figure 8, which “illustrates a flowchart of a data splitting process 800 performed by the data splitting module” and indicates that “*sensitive data ‘S’* is received by the data splitting module of ... the cryptographic engine 220” followed by the step where “the data splitting module then generates *a substantially random number value, or string or set of bits, ‘A.’*” EX1003 (Dickinson), 20:20–29, FIG. 8 (below).

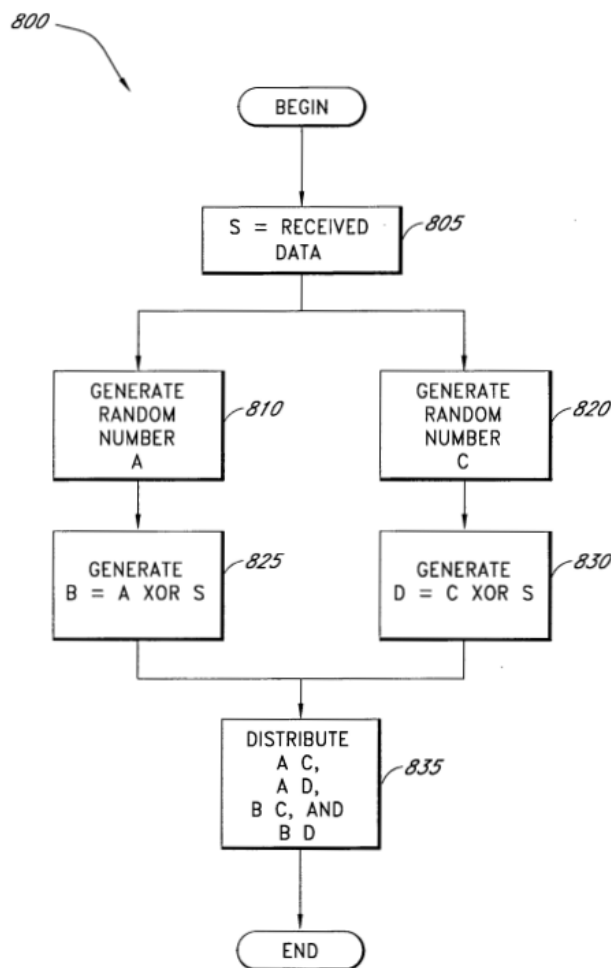


FIG. 8

EX1003 (Dickinson), FIG. 8.

250. Furthermore, Dickinson explains that “the trust engine 110, may advantageously perform ... only some or all of the *cryptographic functions, such as data encryption and decryption*” in the course of generating the plurality of shares. EX1003 (Dickinson), 16:6–14.

251. Lastly, Dickinson also teaches that the “sensitive data”—to which the operations are performed—is a “*data set.*” For example, Dickinson states that in one example, “sensitive data S includes enrollment *authentication data.*” EX1003 (Dickinson), 21:34-35; *see also* EX1003 (Dickinson), 20:20-29 (“[T]he authentication engine 215 and the cryptographic engine 220 each include a data splitting module 520 and 610, respectively, for splitting sensitive data, such as, for example, the authentication data and the cryptographic key data.”).

252. A POSITA would have understood that such *authentication data* includes a set of information, such as a username and password. *See* EX1019 (Brennan), [0012] (“The first web site authenticates the user using one authentication method, for example the *username and password.*”); EX1020 (Bennet), [0005] (“The connections shown provide a path for user client 12 to provide the user’s FI [Financial Institution] *username and password* to portal server 16, a path for portal server 16 to read and write user *authentication data* (such as *username, password, associate FI, etc.*)”).

253. Dickinson also teaches that the “communications ... to the data storage facilities D1 through D4 may include the transmission of sensitive data *to be stored*.” EX1003 (Dickinson), 19:29-31; *see also* EX1003 (Dickinson), 20:4-6 (“***By distributing the sensitive data into distinct and independent storage facilities D1 through D4***, some or all of which may be advantageously geographically separated, the depository system 700 provides redundancy along with additional security measures.”). Since Dickinson’s “sensitive data” such as “authentication data” includes a “set” of items of information to be stored, Dickinson’s “sensitive data” satisfies the “*data set*” limitation.

254. Notably, Dickinson’s “sensitive data” qualifies as a claimed “data set” under Patent Owner’s own definition of the term. In a prior litigation involving U.S. Patent No. 11,178,116—which belongs to the same patent family as the ’194 Patent and shares its specification—Patent Owner adopted a broad construction of “data set” as meaning a “collection of information for storage.” EX1021 (PO’s Google Litigation Construction), 8. The “*sensitive data*” in Dickinson satisfies Patent Owner’s construction of “*data set*” because the sensitive data, such as “enrollment authentication data” is a collection of information that is stored by Dickinson’s system.

255. **In sum**, Dickinson discloses “*generating* [Dickinson’s splitting], *using an electronic computing system that includes processing circuitry* [Dickinson’s trust

engine], *a plurality of shares* [Dickinson's portions] by performing a cryptographic operation [Dickinson's mathematical operation] *on a data set* [Dickinson's data]."

b. [1A-2] distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of shares;

256. First, Dickinson discloses the step of "*distributing the data set in the plurality of shares.*" For example, Dickinson discusses Figure (below) 7 and says that "[b]y *distributing* the *sensitive data* into distinct and independent storage facilities D1 through D4, some or all of which may be advantageously geographically separated, the depository system 700 provides redundancy along with additional security measures." EX1003 (Dickinson), 20:1–14. Additionally, Dickinson explains that "the data splitting process 800 advantageously *places portions* of the *sensitive data* in each of the four data storage facilities D1 through D4." EX1003 (Dickinson), 21:10–14.

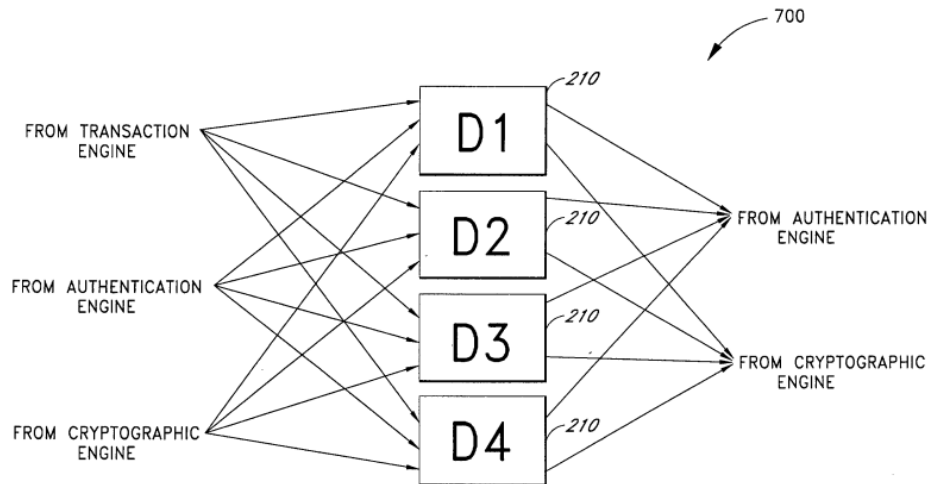


FIG. 7

EX1003 (Dickinson), FIG. 7.

257. **Second**, Dickinson discloses that “the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of shares.” For example, Dickinson explains that “the data splitting module then generates a substantially random number, value, or string or set of bits, ‘A,’” along with “another statistically random number ‘C.’” EX1003 (Dickinson), 20:20–21:9. Dickinson further describes that “[t]he data splitting module then combines the numbers A and C with the **sensitive data S** such that new numbers ‘B’ and ‘D’ are generated.” EX1003 (Dickinson), 20:30–21:9. *See also* EX1003 (Dickinson), 20:33–34 (“For example, number B may comprise the binary combination of A XOR S and number D may comprise the binary combination of C XOR S.”).

258. Dickinson notes that “the random numbers A and C and the numbers B and D are paired such that none of the pairings contain sufficient data, by themselves,

to reorganize and decipher the original *sensitive data S*,” and that “the numbers may be paired as follows: AC, AD, BC, and BD,” such that “the pairings of AC, AD, BC, and BD, were distributed such that *any two* provide one of A and B, or, C and D.” EX1003 (Dickinson), 20:30–21:9, 21:21–32; *see also* EX1003 (Dickinson), FIG. 8 (below).

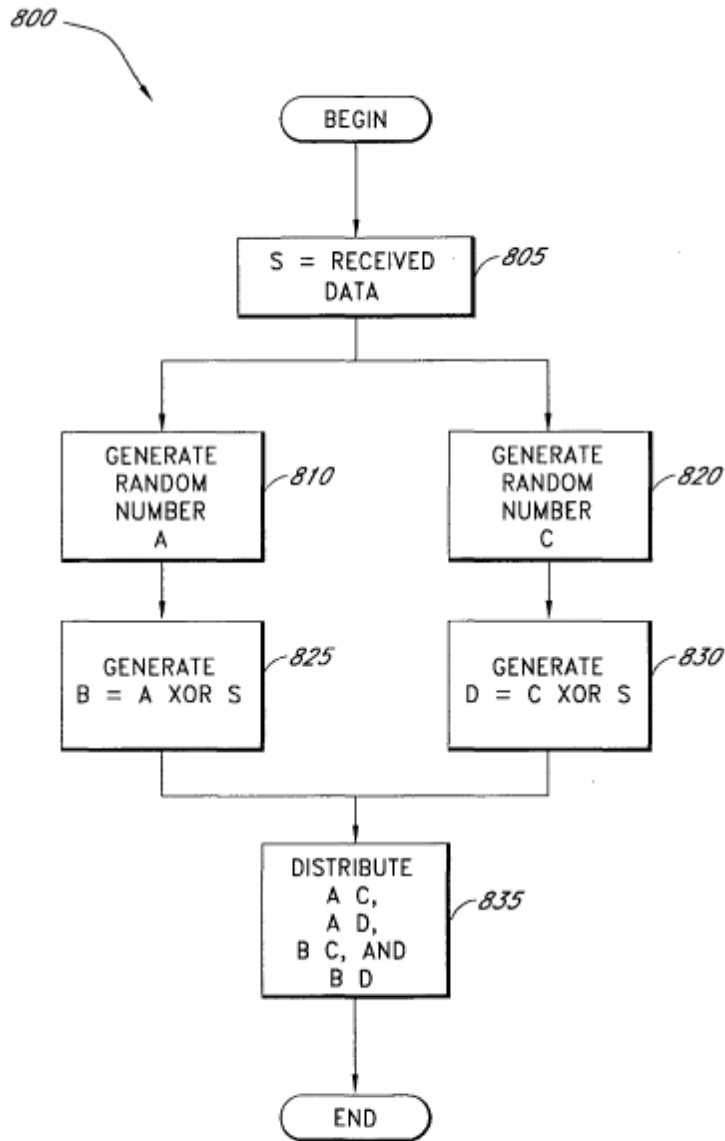


FIG. 8

EX1003 (Dickinson), FIG. 8.

259. Finally, Dickinson clarifies that “only data from *two* of the multiple data storage facilities, D1 through D4, are *needed* to decipher and reassemble the *sensitive data*.” EX1003 (Dickinson), 20:1–14. In other words, the system involves

four total data shares, and any two of those (*i.e.*, fewer than all four) are sufficient to reconstruct the data set.

260. **In sum**, Dickinson discloses “distributing the data set [Dickinson’s data] in the plurality of shares [Dickinson’s portions] such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of shares [Dickinson’s any two pieces].”

iii. [1B] storing the plurality of shares at a plurality of storage devices;

261. Dickinson discloses claim limitation [1B].

262. For example, Dickinson states that “the data splitting process 800 advantageously *places portions* of the sensitive data in each of the four *data storage facilities* D1 through D4.” EX1003 (Dickinson), 21:10–14.

263. Indeed, Dickinson explains that “the authentication engine 215 and the cryptographic engine 220 may advantageously employ their respective data splitting modules to ... then transmit one or more undecipherable *portions* of the sensitive data *to a particular data storage facility*.” EX1003 (Dickinson), 19:27–34.

264. Dickinson also states that “each *data storage facility, D1 through D4*, comprises a *separate and independent storage system*, such as, for example, a directory server.” EX1003 (Dickinson), 20:1–14.

265. **In sum**, Dickinson discloses “*storing the plurality of shares* [Dickinson’s portions] *at a plurality of storage devices* [Dickinson’s four data storage facilities].”

iv. [1C] receiving, at the electronic computing system, request to retrieve the data set;

266. Dickinson discloses claim limitation [1C].

267. For example, Dickinson states that “the vendor system 120 ... forwards ... the authentication *request* to the *trust engine 110*[.]” EX1003 (Dickinson), 28:6–14. Furthermore, Dickinson explains that the *trust engine 110* includes a transaction engine 205.” EX1003 (Dickinson), 13:5–17. Dickinson also teaches that “transaction engine 205 receives the *[vendor request]* ... and generates a *request* for the user’s enrollment authentication data to be assembled from the *data storage facilities D1 through D4*.” EX1003 (Dickinson), 28:6–14. Dickinson also indicates that “the *sensitive data S* includes enrollment authentication data.” EX1003 (Dickinson), 21:33–22:2.

268. **In sum**, Dickinson discloses “*receiving, at the electronic computing system* [Dickinson’s trust engine], *a request to retrieve the data set* [Dickinson’s vendor request and/or transaction request].”

- v. **[1D] identifying from the plurality of storage devices a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices;**

269. Dickinson discloses claim limitation [1D]. Alternatively, Dickinson in view of Hardjono discloses claim limitation [1D]. A POSITA would have been motivated to combine Dickinson with Hardjono for the reasons discussed in §IX.A.3 above.

270. As an initial matter, a POSITA would have known that retrieving data from some (or any) shares that respond fastest (*e.g.*, have the lowest latency) was not just desired but also well known. *See* §V.D above.

271. **First**, Dickinson discloses the step of “*identifying from the plurality of storage devices a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices.*” Dickinson explains that “the transaction engine 205 may broadcast requests to particular **data storage facilities** based on a wide number of criteria, such as, for example, **response time**, server loads, maintenance schedules, or the like.” EX1003 (Dickinson), 19:24–26.

272. This disclosure in Dickinson mirrors the very same written description relied upon in the ’194 Patent to support this limitation. *See* EX1013 (providing a

highlighted version of the '194 Patent's specification where the highlighted portion appears in Dickinson), 18:27–31. A POSITA would have understood that in order to “broadcast requests to particular data storage facilities,” the transaction engine 205 must first identify which “particular data storage facilities” to target from among the available options. Moreover, a POSITA would have recognized that selecting those facilities “based on ... response time” would involve identifying the “*set of fastest-responding storage devices.*” See also §V.D above.

273. **Second**, alternatively, Dickinson in view of Hardjono teaches claim limitation [1D].

274. To the extent Dickinson alone does not fully disclose the concept of “fastest responding storage devices,” it would have been obvious to combine Dickinson with Hardjono. See §IX.A.3 above.

275. As discussed in §IX.A.2 above, Hardjono teaches “**identify[ing] databases** where **k shares** of the block are stored” including “**based on** determining which of the distributed **databases** are currently, most easily, or **most quickly accessible.**” EX1004 (Hardjono), 7:57–64. In my opinion, a POSITA would understand that “most quickly accessible” refers to the response time of the storage devices.

276. **In sum**, Dickinson discloses “identifying from the plurality of storage devices [Dickinson’s data storage facilities] a set of fastest-responding storage

devices [Dickinson’s particular data storage facilities] necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified [Dickinson’s identification of the particular data storage facilities] based at least in part on the response time [Dickinson’s response time] of the storage devices.”

277. **In sum**, in the alternative, Dickinson and Hardjono disclose “identifying from the plurality of storage devices [Dickinson’s data storage facilities] a set of fastest-responding storage devices [Dickinson’s particular data storage facilities] necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified [Hardjono’s identification of the particular databases] based at least in part on the response time [Hardjono’s most quickly accessible databases] of the storage devices.”

vi. [1E] retrieving from the set of fastest-responding storage devices, the minimum number of shares;

278. Dickinson discloses claim limitation [1E]. Alternatively, Dickinson in view of Hardjono discloses claim limitation [1E]. A POSITA would have been motivated to combine Dickinson with Hardjono for the reasons discussed in §IX.A.3 above.

279. I note that the most obvious reason why one would spend the effort to identify which storage devices respond the fastest is so that one could retrieve data from those fastest-responding storage devices. *See* §V.D.1 above.

280. **First**, as discussed in §IX.A.5.iv above, Dickinson discloses “*receiv[ing]* data *portions* from at least the first *two* of the *data storage facilities D1 through D4*[.]” EX1003 (Dickinson), 21:21–32. Dickinson discloses “the data assembling module 525 may *assemble* the *sensitive data S*, when, for example, it receives data *portions* from *at least* the first *two* of the *data storage facilities D1 through D4* to respond to an assemble request by the trust engine 110.” EX1003 (Dickinson), 21:21–32.

281. Additionally, as explained in §IX.A.5.v above, Dickinson teaches that “the transaction engine 205 may broadcast requests to particular *data storage facilities* based on a wide number of criteria, such as, for example, *response time*, server loads, maintenance schedules, or the like.” EX1003 (Dickinson), 19:24–26; *see also* EX1001 (’194 Patent), 18:27–31 (reciting the same language).

282. **Second**, Dickinson in view of Hardjono teaches claim limitation [1E].

283. To the extent Dickinson alone does not fully disclose the concept of “fastest responding storage devices,” it would have been obvious to combine Dickinson with Hardjono. *See* §IX.A.3 above.

284. As explained above, Dickinson teaches “*receiv[ing]* data *portions* from at least the first *two* of the *data storage facilities D1 through D4*[.]” EX1003 (Dickinson), 21:21–32. Hardjono also teaches “identify[ing] databases where *k* shares of the block are stored,” and explains that this “may be based on determining

which of the *distributed databases* are currently, most easily, or *most quickly accessible*.” EX1004 (Hardjono), 7:57–64. In my opinion, a POSITA would understand “most quickly accessible” to refer to the response time of the storage devices. Hardjono further discloses that “[o]nce the databases where k shares are located are identified, server 110 *retrieves the k shares* for the block of data step 425[.]” EX1004 (Hardjono), 7:65–8:2. Given Dickinson’s disclosure of selecting *data storage facilities* based on factors such as *response time*, it would have been obvious to a POSITA to combine this with Hardjono’s approach of identifying the *most quickly accessible* databases in order to determine the “*fastest-responding storage devices*.”

285. **In sum**, Dickinson discloses “retrieving from the set of fastest-responding storage devices [Dickinson’s particular data storage facilities identified using Dickinson’s response time], the minimum number [Dickinson’s two] of shares.”

286. **In sum**, in the alternative, Dickinson and Hardjono disclose “retrieving from the set of fastest-responding storage devices [Dickinson’s particular data storage facilities identified using Hardjono’s most quickly accessible databases], the minimum number [Dickinson’s two] of shares.”

vii. [1F] reconstructing the data set using the minimum number of *shares*; and

287. Dickinson discloses claim limitation [1F].

288. As discussed in §IX.A.5.iv above, Dickinson describes “*receiv[ing]* data *portions* from at least the first *two* of the *data storage facilities D1 through D4*.” EX1003 (Dickinson), 21:21–32. Following this, Dickinson discloses that “the data assembling module 525 may *assemble* the *sensitive data S*, when, for example, it receives data *portions* from at least the first *two* of the *data storage facilities D1 through D4* to respond to an assemble request by the trust engine 110.” EX1003 (Dickinson), 21:21–32.

289. **In sum**, Dickinson discloses “*reconstructing* [Dickinson’s assembling] *the data set* [Dickinson’s sensitive data] *using the minimum number of shares* [Dickinson’s portions from at least the first two of the data storage facilities D1 through D4].

viii. [1G] sending the data set responsive to the request.

290. Dickinson in view of Hardjono teaches claim limitation [1G]. A POSITA would have found it obvious to combine Dickinson with Hardjono for the reasons discussed in §IX.A.3 above.

291. As noted in §IX.A.2 above, Hardjono states that, “[u]pon subsequent receipt of a *request for the data*,” “[t]he *re-constructed data* is then *returned* to the requester.” EX1004 (Hardjono), 3:40–50; *see also* EX1004 (Hardjono), 7:65–8:2 (“Server 110 then re-creates the block from the *k shares*, step 430, and *returns* the *re-created data block* to the *requester*, step 435.”).

292. **In sum**, Dickinson in view of Hardjono discloses “*sending the data set* [Dickinson’s reconstructed data] *responsive to the request* [Dickinson’s request for the data].”

6. Claim 2

i. [2Pre] The method of claim 1, wherein storing the shares comprises:

293. To the extent the preamble is limiting, Dickinson and Hardjono discloses [2Pre] for the same reasons as discussed in §IX.A.5.i above.

ii. [2A] storing a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center; and

294. Dickinson discloses claim limitation [2A].

295. Dickinson discloses that “the depository system 700” shown in FIG. 7 (below) “comprises multiple data storage facilities, for example, *data storage facilities* D1, D2, D3, and D4,” which “may advantageously comprise some or all of the elements disclosed with reference to the depository 210 of FIG. 4,” and may also “comprise[] multiple *geographically separated independent data storage systems*.” EX1003 (Dickinson), 19:17–20, 20:1–14.

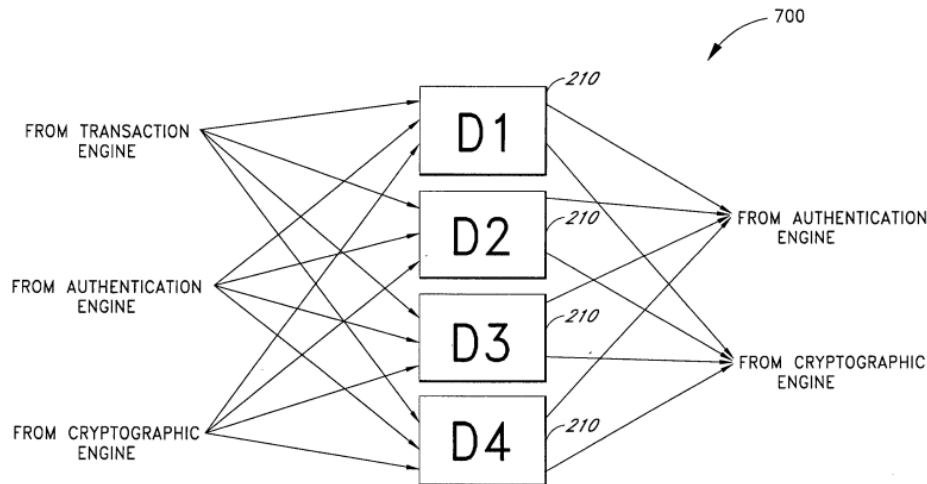


FIG. 7

EX1003 (Dickinson), FIG. 7.

296. As explained in §IX.A.5.iii above, Dickinson discloses that “the data splitting process 800 advantageously places portions of the *sensitive data* in each of the four data *storage facilities* D1 through D4.” EX1003 (Dickinson), 21:10–14. For example, Dickinson discloses that “each of [AC, AD, BC, and BD] is distributed to one of the depositories D1 through D4.” EX1003 (Dickinson), 20:30–21:9.

297. The “*data storage facilities*” in Dickinson comprise, for example, “a directory server, a database server, or the like,” “one or more lightweight directory access protocol (LDAP) servers,” and/or “distinct and physically separated data storage facilities, as disclosed further with reference to FIG. 7.” EX1003 (Dickinson), 14:17–19, 16:25–17:5, 19:16–26.

298. This disclosure in Dickinson corresponds to the same description cited in the '194 Patent as supporting this particular limitation. See EX1013 (providing a

highlighted version of the '194 Patent's specification where the highlighted portion appears verbatim in Dickinson), 13:20-23; 15:28-54, 18:8-31.

299. In my opinion, a POSITA would have understood that each of D1 through D4 may be a geographically separated data center, each containing multiple data storage devices because it was well known that data storage devices are located in data centers. EX1018 (Moulton) at [0036] (“Internetwork 201 enables the interconnection of a heterogeneous set of computing devices and mechanisms ranging from a supercomputer or data center 301 to a hand-held or pen-based device 306.”); Claim 19 (“The method of claim 14 wherein the selected storage nodes comprise at least two storage nodes and the at least two storage nodes are located in *different data centers.*”). See also §V.B.1 above.

300. Dickinson similarly teaches distributed data centers and servers, with storage devices/nodes. EX1003 (Dickinson), 14:17–19 (“FIGURE 2 also illustrates the depository 210. According to one embodiment, the depository 210 comprises one or more data storage facilities, such as, for example, a directory server, a database server, or the like.”), 16:25–17:5 (“FIGURE 4 illustrates a block diagram of the depository 210 of FIGURE 2 according to aspects of an embodiment of the invention. According to this embodiment, the depository 210 comprises one or more lightweight directory access protocol (LDAP) servers. LDAP directory servers are available from a wide variety of manufacturers such as Netscape, ISO, and others.

FIGURE 4 also shows that the directory server preferably stores data 405 corresponding to the cryptographic keys and data 410 corresponding to the enrollment authentication data. According to one embodiment, the depository 210 comprises a single logical memory structure indexing authentication data and cryptographic key data to a unique user ID. The single logical memory structure preferably includes mechanisms to ensure a high degree of trust, or security, in the data stored therein. For example, the physical location of the depository 210 may advantageously include a wide number of conventional security measures, such as limited employee access, modern surveillance systems, and the like. In addition to, or in lieu of, the physical securities, the computer system or server may advantageously include software solutions to protect the stored data. For example, the depository 210 may advantageously create and store data 415 corresponding to an audit trail of actions taken. In addition, the incoming and outgoing communications may advantageously be encrypted with public key encryption coupled with conventional SSL technologies.”), 19:16–26 (“FIGURE 7 illustrates a simplified block diagram of a depository system 700 according to aspects of an embodiment of the invention. As shown in FIGURE 7, the depository system 700 advantageously comprises multiple data storage facilities, for example, data storage facilities D1, D2, D3, and D4. According to one embodiment of the invention, each of the data storage facilities D1 through D4 may advantageously comprise some or

all of the elements disclosed with reference to the depository 210 of FIGURE 4.
...”).

301. Dickinson provides further details of its distributed storage architecture. EX1003 (Dickinson), 20:1–14 (“According to one embodiment, each data storage facility, D1 through D4, comprises a separate and independent storage system, such as, for example, a directory server. According to another embodiment of the invention, the depository system 700 comprises multiple geographically separated independent data storage systems. By distributing the sensitive data into distinct and independent storage facilities D1 through D4, some or all of which may be advantageously geographically separated, the depository system 700 provides redundancy along with additional security measures. For example, according to one embodiment, only data from two of the multiple data storage facilities, D1 through D4, are needed to decipher and reassemble the sensitive data. Thus, as many as two of the four data storage facilities D1 through D4 may be inoperative due to maintenance, system failure, power failure, or the like, without affecting the functionality of the trust engine 110. In addition, because, according to one embodiment, the data stored in each data storage facility is randomized and undecipherable, compromise of any individual data storage facility does not necessarily compromise the sensitive data. Moreover, in the embodiment having geographically [sic] separation of the data storage facilities, a compromise of

multiple geographically remote facilities becomes increasingly difficult. In fact, even a rouge [sic] employee will be greatly challenged to subvert the needed multiple independent geographically remote data storage facilities.”), 20:30–21:9 (“In addition, in STEP 820 the data splitting process 800 generates another statistically random number ‘C.’ According to the preferred embodiment, the generation of the statistically random numbers A and C may advantageously be done in parallel. The data splitting module then combines the numbers A and C with the sensitive data S such that new numbers ‘B’ and ‘D’ are generated. For example, number B may comprise the binary combination of A XOR S and number D may comprise the binary combination of C XOR S. The foregoing combinations preferably occur in STEPS 825 and 830, respectively, and, according to one embodiment, the foregoing combinations also occur in parallel. The data splitting process 800 then proceeds to STEP 835 where the random numbers A and C and the numbers B and D are paired such that none of the pairings contain sufficient data, by themselves, to reorganize and decipher the original sensitive data S. For example, the numbers may be paired as follows: AC, AD, BC, and BD. According to one embodiment, each of the foregoing pairings is distributed to one of the depositories D1 through D4 of FIGURE 7. According to another embodiment, each of the foregoing pairings is randomly distributed to one of the depositories D1 through D4. For example, during a first data splitting process 800, the pairing AC may be sent to depository D2,

through, for example, a random selection of D2's IP address. Then, during a second data splitting process 800, the pairing AC may be sent to depository D4, through, for example, a random selection of D4's IP address.").

302. **In sum**, Dickinson discloses "storing a first subset of the shares [Dickinson's AC] at a first subset of the plurality of storage devices [Dickinson's one or more geographically-separate data facilities D1-D4] that is physically located at a first data center [Dickinson's one or more geographically-separate data facilities D1-D4]."

- iii. **[2B] storing a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.**

303. Dickinson discloses claim limitation [2B].

304. **First**, Dickinson discloses the step of "storing a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center." As detailed in §IX.A.6.ii above, Dickinson teaches that the depository system includes **multiple data storage facilities**—D1, D2, D3, and D4—and that these facilities may comprise geographically separated, independent data storage systems. EX1003 (Dickinson), 19:17–20, 20:1–14, FIG. 7.

305. Additionally, as explained in §IX.A.5.iii above, Dickinson describes distributing the combinations AC, AD, BC, and BD to different depositories among

D1 through D4. EX1003 (Dickinson), 14:17–19, 16:25–17:5, 19:16–26, 20:30–21:9.

306. In my opinion, a POSITA would have understood, for example, that “**BC**” (representing “a second subset of the shares”) could be stored on “one or more lightweight directory access protocol (LDAP) servers,” (serving as “a second subset of the plurality of storage devices”) located in “D3” (representing “a second data center”). See EX1003 (Dickinson), 16:25–17:5, 19:16–26, 20:30–21:9.

307. **Second**, Dickinson discloses that “*the first data center [is] geographically separated from the second data center.*” As discussed earlier in this section, Dickinson teaches “distributing the sensitive data into distinct *and independent storage facilities D1 through D4*, some or all of which may be advantageously *geographically separated.*” EX1003 (Dickinson), 20:1–14. Moreover, a POSITA would have understood that the *independent storage facilities D1 through D4* may be located in separate data centers because that was well known in the art. EX1018 (Moulton) at [0036] (“Internetwork 201 enables the interconnection of a heterogeneous set of computing devices and mechanisms ranging from a supercomputer or data center 301 to a hand-held or pen-based device 306.”); Claim 19 (“The method of claim 14 wherein the selected storage nodes comprise at least two storage nodes and the at least two storage nodes are located in *different data centers.*”). See also §V.B.1 above.

308. **In sum**, Dickinson discloses “storing a second subset of the shares [Dickinson’s BC] at a second subset of the plurality of storage devices [Dickinson’s one or more data storage facilities D1-D4 that do not overlap with the “first subset”] that is physically located at a second data center [Dickinson’s one or more data storage facilities D1-D4 that do not overlap with the “first subset”], the first data center being geographically separated [Dickinson’s geographically separated independent storage facilities] from the second data center.”

7. Claim 3: The method of claim 1, further comprising establishing secure connections between the electronic computing system and the plurality of storage devices.

309. Dickinson teaches the additional limitation of claim [3].

310. In my opinion, Dickinson indicates that “the *data storage facilities D1 through D4* communicate with the transaction engine 205, the authentication engine 215, and the cryptographic engine 220, preferably through conventional *SSL communication* links transferring, for example, XML documents.” EX1003 (Dickinson), 19:16–26.

311. Dickinson further states that “the communications to the authentication engine *comprise secure communications*, such as, for example, *SSL technology*.” EX1003 (Dickinson), 15:13–19. Since these *SSL communications* take place between data storage facilities D1 through D4 and various elements of the *trust engine 110*, it is my view that a POSITA would recognize this as teaching

“*establishing secure connections between the **electronic computing system** and the plurality of storage devices.*”

312. **In sum**, Dickinson discloses “*establishing secure connections [Dickinson’s SSL technology] between the electronic computing system [trust engine] and the plurality of storage devices [Dickinson’s data storage facilities D1 through D4].*”

8. Claim 4: The method of claim 1, wherein the plurality of data blocks share contain a substantially random distribution of the data set.

313. Dickinson discloses the additional limitation of claim [4].

314. As an initial matter, I note that the term “*plurality of data blocks shares*” lacks antecedent basis. For purposes of this IPR only, I understand that Petitioner interprets “*plurality of data blocks shares*” to mean “*plurality of shares.*”

315. Dickinson describes that “the depository 210 may advantageously comprises a *plurality of secure data storage facilities*” which are “configured such that a compromise of the security in one individual *data storage facility* will not compromise ... the authentication data stored therein.” EX1003 (Dickinson), 14:30–15:3.

316. Dickinson also states that “the authentication data are *mathematically operated* on so as to statistically and *substantially randomize* the *data stored in each data storage facility.*” EX1003 (Dickinson), 14:30–15:3. Furthermore, Dickinson

notes that “the *data stored in each data storage facility* is *randomized* and undecipherable.” EX1003 (Dickinson), 20:1–14.

317. In addition, Dickinson explains that “The data splitting module 520 advantageously comprises a software, hardware, or combination module having the ability to *mathematically operate* on various data so as to *substantially randomize* and split the data into *portions*.” EX1003 (Dickinson), 18:20–28. Dickinson also clarifies how the data is handled, stating that “the numbers may be paired as follows: AC, AD, BC, and BD,” and “each of the foregoing pairings is *randomly distributed* to one of the *depositories D1 through D4*.” EX1003 (Dickinson), 21:3–6.

318. **In sum**, Dickinson discloses the “method of claim 1, wherein the plurality of data blocks share [Dickinson’s portions] contain a substantially random distribution of the data set [Dickinson’s substantially randomizes and split the data].”

9. Claim 5: The method of claim 1, wherein the data set can be reconstructed from shares received from at least two storage devices.

319. Dickinson discloses the additional limitation of claim [5].

320. Dickinson discloses “wherein the data set can be reconstructed from shares received from at least two storage devices.” As discussed in §IX.A.5.iv above, Dickinson also describes “receiv[ing] data *portions* from at least the first two of *the data storage facilities* D1 through D4[.]” EX1003 (Dickinson), 21:21–32.

321. **In sum**, Dickinson discloses “*wherein the data set can be reconstructed from shares received [Dickinson’s portions] from at least two storage devices [Dickinson’s first two data storage facilities].*”

10. Claim 6: The method of claim 1, wherein the shares are encrypted with a corresponding number of different keys.

322. Dickinson in view of Hardjono teaches the additional limitation of claim [6]. A POSITA would have been motivated to combine Dickinson with Hardjono for the reasons discussed in §IX.A.3 above.

323. As described in §IX.A.2 above, Hardjono teaches the concept of “encrypting each piece separately using ... different encryption keys.” EX1004 (Hardjono), 1:30–41. Hardjono explains that “[t]his solution provides an additional level of security because possibly multiple keys must be compromised in order to access the entire data.” EX1004 (Hardjono), 1:30–41. In addition, Hardjono states that “[a]ny of a wide variety of conventional encryption processes can be used by the present invention, ... includ[ing] the RSA encryption scheme which employs one or more encryption keys to encrypt the data.” EX1004 (Hardjono), 6:24–33.

324. **In sum**, Dickinson in view of Hardjono discloses “the method of claim 1, wherein the shares are encrypted with a corresponding number of different keys [Dickinson’s one or more keys].”

11. Claim 7

325. Claim 7 only differs from claim 1 for the preamble [7Pre]; and claim limitations [7A], [7B], [7B-1], [7B-3], and [7B-5]. Accordingly, in my opinion, the Dickson/Hardjono System discloses claim limitations [7B-2], [7B-4], and [7B-6] for the same reasons provided in §§IX.A.5 above, particularly [1B] (§IX.A.5.iii above), [1D] (§IX.A.5.v above), [1E] (§IX.A.5.vi above), and [1G] (§IX.A.5.viii above), respectively.

326. In the table below, the limitations in the same row are identical to one another.

Claim 1	Claim 7
[1B]	[7B-2]
[1D] + [1E]	[7B-4]
[1G]	[7B-6]

327. Moreover, the Dickinson/Hardjono System also discloses preamble [7Pre], and claim limitations [7A], [7B], [7B-1], [7B-3], and [7B-5], for the reasons discussed next.

- i. [7Pre] An electronic computing system for securely storing and retrieving data, the electronic computing system comprising:**

328. To the extent the preamble is limiting, Dickinson discloses preamble [7Pre].

329. As outlined in §IX.A.5.i above, Dickinson describes a method for “securely storing and retrieving data.” That section also explains that Dickinson

discloses “an electronic computing system” configured to carry out this method for “securely storing and retrieving data.” EX1003 (Dickinson), 9:7–12 (describing a system that “provide[s] a cryptographic system where one or more *secure* servers, or a trust engine, *stores cryptographic keys and user authentication data.*”), 10:20–32 (“the trust engine 110 comprises one or more *secure* servers for accessing and *storing sensitive information.*”), 38:13 (“assembling the *enrollment data* 410 *retrieved* from the depository 210”).

330. **In sum**, Dickinson discloses “[a]n electronic computing system for securely storing and retrieving data.”

ii. [7A] a processing unit; and

331. Dickinson discloses claim limitation [7A].

332. As discussed in §IX.A.5.v above, Dickinson’s trust engine modules are implemented using software, hardware, or a combination of both. *See, e.g.*, EX1003 (Dickinson), 18:20–21 (“The authentication engine 215 also includes the data splitting module 520 and the data assembling module 525.”), 18:23–24 (“The data assembling module 525 advantageously comprises a software, hardware, or combination module”), 19:3–5 (“The cryptographic engine 220 also comprises a cryptographic handling module 625 configured to perform some or all of a wide number of cryptographic functions. According to one embodiment, the cryptographic handling module 625 may comprise software modules or programs,

hardware, or both.”). A POSITA would have understood that these implementations would involve a processing unit, as the modules are responsible for handling data operations required to perform the storage method. *See also* §V.A above.

333. **In sum**, Dickinson discloses “*a processing unit* [Dickinson’s trust engine].”

iii. [7B] a system memory comprising instructions that, when executed by the processing unit, cause the processing unit to:

334. Dickinson discloses claim limitation [7A]. Dickinson describes that the trust engine includes a “depository 210 [that] comprises a single logical memory structure indexing authentication data and cryptographic key data to a unique user ID.” EX1003 (Dickinson), 16:29–31.

335. Additionally, as noted in §IX.A.5.v above, Dickinson’s trust engine modules are implemented in software, hardware, or a combination of both. *See, e.g.*, EX1003 (Dickinson), 18:20–21 (“The authentication engine 215 also includes the data splitting module 520 and the data assembling module 525.”), 18:23–24 (“The data assembling module 525 advantageously comprises a *software*, hardware, or combination module”), 19:3–5 (“The cryptographic engine 220 also comprises a cryptographic handling module 625 configured to perform some or all of a wide number of cryptographic functions. According to one embodiment, the

cryptographic handling module 625 may comprise software modules or programs, hardware, or both.”).

336. A POSITA would have recognized that such implementations would involve a system memory containing instructions which, when executed by the processing unit, direct that unit to carry out the operations recited in steps [7B-1] through [7B-6] as explained in §§IX.A.11.iv below, IX.A.11.v below, IX.A.11.vi below, IX.A.11.vii below, IX.A.11.viii below, and IX.A.11.ix below, respectively, as these functions involve processing data to perform the storage method. *See also* EX1004 (Hardjono), 8:63-65 (“These software routines comprise a *plurality or series of instructions* to be executed by a processor, such as processor 502 of FIG. 5.”).

337. **In sum**, Dickinson discloses “*a system memory [Dickinson’s single logical memory structure] comprising instructions that, when executed by the processing unit, cause the processing unit to*” perform the method of steps [7B-1] to [7B-6] described next.

- iv. **[7B-1] generate a plurality of shares by performing a cryptographic operation on a data set and distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of the shares and such that the data set cannot be reconstructed using any subset of the shares that includes fewer than the minimum number of the shares;**

338. Dickinson discloses claim limitation [7B-1].

339. Step [7B-1] consists of two components. The first component is identical to claim limitation [1A]: “generate a plurality of shares by performing a **cryptographic operation** on a data set and distributing the data set in **the plurality of shares** such that the data set can be reconstructed using any subset of **the shares** that includes at least a minimum number less than all of the shares.” Therefore, for the same reasons discussed in §IX.A.5.iii above, in connection with claim limitation [1B], Dickinson discloses this first component of claim limitation [7B-1].

340. The second component of claim limitation [7B-1] is also taught by Dickinson: “and such that the data set cannot be reconstructed using any subset of **the shares** that includes fewer than the minimum number of **the shares**.” In my opinion, a POSITA would understand that if reconstruction requires a “minimum number of **shares**,” then, by definition, the data set cannot be reconstructed from any subset of **pieces** containing fewer than that minimum number. *See also* §V.D above.

341. **In sum**, Dickinson discloses instructions that when executed cause the processing unit to “generate [Dickinson’s splitting] a plurality of shares by performing a cryptographic operation [Dickinson’s mathematical operation] on a data set [Dickinson’s data] and distributing the data set [Dickinson’s data] in the plurality of shares [Dickinson’s portions] such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of the shares [Dickson’s any two pieces] and such that the data set cannot be reconstructed using any subset of the shares that includes fewer than the minimum number of the share.”

- v. **[7B-2] store the plurality of shares at a plurality of storage devices;**

342. Dickinson discloses claim limitation [7B-2] for the same reasons discussed in §IX.A.5.iii above, for claim limitation [1B].

- vi. **[7B-3] receive, via the primary interface, a request to retrieve the data set;**

343. Dickinson discloses claim limitation [7B-3].

344. **First**, Dickinson discloses “*receive ... a request to retrieve the data set*” for the reasons discussed in §IX.A.5.iv above for claim limitation [1C]. For example, Dickinson discloses that “the vendor system 120 ... forwards ... the authentication **request** to the **trust engine 110**[.]” EX1003 (Dickinson), 28:6–14. Dickinson also disclosed that “the **trust engine 110** includes a transaction engine

205.” EX1003 (Dickinson), 13:5–17. Dickinson further teaches that “transaction engine 205 receives the [vendor request] ... and generates a request for the user’s enrollment authentication data to be assembled from the data storage facilities D1 through D4.” EX1003 (Dickinson), 28:6–14. Dickinson discloses that “the sensitive data S includes enrollment authentication data.” EX1003 (Dickinson), 21:33–22:2.

345. *Second*, Dickinson discloses receiving the request “via a primary interface” to the same extent the ’194 Patent provides written description for the primary interface. As an initial matter, it is my opinion that the term “primary interface” may lack written description in the ’194 Patent. The only interface specifically described in the ’194 Patent is “any suitable communications interface 2508 (e.g., USB, serial, parallel, Bluetooth, IR, IEEE 1394, ethernet, or any other suitable communications interface) in order to access the original data.” See EX1001 (’194 Patent), 64:51–65:8.

346. If the term “*primary interface*” is understood to correspond to “any suitable communications interface,” Dickinson teaches this feature. Because Dickinson discloses at least the relevant ’194 Patent embodiment, Dickinson satisfies this limitation. See EX1013 (providing a highlighted version of the ’194 Patent’s specification where the highlighted portion appears verbatim in Dickinson).

347. In particular, Dickinson states that “the *data storage facilities D1 through D4* communicate with the transaction engine 205, the authentication engine 215, and the cryptographic engine 220, preferably through conventional SSL *communication links*[.]” EX1003 (Dickinson), 19:16–26. Dickinson also discloses, in Figure 1, that “the cryptographic system 100 includes a user system 105, a *trust engine 110*, a certificate authority 115, and a *vendor system 120*, communicating through a *communication link 125*.” EX1003 (Dickinson), 9:23–24. Therefore, the trust engine 110 receives the request from the vendor system 120 via a communication link 125.

348. **In sum**, Dickinson teaches “*receive, via the primary interface* [Dickinson’s communication link], *a request to retrieve the data set* [Dickinson’s vendor request].”

- vii. **[7B-4] identify, from the plurality of storage devices, a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices, retrieve from the set of fastest-responding storage devices the minimum number of shares;**

349. Dickinson discloses claim limitation [7B-4].

350. Step [7B-4] is composed of two distinct components.

351. The first component—“identify, from the plurality of storage devices, a set of fastest-responding storage devices necessary to retrieve the minimum

number of **shares**, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices”—is the same as claim limitation [1D]. Accordingly, for the same reasons discussed in §IX.A.5.v above, with respect to claim limitation [1D], Dickinson discloses this portion of claim limitation [7B-4].

352. The second component—“*retrieve from the set of fastest-responding storage devices the minimum number of shares*”—is identical to claim limitation [1E]. Therefore, for the same reasons outlined in in §IX.A.5.vi above, in connection with claim limitation [1E], Dickinson also discloses this part of claim limitation [7B-4].

353. **In sum**, Dickinson discloses “identify[ing] from the plurality of storage devices [Dickinson’s data storage facilities] a set of fastest-responding storage devices [Dickinson’s particular data storage facilities] necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified [Dickinson’s identification of the particular data storage facilities] based at least in part on the response time [Dickinson’s response time] of the storage devices.”

viii. [7B-5] reconstruct the data set using exclusively the minimum number of shares; and

354. Dickinson discloses claim limitation [7B-5].

355. As detailed in §IX.A.5.iv above, Dickinson discloses “*receiv[ing]* data *portions* from *at least* the first *two* of the *data storage facilities D1 through D4*[.]” EX1003 (Dickinson), 21:21–32. Following that, Dickinson states that “the data assembling module 525 may *assemble* the *sensitive data S*, when, for example, it receives data *portions* from *at least* the first *two* of the *data storage facilities D1 through D4* to respond to an assemble request by the trust engine 110.” EX1003 (Dickinson), 21:21–32. Because Dickinson teaches “*assembl[ing]* the *sensitive data S* from ... *at least* the first *two* of the *data storage facilities D1 through D4*,” it discloses reconstructing the data using *only two* of the data *portions*.

356. **In sum**, Dickinson teaches “*reconstructing* [Dickinson’s assembling] *the data set* [Dickinson’s sensitive data] *using exclusively the minimum number of shares* [Dickinson’s portions from the first two of the data storage facilities D1 through D4].”

ix. [7B-6] send the data set responsive to the request.

357. Dickinson/Hardjono teaches claim limitation [7B-6] for the same reasons discussed in §IX.A.5.viii above, for claim limitation [1G].

12. Claim 8: The electronic computing system of claim 7, wherein the instructions cause the processing unit to store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.

i. [8Pre]: The electronic computing system of claim 7, wherein the instructions cause the processing unit to

358. Dickinson discloses “[t]he electronic computing system of claim 7” for the same reasons discussed above for claim 7, as described in §IX.A.11 above (Ground I, Claim 7).

359. Dickinson also discloses that “the instructions cause the processing unit to” perform the steps recited in [8A]-[8B]. Namely, Dickinson’s trust engine modules are implemented using software, hardware, or a combination of both. *See, e.g.,* EX1003 (Dickinson), 18:20–21 (“The authentication engine 215 also includes the data splitting module 520 and the data assembling module 525.”), 18:23–24 (“The data assembling module 525 advantageously comprises a software, hardware, or combination module ...”), 19:3–5 (“The cryptographic engine 220 also comprises a cryptographic handling module 625 configured to perform some or all of a wide number of cryptographic functions. According to one embodiment, the cryptographic handling module 625 may comprise software modules or programs, hardware, or both.”). Using instructions to implement software was well known in

the art. *See* EX1004 (Hardjono), 8:63–65 (“These software routines comprise a *plurality or series of instructions* to be executed by a processor, such as processor 502 of FIG. 5.”).

360. In my opinion, a POSITA would have understood that such implementations involve a system memory containing instructions which, when executed by the processing unit, cause it to carry out the first two components of claim [8] (below), as these steps involve data processing required to implement the storage method.

- ii. **[8A]: to store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and**

361. Dickinson discloses claim limitation [8A] for the same reasons discussed in §IX.A.6.ii above (Ground I, 2A).

- iii. **[8B]: to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.**

362. Dickinson discloses claim limitation [8B] for the same reasons discussed in §IX.A.6.iii above (Ground I, 2B).

13. Claim 9

- i. [9] The electronic computing system of claim 7, wherein the instructions further cause the processing unit to generate the plurality of shares in response to receiving a request to store the data set.**

363. Dickinson discloses the additional limitation of claim [9].

364. **First**, Dickinson teaches the step of “*receiving a request to store the data set.*” Specifically, Dickinson explains that “the enrollment process 900 begins at STEP 905 when a user *desires to enroll* with the trust engine 110 of the cryptographic system 100.” EX1003 (Dickinson), 22:7–17. As part of this process, Dickinson discloses “*transmit[ing] enrollment authentication data.*” EX1003 (Dickinson), FIG. 9A (annotated below).

365. Dickinson also describes “*transmit[ing] the enrollment* data, for example, through and [sic] XML document, to the *trust engine 110*, and in particular, to the transaction engine 205[.]” EX1003 (Dickinson), 22:15–17. Further, Dickinson states that “[i]n STEP 925, the authentication engine 215 *forwards* each *portion* of the randomized numbers to one of the data storage facilities D1 through D4.” EX1003 (Dickinson), 22:7–17. As noted in §IX.A.5.iii above, these *portions* are *stored* in the *data storage facilities D1 through D4*. EX1003 (Dickinson), 21:10–14.

366. In my opinion, a POSITA would understand that this *enrollment process*—which results in the distribution and *storing* of data *pieces* in the *data*

storage facilities D1 through D4—constitutes a “*request to store the data set*,” since transmitting the enrollment data with the intention that it be stored is functionally equivalent to making a request for storage.

367. **Second**, Dickinson discloses the step of “*generat[ing] the plurality of shares in response to receiving a request to store the data set.*” Specifically, Dickinson states that “[i]n STEP 920,” “the authentication engine 215 employs the data splitting module to *mathematically operate* on the enrollment authentication data so as to *split* the data into at least *two independently undecipherable, randomized, numbers.*” EX1003 (Dickinson), 23:7–17, FIG. 9A (annotated below). In other words, Dickinson describes that the system proceeds to *store* “each *portion of the randomized numbers*” as part of the enrollment process, without requiring any additional subprocess or explicit command to initiate that storage.

9/21

900

ENROLLMENT DATA FLOW			
SEND	RECEIVE	SSL	ACTION
905 USER	TRANSACTION ENGINE (TE)	1/2	TRANSMIT ENROLLMENT AUTHENTICATION DATA (B) AND THE USER ID (UID) ENCRYPTED WITH THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE) AS (PUB_AE(UID,B))
915 TE	AE	FULL	FORWARD TRANSMISSION
920			AE DECRYPTS AND SPLITS FORWARDED DATA
925 AE	THE X1h DEPOSITORY (DX)	FULL	STORE RESPECTIVE PORTION OF DATA
WHEN DIGITAL CERTIFICATE REQUESTED			
930 AE	CRYPTOGRAPHIC ENGINE (CE)	FULL	REQUEST KEY GENERATION
935			CE GENERATES AND SPLITS KEY
945 CE	TE	FULL	TRANSMIT REQUEST FOR DIGITAL CERTIFICATE
950 TE	CERTIFICATION AUTHORITY (CA)	1/2	TRANSMIT REQUEST
955 CA	TE	1/2	TRANSMIT DIGITAL CERTIFICATE
960 TE	USER	1/2	TRANSMIT DIGITAL CERTIFICATE
TE	MS	FULL	STORE DIGITAL CERTIFICATE
965 TE	DX	FULL	STORE RESPECTIVE PORTION OF KEY

FIG. 9A

EX1003 (Dickinson), FIG. 9A (annotated).

368. **Third**, Dickinson discloses the step “*wherein the instructions further cause the processing unit to generate the plurality of shares.*” As discussed in §IX.A.5.v above, Dickinson’s trust engine modules are implemented using software, hardware, or a combination of both. *See, e.g.*, EX1003 (Dickinson), 18:20–21 (“The authentication engine 215 also includes the data splitting module 520 and the data assembling module 525.”), 18:23–24 (“The data assembling module 525 advantageously comprises a software, hardware, or combination module ...”), 19:3–5 (“The cryptographic engine 220 also comprises a cryptographic handling module 625 configured to perform some or all of a wide number of cryptographic functions. According to one embodiment, the cryptographic handling module 625 may comprise software modules or programs, hardware, or both.”). In my opinion, a POSITA would have understood that these implementations include a system memory containing instructions which, when executed by the processing unit, “*cause the processing unit to generate the plurality of shares.*”

369. **In sum**, Dickinson discloses “*wherein the instructions further cause the processing unit [Dickinson’s trust engine module] to generate the plurality of shares [Dickinson’s splitting the data into pieces] in response to receiving a request to store the data set [Dickinson’s desire to enroll and transmitting of enrollment data].*”

14. Claim 14

370. Claim 14 only varies from claim 7 for the preamble [14Pre] and claim limitation [14B]. The Dickinson/Hardjono System discloses claim limitations [14B]–[14H] for the same reasons provided in §§IX.A.11.iv above–IX.A.11.ix above for claim limitations [7B-1]–[7B-6].

371. In the table below, the limitations in the same row are identical to one another.

Claim 7	Claim 14
[7B-1]	[14B]
[7B-2]	[14C]
[7B-3]	[14D]
[7B-4]	[14E] + [14F]
[7B-5]	[14G]
[7B-6]	[14H]

372. To the extent the preamble is limiting, the Dickinson/Hardjono System also discloses preamble [14Pre] and claim limitation [14A] for the reasons provided next.

- i. [14Pre] A non-transitory computer-readable storage medium comprising instructions that, when executed at an electronic computing device, cause the electronic computing device to:**

373. To the extent the preamble is limiting, Dickinson discloses preamble [14Pre].

374. **First**, as discussed in §IX.A.11.i above, Dickinson discloses an “*electronic computing system*,” which POSITA would have recognized as being equivalent to an “*electronic computing device*.”

375. **Second**, Dickinson discloses a “*non-transitory computer-readable storage medium comprising instructions that ... cause the electronic computing device to*” carry out the steps of claim limitations [14A] through [14H]. Dickinson states that the trust engine includes a “depository 210 [that] comprises a single ***logical memory structure*** indexing authentication data and cryptographic key data to a unique user ID.” EX1003 (Dickinson), 16:29–31.

376. Furthermore, as explained in §IX.A.5.v above, Dickinson’s trust engine modules are implemented using software, hardware, or a combination thereof. *See, e.g.*, EX1003 (Dickinson), 18:20–21 (“The authentication engine 215 also includes the data splitting module 520 and the data assembling module 525.”), 18:23–24 (“The data assembling module 525 advantageously comprises a software, hardware, or combination module”), 19:3–5 (“The cryptographic engine 220 also comprises a cryptographic handling module 625 configured to perform some or all of a wide number of cryptographic functions. According to one embodiment, the cryptographic handling module 625 may comprise software modules or programs, hardware, or both.”).

377. In my opinion, a POSITA would have understood that such implementations include a “non-transitory computer-readable storage medium containing instructions which, when executed, cause the electronic computing device” to perform the steps of claim limitations [14A] through [14H] as explained in §§IX.A.14.ii below, IX.A.14.iii below, IX.A.14.iv below, IX.A.14.v below, IX.A.14.vi below, IX.A.14.vii below, IX.A.14.viii below, and IX.A.14.ix below, as these steps involve data processing to implement the storage method.

378. **In sum**, Dickinson discloses “[a] non-transitory computer-readable storage medium comprising instructions that, when executed at an electronic computing device, cause the electronic computing device to” perform [14-A] to [14-H].

ii. [14A] receive a request to write a data set to a storage location;

379. Dickinson discloses claim limitation [14A].

380. **First**, Dickinson discloses the step of “*receive a request to [store] a data set to a storage location.*” As described in §IX.A.13.i above, Dickinson explains that “the enrollment process 900 begins at STEP 905 when a user *desires to enroll* with the trust engine 110 of the cryptographic system 100.” EX1003 (Dickinson), 22:7–17. As part of this *enrollment process*, Dickinson teaches “*transmit[ting] enrollment authentication data.*” EX1003 (Dickinson), FIG. 9A (annotated below).

381. Dickinson further describes “*transmit[ing] the enrollment* data, for example, through and [sic] XML document, to the *trust engine 110*, and in particular, to the transaction engine 205.” EX1003 (Dickinson), 22:15–16. Additionally, Dickinson states that “[i]n STEP 925, the authentication engine 215 *forwards* each *portion* of the randomized numbers to one of the data storage facilities D1 through D4.” EX1003 (Dickinson), 22:7–17. As discussed in §IX.A.5.iii above, these portions are stored at the data storage facilities D1 through D4. EX1003 (Dickinson), 21:10–14.

382. In my opinion, a POSITA would have understood that this enrollment process—which results in the distribution and *storing* of data *pieces*—constitutes a “*request to [store] a data set to a storage location*” since transmitting the enrollment data for storage effectively serves as a request for that data to be stored.

9/21

900

ENROLLMENT DATA FLOW			
SEND	RECEIVE	SSL	ACTION
905 USER	TRANSACTION ENGINE (TE)	1/2	TRANSMIT ENROLLMENT AUTHENTICATION DATA (B) AND THE USER ID (UID) ENCRYPTED WITH THE PUBLIC KEY OF THE AUTHENTICATION ENGINE (AE) AS (PUB_AE(UID,B))
915 TE	AE	FULL	FORWARD TRANSMISSION
920			AE DECRYPTS AND SPLITS FORWARDED DATA
925 AE	THE X1h DEPOSITORY (DX)	FULL	STORE RESPECTIVE PORTION OF DATA
WHEN DIGITAL CERTIFICATE REQUESTED			
930 AE	CRYPTOGRAPHIC ENGINE (CE)	FULL	REQUEST KEY GENERATION
935			CE GENERATES AND SPLITS KEY
945 CE	TE	FULL	TRANSMIT REQUEST FOR DIGITAL CERTIFICATE
950 TE	CERTIFICATION AUTHORITY (CA)	1/2	TRANSMIT REQUEST
955 CA	TE	1/2	TRANSMIT DIGITAL CERTIFICATE
960 TE	USER	1/2	TRANSMIT DIGITAL CERTIFICATE
TE	MS	FULL	STORE DIGITAL CERTIFICATE
965 TE	DX	FULL	STORE RESPECTIVE PORTION OF KEY

FIG. 9A

EX1003 (Dickinson), FIG. 9A (annotated).

383. **Second**, Dickinson discloses the step of “*receiving a request to write the data set.*” In my opinion, a POSITA would have recognized that Dickinson’s disclosure of “distributing the sensitive data into distinct and independent *storage facilities D1 through D4*,” where the data is subsequently “*stored*,” would involve “*writing*” the data to those storage locations, as recited in the claim. This is because storing a data set inherently requires that the data be written to some form of memory, whether short-term or long-term. *See* §V.A above.

384. **In sum**, Dickinson discloses “*receiv[ing] a request to write a data set [Dickinson’s desire to enroll and transmitting enrollment authentication data] to a storage location [Dickinson’s storing respective portion of data].*”

- iii. **[14B] generate a plurality of shares by performing a cryptographic operation on the data set and distributing the data set in the plurality of shares such that the data set can be reconstructed using any subset of the shares that includes at least a minimum number less than all of the shares and such that the data set cannot be reconstructed using any subset of the shares that includes fewer than the minimum number of the shares;**

385. Dickinson discloses claim limitation [14B] for the same reasons discussed in §IX.A.11.iv above, for claim limitation [7B-1].

- iv. **[14C] store the plurality of shares at a plurality of storage devices;**

386. Dickinson discloses claim limitation [14C] for the same reasons discussed in §IX.A.11.v above, for claim limitation [7B-2].

v. [14D] receive a request to retrieve the data set;

387. Dickinson discloses claim limitation [14D] for the same reasons discussed in §IX.A.11.vi above, for claim limitation [7B-3].

vi. [14E] identify, from the plurality of storage devices, a set of fastest-responding storage devices necessary to retrieve the minimum number of shares, wherein the set of fastest-responding storage devices are identified based at least in part on the response time of the storage devices;

388. Dickinson discloses claim limitation [14E] for the same reasons discussed in §IX.A.11.vii above, for claim limitation [7B-4].

vii. [14F] retrieve from the set of fastest-responding storage devices, the minimum number of shares;

389. Dickinson discloses claim limitation [14F] for the same reasons discussed in §IX.A.11.vii above, for claim limitation [7B-4].

viii. [14G] reconstruct the data set using exclusively the minimum number of shares; and

390. Dickinson discloses claim limitation [14G] for the same reasons discussed in §IX.A.11.viii above, for claim limitation [7B-5].

ix. [14H] send the data set responsive to the request.

391. Dickinson discloses claim limitation [14H] for the same reasons discussed in §IX.A.11.ix above, for claim limitation [7B-6].

- 15. Claim 16: The non-transitory computer-readable storage medium of claim 14, wherein the instructions further cause the processing unit to generate the plurality of shares in response to receiving the request to write the data set to the storage location.**

392. The Dickinson/Hardjono System discloses the additional limitation of claim 16 for the same reasons provided in §IX.A.13.i above (Ground I, Claim 9).

393. In addition, in my opinion, a POSITA would have understood that the phrase “*writ[ing]* a data set to storage” is functionally equivalent to “[*storing*] a data set to storage.” A POSITA would have recognized that “[*storing*] a data set” inherently involves “*writ[ing]* a data set” because *storing* requires committing the data to either short-term or long-term memory. *See* §V.A above.

16. Claims 10–13, 15, and 17–20

394. The Dickinson/Hardjono System satisfies the additional limitation of claims 10 and 17 as provided in §IX.A.7 above (Claim 3);

395. The Dickinson/Hardjono System satisfies the additional limitation of claims 11 and 18 as provided in §IX.A.8 above (Claim 4);

396. The Dickinson/Hardjono System satisfies the additional limitation of claims 12 and 19 as provided in §IX.A.9 above (Claim 5);

397. The Dickinson/Hardjono System satisfies the additional limitation of claims 13 and 20 as provided in §IX.A.10 above (Claim 6); and

398. The Dickinson/Hardjono System satisfies the additional limitation of claim 15 as provided in §IX.A.12 above.

B. Ground II: Dickinson, Hardjono, And Moulton Render Obvious Claims 2, 8, And 15

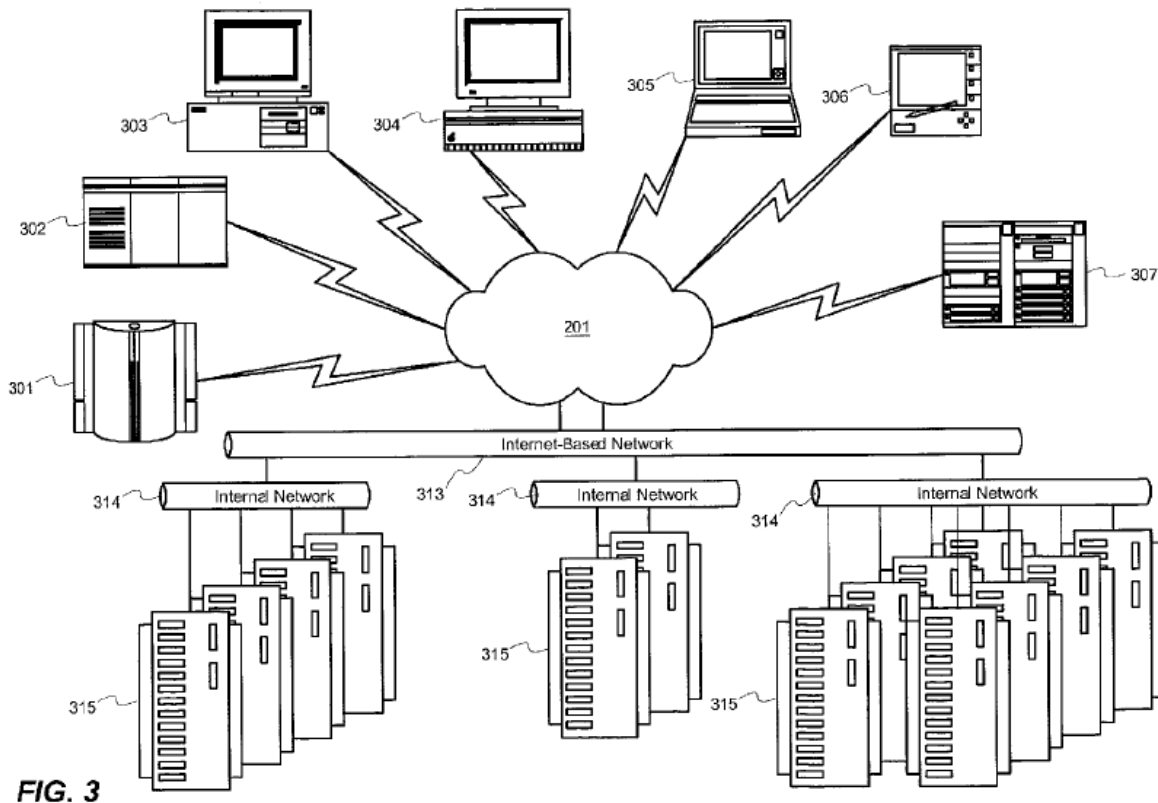
1. Overview of Moulton (EX1018)

399. Moulton is directed to a “distributed network storage.” EX1018 (Moulton) at Title. *See also* EX1018 (Moulton), [0002] (“The present invention relates, in general, to network data storage, and, more particularly, to software, systems and methods for intelligent management of globally distributed network storage.”).

400. Moulton discloses that the distributed storage system includes “storage nodes.” EX1018 (Moulton) at FIG. 3, [0017] (“In a particular implementation, a data storage system is provided that includes a plurality of storage nodes.”). The storage nodes are each located at a physical location with its own “contexts.” EX1018 (Moulton) at [0017] (“In a particular implementation, a data storage system is provided that includes a plurality of storage nodes, where each node exists at a physical location having one or more contexts.”).

401. The storage nodes are connected to each other via a communications network. EX1018 (Moulton) at [0023] (“The system and method of the present invention endeavor to optimize the storage contained in a diverse collection of network-accessible storage nodes.”); [0024] (“The present invention is illustrated

and described in terms of a distributed computing environment such as an enterprise computing system using public communication channels such as the Internet.”). For example, as shown in FIG. 3 (below), “[i]nternetnetwork 201 enables the interconnection of a heterogeneous set of computing devices and mechanisms ranging from a supercomputer or data center 301 to a hand-held or pen-based device 306.” EX1018 (Moulton) at [0036].



EX1018 (Moulton) at FIG. 3.

402. Moulton teaches selecting one or more of the storage nodes to store data. See EX1018 (Moulton) at [0027] (“The present invention enables a mechanism

to strategically select the storage location or locations suitable for a specific task based on the varying characteristics associated with these locations.”); Claim 14 (“selecting one or more of the plurality of storage nodes having an associated context satisfying the desired criteria; and executing the storage task in the one or more selected storage nodes.”).

403. Moulton also teaches that the selected storage nodes for storing the data may be “located in *different data centers*.” EX1018 (Moulton) at Claim 19 (“The method of claim 14 wherein the selected storage nodes comprise at least two storage nodes and the at least two storage nodes are located in different data centers.”).

2. Overview of Dickinson in Combination with Hardjono and Moulton

404. As explained in §IX.A.6 above (Ground I, Claim 2), Dickinson teaches storing the first and second subsets of the shares at two geographically separated data storage facilities, each of which may be separate data centers.

405. If Patent Owner argues that Dickinson’s geographically separate data storage facilities are not data centers, that would have been obvious in view of Moulton. As explained in §IX.B.1 above, a distributed storage system with storage nodes in different data centers was well known in the art. EX1018 (Moulton) at [0036] (“Internetwork 201 enables the interconnection of a heterogeneous set of computing devices and mechanisms ranging from a supercomputer or data center 301 to a hand-held or pen-based device 306.”); Claim 19 (“The method of claim 14

wherein the selected storage nodes comprise at least two storage nodes and the at least two storage nodes are located in *different data centers.*”). *See also* §V.B.1 above.

406. It would have been obvious to augment the Dickinson/Hardjono combination with the teachings of Moulton to provide the geographically separated data storage facilities of Dickinson/Hardjono in “different data centers,” as I explain next. *See* EX1018, [0036] (“Internetwork 201 enables the interconnection of a heterogeneous set of computing devices and mechanisms ranging from a supercomputer or data center 301 to a hand-held or pen-based device 306.”), Claim 19 (“The method of claim 14 wherein the selected storage nodes comprise at least two storage nodes and the at least two storage nodes are located in different data centers.”).

i. Reasons to Combine Dickinson/Hardjono with Moulton

407. A POSITA would have been motivated to combine Dickinson/Hardjono with Moulton for at least the following three reasons.

408. **First**, Moulton teaches a “globally distributed network storage” that can be “readily scaled to a virtually unlimited number of [storage] nodes.” EX1018 (Moulton) at [0002] (“The present invention relates, in general, to network data storage, and, more particularly, to software, systems and methods for intelligent management of globally distributed network storage.”), [0029] (“Because the logical

connection between processes implemented by a network is independent of the physical connection, internetworks are readily scaled to a virtually unlimited number of nodes over long distances.”).

409. Moulton further teaches that some of such storage nodes may be placed in data centers. EX1018 (Moulton) at [0036] (“Internetwork 201 enables the interconnection of a heterogeneous set of computing devices and mechanisms ranging from a supercomputer or data center 301 to a hand-held or pen-based device 306.”); Claim 19 (“The method of claim 14 wherein the selected storage nodes comprise at least two storage nodes and the at least two storage nodes are located in *different data centers*.”). Therefore, to scale the Dickinson/Hardjono’s storage system to a larger capacity, a POSITA would have been motivated to place the data storage facilities in data centers, as taught by Moulton.

410. **Second**, Dickinson teaches that “the depository system 700” shown in FIG. 7 “comprises multiple data storage facilities, for example, *data storage facilities* D1, D2, D3, and D4,” which “may advantageously comprise some or all of the elements disclosed with reference to the depository 210 of FIG. 4,” and may also “comprise[] multiple *geographically separated independent data storage systems*.” EX1003 (Dickinson), 19:17–20, 20:1–14.

411. Moreover, Moulton specifically teaches that such geographically separated data storage systems may be implemented using “different data centers.”

EX1018 (Moulton) at Claim 19. Accordingly, a POSITA would have been motivated to implement Dickinson's "geographically separated independent data storage systems" using "different data centers."

412. **Third**, a POSITA would have understood that the combination simply involves applying a known technique—Moulton's teaching of using storage nodes in "different data centers"—to a known system, namely Dickinson/Hardjono's storage architecture, that is ready for improvement in the capacity and geographical scalability. The application of Moulton's teachings to the Dickinson/Hardjono system would have predictably improved the scalability of the Dickinson/Hardjono system.

413. Next, a POSITA would have had a reasonable expectation of successfully implementing Moulton's teachings in Dickinson/Hardjono's storage system for at least the following two reasons.

414. **First**, both the Dickinson/Hardjono system and Moulton are directed to a networked distributed storage system. *See* EX1003 (Dickinson), 20:2-6 ("According to another embodiment of the invention, the depository system 700 comprises multiple geographically separated independent data storage systems. By distributing the sensitive data into distinct and independent storage facilities D1 through D4, some or all of which may be advantageously geographically separated, the depository system 700 provides redundancy along with additional security

measures”), 19:23-24 (“Communications from the transaction engine 205 may advantageously include requests for data, wherein the request is advantageously broadcast to the IP address of each data storage facility D1 through D4.”)

415. Hardjono similarly teaches networked distributed storage (database) systems. EX1004 (Hardjono), 1:8-9 (“More particularly, this invention relates to secure data storage using distributed databases.”), 3:58-67 (“The couplings 135 between client systems 120 and server 110 as well as the couplings 145 between server 110 and databases 131-133 can be any of a wide variety of conventional communication ‘channels’ over which data transfer can occur. Examples of such couplings 135 and 145 include, but are not limited to, networks (such as the Internet or a local area network (LAN)), ‘direct’ connections (such as a dial-in connection using a modem over a dedicated phone line), ‘manual’ connections (such as transporting magnetic or optical disks), etc.”).

416. Moulton also similarly teaches networked distributed storage systems. EX1018 (Moulton), [0002] (“The present invention relates, in general, to network data storage, and, more particularly, to software, systems and methods for intelligent management of globally distributed network storage.”), [0036] (“Internetwork 201 enables the interconnection of a heterogeneous set of computing devices and mechanisms ranging from a supercomputer or data center 301 to a hand-held or pen-based device 306.”).

417. A POSITA would have thus understood that storage nodes in data centers—as taught by Moulton—would have been readily accessible by the computing devices in the Dickinson/Hardjono system—and they would have operated just like other storage devices accessible via a communications network, *i.e.*, to store and provide data shares generated by Dickinson/Hardjono’s system. Therefore, a POSITA would have reasonably expected that the combination yields the predictable result of providing and operating storage devices in data centers.

418. **Second**, the combination would have merely amounted to a simple substitution of a data storage facility in Dickinson/Hardjono with a data storage node in a data center, as provided in Moulton, to yield the predictable result of providing a distributed storage system that provides data storage nodes in geographically separated data centers accessible via a communications network. *See* EX1018 (Moulton), Claim 19 (“The method of claim 14 wherein the selected storage nodes comprise at least two storage nodes and the at least two storage nodes are located in different data centers.”).

419. Thus, modifying the Dickinson/Hardjono System with Moulton’s teaching would have been simple and would not have required extensive experimentation or imposed undue burden, because both already disclose distributed storage systems.

420. Similarly, the combination would have yielded predictable results (because providing storage facilities in data centers that are accessible via a communications network was well known).

3. Claim 2

i. [2Pre] The method of claim 1, wherein storing the shares comprises:

421. To the extent the preamble is limiting, the Dickinson/Hardjono/Moulton combination teaches preamble [2Pre] for the same reasons as discussed in §IX.A.5 above (Ground I, Claim 1).

ii. [2A] storing a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center; and

422. The Dickinson/Hardjono/Moulton combination teaches “*storing a first subset of the shares at a first subset of the plurality of storage devices*” for the same reasons provided in §IX.A.6.ii above (Ground I, 2A).

423. Furthermore, the Dickinson/Hardjono/Moulton combination teaches that the “*first subset of the plurality of storage devices*” “*is physically located at a first data center*” because Moulton discloses that storage nodes may be physically located at “different data centers.” EX1018 (Moulton) at [0036] (“Internetwork 201 enables the interconnection of a heterogeneous set of computing devices and mechanisms ranging from a supercomputer or *data center* 301 to a hand-held or pen-based device 306.”); Claim 14 (“selecting one or more of the plurality of storage

nodes having an associated context satisfying the desired criteria; and executing the storage task in the one or more selected storage nodes.”); Claim 19 (“The method of claim 14 wherein the selected storage nodes comprise at least two storage nodes and the at least two storage nodes are *located in different data centers.*”).

424. **In sum**, the Dickinson/Hardjono/Moulton combination “*storing a first subset of the shares [Dickinson’s AC] at a first subset of the plurality of storage devices [Dickinson’s one or more geographically-separate data facilities D1-D4] that is physically located at a first data center [one of Moulton’s data centers].*”

- iii. **[2B] storing a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.**

425. The Dickinson/Hardjono/Moulton combination teaches “*storing a second subset of the shares at a second subset of the plurality of storage devices*” for the same reasons provided in §IX.A.6.iii above (Ground I, 2B).

426. Furthermore, the Dickinson/Hardjono/Moulton combination teaches that the “*second subset of the plurality of storage devices*” “*is physically located at a second data center*” because Moulton discloses that storage nodes are physically located at “different data centers.” *See* EX1018 (Moulton) at [0036] (“Internetwork 201 enables the interconnection of a heterogeneous set of computing devices and mechanisms ranging from a supercomputer or *data center* 301 to a hand-held or pen-

based device 306.”); Claim 14 (“selecting one or more of the plurality of storage nodes having an associated context satisfying the desired criteria; and executing the storage task in the one or more selected storage nodes.”); Claim 19 (“The method of claim 14 wherein the selected storage nodes comprise at least two storage nodes and the at least two storage nodes are *located in different data centers.*”).

427. The Dickinson/Hardjono/Moulton combination also teaches that the “*the first data center being geographically separated from the second data center.*” For example, Dickinson teaches that the data storage facilities are geographically separate from one another. *See* EX1003 (Dickinson), 19:17–20 (“As shown in FIGURE 7, the depository system 700 advantageously comprises multiple data storage facilities, for example, data storage facilities D1, D2, D3, and D4. According to one embodiment of the invention, each of the data storage facilities D1 through D4 may advantageously comprise some or all of the elements disclosed with reference to the depository 210 of FIGURE 4.”), 20:1–14 (“According to one embodiment, each data storage facility, D1 through D4, comprises a separate and independent storage system, such as, for example, a directory server. According to another embodiment of the invention, the depository system 700 comprises multiple geographically separated independent data storage systems. By distributing the sensitive data into distinct and independent storage facilities D1 through D4, some or all of which may be advantageously geographically separated, the depository

system 700 provides redundancy along with additional security measures. For example, according to one embodiment, only data from two of the multiple data storage facilities, D1 through D4, are needed to decipher and reassemble the sensitive data. Thus, as many as two of the four data storage facilities D1 through D4 may be inoperative due to maintenance, system failure, power failure, or the like, without affecting the functionality of the trust engine 110. In addition, because, according to one embodiment, the data stored in each data storage facility is randomized and undecipherable, compromise of any individual data storage facility does not necessarily compromise the sensitive data. Moreover, in the embodiment having geographically [sic] separation of the data storage facilities, a compromise of multiple geographically remote facilities becomes increasingly difficult. In fact, even a rouge [sic] employee will be greatly challenged to subvert the needed multiple independent geographically remote data storage facilities.”).

428. Furthermore, Moulton teaches that those geographically separate data storage facilities are data centers. See EX1018 (Moulton) at [0036] (“Internetwork 201 enables the interconnection of a heterogeneous set of computing devices and mechanisms ranging from a supercomputer or data center 301 to a hand-held or pen-based device 306.”); Claim 19 (“The method of claim 14 wherein the selected storage nodes comprise at least two storage nodes and the at least two storage nodes are located in *different data centers.*”).

429. **In sum**, the Dickinson/Hardjono/Moulton combination discloses “*storing a second subset of the shares [Dickinson’s BC] at a second subset of the plurality of storage devices [Dickinson’s one or more data storage facilities D1-D4 that do not overlap with the “first subset”] that is physically located at a second data center [one of Moulton’s data centers], the first data center [one of Moulton’s data centers] being geographically separated from the second data center [another one of Moulton’s data centers].*”

4. **Claim 8: The electronic computing system of claim 7, wherein the instructions cause the processing unit to store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.**

- i. **[8Pre]: The electronic computing system of claim 7, wherein the instructions cause the processing unit to**

430. Dickinson discloses “[*t*]he electronic computing system of claim 7” for the same reasons provided above for claim 7, as described in §IX.A.11 above (Ground I, Claim 7).

431. Dickinson also discloses that “*the instructions cause the processing unit to*” perform the steps recited in claim limitations [8A]-[8B]. Namely, Dickinson’s trust engine modules are implemented using software, hardware, or a combination of both. *See, e.g.*, EX1003 (Dickinson), 18:20–21 (“The authentication

engine 215 also includes the data splitting module 520 and the data assembling module 525.”), 18:23–24 (“The data assembling module 525 advantageously comprises a *software*, hardware, or combination module”), 19:3–5 (“The cryptographic engine 220 also comprises a cryptographic handling module 625 configured to perform some or all of a wide number of cryptographic functions. According to one embodiment, the cryptographic handling module 625 may comprise software modules or programs, hardware, or both.”).

432. Using instructions to implement the software was well known. *See also* EX1004 (Hardjono), 8:63-65 (“These software routines comprise a *plurality or series of instructions* to be executed by a processor, such as processor 502 of FIG. 5.”).

433. In my opinion, a POSITA would have understood that such implementations involve a system memory containing instructions which, when executed by the processing unit, cause it to carry out the first two components of claim [8] (below), as these steps involve data processing required to implement the storage method.

- ii. **[8A]: to store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and**

434. The Dickinson/Hardjono/Moulton combination teaches claim limitation [8A] for the same reasons discussed in §IX.A.6.ii above (Ground II, 2A).

- iii. **[8B]: to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.**

435. The Dickinson/Hardjono/Moulton combination teaches claim limitation [8A] for the same reasons discussed in §IX.A.6.iii above (Ground II, 2B).

- 5. **Claim 15: The non-transitory computer-readable storage medium of claim 14, wherein the instructions cause the processing unit to store a first subset of the shares at a first subset of the plurality of storage devices that is physically located at a first data center and to store a second subset of the shares at a second subset of the plurality of storage devices that is physically located at a second data center, the first data center being geographically separated from the second data center.**

436. The Dickinson/Hardjono/Moulton combination teaches the additional limitation of claim 15 for the same reasons discussed in §IX.A.12 above (Ground II, Claim 8).

X. Conclusion

437. In my opinion, the challenged claims 1–20 of the '194 Patent are unpatentable.

XI. Availability for Cross-Examination

438. In signing this declaration, I recognize that the declaration will be filed as evidence in a case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross

examination in the case and that cross examination will take place within the United States. If cross examination is required of me, I will appear for cross examination within the United States during the time allotted for cross examination.

XII. Right to Supplement

439. I reserve the right to supplement my opinions in the future to respond to any arguments that the Patent Owner may raise and to take into account new information as it becomes available to me.

XIII. Jurat

440. I 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 §1101 of Title 18 of the United States Code.



July 8, 2025

Dr. Erez Zadok